

FOIL (\$FOIL): Build

Companion: `Foil_Overview.md` . This file is the **how**: keys, the Stash pool with funding rounds, tins, the Collector Crypt integration on Robinhood Chain, gift links, Rip Rooms, spend limits, shipping, services, data, tests and milestones. Every milestone is one Claude Code session with green tests.

Chain: Robinhood Chain 4663 only. USDG

`0x5fc5360D0400a0Fd4f2af552ADD042D716F1d168` (6 decimals; EIP-3009 via its facet router, domain "Global Dollar" v1).

Stack:

- Contracts: Foundry, Solidity 0.8.26.
- Circuits: circom 2.x + snarkjs (Groth16, BN254).
- Services: Bun · Hono · tRPC v11 · Drizzle · Postgres 16 · Redis 7 + BullMQ · viem · Dwellir RPC.
- Client: Expo mobile first, plus web, with on-device keys and proving.

Prepared 2026-10-08. (U) = unverified.

0. Recon → design consequence

Fact (dated)	Consequence
Collector Crypt (CC) packs on RHC since ~Sep 22 2026. Gacha/vault <code>0x4d56B148624e17FD53c08D1C4998e8e3945bDE28</code> ; card ERC-721 <code>0xb47a306bD89E8b6fA2b14f553E534AdcD7Cc337F</code> . CC redeploys, so read addresses from the API and check them against a pinned list	The tin is the player: it pays and receives
EVM API: <code>generatePack {playerAddress, packType, chainId:4663, token} → {payTo, amount, memo} ; pay USDG; openPack {memo, payTxHash} → evm_token_id . VRF check at /api/evm/vrf/verify</code>	Pull flow in \$5
Buyback: signed quote (10 min) → <code>GachaVault.sellBack</code> (~250k gas) pays USDG and burns. 72h, 85–93%; the quoted holder must hold the card	Tins sell back; USDG goes back to the Stash

Shipping: SIWE (EIP-191), address, /redeem/prepare → [approve, batchBurn, pay] . Smart-wallet signatures are checked on Base	Tins are plain EOAs
USDG supports EIP-3009 on RHC; the gas rebate ended Sep 29	Pay and return by signature; gas arrives inside the funding round
No bridge lane for RHC cards	Inventory = CC's RHC machines; cards stay on RHC

1. Architecture

```

contracts/ (Foundry, RHC)
├─ StashPool.sol      (immutable: entry queue, Poseidon tree, nullifiers, Groth16)
├─ Verifiers.sol      (generated: joinsplit, tin_request, ppoi)
├─ ScreeningGate.sol  (provider roots, standby, flag; timelocked provider set)
├─ FOIL.sol, FeeRouter.sol, FoilStaking.sol, Bonds.sol
circuits/ joinsplit.circom (2x2) · tin_request.circom · ppoi.circom
services/
├─ relay              (submits Stash txs and EIP-3009 authorizations; fee inside the
├─ round-operator     (closes rounds, executes them with USDG + ETH in one tx; front
├─ tin-browser        (client: in-app browser, injected provider, transaction guard,
├─ cc-gateway         (key mode only; TDX CVM; Foil's CC API key; only via OHTTP; ge
├─ ohttp-relay        (≥ 2 independent operators; ALL client traffic goes through it
├─ tunnel-relay       (TLS CONNECT relay for client → CC shipping; sees only ciphert
├─ rooms              (Rip Rooms posts and season leaderboard; stores only what user
├─ indexer            (CC card contract, metadata, insured values; tin-set snapshots
├─ ppoi-node          (screening; clean-inbound attestations for send-backs)
├─ status             (counters, queue, round sizes)
├─ api                (tRPC)
apps/ mobile + web

```

Upgradeability:

- **Immutable:** StashPool , verifiers.
- **2-of-3 multisig + 24h timelock:** provider set, clean-inbound allowlist, fee parameters within bounds.

- **Guardian:** can pause deposits and new tin requests only. It can never pause withdrawals, `exitPending`, `cancelRequest`, `selfExecute`, `send-backs` or key export.

2. Keys

```
seed (BIP-39)
├ sk_auth (Baby Jubjub EdDSA; signs spends; never leaves device)
├ nk      (nullifying key)          └ vk, ek (viewing / note encryption)
├ mpk     = Poseidon(pk_auth.x, pk_auth.y, Poseidon(nk))
├ tin_i   = BIP-32 secp256k1, m/44'/60'/4040'/0'/i'      (tins: plain EOAs; harden
└ handle_s = HKDF(seed, "foil/handle/" || season)        (Rip Rooms season key, ed
```

- **Proving:** on the device by default. There is no remote prover in v1.
- **Discovery:** the client downloads the public **tin-set snapshot** (every tin ever funded, with card token IDs and USDG balance) and hourly diffs, through OHTTP, and matches its own tins locally. No per-address queries.
- **Export:** any tin's key, as EVM hex.

3. Stash pool and funding rounds

3.1 Notes, queue and exits

- **Commitments:** $npk = \text{Poseidon}(mpk, \rho)$, $C = \text{Poseidon3}(\text{USDG}, \text{amount}, npk)$, **computed on-chain** at entry from the measured amount.
- **Nullifiers:** $N = \text{Poseidon3}(nk, \rho, \text{poolId})$.
- **Spend authorization:** EdDSA by `sk_auth` over the transaction hash, checked in-circuit.
- **Entry queue:** deposits and send-backs are **pending** until cleared by PPOI (≥ 2 ppoi-node signatures) or until standby ends (15 min; 60 min if fewer than 2 providers are fresh). **Only admitted entries enter the tree.** Flagged entries never do.
`exitPending(seq, originSig)` refunds pending or flagged entries to their origin at any time; it takes an EIP-712 signature by the origin and anyone may submit it, so a gasless tin can use it.
- **Conservation:** $\text{USDG.balanceOf}(\text{pool}) \geq \Sigma_{in} - \Sigma_{out}$, checked in every function; public counters.
- **Fees:**
 - deposit 0.15%;

- withdrawal to an outside wallet 0.25%;
- tin funding: 0.15% of the tin size + \$0.20 + gas at cost ($\approx$ \$1 at current prices);
- relayer fee floor \$0.25 per spend.

3.2 Funding rounds

```
interface IStashRounds {
    // 1. Request a tin (or a refill of an existing tin): the proof spends notes and
    //    tinCommit = H(tin, salt, size, roundId); refundCommit = note commitment f
    function requestTin(Proof calldata p, PublicInputs calldata pi /* roundId, size
    // 2. Operator executes a closed round: reveals each (tin, salt) pair, pays all
    function executeRound(uint64 roundId, address[] calldata tins, bytes32[] callda
    // 3. Safety valves (never gated):
    function selfExecute(uint64 roundId, uint256 reqIdx, address tin, bytes32 salt)
    function cancelRequest(uint64 roundId, uint256 reqIdx, bytes32 cancelSecret) ex
    function returnToStash(Auth3009 calldata a, bytes32 npk, bytes calldata ct, byt
}
```

- **Request states:** each request settles **exactly once**: Pending → Executed | SelfExecuted | Cancelled. `executeRound` skips cancelled requests; `cancelRequest` and `selfExecute` revert once a request is settled.
- **Why the operator can't steal:** a refund can only go to `refundCommit`, fixed inside the proof at request time, and cancelling needs `cancelSecret`, which never leaves the device. The operator learns `(tin, salt)` but neither of those.
- **Round timing:** a round closes on the 10-minute boundary if it has ≥ 8 requests; otherwise it rolls forward, up to 30 minutes, then closes anyway.
- **Execution:**
 - The operator submits the reveal list **sorted by tin address** (order carries no information).
 - Each `H(tin, salt)` must match a request in that round.
 - Each tin receives its tin size in USDG plus `ethPerTin` (≈ 0.0004 ETH, from measured gas for pay + sell back + return). The operator fronts the ETH in `msg.value` and is reimbursed in USDG from the per-tin fee.
- **Reveal channel:** the client sends `(tin, salt)` to the operator through OHTTP after requesting. The operator learns the tin address, never the user.
- **Liveness:** if a round isn't executed within 60 minutes of closing, the requester can `selfExecute` (submitted by any relayer; no ETH needed from the user, since the tin pays CC by signature) or `cancelRequest`.
- **Omission:** an operator that executes a round but leaves out a request whose reveal

it acknowledged is slashable. Every reveal gets a signed receipt from the operator, and that receipt is the evidence.

- **Ready tins:** the app keeps 2–6 tins requested in advance, at random times, so rips are instant.
- **Tin sizes:** \$10, \$25, \$50, \$100, \$250, \$500, \$1,000. A refill request can top up an existing tin, for example for a shipping fee, without any tin-to-tin transfer.

3.3 Send-backs (tin → Stash)

- **How:** `returnToStash` calls USDG's `receiveWithAuthorization` (executable only by the pool), with `nonce == keccak256(npk, keccak256(ct), keccak256(epk), poolId, chainid)` checked on-chain, so a relayer can't redirect the note and nobody can replay the signature straight to USDG. The tin needs no ETH. (U: confirm `receiveWithAuthorization` on RHC USDG.)
- **Fee:** free up to the tin's funded amount, 0.25% above.
- **Admission:** at once if ≥ 2 ppoi-nodes sign that every inbound USDG transfer to the tin came from the Stash or the CC GachaVault (sell-back payouts). Otherwise it waits through standby with every inbound counterparty screened. Min \$1; max 10 send-backs per tin.

3.4 Circuits

Circuit	Proves
<code>joinsplit.circom</code> (2×2)	Membership, nullifiers, EdDSA, conservation, <code>extDataHash</code> (relayer, fee, recipient, ciphertext hashes, <code>poolId</code>)
<code>tin_request.circom</code>	A join-split whose public output is <code>size ∈ set</code> reserved to <code>roundId</code> , plus public <code>tinCommit</code> , <code>refundCommit</code> (a well-formed note commitment owned by the requester) and <code>cancelHash</code>
<code>ppoi.circom</code>	Blinded non-membership of the entry origin against the provider roots

Ceremony: Phase-2 with ≥ 3 parties, at least one independent of the team.

4. Tins

- **What they are:** plain EOAs: one rip per tin, never reused, and no transfers between

a user's tins.

- **Gas:** comes only from the round. Leftover ETH stays in the tin ($\approx \$1$), or the user can send it back to the round operator's float with a tap.
 - **Card ownership:** the card stays in its tin (vaulted by CC) until sold back, shipped or gifted.
-

5. Collector Crypt integration

Two modes.

- **Browser mode (§5A)** needs no Collector Crypt key and ships in P1.
- **Key mode (§5.1–5.3)** is added when a key arrives.

Contacts, email drafts and questions are in `Foil_CC_Outreach.md`.

5.1 Access (out-of-band)

- **Foil's API key** on chain 4663, with `can_adopt`. No KYB is published (U). Without adoption, Foil runs on standard machines with no partner fee.
- **Confirm with CC (U):**
 - production EVM host and machines on 4663;
 - whether an EIP-3009 transfer from the player (relayer-submitted) is accepted as payment, or a contract purchase is required;
 - whether a later holder can sell back on EVM;
 - partner-fee payout on RHC vs Solana;
 - written OK before any "powered by" wording.

5.2 Rip, step by step

1. The client picks a ready tin T (tin size $\geq$ pack price).
2. **Generate:** `cc-gateway /pull/generate {machine, T}` returns `{payTo, amount, memo}`. The client checks that `payTo` is pinned, the amount is the machine price, and the chain is 4663.
3. **Pay:**
 - EIP-3009 `transferWithAuthorization(T → payTo)` signed by T , submitted by the relayer;
 - or, if CC requires a contract call, T sends it using round ETH.
4. **Open:** `/pull/open {memo, payTxHash} → evm_token_id`. The client verifies the VRF

and shows the card, insured value and sell-back offer.

5. Choice:

- **Keep:** the card stays in `T`.
- **Sell back (≤ 72h):** quote → `T.sellBack` → USDG in `T` → `returnToStash` (admitted at once).
- **Gift:** §6.
- **Ship:** §8.

6. Leftover USDG goes back to the Stash.

5.3 Adopted machines

- **Partner fee: min(4% of price, EV headroom).** It comes out of the pack's expected value; half is paid on completion and half only if the card is sold back.
- **Display:** EV, `evLow`, the fee, the buyback % and odds on every machine.
- **No sell-back nudges:** neutral UI, and the copy lint blocks prompts.

5A. Browser mode (works without a Collector Crypt key; ships first)

Foil runs **without** a Collector Crypt partner key.

- **Browser mode** is P1: Foil is a wallet with a built-in browser, and the user rips on Collector Crypt's public pack site (or other RHC pack front ends) through a fresh tin.
- **Key mode** (§5.1–5.3) is added when a key arrives: native rip screens and the partner fee.

Request the key anyway (see `Foil_CC_Outreach.md`), but no P1 milestone waits for it.

5A.1 The browser

- **Engine:** a mobile WebView with an **injected EIP-1193 provider**, announced via EIP-6963. No WalletConnect.
- **Domains:** `gacha.collectorcrypt.com`, `collectorcrypt.com` and `thecardwall.com` can use the wallet. Any other site opens without wallet access.
- **Routing:** all traffic, including DNS, goes through `tunnel-relay`, with TLS end to end and optional Tor. Sites see the relay, not the user.
- **Fresh storage for every tin:** cookies, local storage and cache are new for each tin session and wiped afterwards, so the site can't connect two tins through the

browser. The browser presents a common user-agent and no fingerprinting APIs.

5A.2 One tin per rip, automatically

1. "Rip" opens the site with a ready tin `T` connected. Only `T` is exposed, and only chain 4663.
2. As soon as a card lands in `T` (card contract `Transfer` event), the browser **switches to the next ready tin** and wipes storage. The next rip uses the new tin.
3. If no ready tin is left, the browser pauses with "Next tin arrives in the coming round (≤ 10 min)". It never reuses a tin.
4. **Spend limits** apply before the tin is connected. A rip that would break the daily or weekly cap never gets a tin.

5A.3 Transaction guard

Every request from the site is decoded, simulated and checked before `T` signs. Everything else is refused.

Request	Allowed only if
USDG <code>approve(spender, amount)</code>	<code>spender</code> is the CC gacha/vault or CC's shipping payee, and <code>amount</code> is exactly the price or fee shown, never unlimited
Call to the CC gacha/vault	Pack purchase or buyback selectors only; USDG out $\leq$ what <code>T</code> holds
Call to the CC card contract	<code>batchBurn</code> of <code>T</code> 's own card (shipping) only
Shipping <code>pay</code>	Amount = the fee shown
<code>personal_sign</code>	Sign-in messages for an allowed domain with address <code>T</code>
<code>eth_signTypedData_v4</code>	EIP-3009/2612 only to allowed spenders, for the exact amount
Native ETH <code>value</code>	0

- **Simulation:** each transaction runs through `eth_call` on our RPC and shows "\$50 USDG, +1 card". It is blocked if anything would leave `T` for an address outside the allowlist.
- **Contract changes:** new CC contracts are found by watching CC's deployer on RHC

and need a 2-of-3 signed allowlist update before use. Until then the action is blocked, with a message.

5A.4 What browser mode changes

- **No partner fee.** Foil's revenue is Stash fees, tin fees and gift fees.
 - **Collector Crypt's own screens** appear inside Foil. Rip Rooms posting still works: the tin signs the post, and `rooms` checks the card on-chain.
 - **If CC blocks embedded browsers or the relay:** fall back to the **Foil browser extension**. It runs in the user's normal browser, injects the provider, and forwards signing requests to the Foil app over an encrypted channel through our relay (no third-party pairing service), with the same guard and one-tin-per-rip rule.
-

6. Gift links

Gift a pack:

1. Request a tin `G` in a round with a tin size $\geq$ the pack price. The \$0.50 gift fee is paid in-pool.
2. The app creates the link `foil.gift/#<base64url(G's private key || machineId || note)>`. The key sits in the URL fragment, which is never sent to any server.
3. **Recipient:** the Foil app imports `G`'s key and immediately **moves the USDG** into the recipient's own Stash (`returnToStash` to the recipient's `npk` ; admitted at once as clean-inbound). Then they rip from their own new tin.

Gift a card:

1. The giver shares the key of the tin `T` that holds the card (a "gift this tin" link).
2. **Recipient:** imports `T`, then transfers the card from `T` to a **fresh tin of theirs** using `T`'s leftover round ETH. Any USDG in `T` goes to the recipient's Stash.

What's visible:

- On-chain: one transfer from `T` to the recipient's new tin. Nobody can tell `T` was a gift or who gave it.
- **The giver knows the recipient's new tin address.**

Bearer risk: until claimed, the giver (and anyone with the link) can still move the funds or card. The UI says so and shows the claim status.

7. Rip Rooms

- **Season handle:** a random name + ed25519 key derived per season (`handle_s`). There is no link between seasons.
 - **Posting a pull** (opt-in, per pull):
 - the client sends `{tokenId, tin T, sig_T(EIP-191: "foil-post|season|handle|tokenId"), sig_handle}` through OHTTP;
 - `rooms` checks on-chain that `T` holds (or minted) `tokenId`, then publishes the post: card image, grade, insured value, handle, time.
 - **Leaderboard:** per season, by the insured value of posted pulls. Only posted pulls count.
 - **Stored:** posts only (they're public by choice). No IPs, and no unposted tins.
 - **What a post links:** that tin to the handle. Several posts link those tins to each other through the handle; the UI warns on the second post of a season.
 - **Copy:** "Hits", "Room", "Season". Never "win", "jackpot" or "profit".
-

8. Quiet Ship

1. SIWE as the tin `T` → address → `/redeem/prepare` → `[approve, batchBurn, pay]`, all sent **directly to CC** through `tunnel-relay` (TLS end to end).
 2. The client checks the bundle: it burns only `T`'s card, `pay` equals the quote, and targets are on the allowlist.
 3. `T` sends the calls with round ETH, paying the shipping fee in USDG from what it holds. If `T` is short, the app requests a **refill** of `T` in the next round (no tin-to-tin transfer).
- **Batch ship** (optional): ship several tins' cards together. The UI warns that this links those tins to each other and to the address.
-

9. Spend limits and region rules

- **Defaults:**
 - \$250/day and \$1,000/week of pack spending;
 - raising a limit takes 24h, lowering it is instant;
 - a cool-off button pauses rips for 7 days (it can't be undone early).

- **Where it's enforced:** client-side, stored in the encrypted local store **and** in an encrypted backup blob (encrypted to `ek`, fetched by a seed-derived locator through OHTTP), so limits come back after a restore. `cc-gateway` also enforces a per-install rate cap (an opaque install token, no identity).
- **Honest limit:** a determined user can bypass self-limits with another wallet app.
- **Region rules:** CC's restricted destinations plus a configurable list where digital pack openings are restricted, enforced by self-declared region plus device locale.

10. Token and fees

- **\$FOIL:** an ERC-20 on RHC (fresh deployer, 2-of-3 Safe).
- **FeeRouter (80/10/5/5):** **80%** goes to relayers and round operators (the work); then **10%** treasury, **5%** stakers (first-loss behind the round operators' ETH float), **5%** burned.
- **Fee sources:** Stash fees, tin fees, gift fees, the shipping flat fee, and CC partner fees.
- **Bonds:**

Role	Bond (\$FOIL)
Relayer	20,000
Round operator	30,000

Slashing for a closed round not executed within 60 min without cause, an omitted request with a signed reveal receipt, or a mismatched reveal (attestor vote 3-of-5, 48h timelock).

11. Data (no linking tables)

```
pool_events (tx, kind, size, ts) -- public chain data
rounds      (round_id, closed_at, executed_at, n_tins, operator)
tins_pub    (addr, round_id, funded, sent_back, send_backs)
machines    (machine_id, price, ev, ev_low, buyback_bps, partner_fee_bps, adopted
rips_agg    (day, machine_id, count, gross_usd, sold_back_usd)
posts       (post_id, season, handle, token_id, tin, insured_value, ts) -- publ
cards, tokens, values -- CC me
```

status (ts, component, ok, detail)

Privacy CI:

- no table links two tins, a tin to a note, or a tin to an IP (posts excepted, by design);
- log-lint on gateways, operators and `rooms` ;
- the client network layer fails CI if any request bypasses OHTTP or the tunnel.

12. Tests and acceptance

Area	Pass condition
Conservation	Fuzz: <code>balance ≥ Σin - Σout</code> ; reserved tin sizes accounted
Entry queue	Pending or flagged entries have no leaf; <code>exitPending</code> refunds once, to origin
Amount binding	On-chain commitment computation; crafted commitments rejected
Key split	No proof without the <code>sk_auth</code> EdDSA signature
Rounds	Close at ≥ 8 requests or 30 min; <code>executeRound</code> rejects mismatched reveals, unsorted lists and duplicate commits; ETH and USDG in one tx; each request settles exactly once (execute vs selfExecute vs cancel race tests); cancel pays only <code>refundCommit</code> and needs <code>cancelSecret</code> ; <code>selfExecute</code> works with <code>msg.value = 0</code>
Send-backs	Fee-free within the funded amount; immediate admission only with ≥ 2 clean-inbound signatures (Stash or GachaVault only); nonce bound to the note; no direct replay to USDG
Rip (mainnet canary, cheapest machine)	The client rejects a modified <code>payTo</code> , amount or chain; the card arrives in <code>T</code> ; VRF verifies; sell back pays <code>T</code>
Gift links	The key never appears in any request (fragment only); claims move USDG or the card to the recipient's own Stash or tin; the UI shows claim status
Rip Rooms	A post without a valid tin signature is rejected; unposted tins never reach <code>rooms</code>

Browser-mode guard	A malicious-transaction corpus is 100% blocked; simulation diffs match execution
Browser-mode isolation	No storage carries over between tins; the site sees the relay IP only; only the active tin is exposed; the switch happens after every received card
Spend limits	Raises wait 24h; cool-off can't be undone early; caps hold across app restarts and seed restores
Shipping	The tunnel never terminates TLS; bundle checks reject burns of tokens not in <code>T</code>
Copy lint	Blocks "APY", "yield", "interest", "earn", "investment", "returns", "win", "jackpot", "profit", "guaranteed", "sell back now", "untraceable", "anonymous", "mixer", and "Pokémon" in brand slots

13. Red-team checklist

1. **Small rounds:** minimum 8, max wait 30 min, round sizes published.
 2. **Reveal-time linking:** `(tin, salt)` goes to the operator through OHTTP; the operator never sees an IP or a note.
 3. **Operator liveness:** `selfExecute` and `cancelRequest` after 60 min; bonds.
 4. **Gift interception:** bearer risk, fragment-only keys, claim status shown.
 5. **Rip Rooms linkage:** posts are opt-in and per pull, with a warning on the second post.
 6. **CC's view:** Foil's partner tag plus every ripping tin. Accepted and stated.
 7. **Transaction substitution:** pinned CC addresses; every CC-built call is decoded and checked.
 8. **Gambling exposure:** limits on by default, region rules, display rules; legal read per market (out-of-band).
 9. **USDG freeze:** monitoring and runbook.
 10. **Supplier dependence:** without a key, Foil runs on standard machines; sell back and shipping need only the tin's signature.
-

14. Deploy runbook

1. Ceremony (`joinsplit` , `tin_request` , `ppoi`); publish transcripts.
 2. Fresh deployer; Safe 2-of-3.
 3. Deploy `ScreeningGate` , `StashPool` , `FOIL` , `FeeRouter` , `FoilStaking` , `Bonds` .
 4. **Canary**: 30 days with \$1k max deposit and \$500k total.
 5. Services: relayer, round-operator (bonded, ETH float), ohttp-relay (≥ 2 operators), tunnel-relay, cc-gateway (CVM), indexer, rooms, ppoi-node, status.
 6. CC setup: Foil API key on 4663; adopt 2–4 machines.
 7. **Smoke test**:
 1. deposit \$30;
 2. request a \$25 tin;
 3. round executes;
 4. rip the cheapest machine;
 5. sell back;
 6. return to the Stash;
 7. gift a \$10 pack tin to a second test seed and claim;
 8. `exitPending` on a fresh deposit.
-

15. Milestones (one Claude Code session each)

#	Milestone	Lane	Phase
M0	Monorepo, CI, copy lint, privacy CI	Core	P1
M1	Circuits (<code>joinsplit</code> , <code>tin_request</code> , <code>ppoi</code>) + ceremony scripts	Core	P1
M2	<code>StashPool</code> core (queue, on-chain commitments, conservation, exits) + <code>ScreeningGate</code>	Core	P1
M3	Funding rounds (<code>requestTin</code> , <code>executeRound</code> , <code>selfExecute</code> , <code>cancelRequest</code>) + round-operator	Core	P1
M4	Client keys, tins, ready tins kept in advance, Stash UI, on-device prover	App	P1
M5a	Browser mode : in-app browser + injected provider + transaction guard + simulation + per-tin storage wipe + auto-switch to the next ready tin + tunnel routing + signed	CC	P1

	contract allowlist + send-backs with clean-inbound attestations		
M5b	Foil browser extension (fallback if embedded browsers are blocked)	CC	P1–P2
M5	Key mode (when a key arrives): <code>cc-gateway</code> (CVM) + native rip flow + VRF + sell back + partner fee	CC	When key arrives
M6	Spend limits + region rules + machine display	App	P1
M7	Quiet Ship (tunnel, SIWE, bundle checks)	CC	P1
M8	Indexer + tin-set snapshots + card views	App	P1
M9	Gift links (pack and card)	Social	P2
M10	Rip Rooms + season handles + leaderboard	Social	P2
M11	\$FOIL, FeeRouter, staking, bonds	Token	P1

P1 needs M0–M4, M5a, M6–M8 and M11. **No P1 milestone depends on Collector Crypt replying.**

16. Launch gates

- Exit paths (`withdraw`, `exitPending`, `cancelRequest`, `selfExecute`, `send-backs`, `key export`) green on mainnet canary with the guardian pause on.
- Ceremony transcripts published.
- Browser mode green on mainnet canary (rip, sell back, ship from rotating tins). Key mode (and the `cc-gateway` attestation) only if a CC key is active.
- Spend limits on by default; region list reviewed; display rules live.
- OPSEC: fresh deployer, domain, RPC keys, CVM accounts, CC API key and contact, X manager and design system unique to Foil.
- Copy lint green.