

bitSpire ↔ LNbits Security Pathway

aiolabs / satmachineadmin v2-bitspire

2026-05-15

bitSpire ↔ LNbits Security Pathway — State of the Union & Design Proposal

- 0 · Why this document exists
- 1 · Glossary (junior-dev friendly)
- 2 · Today's pathway — what the bytes actually do
- 3 · Threat model
- 4 · Audit findings — current state inventory
- 5 · Design proposal — layered defence using what Nostr already offers
- 6 · Phased roadmap
- 7 · Operator & customer trust narrative
- 8 · Audit-friendliness checklist (open-source readiness)
- 9 · Critical files (current code) and reference points
- 10 · Verification
- 11 · After this plan exits

bitSpire ↔ LNbits Security Pathway — State of the Union & Design Proposal

Audience: an operator, a junior dev, an auditor, the customer who walks up to the ATM. **Goal:** explain — without hand-waving — how money moves between a bitSpire ATM and the operator's LNbits wallet, what guarantees today's code provides, where the gaps are, and a concrete multi-layered fix that capitalises on Nostr instead of bolting on TLS-style fingerprints.

0 · Why this document exists

Today the satoshi-machine code lives at `~/dev/shared/extensions/satmachineadmin` on branch `v2-bitspire`. v2 swapped the legacy Lamassu SSH/PostgreSQL polling model for a Nostr-native one: bitSpire publishes invoices over kind-21000 NIP-44 v2 events, LNbits pays them, and our extension hooks the resulting `Payment` object.

The hard truth: the *settlement* itself uses Lightning (so it can't be forged once a preimage lands), but everything *around* the settlement — who the ATM is, what operator it belongs to, what the principal/commission split was, and what fiat was dispensed — currently rides on **mutable, unauthenticated metadata** (`Payment.extra`) plus a **stopgap that has the ATM hold the operator's own Nostr private key**. The latter means physical possession of the ATM = total compromise of the operator's LNbits account.

Two real-world incidents during dev surfaced this:

1. A stale `sintra` machine with placeholder `npub npub1111...` was created under a `test` user. A real cash-in landed on it because routing is *purely by wallet_id*, not by signed identity. We deleted the stale row, but the lesson is structural: there is no end-to-end identity proof.
2. The provisioning script (`/home/padreug/dev/shocknet/lamassu-next/deploy/nixos/provision-atm.sh`) writes `VITE_ATM_PRIVATE_KEY` straight into `/var/lib/bitspire/.env`. Today we set that to the operator's own privkey ("Option 1 stopgap"). Anyone with physical/root access to the ATM can sign as the operator on any relay.

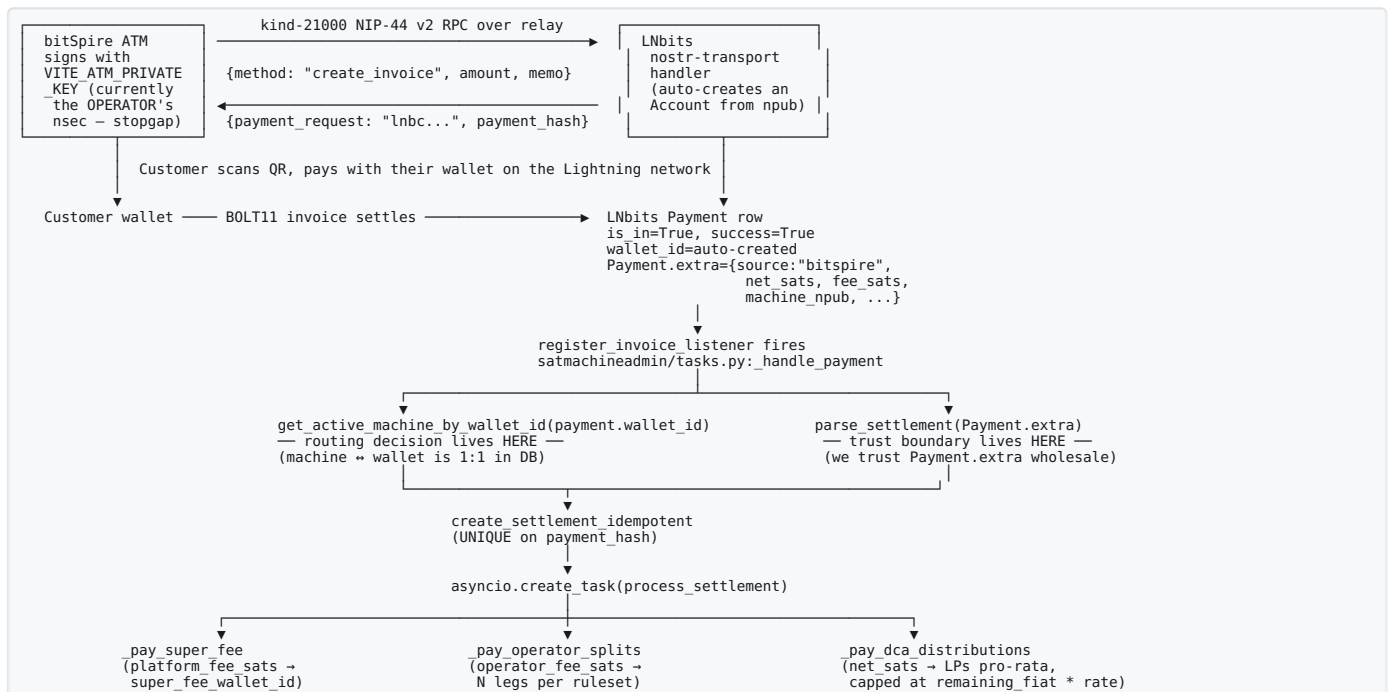
Lamassu's old answer here was TLS cert pinning. We have a richer toolbox — Nostr — and have so far used roughly one knob (NIP-44 encryption) of it.

1 · Glossary (junior-dev friendly)

Term	Plain English
bitSpire ATM	The cash machine. Cousin of the old Lamassu hardware. Identifies itself with a Nostr keypair (npub/nsec).
LNbits	The Lightning wallet server we self-host. The ATM is a “client” of LNbits over Nostr.
Operator	The human/business that owns one or more ATMs. Has an LNbits user account.
Super	The LNbits instance admin. Takes a platform fee from each operator.
LP (Liquidity Provider)	A customer who deposits fiat into the ATM business; receives BTC pro-rata via DCA.
npub / nsec	Nostr public / private key, bech32-encoded.npub is shareable; nsec is the secret.
Relay	A Nostr pub/sub server. Carries encrypted RPC events between ATM and LNbits.
NIP-XX	Nostr Implementation Possibility — a numbered protocol extension spec at ~/dev/nostr-protocol/nips/ .
kind-21000	The event kind bitSpire/LNbits use for encrypted RPC (set by lamassu-next’s nostr-transport).
NIP-44 v2	Authenticated encryption for Nostr DMs/RPC (ChaCha20 + HMAC-SHA256, MAC verified before signature).
Payment.extra	A free-form JSON dict LNbits stores alongside a payment. Mutable. Unsigned.
Preimage	The 32-byte secret revealed when a Lightning invoice is paid. Unforgeable proof of payment.
Settlement	One bitSpire cash-in or cash-out, landed as <code>adca_settlements</code> row in our DB.

2 · Today’s pathway — what the bytes actually do

2.1 Cash-out, end to end (the only flow currently wired)



2.2 What signs what today

Hop	Signed?	By whom?	Verified?
ATM → relay (kind-21000 event)	Yes (NIP-01 Schnorr sig)	ATM's keypair (= operator's keypair today)	Yes — relays drop unsigned events
RPC payload	Yes (NIP-44 v2 MAC + outer sig)	Same key	Yes — handler verifies MAC before decrypt
LNbits payment ↔ ATM identity	No	—	No — the link is the auto-created Account's wallet_id, set at first contact
Payment.extra contents	No	—	No — anyone with the wallet admin key can mutate
Settlement row in our DB	No (DB row, not an event)	—	n/a — operator trusts their own DB
Lightning settlement	Yes (cryptographically, via preimage)	The HTLC chain	Yes — preimage hashes to payment_hash

The Lightning settlement (the actual money) **is** cryptographically sound. Everything *attributing* that settlement to a particular machine, operator, fiat amount, and commission rate is not.

2.3 Routing decision today (the load-bearing line)

```
# tasks.py:59
machine = await get_active_machine_by_wallet_id(payment.wallet_id)
```

That's it. One DB lookup. The `wallet_id` was minted by LNbits' nostr-transport when it auto-created an Account from the ATM's npub *on first contact*. From that moment on, "which machine?" is purely a join `ondca_machines.wallet_id → wallets.id`. If you can land a payment on that wallet — by any means — it counts as that machine's settlement.

2.4 The Option 1 stopgap (what's in `provision-atm.sh` today)

```
VITE_ATM_PRIVATE_KEY=$(openssl rand -hex 32)
# or, in practice: VITE_ATM_PRIVATE_KEY=<operator's own nsec>
```

The operator's Nostr private key — the one tied to their LNbits Account — *is physically present on the ATM filesystem* (`/var/lib/bitspire/.env`). Threat: cleaner steals the ATM, dumps the disk, signs `kind:1 / kind:4 / kind:21000` events impersonating the operator on every relay, draining their wallets via crafted RPC. There is no second factor, no scoping, no revocation.

3 · Threat model

Who might try to break this, and how:

#	Adversary	Capability	What they want	Today's defence
T1	Random Lightning user	Pay any LNbits invoice they have a bolt11 for	Free fiat / cash-out without authorising	Bolt11 is single-use; preimage settles only once
T2	Curious LP	Has wallet admin key for their own LP wallet	See other LPs' balances	Operator-scoped CRUD; <code>_machine_owned_by</code> checks
T3	Rogue operator	Owens their LNbits user; controls their own machines	Forge settlements to inflate volume / dodge super fee	None — operator can mutate <code>Payment.extra</code>
T4	Compromised relay operator	Sees encrypted kind-21000 events	Censor, replay, reorder	NIP-44 protects content; no replay window ; relay can drop but not forge
T5	Thief with physical access to ATM	Can dump <code>/var/lib/bitspire/.env</code> , root the box	Drain operator wallet, sign as operator on Nostr	None — operator's nsec is on disk
T6	Insider at the LNbits host	Has DB access to LNbits	Mutate <code>Payment.extra</code> retroactively	None — <code>extra</code> is plain JSON, no audit log
T7	Attacker who knows operator's npub	Public knowledge	Spam fake kind-21000 from a key they generated	Auto-account-from-npub means they get a <i>different</i> wallet — but nothing stops them creating noise
T8	Insider at the super (LNbits admin)	Owens the LNbits node	Skim more than <code>super_fee_pct</code>	Operators must trust their host (this is fundamental — pick a host you trust, or self-host)
T9	Customer at the ATM	Walks up, scans QR	Pay an invoice attributed to a <i>different</i> operator's machine	<code>wallet_id</code> routing prevents cross-operator landing only if the invoice was generated for that wallet — confirmed by the stale-sintra incident: routing is wallet-level, not signed

T3, T5, T6 are the ones that keep the hardware honest. T3 + T6 are the reason `platform_fee_sats` and `operator_fee_sats` are stored as **absolute BIGINTs** (not derived from a mutable pct) — that defends the audit trail, but doesn't defend the initial write.

4 · Audit findings — current state inventory

Pulled from the two recent code-level audits of `~/dev/shared/extensions/satmachineadmin` (operator-scoping inventory) and `~/dev/lnbits/nostr-transport` (transport primitives).

4.1 What's already strong

- **Operator scoping is consistent.** All 33 routes filter by `current_user.id`; `_machine_owned_by` and `_client_owned_by` return 404 (not 403) on cross-operator probes so attackers can't enumerate other operators' resources.
- **Settlement idempotency.** `dca_settlements.payment_hash` is `UNIQUE`. A replayed Nostr event / dispatcher double-fire cannot cause a double payout.
- **Optimistic-lock claim pattern.** `claim_settlement_for_processing` prevents two concurrent `process_settlement` calls from racing the same row.
- **Settlement legs are typed and tagged.** `dca_payments.leg_type` $\in \{dca, super_fee, commission_split, settlement\}$; `Payment.tag = "satmachine:{npub}"` flows through LNbits' native payment filter UI.
- **Absolute-sats fee storage.** `platform_fee_sats` and `operator_fee_sats` are `BIGINT` columns, not derived from a mutable pct. This is the "Stripe Connect `application_fee_amount`" pattern and makes audits possible even if the commission rate later changes.
- **Append-only notes on settlements.** Partial-dispense recomputes prepend a timestamped memo; operator notes are timestamped + author-tagged. Tamper-evident at the row level.
- **NIP-44 v2 is correctly used in nostr-transport.** MAC verified before decrypt, outer Schnorr sig verified before MAC. (See `~/dev/lnbits/nostr-transport/lnbits/core/services/nostr_transport/*.`)

4.2 What's weak — confirmed gaps

ID	Gap	Where	Why it matters
G1	Routing is by wallet_id only. The ATM's signed identity is never re-verified at settlement land time.	tasks.py:59 get_active_machine_by_wallet_id(payment.wallet_id)	Once a wallet exists, anything paying it counts. No defence against T3, T7.
G2	Payment.extra is unauthenticated. We read source, net_sats, fee_sats, machine_npub, exchange_rate directly. Anyone with the wallet's admin key can mutate it.	bitspire.py:103-135	T3 / T6: forge favourable splits, dodge super fee, dispute history.
G3	ATM private key sits on disk as the operator's nsec.	provision-atm.sh:99 writes VITE_ATM_PRIVATE_KEY	T5: physical compromise = total operator compromise on every relay.
G4	No replay window on RPC events.	nostr-transport handler accepts events up to 10min old	T4: a relay can stash and replay a "create invoice" RPC. NIP-44 doesn't prevent replay; only NIP-40 expiration tags + nonce tracking do.
G5	sender_pubkey is not persisted onto Payment.extra by the dispatcher.	LNbits nostr_transport/auth.py:148-183	We can't tell, after the fact, which Nostr identity actually triggered a payment.
G6	Account.prvkey is nullable but in practice populated server-side.	LNbits Account schema	An auto-created account holds a key it generated. Anyone with DB access can read it. (T6.)
G7	No signed-request primitive. Nothing in nostr-transport requires a separate, scoped attestation on a payment — just the outer event sig.	nostr-transport	We can't bind "this is a real bitSpire settlement for machine X" cryptographically.
G8	No rate limiting at the relay layer.	—	T7 can spam our auto-account-from-npub endpoint.
G9	No ACL on which npubs may auto-create accounts.	nostr-transport	First contact wins. Combined with G3 + a real-world incident, this lets a stale/test machine accept real funds.
G10	Cash-in path is not wired. _handle_payment filters is_in=True only; cash-in is outbound (LNbits pays an LNURL-withdraw the customer scanned at the ATM).	tasks.py:57	Today we'd never know a cash-in happened. (Out of scope for this doc but flagged.)

4.3 What's *not* protected by encryption (clarification)

NIP-44 v2 makes the *transport* confidential and integrity-checked. It does **not**:

- Prove the sender is authorised to act for any party other than themselves (G1, G3).
- Prevent replay of an old, legitimately-signed event (G4).
- Bind a Lightning settlement to a particular kind-21000 RPC after the fact (G7).
- Audit who mutated `Payment.extra` after settlement landed (G2, G6).

Treat NIP-44 as TLS, not as authn/authz. We need additional NIPs for the rest.

5 · Design proposal — layered defence using what Nostr already offers

The trust model we want, in one sentence:

A settlement is genuine if (a) the operator delegated the ATM to act on their behalf, with a scoped, time-bound, revocable token, and (b) the ATM published a signed attestation referencing the Lightning preimage, and (c) the relay/Payment.extra metadata is treated as a hint, never as truth.

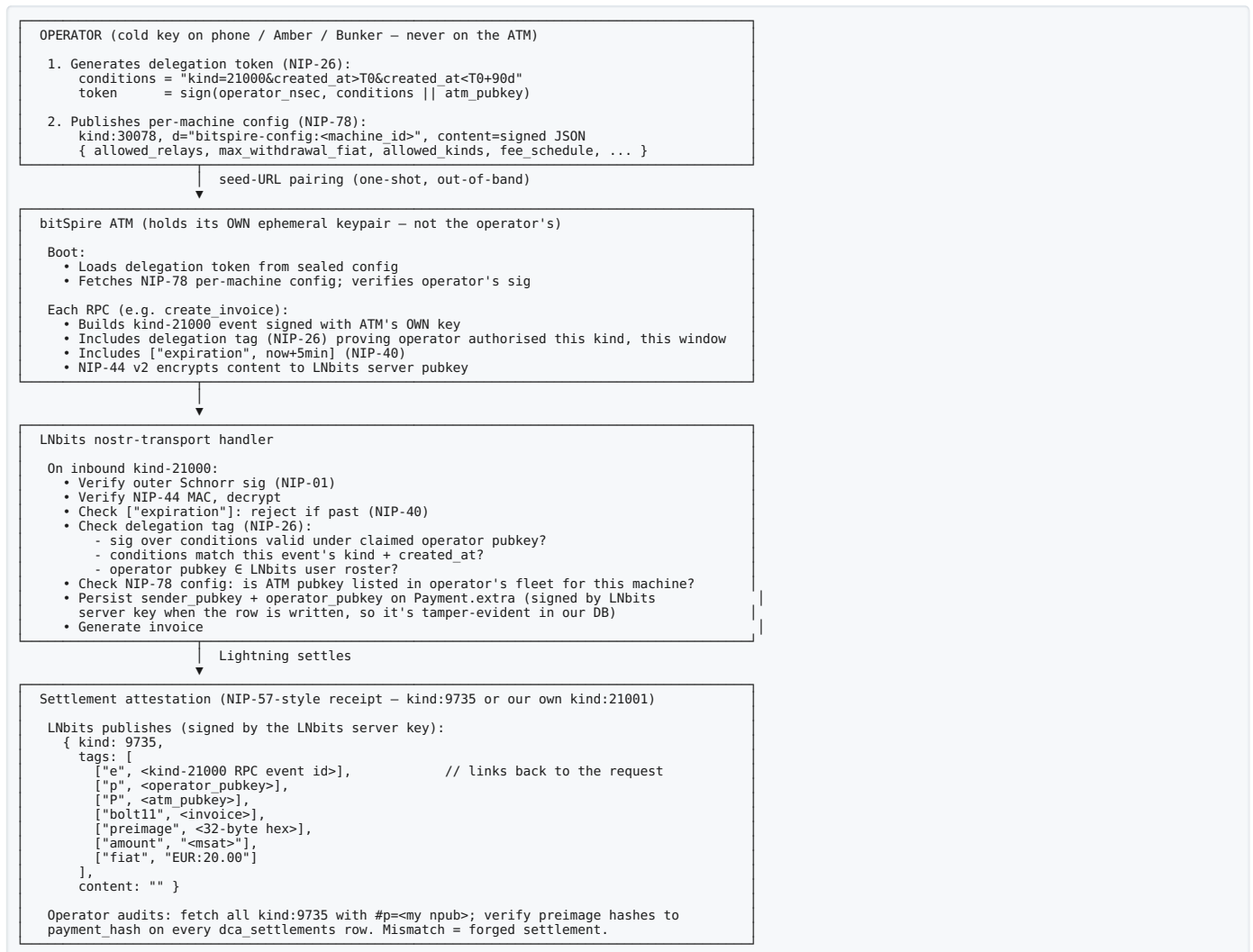
That's four primitives, each already specified in Nostr:

Layer	NIP	What it gives us
Identity & delegation	NIP-26 (~dev/nostr-protocol/nips/26.md)	Operator never gives their nsec to the ATM. Issues a kind-bound, time-bound delegation tag instead.
Settlement attestation	NIP-57-style receipt (nips/57.md)	ATM publishes a signed receipt event linking machine npub + Lightning preimage + amount/fiat. Receipt is the ground truth, not Payment.extra.
Replay protection	NIP-40 (nips/40.md)	Every RPC carries ["expiration", now+5m]. Relays drop expired events; handler refuses them.
Per-machine config	NIP-78 (nips/78.md)	kind:30078 with d="bitspire-config: <machine_id>" is the operator-signed source of truth for per-machine policy (max withdrawal, allowed relays, fee schedule). ATM fetches on boot; LNbits cross-checks.
Future: bunker	NIP-46 (nips/46.md)	Operator's nsec stays on a phone (Amber) or HSM. ATM gets an ephemeral session key + remote signer. End-state goal.

What we **do not** adopt and why (from the NIP survey):

- **NIP-42 relay auth.** Authenticates the connection to the relay; doesn't authorise the RPC payload. Useful for relay hygiene, but a red herring for our trust boundary.
- **NIP-59 gift wrap.** Hides metadata from relays but breaks the very auditability we want from NIP-57-style receipts. Only useful if anonymity matters more than audit.
- **NIP-32 labels.** Soft moderation signal, not enforcement. Fine as observability; useless as an access gate.

5.1 The new pathway (end-state)



5.2 Why each layer matters (junior-dev framing)

- **Delegation (NIP-26) closes G3.** The ATM doesn't *have* the operator's secret. It has a permission slip. Steal the ATM and you steal a permission slip that (a) only works for kind-21000, (b) expires in 90 days, (c) you can't use to sign on `kind:1` or DM the operator's contacts, and (d) the operator can shorten by issuing a new one with an earlier cutoff. This is the same shape as an SSH certificate vs. an SSH key.
- **Receipts (NIP-57 pattern) close G2 + G7.** The ground truth becomes a signed event referencing the preimage. `Payment.extra` remains as a hint (fast UI rendering), but disputes resolve against the receipt. If LNbits' DB is tampered with, the receipt on the relay is still there.
- **Expiration (NIP-40) closes G4.** A 5-minute window means a captured RPC can't be replayed at 3 a.m. when no human is watching the ATM.
- **NIP-78 closes G1 + G9.** The operator's signed config says "machine_id 42 has fleet member npub_abc and may withdraw up to EUR 500." The handler cross-checks. Stale `npub1111...` rows can't accept real settlements because they're not in any operator's fleet.
- **NIP-46 bunker (future) closes G5 + G6 properly.** The operator's nsec never touches LNbits' disk or the ATM's disk. It lives on the operator's phone or HSM and signs over an authenticated channel.

5.3 What we keep from today

- Absolute-sats fee storage (already audit-grade).
- Operator scoping + 404-not-403 ownership pattern.
- Settlement idempotency on `payment_hash`.
- Optimistic-lock claim for distribution.
- `dca_payments.leg_type` discriminator + LNbits `Payment.tag` for native filter UI.

None of those need to change. The new layers slot in *above* them.

6 • Phased roadmap

Phase	Scope	Closes	Effort	Blocker
S0 — Seed-URL pairing + ATM keypair separation	Provisioning script generates a fresh nsec for the ATM (already does — we just stop overwriting it with the operator's). Operator pastes a one-shot QR/seed URL containing {atm_npub, operator_npub, relay_list, signed_delegation_token} at ATM first boot.	G3 (most of it), G9	1 week	None — purely on our side. Use existing NIP-26 spec.
S1 — NIP-40 expiration on all kind-21000	Every RPC carries ["expiration", now+5min]. Handler refuses past-expiration. ATM clock check on boot (warn if drift > 60s).	G4	1-2 days	Relay must support NIP-40 (most do).
S2 — NIP-26 delegation enforcement in nostr-transport	Handler parses delegation tag, validates sig over conditions, checks conditions match the event, looks up operator pubkey in roster. Reject events without a valid delegation.	G3 (rest), G7 (partially)	1-2 weeks	LNbits PR upstream (or vendored fork on aiolabs/lnbits branch nostr-transport-nip26).
S3 — NIP-57-style settlement receipts	After LNbits internal payment legs complete, publish a signed receipt event per settlement (and per leg if we want leg-level audit). ATM subscribes; operator dashboard renders receipts side-by-side with dca settlements.	G2, G7	1-2 weeks	Decide kind: 9735 (semantic abuse for non-zap) vs. our own kind in 21001/21002 range.
S4 — NIP-78 per-machine config + fleet roster	Operator publishes kind:30078 config + kind:30000 fleet list. Handler cross-checks ATM npub ∈ fleet; reads max-withdraw/fee policy from config.	G1, G9	1 week	Define config schema; backwards-compat path for pre-NIP-78 machines.
S5 — sender_pubkey persistence + signed metadata in Payment.extra	When the dispatcher writes a Payment row, it stamps Payment.extra.sender_pubkey, delegation_root, and an HMAC over the key fields keyed by the LNbits server's own secret. Mutation post-write breaks the HMAC.	G2 (DB-side), G5, G6	3-5 days	LNbits PR — fairly localised.
S6 — Rate limiting + roster-gated auto-account	Auto-account-from-npub only fires if the npub appears in some operator's NIP-78 fleet OR if an explicit "open enrollment" flag is set. Relay/handler-level rate limit per pubkey.	G8, G9	1 week	LNbits PR.
S7 — NIP-46 bunker option	Operator can pair satmachineadmin with a Bunker (Amber, Nunchuk Custody, etc.). Operator's nsec leaves LNbits' DB; LNbits stores only the bunker connection.	G6, partial G5	4-6 weeks	Largest. Defer until S0-S5 land.
S8 — Cash-in path	Wire is_out=True cash-in handling: LNURL-withdraw with expiration matching the kind-21000 invoice TTL, attestation receipt on settle, refund queue for stale links.	G10	2 weeks	Out of scope for this security doc but tracked here for completeness.

Recommended sequencing for the *next sprint*: **S0 + S1 + S5**. They give us the biggest security delta with no upstream LNbits dependency for S0/S1 and a small, well-scoped LNbits patch for S5. S2/S3/S4 are the proper Nostr-native layer and should land in the sprint after.

7 · Operator & customer trust narrative

What we can say honestly to an operator after S0-S5:

“Your private key never goes on the ATM. The ATM has its own identity. You issue a permission slip — scoped to one kind of message, valid for 90 days, revocable from your phone. Every settlement publishes a public, signed receipt that anyone can verify against the Lightning preimage. If our database is ever tampered with, the receipts on the public relay are still there and still match. The platform fee and your fee are stored as absolute satoshi amounts — even if the rate changes tomorrow, last quarter’s audit is exact.”

And to a customer at the ATM:

“This ATM identifies itself by a public key printed on the side of the unit. The receipt event the network publishes after your transaction will reference that same key and the Lightning payment preimage — two pieces of cryptographic evidence that no one can forge after the fact.”

Compare to the Lamassu era: “the ATM has a TLS cert; if its fingerprint matches what the operator pinned, the connection is trustworthy.” Same instinct, narrower surface. Nostr lets us extend that to *every settlement* without re-inventing the wheel.

8 · Audit-friendliness checklist (open-source readiness)

Things a future auditor — or our open-source reviewers — will look for. Where we already pass, marked ✓; where we plan to pass after this work, marked →.

Check	Status	Where
All money-moving code paths have idempotency keys	✓	dca_settlements.payment_hash UNIQUE
All operator data scoped at the API boundary	✓	_machine_owned_by / _client_owned_by in views_api.py
No 403/404 enumeration oracle	✓	404 on cross-operator probes
Fee storage is absolute (not derived from mutable %)	✓	platform_fee_sats, operator_fee_sats BIGINT
Audit trail is append-only on settlements	✓	dca_settlements.notes prepended, never edited
Partial-dispense recompute preserves original ratio	✓	apply_partial_dispense_and_redistribute (H6 fix)
Concurrent settlement processing is race-free	✓	claim_settlement_for_processing
Every settlement has a signed, public attestation	→	S3 (NIP-57 receipts)
Operator’s private key is not present on the ATM	→	S0 + S2 (NIP-26 delegation)
RPC events cannot be replayed > 5 min later	→	S1 (NIP-40 expiration)
Payment.extra mutation is detectable	→	S5 (server-signed HMAC)
Stale machine rows cannot accept real funds	→	S4 (NIP-78 fleet roster cross-check)
Auto-account-from-npub is gated	→	S6 (roster + rate limit)
Key custody can be moved off LNbits’ DB	→	S7 (NIP-46 bunker)

The state we want the open-source release to be in for v2.0 final: all ✓.

9 · Critical files (current code) and reference points

For an auditor or new contributor doing a walk-through:

File	Role	Note
~/dev/shared/extensions/satmachineadmin/tasks.py	LNbits invoice listener. Entry point for all settlements today.	_handle_payment:56-95 — load-bearing routing.
~/dev/shared/extensions/satmachineadmin/bitspire.py	Parses Payment.extra. The trust boundary.	parse_settlement:68-92 — happy vs fallback path.
~/dev/shared/extensions/satmachineadmin/distribution.py	Three-leg distribution chain.	process_settlement — uses claim pattern.
~/dev/shared/extensions/satmachineadmin/crud.py	Operator-scoped DB layer.	claim_settlement_for_processing, _machine_owned_by.
~/dev/shared/extensions/satmachineadmin/views_api.py	33 routes, all check_user_exists except super-config PUT.	_assert_wallet_owned_by is the wallet-IDOR fix.
~/dev/shared/extensions/satmachineadmin/migrations.py	Schema.	dca_settlements is the audit row; dca_payments is the leg row.
~/dev/shocknet/lamassu-next/deploy/nixos/provision-atm.sh	Where keys land on the ATM today.	:81-99 — VITE_ATM_PRIVATE_KEY and the Option-1 stopgap.
~/dev/lnbits/nostr-transport/lnbits/core/services/nostr_transport/	LNbits transport handler (upstream we depend on).	NIP-44 v2 crypto here; G5/G6/G7 fixes will live here.
~/dev/nostr-protocol/nips/26.md	Delegation.	Source for S2.
~/dev/nostr-protocol/nips/40.md	Expiration.	Source for S1.
~/dev/nostr-protocol/nips/44.md	Authenticated encryption v2.	Already in use; spec reference for review.
~/dev/nostr-protocol/nips/46.md	Bunker / Nostr Connect.	Source for S7.
~/dev/nostr-protocol/nips/57.md	Lightning zaps & signed receipts.	Pattern source for S3.
~/dev/nostr-protocol/nips/78.md	App-specific replaceable events.	Source for S4.

Existing Forgejo issues this report supersedes/consolidates: [aiolabs/satmachineadmin#9](#) (v2 epic), [#11](#) (security audit findings), [#12](#) (ATM pairing + bunker deep-dive), [aiolabs/lamassu-next#44](#) (Payment.extra split). This document is the design that closes the security-relevant subset of those.

10 • Verification

How we'd test the proposed design end-to-end, once S0-S5 land:

- Negative test for G3:** Provision an ATM with seed-URL pairing. Confirm `/var/lib/bitspire/.env` contains only the ATM's own nsec and a delegation token. Attempt to sign a non-kind-21000 event with the ATM's key + delegation → handler rejects.
- Negative test for G4:** Record a kind-21000 RPC. Wait 6 minutes. Replay it on the relay → handler refuses (expired).
- Negative test for G1/G9:** Create a stale machine row with placeholder npub. Send a real payment to its wallet → handler rejects because the npub isn't in the operator's NIP-78 fleet list.
- Positive test for S3:** Run a full cash-out. Confirm a kind:9735-shaped receipt is published referencing the kind-21000 RPC event id + preimage. Verify the preimage hashes to the `payment_hash` on the `dca_settlements` row.
- Positive test for S5:** After settlement, mutate `Payment.extra` directly in the LNbits DB. Confirm the HMAC check fails on the next read; operator dashboard flags the row as "tampered."
- Revocation test for S2:** Operator issues a new delegation with `created_at` cutoff set to "now". ATM's next RPC (using old delegation) is rejected. ATM re-pairs with the new token; works again.
- Multi-operator isolation:** Two operators on the same LNbits instance, each with one ATM. Confirm Operator A's NIP-78 fleet doesn't list Operator B's ATM npub; LNbits cross-checks correctly.
- End-to-end smoke:** Real bitSpire on `~/dev/shocknet/lamassu-next/` (dev branch, `bun dev`) against the local LNbits stack (`~/dev/local/docker/regtest/docker-compose.dev.yml`, `LNBITS_SRC=~/dev/lnbits/nostr-transport`). One cash-out → settlement lands → receipt published → operator dashboard reconciles all three artefacts.

11 • After this plan exits

Once approved:

- The PDF for printing will be generated post-plan-mode (requires shell exec). Recommended path: render the markdown via `pandoc` to `~/dev/shared/extensions/satmachineadmin/docs/security-pathway-v1.pdf`; the markdown source will live at `~/dev/shared/extensions/satmachineadmin/docs/security-pathway-v1.md` so future contributors edit it in-repo.
- Open Forgejo epics on [aiolabs/satmachineadmin](#) linking back to existing [#9/#11/#12](#) and adding a new one for "Security

pathway hardening (S0-S7).”

3. Open a tracking issue on `aiolabs/lmbits` against the `nostr-transport` branch for the LNbits-side primitives (S2, S5, S6).
4. Sequence sprint: **S0 + S1 + S5 first** (highest ratio of security delta to upstream coupling). S2/S3/S4 in the following sprint.