TENRAI
TECHNOLOGIES

TSP-2: Zero-Knowledge Confidential Payments

A bounded-state payment protocol for TRAIDAG
with current-state wallet recovery and deterministic settlement

Tenrai Technologies

tenrai-technologies.org

Technical Whitepaper · September 2026

ABSTRACT

Confidential payment systems must retain enough information to prevent repeated spending and to recover unspent funds. These obligations become difficult to reconcile with pruning when recovery depends on transaction history or the current state records every historical spend. TSP-2 addresses this problem with a finite, reusable collection of live value cells. Each confidential cell contains a value commitment and the complete encrypted information needed by its owner. Consumption replaces the cell incarnation; a generation counter prevents replay without an append-only nullifier set.

The protocol supports optional confidential payments of Tenrai's native asset. Zero-knowledge arguments establish spending authority, integer value conservation and consistency between each output commitment and its recovery capsule. Public input references are retained deliberately: amounts are confidential subject to public-flow leakage, but the transaction graph is observable. A native public exit reuses consumed storage, so an exhausted allocation budget does not by itself immobilise an existing confidential balance.

We define the state transition and prove conditional results for one-use consumption, value preservation, deterministic execution, current-state recovery and bounded storage. A separate, resource-limited proof inbox shares TRAIDAG's certified cuts without placing heavy proof obligations in work-block bodies. An analytical 100-block-per-second profile separates work production, proof throughput and communication cost. The construction makes recovery a property of current replicated state rather than of an optional historical archive.

Document scope

This standalone companion whitepaper presents the TSP-2 zero-knowledge confidential-payment protocol as an extension of TRAIDAG v1.2.3. It preserves stake-certified ordering and native-asset accounting while adding reusable value cells, a zero-knowledge payment relation and an independently budgeted proof inbox. Use of confidentiality is optional; validation of activated rules is common to pruned, full-history and archive nodes.

The mathematical results concern the specified transition system under the assumptions stated in section 2. The cryptographic relation is parameterised by a fixed proof profile; its STARK-based instantiation is not treated as an already audited implementation. Numerical exhibits are encoding arithmetic, capacity models and executable state-model results, not distributed throughput measurements. This distinction separates the protocol's invariants from the engineering performance of a particular client.

The term *current-state recovery* means recovery of all presently unspent outputs belonging to a supported seed-derived account. It does not require retention of spent-payment history, and it does not imply transaction-graph anonymity. The current-state and privacy definitions are given independently so that neither claim depends on the other.

Protocol basis

The base specification is *TRAIDAG: Certified Ordering over a Proof-of-Work Block Graph*, Technical Whitepaper v1.2.3, Tenrai Technologies, September 2026. References of the form *TRAIDAG, Section 4.4* identify that protocol specification, not independent evidence for TSP-2. The payment construction, state rules and analytical results are developed in this paper. External references establish the named primitives and comparison constructions; they do not establish this protocol's end-to-end security.

Notation and units

A cut height is a finalised protocol counter, not a timestamp. Monetary values are nonnegative integers in native base units. Concatenation is denoted by $\|$; H_D denotes a domain-separated SHA-384 construction with unambiguous field encoding [1]. One MiB is 2^{20} bytes and one GiB is 2^{30} bytes. Decimal GB denotes 10^9 bytes. A model day contains 86,400 seconds.

Authors

Qiro, Voss and Neryn — Tenrai Technologies
Cryptis and ZeroBytes — Cryptix Network
Shard — Rift Project

Author names are pseudonyms. The document describes a protocol and its deployment requirements; it does not constitute an independent implementation audit.

Contents

1 Introduction **1**
1.1 Contributions and scope . . . 1

2 System model and assumptions **2**
2.1 Certified execution and node roles . . . 2
2.2 State and notation . . . 2
2.3 Cryptographic and operational premises . . . 2

3 A finite arena of live value **3**
3.1 Records and incarnations . . . 3
3.2 Deterministic replacement . . . 3
3.3 Finite counters and saturation . . . 4

4 Cryptographic representation and recovery capsules **4**
4.1 Seed-derived ownership and receiver binding . . . 4
4.2 Note plaintext and commitment . . . 5
4.3 Complete encrypted output data . . . 5
4.4 The payment relation . . . 5

5 Payment classes and native-value accounting **6**
5.1 Five bounded transitions . . . 6
5.2 Disjoint aggregates . . . 7
5.3 Fees and a funded exit . . . 7
5.4 Direct-inbox fee attribution . . . 8

6 Deterministic admission and settlement **8**
6.1 Anchors, transaction identities and applicability . . . 8
6.2 Two queues within one cut . . . 9
6.3 Fixed phase order and atomic application . . . 9
6.4 Deterministic failures and data retention . . . 10
6.5 Integer semantics . . . 10

7 Safety analysis **10**
7.1 One-use consumption and authorisation . . . 10
7.2 Value conservation . . . 11
7.3 Capacity-safe terminal exit . . . 12
7.4 State equality . . . 12

8 Privacy and public information **13**
8.1 Confidential values with an observable graph . . . 13
8.2 Recipient privacy and cryptographic composition . . . 13
8.3 Privacy and recoverability are distinct . . . 14

9 Current-state recovery and pruning **14**
9.1 Authenticated complete records . . . 14
9.2 Recovery from a seed . . . 14
9.3 Catch-up without a permanent event log . . . 15
9.4 Retention, snapshots and serving . . . 15

10 Bounded storage and finite capacity **16**
10.1 A storage bound independent of transaction count . . . 16
10.2 Reference-size arithmetic . . . 17
10.3 The cost of a finite arena . . . 17

11 Resource isolation and the 100-BPS profile **18**
11.1 Logical work and independent service . . . 18
11.2 An analytical resource profile . . . 18
11.3 Rates, payloads and bandwidth . . . 19
11.4 Processing and capacity conditions . . . 19

12 Operational security and protocol continuity **20**
12.1 Bounded work under adversarial input . . . 20
12.2 Durability and state reclamation . . . 20
12.3 Old outputs and upgrades . . . 20
12.4 Faults beyond the state argument . . . 21

13 Related work **21**

14 Conclusion **22**

A Canonical representation **23**
A.1 Fixed cell layout . . . 23
A.2 Transaction and witness envelope . . . 23
A.3 Acyclic identity construction . . . 24
A.4 Key derivation and signing contexts . . . 24
A.5 Domain families . . . 24

B Transition and recovery algorithms **25**
B.1 Current-state application . . . 25
B.2 Complete recovery . . . 26

C Reproducible analytical exhibits **27**
C.1 State-model evidence . . . 27
C.2 Reproduction and numerical scope . . . 27

1 Introduction

Pruning is useful only if the information removed is no longer needed. For public payments, a current unspent-output set separates spendable value from the signatures and transactions that created it. Confidential payments require an additional distinction: a validator's ability to verify a spend is not the same as an owner's ability to discover and reconstruct the underlying coin. A compact root may serve the former purpose while being insufficient for the latter.

A recovery design must therefore answer a concrete question. A user returns after a long absence with the correct seed but no wallet database. The original transaction bodies have been deleted by ordinary pruned nodes. Which authenticated data still let that user identify, decrypt and spend every remaining output? TSP-2 answers by retaining the encrypted recovery capsule as part of each live output. The capsule has exactly the lifetime of the money it describes.

A second issue is the lifetime of double-spend state. An append-only nullifier set records an additional object for every consumed note. Charging a larger fee changes the cost of producing those objects but does not change their cumulative lifetime. TSP-2 instead exposes the consumed output references and removes the old live records. A bounded generation field distinguishes successive uses of the same physical slot. This is a deliberate privacy choice, not an assertion that all confidential-payment systems must reveal their input graph.

The payment relation uses zero knowledge to separate public verification from private amounts and authorisation material. Each proof binds the complete transaction intent, validates the owners' spending authority and establishes that the recovery ciphertexts encode precisely the committed outputs. The node still checks current input incarnations itself. A valid proof about an old input does not make that input spendable after another transaction consumes it.

1.1 Contributions and scope

The central contribution is a joint state contract for confidentiality, recovery and finite storage. A fixed-capacity arena retains complete current records; consumption removes obsolete incarnations; and an in-arena public exit avoids dependence on newly allocated ordinary UTXOs. The state machine has no archival nullifier or transaction-ID list in its one-use predicate.

The second contribution is the coupling of that contract to TRAIDAG's execution model. Heavy proof envelopes use a separate certified inbox with an independent resource prefix. Both services remain inside the same cut and the same fixed phase order. This removes a particular proof-duplication bottleneck without conflating independent queue selection with independent physical hardware.

The analysis establishes one-use consumption, exact native-value accounting, deterministic state equality, age-independent current-state recovery, bounded retained state and storage-feasible terminal exit. These claims are stated under explicit cryptographic and availability assumptions. Amount confidentiality and recipient-key privacy are treated separately from the publicly observable spend graph.

The first profile supports only the native asset and five bounded payment classes. It does not introduce hidden input selection, a private DEX, confidential contract callbacks or arbitrary uploaded proof programs. Those features would require different statements and state effects rather than reinterpretation of the payment relation defined here.

2 System model and assumptions

2.1 Certified execution and node roles

TRAIDAG separates permissionless work-block production from stake-certified registration and execution (TRAIDAG, Sections 4.1–4.4). A work block fixes its payload order and an immutable finalised anchor. A cut advances the ledger by applying an exact resource-bounded queue prefix. TSP-2 extends that state transition, not the fault threshold or the meaning of finality.

For a fixed eligible stake snapshot with total weight W, the base certificate threshold is $Q = \lfloor 2W/3 \rfloor + 1$. Agreement relies on Byzantine weight below $W/3$ and the base locking, authentication and epoch-transition rules. Progress additionally requires an available correct quorum, complete required data and sufficient processing capacity. The same assumptions apply to a cut containing confidential work. A certificate is not permission to accept an invalid state transition (TRAIDAG, Sections 2.1 and 5.3).

A *fully validating node* verifies the activated rules from its accepted starting state. A pruned node retains current state and all outstanding retention obligations. A full-history node additionally serves complete historical bodies; an archive node additionally retains historical state versions and query structures. These are storage roles, not distinct validity predicates.

2.2 State and notation

At finalised height h, write the extended state as

$$\mathcal{S}_h = (B_h, A_h, Q_h^W, Q_h^C, V_h^C, V_h^P, \Gamma_h), \tag{1}$$

where B_h contains the remaining base state, A_h is the live-value arena, Q_h^W and Q_h^C are the work and confidential queues, and V_h^C, V_h^P are the confidential and public-cell value aggregates. Γ_h contains the finite parameter registry, fee buckets and retained provenance not already assigned to another component. The decomposition is logical: an implementation may share indexes but may not count the same value twice.

Table 1. Principal symbols. Counts refer to committed logical state.

Symbol	Meaning
$K = 2^d$, B_c	Arena capacity and canonical bytes per cell.
L_h	Number of non-FREE cells at height h.
s, g, D	Slot index, generation and digest of an input record.
m, n	Arena cells consumed and created by a transaction.
$v, e, f = v + e$	Spendable amount, owned exit allowance and total face value.
$\Pi, \mathcal{R}$	Activated zero-knowledge argument and payment relation.
$\mathcal{B}_W, \mathcal{B}_C$	Per-service deterministic resource vectors.
R_b, ρ	Model work-block rate and cut cadence.

2.3 Cryptographic and operational premises

The state results use four classes of premises. First, accepted state commitments and transaction contexts are binding, with canonical encodings and collision-resistant domain-separated hashes. Second, the proof system is a knowledge-sound zero-knowledge argument for the exact payment relation; the embedded signature scheme is unforgeable. Knowledge soundness is relevant because authorisation requires possession of a valid witness, not merely the mathematical existence of a signature.

Third, supported seed derivation and the accepted encryption relation satisfy their correctness conditions. Recovery statements allow the correctness-error probability of the chosen cryptographic suite in its stated key-generation and ciphertext-generation model. A rate averaged over honest randomness is not automatically a uniform bound against a sender who searches for exceptional accepted ciphertexts. A capsule hash alone does not satisfy this premise: the full capsule must encode the same note and receiver bound by the creation relation. Fourth, state installation is atomic, current data are available from a valid accepted state, and the node implements the deterministic transition function.

Confidentiality uses additional assumptions: hiding of the randomised note commitment, security of the complete capsule construction, zero knowledge of the argument and, for recipient anonymity, key privacy of that construction. These properties are not implied by one another. In particular, ordinary message confidentiality is distinct from concealing which public key received a ciphertext [2].

The adversary may inspect public records, pay for maximum permitted load, select correlated transactions, exploit network metadata and retain history that other nodes prune. A compromised user key is a compromised authorisation, not a cryptographic forgery. Local data unavailability can prevent progress; it cannot change the accepted balance.

3 A finite arena of live value

3.1 Records and incarnations

The arena contains exactly $K = 2^d$ addressable cells, with $1 \le d \le 31$ in the reference schema. This bound keeps both slot indices and the fully occupied root count representable as u32 values. Every cell is one of FREE, CONFIDENTIAL or PUBLIC. A confidential cell holds a note commitment and its encrypted recovery capsule. A public cell holds a native amount and an ordinary spend lock. A free cell retains its generation and last-mutation height, together with the canonical structural header; its value payload is zero.

Every cell has a fixed-width generation g, a last-mutation height t, and a birth height for its current live output. A spending reference binds at least

$$\mathrm{ref}(c) = (s, g, \mathrm{noteid}(c), \mathrm{cm}(c), D(c)). \tag{2}$$

The expected digest commits the complete record, including its capsule where present. Thus a reference identifies one incarnation, not merely a reusable location. The transaction's semantic output identity is independent of the slot chosen later by execution.

3.2 Deterministic replacement

If a transaction consumes m arena cells and creates n, its occupancy condition is

$$L' = L - m + n \le K, \qquad L = N_{\mathrm{conf}} + N_{\mathrm{public}}. \tag{3}$$

Here m counts distinct, currently occupied input cells and n counts new arena outputs. Duplicate source references are rejected before this calculation. Ordinary UTXOs outside the arena do not contribute to m. Entering the confidential system from ordinary UTXOs can require new capacity; replacing two existing cells with two new cells does not.

Execution first reserves every input incarnation. It clears consumed records, increments their generations, and assigns outputs to consumed slots in ascending slot order. Additional outputs use the lowest remaining FREE slots. Output descriptors retain their canonical transaction order. A subtree free-count index permits selection in bounded tree work rather than a scan of all K cells.

A reused slot inherits its advanced generation. A previously free slot retains its existing generation when first allocated, since no live incarnation is consumed at that allocation. Any later consumption advances it. This rule is sufficient because a transaction may reference only a live record, never a FREE header.

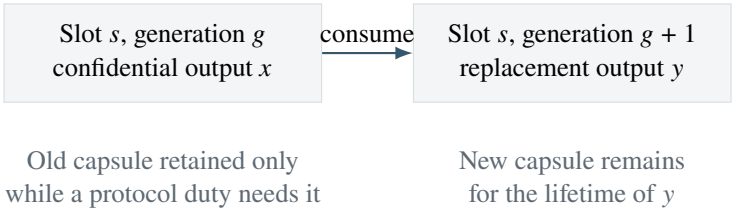


Figure 1. Slot reuse changes the incarnation. The current record and fixed-width generation replace an append-only historical spend marker.

3.3 Finite counters and saturation

Generations are unsigned 128-bit integers with no wrap. With zero initial generations, u64 cut heights and at most $2^{32} - 1$ slot-consumption events per cut, every reachable generation satisfies

$$g \leq (2^{64} - 1)(2^{32} - 1) < 2^{96} < 2^{128}. \tag{4}$$

This is a bound over the specified protocol domain. Snapshot import and migration preserve it rather than accepting arbitrary near-overflow counters. The analytical resource profile in section 11 imposes much smaller per-cut mutation limits.

No cell is reclaimed because of age, inactivity or an inference that its key has been lost. At capacity, transactions with positive net growth can be inapplicable. Existing occupancy-preserving replacements and the in-arena terminal exit retain their storage feasibility. A larger arena is a parameter change with an explicit state migration; it is not a local disk-management decision.

4 Cryptographic representation and recovery capsules

4.1 Seed-derived ownership and receiver binding

The initial wallet profile uses one recovery account per effective seed, chain and TSP namespace. The effective seed includes any mnemonic passphrase selected by the owner. Spending and recovery keys use distinct HKDF-SHA384 labels [3]. The derivation depends on fixed public protocol constants, not on a locally stored account counter or the version of a later proof backend.

The candidate primitive profile derives seeds for ML-DSA-65 spending keys and ML-KEM-768 recovery keys [4, 5]. Let pk_s, sk_s denote the spending key pair and ek, dk the encapsulation and decapsulation keys. Define

$$S = \mathrm{H}_{\mathrm{SPK}}(pk_s), \qquad E = \mathrm{H}_{\mathrm{EPK}}(ek). \tag{5}$$

A receiver descriptor binds the chain, namespace, note and encryption suites, pk_s, ek and a spending-key signature over their complete context. The descriptor is exchanged off chain and appears as private witness material in output creation. Its signature certifies the spending-key holder's approval of the key pairing; it does not establish a real-world identity or prove that an arbitrary descriptor came from a particular wallet seed. The recovery theorem applies to descriptors generated by the supported deterministic wallet profile.

The key-pair binding matters for recovery. A sender must not be able to commit value to one spending key while encrypting the note to an unrelated recovery key and still present the result as a valid output for the original descriptor. Nor does knowledge of the note opening authorise spending: the sender

normally knows the amount and randomness it chose. The spend relation requires a signature under the receiver-controlled spending key.

4.2 Note plaintext and commitment

A confidential note contains a spendable value $v \geq 1$ and an owned exit allowance $e = E_{\text{note}}$. Its face value is $f = v + e$, subject to the native amount bound. The allowance is existing native value, not a newly issued voucher. The reference note encoding is

$$\begin{aligned} \text{NotePlain} = {} & \text{u16}(\text{suite}) \parallel \text{u128}(v) \parallel \text{u128}(e) \\ & \parallel S \parallel E \parallel O \parallel \text{u8}(i) \parallel r, \end{aligned} \tag{6}$$

where O is a precomputable transaction origin, i is the output index and r is 48 bytes of fresh note randomness. The note occupies 227 bytes before canonical zero padding to a 256-byte encryption plaintext. Its commitment is

$$\text{cm} = \text{H}_{\text{NOTE}}(\text{chain} \parallel \text{namespace} \parallel \text{NotePlain}). \tag{7}$$

The binding and hiding roles of this construction are separate assumptions. Hash output length alone is not a proof of hiding under arbitrary randomness. Honest wallets use independent cryptographic randomness for note creation and encapsulation; consensus cannot certify that a malicious creator kept its randomness secret.

4.3 Complete encrypted output data

The candidate E2 capsule combines ML-KEM-768, HKDF-SHA384 and AES-256-GCM with a 128-bit authentication tag [5, 3, 6]. Let encapsulation under ek produce (k, c_{kem}). Associated data a binds chain, namespace, suite identifiers, origin, output index and the note commitment. The symmetric key and nonce are independently labelled expansions:

$$\begin{aligned} p &= \text{HKDF-Extract}(\text{salt}, k), \\ k_{\text{aead}} &= \text{Expand}(p, \text{key-label} \parallel \text{H}(a), 32), \\ \nu &= \text{Expand}(p, \text{nonce-label} \parallel \text{H}(a), 12), \\ c_{\text{aead}} &= \text{AES-GCM.Enc}(k_{\text{aead}}, \nu, \text{pad}_{256}(\text{NotePlain}), a). \end{aligned} \tag{8}$$

The salt and all labels are namespace-separated. The full KDF and associated-data byte formats are parameters of the encryption suite. A sender cannot override the nonce derivation. Fresh encapsulation randomness is required by the honest construction.

ML-KEM-768 uses a 1,184-byte encapsulation key and a 1,088-byte KEM ciphertext [5, Table 3]. Including two u16 identifiers and the 16-byte authentication tag gives the capsule size

$$B_{\text{capsule}} = 2 + 2 + 1088 + 256 + 16 = 1364 \text{ bytes}. \tag{9}$$

The complete capsule remains in the live record. A root or capsule digest cannot reconstruct an erased ciphertext and is not a substitute for this retention rule.

4.4 The payment relation

Let x contain the exact transaction core, resolved input contexts and activated relation identifiers. Let w contain private note openings, receiver descriptors, authorised spending signatures and the output-encryption witnesses. For an admissible statement and witness, the proof interface is

$$\pi \leftarrow \text{Prove}_{\Pi}(x, w), \qquad \text{Verify}_{\Pi}(x, \pi) = 1. \tag{10}$$

An honest prover takes a witness satisfying $\mathcal{R}(x, w) = 1$. Completeness concerns acceptance of such proofs; knowledge soundness concerns accepted proofs produced by an adversary. The validity arguments below therefore include the proof system's soundness and extraction errors.

The relation checks the class-specific counts and kinds; the binding of each input opening to its committed record; ownership signatures over the complete intent; bounded integer value equations; and the full output commitment, receiver-certificate, KEM and AEAD computations. Proving a value balance while merely hashing an arbitrary output ciphertext is not this relation.

Public input references make hidden membership proofs unnecessary. The node resolves the referenced records under the admission anchor and rechecks their current incarnations at execution. The proof is therefore independent of changes to unrelated cells and of the final output slots selected by allocation.

The intended argument family is a transparent, hash-based ZK-STARK [7]. A profile fixes its field representation, transcript, masking, commitment primitives, soundness parameters, decoder and deterministic verifier. These are cryptographic parameters of Π, not choices delegated to a transaction sender. The post-quantum condition applies to the complete accepted proof and the payment primitives, including the non-interactive transcript transformation and any outer compression step. In that setting, the soundness, extraction, signature and commitment assumptions are interpreted against the specified quantum adversary, not inherited from a classical-only proof. The candidate E2 composition does not inherit recipient key privacy solely from ML-KEM's message-security claim; section 8 keeps that property explicit.

5 Payment classes and native-value accounting

5.1 Five bounded transitions

TSP-2 represents the native Tenrai asset in three relevant locations: ordinary UTXOs, confidential cells and public cells. The latter two are additional state domains, not additional assets. A public cell is immediately a native-value claim; redemption into the ordinary UTXO map is a storage transition rather than the creation of a redeemable token.

Table 2. Payment classes. At most two sources and two destinations are used, subject to the listed class restrictions.

Class	Sources	Destinations	ZK
ENTER	Ordinary UTXOs and/or PUBLIC cells	One or two CONFIDENTIAL cells; change is confidential	Yes
TRANSFER	One or two CONFIDENTIAL cells	One or two CONFIDENTIAL cells	Yes
EXIT	One or two CONFIDENTIAL cells	Exactly one PUBLIC cell	Yes
PUBLIC	One or two PUBLIC cells	One or two PUBLIC cells	No
REDEEM	One or two PUBLIC cells	One or two ordinary UTXOs	No

PUBLIC and REDEEM use public ML-DSA authorisations and can occur in ordinary work payloads. ENTER, TRANSFER and EXIT use the confidential inbox. Every transaction has one or two distinct sources in total, not two of each source type. All amounts and outputs satisfy their native bounds. No class invokes a user-defined contract or combines source types outside its declared shape. A child output is spendable only after a finalised anchor includes it, preserving the base exclusion of unfinalised parent–child dependencies (TRAIDAG, Section 4.7.1).

For a confidential-to-public partial payment, the initial profile first creates a selected confidential output and, after finality, exits that output. EXIT itself has no simultaneous confidential change. This keeps the

terminal transition small and ensures that its storage argument does not depend on an extra destination allocation.

5.2 Disjoint aggregates

Let V_B be all existing base value domains except the new cell values and the burn total. In particular, V_B includes funded work and stake reserves. Let I_{minted} and $I_{\max}$ retain their base meaning. The extended invariant is

$$V_B + V_C + V_P + V_{\text{burned}} = I_{\text{minted}} \le I_{\max}. \tag{11}$$

The independent base rule for the authorised issuance delta remains in force. No confidentiality operation can increase I_{minted}. Moving value into a cell removes it from its previous domain in the same atomic write set.

Write F for the public transaction fee, P for total public source value, A for total public destination value and f for each confidential note's face value. The required integer relations are

$$\begin{aligned}
\mathsf{ENTER}: &\quad P = \sum f_{\text{out}} + F,\\
\mathsf{TRANSFER}: &\quad \sum f_{\text{in}} = \sum f_{\text{out}} + F,\\
\mathsf{EXIT}: &\quad \sum f_{\text{in}} = A + F,\\
\mathsf{PUBLIC}: &\quad \sum P_{\text{in}} = \sum P_{\text{out}} + F,\\
\mathsf{REDEEM}: &\quad \sum P_{\text{in}} = \sum A_{\text{out}} + F.
\end{aligned} \tag{12}$$

The fee is credited exactly once to existing-value reward liabilities. In a confidential transfer, V_C decreases by F. On exit it decreases by $A + F$, while V_P increases by A. These updates never materialise both a public cell and an ordinary UTXO for the same destination.

5.3 Fees and a funded exit

For ordinary confidential attempts the minimum fee is a deterministic function of reserved resources:

$$F_{\min} = f_0 + p_B B_{\text{fee}} + p_M M_{\text{reserved}} + p_O N_{\text{created}}. \tag{13}$$

All coefficients are versioned integers in native base units. B_{fee} is a precomputable byte reservation for the selected class and proof profile: fixed envelope fields, capsule and public-witness lengths, charged framing, and the profile's maximum proof length. It is independent of proof randomness and of the eventual proof bytes. Admission requires the actual encoded length to fit that reservation. Thus the fee can be bound into the intent before producing signatures and a proof; there is no fee–proof-length fixed point. The rule charges neither a percentage of the hidden amount nor a measurement of local verification time. Optional tips form part of the authorised intent. Deletion and cache hits do not produce negative mass or release already reserved execution capacity.

Each live confidential note contains $e = E_{\text{note}}$ as owned value. Its note suite preserves a one-input/one-public-output EXIT profile with mandatory fee at most e. Consequently, for face value $v + e$, the terminal public amount satisfies

$$A = v + e - F_{\text{exit}} \ge v > 0. \tag{14}$$

The user funds this allowance when creating the note. It is neither a separate native reserve nor value already paid to validators. Two notes exiting together can release an unused allowance into the destination. The wallet displays the spendable amount and the total owned face value distinctly.

Equation (14) applies to the mandatory one-input terminal fee; a voluntary tip can reduce the returned amount further. A multi-input EXIT independently satisfies the positive-output and amount-range guards. A stable funded exit is an obligation of the suite lifecycle. A later fee or verifier change preserves that path or supplies a funded migration. This condition establishes an affordable valid operation, not a deadline for admission under congestion or censorship.

5.4 Direct-inbox fee attribution

A directly registered confidential request has no containing work block. Its fee therefore cannot use the base rule that rewards the block containing the first effective occurrence. TSP-2 splits effective fees 50/50 with the base's carried rounding residue, in canonical execution order. For denominator $b = 10^4$ and work share $\omega = 5000$,

$$F_W = \left\lfloor \frac{\omega F + \epsilon}{b} \right\rfloor, \quad \epsilon' = (\omega F + \epsilon) \bmod b, \quad F_S = F - F_W. \tag{15}$$

The stake share goes to the execution-height accounting pool. The work share enters a bounded existing-value fee bucket for anchor $a = h - 1$. At that anchor's normal closure it joins the confidential-fee carry. For $N > 0$ registered blocks, quotient and remainder allocate the sum exactly in ascending block-ID order; for $N = 0$, the scalar carry persists to the next nonempty closure.

Allocation descriptors feed the existing bounded reward-settlement machinery. A work claim remains payable only after execution and anchor closure, and never receives an addition after payment. Fee carry resides inside the work reserve, distinct from unminted subsidy carry. Thus empty periods need not create an unbounded list of new obligations. Ordinary work-carried PUBLIC and REDEEM payments retain the base's containing-block fee treatment.

A conflicting, expired or otherwise non-applicable admitted payment has no value effects and pays no fee. Its reserved processing still counts. This follows the native-payment no-effect rule (TRAIDAG, Section 4.7.3): a source cannot be consumed again merely to charge a failed attempt.

6 Deterministic admission and settlement

6.1 Anchors, transaction identities and applicability

A heavy envelope names an immutable finalised admission anchor under which every source exists. Its transaction core separately binds the exact intended source incarnations, destinations, fees and execution interval. It can be presented under a newer eligible anchor when those inputs are unchanged. This rebasing changes neither spending intent nor the identity of an already queued envelope.

Registration obtains the full witness-bearing bytes, checks the anchor, resolves the input records, verifies the proof and public authorisations, and reserves the complete resource vector. The queue commits the exact admitted witness identity and its validated provenance. A semantic transaction ID deduplicates pending intent; a witness ID identifies the concrete bytes used for availability and verification.

Execution rechecks each input against current state. Successful proof verification is not a cached assertion of perpetual unspentness. A request can be structurally admissible at registration and later have no effect because another transaction consumes an input first. This does not retrospectively invalidate its original registration.

6.2 Two queues within one cut

Heavy confidential envelopes are not valid payload items inside ordinary work blocks. Let Q^W, Q^C be the previously committed work and confidential queues. Their execution sets are

$$E_W = \text{prefix}(Q^W; \mathcal{B}_W), \qquad E_C = \text{prefix}(Q^C; \mathcal{B}_C). \tag{16}$$

Each prefix is the longest initial segment satisfying every component of its resource budget. Selection stops at the first entry that does not fit. A later smaller entry does not move ahead. All admitted occurrences count, including ones that later fail current-state guards.

Work blocks retain TRAIDAG's execution-ID/hash permutation and their committed internal transaction order. Confidential entries execute in FIFO order. A new confidential batch appends in ascending order of

$$\big(\mathrm{H}_{\mathrm{CREG}}(\text{prevCut} \parallel \text{txid}), \text{txid}\big). \tag{17}$$

Registration height precedes the within-batch key, so later admissions cannot insert ahead of earlier requests. Proposer discretion over newly admitted candidates remains; deterministic ordering is not a proof of unbiased admission or freedom from extraction.

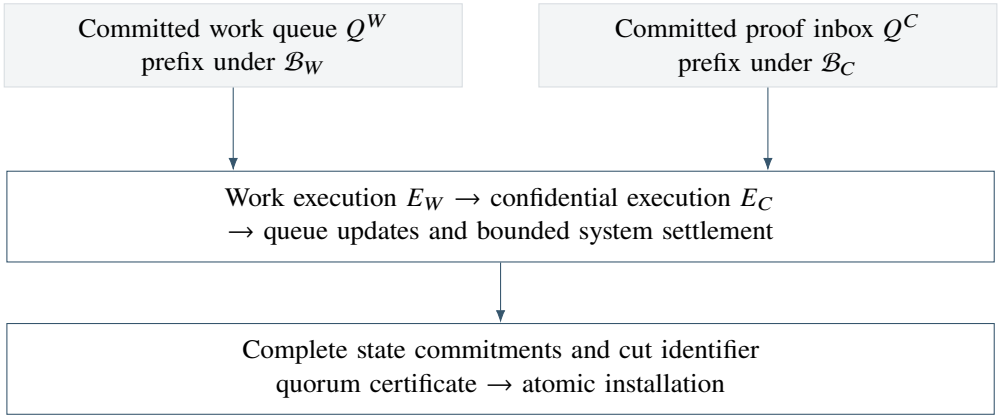


Figure 2. Independent resource prefixes, one state transition. Proof saturation does not alter the work-prefix selection rule; physical resources and certified finality remain shared.

6.3 Fixed phase order and atomic application

A cut resolves both prefixes from the immutable predecessor state. New registration batches are checked against their anchors in a read-only view. The node then executes E_W, executes E_C, updates both queues and performs base system settlement with the fee extension. It forms the state roots and cut ID only after these effects are defined. Both queues and both registration batches are committed by the new cut schema.

A native source consumed in the work phase is unavailable to a later ENTER in the confidential phase. Neither phase may consume a child output absent from the request's admission anchor. These rules remove any dependence on thread scheduling or which service's proof work finishes first.

A successful confidential operation produces one write set: consume exact inputs, advance generations, allocate outputs, install full capsules or public records, update value domains and fee liabilities, and recompute authenticated counts and mutation summaries. The enclosing cut is installed atomically with its certificate, queue state and indexes. A failure exposes either the complete predecessor or the complete successor, never both an old spendable input and its new output.

6.4 Deterministic failures and data retention

Malformed encoding, invalid proof, invalid signature and inconsistent output encryption are admission failures. For an admitted and fully available envelope, the no-effect priority is: execution interval; current input conflict; public authority or policy mismatch; prohibited current suite operation; public-effect custody guard; then net-growth capacity. Invariant failures such as unexplained custody underflow are distinguished from an ordinary policy limit and stop local validation.

A missing body, input record or committed capsule is unavailable data, not proof of nonexistence. Nodes obtain the data before certifying or applying the cut. Registered envelopes and validated historical provenance remain pinned through execution and committee handoff. Provisional data for a live consensus value obey the base retention rules; a view timeout alone does not discharge them (TRAIDAG, Section 4.4.1 and Appendix B.2).

All parsing, integer arithmetic, allocation and verification rules are functions of canonical inputs. Consensus uses no local clock, floating-point rounding, operating-system random challenge or database iteration order. Parallel preverification is an implementation optimisation; state effects and reported errors remain in the prescribed order.

6.5 Integer semantics

Amounts use nonnegative native base units with a manifest bound $M_{\text{amount}} < 2^{120}$. Stored amounts are u128 values; intermediate sums and products use checked u256 arithmetic or a wider explicitly bounded representation. For example, the product of two values below 2^{120} is below 2^{240} and fits u256. Bounds are established before multiplication or allocation, not recovered by clamping an already overflowed result.

The proof relation reproduces these integer semantics using range-constrained limbs and explicit carries. Source references are pairwise distinct, every output is positive and within the amount bound, and counts are checked before summation. These are independent constraints, not consequences of a total-balance equality. An equality in a finite field alone is insufficient: residues can agree although the corresponding currency integers differ. Where an operation uses floor division, its integer quotient and remainder satisfy

$$ab = qd + r, \qquad d > 0, \qquad 0 \le r < d, \tag{18}$$

with bounds on every intermediate. Field inversion is not a substitute for that condition. Similarly, a 384-bit hash is represented injectively by constrained bytes or limbs; reducing it modulo one smaller field element would define a different identifier. These rules apply equally to accepted and rejected attempts and to every node architecture.

7 Safety analysis

The results in this section refer to an accepted execution history beginning from a valid initial state. Initial activation creates K FREE cells with zero generations and zero confidential/public-cell aggregates; it does not copy native value into the new domain. They are conditional on section 2's cryptographic and atomicity premises. They do not turn a quorum certificate into a substitute for execution validation.

7.1 One-use consumption and authorisation

Lemma 7.1 (Incarnation exclusion)**.** *Within one valid accepted history, a live cell incarnation can be consumed successfully at most once.*

Proof. Let the incarnation be named by (s, g, D), with its full reference as in equation (2). The first successful consumption removes the corresponding live record and changes the slot generation from g to $g + 1$. If the slot is reused immediately, the replacement has generation $g + 1$ and a newly committed identity; if it remains free, its header still records $g + 1$. Subsequent generations are nondecreasing and never wrap by equation (4). Therefore no later current-reference check can match the old live incarnation. Atomic installation prevents interrupted execution from preserving that old input alongside the completed destination effects. Hence a second successful consumption contradicts the current-reference predicate. □

Corollary 7.2 (No historical replay index)**.** *The one-use property of TSP-2 cell inputs does not require a permanent transaction-ID blacklist or spent-nullifier set.*

Proof. Every supported payment has at least one one-use source. A replay of a cell-funded payment names an incarnation already excluded by lemma 7.1; a replay funded by an ordinary UTXO fails the base current-unspentness rule. Fresh proof randomness does not change those references. Once pending deduplication and explicit serving obligations end, the historical intent ID is not needed for this check. □

Proposition 7.3 (Owner-authorised consumption)**.** *Under knowledge soundness of* Π*, unforgeability of the spending signature and binding of the note commitment and intent, an efficient party lacking the owner's authorisation cannot cause an accepted confidential consumption, except with the corresponding cryptographic failure probability.*

Proof. An accepted confidential consumption contains a proof for the resolved current input commitment and full transaction intent. Knowledge soundness provides an extractor for a witness to the accepted statement, within the extraction error assumed in section 2. That witness includes a note opening and a spending signature whose public key hashes to the note's spending commitment. Binding prevents replacing that opening or intent with another one while retaining the accepted commitments. If the owner did not authorise the extracted intent under that key, the extracted signature is a forgery. A receiver-descriptor certificate has a different domain and context and cannot supply the required spend signature. Knowledge of the sender-created note opening alone is therefore insufficient. □

7.2 Value conservation

For analysis, associate each live confidential commitment with the face value in its valid creation witness. Binding and the successful-spend relation ensure consistency when it is later consumed. This conceptual opening map is not a public state table and need not be available to validators.

Theorem 7.4 (Native-value preservation)**.** *Assume the initial state satisfies equation* (11)*, the aggregates represent their value domains, and accepted confidential proofs satisfy equation* (12) *with authenticated sources and bounded integer arithmetic. Each successful TSP-2 payment preserves the extended native-value equation and has zero authorised issuance delta.*

Proof. Consider each class. ENTER removes public source value P and creates confidential face value $P-F$, while adding F to base reward liabilities. TRANSFER removes $\sum f_{\text{in}}$ and creates $\sum f_{\text{out}} = \sum f_{\text{in}} - F$, again crediting F to liabilities. EXIT removes confidential value $A + F$, creates public-cell value A and credits F. PUBLIC replaces public sources with public destinations whose total is smaller by F. REDEEM removes public-cell value and creates ordinary outputs smaller by F, with the same fee credit.

In every case the sum of live value domains changes by zero. No class writes the native mint counter or invokes subsidy creation. Lemma 7.1 excludes counting a source twice; the atomic write set excludes counting both its old and replacement representation. Fees are distributed by subtraction after the work

share, so equation (15) preserves $F_W + F_S = F$ exactly. No-effect transactions produce the empty write set. Induction over the canonical transaction sequence, followed by the base's separately authorised system transitions, proves the result. □

A matching public aggregate cannot establish the proof premise. For example, a flawed relation might permit one user to claim another user's confidential value without increasing total native supply. The state equation detects neither arbitrary ownership forgery nor a broken hiding construction. Authorisation, proof knowledge soundness and accounting are separate parts of the argument.

7.3 Capacity-safe terminal exit

Theorem 7.5 (Storage-feasible exit)**.** *Let a valid EXIT consume $m \in \{1, 2\}$ confidential cells and create one PUBLIC cell. If the source references, proof, fee and policy conditions are satisfied, the transaction cannot fail solely for lack of a free arena slot or ordinary-UTXO allocation space.*

Proof. Its new occupancy is $L' = L - m + 1 \leq L \leq K$. Allocation first uses consumed slots, so at least one suitable slot exists within the transaction's own write set. The public payload fits the same fixed cell size. EXIT creates no ordinary UTXO and therefore does not consult an external ordinary-output allocation budget for its destination. The output-positivity guard gives a positive destination; the mandatory single-input terminal fee additionally preserves the lower bound in equation (14). Thus neither of the stated storage shortages can prevent this otherwise valid transition. □

The theorem establishes feasibility of the terminal operation, not arbitrary splitting at full occupancy. A one-input/two-output payment has $L' = L + 1$ and can require a free slot. Nor does storage feasibility imply prompt registration under censorship or a quota guaranteeing every exit in one cut. PUBLIC payments keep the terminal native claim usable inside the arena; REDEEM moves it to the ordinary UTXO set when that allocation is available.

7.4 State equality

Theorem 7.6 (Deterministic cut execution)**.** *Two conforming nodes with the same valid predecessor state, activated manifest and complete certified cut input derive identical TSP-2 effects, fee allocations, result codes and successor commitments, independently of local data-arrival order and execution scheduling.*

Proof. The predecessor fixes both queues and their reservation vectors. Equation (16) therefore fixes the same execution sets. The base ordering fixes the work sequence; FIFO order fixes the confidential sequence; and the phase rule fixes their concatenation. Each canonical envelope resolves the same retained anchor provenance. The deterministic verifier and fixed guard precedence give the same admission and applicability decisions.

For a successful operation, exact integer equations determine amounts and fees, while the consumed-slot-first rule determines output locations. Generation increments, record encodings, last-mutation heights and subtree summaries are consequently equal. A no-effect operation changes neither node. Induction over the operation sequence gives identical intermediate states. The bounded system phase, composite roots and cut identifier are then equal by the base rules and the explicit fee extension. Neither node uses a local timer or completion order as an input to any step. □

Agreement on which cut is accepted remains the responsibility of TRAIDAG's certified-consensus rules. Theorem 7.6 establishes equality for a shared accepted input; it does not claim that exceeding the base fault assumptions cannot produce conflicting histories.

8 Privacy and public information

8.1 Confidential values with an observable graph

TSP-2 separates a confidential witness from a public spending statement. Confidential input public keys, spending signatures and note openings can remain inside the zero-knowledge witness. The public ledger nevertheless identifies the cells consumed and the outputs created. Transaction timing, class, counts, fees and public boundaries are also visible.

For a transaction sequence T, its intended structural leakage can be described as

$$\mathcal{L}(T) = \big(G(T), \text{classes}, \text{counts}, \text{fees}, \text{public values}, \text{heights}, \text{profiles}\big), \tag{19}$$

where $G(T)$ is the graph of public input references and output identities. This expression specifies what a confidentiality argument would condition on; it does not itself prove confidentiality of the candidate encryption composition. Network observations and information supplied by counterparties are additional inputs to an observer.

Table 3. Observation boundary of the confidential payment classes.

Information	Treatment
Confidential amount and note opening	Not published in plaintext; constrained by the proof and subject to deductions from public flows.
Long-term receiver descriptor	Exchanged off chain; not a public output field. Ciphertext recipient anonymity is a separate key-privacy property.
Spent input references and new outputs	Public. Observers can retain the transaction graph.
Input/output counts, class and fee	Public and bounded; no permanent dummy-nullifier mechanism.
Entry and exit values	Public through source/destination values and the accounting relation.
Counterparty knowledge and network metadata	Not concealed by the on-chain ZK argument.

A commitment does not hide a value already forced by public conservation equations. For an ENTER with one confidential output, public source sum P and fee F reveal its face value $f = P - F$; a public fixed allowance also reveals $v = f - E_{\text{note}}$. With two outputs, the sum is still known. Later graph constraints or external disclosures can reduce the remaining uncertainty. These deductions are consistent with zero knowledge: the proof need not reveal anything beyond a public statement that already determines the value.

8.2 Recipient privacy and cryptographic composition

Omitting a receiver's long-term address from an output is necessary for the intended receiver protection, but not sufficient for anonymity. Key privacy asks whether an observer given a ciphertext can distinguish which of several known public keys was used [2]. The requirement concerns the entire exposed capsule, associated data and commitments, not merely an underlying KEM in isolation.

The note's spending- and encryption-key commitments are inside the hidden plaintext. The public note commitment includes fresh randomness, and the proof keeps the receiver descriptor private. Whether these components jointly give recipient anonymity depends on their composition and the adversary's auxiliary knowledge. Accordingly TSP-2 specifies recipient key privacy as a separate cryptographic property rather than inferring it from ML-KEM or from the word "ZK."

The sender knows its chosen destination and amount and can retain the resulting output identity. Multiple public inputs may support a common-control inference. Restoring a wallet through repeated requests for precisely its own cells can also reveal associations; the current-state range protocol instead permits local scanning of shared ranges. None of these observations is undone when another node prunes its history.

8.3 Privacy and recoverability are distinct

A capsule can be perfectly recoverable by its intended owner while providing insufficient recipient privacy. Conversely, a cryptographically private note can become operationally unusable if its recovery bytes are erased. TSP-2 makes the retention obligation unconditional for live outputs, while the confidentiality claims remain attached to the selected cryptographic profile. This separation avoids treating a compact root, a hidden amount and an anonymous transaction graph as interchangeable properties.

9 Current-state recovery and pruning

9.1 Authenticated complete records

Each cell is committed through a fixed-depth binary tree over all K slots, including FREE headers. For leaf s, define the tuple

$$T_s = (D_s, n_s, t_s), \quad D_s = \mathrm{H}_{\mathrm{CELL_LEAF}}(\mathrm{u32}(s) \,\|\, \mathrm{enc}(c_s)), \quad n_s = \mathbf{1}[\mathrm{kind}(c_s) \neq \mathsf{FREE}]. \tag{20}$$

The last-mutation height t_s is read from the committed record. An internal node at depth j has

$$\begin{aligned} D_j &= \mathrm{H}_{\mathrm{CELL_NODE}}(\mathrm{u16}(j) \,\|\, \mathrm{enc}(T_l) \,\|\, \mathrm{enc}(T_r)), \\ n_j &= n_l + n_r, \qquad t_j = \max(t_l, t_r). \end{aligned} \tag{21}$$

The exposed root descriptor binds the namespace, schema, capacity, record size and root tuple. Counts and maximum heights are derived, not independent claims supplied by a server. Every changed or newly allocated cell receives the containing finalised height as its mutation height.

A conforming validator retains every current cell's full bytes. This includes the complete capsule of a confidential output regardless of birth height. A snapshot export enumerates all fixed slot ranges and permits reconstruction of the complete root. A sparse encoding is equivalent only when it also authenticates FREE ranges and covers every slot exactly once. Individual inclusion proofs for a selected subset do not establish completeness.

9.2 Recovery from a seed

Let $\mathrm{Keys}(z)$ be the fixed account derivation from effective seed z. For an authenticated arena snapshot A_h, the wallet enumerates the current cells, locally decapsulates and decrypts confidential capsules, and checks each successful opening against its note commitment, origin, key commitments and value rules. Public cells controlled by a supported seed-derived ordinary lock are recognised through the corresponding public-key derivation. Independently imported keys remain separately backed-up accounts.

Theorem 9.1 (Age-independent recovery)**.** *Let c be a currently unspent confidential output created for a valid supported seed-derived receiver descriptor. Under accepted-output relation correctness, successful capsule decryption and complete authenticated current-state enumeration, the owner recovers c from the effective seed and the current arena, without its original transaction body, old membership path or creation-time wallet database. The result is independent of the output's age.*

Proof. Creation installs the entire capsule and commitment in the live cell. The state rules never remove or alter that output's capsule merely because height advances, and a successful consumption is the only ordinary event that replaces the incarnation. Since c is still unspent, its committed capsule is present in A_h. Complete enumeration therefore reaches it.

The seed profile deterministically recreates the receiver's recovery and spending keys from public constants. The accepted output relation ties the capsule to the same descriptor, note and commitment. Under capsule correctness, decryption returns the note opening; recomputation of its commitments confirms its identity and amount. The wallet now has the opening and its spending authority, while the live record supplies the current spending reference. No step requires the creation height's historical body or an old Merkle witness. The proof uses only that the output remains live, not how many heights have elapsed since birth. □

The theorem is a recoverability result under stated premises. If the encryption profile establishes a uniform bound $\varepsilon_{\mathrm{dec}}$ for each of r relevant accepted capsules in the applicable adversarial model, a union bound contributes at most $r\varepsilon_{\mathrm{dec}}$ to failure probability. This implication does not assert that an honest-encapsulation average alone supplies that bound. Binding and proof-system failures contribute separately. Neither the theorem nor a standard encryption algorithm guarantees recovery with a missing seed or from destroyed current replicas.

9.3 Catch-up without a permanent event log

After scanning a snapshot at height h_0, the wallet obtains a newer authenticated root at height h_1 on the same accepted history and under the same arena schema. A capacity or schema migration starts a complete authenticated scan under the new layout rather than comparing incompatible subtree positions. It descends only into subtrees whose current maximum mutation height exceeds h_0. Unchanged subtrees can be reused. Changed leaves replace the wallet's view with their current records, including FREE headers.

Lemma 9.2 (Complete current-state difference)**.** *For valid states on the same accepted history, a subtree whose current maximum mutation height is at most h_0 contains no cell changed after h_0. Every cell changed after h_0 lies on a path of subtrees whose maxima exceed h_0.*

Proof. A cell's mutation height is assigned on every change and never decreases along the accepted history. Thus a cell changed after h_0 has current $t_s > h_0$, even if it is later consumed or becomes FREE. Each ancestor stores the exact maximum of its descendants, so every ancestor on its path has maximum greater than h_0. Conversely, a subtree maximum at most h_0 bounds every descendant mutation height by h_0, excluding a later change. The fixed tree covers all slots, including deletions represented by FREE headers. □

If most cells change during the scan, catch-up approaches a full current-state scan. The work is bounded by the current arena, not by every intermediate rewrite. The procedure recovers current holdings, not all incoming payments that were subsequently spent between snapshots. The wallet labels its balance by the verified snapshot height and catches up before constructing a new spend.

9.4 Retention, snapshots and serving

Current-data serving is part of the node role. It does not require unlimited simultaneous downloads or a separate historical snapshot for every client. Shared immutable snapshot images have fixed quotas; an expired image produces a restart or resume response. It never authorises a wallet to report an incomplete result as zero.

Old proofs, consumed capsules and obsolete cell versions become prunable once all registration, execution, handoff and history-serving obligations end. Current capsules and generation headers do not. Pending requests pin their precise bytes and validated anchor provenance, with admission reserving their storage before the duty is created.

Table 4. Storage roles after TSP-2 activation. All roles apply the same validity rules.

Role	Required current information	Additional historical service
Pruned	All live cells and capsules, generations, roots, accounting and pending duties	Bounded recent history; current-state wallet recovery.
Full history	The same complete current state	Complete historical bodies and proof replay.
Archive	The same complete current state	Historical state images and indexed past-state queries.

A joining node still distinguishes full-history validation from checkpoint synchronisation. The latter accepts an independently authenticated recent starting state and validates its continuation; it does not recreate verification of omitted past transitions. Snapshot installation checks complete data, roots and aggregate consistency and is atomic. TSP-2 adds its current capsules and queue provenance to that package without changing the base checkpoint trust boundary (TRAIDAG, Section 9.4).

10 Bounded storage and finite capacity

10.1 A storage bound independent of transaction count

Let B_c be the fixed canonical cell size and B_T the stored bytes per complete cached tree tuple. Let s be the maximum number of additional full snapshot images. Let $D_A, D_Q, D_P, D_H, D_J, D_M$ be fixed quotas for anchor versions, the confidential queue, shared provenance, recent serving history, journals/staging and remaining metadata respectively.

Theorem 10.1 (Bounded retained representation)**.** *For fixed protocol and storage parameters, a TSP-2 node obeying the stated retention quotas can represent its current arena, complete tree cache and bounded auxiliary data using at most*

$$D_{\mathrm{TSP}} \le (1+s)\big(KB_c + (2K-1)B_T\big) + D_A + D_Q + D_P + D_H + D_J + D_M. \tag{22}$$

This bound contains no term proportional to the cumulative number of past confidential payments.

Proof. There are exactly K cells of fixed width B_c. Replacing a cell changes its bytes but not the number or width of slots. A complete binary tree over those slots has $2K-1$ cached nodes, each of fixed width B_T. At most s additional images duplicate that bounded representation. Each auxiliary class is limited by its own fixed quota; registration reserves the required pending-data space, and new obligations that exceed a quota are not admitted.

Generations and heights have fixed widths and satisfy the protocol-domain bounds. No permanent nullifier or historical intent index is consulted. Obsolete versions, snapshots and journals are reclaimed according to their quotas rather than retained for every past update. Summing the disjoint upper bounds gives equation (22), independently of how many occupancy-preserving payments have already occurred. □

The theorem bounds a compliant representation. A storage engine that retains obsolete copy-on-write pages, WAL segments or compaction outputs outside the quotas does not satisfy its premises. Physical reclamation, not merely logical key deletion, determines disk usage. An overloaded or full-disk node stops or waits rather than deleting current monetary data.

10.2 Reference-size arithmetic

For the analytical profile $d = 21$, $K = 2^{21}$ and $B_c = 4096$, the canonical cells occupy

$$D_{\text{cells}} = 2^{21} \cdot 4096 = 8,589,934,592 \text{ bytes} = 8 \text{ GiB}. \tag{23}$$

Each cached tree tuple has 48 digest bytes, a four-byte occupied count and an eight-byte mutation height. Hence

$$D_{\text{tree}} = (2^{22} - 1) \cdot 60 = 251,658,180 \text{ bytes} < 240 \text{ MiB}. \tag{24}$$

The current cell-and-tree image is approximately 8.2344 GiB. Two published recovery snapshots and one building image give at most four such images including current state: approximately 32.9375 GiB. These are exact format calculations, not measurements of a particular database.

Table 5. Auxiliary storage terms in the analytical profile. The terms bound different retention duties and are not annual additions.

Component	Profile allowance
Current cells plus complete tree cache	About 8.2344 GiB.
Current image plus three snapshot images	About 32.9375 GiB total.
Recent anchor versions	At most 96 heights of bounded deltas; illustrative 502.125 MiB record/path allowance.
Confidential inbox and its provenance	At most 256 entries and 64 MiB.
TSP input provenance across both services	Separate 256-MiB ceiling.
Recent proof/body serving and receipts	256-MiB finite-window allowance.
Journals, staging and other metadata	Bounded deployment terms; 40 GiB is a combined TSP partition target, not a measured footprint.

The anchor illustration is $96 \cdot 1024 \cdot (4096 + 21 \cdot 60)$ bytes, conservatively charging a full path for each touched cell. Physical sharing may lower actual use; snapshot manifests, file alignment and recovery copies must fit their assigned terms rather than disappear from the accounting.

10.3 The cost of a finite arena

Storage can increase as previously free slots become occupied, but cannot exceed the configured arena bound. Continued replacement at a fixed population does not accumulate spend records. Lost or dormant coins still occupy their slots; this is the necessary consequence of preserving their spending and recovery rights.

A finite arena cannot support an unlimited simultaneous population of independent outputs. At saturation, new entries and net-growing splits can fail while equal-count transfers and terminal exits remain storage-feasible. Creation charges and growth quotas regulate admission, but a wealthy adversary can still compete for occupancy. Enlarging K is a coordinated capacity change, not an eviction policy.

The bound applies to the TSP component. Ordinary UTXOs, stake, rewards and any separately activated application state retain their own growth rules. Nor does pruning remove the bandwidth and CPU required to validate an envelope when it first arrives. A bounded live-state claim is neither a whole-node fixed-disk guarantee nor a zero-traffic claim.

11 Resource isolation and the 100-BPS profile

11.1 Logical work and independent service

Resource accounting covers the complete wire envelope, proof invocations, public signatures, cell and ordinary-state operations, hash work, temporary verifier memory, receipts and retained provenance. The manifest assigns fixed logical costs to these operations. A cached proof can reduce physical work without changing the amount reserved by an admitted occurrence.

Proposition 11.1 (Prefix-selection independence)**.** *For fixed predecessor work queue Q^W and work reservation vector $\mathcal{B}_W$, changing the confidential queue or exhausting its independent execution quota does not change the work prefix selected by equation* (16)*.*

Proof. E_W is computed solely from Q^W, its committed per-entry resource counters and $\mathcal{B}_W$. No counter derived from the confidential prefix enters this selection. Thus two predecessor configurations agreeing on those work inputs select the same E_W, regardless of Q^C or the value of E_C. □

This proposition is intentionally narrower than complete performance isolation. Both services use the same CPUs, storage, network and quorum. Shared provenance limits can affect new admission; the fixed phase order also resolves cross-service conflicts. The structural result excludes confidential-prefix truncation of an already fixed work prefix, not all indirect competition.

11.2 An analytical resource profile

Table 6 retains the TSP-2 analytical envelope. It partitions the base's total registration and execution budgets instead of adding uncharged proof work above them. Unused-budget borrowing is not part of the initial rule.

Table 6. Analytical cut budgets. Byte and count ceilings are specified independently; satisfying one does not imply satisfying all of them.

Resource	Registration	Execution
Work body reservation	6 MiB	9 MiB
Confidential body reservation	2 MiB	3 MiB
Combined body ceiling	8 MiB	12 MiB
Work logical mass	48 million	72 million
Confidential logical mass	16 million	24 million
Combined logical mass	64 million	96 million
Heavy confidential occurrences	8	12
Work-block count, 100-BPS profile	384	512

A heavy envelope has at most two sources and two destinations, a 128-KiB proof-size envelope and a 160-KiB complete-wire ceiling. The proposed per-request mass ceiling is two million units, conditional on the selected verifier fitting that cost model. The verifier scratch target is 256 MiB; process memory additionally includes base execution, TenQ evaluation, caches and simultaneous tasks.

Arena mutation events are limited to 1,024 per cut across both services. The work phase receives at most 960 and the confidential phase 64. Consumed and created cells count as events even when they reuse the same physical page. The individual-operation and queue limits ensure that every admitted execution unit fits within its promised service budget. Historical profiles remain attached to registered work when parameters change.

The difficulty controller's work-saturation predicate includes the work reservations, public-cell mutation allowance and shared input-provenance capacity. An exhausted confidential-only quota does not alone make work admission saturated. These are deterministic committed counters; local utilisation is not a difficulty input.

11.3 Rates, payloads and bandwidth

TRAIDAG's rate-test profile separates a target work rate of $R_b = 100\,\mathrm{s}^{-1}$ from a target cut cadence of $\rho = 2.5\,\mathrm{s}$ (TRAIDAG, Section 9.6). The expected work count in that cadence model is

$$\bar{N}_W = R_b \rho = 250. \tag{25}$$

With the 6-MiB work reservation, this gives an average encoded allowance of

$$\bar{B}_W \leq \frac{6 \cdot 2^{20}}{250} = 25,165.824 \text{ bytes per work block.} \tag{26}$$

Mass, authorisation and state-operation ceilings can be tighter. The calculation does not support 100 maximum-size one-MiB blocks per second.

For the confidential service, the count-only ceiling is

$$\tau_C \leq \frac{8}{2.5} = 3.2 \text{ admitted occurrences per second.} \tag{27}$$

An execution budget of twelve per cut permits backlog service above that nominal new-arrival cap when other constraints permit. Distinct successful transfers can be fewer than admitted occurrences. Their wallet proving time is additional to registration and execution latency, and is not constrained by a ten-millisecond work-block interval.

Eight maximum-size envelopes with four-byte length fields account for

$$\begin{aligned} B_{C,\mathrm{cut}} &= 8(163,840 + 4) = 1,310,752 \text{ bytes,} \\ \Gamma_C &= \frac{B_{C,\mathrm{cut}}}{2.5} = 524,300.8 \text{ bytes/s.} \end{aligned} \tag{28}$$

This is approximately 0.500012 MiB/s, or 45.30 decimal GB per model day of one-delivery body ingress. It is network traffic, not an annual addition to retained current state. Votes, proposals, duplicate delivery, egress, snapshot service and rejected traffic add their own terms.

11.4 Processing and capacity conditions

A sufficient operating condition for the target cadence has the form

$$T_W^{\max} + T_{C,\mathrm{reg}}^{\max} + T_{C,\mathrm{exec}}^{\max} + T_{\mathrm{sys}}^{\max} + T_{\mathrm{comm}}^{\max} + T_{\mathrm{durable}}^{\max} \leq \rho, \tag{29}$$

for the intended load, topology and hardware model. The terms represent qualified worst-case service stages, not quantities that a node uses to select a shorter valid cut. Overlap and caching may improve a measured implementation, but the protocol cannot rely on them to reinterpret logical costs. Network synchrony assumptions are necessary for any finite communication term.

The 100-BPS statement is consequently an operating-profile claim contingent on the joint work represented in equation (29). Prefix independence proves a scheduling property, not that a particular STARK relation fits the proposed execution time. In particular, verifying up to two input signatures, two output-descriptor certificates and two complete capsule encryptions inside a proof contributes to the prover cost. The selected relation cannot omit recovery checks merely to fit a target envelope.

There is also a separate occupancy condition. If every admitted request creates two net-new cells at the rate in equation (27), an empty 2^{21}-cell arena reaches capacity after

$$\frac{2^{21}}{2 \cdot 3.2} = 327,680 \text{ seconds} \approx 3.79 \text{ days}. \tag{30}$$

This sustained-growth scenario is not a usage forecast. It shows why replacement throughput, net-growth admission and total live capacity are different parameters. A workload dominated by two-input/two-output transfers has zero net occupancy growth, while an equally fast stream of new entries does not. Creation prices and net-growth quotas therefore address a different resource from proof throughput.

12 Operational security and protocol continuity

12.1 Bounded work under adversarial input

Cheap length, type and profile checks precede expensive verification. Global limits bound undecoded messages, preverification queues, concurrent proof tasks and body fetches; distributing traffic across many peers must not bypass the global bounds. Invalid unregistered proofs do not automatically pay an on-chain fee. Economic charges supplement these controls but cannot replace them.

Valid conflicting requests also consume resources. Pending intent deduplication limits identical queued copies, while exact witness/context cache keys prevent a previous result from authorising different bytes. A later input conflict still requires the current-state check and gives no value or generation change. Reserved work is not refunded after discovering that a request has no effect.

Current-state recovery is an independent bandwidth consumer. Repeated multi-gigabyte downloads require bounded range sizes, shared immutable images and global egress scheduling. Service throttling changes response time, not which live outputs belong in the committed state. A server cannot establish a complete balance by returning only selected valid inclusion proofs.

12.2 Durability and state reclamation

The arena, tree, value aggregates and queue updates share the atomic cut boundary. Recovery exposes either the prior committed image or the complete new one. A checksum-valid mixture of old generations and new destination records is not a valid snapshot. Generation rollback is especially important: a stale record must not reappear as a spendable incarnation after a crash.

A bounded physical representation can use reusable pages, a fixed current tree and bounded journals. Copy-on-write snapshots or an LSM engine are also compatible if obsolete physical versions are reclaimed within the storage contract. Memory pressure and disk exhaustion are local failures, not instructions to drop current capsules or weaken proof verification. Extra hardware can improve performance but cannot change the transition function.

12.3 Old outputs and upgrades

A proof backend may change while proving the same immutable note relation. Such a change need not require owners to refresh unspent outputs. The system retains the decoding, deterministic seed derivation and terminal spending support of each note suite while that suite has live outputs or outstanding registered requests and provenance obligations. A zero current output count alone does not permit retirement of a suite still referenced by the queues. A finite suite registry rejects unsupported new suite creation at saturation rather than invalidating older money.

Activation binds the state and cut schemas, transaction classes, inbox ordering, fee attribution, cryptographic profiles and all resource ceilings. Pending work and its promised execution/retention obligations retain their interpretation across the boundary. A node lacking an activated verifier is not a fully validating node for that network. Transport negotiation cannot activate a new consensus version.

The direct-inbox and live-cell model also requires native wallets and payment interfaces to recognise PUBLIC cell references. A PUBLIC coin is not a secondary token; nevertheless an application that understands only ordinary UTXO references needs the typed native adapter before it can accept that source. Applications accepting these sources use the same checked native adapter; copying their face values into a second spendable ledger is not permitted.

12.4 Faults beyond the state argument

Correct state accounting does not repair a compromised seed, a broken proof primitive or unavailable current replicas. A fundamental proof-soundness failure may create counterfeit claims against existing deposited value without increasing native supply. There is no general public procedure that then distinguishes every honest hidden claim from every forged one. A public exposure ceiling limits native value in the confidential domain but does not prove claimant correctness.

A coordinated pause or migration changes protocol operation explicitly; it does not grant maintainers a universal decryption or spending key. The analysis begins from an empty, zero-value arena or from a separately authenticated valid state. Importing another payment system's balances requires a value-preserving migration; the theorems do not establish the validity of an arbitrary imported snapshot.

13 Related work

Confidential Transactions in Elements demonstrate the separation between hidden amounts and public UTXO-style transaction structure [8]. TSP-2 uses that distinction in its privacy scope, but specifies a hash-based note commitment and a general zero-knowledge relation rather than importing Elements' group-based balance construction. Public spend links are a conscious part of the state model.

Quisquis approaches anonymous payments through updatable keys and zero-knowledge arguments, avoiding the same ever-growing set of candidate unspent coins described in its comparison with note-based systems [9]. Its DDH-based construction is not a post-quantum component of TSP-2. The relevance is architectural: privacy state need not always be organised as an append-only historical note set. TSP-2 instead chooses explicit live-output deletion and accepts the corresponding graph disclosure.

STARKs provide a transparent argument family for computational integrity and zero knowledge [7]. Their role here is to prove the payment relation while leaving ordering and current-source checks to the ledger. The existence of that proof family does not determine the costs of this particular relation or prove recipient anonymity of a chosen capsule composition.

TSP-2 supplies the transaction and recovery context around standardised ML-DSA, ML-KEM, HKDF and GCM components [4, 5, 3, 6]. Recipient anonymity additionally invokes the distinct notion of key privacy [2].

TRAIDAG's separation of work production from certified execution allows the confidential inbox to share finality without equating every work block with another heavy proof. TSP-2 preserves that separation and its conditional agreement model (TRAIDAG, Sections 4–5).

14 Conclusion

TSP-2 makes current money, rather than transaction age, the unit of retention. A confidential output carries its own complete recovery capsule until it is spent. Consumption advances a reusable slot generation, replacing the old incarnation without preserving a permanent marker for every historical payment. The same arena supports a native public exit, which remains storage-feasible at full occupancy.

Under the stated cryptographic and execution assumptions, the transition preserves one-use spending, native value and deterministic roots. Complete authenticated current state suffices to recover a supported seed's unspent outputs, and fixed arena and auxiliary quotas bound the TSP representation independently of cumulative traffic. The cost of these properties is explicit: finite simultaneous capacity, complete current-data service and a publicly observable input/output graph.

The separate proof inbox gives confidential requests a defined share of TRAIDAG's resources without placing their proof bodies in every work block. The 100-BPS profile therefore remains a joint capacity question with calculable byte, count, state and verification terms. Confidentiality is optional for users, but its accounting, recovery and validation rules are part of the same ledger.

A Canonical representation

This appendix collects the reference layouts used in the size calculations and state arguments. All integers are unsigned, fixed-width and big-endian. Counts and lengths precede variable fields and are checked before allocation. Hash384 is 48 bytes. The activated manifest fixes literal class tags, domain preimages and cryptographic profile identifiers; unknown critical values and noncanonical encodings are rejected.

A.1 Fixed cell layout

Table 7. The 4096-byte cell representation. Payload padding is canonical and included in the record commitment.

Offset	Field	Bytes
0	Record schema	2
2	Kind: FREE, CONFIDENTIAL or PUBLIC	1
3	Slot index	4
7	Generation	16
23	Last-mutation height	8
31	Current output birth height; zero when FREE	8
39	Note suite; zero when not confidential	2
41	Output origin; zero when FREE	48
89	Output index; zero when FREE	1
90	Note commitment; zero for FREE/PUBLIC	48
138	Payload length	2
140	Payload followed by zero padding	3956

A CONFIDENTIAL payload holds the 1364-byte capsule from equation (9). A PUBLIC payload contains u128(amount) and the base 41-byte lock. FREE has no value payload but retains its structural identity, generation and mutation height. No uncommitted ciphertext appendage is permitted outside the record.

A.2 Transaction and witness envelope

```
Core = u16(protocol_version) || chain[48] || namespace[48]
    || u8(class) || u16(note_suite) || u16(proof_profile)
    || u64(valid_from) || u64(valid_until) || nonce[32]
    || u128(fee)
    || u8(native_input_count) || NativeRef[...]
    || u8(cell_input_count) || CellRef[...]
    || u8(output_count) || OutputDescriptor[...]

NativeRef = previous_txid[48] || u32(output_index)
CellRef = u32(slot) || u128(generation) || note_id[48]
    || expected_commitment[48] || expected_record_digest[48]
OutputDescriptor = u8(kind) || u16(note_suite) || origin[48]
    || u8(index) || commitment[48]
    || u16(payload_length) || payload[...]

Envelope = Core || u8(public_witness_count) || Witness[...]
    || u32(proof_length) || proof[...]
Witness = u8(ML_DSA_profile) || public_key[1952]
    || signature[3309]
```

Class rules restrict kinds, counts and witness roles. Native references precede cell references in the signable combined source vector; source indices refer to that vector. The total source count is at most two, and no source appears twice. Within each vector the encoded order is binding, not replaced by local sorting. PUBLIC and REDEEM have zero proof length and no confidential suite. REDEEM descriptors use a class-specific ordinary-destination tag with a positive u128 amount and the base 41-byte lock. They create no arena cell; their suite and commitment fields are canonical zero. Their origin and output index remain bound to the transaction. Public amounts and locks resolved from the anchor enter the signature and proof contexts even when not repeated in the wire reference. Confidential amounts are never substituted by public cleartext fields.

A.3 Acyclic identity construction

Origin O binds chain, namespace, class, the fixed transaction prefix, nonce, fee and ordered source references, excluding output fields and witnesses. Each output binds O and its canonical index. Commitments, associated data and complete capsules are then constructed; the completed Core determines txid; signatures and proof follow last.

$$\begin{aligned} O &= \mathrm{H}_{\mathrm{ORIGIN}}(\text{fixed intent prefix} \parallel \text{ordered sources}),\\ \text{noteid}_i &= \mathrm{H}_{\mathrm{NOTEID}}(\text{chain} \parallel \text{namespace} \parallel O \parallel \text{u8}(i)),\\ \text{txid} &= \mathrm{H}_{\mathrm{TX}}(\text{Core}),\\ \text{wtxid} &= \mathrm{H}_{\mathrm{WITNESS}}(\text{txid} \parallel \text{witnesses} \parallel \pi). \end{aligned} \tag{31}$$

These formulae use the profile's canonical, length-bound preimages. In particular, the witness preimage contains the public-witness count, each complete witness, the proof-length field and the exact proof bytes, not an ambiguously concatenated byte stream. The origin, commitment and identifier dependency graph contains no reference to the future output slot or the current cut ID. Installing an output binds its logical identity to the actual slot and generation in the cell commitment.

A.4 Key derivation and signing contexts

The initial wallet profile derives 32 bytes for ML-DSA's internal key-generation seed and 64 bytes for ML-KEM's internal (d, z) seeds from separate HKDF labels. The input is the effective 256-bit seed supplied by the wallet format. Mnemonic/passphrase conversion is fixed by that format, not inferred from a library's default random key generator.

The receiver certificate signs a 48-byte digest over the DESC domain, chain, namespace, note and encryption suites, S and E, using the pure ML-DSA interface and literal context `TSP2_DESC`. A source authorisation signs the SPEND-domain digest of chain, namespace, protocol version, txid, source role, source index and the resolved source-context digest, with literal context `TSP2_SPEND`. Role 0 denotes a public source; role 1 denotes a confidential cell. A certificate for a key pairing is not interchangeable with either source authorisation.

The E2 KDF salt is domain-separated by chain and namespace. The AEAD-key and AEAD-nonce expansion labels include the associated-data digest. The proof and wallet evaluate exactly the same key encodings, KDF, padding and encryption operations. Changing these interfaces is a suite change rather than an implementation preference.

A.5 Domain families

Literal reference domains use the prefix `TRAI_TSP2_` and suffix `_V1`. The distinct middle names are `CELL_LEAF`, `CELL_NODE`, `ARENA_ROOT`, `STATE`, `ORIGIN`, `NOTEID`, `NOTE`, `TX`, `WITNESS`, `SPEND`, `DESC`, `SPK`,

EPK, AAD, CREG, PROOF_STMT and SNAPSHOT. Spending-seed, recovery-seed, AEAD-key and AEAD-nonce derivations also have separate labels. The root and signed-context layouts bind these domains with their defined lengths and field widths; mathematical abbreviations in the main text do not authorise alternative serialisations.

B Transition and recovery algorithms

Algorithms use the activated relation and atomic, certified cut installation.

B.1 Current-state application

```
ApplyConfidential(S, registered_request):
    require exact admitted envelope and retained provenance
    evaluate fixed applicability guards in canonical priority
    require every input matches its current incarnation
    require class-specific integer balance and proof relation
    require occupancy - consumed + created <= capacity

    W = new transaction overlay over S
    reserve all distinct sources
    for each consumed slot in ascending order:
        clear old incarnation in W
        advance generation; set mutation height
    allocate outputs: consumed slots first, then lowest FREE
    install every complete capsule or native PUBLIC record
    move native value among disjoint domains
    credit exactly one work/stake fee split
    recompute cell, occupancy and mutation-height summaries
    return W

CommitCut(prior, cut):
    select both prefixes from prior queues
    validate new admissions against their anchors
    apply work prefix, then confidential prefix
    update both queues; apply bounded system settlement
    recompute full roots and verify certificate
    install state, certificate, indexes and duties atomically
```

No-effect returns no writes. Missing or inconsistent state stops validation.

B.2 Complete recovery

```
Recover(seed, snapshot):
    authenticate network, finalised reference and manifest
    derive the supported spending and recovery keys
    fetch every required current slot range
    verify coverage and reconstruct the committed arena root
    for every current CONFIDENTIAL cell:
        try capsule decryption locally
        check opening, commitment, key binding and value rules
        retain matching current references and note openings
    recognise supported seed-controlled PUBLIC locks
    return holdings labelled by the verified snapshot height

CatchUp(h0, current_root):
    authenticate the newer accepted root
    descend subtrees whose maximum mutation height > h0
    verify current changed records, including FREE headers
    replace the corresponding wallet view
```

Recovery operates on immutable shared snapshot images. A client switching images re-establishes a consistent root and range coverage. An incomplete transfer never establishes a complete zero balance.

C Reproducible analytical exhibits

C.1 State-model evidence

The companion Python model executes the five payment classes with finite cells, generation checks, atomic overlays, value accounting and current-image recovery. Its test run records 31,640 successful assertions, including 5,000 consecutive replacements at constant live occupancy. A separate unspent output remains byte-identical during the sequence and is recovered from the final serialised current image without the transaction history.

Table 8. Executed state-model observations. The model abstracts cryptographic verification and uses explicitly unencrypted test capsules.

Exhibit	Result
Model arena	16 fixed cells; 65,536 canonical bytes.
Consecutive confidential replacements	5,000.
Live occupancy during replacement sequence	Two cells, unchanged.
Unspent cold-output record	Byte-identical before and after the sequence.
Recovery source	Final serialised current-state image.
Full-arena EXIT and PUBLIC payments	Accepted in the model.
Old incarnation and malformed snapshot	Rejected in the model.
Model assertions	31,640 passed.

This experiment checks the algebraic and state-machine properties named in the table. Signature and proof outcomes are model oracles, and test capsules are labelled `TEST_ONLY_NOT_ENCRYPTED`. It therefore supplies no empirical estimate of proof soundness, recipient privacy, proving time or distributed throughput. Those are different observables from constant occupancy and replay rejection.

C.2 Reproduction and numerical scope

The source package contains a reference state model, its generated report and a separate script that recalculates the paper's storage, generation, fee and bandwidth exhibits. Running

```
python3 supplement/reference_model.py
python3 supplement/recalculate.py
```

regenerates the JSON reports using Python's standard library. Their hashes are listed in the distribution manifest. The model namespace is `TSP2_MODEL_`, deliberately distinct from production commitments.

The analytical profile is a parameterised example. Its 8-GiB record bound, approximately 32.94-GiB four-image bound, 3.2/s occurrence ceiling and 45.30-GB/day body-flow calculation follow from the stated inputs. None is a substitute for a measured database footprint or the joint processing condition in equation (29). The invariants remain parameterised when those operating values change.

References

[1] National Institute of Standards and Technology. *Secure Hash Standard (SHS)*. FIPS PUB 180-4, August 2015. SHA-384, Section 6.5. https://doi.org/10.6028/NIST.FIPS.180-4.

[2] M. Bellare, A. Boldyreva, A. Desai and D. Pointcheval. *Key-Privacy in Public-Key Encryption*. ASIACRYPT 2001, LNCS 2248. Full version, September 2001, Sections 1–2. https://cseweb.ucsd.edu/~mihir/papers/anonenc.pdf.

[3] H. Krawczyk and P. Eronen. *HMAC-based Extract-and-Expand Key Derivation Function (HKDF)*. RFC 5869, May 2010. Sections 2–3. https://www.rfc-editor.org/rfc/rfc5869.

[4] National Institute of Standards and Technology. *Module-Lattice-Based Digital Signature Standard*. FIPS 204, 13 August 2024. ML-DSA interfaces and Table 2. https://doi.org/10.6028/NIST.FIPS.204.

[5] National Institute of Standards and Technology. *Module-Lattice-Based Key-Encapsulation Mechanism Standard*. FIPS 203, 13 August 2024. ML-KEM interfaces and Table 3. https://doi.org/10.6028/NIST.FIPS.203.

[6] M. Dworkin. *Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*. NIST Special Publication 800-38D, November 2007. https://doi.org/10.6028/NIST.SP.800-38D.

[7] E. Ben-Sasson, I. Bentov, Y. Horesh and M. Riabzev. *Scalable, Transparent, and Post-Quantum Secure Computational Integrity*. IACR Cryptology ePrint Archive, Paper 2018/046, 2018. https://eprint.iacr.org/2018/046.

[8] Elements Project. *Confidential Transactions*. Official feature documentation. https://elementsproject.org/features/confidential-transactions. Accessed 18 September 2026.

[9] P. Fauzi, S. Meiklejohn, R. Mercer and C. Orlandi. *Quisquis: A New Design for Anonymous Cryptocurrencies*. ASIACRYPT 2019; full version, IACR Cryptology ePrint Archive, Paper 2018/990. https://eprint.iacr.org/2018/990.