# SPEC-1A — Lucid RDP: Background & Requirements

## Background

Lucid RDP aims to deliver a **blockchain‑integrated Remote Desktop program** with a **unique single‑use ID** for every remote access session and a **trust‑nothing (zero‑trust)** policy that voids malicious use. It is a **peer‑to‑peer** remote desktop platform where every session is **auditable**, **privacy‑preserving**, and **economically sustainable**.

**Target runtime**: A **Raspberry Pi 5 (8GB RAM)** running **Ubuntu Server (64‑bit)**. All build, packaging, and CI tasks run **natively on a Pi 5** using a **mounted NVMe drive**.

**Network & privacy**: The **entire system operates via the Tor network**. All GUIs and APIs are exposed as **.onion** services; clearnet ingress is disabled. Peer connectivity, payout interactions, and data‑chain sync run through Tor.

**Economic model**: A **decentralized remote access** network where **nodes are paid monthly**. Production payouts use **TRON Mainnet + USDT‑TRC20** behind an isolated **Tron‑Node System** service; a sandbox testnet mirrors the flow for product testing.

**Data & consensus**: Each node becomes part of the **database system for all functions** via a DHT + CRDT overlay and contributes to an **On‑System Data Chain** that stores **encrypted, lossless‑compressed** session chunks and manifests **on‑chain** (never raw plaintext). A redundant encrypted copy also resides on the user’s device.

## Requirements

### Must Have (M)

* **R-MUST-001**: Server runs on Raspberry Pi 5 (8GB) + Ubuntu Server 64‑bit; all services containerized (Docker + Compose) and start on boot.
* **R-MUST-002**: Build & release pipelines execute **on Pi 5** using NVMe workspace; artifacts: multi‑arch Docker images (arm64) and Pi‑flashable appliance image.
* **R-MUST-003**: Remote desktop host support using open components (e.g., xrdp/FreeRDP or Wayland‑friendly equivalent) with clipboard and file‑transfer toggles.
* **R-MUST-004**: End‑to‑end encrypted P2P transport; Tor hidden‑service routing as default; session cannot be decrypted by third parties.
* **R-MUST-005**: Session audit trail: actor identity, timestamps, resource access, optional keystroke/window focus metadata; logs compressed, chunked, encrypted locally.
* **R-MUST-006**: Immutable anchoring: each session’s manifest/chunk hashes anchored to a blockchain ledger; local storage retains encrypted chunks.
* **R-MUST-007**: **Dual payout/ledger modes** selectable at deploy time: Mainnet (real payouts) and Testnet/Sandbox (no‑value payouts).
* **R-MUST-008**: Wallet management with hardware‑backed key storage (Pi TPM/secure element if available, else passphrase‑protected software vault) and role‑based access.
* **R-MUST-009**: Minimal web admin UI on the Pi for provisioning, viewing session manifests, exporting proofs, switching ledger modes, and rotating keys.
* **R-MUST-010**: Network isolation: separate Docker networks for ledger traffic, service ops, and wallet/admin; firewall rules enforced on Ubuntu.
* **R-MUST-011**: OTA update mechanism with signed releases and rollback.
* **R-MUST-012**: **Single‑use Session IDs** anchored on‑chain (one‑time ID + ephemeral keypair; non‑replayable).
* **R-MUST-013**: **Trust‑nothing policy engine** (default‑deny for input/clipboard/file transfer; JIT approvals; signed policy snapshot mismatch ⇒ session voided).
* **R-MUST-014**: **Tor‑only access** for all GUIs/APIs (.onion only).
* **R-MUST-015**: **Discrete TRON interaction** via an isolated **Tron‑Node System**; other services may not call TRON directly.
* **R-MUST-016**: **On‑System Data Chain** stores **encrypted, lossless‑compressed** session chunks/manifests on‑chain; redundant encrypted device copy.
* **R-MUST-017**: **Every node participates in the database overlay** (DHT + CRDT replication of encrypted metadata).
* **R-MUST-018**: **Monthly payouts**; node‑worker payouts via **PRKYC**, non‑worker/end‑user via **PR0**.
* **R-MUST-019**: **No SQL usage permitted** — all persistence uses **MongoDB** with replica sets and sharding (arm64‑compatible).
* **R-MUST-020**: **All service‑to‑service calls traverse Tor**; clearnet ingress disabled.

### Should Have (S)

* **R-SHOULD-001**: Hardware‑accelerated video encode (Pi 5 V4L2/FFmpeg) to minimize bandwidth and CPU.
* **R-SHOULD-002**: **Tor-only transport**; QUIC/UDP fallback is **not permitted**.
* **R-SHOULD-003**: Pluggable identity providers for operators (email‑magic link + TOTP), with local‑first accounts on Pi.
* **R-SHOULD-004**: API endpoints to fetch manifests, proofs, and session metadata; token‑scoped.
* **R-SHOULD-005**: Encrypted off‑appliance backups to S3‑compatible object storage.
* **R-SHOULD-006**: Basic rate‑limiting and anomaly detection for session abuse.
* **R-SHOULD-007**: Headless first‑run setup via HDMI or serial + QR code bootstrap.
* **R-SHOULD-008**: Optional observer role for read‑only session viewing when permitted by the sharer.

### Could Have (C)

* **R-COULD-001**: Enterprise SSO (OIDC/SAML) for the admin UI.
* **R-COULD-002**: KYC integration and payout caps for compliance when using Mainnet.
* **R-COULD-003**: Federation across multiple Pis for HA and shard replication of encrypted chunks.
* **R-COULD-004**: Mobile viewer apps.
* **R-COULD-005**: Post‑quantum crypto experiments (testnet only).
* **R-COULD-006**: Granular privacy filters (e.g., redact specific windows or keystrokes).

### Won’t Have (W) — for MVP

* **R-WONT-001**: Centralized relay dependency (system must function end‑to‑end P2P/Tor with no cloud relay required).
* **R-WONT-002**: x86\_64 server target or Windows server build.