Falcon Vault + Cold Signer — Implementation Report

Date: 2026-07-28
Repository: Falcon-faucet-wallet
Branch: main
Scope: Air-gapped vault custody on Falcon Ledger (portal vault UI + cold signer PWA)

1. Summary

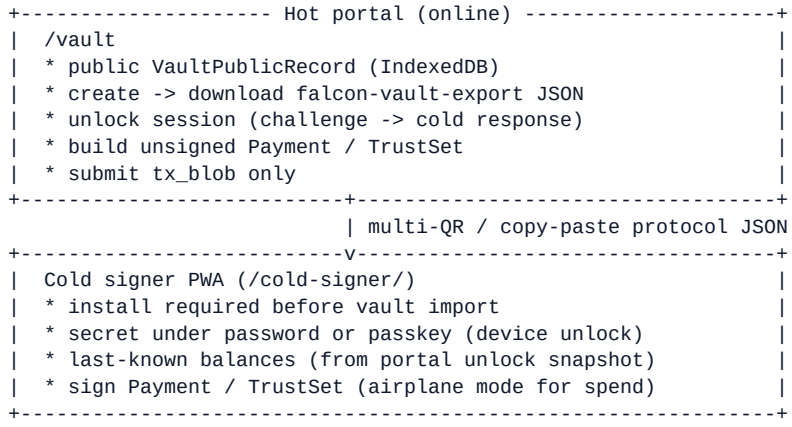
Falcon Ledger vault custody separates online portal access from offline key material. The portal **Vault** (`/vault`) stores only public metadata and builds/submits transactions. The **Cold Signer** PWA (`/cold-signer/`) holds the Falcon-512 secret, signs offline, and exchanges packages with the portal via multi-part QR codes or copy/paste of the same protocol JSON.

Component	Location
Portal vault UI	/vault (Wallet → Vault)
Cold signer PWA	/cold-signer/ (public/cold-signer/ production assets)
Cold signer source	cold-signer/ (Vite PWA)

2. Objectives

- Vault secret never remains on the hot browser after create (encrypted export file only)
- Cold device unlock via password (recommended) or passkey
- Multi-part QR transport for Falcon-sized payloads
- One-device workflow via **Copy full payload** / **Paste payload**
- Cold-signed **Payment** (FALCON and F-USDC), named destinations, and F-USDC **TrustSet**
- Last-known FALCON and F-USDC balances on cold after unlock with the portal

3. Architecture



3.1 Custody

Layer	Holds secrets?	Role
Portal /vault	No (after create)	Public address, build/submit, session unlock
Encrypted vault file	Yes (passphrase)	Offline backup / cold import (falcon-vault-export)
Cold device	Yes (re-encrypted under cold password/passkey)	Sign; secret never sent to servers

3.2 Protocol (v1)

Types: `src/lib/vault-protocol.ts`. Framing: `src/lib/multi-qr.ts`.

Direction	Content type	Payload
Portal → cold	vault-unlock-chal	Challenge + on-chain snapshot (FALCON + F-USDC)
Cold → portal	vault-unlock-resp	Falcon-signed challenge proof
Portal → cold	unsigned-tx	Payment or TrustSet tx_json + display fields
Cold → portal	signed-tx	tx_blob

Copy/paste uses the same JSON bodies as multi-QR (**Copy full payload** / **Paste payload**).

4. Features

4.1 Portal vault (/vault)

- Vault page and Wallet **Vault** entry
- Create vault: Falcon-512 keygen → passphrase-encrypted export download
- Public-only storage after create (`vault-store`)
- Import public metadata from an existing vault export file
- Locked state: Receive + Unlock vault
- Unlock: account fetch → multi-QR challenge (FALCON + F-USDC snapshot)
- Unlock response verification (Falcon-512 verify)
- Time-boxed session (~10 minutes)
- Send FALCON / F-USDC via cold sign
- Named destinations (`alice.bob` via `/api/wallet/name`)
- Add F-USDC trust line via cold-signed TrustSet
- Multi-QR display with **Copy full payload**
- Multi-QR scanner with **Paste payload**

4.2 Cold signer (/cold-signer/)

- PWA install before vault import (valid icons, service worker)
- Online permitted for install and import; airplane mode for signing spend
- Vault import by file upload; password recommended for device unlock
- Device unlock (password / passkey) → read-only last-known balances
- Unlock vault: camera with **Paste payload** on the same screen
- Sign transaction: camera with **Paste payload**; preview Payment or TrustSet
- Copy unlock-response and signed-tx payloads
- Cache last-known FALCON + F-USDC from unlock challenge

4.3 Supporting work

- `qrcode` dependency and `pnpm-lock.yaml` for production install
- Cold app excluded from Next.js TypeScript project
- Valid PNG app icons for installability
- Stable camera UI (scan screen not torn down by online/offline transitions)

5. User flows

5.1 Create vault and load cold device

1. Portal: `/vault` → Create vault → set export password → download JSON → confirm → public record saved
2. Cold: open `/cold-signer/` → **Install app** → open from home screen
3. Cold: import vault file → set cold unlock password
4. Keep the export file offline (e.g. SD card) for recovery; do not store the secret on the portal again

5.2 Unlock portal session

1. Portal: Unlock vault (live balances included in challenge)
2. Cold: device unlock → **Unlock vault** → scan or paste challenge
3. Cold: **Copy full payload** (or show QR) for the response
4. Portal: scan or paste response → session open

5.3 Send FALCON or F-USDC

1. Portal vault unlocked → Send → asset → destination (r… or name) → amount
2. Prepare unsigned package → cold **Sign** → review → approve
3. Portal scan or paste signed blob → submit

5.4 Add F-USDC trust line

1. Portal vault unlocked and funded; F-USDC shows no trust line
2. **Add F-USDC trust line (cold sign)**
3. Cold: review TrustSet (issuer, limit) → approve
4. Portal: submit signed blob → refresh balances

6. Testing

6.1 Single-device (copy/paste) — pass

Area	Result
Vault create + encrypted file download	Pass
Cold PWA install + file import + password unlock	Pass
Unlock challenge / response payloads	Pass
Portal session after cold unlock	Pass
Send FALCON (cold sign + submit)	Pass
Send F-USDC (with trust line)	Pass
Named destination resolution	Pass
Receive (address QR / copy)	Pass

Area	Result
TrustSet for F-USDC	Pass
Last-known FALCON + F-USDC on cold	Pass
Multi-part payload via paste	Pass

Password is the recommended cold unlock method. Passkey may fail on some Android Credential Manager configurations.

6.2 Two-device live barcodes and offline cold — pass

Field test: second physical device as cold signer.

Step	Result
Open cold signer from deploy URL on second device	Pass
Install / load vault	Pass
Airplane mode on cold device	Pass
Device login	Pass
Unlock vault via live multi-QR with portal	Pass
Send transaction (barcode cold sign, portal submit)	Pass

Scenario	Result
Unlock multi-QR (portal ↔ cold cameras)	Pass
Payment send multi-QR + submit	Pass
Real-world two-device camera use	Pass

Cold device ran offline (airplane mode) after load from URL. Login and send completed end-to-end.

7. Security

Control	Behavior
Portal after create	Public vault record only; no secret
Cold import	Installed PWA; file upload
Device unlock	Password or passkey; last-known balances only
Spend / TrustSet	Cold signs; airplane mode for spend
Unlock snapshot	Display-only; not part of signed challenge bytes
Approval	User reviews destination, amount, or trust issuer on cold before sign

Operators remain responsible for verifying transaction details on cold, physical security of the cold device and password, and safekeeping of the offline export file.

8. Code map

Area	Path
Vault UI	src/app/vault/page.tsx

Area	Path
Multi-QR	src/lib/multi-qr.ts, src/components/MultiQrDisplay.tsx, MultiQrScanner.tsx
Protocol	src/lib/vault-protocol.ts
Export / store / session	src/lib/vault-export.ts, vault-store.ts, vault-session.ts
Unlock verify	src/lib/vault-unlock-verify.ts
Unsigned builders	src/lib/falcon-tx-sign.ts (buildPaymentTxJson, buildFusdcPaymentTxJson, buildTrustSetTxJson, signTxJson)
Cold app	cold-signer/src/App.tsx, cold-signer/src/components/MultiQr.tsx
Cold storage	cold-signer/src/lib/coldVaultDb.ts
Production PWA assets	public/cold-signer/

```
npm run dev:cold
npm run build:cold
npm run verify:multi-qr
```

9. Optional follow-ups

- Stress tests for multi-QR frame loss and slow scans near `LastLedgerSequence` expiry
- Broader passkey testing on additional Android devices
- Additional cold-signed transaction types as needed

10. Conclusion

Vault and cold signer support create, receive, unlock, Payment (FALCON and F-USDC), named destinations, and F-USDC TrustSet. Testing covers single-device copy/paste and two-device live multi-QR with the cold device in airplane mode (URL load → offline login → send).