The Achievable Floor

Josh · Secure Sovereign

Published July 5, 2026 · Updated August 13, 2026

Contents

- I. Framing the Problem
- II. Taxonomy of Channels by Cost
- III. What Consensus Can Actually Close
- IV. Why the Free Channels Resist Closure
- V. Custody of Data That Already Exists
- VI. The Actual Floor
- VII. Implementation Path
- VIII. Conclusion

I. Framing the Problem

The Bitcoin spam debate blurs **two layers.** Consensus forces every validating node to accept whatever is in a valid block, monkey jpegs included. **Relay and storage are policy.** No node must forward a transaction before it confirms, and no node must keep every byte forever after validation. Pruning exists because long-term storage is optional.

Pruning has a cost. A pruned node cannot re-check the full chain from its own disk without asking someone else for old blocks. Pruning also does not remove the first-sync cost. Every pruned node still downloads and validates the whole history, including non-monetary data, during initial block download. The spam still passes through every new node at first sync, whether that node keeps the data afterward. Bandwidth and validation time at join are permanent costs on every participant who syncs. **Pruning changes when you pay for storage. It does not remove the download.**

Refusing to relay is not banning. Miners and permissive nodes still include transactions that pay competitive fees, regardless of one operator's mempool policy. Relay limits on `OP_RETURN` and similar fields are per-node settings: any operator can tighten them, loosen them, or turn them off. **Consensus caps bind every participant or fail to activate.**

The fight is usually about consensus capture, forking risk, and who decides which use cases are legitimate. For OP_RETURN and data-embedding politics, see *Who Controls Bitcoin, §V* and *Argument Map, Parts VI–VII and XXII*.

Set politics aside. If consensus could change freely, how much non-monetary data could rules actually eliminate, and where is the hard floor?

Spam has a workable definition. A spam transaction does not settle money, and it imposes lasting costs on every validating node. Lightning channel opens and closes settle money. Timelocked outputs and multisig setups settle money. A JPEG stuffed into a Taproot envelope does not. That follows from how Bitcoin works: every validating node must process the full chain history to know current balances. Using the payment layer as a cheap bulk data store pushes costs onto a network built for transfers, not file hosting. For the full case against non-monetary embedding, see *Bitcoin Is Not a Hard Drive*. Demanding a definition that can never be met is not a technical objection.

II. Taxonomy of Channels by Cost

Non-monetary data can enter blocks through several different fields and transaction shapes. Those paths are not equally cheap. The ladder below ranks them by what it costs the person embedding the data.

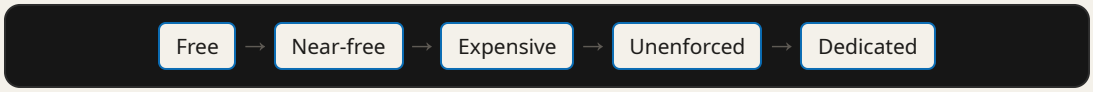

Figure: Channel cost ladder. Consensus can close dedicated and unenforced paths without breaking normal payments.

Free channels cost nothing beyond the ordinary transaction fee.

The main free path is any field that stores a hash of a key or script instead of revealing the key or script itself: P2PKH or P2WPKH destinations, Lightning hash locks, and P2SH or P2WSH script hashes. A fake 20 or 32 byte string looks the same as a real hash. Pay-to-Fake-Key and Pay-to-Fake-Multisig (payments to fake keys or multisig that look like real addresses) predate `OP_RETURN`. The 2010 WikiLeaks Cablegate dump used this method.

Inviscription is one such scheme: it splits encrypted data across hash-locked outputs, multisig key slots, or timelock fields. The decryption key may appear in a later transaction or never on chain.

Output amounts are free too. Consensus cannot tell whether a chosen value encodes data or is just a payment. Attackers can hide data in the satoshi amounts of outputs, not only in scripts or witness data.

Structural fields add more. `nSequence` and `nLockTime` can each carry a few bytes of data. Input and output order is unconstrained. Consensus could force fixed values for all of these, but that would break how ordinary wallets build transactions. **You would harm normal payments without stopping spam**, the same problem as trying to ban flexible output amounts.

Near-free channels cost a small number of trial attempts.

Taproot's 32 byte pubkey field is the standard example. Roughly half of all 32 byte strings are valid public keys on Bitcoin's curve. Hiding a chosen payload as a fake pubkey takes about two random tries on average.

The same low cost applies to other standard address fields: P2WPKH and P2PKH hash slots accept any 20-byte string, and P2TR pubkeys need only a few tries. Researchers measuring real blocks have found hundreds of kilobytes of data embedded this way.

Expensive channels require brute-force search. The attacker keeps trying random keys or signatures until one matches a desired bit pattern. Cost grows exponentially with how many bits they want to fix. Vanity addresses use the same trick, pointed at chosen targets.

Unenforced channels have no content rules today.

Undefined witness versions and OP_SUCCESS opcodes are upgrade hooks. Consensus treats anything after an OP_SUCCESS byte in Tapscript as valid, no matter what it contains. The Taproot annex is extra witness data left out of the signature that authorizes the spend, with no limit on what it holds. **The annex can hold raw bytes. Nothing has to be disguised as a hash or public key.**

Dedicated channels exist to carry data, or hold large script blobs with no payment meaning.

`OP_RETURN` is an output that cannot be spent. Core v30 relay policy allows up to 100,000 bytes of `OP_RETURN` data per transaction, across multiple outputs. Consensus itself places no byte limit. In June 2023, PR #27832 narrowed the documented meaning of `-datacarriersize` so it covered only `scriptPubKey` outputs, not witness or inscription fields. Core v30 then removed the relay cap entirely in 2025. Separately, the Taproot envelope hides data inside an `OP_FALSE OP_IF` branch that never runs. SegWit's witness discount and Taproot's removal of the old 10,000 byte script ceiling made large envelope payloads practical.

CHANNEL	EMBEDDING COST	CLOSABLE AT CONSENSUS?	COST TO CLOSE
Dedicated	Fee-paid, large payloads	Yes	No hit to normal payments
Unenforced	Low; no validation yet	Yes	Tighten unused upgrade hooks
Expensive	Exponential with chosen bits	No	Cost is the only limit
Near-free	~2 trial attempts	Partial	Filters barely help
Free	None beyond tx fee	No	Lose privacy, precision, or supply audit

Closability by channel type

III. What Consensus Can Actually Close

Closable and unclosable paths are not the same problem. `OP_RETURN` and Taproot envelope embedding can be closed at consensus. Sending a transaction straight to a mining pool, private miner peering, and other off-relay paths cannot be closed at consensus without redesigning how mining works. After closing what consensus can close, some spam may still reach blocks through paths consensus cannot shut. **That is not an argument against closing the paths consensus can shut.** Treating those as one objection is a mistake.

Dedicated and unenforced channels can close at consensus. Ordinary payments do not need them.

A consensus hard cap on `OP_RETURN` closes bulk dedicated-data outputs and binds every participant. Relay limits bind only the operators who choose them.

Consensus can also cap data hidden in script branches that never run, including the `OP_FALSE` `OP_IF` envelope used for inscriptions. *Permanent Data Channel Closure* does that by invalidating `OP_IF` and `OP_NOTIF` anywhere in Tapscript, even unexecuted. Current P2WSH Lightning is unaffected. Taproot Miniscript and taproot-channel HTLC leaves must split those branches into separate leaves.

Consensus can also restrict unused witness versions, ban or cap the annex, and cap the Taproot control block (the path that reveals which script branch is being spent) so large unused branches cannot smuggle big payloads.

A consensus per-output miner fee makes creating many outputs expensive. Unlike a minimum output value (capital the spammer can recycle on spend), the fee is paid to the miner as ordinary transaction fee and does not come back. That hits spam that spreads data across lots of small

outputs. Higher fees make each of these limits cost the embedder more. See *Static Per-Output Miner Fee*.

The rows below close dedicated and unenforced embedding channels. **§V addresses data already on chain** through UTXO set commitments (less UTXO state to store). That layer is independent of the caps here. Initial block download still carries the full history; AssumeValid does not remove that cost.

CONSENSUS MEASURE	CHANNEL CLOSED	INDEPENDENT SOFT FORK?
OP_RETURN hard byte cap	Dedicated (OP_RETURN outputs)	Yes
Taproot envelope push cap	Dedicated (OP_FALSE OP_IF branches)	Yes
Witness version restriction	Unenforced (OP_SUCCESS hooks)	Yes
Annex disallow or cap	Unenforced (Taproot annex)	Yes
Control block size cap	Unenforced (deep Merkle path hiding)	Yes
Per-output miner fee	Low-value UTXO spam (output count)	Yes
UTXO set commitments	Permanent storage burden for data already on chain (§V)	Parallel; not required for §III caps

Closable measures. First five rows: Permanent Data Channel Closure. Per-output fee: Static Per-Output Miner Fee plus optional Dynamic Escalation. Commitments remain a parallel track (§V). §IV fields omitted because monetary design requires them.

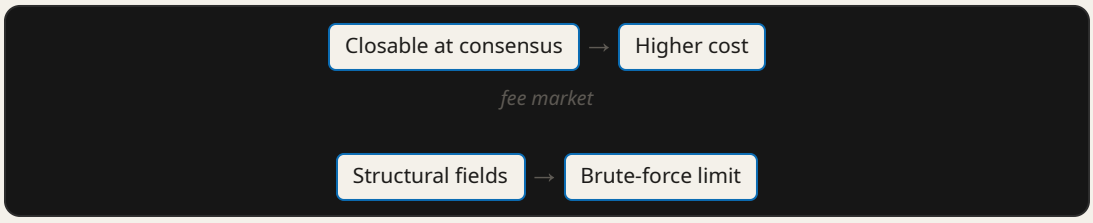

Figure: §III closes OP_RETURN, envelope, annex, control block, and per-output fee. §IV fields cannot.

IV. Why the Free Channels Resist Closure

Closing free or near-free channels either breaks normal payments or barely raises the cost of spam.

Checking whether a fake pubkey is a valid curve point barely helps. With near-free channels, an attacker can keep trying random values until almost any payload passes the check, so the filter blocks almost nothing.

Requiring the real pubkey, or a zero-knowledge proof of one, when an output is created closes hash-based channels but ends hash-then-reveal privacy for every user. P2PKH and P2WPKH withhold the pubkey until spend time so funds stay safe if the cryptography behind the address is ever broken. **That protects all users; giving it up to chase spam would be a bad trade.**

The zero-knowledge variant keeps privacy but still fails. An attacker can brute-force a real keypair until its hash encodes a payload, and every payment would need extra zero-knowledge verification overhead.

The amount channel costs the attacker nothing. Sequence numbers, locktimes, and input or output ordering work the same way.

Rounding all amounts to coarser units removes some low bits, not the channel. The attacker still picks which multiple to use. Across 21 million bitcoin, coarsening still leaves tens of bits per output, and every user loses payment precision.

Fixing input order or mandating single sequence and locktime values hits the same wall. There is no single correct value, only a range. One mandated choice breaks ordinary wallet use.

Encrypted amount schemes, as in Liquid and Mimblewimble, hide amounts on chain but remove the ability for anyone to audit total supply.

Inscriptions, BRC-20, and Rune issuance pay fees and look like ordinary payments at the consensus layer. Users will keep paying for them. Closing those paths gets politically harder; the free channels stay open.

Bitcoin was built for payments, not token registries, inscriptions, or bulk data availability. The free fields exist because payments need them.

V. Custody of Data That Already Exists

Sections I–IV are about new data reaching the chain. **Two separate problems remain for data already stored:** how much history a new node must download at first sync, and how much UTXO state every node must keep afterward.

Initial block download. A full node still downloads the chain from genesis: every block, every transaction, every witness, including years of non-monetary data. Bitcoin Core's AssumeValid setting can skip script verification for blocks before a trusted hash, which cuts CPU during sync. **It does not skip the download.** The spam still crosses the wire for every new node. AssumeValid does not close embedding fields for new transactions, and it is not a substitute for consensus caps on dedicated channels.

UTXO set commitments. Fake hash outputs stay in the UTXO set until spent. Without the secret behind the hash, no one knows whether an output will ever move. Inscription outputs already dominate UTXO count while holding almost no monetary value.

UTXO set commitments let a node keep a short root hash instead of the full set, and check spends using proofs the spender provides. Every node no longer has to store the whole set locally; spenders supply what is needed to prove ownership. Utreexo is one design. Putting that root in the block header under consensus rules removes the need to trust special bridge nodes that hand out set data.

Some production designs ask peers to agree on the root rather than committing it in consensus rules. That requires trusting that most peers report the same value. Commitments change ongoing set storage. They do not remove the IBD download cost above.

VI. The Actual Floor

`OP_RETURN`, the Taproot envelope, undefined witness versions, and the annex can close at consensus. Payments do not need them.

Dedicated channels and witness-discounted envelopes drove most of the bandwidth. What remains is built into payment design: hash fields, amounts, sequence, locktime, and ordering. Close those and you lose hash-then-reveal privacy, satoshi precision, or supply auditability. **UTXO commitments can reduce how much old state each node stores. AssumeValid can cut script-check CPU during sync; it does not cut the download. Neither removes the embedding floor in §IV.**

Some embedding capacity is built into payment fields on purpose. **Those fields exist for security and privacy, not as spare storage.**

The blockspace impact is measured, not guessed. A full chain scan across 912,723 blocks and roughly 1.235 billion transactions finds spam's share of blockspace about 17 times higher than its pre-inscription baseline. Non-monetary data accounts for an estimated 12 to 19% of total chain storage; for the full operator-cost model behind those figures, see *Full Cost of Running a Bitcoin Node*. Per Mempool Research, 29.6% of all UTXOs are inscription-related, holding about 415 BTC in total value. Blocks ran between 91 and 97% full across multiple weeks in 2026. After Core v30 removed the relay cap on `OP_RETURN`, Renaud Cuny's December 2025 analysis found large OP_RETURN activity starting immediately while inscription witness data continued at scale, roughly 36% of blockspace non-financial as of December 2025. The policy change opened a new channel without closing the old one. Claims that the impact is negligible require ignoring documented methodology and published measurements. For the governance timeline behind removing the relay cap, see *Who Controls Bitcoin, §V*.

Another reply says the block weight limit already makes spam acceptable: each block is capped at roughly four million weight units (often called a four-megabyte ceiling), so damage has an upper bound.

That is a per-block validation rate limit, not a verdict on what the capacity is for. The cap keeps one block finite to validate. It does not bound the cumulative stock of confirmed non-monetary bytes. Every spam byte still hits every future IBD, still occupies archival storage, and for fake-hash or dust-shaped outputs still accumulates in the UTXO set. **A capped rate over years is still a large permanent burden.**

The same ceiling is the monetary settlement budget. Blocks at 91 to 97% full with a large non-financial share are non-monetary demand competing for scarce payment capacity, not harm already contained. BIP141 makes the bound asymmetric: witness bytes cost one weight unit and non-witness bytes cost four, so envelope payloads buy more raw data per weight unit than base-transaction payment data. Weight also rate-limits new outputs per block without forcing never-spent spam UTXOs out of the set afterward. The inscription-related UTXO share above is bounded rate with unbounded accumulation of the wrong state.

WHAT THE WEIGHT LIMIT DOES	WHAT THE SPAM DEBATE IS ABOUT
Caps bytes (by weight) in one block	Permanent IBD, storage, and UTXO costs across all future nodes
Keeps one-block validation finite	Whether non-monetary use should consume the scarce settlement budget
Same MWU budget for all tx types	Witness discount admits more envelope payload per MWU than base payment data
Rate-limits new outputs per block	Does not evict unspendable or never-spent spam UTXOs afterward

Block weight is a DoS bound on one block, not a verdict that filling it with non-monetary data is acceptable.

The weight limit answers whether one block can overwhelm a node. The spam problem asks who permanently pays bandwidth, disk, and UTXO set for non-monetary state. Those are different threats. For the same point as a justification failure, see *Bitcoin Is Not a Hard Drive, §VI*.

Cost per embedded byte. The figures below are structural arithmetic from §II channel sizes, BIP141 weight rules, and Core v30 relay defaults. They are not a measurement of the live chain.

Payload size. Core v30 defaults `-datacarriersize` to **100,000 bytes** of aggregate `OP_RETURN` `scriptPubKey` per transaction. Consensus imposes no byte cap. Both dedicated rows use **1,024 bytes** of payload. The per-transaction size limit binds before the datacarrier limit. The hash row uses 20 bytes per fake P2WPKH output; the near-free row uses 32 bytes per fake P2TR pubkey (§II).

Witness discount (BIP141). Witness bytes count one weight unit; non-witness bytes count four. Envelope payload sits in witness. Hash and pubkey payloads in `scriptPubKey` are non-witness, so they cost about four times as much per embedded byte.

Bytes used. An `OP_RETURN` with a 1,024-byte push is 1,039 vbytes (8-byte value, 3-byte varint, 1,028-byte script). P2WPKH with a 20-byte fake hash is 31 vbytes. P2TR with a 32-byte fake pubkey is 43 vbytes. Envelope at 1,024 witness payload bytes is 43 vbytes of output plus 256 vbytes of witness (wrapper and spend overhead omitted; full minimal spend ≈377 vbytes).

After closing dedicated channels. Dust defaults at 3 sat/vbyte (`DUST_RELAY_TX_FEE = 3000` sat/kvB) are 294 sat for P2WPKH-shaped outputs and 330 sat for P2TR-shaped. `OP_RETURN` outputs are zero-value and omitted. Dedicated rows show fee cost only; hash and pubkey rows add the dust floor because embedding creates spendable-looking outputs.

CHANNEL	PAYLOAD BYTES (SPEC)	VBYTES / EMBEDDED BYTE	SATS / BYTE @ 10 SAT/VB	@ 50 SAT/VB	@ 100 SAT/VB
OP_RETURN (dedicated)	1,024	1.01	10.1	50.7	101.5
Taproot envelope (dedicated)	1,024	0.29	2.9	14.6	29.2
P2WPKH fake hash (free, after §III)	20	1.55	30.2	92.2	169.7
P2TR fake pubkey (near-free, after §III)	32	1.34	23.8	77.5	144.7

BIP141 accounting, Core v30 `-datacarriersize` (100k), dust at 3 sat/vB. Fee columns: (vbytes × rate + dust where applicable) ÷ payload bytes.

Closing dedicated channels does not make embedding cheaper. It removes witness-discounted envelopes and bulk `OP_RETURN`, and forces data into hash and pubkey outputs that each pay a dust floor. At every fee rate in the table, hash and pubkey rows cost more per byte than `OP_RETURN` at 1,024 bytes.

Policy and fees still matter. Consensus can close high-bandwidth channels and reduce per-node storage load. Eliminating all non-monetary data would mean stripping payment properties Bitcoin needs. The realistic goal is to keep non-monetary use costly and unwelcome.

VII. Implementation Path

The §III dedicated and unenforced caps are specified as one pre-proposal: *Permanent Data Channel Closure* (output templates, 83-byte OP_RETURN `scriptPubKey`, 256-byte item caps, aggregate witness limits, annex ban, control-block cap, Tapscript `OP_IF` / `OP_NOTIF` invalid, Tapleaf and P2WSH unreferenced-push analysis). Envelope closure is by invalidating `OP_IF` in Tapscript, not by a payload cap on skipped branches; Miniscript and taproot-channel HTLC leaves must split into separate leaves. Undefined witness versions cannot be created or spent; a future version is added by amending the template list in the same soft fork that defines spending. BIP141 weight is unchanged.

The per-output fee is *Static Per-Output Miner Fee*: a consensus floor on `tx_fee` of `static_fee × n_outputs`, paid as ordinary miner fee, not minted. *Dynamic Escalation of the Per-Output Miner Fee* is the optional anti-decay layer on top of that floor.

Consensus caps on dedicated embedding channels do not close every cheap field on the input side. Inputs still carry low-cost fields such as `nSequence`, `nLockTime`, and ordering. A permanent per-output creation cost closes those paths indirectly. Inputs spend outputs created earlier. If new outputs cost something to create under consensus rules, stuffing data into later spend fields still requires paying that creation cost up front. The floor does not ban input fields directly. It removes the incentive to mint cheap outputs just to encode data when they are spent. Dedicated-channel caps and a consensus per-output fee work together; they are not competing proposals. The three pre-proposals are meant to be considered as one stack. Separate activation is possible. The combined effect is stronger.

UTXO commitments are an optional parallel layer. Each §III embedding cap can soft-fork on its own. Commitments change how much set state a node must keep; they are not a prerequisite for the caps, and the caps are not a prerequisite for commitments. Outputs that would fail a new rule need grandfathering.

Policy tools (mempool filters, envelope detection, miner fee multipliers) stay useful but do not bind the network. A transaction that pays a willing miner still confirms. **Only consensus binds everyone.**

VIII. Conclusion

Even if consensus could change without a political fight, **non-monetary data on chain cannot be eliminated entirely.** Payments still need hash addresses, flexible amounts, and auditable supply.

Data that never appears on chain, including Taproot script branches never revealed in a spend, is outside this analysis.

The §III caps belong in a formal spec. The Bitcoin Commons consensus spec is the reference. For why a human-readable specification matters as governance infrastructure, see *Why Bitcoin Needs a Specification*. For the dollar cost of leaving non-monetary data unpriced, see *Full Cost of Running a Bitcoin Node*. The three-BIP stack that implements §III plus a UTXO-creation floor: *Permanent Data Channel Closure*, *Static Per-Output Miner Fee*, *Dynamic Escalation of the Per-Output Miner Fee*.

SECURESOVEREIGN

Markdown is the canonical source: https://secsov.com/articles/the-achievable-floor/index.md

Formatted snapshot of https://secsov.com/articles/the-achievable-floor

Josh · Secure Sovereign