Plano de Implementação Completa

SNE Vault Protocol

SNE Labs

23 de dezembro de 2025

Resumo

Este documento é a versão LaTeX do plano de implementação do SNE Vault Protocol. Ele cobre arquitetura física e on-chain, repositórios, fases de implementação, serviços, UI, relayer, zeroization, governança SNIPs, testes, deploy, riscos e checklist para validação completa.

Sumário

1 Visão Geral

Implementar uma infraestrutura física soberana para processamento de sinais de mercado com:

- Dual-Kernel Architecture: **SNE Radar (Intelligence) + SNE Pass (Custody)**.
- Proof of Uptime (PoU) — prova de existência física via ASIC/BitAxe.
- Zeroization — autodestruição física via Tamper-Detection Line.
- Governança SNIPs — votação ponderada por uptime.
- Handshake on-chain — ativação via Scroll L2.

2 Repositórios do Ecossistema

Repo	**Função / Conteúdo**
SNE-Labs/SNE-Labs	Frontend Dashboard, documentação, visualização de licenças, keys, boxes, PoU, governança.
SNE-Labs/SNE-Scroll-Passport	Componentes Web3 (Wallet, Balance, Gas), API Proxy e cache.
Scroll L2	Registry on-chain: SNELicenseRegistry, SNEPoURegistry, Governança SNIPs.
4LFR3Dv1/SNE-V1.0-CLOSED-BETA-	Smart contracts, ABIs, deploy scripts.
4LFR3Dv1/ordm-testnet	Ambiente de testes e validação.

3 Arquitetura do Ecossistema

3.1 Camada 1: Hardware Físico (Edge Nodes)

Componentes principais:

- SNE Box (Tier 3): Edge Controller (ARM/x86), BitAxe ASIC (PoU Generator), SNE Pass (Secure Element), Tamper-Detection Line.
- SNE Radar: ingestão, NTE (RAM), AVX-512 acceleration.
- SNE Pass: Kroot, HKDF, Sign, TPM Quotes.

3.2 Camada 2: Blockchain (Scroll L2)

Contratos principais:

- `SNELicenseRegistry` (ERC-721): gerenciamento de licenças.
- `SNEPoURegistry`: submissão de Merkle roots, contestação, uptime.
- `SNEGoverna nce`: SNIPs, votação e execução.

3.3 Camada 3: Frontend / Visualização

Dashboard com:

- Visualização de licenças, keys, boxes, status de nós, PoU, Merkle roots e governança SNIPs.
- SNE Scroll Passport: wallet connection, balance, gas tracker e API proxy.

4 Fases de Implementação

Resumo das fases com semanas estimadas.

Fase 0 — Preparação e Análise (Semana 1)

- Clonar repositório beta e extrair ABIs.
- Documentar endereços deployados.
- Configurar monorepo ou packages.
- Alinhar dependências (Viem, Wagmi, React, TS).

Fase 1 — Infraestrutura Base (Semanas 2–3)

- Configurar Wagmi/Viem no dashboard.
- Criar clients públicos e exports de contratos.
- Criar types TypeScript compartilhados.
- Configurar TanStack Query e TTLs por tipo de dado.

Fase 2 — Integração com Passport (Semanas 4–5)

- Integrar componentes de wallet, balance e gas.
- Remover dados hardcoded e usar hooks que leem contratos on-chain.

Fase 3 — Serviços On-Chain (Semanas 6–8)

Implementar serviços que leem contratos:

- `getLicensesForAddress`
- `checkLicenseAccess`
- `getNodeStatus`
- `getPoUData`
- `getMerkleRoots`
- `getSNIPs`
- `getVotingPower`

Fase 4 — UI Completa do Dashboard (Semanas 9–11)

Componentes:

- NodeStatusCard
- PoUDisplay
- SNIPsList
- HandshakeStatus
- Dashboard principal (integra todos)

Fase 5 — API Proxy no Passport (Semanas 12–13)

- Endpoints: `/api/sne/licenses`, `/api/sne/nodes/:id/status`, `/api/sne/nodes/:id/pou`, `/api/sne/governance/snips`, `/api/sne/merkle/roots`.
- Cache (Redis), rate limiting e tratamento de erros.

Fase 6 — Handshake Visualization (Semanas 14–15)

- Service para handshake status (read-only).
- Componente HandshakeStatus para visualização.

Fase 7 — Testes e Validação (Semanas 16–17)

- Testes unitários (Vitest), integração e E2E (Playwright).
- Testes on-chain (testnet, forking).

Fase 8 — Deploy e Produção (Semanas 18–19)

- Ambientes e variáveis (.env.production).
- Deploy: build e vercel/railway/infra própria.
- Monitoramento e documentação final.

5 Serviços On-Chain — Exemplos de Snippets

Abaixo estão snippets representativos que ilustram como os serviços interagem com Viem/publicClient. Ajustar ABIs e endereços conforme implementações reais.

Exemplo: getLicensesForAddress (Typescript)

Listing 1: getLicensesForAddress (exemplo)

```
export async function getLicensesForAddress(address: Address): Promise<SNELicense[]> {
  const logs = await publicClient.getLogs({
    address: contracts.SNELicenseRegistry,
    event: parseAbi([
      'event Transfer(address indexed from, address indexed to, uint256 indexed tokenId)'
    ]),
    args: { to: address },
  });

  // map logs -> licenses (ownerOf, getLicenseStatus, getNodeId)
}
```

Exemplo: Viem client (viem-client.ts)

Listing 2: viem-client.ts

```
import { createPublicClient, http } from 'viem';
import { scroll, scrollSepolia } from 'viem/chains';

const isTestnet = import.meta.env.VITE_NETWORK === 'testnet';
const chain = isTestnet ? scrollSepolia : scroll;

export const publicClient = createPublicClient({
  chain,
  transport: http(
    isTestnet ? 'https://sepolia-rpc.scroll.io' : 'https://rpc.scroll.io'
  ),
});
```

6 Relayer (Resumo)

- Verifica assinaturas das provas recebidas do hardware.
- Agrupa proofs em batches (ex.: 100), cria Merkle tree e submete `submitMerkleRoot`.
- Gerencia gas strategy (IMMEDIATE / OPTIMISTIC / BATCHED) com retries e backoff.
- Exponha métricas (Prometheus) e logs estruturados.

7 Zeroization — Modelo Proposto

- Zeroization Certificate: emitido pelo SNE Box após evento físico de zeroization. Contém metadata assinada pelo EK e um nonce.
- Evento on-chain `ZeroizationOccurred(nodeId, certHash, timestamp)` registra ocorrência sem expor segredos.
- Playbook de RMA e revogação de licenças.

8 Governança SNIPs

- SNIP lifecycle: draft → active → passed/rejected → executed.
- Voto ponderado por uptime (Pvoto integral ProofUptime(t) dt).
- Implementar quorum, timelocks e contestability.

9 Checklist de Implementação (MVP e Produção)

Task	Status / Nota
Clonar e analisar repositório beta	[ ]
Extrair ABIs e documentar endereços	[ ]
Configurar monorepo / packages	[ ]
Configurar Wagmi / Viem	[ ]
Criar types TypeScript	[ ]
Implementar serviços on-chain principais	[ ]
Criar relayer e gas-manager	[ ]
Zeroization proof model	[ ]
Indexador / Subgraph	[ ]
Auditoria de contratos	[ ]
Pentest hardware/firmware	[ ]
E2E testnet com hardware real	[ ]
Deploy e monitoramento	[ ]

10 Riscos Críticos e Mitigações

Segurança de chaves e cadeia de custódia

Risco: comprometimento físico.
Mitigação: HSM/TEE, attestation, provisioning seguro, key rotation, playbook de incidente.

Zeroization e evidência

Risco: provar zeroization sem vazar segredos.
Mitigação: ZeroizationCertificate + evento on-chain com hash e prova de attestation.

Escalabilidade de logs

Risco: uso de getLogs fromBlock: earliest.
Mitigação: indexer/subgraph, pagination, checkpoints.

Governança e ataques econômicos

Risco: voto comprável / Sybil.
Mitigação: quorum, timelocks, slashing, requisitos de bond.

11 Testes e Validação

- Unitários: vitest para serviços.
- Integração: testes com wagmi/viem providers mock.
- E2E: Playwright com fluxo de wallet.
- On-chain: Foundry/Hardhat (forking, reorg simulation).
- Hardware: testes de zeroization e attestation em bancada.

12 Deploy e Operação

- Ambientes: dev / testnet / mainnet (Scroll L2).
- CI/CD: pipelines para build, tests, deploy canary e rollback.
- Observability: Prometheus, Grafana, Sentry.
- Runbooks: incident response, zeroization drill, relayer outage.

13 Métricas de Sucesso

- Redução de dados hardcoded para 0%.
- Integração on-chain em 100% dos fluxos.
- Dashboard mostrando PoU, Merkle roots, status de nós.
- API proxy com cache hit rate $> 80\%$.
- Auditoria: zero high-severity outstanding.

14 Próximos Passos Imediatos

1. Criar Threat Model & SRS (prioridade alta).
2. Implementar indexer/subgraph.
3. Finalizar PoU economic model e stress-tests.
4. Planejar auditoria smart contracts e pentest hardware.
5. Executar testes E2E na testnet com hardware simulado.

Apêndices

A. Estrutura de arquivos sugerida

```
SNE-Monorepo/
          packages/
                sne-vault/
                sne-passport/
                sne-contracts/
                sne-sdk/
          apps/
                sne-box-firmware/
          package.json
```

B. Exemplo de .env.production

```
VITE_NETWORK=mainnet
VITE_SCROLL_RPC=https://rpc.scroll.io
VITE_WALLETCONNECT_PROJECT_ID=your_project_id
VITE_SNE_API_URL=https://pass.snelabs.space/api
VITE_SNE_LICENSE_REGISTRY=0x...
VITE_SNE_POU_REGISTRY=0x...
VITE_SNE_GOVERNANCE=0x...
```

C. Glossário

PoU Proof of Uptime.
EK Endorsement Key (Secure Element).
TPM Trusted Platform Module.
SNIP SNE Improvement Proposal (governança).

Documento gerado a partir do Plano de Implementação — SNE Vault Protocol.