

SNE Vault Protocol: Sovereign Physical Infrastructure

A Dual-Kernel Architecture for Truth & Custody (Hardened Specification)

Renan Melo

sneradar@gmail.com

<https://snelabs.space/>

Resumo

O SNE Vault é uma arquitetura físico-criptográfica projetada para transformar soberania operacional em propriedades técnicas auditáveis e verificáveis. Esta versão endurecida formaliza: (1) atestação de hardware com Endorsement Keys e TPM quotes; (2) um protocolo PoU (Proof of Uptime) baseado em desafio on-chain com respostas assinadas por hardware e agregadas via Merkle; (3) handshake de liberação de modelo com AEAD (AES-256-GCM) e HKDF; (4) mitigação explícita contra remanência de RAM, DMA e side-channels; (5) provisões de supply-chain e firmware assinado; (6) ingestão de feeds autenticados e redundantes; e (7) desenho econômico de governança (bonding/slashing/cap). Inclui apêndices com pseudocódigo e interfaces on-chain. O documento foi escrito para ser implementável, auditável e resistente a ataques praticáveis.

Sumário

1	Visão Geral e Objetivos Técnicos	4
2	Especificação Criptográfica (HKDF + AEAD)	4
2.1	Primitivas Criptográficas	4
2.2	Chave Raiz	4
2.3	Derivação da Chave de Sessão	5
2.4	Derivação do Nonce (IV)	5
2.5	Dados Autenticados Associados (AAD)	5
2.6	Criptografia e Decriptação	5
3	Protocolo de Contestação com Preservação de Privacidade	5
3.1	Construção da Leaf Merkle	6
3.2	Publicação On-chain	6
3.3	Procedimento de Contestação	6
3.4	Garantias de Privacidade	6
4	Resumo do Protocolo	6
5	Arquitetura de Hardware e Tiers	8
5.1	Classificação por Tier	8
5.2	Root of Trust e Endorsement Keys	8
6	Handshake de Ativação — Formalização	8
6.1	Assinaturas, nonces e derivação de chaves	8
6.2	Anti-replay e monotonic counters	8
7	Proof of Uptime (PoU) — Protocolo Formal	9
7.1	Objetivo	9
7.2	Mensagens e verificação	9
7.3	Agregação, contestação e slashing	9
8	Secure Element / Firmware / Provisioning	9
8.1	Measured boot e firmware signing	9
8.2	Provisionamento seguro e cadeia de custódia	10
9	Proteções físicas e zeroization robusta	10
9.1	Tamper Detection e zeroization	10
9.2	Mitigação de remanência de DRAM	10
10	Mitigações contra Side-Channel e Interfaces	10
10.1	Side-channel	10
10.2	I/O e Debug	10
11	Ingestão de Dados — Autenticação e Redundância	11
11.1	Feeds autenticados	11
11.2	Ordenação e timestamping	11
12	Governança e Economia	11
12.1	Bonding e Slashing	11
12.2	Limites e Caps	11

13 Threat Model — Precisão	11
14 Plano de Métricas e Auditoria	11
14.1 Testes Criptográficos	11
14.2 Testes de Hardware e Ataques Físicos	12
14.3 Testes de Execução e Latência	12
14.4 Testes On-chain	12
14.5 Critérios de Aprovação	12
15 Plano de Testes e Métricas Obrigatórias	12
15.1 Testes físicos	12
15.2 Testes criptográficos e protocolo	12
15.3 Métricas a publicar para piloto	13
16 Conclusão	14

1. VISÃO GERAL E OBJETIVOS TÉCNICOS

Objetivo Fornecer a operadores individuais a capacidade de executar leitura de mercado e assinar transações com provas verificáveis de que as operações ocorreram em hardware físico sob seu controle, preservando propriedade intelectual do motor de decisão (NTE).

Propriedades técnicas exigidas

- **Confidencialidade e integridade** dos artefatos proprietários (blobs θ) via AEAD (AES-256-GCM) e derivação KDF/HKDF.
- **Custódia isolada** por Secure Element / TPM com measured boot, PCR quotes e Endorsement Key (EK).
- **Atestação de corporeidade** por PoU challenge–response assinado por EK/ASIC, com agregação Merkle para eficiência on-chain.
- **Resistência física e logística** (tamper line, zeroization robusta e procedimentos de provisioning).
- **Robustez na ingestão de dados** (feeds autenticados, redundância, cross-validation).
- **Governança economicamente resistente** (bonding, slashing, limites por operador).

2. ESPECIFICAÇÃO CRIPTOGRÁFICA (HKDF + AEAD)

Este anexo define formalmente o esquema obrigatório de derivação de chaves, geração de nonce e criptografia autenticada utilizado pelos nós SNE Vault para execução segura de blobs criptografados θ .

2.1 Primitivas Criptográficas

- Função Hash: SHA-256
- KDF: HKDF (RFC 5869)
- AEAD: AES-256-GCM (nonce de 96 bits)
- RNG: CSPRNG com lastro em hardware (TPM / Secure Element)

noncer é gerado por CSPRNG com lastro em hardware no momento da construção da leaf.

2.2 Chave Raiz

Um segredo raiz vinculado ao dispositivo, denotado por K_{root} , é gerado durante o processo de provisionamento e armazenado exclusivamente dentro do Secure Element (SE) ou armazenamento NV do TPM.

- K_{root} **NUNCA** deve sair do hardware seguro
- K_{root} deve ser destruído imediatamente em caso de detecção de violação física
- Backup, exportação ou replicação de K_{root} são estritamente proibidos

2.3 Derivação da Chave de Sessão

Para cada sessão de execução, uma chave efêmera de criptografia K_{enc} é derivada conforme:

$$K_{enc} = \text{HKDF-Expand}(\text{HKDF-Extract}(K_{root}, salt), \text{"SNE-ENC"} \parallel \text{nodeID} \parallel t, 32)$$

Onde:

- $salt$ é um valor aleatório por sessão (128 bits)
- t representa um timestamp monotônico ou contador de sessão

O valor salt DEVE ser gerado por CSPRNG com lastro em hardware (TPM/SE) e NUNCA reutilizado entre sessões.

2.4 Derivação do Nonce (IV)

Cada nonce utilizado pelo AES-GCM DEVE ser único para cada instância de K_{enc} .

$$K_{enc} = \text{HKDF-Expand}(\text{HKDF-Extract}(K_{root}, salt), \text{"SNE-ENC"} \parallel \text{nodeID} \parallel t, 32)$$

Onde:

- ctr é um contador monotônico local da sessão (64 bits)
- Reutilização de nonce é considerada falha crítica de segurança

O contador ctr_IV utilizado para derivação de nonce AEAD é logicamente distinto do contador ctr_PoU incluído nas provas de uptime, mesmo que ambos sejam mantidos pelo Secure Element.

2.5 Dados Autenticados Associados (AAD)

Os seguintes dados devem ser vinculados ao ciphertext via AAD:

$$AAD = \text{nodeID} \parallel \text{firmware_version} \parallel t$$

Isso previne ataques de replay, reutilização cruzada entre dispositivos e downgrade de firmware.

2.6 Criptografia e Decifração

- Criptografia: $C = \text{AES-256-GCM-Encrypt}(K_{enc}, IV, \theta, AAD)$
- Qualquer falha de autenticação deve abortar a execução e acionar zeroização imediata

3. PROTOCOLO DE CONTESTAÇÃO COM PRESERVAÇÃO DE PRIVACIDADE

Este anexo define o mecanismo de contestação das provas de uptime (PoU), assegurando verificabilidade pública sem exposição indevida de metadados sensíveis ou propriedade intelectual.

3.1 Construção da Leaf Merkle

Cada leaf $leaf_i$ é construída conforme:

$$leaf_i = H(H(proof_i \parallel meta_{public}) \parallel commit_{private})$$

Onde:

- $proof_i$ é a prova assinada pelo EK do dispositivo
- $meta_{public}$ inclui timestamp e identificador do nó
- $commit_{private} = H(meta_{private} \parallel nonce_r)$

3.2 Publicação On-chain

Apenas a raiz Merkle R e metadados mínimos são publicados on-chain:

$$R = \text{MerkleRoot}(\{leaf_i\})$$

Nenhuma informação suficiente para reconstrução de θ ou do ambiente de execução é revelada nesta etapa.

3.3 Procedimento de Contestação

Em caso de contestação:

1. O contestante fornece $leaf_i$, prova de inclusão Merkle e $proof_i$
2. O operador revela $(meta_{private}, nonce_r)$
3. O contrato verifica $commit_{private}$ e a validade da assinatura EK

3.4 Garantias de Privacidade

- Metadados privados só são revelados sob contestação válida
- O reveal é limitado estritamente à leaf contestada
- Não há exposição de blobs θ ou chaves derivadas

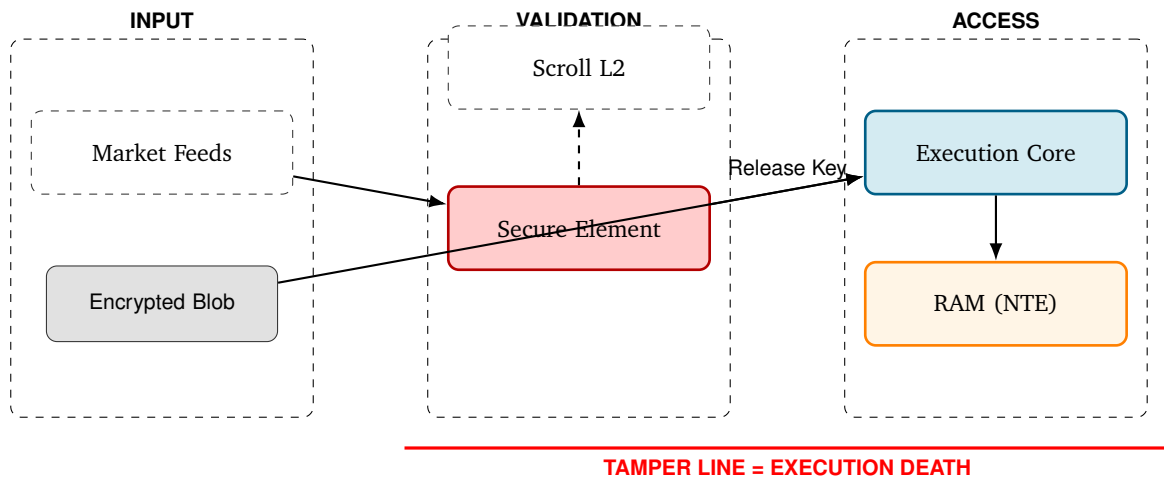
4. RESUMO DO PROTOCOLO

1. Operador solicita ativação física (PoP) e gera assinatura $\sigma = \text{Sign}(sk_{user}, n)$.
2. Secure Element verifica $\text{Verify}(pk_{user}, n, \sigma)$ e checa licença on-chain (SNELicenseRegistry).
3. Após verificação, Secure Element deriva $K_{enc} = \text{HKDF}(K_{root}, \text{info} \parallel nodeID \parallel n)$ e libera chave transitória para AEAD que decripta blob θ diretamente em RAM.
4. PoU: contrato na Scroll L2 emite $challenge_{L2}$; ASIC/SE responde com

$$quote = \text{Sign}_{EK}(H(challenge_{L2} \parallel ID_{node} \parallel \theta \parallel PCRs \parallel ctr))$$

respostas são verificadas pelo relayer, agregadas em Merkle root e publicadas on-chain.

5. Tamper detection aciona zeroization: overwrite sistemático da RAM, desligamento controlado e marcação on-chain do evento.



5. ARQUITETURA DE HARDWARE E TIERS

5.1 Classificação por Tier

- Tier 1** Servidores x86_64 com suporte AVX-512 para NTE de alta frequência; Secure Element TPM 2.0 discreto; BitAxe/ASIC opcional.
- Tier 2** Edge nodes x86/ARM de consumo com aceleração SIMD (NEON em ARM) e Secure Element (SE).
- Tier 3 (SNE Box)** Dispositivo dedicado com separação galvânica entre módulo de chaves e interface de rede; BitAxe ASIC para PoU; chassi tamper-resistant.

5.2 Root of Trust e Endorsement Keys

Cada dispositivo deve possuir:

- Uma **Endorsement Key (EK)** única, assinada pelo fabricante (CA_HW). EK é armazenada no SE/ASIC e não é exportável.
- Capability de **TPM quotes**: PCRs são atualizados durante measured boot; quotes incluem hashes de firmware, bootloader e versões aprovadas.
- Processo de **provisionamento** que registra EK_pub e metadados on-chain ou em repositório público de atestação, assinado pelo provedor de fabricação.

6. HANDSHAKE DE ATIVAÇÃO FORMALIZAÇÃO

6.1 Assinaturas, nonces e derivação de chaves

$$n \xleftarrow{\$} \{0, 1\}^{256}, \quad \sigma = \text{Sign}(sk_{user}, n)$$

SE realiza:

$$\text{Verify}(pk_{user}, n, \sigma)$$

Se válido e `checkAccess(nodeID) = true` on-chain:

$$K_{enc} = \text{HKDF}(K_{root}, \text{info} || \text{nodeID} || n)$$

K_{enc} é usado em AES-256-GCM (AEAD) para decriptar blob θ em RAM.

6.2 Anti-replay e monotonic counters

SE mantém um contador monotônico ctr incrementado a cada ativação; o quote / PoU incluem ctr para prevenir replay.

7. PROOF OF UPTIME (POU) PROTOCOLO FORMAL

7.1 Objetivo

Provar que o nó responde a desafios on-chain em tempo útil com assinatura vinculada ao EK e aos PCRs, prevenindo pré-computação / replay.



Physical Constraint
Energy + Heat + Time

7.2 Mensagens e verificação

1. Contract emite $challenge_{L2} \xleftarrow{\$} \{0, 1\}^{256}$ e registra $(challenge_{L2}, t_{emit})$.
2. Relayer encaminha challenge ao nó.
3. ASIC/SE computa:

$$payload = H(challenge_{L2} \parallel ID_{node} \parallel t \parallel PCRs \parallel ctr)$$

$$proof = \text{Sign}_{EK}(payload)$$

4. Nó retorna $proof$ e metadados $(ID_{node}, t, PCRs, ctr)$ ao relayer.
5. Relayer verifica $Verify_{EK}(proof, payload)$ e que EK foi emitida por CA_HW; agrega proofs em Merkle tree e publica root na Scroll L2.

A arquitetura admite múltiplos relayers concorrentes. A omissão ou censura sistemática de proofs por um relayer constitui falha econômica passível de slashing conforme política de governança.

7.3 Agregação, contestação e slashing

Publicar somente a Merkle root reduz custo on-chain; o contrato mantém janela e regras de contestação. Contestações válidas desencadeiam verificação off-chain e slashing do bond do operador conforme política.

8. SECURE ELEMENT / FIRMWARE / PROVISIONING

8.1 Measured boot e firmware signing

- Firmware/bootloader assinados por HW_sign e SW_sign; SE aceita apenas imagens válidas e verifica revogação via CRL ou manifest on-chain.
- PCRs cobrem estágios de boot e firmware; quotes servem como evidência de integridade.

8.2 Provisionamento seguro e cadeia de custódia

Provisionamento em ambiente controlado deve registrar:

1. EK_pub, certificado por CA_HW;
2. serial, batch, metadados de fábrica;
3. hash do firmware instalado;
4. assinatura do operador aceitando o device.

Registros podem ser publicados em repositório público de atestacao ou on-chain.

9. PROTEÇÕES FÍSICAS E ZEROIZATION ROBUSTA

9.1 Tamper Detection e zeroization

Chassi com sensores redundantes (corte, luz, temperatura). Procedimento de zeroization:

1. Overwrite da região de RAM usada pelo blob θ com padrões pseudo-aleatórios k passagens.
2. Invalidação de chaves transitórias no SE (set-to-null) e incremento de ctr .
3. Interrupção controlada de Vcc via power-fuse para reduzir remanência.
4. Registro seguro do evento (signed log) e, quando possível, notificação on-chain de estado Violated.

9.2 Mitigação de remanência de DRAM

Recomendações práticas:

- uso de módulos com refresh rápido e memory scrambling quando disponível;
- minimizar janelas de energia residual; evitar long power discharge;
- rotina de ruído/desgauss na alimentação em casos extremos (dentro de limites seguros).

10. MITIGAÇÕES CONTRA SIDE-CHANNEL E INTERFACES

10.1 Side-channel

- Implementações constant-time; masked / threshold signatures no SE quando aplicável.
- Blindagem EM, filtros de linha e injeção controlada de ruído durante operações sensíveis.
- Testes DPA/SPA obrigatórios em QA.

10.2 I/O e Debug

- JTAG/UART desabilitados fisicamente após provisioning; habilitação apenas em ambiente seguro com M-of-N multi-factor.
- IOMMU e whitelist de DMA.
- Telemetria mínima; dados sensíveis publicados apenas como commitments/ hashes on-chain.

11. INGESTÃO DE DADOS AUTENTICAÇÃO E REDUNDÂNCIA

11.1 Feeds autenticados

O nó deve:

1. validar assinaturas de pacotes/ticks;
2. manter $N_{feeds} \geq 3$ e aplicar regras de consenso (majority, median feed);
3. opcionalmente gravar hash commitments de batches para auditoria.

11.2 Ordenação e timestamping

Ingressos recebem timestamp seguro e ordenação determinística; discrepâncias acionam fall-back (delay until reconciliation).

12. GOVERNANÇA E ECONOMIA

12.1 Bonding e Slashing

Operadores depositam bond on-chain. Policies:

- proofs inválidos ou maliciosos resultam em slashing parcial;
- disputas possuem janela de contestação; vencedores recebem recompensa parcial.

12.2 Limites e Caps

Para reduzir risco de captura:

- cap sobre P_{voto} por operador;
- decaimento temporal do poder de voto exigindo manutenção contínua de uptime.

13. THREAT MODEL PRECISÃO

Modelo assume adversário com recursos de laboratório (decapsulation, micro-probing) capaz mas custoso. Pressuposições:

- atacante pode obter acesso físico temporário;
- atacante pode tentar comprometer supply-chain/firmware;
- atacante não tem acesso ao EK privado se EK estiver em SE com proteção adequada.

14. PLANO DE MÉTRICAS E AUDITORIA

Este anexo define os testes mínimos obrigatórios para validação do SNE Vault antes de qualquer implantação em produção.

14.1 Testes Criptográficos

- Verificação de unicidade de nonce (AES-GCM)
- Testes de falha forçada de autenticação
- Validação de entropia do RNG de hardware

14.2 Testes de Hardware e Ataques Físicos

- Cold-boot attacks (retenção de DRAM)
- Ataques por análise diferencial de potência (DPA)
- Testes de violação física e resposta do kill-switch

14.3 Testes de Execução e Latência

- Latência de handshake (p50 / p95 / p99)
- Tempo de derivação de chaves
- Overhead de zeroização pós-execução

14.4 Testes On-chain

- Submissão e validação de raiz Merkle
- Contestação válida e inválida
- Slashing e penalidades incorretas

14.5 Critérios de Aprovação

O sistema só é considerado apto para produção se:

- Nenhuma reutilização de nonce for detectada
- Todos os testes físicos resultarem em zero vazamento de chave
- Contratos on-chain forem auditados formalmente

15. PLANO DE TESTES E MÉTRICAS OBRIGATÓRIAS

15.1 Testes físicos

- Tamper suite (cortes, aberturas, choque térmico, vibração) — medir FPR / FNR.
- Cold-boot experiments para verificação de remoção de remanência.
- Decapsulation sampling para resistência a micro-probing.

15.2 Testes criptográficos e protocolo

- Verificação de EK chain: testes end-to-end de quote verification.
- PoU adversarial: simuladores de replay/precompute e contestação.
- Benchmarks E2E: ingestão $\rightarrow V_t \rightarrow$ inferência \rightarrow assinatura (median, p95, p99).

15.3 Métricas a publicar para piloto

- Latências (median/p99) do handshake e decisão.
- Taxa de heartbeats válidos / total emitidos.
- Resultados de testes tamper (FPR, FNR).
- Entropia do RNG (NIST tests) e RNG health logs.
- Relatórios de pentest físico/firmware.

APÊNDICE A PSEUDOCÓDIGO (POU / HANDSHAKE)

```
// On-chain contract (simplified)
contract SNELicenseRegistry {
    mapping(nodeID => bool) public authorized;
    mapping(windowID => bytes32) public merkleRoot;
    // functions: checkAccess, submitMerkleRoot, contestRoot, slash, etc.
}

// Node: Handshake
sigma = Sign(sk_user, n)
Send sigma to node
if Verify(pk_user, n, sigma) && checkAccess(nodeID):
    salt = CSPRNG_SE()
    K_enc = HKDF-Expand(
        HKDF-Extract(SE.K_root, salt),
        "model_load" || nodeID || n,
        32
    )
    theta = AES256GCM_Decrypt(K_enc, blob)
    load theta into RAM
else abort

// PoU loop (node)
while node_online:
    challenge = fetchChallengeFromRelayer()
    payload = H(challenge || nodeID || timestamp || PCRs || ctr)
    proof = Sign_EK(payload)
    send proof+meta to relayer
    ctr += 1

// Relayer
collect proofs window W:
    build MerkleTree(proofs)
    merkleRoot = root(Tree)
    submit merkleRoot to Scroll L2 contract
```

APÊNDICE B INTERFACES E CAMPOS

- **Quote payload:** $(challenge_{L2}, ID_{node}, t, PCRs, ctr)$
- **Proof on-device:** $proof = \text{Sign}_{EK}(H(payload))$
- **Merkle leaf:** $L_i = H(proof || meta)$

- **On-chain root:** $root = \text{MerkleRoot}(\{L_i\})$
- **Contract actions:** `submitRoot(window)`, `contestRoot(root, proof, leaf)`, `slash(operator)`

16. CONCLUSÃO

A especificação endurecida do SNE Vault converte afirmações de soberania em propriedades técnicas verificáveis: execução efêmera em RAM com AEAD/HKDF, custódia isolada com EK/TPM quotes, PoU challenge–response assinado por hardware e agregação Merkle, e políticas de governança com bonding/slashing. O desenho aceita limites da segurança física contra adversários laboratoriais de custo extremo e, em vez de negar essa realidade, ele aumenta o custo de ataque, torna eventos detectáveis e cria incentivos econômicos para operação honesta. Com métricas, planos de teste e interfaces on-chain, o SNE Vault fornece uma base prática e auditável para implantação em ambientes críticos.

REFERÊNCIAS ESSENCIAIS

1. Melo, R. (2025). *SNE Vault: Infrastructure for Physical Digital Sovereignty*. SNE Labs Whitepaper.
2. Trusted Computing Group. TPM 2.0 specifications.
3. NIST SP 800-90 / 800-38 for RNG and AES modes.
4. Papers on side-channel mitigations and threshold cryptography.