# SPEC-1D — Lucid RDP: Build, Test, Run & Connectivity Outline

## Background

Lucid RDP is a Tor-only, blockchain-anchored remote desktop platform designed to record and prove what happened in each session without exposing plaintext. Every session is given a single-use ID; encrypted chunks and a manifest are anchored to an on‑system data chain while monetary payouts use TRON (USDT). Target runtime is a Raspberry Pi 5 (8GB) running Ubuntu Server; all services run in Docker and start on boot.

## Requirements (scope of this outline)

**Must** - Pi 5 + Ubuntu Server 64‑bit; all services containerized and auto‑start. - Tor‑only ingress/egress; clearnet disabled. - Session capture: host RDP, audit trail, chunk/manifest encryption and anchoring. - MongoDB 7 only (no SQL); replica/shard where applicable. - TRON integration isolated behind a Tron-Node service; dual modes: Shasta (test) and Mainnet (prod). - Minimal Admin UI for provisioning, proofs export, ledger mode switch, and key rotation.

**Should** - Hardware‑accelerated H.264 encode on Pi (V4L2) via FFmpeg. - Signed OTA updates and rollback. - Read‑only API for manifests/proofs and encrypted S3-compatible backups.

## Method (how components connect)

**Component graph**

@startuml  
skinparam componentStyle rectangle  
skinparam shadowing false  
  
package "Raspberry Pi 5 (Ubuntu Server)" {  
 [Admin UI (Next.js/Node 20)] as AdminUI  
 [RDP Host (xrdp/Wayland)] as RDP  
 [Session Recorder] as Rec  
 [Chunker+Compressor (Zstd)] as Chunker  
 [Encryptor (XChaCha20-Poly1305)] as Enc  
 [Merkle Builder (BLAKE3)] as Merkle  
 [On-System Chain Client] as OSC  
 [DHT/CRDT Node] as DHT  
 [Tron-Node Client (TronWeb)] as TronClient  
 [Wallet Daemon] as Wallet  
 [MongoDB 7] as DB  
 [Encrypted Chunk Store] as Store  
 [Tor HS & SOCKS] as Tor  
}  
  
package "On-System Data Chain" {  
 [LucidAnchors] as OSAnchors  
 [LucidChunkStore] as OSChunks  
}  
  
package "TRON (Shasta/Mainnet)" {  
 [PayoutRouterV0] as PR0  
 [PayoutRouterKYC] as PRKYC  
 [USDT-TRC20] as USDT  
}  
  
AdminUI -down-> RDP  
RDP --> Rec  
Rec --> Chunker --> Enc --> Store  
Enc --> Merkle  
Enc --> OSC : push encrypted chunks  
Merkle --> OSC : anchor(manifest/root)  
OSC --> OSAnchors  
OSC --> OSChunks  
AdminUI --> DB  
AdminUI --> TronClient : purchases, payouts  
Wallet -left- TronClient  
DHT <..> DHT : gossip/replicate (encrypted indexes)  
Tor .. AdminUI  
Tor .. OSC  
Tor .. TronClient  
Tor .. RDP  
PR0 --> USDT  
PRKYC --> USDT  
@enduml

**Data flow (per session)** 1) RDP session starts → Recorder captures audit/video → Chunker (8–16 MB) compresses (Zstd) → Encryptor wraps each chunk (XChaCha20‑Poly1305). 2) Merkle Builder computes BLAKE3 over ciphertext → session manifest with root + counts. 3) On‑System Chain Client anchors manifest/root (events preferred) and optionally per‑chunk anchors. 4) Admin can trigger payouts: TronClient calls PayoutRouterV0 (or KYC) to transfer USDT.

## Implementation (how it will be built and run)

**Platform & runtime** - Hardware: Raspberry Pi 5 (8GB) + NVMe. OS: Ubuntu Server 64‑bit. - All services run as Docker containers orchestrated by Docker Compose. Systemd ensures compose auto‑start on boot. - Networking: separate Docker networks for (a) service ops, (b) ledger/data‑chain, (c) wallet/admin UI; all routed via Tor.

**Languages & libs** - Node.js 20 LTS (Admin UI, Tron client, chain client, API), Python 3.12 optional tooling. - MongoDB 7 server + Node driver (or Mongoose). - FFmpeg with h264\_v4l2m2m (Pi HW encode) and xrdp for host. - Cryptography: libsodium (XChaCha20‑Poly1305), BLAKE3.

**Build layout (monorepo)**

/infra  
 docker-compose.yml  
 Dockerfile.base  
 docker/tor/, docker/mongo/, docker/xrdp/, docker/admin-ui/, docker/chain-client/, docker/tron-node/  
/apps  
 /admin-ui (Next.js)  
 /recorder (daemon + ffmpeg/xrdp helpers)  
 /chunker (native addon or Python)  
 /encryptor (libsodium binding)  
 /merkle (BLAKE3 binding)  
 /chain-client (Node service)  
 /tron-node (Node service using TronWeb)  
 /walletd (key mgmt)  
 /dht-node (CRDT/DHT)  
 /exporter (S3 backups)  
/contracts  
 LucidAnchors.sol, PayoutRouterV0.sol, PayoutRouterKYC.sol, ParamRegistry.sol, Governor.sol  
/ops  
 cloud-init/, ota/, monitoring/

**Build steps** 1) Cross-compile FFmpeg with Pi V4L2; bake into xrdp image. 2) Build Dockerfile.base (Node 20 + build deps), then per-app images. 3) Contracts: compile & test; deploy to Shasta (test) and record addresses. Bake addresses into images via build args. 4) Compose up: docker compose --env-file .env up -d on the Pi. 5) First‑run bootstrap: cloud‑init + QR provisioning; Admin UI reachable via .onion.

**Run modes** - **Sandbox/Test**: TRON Shasta, payout routers in test mode; optional PQ crypto for chunk keys. - **Production**: TRON Mainnet + USDT; PR0 (non‑KYC) and PRKYC (KYC‑gated) both deployed; runtime policy selects which to call.

## Test Plan — per script/service

All tests run on the Pi in CI (GitHub Actions runner on-device or self‑hosted). Use Docker to isolate dependencies. Integration and e2e tests route over Tor.

**1) /admin-ui (Next.js/Node)** - *Unit*: Jest/RTL — components (policy editor, payout forms), reducers, API clients. - *Integration*: Playwright — .onion access behind tor SOCKS; login (email magic + TOTP), manifest/proof export. - *Contract mocks*: MSW to simulate chain-client and tron-node APIs.

**2) /recorder (daemon)** - *Unit*: Python or Node tests for process control wrappers; synthetic frame generator. - *Integration*: Spawn xrdp + ffmpeg HW encode; verify chunks emitted/min; assert CPU < threshold. - *Faults*: Kill ffmpeg mid-session → expect auto-restart and gap markers.

**3) /chunker** - *Unit*: Chunk boundaries (8–16 MB policy), Zstd ratio ≥ baseline; deterministic splitting with same seed. - *Property tests*: QuickCheck-style for edge sizes.

**4) /encryptor (libsodium)** - *Unit*: XChaCha20‑Poly1305 nonce uniqueness; decrypt round‑trip; HKDF‑BLAKE2b key derivation vectors. - *Fuzz*: Feed corrupted chunks → verify AEAD fails and recorder flags incident.

**5) /merkle (BLAKE3)** - *Unit*: Root matches reference for known vectors; per‑chunk hash audit. - *Golden tests*: Fixed set of ciphertexts → compare root to golden.

**6) /chain-client (on‑system data chain)** - *Unit*: Manifest builder schema validation. - *Integration*: Start local on‑system chain (dev mode); submit registerSession & optional anchorChunk; assert events and idempotency.

**7) /tron-node (TronWeb)** - *Unit*: Router ABI bindings; reason‑code encoder. - *Integration*: Against Shasta — disburse (PR0) and disburseKYC (PRKYC) happy path; pause/unpause circuit breaker; per‑tx cap enforced. - *Security*: Negative tests for signature expiry and wrong signer (KYC route).

**8) /walletd** - *Unit*: Keystore lock/unlock; Ledger stub when HW present; 2‑of‑3 multisig flow simulation. - *Integration*: Key rotation via Admin UI; signed payout approval path.

**9) /dht-node (CRDT)** - *Unit*: CRDT merge (no conflicts) for encrypted indexes. - *Integration*: 3‑node gossip in Docker network; node churn; data availability checks.

**10) /exporter (S3 backups)** - *Unit*: Client‑side encryption envelope; MinIO integration. - *Integration*: Export/restore manifest+chunks; offline restore replays anchors.

**11) Contracts** - *Unit*: Solidity tests for LucidAnchors, PayoutRouterV0, PayoutRouterKYC, ParamRegistry, Governor. - *Integration*: Deploy to Shasta; event emission vs minimal storage; role gating; daily limits (PRKYC). - *Invariants*: No upgrade paths; ownership renounced; pausable works.

**12) End‑to‑End** - Tor‑only path: start session → anchor → request payout → USDT transfer seen; verify local DB sessions/chunks/payouts updated. - Failure drills: lost network, Tor restart, Mongo primary failover, TRON RPC timeout → all should degrade safely and recover.

**Automation** - make test-unit, make test-integration, make test-e2e targets; results exported to Admin UI Diagnostics.

## Implementation Steps (build → ship)

1. Stand up monorepo; scaffold services and Dockerfiles; compose networks and secrets.
2. Implement recorder/chunker/encryptor/merkle path with golden tests.
3. Implement chain-client and local on‑system chain dev net; wire Admin UI to show manifests.
4. Implement tron-node + walletd; deploy routers to Shasta; integrate payouts.
5. Enable Tor HS for Admin UI and APIs; enforce .onion‑only.
6. Add DHT/CRDT overlay for encrypted indexes.
7. Wire S3 backup/exporter; OTA update path.
8. Performance hardening; security review; Release Candidate image.

## Milestones

* **M1 (Weeks 1–2)**: Contracts + unit tests; recorder pipeline prototype.
* **M2 (Week 3)**: Shasta deploy; end‑to‑end (anchor+payout) on Pi.
* **M3 (Week 4)**: Mainnet addresses baked; compliance signer live.
* **M4 (Weeks 5–6)**: S‑features (encode, backups, APIs), KYC route toggle.
* **M5 (Week 7)**: RC image, OTA, docs.

## Gathering Results

* **On‑chain**: Export anchor/payout txids; reconcile with local Mongo.
* **Performance**: Anchor submit latency <300 ms (pre‑confirmation); USDT transfer initiation <2 s; Pi CPU and temp budgets respected.
* **Security**: Quarterly key rotation drills; chaos tests on testnet; incident logs zero false negatives for tampered chunks.

## Need Professional Help in Developing Your Architecture?

Please contact me at [sammuti.com](https://sammuti.com) :)