

DSM's Native Token Emissions

A High Level Explainer

Deterministic Join Triggered Emissions

a locked source vault, a proof carrying uniform lottery over sharded identities, sender pays credits, and no miner, clock, or coordinator

Brandon "Cryptskii" Ramsay

Irrefutable Labs Inc.

September 2026

Abstract

DJTE is how DSM's native units leave the vault they were born in. Every unit that will ever exist sits from the start inside a locked source vault under a committed policy, with a remaining counter that can only go down. Nothing is ever minted. What triggers a release is a device joining the network: when it passes the spend gate by paying the required set of storage replicas for its genesis storage, it produces a Join Activation Proof, and each such proof drives exactly one emission. The recipient is drawn uniformly at random over every activated identity, and the draw comes with a proof any device can check offline: a seed from committed state, exact rejection sampling, a descent through a count tree that routes a global rank into one shard, and an inclusion proof that the identity at that position is the winner. No miner, no block, no timer, no coordinator, and no global roster that anyone has to enumerate.

The paper proves the pieces a Bitcoin reader will ask about: determinism, single use of each join proof, exact equivalence of sharded sampling to sampling the whole population, a strict supply upper bound, and deterministic convergence of forks once the same transitions are seen. It also proves that choosing among 2^b shards needs b bits of information, so the descent cost is minimal, and it rejects random walks as a replacement because they introduce a policy knob and a bias.

The second half of the paper is about spam: DSM has no mempool to flood. Random bytes cannot bind to a precommitted chain; a device processes inboxes only for contacts it has established and can block one without touching the rest; a zero effect transition is not a transition; and a deterministic bundle of 1000 sender pays credits prices sustained valid traffic without a clock. This explainer accompanies the DJTE paper [1] and the DSM papers it rests on [2, 3]. It is written for a Bitcoin reader and organised around that reader's objections. Shaded boxes state how Bitcoin, proof of stake rewards, airdrops, and fee markets solve the corresponding problem and where this differs. Costs and boundaries are stated where they arise.

Contents

I	What Is Claimed, and Who Decides	3
1	The Central Question	3
2	The Entire Idea in One Diagram	4
3	Six Answers Before the Details	5
4	The Source Vault Is Not a Mint	5
5	The Join Is the Trigger	6
6	Nobody Runs the Lottery	7
7	Storage Nodes Are Mirrors; Users Do Nothing	7
II	The Construction	7

8	Identities, Shards, and the Activation Accumulator	8
9	The Count Tree	8
10	Drawing the Winner	8
11	Consuming the Join Proof Exactly Once	9
12	The Supply Schedule	9
13	What a Verifier Checks	10
14	Forks and Convergence	11
 III Attacks, and Spam		11
15	Identity Grinding and Shard Targeting	11
16	Replaying a Join Proof	12
17	The Bootstrap: Zero or Few Identities	12
18	Why Random Walks Are Rejected	12
19	Meaningless Spam: Precommitment and Contact Gating	12
20	Valid Flooding: Sender Pays Credit Bundles	13
21	Parsing Floods: Staged Verification and Bounded Inboxes	14
22	Attack Matrix	14
 IV Assessment		15
23	Relation to Everything Else That Works	15
24	What Is Not Introduced, and What Is Not Claimed	15
25	The Complete Causal Architecture	16
26	Final Invariant	16
27	Conclusion	16

Part I

What Is Claimed, and Who Decides

The Bitcoin reader's objections, in the order they occur. Everything here is stated normatively in the DJTE paper; the mechanism is built in Part II and attacked in Part III.

1 The Central Question

Every token system has to answer one question about its own units:

Where do the units come from, who decides who gets them, and what stops that party from printing more?

Bitcoin's answer is mining: a coinbase per block, a fixed schedule, halvings, and proof of work as the lottery. Every node checks the schedule. That answer needs a chain, a block cadence, and a clock. DSM has none of those, so it needs a different one.

DSM answers:

All units already exist, locked in a source vault whose remaining balance only decreases. A device joining the network, and paying its storage replicas to do so, produces one join proof; one join proof drives one emission. The recipient is drawn uniformly over every activated identity by a lottery nobody runs, whose seed is committed state and whose every step carries a proof.

The rule everything follows from.

emission is a vault transition, not minting; the trigger is a join; the draw is a proof.

A verifier accepting an emission does what a verifier accepting any vault transition does [3]: checks that the vault's generation was consumed once, that the remaining counter went down by exactly the amount that left, and that the guard on the branch, here a fresh join proof and a correctly derived winner, was satisfied.

Two jobs, never conflated. *Mechanical fairness* is the question *does every activated identity have exactly probability $1/N$ per emission?* That is a theorem of the count tree descent and holds regardless of how identities are distributed across shards.

Social fairness is the question *is it worth grinding out ten thousand identities to win ten thousand tickets?* That is not DJTE's question. It belongs to DSM's spend gate, which makes an identity cost three storage receipts before it can do anything. DJTE is mechanically correct under Sybil behaviour and says so; the paper delegates the economics explicitly.

Bitcoin Comparison

Bitcoin's issuance lottery is weighted by hashpower, run by block cadence, and capped by a schedule every node enforces. Your chance of the next coinbase is proportional to the work you burn.

DJTE's issuance lottery is weighted uniformly over activated identities, run by device joins, and capped by a decreasing counter inside a vault that every verifier enforces. Your chance of the next emission is one in N , and the only way to raise it is to become several identities.

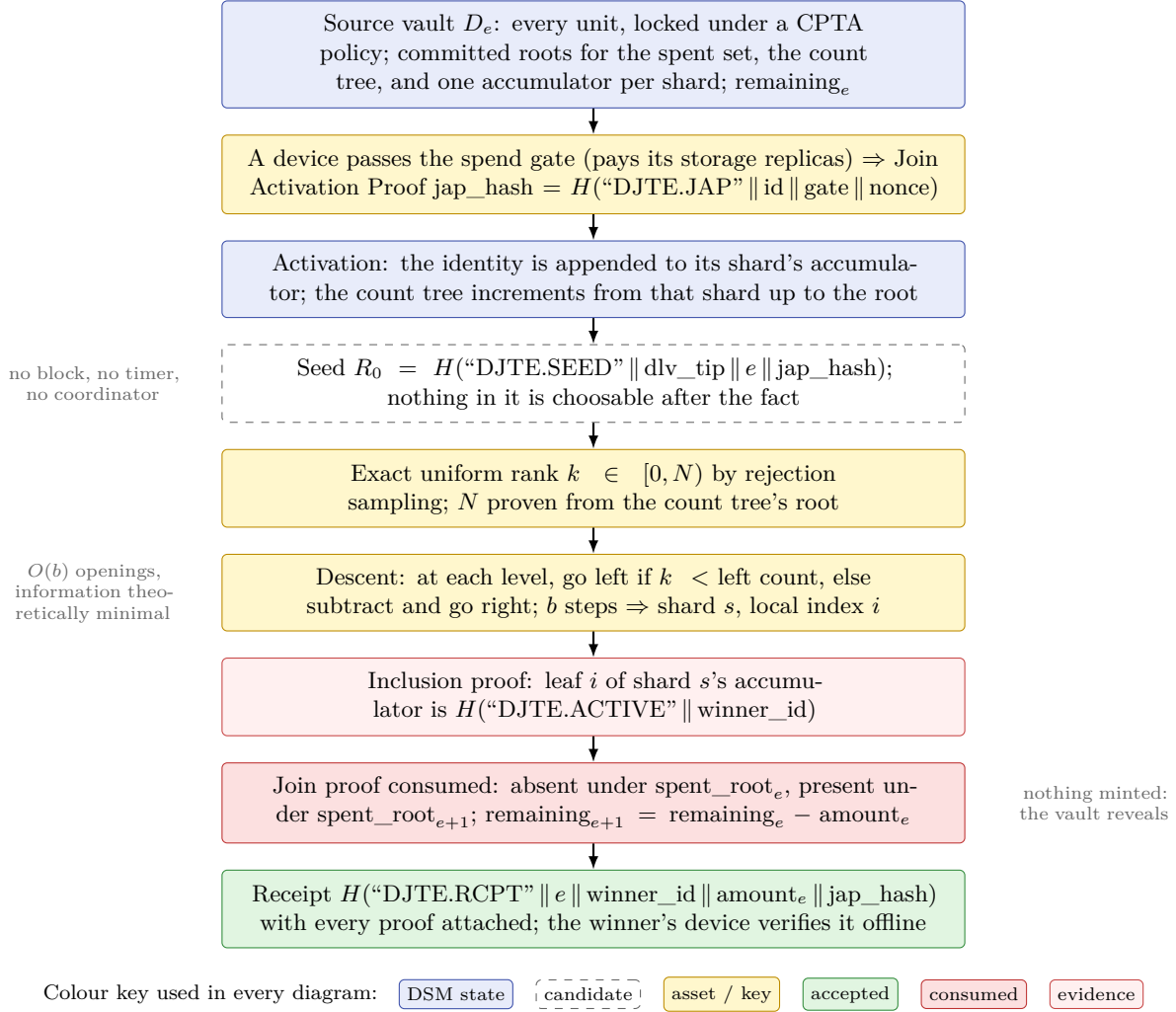
Architectural consequence:

There is no miner to bribe, no timing to manipulate, and no party who could emit more than the counter allows. The lottery is a deterministic function of committed roots and one consumed join proof.

Tradeoff / boundary:

Uniform over identities is only as fair as identity creation is expensive. DJTE remains correct under identity grinding; whether grinding pays depends on the spend gate charging more than an emission is worth. That is DSM's spend gate property, stated here as an assumption, not DJTE's theorem.

2 The Entire Idea in One Diagram



Three features of that picture carry the design.

The vault is above the line. Every unit exists before the first join. The remaining counter starts at the total and can only decrease; when it reaches zero the vault emits zero forever. That is the supply cap argument.

The join is the only trigger. No block, no timer, no coordinator produces an emission. A device paying its storage replicas produces one join proof, and that proof is consumed exactly once, by the same absent before, present after discipline used for every one time item.

Every step of the draw is an opening a verifier recomputes. The seed is a hash of committed state. The rank is exact. The descent is b Merkle openings. The winner is an inclusion proof. Two verifiers holding the same committed state compute the same winner without consulting anyone.

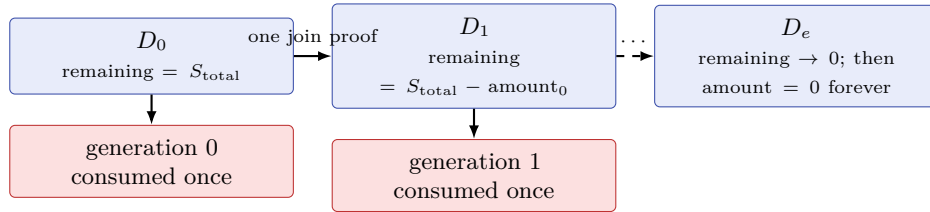
3 Six Answers Before the Details

The question the reader has	The answer, and where it is earned
Where do the units come from?	A locked source vault that holds every unit from the start. Emission moves units out; nothing is created. The remaining counter is the cap. (Sections 4, 12)
What triggers an emission with no block?	A device joining: passing the spend gate yields a Join Activation Proof, and each proof drives exactly one emission. The rate is set by joins and nothing else. (Sections 5, 11)
Who runs the lottery?	Nobody. Seed from committed state, exact uniform rank, count tree descent, inclusion proof. Every step is verified by whoever receives the receipt. (Sections 6, 10)
How is it uniform if the data is sharded?	Because ranks are routed by counts. A shard with more identities gets proportionally more draws; an empty shard gets none. The paper proves this equals sampling the concatenated global list. (Sections 9, 10)
What stops flooding, with no mempool and no fee market?	Random bytes cannot bind to a precommitted chain; inboxes process only established contacts; a zero effect step is not a step; and 1000 sender pays credits per bundle price sustained valid traffic. Receiving never debits. (Sections 19 to 21)
What if two devices see different orders?	States form a DAG. The canonical tip is the highest emission index, ties broken by the smallest vault tip. Two verifiers with the same set of valid states compute the same tip. (Section 14)

Just enough notation. *Source vault state* D_e . At emission index e : the vault tip, the spent set root, the count tree root, one accumulator root per shard, and remaining_e . *Shard*. $\text{Shard}(\text{id})$ is the first b bits of $H(\text{"DJTE.SHARD"} \parallel \text{id})$; the paper recommends $b \in [16, 20]$. *JAP*. The join proof: identity, evidence the replica payments were made, and a per activation nonce derived from the identity's chain tip, hashed together. *Count tree*. A binary tree over shard prefixes; each node stores how many activated identities sit under that prefix; the root stores N . *Accumulator*. Per shard, an append only Merkle structure whose leaves are activated identities, with inclusion proofs by index. *Schedule II*. $(S_{\text{total}}, b, E=16, M_0, r_0)$: epoch i allows $2^i M_0$ emissions of $\lfloor r_0/2^i \rfloor$ each.

4 The Source Vault Is Not a Mint

The word *emission* suggests new coins being struck. The source vault is not that. It is a Deterministic Limbo Vault under a CPTA policy, already holding every unit, with an emission branch whose guard is a fresh join proof and a correctly derived winner.



Each emission is one vault transition consuming one generation. The remaining counter is a field of the committed state, updated by

$$\text{remaining}_{e+1} = \text{remaining}_e - \text{amount}_e, \quad \text{amount}_e \leq \text{remaining}_e,$$

and a transition that violates either half is simply invalid. The paper's supply lemma is short: the counter starts at the total and never goes negative, so the distributed total never exceeds S_{total} .

Key Idea

The cap is not a rule nodes agree to enforce. It is arithmetic inside the vault's own committed state, checked by every verifier of every receipt. There is no context in which more can leave than remains, because every context is a vault transition.

Bitcoin Comparison

Bitcoin's twenty one million cap holds because every node checks every coinbase against the schedule, and a block that pays too much is rejected by all of them.

DJTE's cap holds because every emission is a transition over a vault whose remaining balance is committed state, and a transition that decrements by more than it has is not derivable. The check is per receipt, by the recipient, not per block, by the network.

Architectural consequence:

A verifier with the vault's committed roots and one receipt can confirm the cap held for that emission without a chain, a block height, or anyone else's word.

Tradeoff / boundary:

The verifier confirms *that* emission. Confirming the whole history is walking the receipts, which is why the paper spends a section on bounded proof size and pruning (Section 13).

5 The Join Is the Trigger

Bitcoin has a block to trigger issuance. There is no block here, so issuance is tied to the one event every participant performs exactly once: joining.

A device joins by passing the spend gate, which means paying a flat rate to the required number of distinct storage operators for its genesis storage [3]. When it does, it produces a Join Activation Proof committing to its identity, the evidence of those payments, and a nonce derived from its own chain tip. The proof's hash is a one time item. Consuming it is what advances the source vault.



Two consequences follow. The emission *rate* is set by how many devices join, and by nothing else; there is no difficulty adjustment because there is no target cadence. And the join proof is the same shape as every other one time item in DSM: it is proved absent from a spent set before, and present after, so it can drive at most one emission (Section 11).

Bitcoin Comparison

A Bitcoin block is produced roughly every ten minutes by whoever wins the work race; issuance is decoupled from who uses the network.

A DJTE emission is produced exactly when a device joins; issuance is coupled to adoption and to nothing else.

Architectural consequence:

No block cadence, no difficulty adjustment, no miner. The supply curve is a function of joins against a fixed halving shaped schedule.

Tradeoff / boundary:

If nobody joins, nothing is emitted; if joins are bursty, emissions are bursty. DJTE does not smooth this, because smoothing would need a clock.

6 Nobody Runs the Lottery

There is no party who picks the winner. Every input to the draw is already committed and every step is an opening.

The seed is $R_0 = H(\text{"DJTE.SEED"} \parallel \text{dlv_tip} \parallel e \parallel \text{jap_hash})$. The vault tip is committed; the index is the next integer; the join proof hash is the one being consumed. Nothing in R_0 can be chosen after the fact by the joiner, by a storage node, or by a previous winner. The rank k is drawn from R_0 by exact rejection sampling: if R_0 falls outside the largest multiple of N below 2^{256} , it is rehashed and retried, so there is no modulo bias. The descent and the inclusion proof are Merkle openings the verifier recomputes.

Key Idea

Two verifiers holding the same committed vault state and the same consumed join proof compute the same winner. That is the paper's determinism theorem, and it is what "nobody runs the lottery" means: the winner is a function, not a decision.

Proof of Stake Comparison

Proof of stake reward selection is typically weighted by stake and seeded from a randomness beacon that validators contribute to and can, at the margin, bias by withholding.

DJTE selection is weighted uniformly over identities and seeded from a hash of committed state plus the consumed join proof. There is no beacon, no contributor, and no withholding surface.

Architectural consequence:

The only way to influence who wins is to change N by joining, which costs the spend gate fee and gives you one ticket in N .

Tradeoff / boundary:

There is also no way to *decline* to be in the draw; every activated identity is a candidate for every emission until the vault is empty. A device that joined and vanished still occupies its slot.

7 Storage Nodes Are Mirrors; Users Do Nothing

The paper states who acts.

End user devices create and hold identities, produce and verify proof carrying objects, and maintain local deterministic state. Users approve actions in the wallet. They do not manage shards, select winners, run an ordering protocol, or perform manual cryptography. From the user's side, DJTE "just happens": unlock the spend gate once, receive receipts in the inbox when online, and see the balance update after local verification.

Storage nodes store and relay objects by content address and may reject malformed protobuf. They do not vote, order, sign, or decide validity. They are not trusted for correctness.

Policy anchors are immutable content addressed policy objects (CPTA, DLV references) that devices read during verification. They are inputs, not authorities.

Bitcoin Comparison

A Bitcoin full node validates every block; a miner orders transactions; a user waits for confirmations. Three roles, all load bearing.

In DJTE the storage node holds bytes, the device verifies, and there is no orderer. The user's only protocol act is paying to join.

Architectural consequence:

There is no role to capture. A storage node that lies is caught by the content address; a storage node that withholds is an availability problem; neither can mint, pick a winner, or spend a join proof.

Tradeoff / boundary:

"No global roster" also means no global view. A device learns about emissions it is not party to only by fetching receipts, and Section 14 is about what happens when two devices have fetched different sets.

Part II

The Construction

The objects the paper defines (shards, accumulators, the count tree, the spent set) and the algorithms over them, in the order they are built.

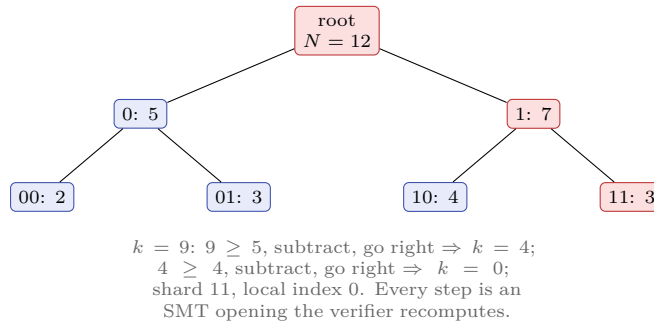
8 Identities, Shards, and the Activation Accumulator

An identity is a fixed length byte string, in practice the hash of a genesis anchor. Its shard is the first b bits of $H(\text{"DJTE.SHARD"} \parallel \text{id})$, which partitions identities into 2^b shards deterministically and without anyone assigning anything.

Each shard s keeps a Shard Activation Accumulator: an append only Merkle structure (a Merkle Mountain Range qualifies) whose leaf at index i is $H(\text{"DJTE.ACTIVE"} \parallel \text{id}_i)$ for the identity activated at position i in that shard. It commits to an ordered list and yields an inclusion proof that a given leaf sits at a given index. Activating an identity appends one leaf to one shard's accumulator and increments one path in the count tree. No global list is touched, because no global list exists.

9 The Count Tree

The count tree is what lets a verifier sample uniformly over N identities without ever seeing a list of N identities. It is a complete binary tree over shard prefixes: the node for prefix p stores how many activated identities have a shard beginning with p ; leaves are the 2^b shards; the root stores N . Every node is committed in a Sparse Merkle Tree keyed by prefix, so any node's count can be opened against the root.



Key Idea

Ranks are routed by counts. A shard with more identities receives proportionally more of the rank space; an empty shard receives none. That is not a skew to correct. It is what “uniform over individuals” means. The paper proves the descent is equivalent to indexing into the concatenation of all shard lists in shard order.

10 Drawing the Winner

At emission index e , with vault state D_e and a join proof jap_hash being consumed:

1. **Seed.** $R_0 = H(\text{"DJTE.SEED"} \parallel \text{dlv_tip} \parallel e \parallel \text{jap_hash})$.
2. **Total.** Open the count tree's root to obtain N . If $N = 0$, the winner is undefined and the transition must set $\text{amount}_e = 0$ (Section 17).
3. **Rank.** $k = \text{UniformIndex}(R_0, N)$: exact rejection sampling with domain separated reseeding, so k is exactly uniform on $\{0, \dots, N - 1\}$.

4. **Descent.** From the root with rank k : at each depth, open the two children; if k is less than the left count, go left; otherwise subtract the left count and go right. After b steps the prefix is a shard s and the residual k is a local index i .
5. **Inclusion.** Open shard s 's accumulator at index i ; the leaf there must be the hash of the winner, $H(\text{"DJTE.ACTIVE"} \parallel \text{winner_id})$.

The receipt carries every opening. A verifier that holds D_e 's roots recomputes steps 1 to 5 and either arrives at the same identity or rejects.

Bitcoin Comparison

A Bitcoin coinbase names its recipient by the miner's choice; the lottery decided *whether* the miner could write a block, not *whom* the block pays.

A DJTE receipt names a recipient the joiner did not choose and cannot influence beyond adding one identity to N . The lottery decides the recipient directly, and the receipt proves it.

Architectural consequence:

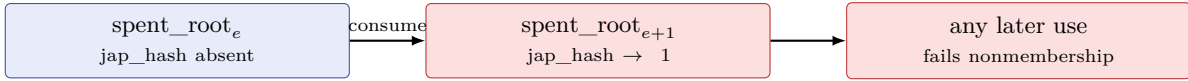
The joiner is not the winner. Joining is what makes an emission happen; the units go to a uniformly drawn activated identity, which may be anyone including the joiner with probability $1/N$.

Tradeoff / boundary:

The joiner pays the spend gate fee and receives, in expectation, $1/N$ of one emission. Joining is not proposed as an investment. Joins are the trigger.

11 Consuming the Join Proof Exactly Once

The spent set is a Sparse Merkle Tree mapping $\text{jap_hash} \rightarrow 1$ for every consumed proof. A transition with nonzero emission must carry a nonmembership proof of the join proof's hash under spent_root_e and a membership proof under spent_root_{e+1} .



This is the paper's single use theorem, and it is the same argument the state machine makes for every one time item: absent before, present after, and the "after" is committed permanently, so a second consumption from the same prior state cannot supply the nonmembership proof it needs.

12 The Supply Schedule

The schedule is a policy tuple $\Pi = (S_{\text{total}}, b, E, M_0, r_0)$ with $E = 16$ epochs. Epoch i allows at most $M_i = 2^i M_0$ emissions of $r_i = \lfloor r_0/2^i \rfloor$ units each: twice as many emissions, half the size, per epoch. The shape is familiar.

Epoch	Max emissions	Per emission	Note
0	M_0	r_0	
1	$2M_0$	$\lfloor r_0/2 \rfloor$	
2	$4M_0$	$\lfloor r_0/4 \rfloor$	
$\vdots$	$\vdots$	$\vdots$	
15	$2^{15}M_0$	$\lfloor r_0/2^{15} \rfloor$	last epoch
any	(as above)	$\min(r_{\text{epoch}(e)}, \text{remaining}_e)$	the counter always wins

The last row matters most. Whatever the epoch says, $\text{amount}_e = \min(r_{\text{epoch}(e)}, \text{remaining}_e)$, and once remaining is zero the vault emits zero forever. The schedule shapes the curve; the counter caps it, independently of how many devices join.

Bitcoin Comparison

Bitcoin halves every 210,000 blocks, which is a clock in disguise: block height stands in for time, and the difficulty adjustment keeps the two roughly aligned.

DJTE halves every M_i emissions, which is a count of joins. No height, no time, no adjustment.

Architectural consequence:

The halving schedule is a pure function of the emission index. Two verifiers never disagree about which epoch an emission is in.

Tradeoff / boundary:

Epochs are not time boxed. If joins are slow, an epoch is long in wall clock terms; if fast, short. DJTE does not care, and cannot, because it has no clock to care with.

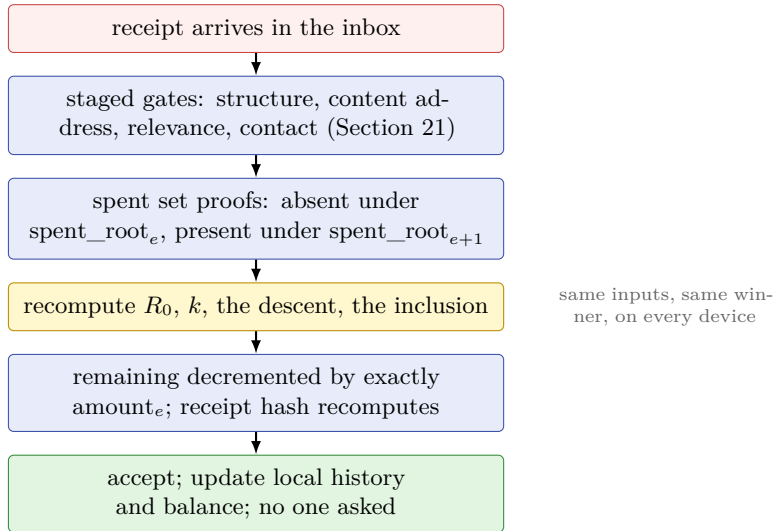
13 What a Verifier Checks

A candidate next state D_{e+1} is valid if and only if all of the following hold:

1. the index increments: $e_{\text{next}} = e + 1$;
2. the spent set was updated correctly: spent_root_{e+1} is spent_root_e plus the insertion of jap_hash , and both proofs verify;
3. the winner was selected correctly: the rank derivation and descent openings verify under count_root_e , and the accumulator inclusion verifies under $\text{acc_root}_{e,s}$;
4. the supply updated: $\text{remaining}_{e+1} = \text{remaining}_e - \text{amount}_e$ with $\text{amount}_e \leq \text{remaining}_e$;
5. the receipt binds: rcpt_e matches the provided values and proofs.

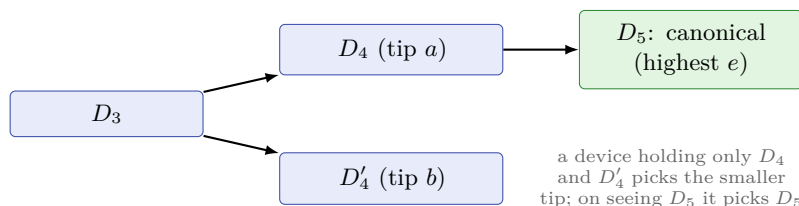
Verification is strict fail: a missing proof or a mismatch is invalid, with no best effort acceptance.

Proof size. A receipt carries the two spent set proofs, the descent openings, and one accumulator inclusion: $O(\log(\text{SMT depth})) + O(b) + O(\log |L_s|)$, independent of N . The descent openings pack into one multiproof, meaning the revealed node values plus the minimal sibling set needed to recompute the root, rather than b separate witnesses. Devices batch verify and cache count tree nodes, which change only when an activation lands under that prefix. History grows; one receipt stays bounded, and DSM can prune raw receipts under policy while retaining committed roots.



14 Forks and Convergence

Devices may observe emission transitions in different orders, or observe different subsets. The paper does not pretend this away. Valid states form a DAG: $D_e \rightarrow D_{e+1}$ is an edge when the transition verifies, and forks can exist. The canonical tip rule is a pure function of the set of valid states a device holds: choose the tip with the highest emission index, and break ties by the lexicographically smallest vault tip.



Key Idea

DJTE is not consensus. Two devices with different sets of valid states may hold different tips; two devices with the same set hold the same tip, because the rule reads nothing but the set. Convergence follows from sharing views, not from agreeing.

Bitcoin Comparison

Bitcoin resolves forks by most accumulated work, and nodes reach the same chain because they eventually see the same blocks and apply the same rule.

DJTE resolves forks by highest index then smallest tip, and devices reach the same tip because they eventually see the same states and apply the same rule. The rule is cheaper; the assumption that views are eventually shared is the same.

Architectural consequence:

No device needs a global view to act. It verifies what it has, picks a tip by a pure function, and updates when it learns more.

Tradeoff / boundary:

Until two devices share a view, they may disagree about which emission at index e is canonical. The paper states this as what DJTE is: a mechanical fairness mechanism, not an agreement protocol.

Part III

Attacks, and Spam

Identity grinding, replaying a join proof, the empty population bootstrap, why random walks are rejected, and then the second half of the paper: why there is no mempool to flood.

15 Identity Grinding and Shard Targeting

The paper names the one real fairness risk: malleable identity creation. An attacker who can create identities cheaply can create many, and each is one ticket. An attacker who can also steer which shard an identity lands in gains nothing extra, because shards are not prizes, ranks are routed by counts, and a heavily populated shard is as fair per identity as an empty one. Cheap identities are the whole game.

The mitigation is not in DJTE. It is DSM's spend gate: an identity costs three storage receipts before it can write a byte, and device and identity binding raise the cost of manufacturing identities at scale. DJTE is mechanically correct under Sybil behaviour; the social meaning of "fair" depends on the gate.

Bitcoin Comparison

Bitcoin’s lottery cannot be gamed by Sybils because tickets are hashpower, not identities; a thousand miners with one hash each are one hash.

DJTE’s lottery can be gamed by Sybils exactly to the extent that identities are cheap, and it says so.

Architectural consequence:

The economic question “is an identity worth more than an emission?” is the spend gate’s to answer, and the answer can be tuned by policy without touching DJTE.

Tradeoff / boundary:

If the gate is ever cheaper than an emission, grinding pays, and DJTE’s uniformity becomes uniformity over grinders.

16 Replaying a Join Proof

A joiner keeps a copy of its join proof and submits it again. The second transition must prove the hash absent under the current spent root; it is present, so the proof cannot be constructed, and any verifier rejects. This holds along one lineage. Across a fork, meaning two states descending from the same D_e that each consume the proof, both are individually valid, and the canonical tip rule picks one; the loser’s emission never lands in any device’s canonical history once views are shared.

17 The Bootstrap: Zero or Few Identities

At start, N is small or zero. If $N = 0$, winner selection is undefined and the emission transition must set $\text{amount}_e = 0$; any nonzero winner claim is invalid because it cannot supply the rank mapping and inclusion proofs. If $N > 0$ but most shards are empty, the descent routes ranks only into nonempty subtrees; empty shards get exactly zero probability, which is correct. No special coordination is needed, only strict fail handling of $N = 0$.

18 Why Random Walks Are Rejected

A natural suggestion is to replace the b step descent with a random walk over shards. The paper rejects it twice. First, choosing among 2^b shards requires b bits of decision information, so an $O(b)$ descent is information theoretically minimal and no cheaper proof carrying scheme exists. Second, a random walk needs a step policy and a mixing guarantee; exact global uniformity is then not proof carrying without heavy extra machinery, and “walk until mixed” introduces an implicit iteration parameter, a policy knob, and bias if mixing is insufficient or adversarially structured. The linear in b cost is not a flaw; it is the price of selecting among 2^b possibilities with a proof, and the right response is to compress the witnesses, not to change the semantics.

19 Meaningless Spam: Precommitment and Contact Gating

A reader will ask what stops flooding in a system with no mempool and no fee market. Most of the answer is structural.

Precommitment. A DSM transition is not an arbitrary message. A receiver’s relationship specific chain evolves only by transitions that match the expected commitment context: known tips, expected indices, prior receipts, policy anchors. The paper defines a relevance predicate $\text{Rel}(\text{obj}, \text{ctx})$ that checks this before any deep proof work; an attacker lacking the receiver’s context produces objects that satisfy it with negligible probability. Random bytes cannot be appended to a precommitted chain.

Contact gating. A device processes inbox objects only for relationships it has established. The paper’s predicate Allow is derived from the receiver’s own local contact set and nothing else. A device can be online without syncing any particular contact’s stream, and can block a contact and stop ingesting their objects entirely, without affecting any other relationship or global correctness. The paper’s lemma: if the victim disallows the attacker, no per contact processing cost is incurred, online or not.

You cannot send zero. A transition is state advancing only if it is sender authored and strictly advances the sender’s chain tip, with no duplicates, no replays, and no zero effect updates. “Send nothing a million times” is not representable as an accepted object.

Offline bilateral transfers are naturally spam resistant: proximity is required, both parties consent and validate, and there is no broadcast surface.

Mempool Comparison

A blockchain exposes one global pending queue. Anyone can submit transactions that cost every node bandwidth and validation even if never included, and congestion, censorship by spam, and MEV all live there.

DSM has no shared queue. Relevance is relationship specific, processing is per contact and optional, and irrelevant objects fail before the expensive checks.

Architectural consequence:

“Spamming the network” is not a globally amplifying act, because there is no global surface it lands on. The residual risk is localized pressure on storage and relay capacity from *valid* traffic.

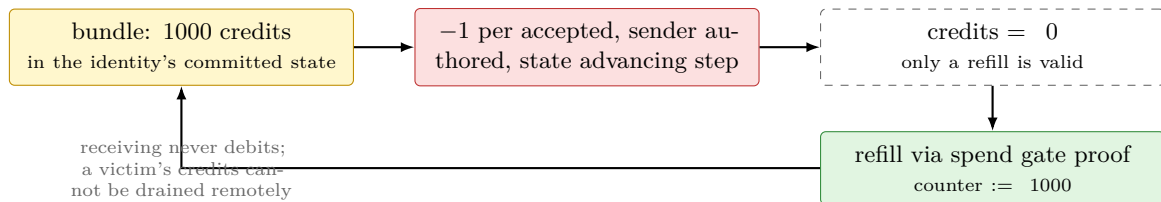
Tradeoff / boundary:

That residual is real, and it is what the credit bundle in the next section prices.

20 Valid Flooding: Sender Pays Credit Bundles

Even with the structure above, a well funded party can generate valid, state advancing traffic against storage and relay capacity. The credit bundle is a deterministic economic backstop, with no time window and no storage node discretion.

The policy $\Pi_{\text{cred}} = (C_{\text{bundle}}, \text{DebitRule}, \text{RefillRule})$ is a constant anchored in a CPTA policy; the baseline is $C_{\text{bundle}} = 1000$. Each identity carries a credit counter inside its committed state. A transition debits exactly one credit if and only if it is sender authored, verifies and is accepted, and advances the sender’s chain tip. Inbound receipts and emissions, malformed objects, duplicates, replays, and non advancing objects debit zero. When credits reach zero, the only valid sender authored state advancing transition is a refill, which carries proof of satisfying the spend gate conditions and resets the counter to C_{bundle} .



Why 1000: high enough that typical humans never notice it; low enough that sustained valid flooding forces frequent refills, converting throughput abuse into a direct economic burn; automatable by heavy integrators because there is no monthly schedule; and enforceable by verifiers, not storage nodes, so no node can “grant someone a million.” The paper’s guidance is to start at 1000 and to change it only through policy anchors, as globally verifiable constants.

Fee Market Comparison

Bitcoin prices congestion with a fee market coupled to block cadence: pay more to get into the next block, and the price is set by a global queue and a clock.

DJTE prices the sender's own accepted steps with a fixed bundle and a refill, and nothing else is priced, because nothing else is shared.

Architectural consequence:

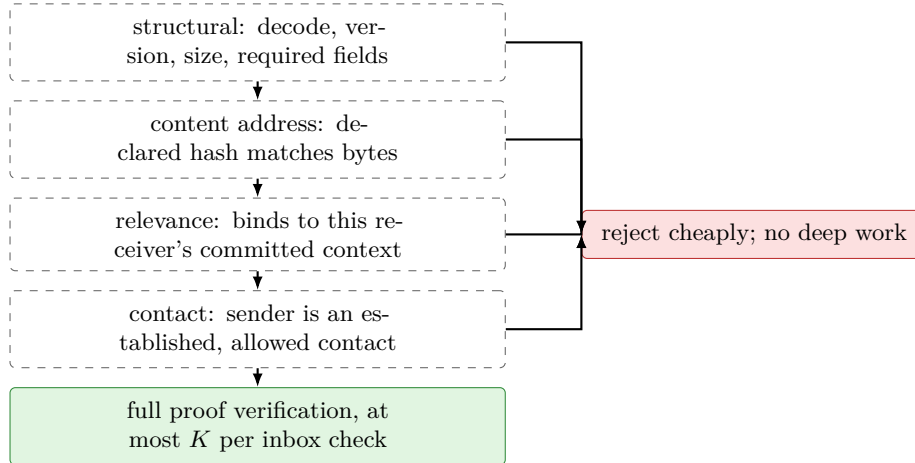
No “pay to jump the queue” semantics, because there is no queue. No timestamp games, because there is no time. A victim cannot be grieved into spending, because receiving is free.

Tradeoff / boundary:

A fee market lets urgent transactions outbid patient ones. DJTE offers no priority at all; every accepted step costs one credit, and there is nothing to outbid.

21 Parsing Floods: Staged Verification and Bounded Inboxes

The last vector is CPU: make devices and nodes parse and verify garbage. Verification is staged, cheap first and strict fail: (1) the structural gate: protobuf decode, envelope version, size bounds, required fields; (2) the content address gate: the declared hash matches the bytes; (3) the relevance gate: Rel = 1; (4) the contact gate: Allow = 1; (5) full proof verification only for survivors. And inbox processing is bounded without time: a device processes at most K candidate objects per inbox check action, a deterministic local constant, so one action performs $O(K)$ parsing and at most $O(K)$ deep verifications regardless of how many objects exist remotely.



22 Attack Matrix

Attacker does	DJTE outcome	What it rests on
replays a join proof	second transition fails nonmembership	spent set SMT
crafts identities to target a shard	no advantage; uniform per identity	count tree routing
grinds many cheap identities	many tickets, mechanically correct	spend gate cost (assumption)
runs storage nodes, withholds	availability only; cannot mint or pick	content addressing
floods random bytes	fails structural / relevance gates	precommitment
floods from an established contact	victim blocks the contact; no cost	contact gating
floods valid traffic at scale	burns 1000 credits per bundle, refills at the gate	credit policy
withholds states to fork	fork exists until views are shared	canonical tip rule
emits with $N = 0$	amount must be zero; nonzero claim invalid	strict fail

One row needs comment. The third row is the only one where the outcome is “correct but possibly unfair,” and the paper is direct that the fairness there is purchased by the spend gate, not by DJTE.

Part IV

Assessment

How this sits against everything that works, what is deliberately not introduced or claimed, and the whole design in one picture.

23 Relation to Everything Else That Works

	Trigger	Who picks the recipient	Weight	Clock in the rule	Spam surface
Bitcoin mining	block found	the miner, of itself	hashpower	height as time	global mempool, fee market
Proof of stake rewards	slot / epoch	beacon + stake	stake	yes	global mempool
Merkle airdrop	a claim tx	a published list	list membership	no	the chain’s
Faucet	a request	the operator	operator policy	usually	the operator
DJTE	a device joins	a proof carrying draw	uniform per identity	none	none global; contact gated, credit priced

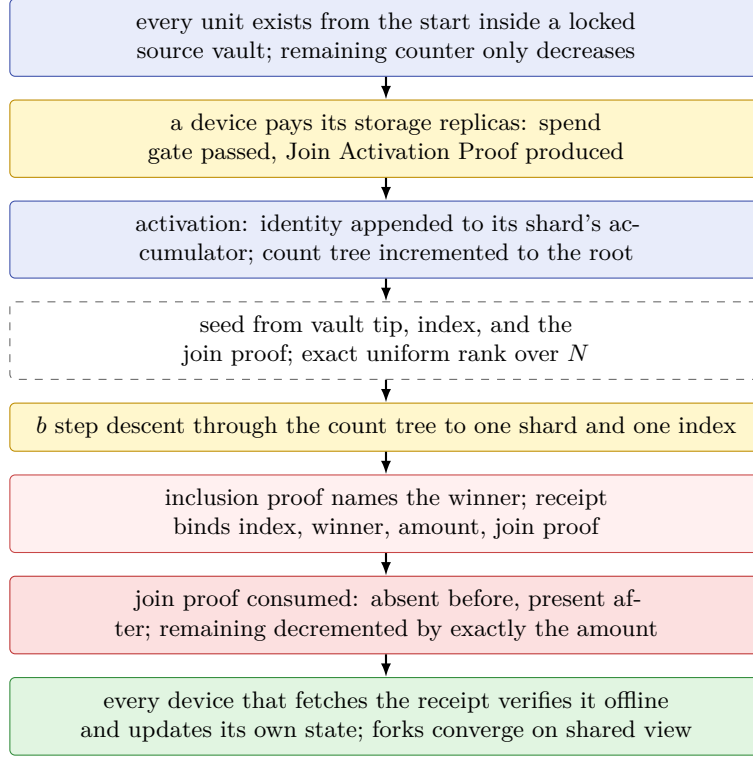
Mining pays work and needs a chain and a clock. **Proof of stake** pays stake and needs a beacon that contributors can lean on. **Airdrops** pay a list someone wrote. **Faucets** pay whoever an operator lets them pay. DJTE is the only row where the trigger is adoption, the recipient is a function of committed state, the weight is one per identity, and no clock or operator appears anywhere in the rule.

24 What Is Not Introduced, and What Is Not Claimed

Deliberately not introduced. No consensus and no leader. No wall clock time, timestamps, or block cadence. No miner, beacon, or coordinator. No global roster and no global roster scan. No global mempool and no fee market. No storage node signature or vote. No time based throttle; the credit bundle and the inbox bound are constants. No random walk selection. No best effort acceptance.

Deliberately not claimed. That DJTE is consensus; forks can exist until devices learn more states, and convergence is deterministic once the same set is known. That DJTE is a social oracle; it is a mechanical fairness mechanism under a Sybil resistant spend gate. That uniform over identities is fair if identities are cheap. That emissions are smooth in time; they follow joins. That a full history is free to verify; one receipt is bounded, and history is pruned under policy. That b can be made sublinear; it cannot without losing the information needed to choose a shard. Or that conformance follows merely because the primitives exist in a repository.

25 The Complete Causal Architecture



26 Final Invariant

A transition $D_e \rightarrow D_{e+1}$ with nonzero emission is valid only if

$$\begin{aligned}
 \text{Valid}(D_e \rightarrow D_{e+1}) \iff & e_{\text{next}} = e + 1 \\
 & \wedge \text{jap_hash} \notin \text{spent_root}_e \wedge \text{jap_hash} \in \text{spent_root}_{e+1} \\
 & \wedge \text{winner_id} = \text{SAA}_s[i] \text{ where } (s, i) = \text{Descent}(\text{UniformIndex}(R_0, N)) \\
 & \wedge \text{remaining}_{e+1} = \text{remaining}_e - \text{amount}_e \wedge \text{amount}_e \leq \text{remaining}_e \\
 & \wedge \text{rcpt}_e = H(\text{"DJTE.RCPT"} \parallel e \parallel \text{winner_id} \parallel \text{amount}_e \parallel \text{jap_hash}),
 \end{aligned}$$

with $R_0 = H(\text{"DJTE.SEED"} \parallel \text{dlv_tip} \parallel e \parallel \text{jap_hash})$ and N opened from count_root_e . Consequently: the same state and the same consumed join proof yield the same winner everywhere; no join proof drives two emissions in one lineage; the winner is uniform over the concatenation of all shard lists; the distributed total never exceeds S_{total} ; and two verifiers with the same set of valid states hold the same canonical tip.

Nothing is minted; the vault reveals.
Nobody runs the lottery; the join draws the ticket.
No clock, no mempool, no coordinator.

vault $\rightarrow$ join $\rightarrow$ activate $\rightarrow$ seed $\rightarrow$ rank $\rightarrow$ descend $\rightarrow$ prove $\rightarrow$ consume $\rightarrow$ receipt $\rightarrow$ verify

27 Conclusion

DJTE gives DSM an issuance mechanism that needs none of the things Bitcoin's needs. Every unit is already inside a locked source vault; a device joining the network is the only trigger; the recipient is drawn uniformly over every activated identity by a lottery whose seed is committed state and whose every step

is a Merkle opening; the join proof is consumed once by the same absent before, present after discipline used everywhere; and the supply cap is arithmetic inside the vault, checked by every verifier of every receipt. Sharded storage costs nothing in fairness, because ranks are routed by counts, and the $O(b)$ descent is the minimum information any proof carrying shard selection can require.

The second half of the design is that there is no mempool to flood. Random bytes cannot bind to a precommitted chain, a device processes only the contacts it chose and can block one without touching the rest, a zero effect step is not a step, and a fixed bundle of sender pays credits prices the residual, valid traffic at scale, without a clock or a node's discretion.

Its boundaries are stated. DJTE is not consensus; forks converge on shared view, not by agreement. It is not a social oracle; uniform over identities is fair only when identities cost more than emissions, and that cost is the spend gate's, not DJTE's. It is not smooth in time, because it has no time. Within those bounds, it is a proof carrying, deterministic issuance system that is structurally hostile to meaningless flooding.

References

- [1] B. Ramsay. *DJTE: Deterministic Join Triggered Emissions for the Deterministic State Machine. A Proof-Carrying, Consensus-Free Emission System with Shard-Equivalent Global Uniform Sampling, with Deterministic Credit Bundles, Contact-Gated Inboxes, and Precommitment-Driven DoS Minimality*. Irrefutable Labs Inc., December 2025.
- [2] B. Ramsay. *Deterministic State Machine: A Concise, Post-Quantum Specification*. Irrefutable Labs Inc.
- [3] B. Ramsay. *Deterministic State Machines Explained Mathematically: Bilateral State, Sparse Merkle Trees, Determinism, Precommitment, Linear Resources, Canonical State Chaining, and Conflict-Local Finality*. Irrefutable Labs Inc., 2026.
- [4] S. Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. 2008.