Hiding Amounts, Not the Money Supply

Hathor's shielded outputs hide every transaction amount — and anyone can still audit every token's total supply

How many of these exist?

It is the most basic question anyone can ask about a currency. On a transparent ledger, the answer is a loop: walk the UTXO set, add up the amounts. Hide the amounts and the loop breaks — as the field was just reminded when Zcash disclosed a four-year-old counterfeiting bug in its newest shielded pool, with *"no definitive way to determine, using only cryptography, whether such exploitation occurred."*[1]

Hathor's shielded outputs — on testnet today, not yet on mainnet — are built so that question never loses its answer:

> **Transaction amounts (and optionally which token is moving) are hidden behind cryptographic commitments — yet the total supply of every token remains a publicly auditable number, verifiable by anyone running a full node, with one curve-point comparison, at every block.**

Think of the ledger as a book of sealed envelopes. Every individual amount is sealed and stays sealed. Yet anyone can check, at any time and without opening a single envelope, that the sum of everything sealed equals exactly what publicly went in minus what publicly came out.

One scope note: this is **amount confidentiality, not anonymity**. Addresses and the transaction graph stay public — a deliberate trade, aimed at users whose transaction *graph* is public business anyway but whose *numbers* are nobody's business: payroll, B2B settlement, tokenized real-world assets, treasuries.

What an observer sees	Hathor shielded outputs
Amounts	**Hidden**
Token type	**Optionally hidden**
Sender address	Public
Transaction graph	Public
Token issuance (mint / melt)	Always public
Total supply of every token	**Publicly auditable at every block, one equation**

The audit, in one equation

Every shielded output hides its amount v inside a **Pedersen commitment** $C = v \cdot H_T + r \cdot G$, where H_T is a per-token generator anyone can recompute from the public token UID and r is a

[1] "The Orchard Counterfeiting Vulnerability — And Next Steps," Zcash Community Forum, June 2026 (`https://forum.zcashcommunity.com/t/the-orchard-counterfeiting-vulnerability-and-next-steps/56015`). The Orchard shielded pool activated in May 2022; the flaw — an under-constrained element of the Orchard circuit — was identified on May 29, 2026 by security researcher Taylor Hornby during a protocol review commissioned by Shielded Labs, and patched by an emergency upgrade.

random blinding factor. Commitments add up: the sum of commitments commits to the sum of the amounts. Three consensus rules do the rest:

1. **Mint and melt amounts are always public.** There is no such thing as a confidential mint on Hathor, so the expected supply of every token is a plaintext number at every block.
2. **Every transaction must balance as a curve equation** that every full node checks.
3. **Shielded value returns to transparent form only through an explicit, publicly declared step** — never silently.

From these, the audit is three steps: compute every token's expected net shielded amount from purely transparent data; turn those public numbers into an expected curve point; sum the commitments of the live shielded UTXO set and compare the two points.

One comparison. No view keys, no trusted auditor, no cooperation from any holder, no privacy lost — and anyone running a full node can run it continuously: explorers, exchanges, researchers, anyone.

That the comparison *must* pass — and that it certifies every token's supply simultaneously — is a theorem under standard, decades-old assumptions: the discrete logarithm problem on secp256k1 — the curve that has secured Bitcoin since 2009 — with the hash-derived generators and the Fiat–Shamir proofs modeled as random oracles. The core argument is one page of algebra, printed in the appendix below. If a flaw in this design exists, it is findable by checking that page against the published literature. We mean the audit literally: check our algebra.

Why trust it

There are two broad ways to build transaction privacy.

Prove a program. Encode the entire transaction validity rule as a custom zero-knowledge circuit, and have users prove they satisfied it. This is the approach of Zcash and other circuit-based privacy designs, and it is maximally private: amounts, addresses, and the transaction graph all disappear. But the circuit *is* the consensus rule. Write one constraint too loosely and the network will accept transactions that violate the rule you forgot — and zero knowledge guarantees that nobody can tell. That is not hypothetical; it is exactly the failure class behind the Orchard bug, shipped in 2022 and found in 2026. To be fair, Zcash made that trade deliberately — full graph privacy *requires* hiding the validity rule itself — and pairs it with turnstile accounting between its pools as defense in depth. Zcash hides the sender, the recipient, the amount, and the graph; the design described here hides the amount alone. The honest comparison is not "private versus auditable" but where each design puts its trusted computing base.

Check an equation. Hide only the amounts (and optionally the token type) inside commitments whose algebra the verifier checks directly. No circuit, no constraint system, no proving backend — a handful of fixed, well-studied equations over secp256k1.

Hathor's shielded outputs are the second lineage, deliberately. Hathor invented no new cryptographic primitives for this feature. The primitives — Pedersen commitments (1991), range proofs, and asset surjection proofs — are the Confidential Transactions construction running in production in `libsecp256k1-zkp` since 2018.

Hathor's contribution is the consensus policy on top: **no confidential issuance, ever, for any token.** Liquid's Confidential Assets *offers* confidential issuance as an option; Hathor *forbids* it, as a consensus invariant. The algebra enforces conservation; the policy is what turns conservation into a public supply audit covering every token on the network.

Less to audit

The claim here is narrow: *the failure class behind the Orchard bug — an under-constrained custom circuit — requires having a custom circuit, and Hathor has none.* It is not a claim of immunity from bugs in general; Bitcoin's CVE-2018-17144 inflation bug lived in fully transparent consensus code. Simplicity narrows the attack surface without abolishing it.

That said, the design's security story is unusually short:

> **Old assumptions, tiny statements, old code — and one equation.** Everything reduces to discrete log on secp256k1, with Fiat–Shamir in the random-oracle model. Every statement proven is a fixed one-liner, not an application-specific program. Every implementation had years of adversarial exposure securing other people's money before Hathor adopted it.

And because every commitment and proof is on chain and verification is deterministic public algebra, a bug in Hathor's *proof verifiers* — a range or surjection check that was wrong or skipped — would be retroactively detectable: the whole history can be re-validated with corrected code, identifying exactly which proofs should have been rejected. The one failure this does not cover is a break of the discrete-log assumption itself: Pedersen commitments hide unconditionally but bind only computationally, so a discrete-log break would permit inflation the audit could not see. That is the deliberate trade of every commitment-based design, and it is shared by every system whose signatures rest on the same curve.

What changes for Hathor users

- **Nothing, unless you opt in.** Shielded outputs are an opt-in header on ordinary transactions. Transparent transactions stay feeless and unchanged — the feeless model is untouched for everything Hathor does today.
- **Every custom token is supported**, in two tiers: amount hidden, or amount and token both.
- **Issuance is untouched.** Every mint and melt stays public; the total supply of HTR and of every custom token remains a public number, exactly as today.
- **Holders can selectively disclose**: reveal one output's amount and blinding factor, and any third party can verify it against the chain without trusting anyone.

Try it

- **Run it on testnet**
- **Read the design and the proof**
- **Audit it yourself**

The takeaway

Privacy systems fail in the places too complex for humans to fully review. Hathor's answer is to have less to review — three old primitives, one equation — and to make the one property a currency cannot afford to lose verifiable by anyone. Hathor hides individual amounts. It never hides the money supply — not as a promise anyone must trust, but as a theorem about the design that anyone can verify, and an implementation anyone can re-check against it, output by output.

Appendix — The supply-audit theorem

The proof below is stated in a simplified setting — token types visible; no mint, melt, fee, or unshield on the transactions — to keep the algebra on one page. Notes along the way show how each generalization folds into the same argument.

Setup

A shielded output hides its amount v in a **Pedersen commitment**

$$C = v \cdot H_T + r \cdot G$$

where G is the standard secp256k1 generator, H_T is a per-token generator derived by hashing the public token UID, and r is a random blinding factor. The derivation of H_T is "nothing-up-my-sleeve": computing any discrete-log relation among $\{G, H_{T_1}, H_{T_2}, \dots\}$ is as hard as the discrete logarithm problem itself, assuming the hash-to-curve derivation behaves as a random oracle. Transparent inputs and outputs are just the degenerate case $r = 0$ with v and T public.

Two attached proofs close the two ways to cheat; both are Fiat–Shamir proofs, sound under the same discrete-log assumption in the random-oracle model: a **range proof** shows each hidden amount lies in $[1, 2^{40})$, so no "negative" value can wrap around the group order; and, when the token type is hidden too, a **surjection proof** shows the hidden generator is the tag of one of the transaction's *input* tokens, not an invented one. (Token-hidden outputs still fit the form $C = v \cdot H_T + r \cdot G$, with an adjusted blinding factor.)

Consensus verifies, for every transaction, the point equation

$$\sum C_{\text{in}} \;=\; \sum C_{\text{out}} \qquad (\star)$$

exactly — there is no free excess term anywhere in the equation for hidden value to live in.[2]

Note: $(\star)$ is simplified for clarity. Public mints, melts, and fees enter it as extra public terms $a \cdot H_T$ — degenerate commitments — on the matching side. A full unshield (a transaction that moves all its shielded value back to transparent form) publicly declares one aggregate scalar e, entering as $e \cdot G$: think of it as one extra output committing to $v = 0$ with a published blinding factor. It carries no range proof — being a public scalar, $e \cdot G$ is value-free by construction and is checked directly against the equation.

Lemma (per-transaction conservation)

Lemma. *If $(\star)$ holds and all attached range and surjection proofs verify, then — unless the transaction's author can compute a discrete-log relation among the generators $\{G, H_{T_1}, H_{T_2}, \dots\}$ or break the soundness of an attached proof — both of the following hold:*

1. *(value) for each token T: $\sum v_{\text{in},T} = \sum v_{\text{out},T}$, as integers;*
2. *(blinding) $\sum r_{\text{in}} = \sum r_{\text{out}}$.*

Proof sketch. Move everything in $(\star)$ to one side and substitute each commitment's representation $v \cdot H_T + r \cdot G$. The result is a single point identity $\sum_T \delta_T \cdot H_T + \rho \cdot G = O$ whose scalar coefficients are all known to the author. If any coefficient were nonzero, the equation itself would *be* a discrete-log relation among the generators — solving an instance of the discrete logarithm problem. So a computationally bounded author can only satisfy $(\star)$ with every $\delta_T \equiv 0$ and $\rho \equiv 0$ *modulo the group order* q, which is exactly claim (2) and claim (1) mod q.

[2] Designs that allow a transaction a free-form excess point — Mimblewimble's kernel, Sapling's binding signature — must attach a proof that the excess carries no hidden H_T component. Hathor's rule forces the excess to be exactly zero, so the balance equation itself carries the full inflation argument.

The range proofs lift claim (1) from a congruence to an integer equality: each hidden amount is proven to lie in $[1, 2^{40})$ and each transparent amount is bounded by the same cap, so for a transaction with n inputs and outputs, $|\delta_T| < n \cdot 2^{40}$ — and n is bounded by the maximum transaction size, so $|\delta_T| \ll q \approx 2^{256}$. A multiple of q smaller than q in absolute value is zero. The range proofs are load-bearing here, not decorative: without them, conservation would hold only modulo q, and a "negative" value could mint $\approx 2^{256}$ units. □

This lemma is the entire consensus rule. Note what it is *not*: it is not a circuit that might be under-constrained. It is one curve equation whose soundness *is* the discrete log assumption.

Theorem (chain-wide blinding cancellation)

Theorem. *At every blockchain state, the blinding factors of all live UTXOs — transparent and shielded alike — sum to exactly zero:*

$$\sum_{u \,\in\, UTXO\ set} r_u \;=\; 0$$

(Transparent outputs contribute $r = 0$, so the statement needs no case distinction.)

Note: on a chain where full unshields have occurred, each publicly declared scalar counts as one permanent virtual entry, and the invariant reads $\sum_u r_u + \sum_{unshields} e = 0$ — the published scalar plays the role of one extra output's blinding in the inductive step, and nothing else changes.

Proof, by induction on the number of processed transactions. Let R_n be the sum of the blinding factors of the UTXO set after n transactions.

Base case. At genesis the UTXO set is empty, so $R_0 = 0$.

Inductive step. Assume $R_n = 0$. Transaction $n{+}1$ removes its inputs (blinding sum ρ_{in}) from the UTXO set and adds its outputs (blinding sum ρ_{out}). By the Lemma, part (2), $\rho_{\text{out}} = \rho_{\text{in}}$, so

$$R_{n+1} = R_n - \rho_{\text{in}} + \rho_{\text{out}} = R_n = 0.$$ □

The intuition: each blinding factor enters the sum exactly once when its UTXO is created and leaves exactly once when it is spent; no transaction can change the running total.

Remark (order-independence). Hathor's ledger is a DAG, not a single chain. The induction linearizes it without loss of generality: the invariant is a sum, sums are commutative, and any topological order of the confirmed transactions yields the same total. The audit applies at any confirmed ledger state.

Corollary (the public supply audit)

Corollary. *Summing the commitments of every live UTXO — transparent outputs counting as $v \cdot H_T$ — and applying the Theorem:*

$$\sum_u C_u \;=\; \sum_T S_T \cdot H_T + \Big(\sum_u r_u\Big) \cdot G \;=\; \sum_T S_T \cdot H_T$$

*where S_T is the total live value of token T — which equals the **publicly known total supply** of T: formally, by the same induction as the Theorem, run on values instead of blindings — by part (1) of the Lemma, each transaction changes the live value of T by exactly its public net issuance (mints minus melts), so the live total telescopes to the public issuance history. The ledger's entire UTXO set sums to exactly the public total-supply point: the article's claim, as one equation.*

In practice the audit sums only the shielded subset: transparent amounts are public, so their terms simply move to the expected-point side of the comparison — as do the published unshield scalars, as a public $-\big(\sum e\big) \cdot G$ term — exactly the three-step procedure of the main text. Either way, the comparison must pass at every block, under the stated hypotheses. □

The subtlety: you audit all tokens together — and that is enough

When token types are also hidden (the second privacy tier), no observer can partition $\sum C_u$ by token. **It is impossible to audit the balance of a single token in isolation** — the joint equation over all tokens at once is the only check available.

Does that weaken the guarantee? No — and the reason is the same linear independence that powered the Lemma. Suppose the joint equation passes but some token T_1 is secretly inflated by Δ, compensated by a deficit in token T_2. Then the adversary has produced commitments satisfying $\Delta \cdot H_{T_1} - \Delta' \cdot H_{T_2} + \rho \cdot G = O$ with Δ, Δ' genuine nonzero integers — genuine, because the range proofs rule out modular wrap-around — i.e., a discrete-log relation among independently derived generators: a break of secp256k1 itself. A computationally bounded adversary cannot make the joint audit pass while *any individual token* is out of balance. The single equation certifies every token simultaneously, even though no token can be checked alone.

Note where the strength comes from: every transaction was already admitted under the per-transaction Lemma, which enforces per-token balance at admission via the same linear independence. The chain-wide equation is the telescoped consequence of those per-transaction checks, not a substitute for them.