Jurassic Bitcoin: Ossified Tech Debt as an Isomorphic Mosquito for New Features

Patrick Dugan

April 17, 2026
Revised October 8, 2026

Abstract

Bitcoin's pre-SegWit consensus history contains behaviours that are no longer desirable transaction forms but remain informative as design motifs. We isolate three: *transcript multiplicity*, *identifier bifurcation*, and *carrier camouflage*. Each has a measurable legacy seam (FindAndDelete context splits, `SIGHASH_SINGLE` digest collapse, and the `CHECKMULTISIG` dummy-element txid axis), each was deliberately closed at the base layer by the BIP62, BIP66, BIP143, BIP147 and BIP341 lineage, and each reappears as a *controlled* degree of freedom in modern overlay protocols. Our contribution is an observer pipeline that extracts historical windows, materialises seam fixtures against a Bitcoin Core policy oracle, classifies the resulting surfaces with a policy-envelope map, and lifts the abstractions into live TradeLayer prototypes with BitVM-style dispute flows [22, 24, 23, 38, 19]. This revision adds a node-free analytic reproduction of the three seams (every digest quoted here can be recomputed from one dependency-free script), a minimal formalisation of what controlled freedom requires (semantic binding, non-collapse, and authority separation), four Binohash-level primitives that each carry a capability plain Script lacks and a deterministic invariant, a limits result that records three Binohash-covenant escalations that break and why, the UTXORef mechanism layer that turns the motifs into deterministic handles bound to confirmed Bitcoin testnet4 evidence, and an explicit threat model. The point is not to revive old bugs. It is to translate fossilised Bitcoin behaviour into usable mechanics, in the spirit of Robin Linus' Binohash.

1 Introduction

We use *Jurassic Bitcoin* as shorthand for the historical period that ends, in practical terms, before SegWit-era modularity takes over the protocol surface. It names an older Bitcoin in which transaction behaviour still carried many laissez-faire, lightly curated technical fingerprints of an earlier engineering culture: behaviours that were tolerated, under-specified, or simply left in place long enough to ossify. That period is over as a live design regime, but its fossils remain informative. The claim of this paper is that some of those neglected fingerprints are still *isomorphically* usable as protocol mechanics once they are abstracted away from their original, historically contingent transaction forms.

If Jurassic Bitcoin supplies the fossils, the observer pipeline described here is the mosquito in amber: it preserves small but information-rich traces of a vanished protocol ecology, then lets us recover isomorphic mechanics without reviving the original environment itself. SegWit's own witness space was later used in ways nobody specified, through Ordinals and inscriptions [31]; whichever dinosaur one compares that to, the footprint became part of Bitcoin history, and isomorphic utility of unintended surfaces is real.

The central claim is that three motifs recur across both historical Bitcoin and modern overlay design:

1. **Transcript multiplicity**: one broad spend family can admit multiple effective signing transcripts.
2. **Identifier bifurcation**: externally visible identifiers can move while the committed core digest stays fixed.
3. **Carrier camouflage**: overlay commitments can hide inside ordinary transaction topology rather than obvious exotic carriers.

There is an inversion at the heart of the argument. At the base layer, every one of these motifs was first encountered as a *bug*: transaction malleability, a degenerate signature digest, an unconstrained stack element. The base layer responded by removing the freedom, culminating in the canonical witness digests of BIP143 and BIP341 [34, 35, 21, 20, 36, 37]. Overlay protocols, which commit to state through their own digests rather than through txids, can afford to re-admit the same manifold shapes in a controlled way. Section 7 states what "controlled" has to mean for that to be safe.

We study the motifs with a deterministic observability toolchain and then lift them into live prototypes where BitVM-style routes, challenges and resolutions can be exercised, without claiming that the legacy Bitcoin seams themselves are production-ready constructions. The perspective sits downstream of BitVM's dispute-oriented execution model and of Binohash-style transaction introspection, while remaining methodologically closer to differential blockchain testing than to new consensus design [22, 24, 23, 38, 19].

What changed in this revision. The April draft reported the three motifs and the Litecoin grafts. Since then the Bitcoin-side status moved: a TradeLayer grant transaction and a Taproot reserve vault were confirmed on Bitcoin testnet4 and bound into an off-chain route referee that consumes quirk route candidates (Section 9). This revision also adds the analytic reproduction of the seams (Section 6), the formalisation (Section 7), a related-work survey of the BIP lineage and of the newer introspection tricks that build on the same seams (Section 3), a threat model (Section 13), and corrections of record, including the fact that the repository's "2013 payout burst" window at heights 279150–279320 was mined in January 2014.

2 Contributions

1. A reproducible *historical-to-modern* workflow for turning Bitcoin fossils into measured experimental surfaces, with stable specimen identifiers and paper-ready report generation.
2. Three overlay-relevant motifs, each anchored to a legacy seam that the harness measures at the digest level, together with a dependency-free analytic reproduction so the cryptographic facts can be checked without a node.
3. A minimal formalisation of controlled freedom: four properties (semantic binding, non-collapse, authority separation, rotation invariance) that separate a usable isomorphism from a re-imported malleability bug.

4. Four Binohash-level primitives (append-only aggregatable attestation, state-bound handle rotation, a Binohash parameter bench, and a SINGLE-authored/ALL-frozen settlement that gets per-contributor output rows and a closed template with no soft fork), each demonstrated by a deterministic node-free invariant and annotated with the admission properties it enforces.

5. A limits result: three natural Binohash-covenant escalations that each break, for two recurring reasons (the deleted push must be a verifying signature; a core commits its own prevout) plus the reusable constant-one, with the mechanism behind each break reproduced node-free.

6. A policy-envelope mapping method that separates replay-only fossils, ordinary historical cover, and clean modern isomorphisms.

7. Live prototype grafts: nine LTCTEST procedural-token flow graphs, eight of them with dated live-run artifacts, with BitVM-style dispute edges; the UTXORef mechanism-handle layer; and a two-stage referee bound to confirmed Bitcoin testnet4 transactions.

8. An explicit threat model, including the consequence of Bitcoin Core 30.0's datacarrier policy change for the carrier-camouflage motif.

3 Background and Related Work

3.1 Legacy signature hashing and its seams

Before BIP143, every signature in Bitcoin committed to a digest produced by one function, `SignatureHash(scriptC tx, nIn, nHashType)`: blank every `scriptSig`, substitute `scriptCode` at input `nIn`, apply the `SIGHASH` mode, serialise, append the hash type, and double-SHA256. Three details of that function are the seams this paper measures [5].

FindAndDelete. In legacy `OP_CHECKSIG` and `OP_CHECKMULTISIG`, every signature being checked is first deleted from `scriptCode` wherever its push occurs at an opcode boundary, originally so that a signature cannot be asked to sign itself. The consequence is that the digest depends on the *multiset of signature pushes that happen to appear in the script*. For ordinary scripts this is a no-op; for a script that contains the signature bytes it is a transcript-selection knob.

`SIGHASH_SINGLE` degeneracy. If the signed input index has no corresponding output, `SignatureHash` returns the constant `uint256::ONE` instead of a digest. A signature over that constant verifies for any transaction that reaches the same branch with the same key. BIP143 removed the behaviour for witness programs by substituting an all-zero `hashOutputs` in that case [21].

The `CHECKMULTISIG` dummy element. Legacy `OP_CHECKMULTISIG` pops one extra, unused stack element. Its content does not enter the digest but does enter the serialised transaction, so it moves the txid without invalidating any signature. BIP62 catalogued it as a malleability source, `NULLDUMMY` became relay policy, and BIP147 made it consensus for witness version 0 [34, 20].

3.2 How the base layer closed the seams

The lineage is worth stating because it is the reason these surfaces are fossils rather than live affordances. BIP62 enumerated third-party malleability and was withdrawn in favour of a structural fix; BIP66 made DER encoding strict; BIP141 and BIP143 moved witnesses out of the txid and

introduced a digest with no FindAndDelete and no constant-one branch; BIP147 enforced the null dummy; BIP341 and BIP342 dropped `OP_CHECKMULTISIG` from Tapscript entirely in favour of `OP_CHECKSIGADD` and committed the position of `OP_CODESEPARATOR` in the digest [34, 35, 25, 21, 20, 36, 37]. Legacy outputs remain spendable, so consensus still evaluates the seams today; policy fences them. That split between consensus-live and policy-dead is exactly what the harness is built to measure (Section 4).

3.3 Introspection and dispute systems built on the same seams

BitVM reframed Bitcoin Script as a verifier of optimistic computation claims: commit, challenge, and expose one incorrect step [22]. BitVM2 turned that into a permissionless bridge design with a light-client-based pegout [24]. Binohash is the result most relevant here: it exploits FindAndDelete in legacy `OP_CHECKMULTISIG` together with proof-of-work signature grinding to obtain a collision-resistant digest of the spending transaction that is readable in Script, giving limited transaction introspection without a soft fork [23]. Binohash is a demonstration that the FindAndDelete seam is not merely a museum piece; this paper's harness is the bench on which such constructions can be measured against the actual policy envelope. Heilman's Lamport-signature construction, which reads ECDSA signature length through `OP_SIZE`, is a second example of a legacy encoding seam reused as a feature [15], as are Poelstra's Schnorr-plus-`OP_CAT` tricks [29]. Covenant proposals such as `CHECKTEMPLATEVERIFY`, `ANYPREVOUT`, and `OP_CAT` are the soft-fork path to the same expressiveness [32, 7, 16, 27]; the motifs in this paper are what an overlay can do while those proposals remain unactivated.

3.4 Overlay systems whose coordination surfaces we target

The modern isomorphisms are drawn from systems whose public coordination surfaces are richer than their on-chain footprint: Lightning channels and watchtowers [30, 26], discreet log contracts and their oracle attestations [10], Ark rounds and VTXOs [17], statechain handoffs [33], and Taproot Assets proof anchors and universes [28]. Gudgeon et al. give the systematisation of layer-two protocols that these examples sit inside [14]. In each case there is a committed core (a channel state, an outcome, a VTXO, a handoff, an asset transition) and several public handles, transcripts, or publication shapes around it.

3.5 Differential and historical testing

The observer pipeline is methodologically closest to consensus-differential systems that use executable artifacts and cross-checking harnesses as research instruments: Fluffy's multi-transaction differential fuzzing of Ethereum clients and fork-state-aware differential fuzzing of consensus implementations [38, 19]. The difference in emphasis is that we are not primarily hunting divergences between two implementations; we are using a Core oracle plus a narrow shadow executor to *stratify* known seams by era, digest behaviour, and policy outcome.

3.6 Publication policy drift

Carrier camouflage was framed in April against a world where explicit data carriers were policy-limited. Bitcoin Core 30.0 raised the default `-datacarriersize` to 100,000 bytes and admits multiple `OP_RETURN` outputs per transaction for relay and mining [4]. Explicit carriers are therefore no longer gated by default policy. The consequence for Motif III is that camouflage no longer

Window	Heights	Anchor	Status
BIP16 enforcement	173780–173830	P2SH at 173805 [1]	extracted
BIP34 buried activation	227900–227960	height-in-coinbase at 227931 [2]	extracted
Payout burst (“payout2013”)	279150–279320	January 2014 miner payout fanouts	extracted, deep slice
BIP66 boundary	363700–363760	strict DER [35]	extracted
BIP65 boundary	388360–388420	`CHECKLOCKTIMEVERIFY`	planned
CSV boundary	419300–419360	`CHECKSEQUENCEVERIFY`	planned

Table 1: Curated mainnet windows. The repository labels the 279150–279320 window “2013” after the coinbase-scan trial that found it; the blocks themselves were mined on 7–8 January 2014, which this revision corrects throughout.

buys *admission*; it buys resistance to fingerprinting and a cost profile that matches ordinary traffic. Section 13 returns to this.

4 Observability Toolchain

The repository contributes a practical research tool rather than an alternative node implementation [11]. It is a Rust workspace with a CLI (`replay`, `fuzz`, `reduce`, `replay-era`, `fetch-fixtures`, `mint-*-seam`, `summarize`, `museum`, `report`) over four layers (Figure 1).

Historical extraction. Mainnet windows are selected around activation boundaries and structurally interesting transaction families, then extracted into reproducible corpora (Table 1). The pre-SegWit tour is intended to end at CSV rather than at witness-era modular Bitcoin.

Seam-family fixtures and replay. Curated fixture families capture legacy behaviours such as FindAndDelete context splits, `SIGHASH_SINGLE` degeneracy, and `CHECKMULTISIG` dummy-element txid variation. Fixtures are minted on regtest with real funding outpoints, replayed through `testmempoolaccept`, executed by the Rust shadow, and reported with stable identifiers: a `specimen_id` is the SHA-256 of the canonical (key-sorted) JSON of the reduced testcase, so two operators who mint the same seam obtain the same identifier. Each specimen carries the derived legacy sighash digest, txid, policy outcome and reason, and quirk-family fields such as removal counts or dummy-element size.

Historical carrier and sidecar benches. The toolchain also supports ordinary-looking carrier families in the 279150–279320 window, especially payout-shaped topologies, so that overlay publication ideas can be discussed in terms of historical cover rather than only abstract commitment formats (Section 6.3).

Modern live graft harnesses. TradeLayer on Litecoin testnet (LTCTEST) is used as a controllable live environment for procedural-token flows, oracle relays, BitVM-style caches, challenges, resolutions and payout edges, and Bitcoin testnet4 is used for the reserve-bound referee layer (Section 9). This gives a live isomorphism layer without claiming that the exact historical Bitcoin transaction forms survive unchanged.

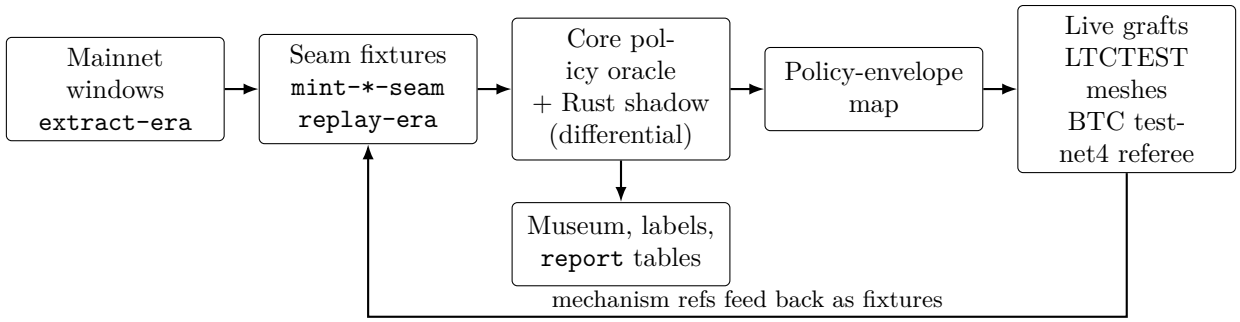


Figure 1: Observer pipeline: extraction, seam minting and replay, differential classification, policy-envelope labelling, and live grafting. Live mechanism references are written back as fixtures so the next replay can test them.

Oracle scope. The Core oracle is `testmempoolaccept` on a current node. Epoch flags from the consensus profile are an organisational lens for stratified replay, not proof of full historical block-validation semantics at height H. The Rust shadow is intentionally narrow (one input, a P2WPKH-style opcode slice, deterministic signature hooks); it exists to classify, not to validate.

5 Three Design Motifs

5.1 Motif I: Transcript Multiplicity

One broad spend family can admit multiple effective signing transcripts. In the legacy seam bench, FindAndDelete subset variation and `SIGHASH_SINGLE` degeneracy both show that small structural changes can alter the effective digest, or collapse it, without changing the broader transaction family. The important abstraction is not the historical script quirk; it is the existence of a transcript-selection manifold with two distinct regions: a benign region in which several transcripts alias to one digest class, and a hazardous region in which the digest collapses to a verifier-known constant. Binohash lives in the first region and makes it load-bearing [23]; the constant-one branch lives in the second and must be fenced.

Modern overlays want this because BitVM- and DLC-adjacent systems depend on alternative off-chain transcripts, branch packages, reveal paths and dispute narratives. A designer may want the same higher-level state machine while preserving freedom over transcript selection, packaging, or attestation form: a compact relay form and a full evidence form of one proof, or a success package and a timeout package over one claim.

5.2 Motif II: Identifier Bifurcation

A public identifier can vary while the committed core stays fixed. The dummy-element seam is the canonical example: txids move while the legacy sighash digest is invariant. The legacy path is policy-rejected today, but it exposes a clean separation between *semantic commitment* and *public handle*. Classical Bitcoin work encountered this family as a malleability problem rather than as a design affordance, which makes the historical contrast useful [3, 8].

Practical systems need visible session identifiers, anchor labels, namespace handles, rendezvous tags and publication references even when the semantic core is fixed elsewhere. The motif says: derive every such handle from the core commitment plus a role and a label, so that rotation is free and verification is unchanged.

5.3 Motif III: Carrier Camouflage

Overlay commitments should be able to hide inside ordinary-looking transaction topology rather than obviously exotic publication surfaces. The historical carrier bench shows that strong candidate covers already existed in ordinary payout and redistribution topologies: high-fanout sprays, exact-value batch redistributions, and large aggregators (Table 5). The point is not simply that big transactions existed. The point is that ordinary historical cover is a model for how modern overlays should think about publication shape, ambient noise and carrier selection. This is adjacent in spirit to transaction-shape privacy work such as PayJoin, where the operational question is how to inhabit ordinary-looking transaction space without leaking a distinctive surface [9, 13].

6 Measured Seam Surfaces

6.1 Regtest-funded seams

Three seam manifests are minted with real regtest prevouts so that Core returns policy-specific reasons instead of `missing-inputs`. The `SIGHASH_SINGLE` manifest has four specimens (two bug forms at a pre-BIP16 representative height, two controls at a post-BIP34 height, each with and without `ANYONECANPAY`); the DUMMYGRIND manifest has three (empty, one-byte zero, and 32-byte dummies); the FindAndDelete manifest has three signature subsets at the BIP16 boundary height. In the funded runs the `aa` and `aaaa` subsets share one digest while `aabb` produces a distinct digest under the same policy rejection reason; the two `SIGHASH_SINGLE` bug specimens collapse to the constant-one digest while the controls diverge; and the non-empty dummy variants change the txid while preserving the sighash digest and are rejected under a `NULLDUMMY`-style reason. The companion note *Jurassic Surfaces* renders those runs as `report -format latex` tables directly from the artifact directories.

6.2 Analytic reproduction without a node

The cryptographic content of each seam is a pure function of legacy serialisation and the original `SignatureHash`, so it can be reproduced without a node. `scripts/seam_digests_standalone.py` ports `FindAndDelete` and `SignatureHash` verbatim (including the `uint256::ONE` branches), builds deterministic specimens from SHA-256-derived placeholder keys and prevouts, and emits the tables below. No signature is verified; the claim is about digest classes and txids, which do not depend on key validity. Digests are rendered byte-reversed, as `uint256::GetHex()` does, so the constant-one digest prints as $0^{63}1$, matching the tripwire value used in Section 9.

Seam A: FindAndDelete context split. The redeemScript is `<aa> OP_DROP <bb> OP_DROP OP_2 <pkA> <pkB> <pkC> OP_3 OP_CHECKMULTISIG`: a script that contains the one-byte "signature" pushes it will be fed. That is the only kind of script for which legacy FindAndDelete changes the scriptCode, and it is a crafted, legacy-only degree of freedom, which is the point. Table 2 shows four signature multisets. Removing `aa` once or twice yields the same scriptCode (deletion is idempotent per push class), so `subset_aa` and `subset_aaaa` alias to one digest; `subset_aabb` deletes a second push class and splits; the baseline with a signature absent from the script is a third class. Every row has a distinct txid because the `scriptSig` differs; the digest classes are what the motif is about.

Variant	Signatures	Removed	scriptCode (SHA-256)	Legacy sighash digest
baseline_cc	cc	0	504221f54f4b3561...	b5dd22b737b575b1...
subset_aa	aa	1	784d55748eaffe81...	fdd8e6839fef0aeb...
subset_aaaa	aa,aa	1	784d55748eaffe81...	fdd8e6839fef0aeb...
subset_aabb	aa,bb	2	0efad003bc30a61a...	6fb01edab44922f3...

Table 2: Seam A, analytic reproduction (SIGHASH_ALL, input 0). Three digest classes over one spend family; the two aa subsets alias.

Variant	Type	In/Out	Legacy sighash digest
sighash_single_bug	0x03	2/1	$0^{63}1$ (constant one)
sighash_single_bug_anyonecanpay	0x83	2/1	$0^{63}1$ (constant one)
sighash_single_control	0x03	2/2	7f39cc0c32cd3c5a...
sighash_single_control_anyonecanpay	0x83	2/2	f0d98ef6e3f1547f...

Table 3: Seam B, analytic reproduction (input index 1). Both bug forms collapse; both controls diverge.

Seam B: SIGHASH_SINGLE degeneracy. A two-input transaction is signed at input index 1 with SIGHASH_SINGLE. With one output the index is out of range and the digest is the constant one, with or without ANYONECANPAY; with two outputs the controls produce ordinary, distinct digests (Table 3). A signature over the constant is valid for every transaction that reaches the same branch, which is why an overlay must treat $0^{63}1$ as a rejection tripwire and never as an identifier.

Seam C: DUMMYGRIND. A 1-of-1 P2SH multisig spend varies only the dummy element: the empty vector (00), a one-byte zero push (01 00), and a 32-byte zero push. The legacy digest blanks every scriptSig and substitutes the redeemScript, so all three share one digest while the serialised transaction, and therefore the txid, differs (Table 4). Note that CScript encodes a one-byte vector as 01 xx and never as OP_1..OP_16, which is why the empty and one-byte-zero dummies are distinct bytes with the same numeric value. The 32-byte variant shows the shape of the affordance: a 256-bit field in the unlocking data that moves the public identifier and nothing else. Relay policy rejects every non-empty dummy under NULLDUMMY; witness version 0 rejects it by consensus [20].

6.3 Historical carrier specimens

Table 5 lists the four specimens the carrier bench treats as covers, with shapes re-verified against a public block explorer for this revision. The 202-output spray is the strongest broad cover: 156 outputs of 0.01 BTC and 18 of 0.02 BTC give it the repeated low-denomination rhythm of a pool payout, although one change output still holds 68% of the value. The 101-output transaction is an exact-value redistribution (50 outputs of 1 BTC and 50 of 0.5 BTC) with no dominant output. The 27-input aggregator is the right carrier for identifier churn because consolidation dominates the visible shape. The mixed P2SH specimen is small and is kept as a migration-era control rather than as a carrier.

Variant	Dummy push	txid	Legacy sighash digest
`dummygrind_empty`	`00`	`645c0d4006a4cec0...`	`f5f83c1dd45ccc26...`
`dummygrind_zero`	`0100`	`53c89506fc9fa99f...`	`f5f83c1dd45ccc26...`
`dummygrind_32`	`20` `00`×32	`ceaa66e4a25c06f7...`	`f5f83c1dd45ccc26...`

Table 4: Seam C, analytic reproduction. Three txids, one digest.

Specimen	txid prefix	Height (date)	Shape	Largest output share	Size (B)
payout spray	`a321554f...`	279209 (2014-01-08)	1 in, 202 P2PKH out; 156×0.01, 18×0.02 BTC	0.682	7027
exact-value batch	`a9ead3b6...`	279234 (2014-01-08)	1 in, 101 P2PKH out; 50×1.0, 50×0.5 BTC	0.250	3625
aggregator	`ac91df69...`	279213 (2014-01-08)	27 in, 19 P2PKH out	0.064	5035
P2SH mix	`b2054298...`	279228 (2014-01-08)	1 in, 2 P2SH + 1 P2PKH out	0.882	501

Table 5: Carrier specimens from the 279150–279320 window, shapes verified October 2026.

7 Controlled Freedom: A Minimal Formalisation

The motifs are safe only when the freedom they introduce sits above a fixed commitment rather than beside it. The following is deliberately minimal; it is what the prototypes actually enforce, not a security proof.

Let S be an overlay semantic state (a lease, an oracle outcome, a VTXO transition, a reserve route) and let $c = H(S)$ be its canonical commitment under a collision-resistant hash of a canonical serialisation. Define:

- $\mathcal{T}(c)$, the set of *transcripts* a verifier accepts as evidence for c (compact and full proof packages, retry-equivalent success proofs, branch-specific challenge packages). Transcript multiplicity is the design decision $|\mathcal{T}(c)| > 1$.
- $\mathcal{N}(c)$, the set of *public handles* that resolve to c (rendezvous labels, proof anchors, round ids, alert handles), with a resolution map $\nu : \mathcal{N}(c) \to \{c\}$. Identifier bifurcation is the design decision that ν is many-to-one and that rotating within $\mathcal{N}(c)$ is a no-op on S.
- $\mathcal{C}$, a family of *carrier* transaction shapes with a shape statistic σ (input and output counts, output-type histogram, repeated amounts, largest-output share). Carrier camouflage is the design decision to publish under a carrier whose σ is drawn from an ambient distribution of ordinary traffic.

Four properties separate a usable isomorphism from a re-imported bug:

P1, semantic binding. Every $t \in \mathcal{T}(c)$ commits to c: if the verifier accepts t for c' then $c' = c$. Aliases are aliases *of a commitment*, never free-floating evidence.

P2, non-collapse. No transcript digest in $\mathcal{T}(c)$, and no candidate the verifier is willing to promote to an identifier, equals a verifier-known constant. Concretely, $0^{63}1$ is rejected before it can become a funding, claim or challenge id. The constant-one branch is the historical instance; the property generalises to any degenerate digest.

P3, authority separation. Admitting a transcript, handle or carrier never substitutes for spend authority. A route transcript candidate is not a spend authorisation; admissibility additionally requires live reserve evidence and an exact payout-vector commitment (Section 9).

P4, rotation invariance. For all $n, n' \in \mathcal{N}(c)$ the verifier's decision is identical. Handles index; they do not authorise.

The contrast with base-layer malleability is now precise. Pre-BIP62 malleability was multiplicity over the *txid*, in a system where the txid was also the commitment that downstream transactions spent from, so there was no stable c above the moving identifier. The base-layer fix made the digest and the txid canonical. Overlays can re-admit multiplicity because their c is not the txid: the seams have the same manifold shape, bound at a different point. That is what "isomorphic" means in the title.

8 Policy-Envelope Mapping as Method

The methodological lens is a policy-envelope map rather than a fourth coequal protocol claim. Each measured surface falls into one of three buckets:

1. **Replay-only fossils**: historically informative, but no longer clean live surfaces on Bitcoin today (FindAndDelete subsets, the constant-one branch, non-null dummies).
2. **Ordinary historical cover**: surfaces that were already ordinary in their native era (the carrier specimens).
3. **Clean modern isomorphisms**: live contemporary grafts that preserve the same abstract motif without relying on the old transaction form.

This classification prevents a common failure mode of "quirk mining": confusing a historical mechanism with a deployable modern protocol. Many legacy mechanisms should remain museum specimens while their abstract shape informs overlay design. The generated crosswalk (`artifacts/grants/policy_envelope_mapping.md`) assigns every fixture, carrier and live flow to a bucket together with its relay compatibility and execution role. Table 6 gives the revised status by motif.

9 From Motifs to Mechanism Handles

The April draft stopped at Litecoin flow graphs. The UTXORef prototype layer [12] has since turned the motifs into deterministic handles that existing Lightning, Taproot Assets, Ark and proof-execution prototypes carry in their evidence bundles, and bound those handles to confirmed Bitcoin testnet4 transactions through two off-chain referees.

Motif	Historical surface	Modern isomorphism	Status (October 2026)
Transcript multiplicity	FindAndDelete context splits; `SIGHASH_SINGLE` degeneracy	LTCTEST relay aliases; fused router-dispute graft; UTXORef transcript switchboard; testnet4 route referee accepts `accepted_transcript_alias_compact`	Live as relay and dispute prelude; bound to a confirmed Bitcoin testnet4 reserve
Identifier bifurcation	Dummy-element txid variation	LTCTEST namespace relays; UTXORef namespace relay matrix; testnet4 route referee accepts `accepted_identifier_namespace_rotated`	Live as namespace motif; bound to a confirmed Bitcoin testnet4 reserve
Carrier camouflage	January 2014 payout and redistribution topologies	LTCTEST settlement and dispute graphs; UTXORef carrier shadow routes; testnet4 watchtower cadence referee with sweep, payout-batch and rebalance profiles	Historically grounded; referee-level on testnet4; on-chain Bitcoin carrier is still an explicit `OP_RETURN`; funded oracle-sidecar replay pending

Table 6: Motif status, revised.

9.1 Deterministic handle derivation

Let $h = H(\text{canonical semantic statement})$ be the `semanticStateHash` of one overlay claim. The mechanism catalogue derives, for a target protocol p:

$$\begin{aligned}
\texttt{transcriptDigest}_k &= \mathrm{SHA256}(h \,\|\, \texttt{"|"} \,\|\, \texttt{packageClass}_k),\\
\texttt{transcriptSwitchboardId} &= H(\{h, p, [\texttt{transcriptDigest}_k]_k\}),\\
\texttt{publicHandleId}_j &= H(\{h, p, \texttt{handleRole}_j, \texttt{label}_j, j\}),\\
\texttt{carrierCommitmentId}_m &= H(\{h, p, \texttt{carrierRouteId}_m, \texttt{ordinaryTopology}_m, m\}),\\
\texttt{rejectionTripwireDigest} &= 0^{63}1.
\end{aligned}$$

Every handle is a function of h (P1 and P4 by construction), the switchboard carries one explicitly rejected variant whose digest is the constant one (P2 as a catalogue entry rather than a comment), and none of the handles is consumed by any spend path (P3). Table 7 lists where the handles are bound.

9.2 Quirk-indexed route referee on Bitcoin testnet4

The route referee binds a quirk route candidate to live reserve evidence before a BitVM route or spend proposal is treated as admissible application state. Its rule is the operational form of P3:

$$\texttt{route transcript candidate} \neq \texttt{spend authority.}$$

A claim core carries the motif, a public handle, `semanticStateHash`, `routeTranscriptHash`, `withdrawalRootHex`, `finalOutputVectorHash`, `liveTraceHash`, the `reserveOutpoint`, and a challenge window. The referee admits a claim only if the route transcript hash is one of the candidates generated from the quirk manifest, the live trace hash matches the current import of the

Target	Prototype	Fields bound into evidence
Lightning	liquidity lease (PTLC/adaptor success, timeout challenge)	`jurassicMechanismRefId`, `transcriptSwitchboardId`, watchtower `publicHandleId`, sweep or splice `carrierCommitmentId` in offer, success and challenge evidence
Taproot Assets	RFQ stablecoin quote	`proofAnchorHandleId`, `proofCarrierCommitmentId`, `proofTranscriptDigest` in quote, settlement and challenge evidence
Ark	liquidity graft	`roundClaimHandleId`, `roundCarrierCommitmentId`, `roundTranscriptDigest` in quote, settlement, challenge and bundle cores
Shinigami-style	proof publication scaffold	verifier handle, proof blob reference, execution trace carrier; scaffold only, no external verifier claimed [18]

Table 7: Mechanism handles bound into UTXORef prototype evidence.

reserve vault, the withdrawal root matches the UTXORef withdrawal queue, the final output vector hash matches the expected payout vector, and the reserve is confirmed and CSV-safe. Four demo outcomes exercise the motifs: a compact transcript alias is accepted, a namespace-rotated handle over the same grant state is accepted, a mutated withdrawal root is rejected with reason `withdrawal_root`, and an unknown route transcript is rejected with reason `route_transcript_candidate`.

9.3 Camouflaged watchtower cadence referee

The second stage asks whether watchers are publishing timely, bound observations of an admitted route while keeping ordinary-looking publication profiles. A cadence claim binds the admitted `routeClaimHash`, the reserve outpoint, the live trace hash, a watchtower epoch, the expected cadence in blocks, a publication height, a carrier profile, a publication handle, and

$$\texttt{semanticAlertHash} = H(\texttt{routeClaimHash}, \texttt{reserveOutpoint}, \texttt{watchtowerEpoch}),$$

which is independent of the carrier handle, so the handle may rotate between `wallet_sweep_checkpoint`, `payout_batch_checkpoint` and `rebalance_checkpoint` profiles while the alert core stays fixed. The demo accepts a sweep-like and a payout-batch checkpoint, rejects a stale checkpoint with reason `cadence_freshness`, and rejects an alert handle that points at the wrong route claim with reason `publication_handle_binding`. Figure 2 shows the two-stage pattern.

9.4 Live Bitcoin anchors and what they do and do not show

Table 8 records the Bitcoin testnet4 transactions the referees import, with confirmation status checked for this revision. Three honest qualifications follow. First, the grant transaction is a bare `OP_RETURN` with a 55-byte TradeLayer payload and no value output: it is exactly the explicit-marker shape that Motif III argues against, so the camouflage demonstrated on Bitcoin lives in the referee's carrier *profiles*, not yet in the on-chain footprint. Second, the reserve outpoint the July referee runs bind was spent four days later through a Taproot script path; an admitted claim is therefore a snapshot at the live import's height, and a fresh import yields a different `liveTraceHash` and a

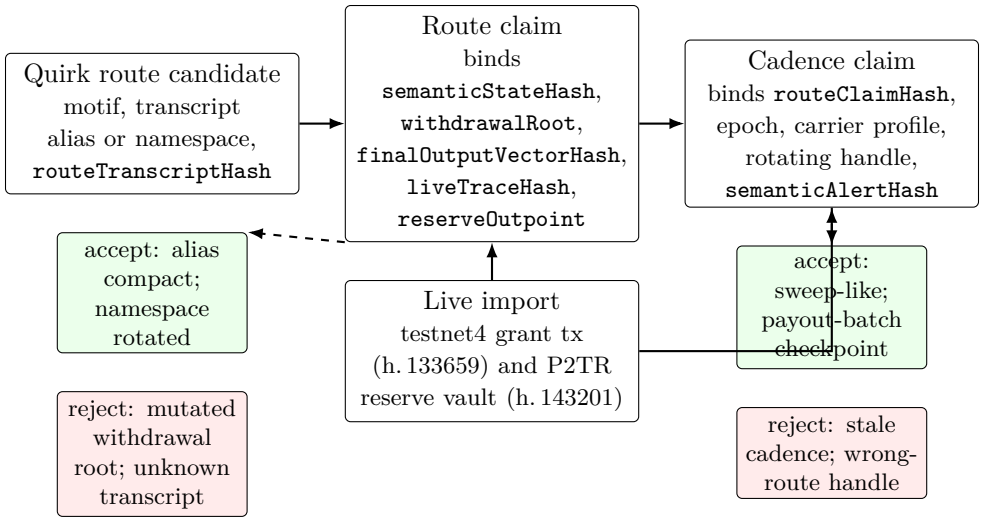


Figure 2: Two-stage referee pattern: a quirk route candidate becomes a reserve-bound route claim, and only an admitted route claim can anchor a camouflaged watchtower cadence claim. Both referees are off-chain objects with challenge evidence; neither changes the vault spend path.

Role	txid prefix	Confirmed	Shape
`tlBTC` grant (TradeLayer tx11)	`7dec37be...`	133659, 2026-05-04	1 P2WPKH input; 1 `OP_RETURN` output, 55-byte payload, 0 sats; 226 B
UTXORef reserve vault	`93f953df...`	143201, 2026-07-06	1 P2WPKH input; output 0 P2TR 20,000 sats (the reserve), output 1 P2WPKH change
Reserve spend	`63f71446...`	143690, 2026-07-10	spends `93f953df...:0` with a 4-element witness (script path); 1 P2WPKH output of 19,000 sats

Table 8: Bitcoin testnet4 anchors imported by the referees, as of October 8, 2026.

different admission. Third, both referees are off-chain: they decide what the application treats as admissible evidence, and the BitVM-style dispute assumption is unchanged.

10 Binohash-Level Primitives

The mechanism handles of Section 9 package the motifs for existing prototypes. This section goes one level deeper and asks the Binohash question directly [23]: can each seam carry a *named primitive* with a capability that plain Script lacks, a precise precondition, and an honest cost or policy line? We give four, demonstrated by deterministic, node-free invariants in `scripts/binohash_level_tools.py` (Primitives 1–3) and `scripts/covenant_limits.py` (Primitive 4), and we then record in Section 10.5 three escalations that do *not* work and why. As with the seam reproduction, no signature is verified; the demonstration is a digest-level identity or collision bound, which does not depend on key validity. Every invariant below is reported `PASS` by its script, and each primitive is annotated with the admission properties (P1–P4) of Section 7 that it enforces.

10.1 Primitive 1: Append-Only Aggregatable Attestation

The constructive half of the `SIGHASH_SINGLE` seam is not the collapse but `SIGHASH_SINGLE|ANYONECANPAY`. Under that flag a signer's digest is a function of only its own input, its own output, and its index: ANYONECANPAY drops every other input from the hash, and SINGLE nulls the earlier outputs and truncates away the later ones. Independently produced "slots" therefore compose into one transaction without a re-signing round. This is the mechanism behind assurance contracts [6]; we lift it to threshold oracle attestation, where each oracle signs one (funding slot, payout slot) pair binding a shared event hash and an aggregator assembles any sufficient subset.

The script demonstrates four facts about a three-oracle example. First, the checked signer's digest is unchanged when every other slot's input and output content is mutated (`digest_independent_of_other_sl` content independence is what lets slots be produced in isolation. Second, appending a later oracle slot leaves all earlier signers' digests fixed (`append_later_slot_preserves_earlier_digests`): aggregation is append-only. Third, removing an *earlier* slot changes a signer's digest (`remove_earlier_slot_chang` because SINGLE binds the signer to its output index; an aggregator may append and may rewrite other slots' contents, but it may not drop or reorder earlier slots. Fourth, a slot whose signed index has no output collapses to the constant one and is refused (`constant_one_collapse_detected_and_refused`), which is property P2 enforced at the point of use. The primitive enforces P1 (each payout output carries the event hash), P2, and P3 (a slot is evidence of one attestation, not spend authority over the aggregate). The precondition, stated honestly, is that signers agree on `nVersion` and `nLockTime` and that composition preserves index order.

10.2 Primitive 2: State-Bound Handle Rotation

The dummy-element seam moves the txid while the sighash is fixed, but the raw seam gives no way for a third party to tell an authentic rotation from an unrelated malleation. The primitive supplies the missing derivation:

$$\texttt{handle} = H(\texttt{stateHash} \,\|\, \texttt{label} \,\|\, \texttt{epoch}),$$

so an indexer verifies a rotation by one recomputation (property P4) and the handle is bound to the committed state (property P1) yet is never read by any spend path (property P3). The script shows two carriers for the same rotation. The *legacy* carrier places the handle in the unused `CHECKMULTISIG` dummy: three epochs give three distinct txids over one fixed sighash digest (`legacy_txid_moves_digest_fixed`), the historic mechanism, which is policy-dead under `NULLDUMMY` and consensus-invalid for witness v0 [20]. The *witness-era* carrier is the deployable one: the committed state is anchored once in a byte-identical `OP_RETURN` output while the rotating handle rides a second `OP_RETURN` output, so the anchor is stable across epochs (`witness_anchor_fixed_across_rotation`) and the whole shape is relay-standard under Bitcoin Core 30.0 [4]. All six handles recompute from the state (`all_handles_verify`). The overlay use is a rotating rendezvous, watchtower-alert, or proof-anchor handle over a fixed channel or reserve state, without re-anchoring the state. The honesty line from Section 13 carries over: rotation defeats third-party indexing, not linkage by anyone who holds `stateHash`.

10.3 Primitive 3: A Binohash Parameter Bench

The third primitive does not propose a competing scheme; it is a cost accountant for Binohash itself, so a protocol designer can choose parameters against a real line rather than a feeling [23]. The script first reproduces the concrete degree of freedom: a redeemScript that embeds the pushes it is fed, so after FindAndDelete the signed scriptCode is a function of chosen data

W_1	W_2	Resistance (bits)	Honest work (bits)	Wall time (s)	Grind cost (USD)
32	32	64	34.6	0.00	0.00
42	42	84	44.6	2.67	0.37
48	48	96	50.6	170.65	23.70
56	56	112	58.6	43687.56	6067.72
64	64	128	66.6	11184014.24	1553335.31

Table 9: Binohash grinding bench, generated by `scripts/binohash_level_tools.py`. Wall time assumes a modest rented fleet; cost assumes $0.50 per GPU-hour at 10^{10} hashes per GPU-second. Both are configurable; the point is the shape of the knee, not the exact dollar figure.

(`scriptcode_is_data_dependent`). That is the surface Binohash grinds against. It then tabulates Linus' two-round parameters: with weights W_1, W_2 the digest has $W_1 + W_2$ bits of collision resistance at roughly $W_2 + 2.6$ bits of honest work. Table 9 gives the line at an illustrative, configurable rented-GPU price; the $W_1 = W_2 = 42$ row lands at 84-bit resistance for well under the order of a hundred dollars of grinding, matching the regime Linus reports, while 56-bit weights are already economically out of reach. The value of the bench is that the knee of that curve is where a BitVM-bridge designer must sit: enough resistance to bind an introspected transaction, little enough work to grind at spend time.

10.4 Primitive 4: SINGLE-Authored, ALL-Frozen Settlement

The fourth primitive is the sound construction that the broken covenant ideas of Section 10.5 were reaching for, and it needs no Binohash, no grinding, and no soft fork. It rests on a duality between two sighash modes that `scripts/covenant_limits.py` verifies node-free. A `SIGHASH_SINGLE|ANYONECANPAY` signature at input index j pins exactly output j and is invariant to every other input and to outputs appended after j: it gives a party its own output row but fences nothing globally. The dual mode `SIGHASH_ALL|ANYONECANPAY` fences the entire output set, including appends, but forces one shared template with no per-party rows. Neither alone gives per-contributor rows *and* a closed template.

Compose them on one transaction, using different sighash types on different inputs (which Bitcoin permits). Each party authors its own output row with a SINGLE|ACP signature at its input, independently and append-only, with no coordination. One *closer* input then freezes the assembled output vector with a single ALL|ACP signature. The result has both properties at once: the script checks (`FACT 5`, all passing) show that after the freeze, tampering with any party's row breaks both that party's row-pin and the closer's freeze simultaneously, an untouched party's row-pin still verifies, and no output can join the table after the freeze. The primitive keeps P1 (every row-pin and the freeze commit to transaction fields) and P3 (a row-pin authorises one row, never the aggregate); P2 and P4 do not apply.

The delta is precise. Over the assurance-contract pattern, which is ALL|ACP over one shared template [6], it adds independent per-contributor rows. Over Primitive 1, which is SINGLE|ACP attestation with no global fence, it adds the closing freeze. The honest caveat is value conservation: legacy sighash does not commit input amounts, so a legacy deployment leaves conservation to inspection, while the BIP143 witness-v0 sighash commits the input amount and closes the hole [21]. The overlay use is a TradeLayer or DLC batch settlement in which each counterparty owns its payout row and a coordinator freezes the batch before broadcast, with no pre-signed payout matrix and no new opcode.

10.5 Limits: Three Binohash-Covenant Mirages

Before claiming the primitives it is worth recording what does *not* work, because the failures are sharp and reusable. A structured design exploration surfaced three natural escalations that each try to turn a seam into a richer, consensus-enforced covenant. All three break, and `scripts/covenant_limits.py` reproduces the mechanism behind each break node-free. Two rocks recur.

The menu covenant (R1: the deleted push must be a verifying signature). The idea embeds m `OP_DROP`-guarded "class" pushes in a legacy `CHECKMULTISIG` redeemScript so that feeding a subset deletes exactly those pushes and selects one of 2^m digest classes, a one-of-k template selector in $O(1)$ data. The digest-class partition is genuine and reproduced (it is power-set injective and delete-idempotent, the same law as Seam A). It is nonetheless not selectable on-chain: legacy FindAndDelete removes only a push that is a *signature*, which must ECDSA-verify against a committed key over the post-deletion digest. A short class constant is not a valid signature, and a fixed signature-shaped constant validates exactly one message, which is the very digest its deletion changes, a 2^{256} grind. Fixing the deleted value removes precisely the free variable that Binohash grinds, so the covenant cannot be ground into existence; the script decays to a plain multisig with decorative pushes.

The script-enforced resolution anchor (R2: the core commits its own prevout). The idea bakes a Binohash constant K into a UTXO so that spending it forces the spend to resolve a public handle to a committed state, with no indexer. But every legacy sighash, and therefore any Binohash core, serialises the spending input's own prevout; the script confirms the core changes when the prevout changes. So K must commit the funding txid, which depends on K through the redeemScript and scriptPubKey: a fixed point of a collision-resistant hash, of the scheme's own security order to construct. And because Binohash pins the *whole* transaction, it cannot be projected onto one output, so there is no variant that keeps a sub-region free; the only retreat, checking K in the light client, collapses the construction into ordinary client-side validation.

The dead-man absence attestation (R3: the constant-one is reusable and evadable). The idea reads the `SIGHASH_SINGLE` constant-one as a self-announcing proof that a reserve output at a fixed index is absent. The constant is verifier-known and identical across unrelated transactions (shown), so a signature over it is reusable, transaction-independent spend authority, which violates authority separation (P3) rather than attesting a timed state. Absence is also evadable: a dishonest custodian keeps a dust decoy at the index, the digest becomes ordinary, and no tripwire fires (shown). The real check is content inspection of the output, which uses none of the degeneracy.

The common root is that a whole-transaction Binohash pin commits the spending input's own prevout and the entire transaction, which is why the sound way to get per-row ownership plus a closed template is the mixed-sighash construction of Primitive 4, not a baked-constant covenant.

10.6 What "live" means here

These are demonstrations, not broadcasts. Primitive 1's composability, Primitive 2's rotation, and Primitive 4's fence duality are exact digest identities that hold for any key, Primitive 3's bound is Linus' analysis reproduced against a cost model, and the three negative results of Section 10.5 are reproduced the same way; all run offline and deterministically. The nearest deployed relatives are the assurance-contract pattern for Primitives 1 and 4 [6], the oracle-sidecar and watchtower-beacon LTCTEST meshes already in the repository for Primitives 1 and 2, and BitVM bridges for Primitive

3 [24]. We deliberately stop short of broadcasting funded Bitcoin transactions for these primitives: that step is the funded oracle-sidecar materialisation listed as future work in Section 13, and it is gated on the same prevout funding as the replay bench.

11 BitVM Prototype Applications

The live prototype layer is deliberately framed as *BitVM-style* application work, not as a claim that the legacy quirks implement BitVM [22].

11.1 Motif wrapper versus vanilla BitVM

Vanilla BitVM is a dispute-oriented execution pattern: participants commit to a computation or state transition, and if one party publishes an incorrect claim, the protocol exposes it through a challenge path. The Jurassic grafts do not change that security model and do not require changing Bitcoin Core. They add a deterministic application wrapper with three jobs. Transcript multiplicity lets one committed state carry compact and full transcript packages, so a dispute graph can choose between short relay forms, full evidence forms, or challenge-specific witness narratives. Identifier bifurcation separates public handles from the semantic core, so an oracle event, a Taproot Assets proof anchor, a watchtower alert, a router branch, or a proof package can all point back to one commitment without overloading one identifier. Carrier camouflage treats publication as a placement problem and attaches carrier hints such as payout batches, rebalances, sweeps, wallet batches, or distribution shadows.

The resulting architecture is not "better BitVM consensus." It is a product-facing packaging layer. A vanilla flow provides the cache, challenge, resolution and payout edges; the wrapper adds transcript aliases, namespace handles, carrier hints, procedural-token state and observer artifacts that make those edges easier to route, index, audit and integrate. The cost is that the application layer acquires explicit watcher, indexing, naming and artifact-management requirements. Those are acceptable in the prototypes because they are deterministic and visible in the flow graph; they are not a simplification of BitVM's dispute assumptions.

11.2 Procedural-token flow graphs

The most concrete prototype result is that the motifs integrate into procedural-token flow graphs. The fused LTCTEST experiment (`short_epoch_router_dispute`) binds four relay preludes to one contract reference before settlement (two transcript-alias attestations tagged `aa` and `aaaa`, two namespace attestations whose blob references end in `namespace-zero` and `namespace-32`, a deliberate echo of Seams A and C), routes bucket loss through `pnl_sweep`, opens two excess `bitvm_cache` branches, and sends one through `bitvm_challenge` to `bitvm_resolve(reject)` and `bitvm_payout` while the other resolves `uphold` and stays blocked (Figure 3). One run therefore exhibits transcript and namespace preludes, a short-epoch routing phase, cache opening, challenge and resolution, and both released and blocked terminal outcomes. Table 10 summarises every prototype surface and its current status.

11.3 Transcript selector, anchor search, carrier-style routing

Transcript multiplicity suggests a search bench for alternative dispute transcripts, branch packaging and attestation forms; Binohash is the relevant comparison point because it makes transaction introspection itself a programmable input to BitVM-adjacent designs [23]. Identifier bifurcation

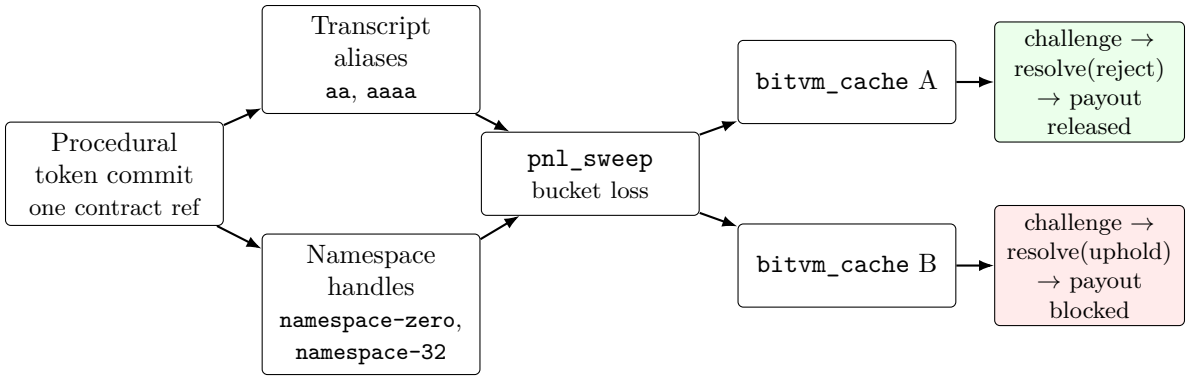


Figure 3: Fused LTCTEST router-dispute flow. Relay preludes for Motifs I and II sit on the same contract reference as the challengeable settlement graph.

suggests search surfaces for public handles: session ids, anchor names, rendezvous labels, publication namespaces. Carrier camouflage suggests that publication and settlement need not be isolated `OP_RETURN`-style acts but can be embedded in ordinary-looking payout and redistribution shapes, with one route proceeding to release while another is challenged and blocked.

12 Applications Beyond BitVM

The same three motifs admit non-BitVM application meshes, and the repository includes live LTCTEST proofs for several.

Oracle sidecars. The *oracle sidecar mesh* binds one semantic sidecar state to two transcript aliases ("fast" and "full" attestations), two public namespace handles, and two payout-shaped carrier hints modelled on the spray and exact-value specimens of Table 5. The Bitcoin replay bench for the same sidecars is where P2 first appeared operationally: the synthetic casebook includes two constant-one publications that the hazard filter rejects by construction.

Watchtower beacons. The *watchtower beacon mesh* expresses one alert state through compact and full proof transcripts with a rotating public alert handle and rebalance- or sweep-shaped carrier hints. The testnet4 cadence referee is the reserve-bound continuation of this mesh. Pisa's accountable-watchtower model is the natural comparison: our contribution is the packaging and publication surface, not the accountability mechanism [26].

Statechain handoffs. The *statechain handoff mesh* treats an ownership transition as a fixed core with acknowledgement and finalisation transcripts, rotated checkpoint handles, and ordinary checkpoint publication shapes [33].

Taproot Assets anchors. The *Taproot Assets anchor mesh* splits one asset transition into compact and full proof packages, rotates asset-id and universe-anchor handles, and publishes under wallet-batch or distribution-shadow hints [28]; the UTXORef RFQ prototype carries the corresponding handle ids.

Prototype surface	Motif emphasis	Status	Notes
Transcript alias relay	I	passed	two accepted relay transcripts over one committed state (LTCTEST)
Identifier namespace relay	II	passed	stable semantic state with rotated public labels (LTCTEST)
Hybrid router-dispute flow	III + dispute routing	passed	released and blocked terminal branches in one run (LTCTEST)
Oracle sidecar mesh	I, II, III	passed	alias, namespace and payout-shaped carrier hints on one sidecar state
Taproot Assets anchor mesh	I, II, III	registered	compact/full proof transcripts, rotated asset and universe handles, wallet-batch hints; live run ready
Watchtower beacon mesh	I, II, III	passed	compact/full alert proofs with rotating public handles
Statechain handoff mesh	I, II, III	passed	alternate handoff transcripts plus rotated checkpoint handles
UTXORef mechanism refs	I, II, III	deterministic tests	handles bound into Lightning, Taproot Assets, Ark and scaffold evidence
Quirk-indexed route referee	I, II	live import	bound to confirmed testnet4 grant and reserve; 2 accepted, 2 rejected cases
Camouflaged watchtower cadence	III	live import	2 accepted profiles, 2 rejected cases, over an admitted route claim
Bitcoin oracle-sidecar replay	III	pending	synthetic replay stops at `missing-inputs`; needs funded prevouts

Table 10: Prototype outcomes, revised. "Passed" rows have dated LTCTEST artifacts with txids; "live import" rows have confirmed testnet4 anchors.

Lightning deposits into tokenised BTC. The LN-to-`tlBTC` path that produced the testnet4 grant is itself a small instance of all three motifs: the deposit carries a wallet invoice receipt, a bridge receipt and a grant transcript that all bind one payment hash; the invoice id, request id and property id are distinct handles over that hash; and the on-chain action is an ordinary managed-token grant rather than a bespoke transaction shape (with the qualification of Section 9 that the grant is an explicit `OP_RETURN`).

13 Threat Model and Limitations

Re-imported malleability. If an overlay keys state on a transcript digest rather than on c, aliases become double-count or replay vectors: a retry-equivalent success proof would be credited twice. P1 and application-level idempotence on $\mathcal{T}(c)$ are the mitigations, and the prototypes mark retry-equivalent packages explicitly so that settlement logic can collapse them.

Collapse hazards beyond the constant one. P2 names $0^{63}1$ because that is the historical instance, but any digest that is independent of the claimed state is equally dangerous. The tripwire should be a predicate over degenerate inputs (out-of-range indices, empty output sets, zero-length transcripts), not a single blacklisted value.

Handle rotation is not privacy against the verifier. Rotation defeats naive indexing by third parties, but anyone holding h can link every handle in $\mathcal{N}(c)$, by construction. The motif is a namespace property; it is not an unlinkability guarantee. Rotation also raises observer cost: a watcher must index every handle class, which is why the prototypes emit the handle derivations as artifacts.

Carrier camouflage: efficacy and ethics. With Core 30.0 policy (Section 3.6), explicit carriers are relay-standard, so camouflage no longer buys admission. What remains is resistance to protocol fingerprinting and a fee profile aligned with ordinary traffic. That is a legitimate goal with precedent in PayJoin [9], but "ordinary-looking" can also be read as evasion of a node operator's stated policy. We restrict the claim to protocol self-fingerprinting, and we note that the Bitcoin-side artifacts produced so far are explicit `OP_RETURN` transactions: the camouflage thesis is demonstrated at the referee layer and on LTCTEST, not yet in Bitcoin's on-chain footprint.

Oracle and shadow limits. `testmempoolaccept` reports current policy at the tip, not historical block validation; the Rust shadow executes a narrow opcode slice; the synthetic Bitcoin replay fixtures stop at `missing-inputs` until real prevouts are funded. The analytic reproduction of Section 6 is deliberately independent of all three so that the digest-level claims do not inherit those limits.

Litecoin is not Bitcoin. LTCTEST is a live engineering environment with TradeLayer available, not evidence about Bitcoin consensus. The testnet4 work narrows that gap for the reserve-bound referees but not for the procedural-token flows.

Snapshot evidence. The reserve outpoint bound by the July referee runs has since been spent (Table 8). Admission is relative to a live import, and the referee rejects stale trace hashes by design; readers reproducing the demo should expect a fresh import to change the admitted set.

No on-chain enforcement. The mechanism handles and both referees are off-chain objects. They decide what an application treats as admissible evidence and produce challenge evidence for disputes; they do not change the Taproot vault spend path, Bitcoin consensus, or BitVM's underlying dispute assumptions.

14 Discussion

The useful interpretation of these results is not that old Bitcoin bugs should be revived. It is that historical Bitcoin reveals protocol degrees of freedom that can be abstracted and measured. Some are cryptographic, such as digest aliasing or txid movement; others are topological, such as publication inside ordinary payout structure. The project's value lies in making those degrees of freedom reproducible, classifiable and portable into modern experimental grafts, with the four properties of Section 7 as the admission test. That Binohash independently made the FindAndDelete seam load-bearing for transaction introspection suggests the vein is real; this paper's claim is narrower and complementary, namely that the same seams, read as motifs, give overlay designers a vocabulary for coordination surfaces they already build by hand.

15 Future Work

Bitcoin-side materialisation. Since April, the testnet4 grant and reserve vault have been confirmed and bound into the referees. The remaining Bitcoin-side blocker is funded-prevout materialisation for the oracle-sidecar replay bench, so that the carrier-camouflage thesis can be evaluated against a live Bitcoin carrier rather than an explicit `OP_RETURN`. A second step is to re-import the reserve after its spend and record how admission changes.

Reusable overlay templates. The UTXORef mechanism catalogue is a first step from scenario-specific harnesses toward parametric declarations of relay preludes, routing buckets and dispute edges. The next step is to make the flow-graph fixtures the source of truth for both the LTCTEST runners and the referees.

Generalised tripwire. Replace the single constant-one check with a predicate over degenerate digest inputs, and add the `SIGHASH_SINGLE` hazard family to the regtest seams as policy-stamped controls.

Mainnet Litecoin and later Bitcoin windows. Rerun the ossification walk against Litecoin mainnet, and extend the Bitcoin tour through the BIP65 and CSV boundaries listed in Table 1.

Binohash-style adaptation by other developers. The broader agenda is to encourage other Bitcoin developers to mine these motifs: alternative transcript alias surfaces for challenge systems, new namespace or anchor search constructions, ordinary transaction topology as commitment cover, contract-layer analogues of the same manifold shapes, and DLC or BitVM systems that use relay and identifier freedom without depending on policy-rejected legacy forms [23]. The message to the community is intentionally modest: take these motifs, test them inside your own covenant, DLC, watchtower, wallet-coordination or BitVM-adjacent setting, and see whether they open design space without inheriting the old pathologies.

Artifacts and Reproduction

Representative repository artifacts:

- `scripts/seam_digests_standalone.py` and `docs/grants/seam_digests_standalone.{json,md,tex}`: the analytic reproduction behind Tables 2–4.
- `scripts/binohash_level_tools.py` and `docs/grants/binohash_level_tools.{json,md}`: Primitives 1–3 of Section 10 with their PASS/FAIL invariants and the cost line of Table 9.
- `scripts/covenant_limits.py` and `docs/grants/covenant_limits.{json,md}`: the SINGLE/ALL fence duality behind Primitive 4 and the three break mechanisms of Section 10.5, 17 node-free checks.
- `fixtures/manifests/*_core_seam_poc.json`, `payout_2013_carrier_camouflage.json`, `oracle_sidecar` seam and carrier manifests.
- `artifacts/grants/policy_envelope_mapping.md`, `overlay_hook_summary.md`, `overlay_carrier_bench_` `bitcoin_defi_graft_map.md`: generated crosswalks.

- `docs/litecoin-bitvm/procedural_tokens.md` and `fixtures/litecoin-bitvm/procedural_flow_graphs` the LTCTEST flow graphs.
- In UTXO-Ref: `bitvm3/utxo_referee/jurassic_bitvm_mechanisms.js`, `quirk_indexed_route_referee.` `camouflaged_watchtower_cadence_referee.js` and their reports.

Representative commands:

```
python scripts/seam_digests_standalone.py
python scripts/binohash_level_tools.py
python scripts/covenant_limits.py
cargo run -p jurassic-bitcoin-cli -- mint-findanddelete-seam \
  --out fixtures/blobs/p2sh-findanddelete-core-seam.json
cargo run -p jurassic-bitcoin-cli -- replay-era \
  --manifest fixtures/manifests/p2sh_findanddelete_core_seam_poc.json \
  --out-dir artifacts/p2sh-findanddelete-core-seam --limit-per-epoch 50 --force
cargo run -p jurassic-bitcoin-cli -- report \
  --dir artifacts/p2sh-findanddelete-core-seam --format latex
node bitvm3/utxo_referee/quirk_indexed_route_demo.js
node bitvm3/utxo_referee/camouflaged_watchtower_cadence_demo.js
```

Conclusion

Bitcoin's old quirks are not just anecdotes. When measured carefully, they define reproducible search surfaces. Transcript multiplicity, identifier bifurcation and carrier camouflage together form a compact vocabulary for reasoning about how ossified historical behaviour can inspire modern overlay systems, and semantic binding, non-collapse, authority separation and rotation invariance are the conditions under which that inspiration is safe to act on. The toolchain, the analytic reproduction, the four Binohash-level primitives, the limits result that fences off three covenant mirages, the Litecoin grafts and the testnet4-bound referees in these repositories show that the motifs can be extracted, classified, formalised, turned into specified primitives with checkable invariants, bounded by what they provably cannot do, and lifted into live BitVM-style and non-BitVM applications without confusing historical fossils for deployable mechanisms.

References

[1] Gavin Andresen. BIP 16: Pay to script hash, 2012.

[2] Gavin Andresen. BIP 34: Block v2, height in coinbase, 2012.

[3] Marcin Andrychowicz, Stefan Dziembowski, Daniel Malinowski, and Lukasz Mazurek. How to deal with malleability of BitCoin transactions, 2013.

[4] Bitcoin Core project. Bitcoin core 30.0 release notes, October 2025.

[5] Bitcoin Core project. Bitcoin core v30.0 source: `src/script/interpreter.cpp` and `src/policy/policy.cpp`, 2025.

[6] Bitcoin Wiki contributors. Contract: Example 3, assurance contracts (`SIGHASH_SINGLE|ANYONECANPAY`), 2012. Deployed later as Hearn's Lighthouse.

[7] Christian Decker and Anthony Towns. BIP 118: SIGHASH_ANYPREVOUT for taproot scripts, 2017.

[8] Christian Decker and Roger Wattenhofer. Bitcoin transaction malleability and MtGox, 2014.

[9] Nicolas Dorier. BIP 78: A simple payjoin proposal, 2019.

[10] Thaddeus Dryja. Discreet log contracts, 2017.

[11] Patrick Dugan. Jurassic Bitcoin: Consensus observability and differential testing harness, 2026.

[12] Patrick Dugan. UTXO-Ref: BitVM-style referee prototypes for reserve-backed overlays, 2026.

[13] Simin Ghesmati, Walid Fdhila, and Edgar Weippl. SoK: How private is Bitcoin? classification and evaluation of Bitcoin privacy techniques. In *Proceedings of the 17th International Conference on Availability, Reliability and Security*. Association for Computing Machinery, 2022.

[14] Lewis Gudgeon, Pedro Moreno-Sanchez, Stefanie Roos, Patrick McCorry, and Arthur Gervais. SoK: Layer-two blockchain protocols. In *Financial Cryptography and Data Security (FC 2020)*, volume 12059 of *Lecture Notes in Computer Science*. Springer, 2020.

[15] Ethan Heilman. Signing a Bitcoin transaction with Lamport signatures (no changes needed). bitcoin-dev mailing list, April 2024. Thread discusses related constructions by Jeremy Rubin.

[16] Ethan Heilman and Armin Sabouri. BIP 347: OP_CAT in tapscript, 2023.

[17] Burak Keceli. Ark: An alternative privacy-preserving second layer solution. bitcoin-dev mailing list, May 2023.

[18] Keep Starknet Strange contributors. Shinigami: Bitcoin script interpreter in Cairo, 2024.

[19] Wonhoi Kim, Hocheol Nam, Muoi Tran, Amin Jalilov, Zhenkai Liang, Sang Kil Cha, and Min Suk Kang. Fork state-aware differential fuzzing for blockchain consensus implementations. In *2025 IEEE/ACM 47th International Conference on Software Engineering*. IEEE Computer Society, 2025.

[20] Johnson Lau. BIP 147: Dealing with dummy stack element malleability, 2016.

[21] Johnson Lau and Pieter Wuille. BIP 143: Transaction signature verification for version 0 witness program, 2016.

[22] Robin Linus. BitVM: Compute anything on Bitcoin, 2023. White paper.

[23] Robin Linus. Binohash: Transaction introspection without softforks, 2026. Technical note, February 2026; discussion at `https://delvingbitcoin.org/t/binohash-transaction-introspection-without-softforks/2288`.

[24] Robin Linus, Lukas Aumayr, Zeta Avarikioti, Matteo Maffei, Andrea Pelosi, Orfeas Thyfronitis Litos, Christos Stefo, David Tse, and Alexei Zamyatin. Bridging Bitcoin to second layers via BitVM2. Cryptology ePrint Archive, Paper 2025/1158, 2025. To appear in USENIX Security 2026.

[25] Eric Lombrozo, Johnson Lau, and Pieter Wuille. BIP 141: Segregated witness (consensus layer), 2015.

[26] Patrick McCorry, Surya Bakshi, Iddo Bentov, Sarah Meiklejohn, and Andrew Miller. Pisa: Arbitration outsourcing for state channels. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies (AFT '19)*. Association for Computing Machinery, 2019.

[27] Malte Möser, Ittay Eyal, and Emin Gün Sirer. Bitcoin covenants. In *Financial Cryptography and Data Security: FC 2016 International Workshops, BITCOIN, VOTING, and WAHC*, volume 9604 of *Lecture Notes in Computer Science*. Springer, 2016.

[28] Olaoluwa Osuntokun. Taproot assets protocol. Draft BIP series, Lightning Labs, 2022.

[29] Andrew Poelstra. CAT and Schnorr tricks I, 2021.

[30] Joseph Poon and Thaddeus Dryja. The Bitcoin Lightning network: Scalable off-chain instant payments, 2016.

[31] Casey Rodarmor. Ordinal theory handbook, 2023.

[32] Jeremy Rubin. BIP 119: CHECKTEMPLATEVERIFY, 2020.

[33] Ruben Somsen. Statechains: Off-chain transfer of UTXO ownership. Scaling Bitcoin 2018, Tokyo, 2018.

[34] Pieter Wuille. BIP 62: Dealing with malleability, 2014. Withdrawn.

[35] Pieter Wuille. BIP 66: Strict DER signatures, 2015.

[36] Pieter Wuille, Jonas Nick, and Anthony Towns. BIP 341: Taproot: SegWit version 1 spending rules, 2020.

[37] Pieter Wuille, Jonas Nick, and Anthony Towns. BIP 342: Validation of taproot scripts, 2020.

[38] Youngseok Yang, Taesoo Kim, and Byung-Gon Chun. Finding consensus bugs in Ethereum via multi-transaction differential fuzzing. In *15th USENIX Symposium on Operating Systems Design and Implementation*, 2021.