Dynamic Escalation of the Per-Output Miner Fee

Josh · Secure Sovereign

Pre-Proposal · Standards Track · Created July 23, 2026

CONTENTS

Abstract

This BIP adds a dynamic escalation layer on top of the static per-output miner fee from its predecessor. It has a hard consensus dependency on *Static Per-Output Miner Fee*: the dynamic rules are inert for any block at or below the static fee BIP's activation height, regardless of this BIP's own signaling or lock-in status. Until that height, this BIP adds no fee floor.

The problem is long-run decay. A static satoshi amount fixed under today's conditions becomes trivial as Bitcoin's price and fee levels rise over decades. Without automatic adjustment, the static fee stops working as a real disincentive, and the only fix is another soft fork to raise the constant. Repeated soft forks to update a fixed number create a permanent dependency on governance processes that can be captured, delayed, or blocked.

This BIP specifies a dynamic fee component derived from the 25th percentile fee rate sampled over 288-block windows, smoothed by an exponential moving average, confirmed by multi-window persistence, and bounded by an upward-only rate-of-change cap. The active per-output fee is the maximum of the static fee and the dynamic fee. In normal and high fee conditions the dynamic fee sits above the static fee and tracks economic conditions. In prolonged low-fee regimes the static fee governs and the dynamic component contributes nothing.

This BIP is an optional long-term upgrade to the static fee. The static fee BIP works fully without it. Dedicated embedding channels are closed by Permanent Data Channel Closure, which does not depend on this BIP.

Motivation

The Known Limitation of the Static Fee

The static per-output miner fee closes both the value vector and the count vector of UTXO spam with a permanent, non-recoverable cost per output. It is the main piece of the fee layer. It has one known limit, already noted in its security considerations: a fixed satoshi amount decays in real terms as Bitcoin's price and fee levels rise.

At 20 sats per output and current BTC prices, creating 100,000 outputs costs 2 million sats. That is a material cost today. At 10x BTC price it is one-tenth the real cost. At 100x BTC price it approaches noise:

BTC PRICE MULTIPLE	COST OF 100K OUTPUTS @ 20 SATS	REAL DETERRENT?
1× (today)	2,000,000 sats	Material
10×	2,000,000 sats (1/10 real cost)	Weakening
100×	2,000,000 sats (1/100 real cost)	Noise

Nominal sats stay fixed; real economic cost decays as BTC appreciates. Without automatic adjustment, another soft fork is eventually required.

The static fee does not need a fix today, but without automatic adjustment it will eventually need a soft fork. That soft fork will face the same political and review surface as any other consensus change, with the extra difficulty of being an explicit fee increase rather than a new capability.

The dynamic escalation layer removes that future dependency. It is calibrated to track multi-decade fee drift, not short-term fee spikes, so it adjusts slowly and predictably rather than reacting to noise.

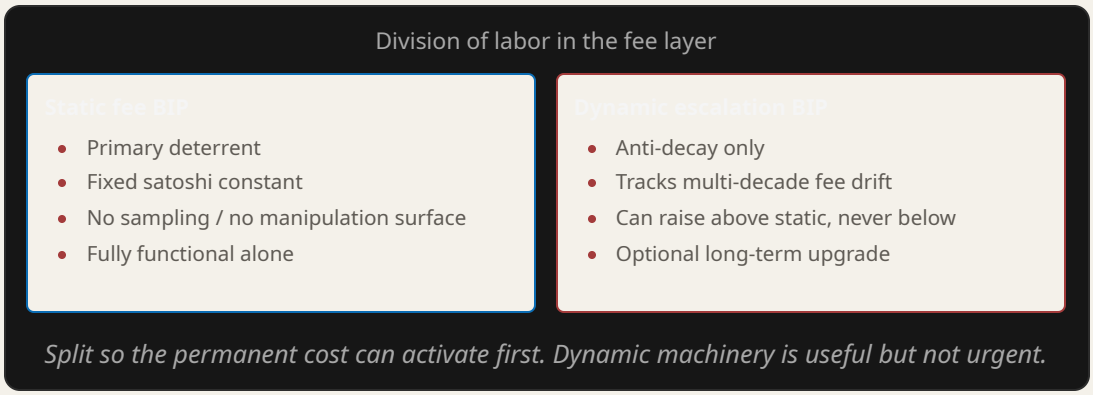
Why This Is a Separate BIP

The static fee is simple, small, and high-leverage. Putting the dynamic machinery in the same proposal raises implementation complexity, expands review surface, and raises the chance that neither piece activates. Splitting the proposals means the most important property, a permanent per-output cost, can land in consensus first. This BIP is the optional upgrade that keeps that cost from decaying over time.

Separation also lets the network gain operational experience with the static fee before adding the dynamic layer. If the static fee is enough in practice, the dynamic BIP may not need to activate on the same timeline. If the static fee decays faster than expected because of BTC price appreciation, the dynamic BIP has a clearer and more urgent case.

Anti-Decay Is the Only Job of the Dynamic Layer

The dynamic component has one job: keep the static fee from becoming economically trivial over decades. It is not meant to respond to short-term spam waves, react to individual fee spikes, or replace the static fee as the primary deterrent. During short-term fee anomalies the dynamic fee should stay stable. During genuine multi-year fee drift it should move up. The design choices below (slow EMA, multi-window persistence, upward-only rate cap) all follow from that single role.



Specification

Dependency

This BIP is only valid when *Static Per-Output Miner Fee* is active. If the static fee BIP has not activated, this BIP has no effect. Implementations must check for static fee BIP activation before applying dynamic escalation logic.

Active Fee

The active per-output fee for any given 288-block period is:

```
active_fee = max(static_fee, dynamic_fee)
```

Where `static_fee` is the constant from the static fee BIP and `dynamic_fee` is computed as specified below. The static fee BIP's validation rule is unchanged except that `static_fee` is replaced by `active_fee`:

```
tx_fee >= active_fee × n_outputs
```

Coinbase accounting is unchanged from existing consensus. The dynamic amount is paid as ordinary `tx_fee`, not minted.

During any period in which the dynamic fee falls below or equals the static fee, the active fee equals the static fee and validation matches the static fee BIP.

Period Alignment

A period is the half-open height interval `[n × 288, (n + 1) × 288)` for integer `n >= 0`, aligned to genesis (`height // 288`). The dynamic fee computed from the transactions in period `n − 1` applies to non-coinbase transactions in period `n`.

This BIP performs no dynamic computation until the first full 288-block period that begins at or after this BIP's `activation_height`. Until that boundary, `active_fee = static_fee`. If the static fee BIP is not yet active, this BIP is inert and adds no floor.

Initial state at the first computation boundary:

- `p25_ema_previous = 0`
- `dynamic_fee_previous = 0`
- `active_fee_previous = static_fee`
- `active_fee_current = static_fee`

Persistence requires two consecutive windows above threshold, so the first computed period cannot raise `dynamic_fee`. The second consecutive window can.

Dynamic Fee Computation

The dynamic fee is computed at each 288-block boundary and cached for the following period. Computation happens only at boundaries, never per block.

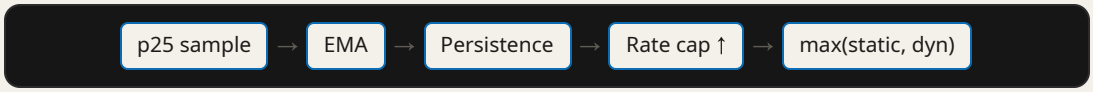


Figure: Five-step pipeline. Each layer exists to track multi-decade drift without reacting to short-term fee noise or cheap manipulation.

Step 1: Sample the fee rate

This step is the sole definition of fee sampling in the fee-layer BIPs. The static fee BIP needs no fee sampling; all fee rate computation starts here.

Collect the fee rate for every fee-paying transaction in the previous 288-block period. Fee rate is in milli-sat/vB as a non-negative integer:

```
vsize = ceil(weight / 4)          # vbytes, weight in WU
rate  = floor(tx_fee * 1000 / vsize)   # milli-sat per vB
```

`tx_fee` is `sum(input_values) - sum(output_values)` as in the static fee BIP. Exclude coinbase transactions. Exclude transactions with `tx_fee = 0`. Rate uses that transaction's own fee and weight only: no package or CPFP attribution.

Compute the 25th percentile of the resulting integer set. Sort ascending. For `N` samples, the 25th percentile is the element at index `floor((N − 1) × 25 / 100)` (0-based). No interpolation on even `N`. If the set has fewer than `min_tx_count` fee-paying transactions, **do not update** `p25_ema_previous` or `dynamic_fee_previous` (hold). Do not set them to zero. A thin sample must not collapse or freeze-from-zero the dynamic component. The period still advances: set `active_fee_previous = active_fee_current` so the grace window tracks the held fee. `active_fee_current` stays unchanged.

Step 2: Apply the exponential moving average

All EMA math is integer, toward zero:

```
ALPHA_DEN = 1000
alpha_n   = <integer, 100 means α = 0.1>
p25_ema   = (alpha_n × p25_current + (ALPHA_DEN − alpha_n) × p25_ema_previous) / ALPHA_DEN
```

A slow `alpha_n` toward 100 tracks multi-year fee drift while ignoring short-term spikes. Because the dynamic layer's only job is anti-decay over decades rather than short-term spam response, `alpha_n` must be tuned slow. A faster value toward 300 is appropriate only if sensitivity analysis shows a real benefit from faster response without instability.

Step 3: Apply multi-window persistence

```
if p25_ema_current > persistence_threshold
   and p25_ema_previous > persistence_threshold:
    candidate_dynamic = (k × p25_ema_current) / 1000
else:
    candidate_dynamic = dynamic_fee_previous
```

`persistence_threshold` is in milli-sat/vB, the same units as `p25_ema`. `k` is an integer vbyte scale that converts that EMA into a per-output satoshi fee. Truncation toward zero. `k` is **not** the serialized size of an output. It is a calibrated scale so that at a reference p25 (about 5 to 10 sat/vB) the dynamic component is the same order of magnitude as `static_fee`. A single `k`, rather than

per-script-type values, matches the static fee's uniform-per-output design. A single elevated 288-block window (~2 days) does not move the dynamic fee. An attacker must sustain elevated fee conditions across at least two consecutive windows (~4 days) to produce any upward movement.

Step 4: Apply the rate-of-change cap

```
R_NUM = 1
R_DEN = 4                          # R = 0.25 unless calibration sets otherwise
cap_base = max(dynamic_fee_previous, static_fee)
max_increase = cap_base + (cap_base × R_NUM) / R_DEN
dynamic_fee_new = min(candidate_dynamic, max_increase)
dynamic_fee_new = max(0, dynamic_fee_new)
```

All integer, toward zero. `cap_base` is never 0 while the static fee BIP is active, so a zero `dynamic_fee_previous` cannot trap the cap at 0. The first upward move is bounded by `static_fee × (1 + R)`, not by an uncapped jump to `candidate_dynamic`.

The dynamic fee may not increase faster than that bound per 288-block period. Decreases are uncapped: the dynamic fee falls freely back toward zero when fee conditions normalize, at which point the static fee governs.

Step 5: Apply the static fee floor

```
active_fee = max(static_fee, dynamic_fee_new)
```

Transaction Fee Floor During the Grace Window

The static fee BIP has no grace window at activation. This BIP changes `active_fee` every 288 blocks, so in-flight transactions need a short hold at period boundaries.

During the first `G` blocks of a new 288-block period, the required floor is the previous period's `active_fee`. After `G` blocks, the new period's `active_fee` applies. Coinbase rules are unchanged.

```
if block_height < period_start + G:
    required_fee = active_fee_previous_period
else:
    required_fee = active_fee_current_period

tx_fee >= required_fee × n_outputs
```

Consensus State

Each node must carry the following state across 288-block period boundaries, in addition to the static fee BIP's activation state:

- `dynamic_fee_previous` (the dynamic fee from the prior period)
- `p25_ema_previous` (the EMA value from the prior period)
- `active_fee_previous` (for grace window floor)
- `active_fee_current` (the active fee for the current period)

All values are deterministic from block history and need no external input. Implementations must also retain EMA state snapshots at every period boundary for the most recent 2,016 blocks of boundaries to support reorg recovery, as specified in Security Considerations.

Parameters

All parameters are consensus parameters adjustable by a future soft fork. Initial values are provisional pending calibration.

PARAMETER	DESCRIPTION	PROVISIONAL VALUE
`k`	Integer vbyte scale from p25 EMA (milli-sat/vB) to sat per output	2-4
`alpha_n`	EMA numerator over 1000 (100 = α 0.1)	100-300 (tune slow)
`persistence_threshold`	Minimum p25 EMA (milli-sat/vB) for a window to count toward persistence	TBD via calibration
`R_NUM / R_DEN`	Maximum upward rate of change per period	1/4
`G`	Grace window in blocks at period boundaries	6-12
`min_tx_count`	Minimum fee-paying transactions for a valid sample	1,000

Prefer slow `alpha_n` unless sensitivity tables show a real benefit from faster response without instability. `k` is a scale, not output vsize.

Activation

This BIP is meant to deploy via a soft fork using BIP-8 or BIP-9 style signaling, with a minimum activation window of one year and no mandatory lock-in fallback. Miners who do not signal are not penalized.

This BIP has an explicit consensus dependency: it is inert for any block whose height is less than or equal to the static fee BIP's `activation_height`, regardless of this BIP's own signaling or lock-in status. Until the static fee BIP is active, `active_fee` is not applied by this BIP (the static BIP's floor is also absent). After the static fee BIP is active and this BIP has completed its first full 288-block period, `active_fee = max(static_fee, dynamic_fee)` as specified. Signaling order is not enough; implementations must enforce the height dependency at the consensus layer.

BIP-8/9 without lock-in-on-timeout is miner signaling. Node-operator interest is not the threshold. User-activated soft fork deployment is a separate choice and is not specified here.

Rationale

Why 288 Blocks as the Sample Window

The dynamic fee's job is tracking multi-decade fee drift, not reacting to short-term spam waves. A 288-block (~2 day) window provides enough transactions for a statistically meaningful 25th percentile during active periods, while staying short enough to detect genuine sustained changes without unreasonable lag. Combined with the EMA, the effective response time to sustained fee changes is much longer than 288 blocks regardless of window size.

Why the 25th Percentile

The 25th percentile represents cheaper legitimate transactions and is less sensitive to outliers than the median. A small number of high-fee transactions cannot push it far upward, which keeps the dynamic fee from pricing out legitimate use during fee spikes driven by monetary activity rather than spam.

Why EMA Rather Than Hard Window Reset

A hard window reset creates cliff-edge transitions and makes the fee reactive to single-window anomalies. The EMA smooths toward prevailing conditions over many periods, which is the right behavior for a mechanism tracking multi-decade drift rather than reacting to individual events.

Why Multi-Window Persistence

A single elevated 288-block window is achievable by a moderately funded actor for whom fee spend is an acceptable operating cost. Requiring two consecutive windows doubles the sustained cost and duration of any manipulation attempt. The persistence check adds no per-block computation and is the main defense against short-term fee flooding aimed at ratcheting the dynamic fee upward.

Why an Upward-Only Rate-of-Change Cap

Even with EMA smoothing and persistence, a sustained campaign could ratchet the dynamic fee upward over many periods. The rate cap bounds how fast that can happen regardless of fee pressure: even a perfectly sustained campaign can raise the dynamic fee by only `R` percent per period, so rapid ratcheting is slow, expensive, and publicly visible on chain. The cap is asymmetric on purpose: the dynamic fee falls freely when pressure stops, so a manipulation attempt does not leave a permanently elevated floor once the attacker withdraws.

Why k Is a Scale, Not Output Size

The p25 EMA is in milli-sat/vB. Multiplying by an integer `k` and dividing by 1000 produces sats per output. Dimensionally `k` has units of vbytes, but it is **not** set to the serialized size of a P2WPKH or P2TR output (31–43 vB). Using literal output vsize would make `dynamic_fee` track weight fees one-for-one and would overprice Lightning and CoinJoin during congestion. Calibration sets `k` so that at a reference p25 of about 5 to 10 sat/vB, `dynamic_fee` is the same order as `static_fee` (provisional `k` in 2–4). A single `k` matches the static fee's uniform-per-output model: the externalized cost being priced is per UTXO slot, not per script type.

Interaction with Permanent Data Channel Closure

The Permanent Data Channel Closure pre-proposal targets dedicated high-bandwidth data channels. This BIP targets UTXO creation economics. They operate on different surfaces. Dynamic escalation does not change that split. See *The Achievable Floor* and *Bitcoin Is Not a Hard Drive*.

Backwards Compatibility

The active fee replaces the static fee in the per-transaction floor. During any period in which the dynamic fee is below the static fee, validation matches the static fee BIP. The dynamic layer is additive: it can only raise the active fee above the static fee floor, never below it.

Wallets and fee estimators that already account for the static per-output fee must also track the current period's active fee, which may differ from the static fee during elevated fee conditions. From the user's point of view there is still one fee; wallets present a single total that uses whichever of the two rates is currently required, including the grace-window hold at period boundaries.

Reference Implementation

High-level pseudocode, assuming the static fee BIP is active:

```
STATIC_FEE = <from static fee BIP>
ALPHA_N = 100
ALPHA_DEN = 1000
K = <2-4 pending calibration>
R_NUM, R_DEN = 1, 4
MIN_TX_COUNT = 1000
G = <6-12>
RATE_SCALE = 1000  # milli-sat/vB

def tx_vsize(tx):
    return (tx.weight + 3) // 4

def tx_fee(tx):
    return sum(inp.value for inp in tx.inputs) - sum(out.value for out in tx.outputs)

def sample_rates(period_blocks):
    rates = []
    for block in period_blocks:
        for tx in block.transactions:
            if tx.is_coinbase or tx_fee(tx) == 0:
                continue
            rates.append((tx_fee(tx) * RATE_SCALE) // tx_vsize(tx))
    rates.sort()
    return rates

def percentile_25(rates):
    n = len(rates)
    return rates[(n - 1) * 25 // 100]

def compute_period_active_fee(last_288_blocks, state):
    rates = sample_rates(last_288_blocks)
    if len(rates) < MIN_TX_COUNT:
        state.active_fee_previous = state.active_fee_current
        return state.active_fee_current  # hold ema and dynamic_fee; rotate grace window

    p25_current = percentile_25(rates)
    p25_ema = (ALPHA_N * p25_current + (ALPHA_DEN - ALPHA_N) * state.p25_ema_previous) //

    if (p25_ema > PERSISTENCE_THRESHOLD and
            state.p25_ema_previous > PERSISTENCE_THRESHOLD):
        candidate = (K * p25_ema) // RATE_SCALE
    else:
        candidate = state.dynamic_fee_previous

    cap_base = max(state.dynamic_fee_previous, STATIC_FEE)
    max_increase = cap_base + (cap_base * R_NUM) // R_DEN
    dynamic_fee = min(candidate, max_increase)
    dynamic_fee = max(0, dynamic_fee)
    active_fee = max(STATIC_FEE, dynamic_fee)
```

```
    state.p25_ema_previous = p25_ema
    state.dynamic_fee_previous = dynamic_fee
    state.active_fee_previous = state.active_fee_current
    state.active_fee_current = active_fee
    return active_fee

def required_fee(block_height, period_start, state):
    if block_height < period_start + G:
        return state.active_fee_previous
    return state.active_fee_current

def is_valid_transaction(tx, block_height, period_start, state):
    if tx.is_coinbase:
        return True
    return tx_fee(tx) >= required_fee(block_height, period_start, state) * len(tx.outputs)
```

Detailed test vectors, integer arithmetic precision requirements, and edge-case handling will be provided in a future numbered BIP submission.

Calibration Checklist

This checklist is separate from and follows the static fee BIP's calibration. The static fee BIP's calibration must pass before this checklist begins.

A. Dynamic Parameter Sensitivity

- Sweep `alpha_n` in {50, 100, 200, 300} × `R` in {1/10, 1/4, 1/2} × `k` in {2, 3, 4} against full chain history.
- For each combination report: number of periods where dynamic fee exceeds static fee; maximum dynamic fee reached; EMA lag to a sustained 10x fee increase; periods required to ratchet 2x under simulated sustained fee pressure.
- Recommend a default parameter tuple only after sensitivity tables exist. Prefer slow `alpha_n` unless tables show a real benefit from faster response.
- Calibrate `persistence_threshold` above the noise floor of quiet periods and below the movement threshold for genuine sustained congestion.
- Confirm the dynamic fee sits visibly above the static fee during normal and high fee conditions at the recommended parameters.

B. Anti-Manipulation Stress Test

- Simulate a moderately funded attacker sustaining elevated p25 fee rates for 2, 4, 8, and 16 consecutive 288-block windows. Report dynamic fee trajectory and cost to the attacker at each duration.
- Confirm the rate-of-change cap prevents rapid ratcheting even under sustained pressure.
- Confirm the dynamic fee falls freely back to the static fee floor once pressure is removed, with no persistent elevation.

C. Collateral Damage at Dynamic Rates

- Lightning channel opens: confirm the active fee remains negligible at the 95th and 99th percentile of simulated dynamic fee values under historical and projected fee conditions.
- Exchange batch payouts: same analysis.
- Confirm the dynamic fee does not price out legitimate high-output transactions during fee spikes driven by monetary activity rather than spam.

D. Consensus Edge Cases

- Specify behavior on reorgs crossing 288-block period boundaries: which period's EMA state and active fee apply to disconnected blocks.
- Specify IBD and assumeutxo interaction: EMA state must be fully reconstructable from block history alone without mempool history.
- Specify integer arithmetic precisely (milli-sat/vB rates, EMA division toward zero, `k × ema / 1000`, rate cap via `cap_base = max(previous, static_fee)`) so all implementations agree bit-for-bit.
- Define the identical percentile algorithm specified in Step 1 (sort ascending, index `floor((N − 1) × 25 / 100)`, no interpolation).
- Confirm grace window `G` is sufficient at period boundaries given historical mempool boundary-crossing data.
- Confirm periods are genesis-aligned `[n×288, (n+1)×288)` and the first full period after activation is the first computation window.

E. Exit Criteria

- Recommended (`k`, `alpha_n`, `R_NUM/R_DEN`, `persistence_threshold`, `G`, `min_tx_count`) published with full sensitivity tables.
- Anti-manipulation stress test B passes at recommended parameters.
- Collateral damage checklist C passes at recommended parameters.
- Consensus edge cases in checklist D resolved and written into Specification.

Security Considerations

Fee rate manipulation. Moving the 25th percentile meaningfully requires sustaining elevated transaction volume across a full 288-block window. The three-layer dynamic architecture (EMA, persistence, rate cap) makes rapid ratcheting slow, expensive, and publicly visible on chain. A state-level actor treating fee spend as an operating cost can still move the dynamic fee upward over many periods; the cap bounds how fast, and the fee falls freely once pressure is removed.

Static fee as irreducible floor. The dynamic fee can never push the active fee below the static fee. The static fee BIP's security properties are fully preserved regardless of what the dynamic layer does.

Low-activity window. If the 288-block sample contains fewer than `min_tx_count` fee-paying transactions, EMA and `dynamic_fee` are held, not zeroed. Zeroing would re-introduce a from-zero rate-cap trap. The static fee still floors `active_fee`.

Period boundary floor. The grace window keeps the required per-transaction floor unambiguous at every period boundary. The previous period's `active_fee` governs during the grace window; the new rate governs after. Coinbase value is not part of this BIP.

Rate cap from zero. `cap_base = max(dynamic_fee_previous, static_fee)` so a zero previous dynamic fee cannot pin `max_increase` at 0.

Reorg safety. On a reorg crossing a 288-block period boundary, the EMA state must revert to the correct state for the reorganized chain tip. Implementations must cache EMA state snapshots at every period boundary. The consensus rule is: retain snapshots for the most recent 2,016 blocks (one difficulty adjustment period, about two weeks) of period boundaries. That bounds the cache to at most seven snapshots at any time and covers any reorg that Bitcoin's proof-of-work economics makes plausible, without relying on heuristic depth estimates. A reorg deeper than 2,016 blocks is treated as a chain split requiring operator intervention, consistent with existing Bitcoin consensus assumptions. Implementations that do not retain the required snapshots cannot correctly validate blocks after a deep reorg and must re-sync from a known-good checkpoint.

References

- *BIP Pre-Proposal: Static Per-Output Miner Fee*. This BIP depends on it.
- *BIP Pre-Proposal: Permanent Data Channel Closure* (Josh / Secure Sovereign, August 2026).
- *Full Cost of Running a Bitcoin Node*, v2.4, July 2026.
- Mempool Research UTXO Set Report, May 2025, tip 892385 (research.mempool.space/utxo-set-report).
- *The Achievable Floor* (2026).
- *Bitcoin Is Not a Hard Drive* (2026).
- Lopp, Jameson. "Goldiblocks: A Dynamic Block Size Limit." OP_NEXT 2025.
- Various Delving Bitcoin and bitcoin-dev threads on UTXO bloat and spam mitigation (2023-2026)

Copyright

RELATED

Depends on Static Per-Output Miner Fee

Works with Permanent Data Channel Closure

Related writing

Full Cost of Running a Bitcoin Node · The Achievable Floor · Bitcoin Is Not a Hard Drive

SECURESOVEREIGN

Markdown is the canonical source: https://secsov.com/bips/dynamic-escalation-per-output-fee/index.md

Formatted snapshot of https://secsov.com/bips/dynamic-escalation-per-output-fee

Josh · Secure Sovereign