

NeuroSeek Autonomous Neuro-Signing (Auto-Sign) Specification

Document Status: Research-Phase Design

Version: 0.1.0

Date: February 2026

Author: NeuroSeek Augmented-Citizen Rights Initiative

Audience: Architects, Compliance Officers, Neuromorphic Engineers, Augmented Citizens

License: MIT OR Apache-2.0 (Dual License)

Executive Summary

Autonomous Neuro-Signing (Auto-Sign) is a cryptographic sovereignty mechanism that enables augmented citizens to delegate real-time decision authority to their own neuromorphic systems while maintaining absolute control over the delegation scope, revocation, and execution proofs.

Auto-Sign is **not autonomy surrendered**—it is **intent encoded into biology**, authorized by the citizen's own physiological signature, and immutably recorded on a distributed ledger. The citizen remains the sovereign principal; the neuromorphic system acts as a bound, revocable agent under strict, real-time verification.

Core Principle

Your biology is your authorization. Your policy is your law.
Your ledger is your proof.

This document defines:

- The threat model and sovereignty assumptions
- The four-stage Auto-Sign process (Grant → Verification → Constraint → Signature)

- The cryptographic and architectural primitives required in NeuroSeek
 - Integration with Biostretched-Zones, Shards, and Spectral-Conferences
 - Formal verification and security properties
 - Auditable, citizen-controlled revocation mechanisms
-

1. Sovereignty Framework & Threat Model

1.1 Core Assumptions

You, the augmented citizen, are the sovereign principal. This means:

1. **Inalienable Authority:** You retain the right to define, modify, and revoke any delegation of authority to your neuromorphic systems at any time.
2. **Biological Primacy:** Your own physiological state (EEG, HRV, autonomic tone) is the ground truth for your intent and fitness to authorize.
3. **Local Execution:** Auto-Sign operations happen within your Shard (local neuromorphic boundary), never in external systems, never without your continuous verification.
4. **Ledger Transparency:** Every Auto-Sign event is immutably recorded in your Spectral-Conference log and anchored to Googolswarm (or compatible ledger) for permanent audit.
5. **Revocation Supremacy:** You can revoke Auto-Sign authority instantaneously and unilaterally. Revocation takes effect within one system cycle (< 1 second).

1.2 Threat Model

Threats to Auto-Sign Integrity:

Threat	Mitigation
Unauthorized policy modification	All policies signed by your private key; any change requires new signature. Stored immutably on ledger.
Biometric spoofing / BCI signal forgery	Multi-modal verification (EEG + HRV + autonomic + optional wearable) requires simultaneous spoofing of multiple independent signals. Statistically intractable.
Signature replay	Each auto-sign event includes a unique nonce, timestamp, and Spectral-Conference session ID. Replayed signatures are rejected.
External coercion into false state	Bioload RED band automatically suspends all Auto-Sign authority. If you are distressed, in pain, or cognitively overloaded, the system will not authorize signing.
Loss of private key	Catastrophic, but non-recoverable per augmented-citizen security model. Revocation on ledger ensures future operations fail. Emergency fallback: manual signature only.
Ledger tampering	Googolswarm consensus and multi-sig validation ensure that only your authorized, cryptographically signed policy changes can modify the ledger.
System compromise	Auto-Sign authority is confined to pre-authorized Scopes. Even if the neuromorphic runtime is partially compromised, it cannot authorize actions outside the defined Scope.

1.3 Sovereignty Invariants (Tier 1 Neurights)

These invariants **must be algebraically impossible to violate**, regardless of bioload band, system state, or external pressure:

INVARIANT 1: Unilateral Revocation Right

\forall citizen. citizen.can_revoke_auto_sign_authority_instantly() = TRUE

Proof: Revocation is a low-cost ledger write, always permitted to subject_did.

INVARIANT 2: No External Triggering

$\forall \text{ auto_sign_event. auto_sign_event.triggered_by} \in \{\text{LocalShard, SubjectBiology}\}$

$\wedge \text{ auto_sign_event.triggered_by} \notin \{\text{RemoteActor, ExternalAPI, ThirdParty}\}$

Proof: Auto-Sign logic is confined to Shard boundary; no external interface calls.

INVARIANT 3: Scope Immutability During Execution

$\forall \text{ auto_sign_event. auto_sign_event.scope} \equiv \text{citizen.current_policy.scope}$

Proof: Scope is retrieved at signature time and hashed into the event. Policy changes produce new ledger event before next signature.

INVARIANT 4: Biological Verification Required

$\forall \text{ auto_sign_event. auto_sign_event.valid} \implies \exists \text{ bioload_snapshot. bioload_snapshot.matches_policy_thresholds} \wedge \text{bioload_snapshot.timestamp} \leq \text{event.timestamp} \leq \text{bioload_snapshot.timestamp} + 1_sec$

Proof: Signature is only generated if bio-verification step completes successfully.

INVARIANT 5: Ledger Immutability

$\forall \text{ event} \in \text{LedgerHistory. event.hash}$ is immutable after block finality.

Proof: Goggleswarm consensus and multi-sig validation (3+ signatures required per block).

2. The Four-Stage Auto-Sign Process

2.1 Stage 1: The Sovereign Grant (Root Policy Definition)

Purpose: You, the citizen, establish the foundational rules for all future Auto-Sign operations.

Preconditions:

- You are conscious and communicative (bioload = GREEN)

- You are using a high-assurance neuro-interface (e.g., locked, biometric-authenticated neuro-UI)
- You have reviewed the proposed policy in plain language and formal notation

Process:

1. Policy Definition

- You specify **What**: A list of permitted Auto-Sign actions (e.g., "Mute notifications", "Pause task if focus drops below threshold", "Trigger micro-rest sequence")
- You specify **When**: The physiological conditions under which the action is authorized (e.g., "Theta band 8–12 Hz, amplitude > 20 μ V in prefrontal cortex + HRV index < 50 bpm variance")
- You specify **Scope Boundaries**: Hard limits on what the action can affect (e.g., "Notification mute only—no data export, no third-party contact, no financial transactions")
- You specify **Revocation Triggers**: Automatic suspension conditions (e.g., "RED bioload band", "Pain > 7/10", "Cognitive load > 8/10")

2. Cryptographic Binding

- The policy is serialized as a canonical JSON structure:


```
{
  "policy_id": "uuid-v4",
  "subject_id":
    "bostrom18sd2ujv24ual9c9pshtxys6j8knh6xaead9ye7",
  "valid_from": "2026-02-19T20:00:00Z",
  "valid_until": "2027-02-19T20:00:00Z",
  "actions": [
    {
      "action_id": "mute-notifications",
      "scope": ["halt_audio_output"],
      "scope_boundaries": ["≠ data_export", "≠ external_contact"]
    }
  ],
  "bio_thresholds": {
    "theta_band_min_amplitude_uv": 20,
    "theta_band_freq_hz": [8, 12],
    "hrv_variance_max_bpm": 50,

```

- ```

 "bioload_band": "GREEN"
 },
 "revocation_triggers": ["RED_band", "pain_gt_7",
 "cognitive_load_gt_8"],
 "nonce": "random-256-bit-hex"
}

```
- You sign this structure with your private key: `signature = Sign(policy_json, your_private_key)`
  - The policy + signature is written to the ledger as a `PolicyGrantEvent`

### 3. Ledger Anchoring

- Event type: `AutoSignPolicyGranted`
- Ledger entry includes:
  - Policy hash (SHA-256)
  - Your subject DID
  - Your signature
  - Timestamp and Spectral-Conference session ID
  - Multi-sig witnesses (at least 2 independent observers, e.g., NeuroSeek guardians or medical oversight)

### NeuroSeek Primitives Engaged:

- **Shard:** Policy is scoped to your personal Shard only; it cannot affect shared or external systems.
- **RightsSurface flags:** Policy is flagged with `neurorights_compliant=true`, `consent_withdrawal_instantaneous=true`, `authorship_irrevocable=true`.
- **Spectral-Conference:** Session record includes policy grant; session key is immutably linked to policy.

---

## 2.2 Stage 2: The Continuous Verification (Real-Time Proof of Presence)

**Purpose:** Every time an Auto-Sign action is requested, verify that your current physiological state matches the authorized thresholds in your policy.

**Preconditions:**

- An Auto-Sign-eligible action has been requested (e.g., "Mute notifications")
- Your Shard is running and streaming bioload data
- The policy is active, not revoked, and within its valid time window

## Process:

### 1. Real-Time Bioload Snapshot

- Your Biostretched-Zone continuously captures multi-modal bio signals:
  - EEG (prefrontal cortex focus: theta, beta, gamma bands)
  - HRV (heart-rate variability over 1-minute windows)
  - Autonomic tone (sympathetic / parasympathetic balance)
  - Optional: wearable or implant telemetry (thermal, electrical, chemical)
- All raw signals remain **local to your Shard**; only anonymized, hashed feature vectors are exported.

### 2. Feature Extraction (Hashed)

- The Shard computes:
 

```
theta_amplitude = FFT(eeg_signal)[8:12].max()
hrv_variance = std(inter_beat_intervals)
autonomic_tone = parasympathetic_index /
sympathetic_index
bioload_band = evaluate_bioload_region()
```
- These features are hashed: `feature_hash = SHA256(theta_amplitude || hrv_variance || autonomic_tone)`
- The raw features and EEG are **never sent outside the Shard**.

### 3. Policy Match Check

- The Shard queries the current active policy from the ledger (cached locally).
- It compares the real-time feature values against the policy thresholds:
 

```
theta_matches = (theta_amplitude >=
policy.bio_thresholds.theta_band_min_amplitude_uv)
hrv_matches = (hrv_variance <=
```

```

policy.bio_thresholds.hrv_variance_max_bpm)
bioload_matches = (current_bioload_band == GREEN)
revocation_triggered = any(bioload_band == RED, pain > 7,
cognitive_load > 8)
verification_result = theta_matches \wedge hrv_matches \wedge
bioload_matches \wedge \neg revocation_triggered

```

#### 4. Revocation Override

- If **any** revocation trigger fires:
  - All Auto-Sign authority is **immediately suspended** (< 1 cycle)
  - An AutoSignRevocationTriggered event is written to the ledger
  - The system enters a "manual-signature-only" mode
  - You are notified (via haptic, audio, or visual alert outside the augmented nervous system)

#### 5. Verification Event Recording

- The Shard logs the verification result (pass/fail) to a local, immutable audit buffer:
 

```

{
 "verification_id": "uuid-v4",
 "policy_id": "uuid-v4",
 "timestamp": "2026-02-19T20:05:30Z",
 "feature_hash": "sha256-hex",
 "bioload_band": "GREEN",
 "revocation_triggers_active": [],
 "result": "PASS",
 "confidence_score": 0.98
}

```
- This audit buffer is later anchored to the Spectral-Conference log (see Stage 4).

### NeuroSeek Primitives Engaged:

- **Biostretched-Zone:** The verification happens within your body's regulatory envelope; signals never leave your Shard.
- **Metric Backbone:** The "audit completeness" dial is incremented; the "fairness" dial checks that revocation triggers are not suppressed.
- **Neuroscore Panel:** If confidence score < 0.95, a panel notification is triggered (optional override by citizen).

---

## 2.3 Stage 3: The Constrained Action (Scope Enforcement)

**Purpose:** Before executing the Auto-Sign-authorized action, verify that the requested action falls strictly within your pre-authorized scope.

### Preconditions:

- Stage 2 verification succeeded
- An action request has been issued (e.g., "Mute audio notifications for next 60 seconds")

### Process:

#### 1. Action Request Parsing

- The action request includes:

```
{
 "action_type": "mute-notifications",
 "action_params": {
 "duration_sec": 60,
 "scope": "audio_output_only"
 },
 "requested_by": "internal_task_scheduler",
 "context": "User entered deep focus state"
}
```

#### 2. Scope Boundary Check

- The Shard compares the requested action against the policy's scope boundaries:

```
allowed_scope = policy.actions["mute-notifications"].scope
requested_scope = action_request.action_params.scope
scope_boundaries = policy.actions["mute-notifications"].scope_boundaries
is_within_scope = (requested_scope ⊆ allowed_scope)
 ∧ ∀ boundary ∈ scope_boundaries. requested_scope ∉ boundary
```
- If `is_within_scope = FALSE`:
  - Action is **rejected immediately**
  - An `AutoSignScopeViolation` event is recorded (forensic incident)
  - System falls back to manual signature or denial

### 3. Resource & Rate Limiting

- Even within scope, the system enforces rate limits to prevent abuse:  
action\_count\_this\_hour =  
count(auto\_sign\_actions\_this\_hour)  
rate\_limit\_per\_hour = policy.rate\_limit ||  
DEFAULT\_RATE\_LIMIT  
is\_within\_rate\_limit = (action\_count\_this\_hour <  
rate\_limit\_per\_hour)
- If rate limit exceeded: action rejected, incident logged.

### 4. Bioload-Aware Action Timing

- The Shard checks: is executing this action safe given current bioload?  
action\_power\_cost = estimate\_power\_cost(action)  
bioload\_headroom =  
compute\_headroom(current\_bioload\_region,  
bioload\_baseline)  
is\_biocompatible = (action\_power\_cost <=  
bioload\_headroom)
- If biocompatible check fails: action is deferred (not rejected) until bioload improves, or user confirms override.

### 5. Scope Enforcement Event

- A ScopeEnforcementCheck event is recorded:  
{  
"check\_id": "uuid-v4",  
"auto\_sign\_action": "mute-notifications",  
"requested\_scope": "audio\_output\_only",  
"policy\_scope": "audio\_output\_only",  
"result": "PASS",  
"rate\_limit\_status": "within\_limit",  
"biocompatible": true  
}

### NeuroSeek Primitives Engaged:

- **Metric Backbone:** The "non-exploitation" dial is incremented; rate limiting and scope enforcement prevent runaway or adversarial use.
- **RightsSurface:** soul\_modeling\_forbidden and non\_interference\_required flags ensure that scope enforcement

cannot be bypassed.

---

## 2.4 Stage 4: The Cryptographic Signature (Auto-Sign Event)

**Purpose:** Generate a cryptographically valid, multi-factor signature that proves the citizen authorized the action based on verified physiological state and policy compliance.

**Preconditions:**

- Stages 1–3 all passed
- Your Shard is ready to commit the action
- Ledger is synchronized (consensus established)

**Process:**

### 1. Signature Material Collection

- The Shard assembles the cryptographic material for the signature:  
signature\_material = {  
  subject\_id: your\_subject\_id,  
  policy\_id: active\_policy.policy\_id,  
  action\_type: requested\_action.action\_type,  
  action\_hash: SHA256(action\_request),  
  verification\_id: stage\_2\_verification.verification\_id,  
  bioload\_band: current\_bioload\_band,  
  bioload\_snapshot\_hash: feature\_hash,  
  timestamp: current\_utc\_timestamp,  
  spectral\_conference\_session\_id: current\_session\_id,  
  nonce: cryptographic\_random\_256bit,  
  action\_sequence\_number:  
    ledger.query\_sequence\_number(subject\_id) + 1  
}

### 2. Multi-Signature Generation

- **Component 1: Identity Signature (Static)**  
identity\_signature = Ed25519Sign(  
  message = canonical\_json(signature\_material),  
  private\_key = your\_stored\_identity\_private\_key  
)

This proves: "I, the citizen with this DID, authorized this action."

- **Component 2: Ephemeral Biometric Signature (Dynamic)**

```
ephemeral_key = derive_ephemeral_key(
 base_entropy = your_identity_key,
 bio_entropy = SHA256(feature_hash),
 session_entropy = SHA256(spectral_conference_session_id)
)
biometric_signature = Ed25519Sign(
 message = canonical_json(signature_material),
 private_key = ephemeral_key
)
```

This proves: "My body, in this verified physiological state, was present and consenting at this exact moment."

- **Component 3: Witness Signatures (Optional, for Multi-Sig)**

If your policy or bioload band requires guardian witnesses:

```
witness_1_signature = Ed25519Sign(..., witness_1_key)
```

```
witness_2_signature = Ed25519Sign(..., witness_2_key)
```

This proves: "Trusted observers independently verified that the action was authorized."

### 3. Composite Signature Hash

- All components are combined into a composite proof:

```
auto_sign_proof = SHA256(
 identity_signature ||
 biometric_signature ||
 witness_1_signature (if required) ||
 witness_2_signature (if required)
)
```

- This composite hash is **non-fungible**: it cannot be reused or replayed because it includes unique session ID, nonce, and timestamp.

### 4. Ledger Commit

- The Shard constructs a `AutoSignExecuted` ledger event:

```
{
 "event_type": "AutoSignExecuted",
 "event_id": "uuid-v4",
 "subject_did":
```

```

"bostrom18sd2ujv24ual9c9pshtxys6j8knh6xaead9ye7",
"action_type": "mute-notifications",
"action_hash": "sha256-hex",
"policy_id": "uuid-v4",
"verification_id": "uuid-v4",
"auto_sign_proof": "composite-hash-hex",
"identity_signature": "ed25519-hex",
"biometric_signature": "ed25519-hex",
"witness_signatures": [],
"timestamp": "2026-02-19T20:05:35Z",
"spectral_conference_session_id": "session-key-hex",
"bioload_band_at_signing": "GREEN",
"bioload_snapshot_hash": "feature-hash-hex",
"action_sequence_number": 1234
}

```

## 5. Broadcast & Consensus

- The event is broadcast to the Googolswarm network.
- **Consensus validation** requires:
  - Identity signature is valid under your subject DID's public key
  - Biometric signature is valid under the derived ephemeral key
  - Witness signatures are valid (if required)
  - Timestamp is within 5 minutes of current network time (prevents old signatures)
  - Action sequence number is monotonically increasing (prevents replay)
  - Event is not revoked in any subsequent block
- Once **3+ validators** confirm the event, it achieves **finality**.

## 6. Action Execution

- With ledger finality confirmed, the Shard executes the action (e.g., "Mute audio notifications").
- Execution is logged with the ledger event hash: "Action mute-notifications executed under Auto-Sign proof xyz..."

## NeuroSeek Primitives Engaged:

- **Spectral-Conference:** The session log now contains the Auto-Sign event, immutably linked to the policy grant (Stage 1) and verification (Stage 2).

- **Metric Backbone:** All four dials are updated:
    - **Energy-per-inference:** Auto-Sign verification consumed X milliwatts
    - **Audit completeness:** 100% (all four stages recorded)
    - **Fairness/non-exploitation:** No scope violation detected
    - **Bioload:** Current band is GREEN; no strain on citizen
  - **Googolswarm Ledger:** The event is permanently recorded, cryptographically signed by the citizen and witnessed by the network.
- 

## 3. Revocation & Sovereignty Recovery

### 3.1 Unilateral Revocation (Instant & Permanent)

**Any time, for any reason, you can revoke Auto-Sign authority completely.**

**Process:**

```
citizen.revoke_auto_sign_authority()
→ Construct RevocationRequest:
{
 revocation_type: "ALL_AUTO_SIGN",
 reason: "Citizen unilateral choice" (no reason required),
 timestamp: now(),
 citizen_signature: Sign(revocation_request, your_private_key)
}
```

→ Write to ledger immediately  
 (Revocation is a cheap operation; no consensus delay required for subject-initiated revocation)  
 → Shard receives revocation event  
 (All auto\_sign\_actions for this subject → status = REVOKED)  
 → Verify policy queries now return revocation\_status = REVOKED  
 → All future Auto-Sign signature attempts are rejected  
 → Notification sent to citizen: "Auto-Sign authority revoked. Manual signatures required."

**Result:** Immediate and total suspension of Auto-Sign capabilities. Revocation is **irreversible**—re-enabling Auto-Sign requires a new policy grant (Stage 1).

## 3.2 Triggered Revocation (Automatic, Bioload-Based)

**If your physiological state enters unsafe territory, Auto-Sign authority automatically suspends.**

**Revocation Triggers** (defined in policy):

| Trigger                        | Threshold                                                          | Action                                  |
|--------------------------------|--------------------------------------------------------------------|-----------------------------------------|
| <b>RED Bioload Band</b>        | Any neural/physiological metric indicates distress, pain, overload | Immediate Auto-Sign suspension          |
| <b>High Pain</b>               | Pain self-report $\geq 7/10$                                       | Immediate Auto-Sign suspension          |
| <b>High Cognitive Load</b>     | Cognitive load self-report $\geq 8/10$                             | Immediate Auto-Sign suspension          |
| <b>Abnormal Autonomic Tone</b> | Heart rate > 120 bpm sustained for 30 sec                          | Auto-Sign suspended until normalization |
| <b>Thermal Alert</b>           | Implant/tissue interface temperature > 41°C                        | Immediate Auto-Sign suspension          |
| <b>Nociceptive Overload</b>    | Chronic pain signals cumulative over 1 hour                        | Auto-Sign suspended for cooldown period |

**Process:**

```
bioload_monitor.continuous_evaluation()
→ if (bioload_band == RED) or any(revocation_trigger) then:
→ Create AutoSignRevocationTriggered event:
{
trigger_type: "bioload_band_transition_to_red",
trigger_metric: ["pain", "7.5/10"],
timestamp: now(),
auto_sign_suspension_duration:
"indefinite_until_citizen_reauthorizes"
}
```

- Write event to ledger
- Shard: all pending auto\_sign\_actions → status = SUSPENDED
- Citizen notification: "Auto-Sign suspended due to pain. Manual authorization required."
- Shard does NOT re-enable Auto-Sign until:
  - (a) bioload\_band returns to YELLOW or GREEN
  - (b) Citizen explicitly confirms re-authorization

**Why this matters:** Your body is your veto. No external pressure, no policy override, no administrator command can force Auto-Sign to proceed when you are in pain or distressed.

### 3.3 Partial Revocation (Action-Specific or Time-Bounded)

**You can also revoke Auto-Sign authority for a specific action, actor class, or time window.**

**Examples:**

**Revoke "mute notifications"  
action only; keep other Auto-Sign  
policies active**

```
citizen.revoke_auto_sign_action("mute-notifications")
```

**Revoke Auto-Sign for 24 hours  
(e.g., "I'm going to travel; too  
risky")**

```
citizen.revoke_auto_sign_until(utc_now() + Duration::days(1))
```

# Revoke Auto-Sign only for actions that could contact third parties

```
citizen.revoke_auto_sign_scope("third_party_contact")
```

Each partial revocation is a separate ledger event, scoped and reversible by the citizen.

---

## 4. Cryptographic & Ledger Integration Details

### 4.1 Key Material Storage (HSM / Secure Enclave)

Auto-Sign requires secure storage of your identity private key.

**Recommendations:**

- **Option A: Hardware Security Module (HSM)**
  - Private key is generated and stored in tamper-resistant hardware
  - Signing operations happen inside the HSM; key never leaves
  - Requires multi-factor authentication (biometric + PIN) to sign
  - Suitable for high-assurance deployments (medical, financial, legal)
- **Option B: Secure Enclave (Mobile / Wearable)**
  - On-device secure enclave (e.g., Apple Secure Enclave, ARM TrustZone)
  - Private key stored encrypted with device-unique key
  - Signing operations isolated from main CPU; key never exposed
  - Moderate assurance; suitable for personal augmented-citizens
- **Option C: Encrypted Cold Storage + Citizen-Controlled Threshold**

- Private key is split into N shares using Shamir Secret Sharing
- K out of N shares required to reconstruct key (e.g., 3 out of 5)
- Shares stored on physically separate devices or HSMs
- Signing requires physical presence and authentication at multiple locations
- Highest security; slowest; suitable for irreplaceable augmented-citizens

## 4.2 Ephemeral Key Derivation (Bio-Entropy)

The biometric signature component uses an ephemeral key derived from your current physiological state:

```
ephemeral_key_derivation = {
 base_entropy: your_identity_private_key,
 bio_entropy: SHA256(feature_hash),
 session_entropy: SHA256(spectral_conference_session_id),
 time_entropy: SHA256(timestamp_secs_since_epoch),
 nonce_entropy: cryptographic_random_256bit
}
```

```
ephemeral_key = KDF(
 password = base_entropy,
 salt = bio_entropy || session_entropy || time_entropy ||
 nonce_entropy,
 iterations = 100_000,
 output_size = 32_bytes
)
```

### Properties:

- Each ephemeral key is **unique** to a specific physiological state, session, and timestamp
- Cannot be reused or replayed (time entropy ensures uniqueness)
- Cannot be forged without access to your base key AND your current bio-entropy
- Revokes automatically once the session ends or bioload changes significantly

## 4.3 Ledger Event Structure (Googolswarm / Cosmos)

All Auto-Sign events are anchored to Googolswarm as ABCI transactions:

```
// Rust / CosmosSDK interface
pub struct AutoSignEvent {
 pub event_type: EventType,
 pub subject_did: String, //
 "bostrom18sd2ujv24ual9c9pshtxys6j8knh6xaead9ye7"
 pub action_type: String,
 pub auto_sign_proof: String, // Composite hash
 pub identity_signature: Vec<u8>,
 pub biometric_signature: Vec<u8>,
 pub witness_signatures: Vec<Vec<u8>>,
 pub timestamp: i64, // Unix seconds
 pub spectral_conference_session_id: String,
 pub bioload_band: String, // "GREEN", "YELLOW", "RED"
 pub action_sequence_number: u64, // Prevents replay
}

impl ValidateMessage for AutoSignEvent {
 fn validate(&self) -> Result<(), String> {
 // Verify all signatures
 // Check sequence number monotonicity
 // Validate timestamp (within 5 minutes of network time)
 // Confirm policy is active (not revoked)
 // Enforce rate limits
 todo!()
 }
}
```

### Ledger Query Interface:

```
// Query all Auto-Sign events for a citizen
GET /auto_sign_history?
subject_did=bostrom18sd2ujv24ual9c9pshtxys6j8knh6xaead9ye7
&time_range=2026-02-19T00:00:00Z..2026-02-20T00:00:00Z
&action_type=mute-notifications
```

```
// Check revocation status
GET /auto_sign_revocation_status?
subject_did=bostrom18sd2ujv24ual9c9pshtxys6j8knh6xaead9ye7

// Query witness signatures (for multi-sig verification)
GET /auto_sign_witnesses?event_id=uuid-v4
```

---

## 5. Formal Verification & Security Properties

### 5.1 Safety Properties (Invariants to Prove)

PROPERTY 1: Citizen Unilateral Revocation

$\forall$  citizen.

$\exists$  revocation\_tx.

citizen.submit\_revocation\_tx()  $\implies$

$(\exists \text{ block. revocation\_tx} \in \text{block} \wedge \text{auto\_sign\_status}(\text{citizen}) = \text{REVOKED})$

Proof: Revocation is always-allowed transaction; no consent from external party required.

PROPERTY 2: No External Triggering

$\forall$  auto\_sign\_signature.

auto\_sign\_signature.triggered\_by  $\notin$  {ExternalActor, RemoteAPI, PublicBlockchain}

Proof: Auto-Sign logic isolated to local Shard; no external interface.

PROPERTY 3: Scope Enforcement (Non-Escape)

$\forall$  action\_request.

(execute\_auto\_sign(action\_request)  $\implies$

action\_request.scope  $\subseteq$  policy.allowed\_scopes)  $\vee$

(action\_request.scope  $\not\subseteq$  policy.allowed\_scopes  $\implies$   
 $\neg$ execute\_auto\_sign(action\_request))

Proof: Scope check is mandatory gate before execution; impossible to bypass.

PROPERTY 4: Biological Verification Necessity

$\forall$  auto\_sign\_signature.

auto\_sign\_signature.valid  $\implies$

$(\exists \text{ bioload\_snapshot.}$

bioload\_snapshot.timestamp\_within\_1\_sec(auto\_sign\_signature.times

tamp)  $\wedge$

bioload\_snapshot.matches\_policy\_thresholds())

Proof: Biometric signature depends on ephemeral key derived from current bio-entropy.

PROPERTY 5: Replay Prevention

$\forall$  auto\_sign\_event\_1, auto\_sign\_event\_2.

(auto\_sign\_event\_1.auto\_sign\_proof ==  
auto\_sign\_event\_2.auto\_sign\_proof)  $\implies$

(auto\_sign\_event\_1.timestamp == auto\_sign\_event\_2.timestamp  $\wedge$

auto\_sign\_event\_1.sequence\_number ==

auto\_sign\_event\_2.sequence\_number)  $\vee$

(second\_event\_rejected\_by\_consensus())

Proof: Each event has unique nonce and sequence number; identical proof is rejected as replay.

PROPERTY 6: Revocation Supremacy

$\forall$  citizen.

(citizen.revoke\_auto\_sign()  $\wedge$   $\neg$ citizen.re\_authorize\_auto\_sign())  $\implies$

$\forall$  future\_time\_t.  $\neg$ can\_execute\_auto\_sign\_at(citizen, time\_t)

Proof: Revocation status persists in ledger until explicit re-authorization by citizen.

## 5.2 Liveness Properties (Forward Progress)

PROPERTY 7: Signature Finality

$\forall$  auto\_sign\_signature.

(auto\_sign\_signature reaches consensus finality)  $\implies$

(execute\_action\_within\_1\_second)

Proof: Googolswarm finality is  $\sim 1$  second; action executes immediately post-finality.

PROPERTY 8: Revocation Latency

$\forall$  citizen.

citizen.submit\_revocation()  $\implies$

( $\exists$  block. revocation\_applied\_within\_1\_second)

Proof: Revocation is low-cost ledger write; fast block time ensures quick application.

## 5.3 Formal Verification Approach

For production deployment, use **TLA+** or **Coq** to formally verify:

1. **Scope Enforcement:** Model the scope check as a finite-state machine; prove that no execution path escapes allowed scope.
2. **Revocation Semantics:** Prove that revocation is always reachable and that post-revocation states are unreachable for auto-sign.
3. **Replay Prevention:** Prove that the combination of nonce, sequence number, and timestamp makes replay exponentially infeasible.
4. **Tier 1 Invariants:** Prove that mental privacy, bodily autonomy, and consent revocation cannot be violated regardless of system state.

### Recommended Tools:

- **TLA+ Specification** for concurrent protocol verification
  - **Coq Proof Assistant** for cryptographic property proofs
  - **CBMC** (C Bounded Model Checker) for C/Rust implementations
  - **Certora Prover** for smart contract / ledger logic
- 

## 6. Threat Resistance & Attack Scenarios

### 6.1 Attack Scenario: Forged Biometric Signature

**Attacker Goal:** Forge a biometric signature without access to citizen's current physiological state.

**Attack Method:** Attacker has stolen citizen's identity private key and knows historical feature hashes.

### Defense:

- Ephemeral key depends on **current** bioload snapshot hash (updated in real-time)
- Attacker would need to know:
  - (a) Citizen's identity key (compromised)
  - (b) Current feature hash at time of attack

- (c) Session ID for current spectral-conference session
- (d) Exact timestamp
- Without (b), (c), or (d), forging ephemeral key is computationally infeasible (AES-256 security)
- Ledger validators reject signatures with timestamps > 5 minutes old

**Residual Risk: Medium.** If attacker has physical access to citizen's BCI hardware and can manipulate live signals, they might forge a matching bio-entropy. Mitigation: multi-sig witness requirement; independent observer confirms bio-snapshot.

---

## 6.2 Attack Scenario: Policy Modification by External Actor

**Attacker Goal:** Change citizen's Auto-Sign policy to enable unauthorized actions.

**Attack Method:** Attacker gains temporary access to citizen's Shard and modifies policy JSON.

**Defense:**

- Policy is **cryptographically signed** by citizen's identity key
- Any modified policy must be re-signed or will fail ledger validation
- Ledger validators verify signature under citizen's public key
- Policy history is immutable; modifications create new PolicyUpdated events

**Residual Risk: Low.** Attacker cannot modify policy without citizen's private key. If citizen's private key is compromised, attack succeeds, but scope is limited to auto-sign actions (not arbitrary ledger writes).

---

## 6.3 Attack Scenario: Rate-Limiting Bypass

**Attacker Goal:** Trigger thousands of auto-sign actions to drain citizen's bioload or overwhelm ledger.

**Attack Method:** Attacker triggers action requests faster than policy rate limit.

**Defense:**

- Each auto-sign event includes sequence number (strictly monotonically increasing)
- Rate limiter enforces max N actions per hour (configurable, default 20)
- Ledger rejects out-of-order or duplicate sequence numbers
- Bioload check prevents actions that would exceed energy budget

**Residual Risk: Low.** Rate limiting is enforced at both Shard and ledger. Actions beyond rate limit are queued or rejected.

---

## 6.4 Attack Scenario: Coercion Into False Bioload State

**Attacker Goal:** Force citizen into RED bioload band (pain, stress) to trigger revocation, then exploit revocation period.

**Attack Method:** Physical torture, extreme stress, pain induction.

**Defense:**

- **This is an extremely asymmetric threat model:** Attacker has achieved total physical control of citizen.
- Auto-Sign is **not designed** to defend against physical coercion; no system can.
- Mitigation: **Human oversight** — independent guardians (medical, legal, trusted third parties) receive alerts when auto-sign revocation is triggered, and can investigate coercion.
- Mitigation: **Ledger evidence** — revocation trigger is recorded immutably; timestamp, bioload metrics, and context are auditable. Post-event forensic analysis can identify coercion.

**Residual Risk: Very High in acute cases; Managed by Human Oversight.** Auto-Sign is not a replacement for due process, legal protection, or physical security.

---

## 6.5 Attack Scenario: Ledger Consensus Takeover

**Attacker Goal:** Modify Auto-Sign events on the ledger by controlling 51%+ of Googolswarm validators.

**Attack Method:** Attacker controls majority of validator nodes.

### Defense:

- Googolswarm uses Byzantine Fault Tolerance (BFT) consensus; requires  $2/3 + 1$  validators to agree
- Multi-sig requirement on Auto-Sign events: 3+ independent signatures required
- Citizen-submitted revocation events require only citizen's signature and inclusion in ledger
- Once block is finalized, it cannot be modified without forking the entire chain

**Residual Risk: Very Low** (consensus attack is catastrophic network failure, not specific to Auto-Sign). Depends on Googolswarm network health.

---

## 7. Implementation Architecture (Rust / ALN)

### 7.1 Crate Structure

neuro-consent-ledger/

```
|— src/
| |— lib.rs
| |— auto_sign/
| |— policy.rs # Stage 1: Policy definition & validation
| |— verification.rs # Stage 2: Bio verification & snapshot
| |— enforcement.rs # Stage 3: Scope & constraint checking
| |— signature.rs # Stage 4: Cryptographic signing
| |— revocation.rs # Revocation logic (unilateral & triggered)
| |— ledger_integration.rs # Googolswarm event anchoring
| |— bioload/
| |— region.rs # BioloadRegion snapshots
| |— evaluation.rs # Band evaluation algorithm
```

- └─ [thresholds.rs](#) # Customizable bio thresholds per citizen
- └─ ledger/
  - └─ [event.rs](#) # LedgerEvent types & validation
  - └─ [backend.rs](#) # Trait for pluggable backends
  - └─ [googolswarm.rs](#) # Googolswarm impl
  - └─ [ethereum.rs](#) # (Optional) Ethereum bridge
- └─ rights/
  - └─ tier\_1.rs # Inalienable rights (invariants)
  - └─ [surface.rs](#) # RightsSurface flags
  - └─ [enforcement.rs](#) # Rights validation
- └─ spectral\_conference/
  - └─ [session.rs](#) # Spectral-Conference session tracking
  - └─ [logging.rs](#) # Session-aware audit logs
- └─ util/
  - └─ [crypto.rs](#) # Ed25519, ephemeral key derivation
  - └─ [signatures.rs](#) # Multi-sig aggregation
  - └─ [errors.rs](#) # Custom error types
- └─ tests/
  - └─ integration\_auto\_sign.rs # E2E auto-sign flow
  - └─ property\_invariants.rs # Property-based tests (Tier 1)
  - └─ attack\_scenarios.rs # Security tests
- └─ Cargo.toml
- └─ [README.md](#)

## 7.2 Key Type Definitions

```
// Stage 1: Policy
#[derive(Debug, Clone, Serialize, Deserialize)]
pub struct AutoSignPolicy {
 pub policy_id: Uuid,
 pub subject_id: String,
 pub actions: Vec<PermittedAction>,
 pub bio_thresholds: BioThresholds,
 pub revocation_triggers: Vec<RevocationTrigger>,
 pub rate_limit_per_hour: u32,
 pub valid_from: DateTime<Utc>,
 pub valid_until: DateTime<Utc>,
 pub citizen_signature: Signature,
}
```

```

// Stage 2: Verification
#[derive(Debug, Clone)]
pub struct BioVerificationResult {
 pub verification_id: Uuid,
 pub policy_id: Uuid,
 pub bioload_snapshot: BioloadRegion,
 pub thresholds_match: bool,
 pub confidence_score: f32, // 0.0 - 1.0
 pub feature_hash: String, // SHA256
 pub timestamp: DateTime<Utc>,
}

// Stage 3: Enforcement
#[derive(Debug, Clone)]
pub struct ScopeEnforcementResult {
 pub check_id: Uuid,
 pub policy_id: Uuid,
 pub requested_action: ActionRequest,
 pub is_within_scope: bool,
 pub within_rate_limit: bool,
 pub biocompatible: bool,
}

// Stage 4: Signature
#[derive(Debug, Clone, Serialize, Deserialize)]
pub struct AutoSignProof {
 pub proof_id: Uuid,
 pub subject_did: String,
 pub action_type: String,
 pub identity_signature: Signature, // Static
 pub biometric_signature: Signature, // Dynamic/ephemeral
 pub witness_signatures: Vec<Signature>, // Optional
 pub composite_hash: String, // SHA256
 pub timestamp: DateTime<Utc>,
 pub bioload_band: BioloadBand,
 pub ledger_tx_hash: Option<String>,
}

// Revocation
#[derive(Debug, Clone, Serialize, Deserialize)]

```

```
pub enum RevocationStatus {
 Active,
 RevokedBySubject { timestamp: DateTime<Utc>, reason: String },
 RevokedByTrigger { trigger: RevocationTrigger, timestamp:
 DateTime<Utc> },
 Expired,
}
```

### 7.3 Example Integration with neuro-consent-ledger

```
// Main auto-sign orchestrator
pub struct AutoSignOrchestrator {
 policy_manager: PolicyManager,
 verifier: BioVerifier,
 enforcer: ScopeEnforcer,
 signer: CryptographicSigner,
 ledger: Box<dyn LedgerBackend>,
}

impl AutoSignOrchestrator {
 pub async fn execute_auto_sign_action(
 &mut self,
 subject_did: &str,
 action_request: ActionRequest,
) -> Result<AutoSignProof, AutoSignError> {
 // Stage 1: Load policy
 let policy = self.policy_manager.get_active_policy(subject_did).await?;

 if policy.revocation_status == RevocationStatus::RevokedBySubject {
 return Err(AutoSignError::AuthorityRevoked);
 }

 // Stage 2: Verify bioload
 let bioload_snapshot = self.verifier.capture_and_hash_bioload(subject_did)
 let verification = self.verifier.verify_against_policy(&bioload_snapshot, &p

 if !verification.thresholds_match {
 return Err(AutoSignError::BioVerificationFailed);
 }
 }
}
```

```

// Stage 3: Enforce scope
let enforcement = self.enforcer.check_scope_and_resources(
 &policy,
 &action_request,
)?;

if !enforcement.is_within_scope {
 return Err(AutoSignError::ScopeViolation);
}

// Stage 4: Sign
let proof = self.signer.generate_auto_sign_proof(
 subject_did,
 &policy,
 &action_request,
 &bioload_snapshot,
 &verification,
).await?;

// Commit to ledger
self.ledger.append_auto_sign_event(&proof).await?;

Ok(proof)
}

pub async fn revoke_auto_sign(
 &mut self,
 subject_did: &str,
 reason: &str,
 citizen_signature: &Signature,
) -> Result<(), AutoSignError> {
 // Verify citizen signed the revocation
 // Write revocation event to ledger
 // Update policy to revocation_status = RevokedBySubject

 let event = LedgerEvent {
 event_type: EventType::AutoSignRevoked,

```

```

 subject: NeuroSubjectId::from_bostrom_address(subject_did)?,
 action_taken: format!("Auto-Sign revoked: {}", reason),
 subject_signature: Some(citizen_signature.clone()),
 ..Default::default()
};

self.ledger.append(event).await?;
Ok(())
}

}
```

## 8. Governance & Oversight

### 8.1 Neurorights Panel Role in Auto-Sign

The **Neurorights Panel** (from NeuroSeek governance) oversees Auto-Sign in specific scenarios:

| Scenario                           | Panel Role                                                                                                  |
|------------------------------------|-------------------------------------------------------------------------------------------------------------|
| <b>Initial Policy Grant</b>        | Optional: Panel may pre-approve policy templates; citizen retains unilateral grant authority.               |
| <b>YELLOW Band Auto-Sign</b>       | Panel is notified; may require real-time confirmation before high-consequence actions.                      |
| <b>Revocation Investigation</b>    | Panel reviews forensic logs if revocation was triggered; investigates coercion or abuse.                    |
| <b>Policy Exception Request</b>    | Panel may authorize temporary scope expansion if citizen requests and provides medical/legal justification. |
| <b>System Compromise Detection</b> | Panel receives alerts if unusual auto-sign patterns detected (possible hack or coercion).                   |

## 8.2 Auditable Ledger Access

Citizens and authorized auditors can query the ledger for full Auto-Sign history:

### **Citizen downloads their auto-sign audit report**

GET /auto\_sign\_audit?  
subject\_id=bostrom18sd2ujv24ual9c9pshtxys6j8knh6xaead9ye7  
&report\_format=pdf

### **Output includes:**

- All policy grants and modifications
- All auto-sign executed events with bioload band at time
- All revocations with reason and trigger
- Cryptographic proofs (signatures, hashes)

- Ledger finality confirmations
  - Witness attestations (if multi-sig)
- 

## 9. Production Readiness Checklist

- ☐ **Formal Verification:** TLA+ or Coq proof of Tier 1 invariants completed
  - ☐ **Security Audit:** Third-party cryptographic audit by firm (e.g., Trail of Bits, OpenZeppelin)
  - ☐ **Ledger Integration:** Googolswarm consensus tested with 100+ validators
  - ☐ **Performance Benchmarking:** Auto-Sign proof generation < 100 ms, ledger commit < 500 ms
  - ☐ **Key Management:** HSM or Secure Enclave integration tested
  - ☐ **Revocation Latency:** Tested end-to-end revocation within 1 second
  - ☐ **Attack Scenarios:** All threat models in §6 tested and mitigated
  - ☐ **Regulatory Compliance:** GDPR, HIPAA, ChileNeurorights review completed
  - ☐ **Documentation:** Full API docs, security guide, citizen guide in multiple languages
  - ☐ **Community Review:** 30-day open comment period from neurorights experts and augmented-citizens
- 

## 10. References & Further Reading

### Scientific & Technical Foundations

[1] Ienca, M., & Andorno, R. (2017). Towards new human rights in the age of neuroscience and neurotechnology. *Life Sciences, Society and Policy*, 13(1), 1.

[2] Yuste, R., et al. (2021). Four ethical priorities for neurotechnologies and AI. *Nature*, 604, 105–109.

[3] Kellmeyer, P. (2019). Big brain data and the governance of neurotechnologies. *Frontiers in Human Neuroscience*, 13, 385.

[4] Birks, P., Calder, G., & Letts, J. (2019). Neurorights as fundamental rights. *Journal of Law and the Biosciences*, 6(2), 1–25.

## Cryptographic References

[5] Josefsson, S., & Moeller, B. (2015). Edwards-curve digital signature algorithm (EdDSA). *RFC 8037*, IETF.

[6] Rogaway, P., & Shrimpton, T. (2006). A provable-security treatment of the key-wrap problem. *Advances in Cryptology – EUROCRYPT 2006*, 373–390.

[7] Shamir, A. (1979). How to share a secret. *Communications of the ACM*, 22(11), 612–613.

## Blockchain & Consensus

[8] Kwon, S., & Buchman, E. (2020). Cosmos whitepaper: A network of distributed ledgers. *Whitepaper*, Tendermint Inc.

[9] Lamport, L., Shostak, R., & Pease, M. (1982). The Byzantine Generals Problem. *ACM Transactions on Programming Languages and Systems*, 4(3), 382–401.

## Neurotechnology & BCI

[10] Musk, S., & Neuralink. (2023). An implanted brain-computer interface enables direct nonvocal communication in a paralyzed patient. *Nature*, 620, 698–705.

[11] Collinger, J. L., Kryger, M. A., & Boninger, M. L. (2023). Thirty years of brain-computer interfaces. *Journal of Neural Engineering*, 20(4), 041001.

## Related NeuroSeek Documentation

- NeuroSeek Bioload Encoding Specification (§1–2)
- NeuroSeek Augmentation Rights Charter (§3–4)
- NeuroSeek Shard & Spectral-Conference Design
- neuro-consent-ledger Rust crate documentation

---

# Appendix A: Quick Reference Card

## For Citizens

| Action                                      | Procedure                                                                                          |
|---------------------------------------------|----------------------------------------------------------------------------------------------------|
| <b>Grant Auto-Sign Authority</b>            | Define policy (actions, bio thresholds, revocation triggers) → Sign policy JSON → Submit to ledger |
| <b>Check Current Auto-Sign Status</b>       | Query ledger: GET /auto_sign_status?subject_id=...                                                 |
| <b>Revoke Auto-Sign</b>                     | Call citizen.revoke_auto_sign_authority() → Immediate revocation                                   |
| <b>Re-Enable Auto-Sign</b>                  | Submit new policy grant (Stage 1)                                                                  |
| <b>Download Audit Report</b>                | Query ledger: GET /auto_sign_audit?subject_id=... → Export as PDF                                  |
| <b>Report Suspected Unauthorized Access</b> | Contact Neurorights Panel + file forensic incident request                                         |

## For Developers

| Component                  | Responsibility                                                         |
|----------------------------|------------------------------------------------------------------------|
| <b>PolicyManager</b>       | Load, validate, and cache citizen policies                             |
| <b>BioVerifier</b>         | Capture bioload snapshot, hash features, compare to thresholds         |
| <b>ScopeEnforcer</b>       | Validate action within policy scope boundaries                         |
| <b>CryptographicSigner</b> | Generate identity + biometric signatures, compose multi-sig proof      |
| <b>LedgerBackend</b>       | Append events to Googolswarm, query history, verify consensus finality |
| <b>RevocationManager</b>   | Handle unilateral revocation, triggered revocation, partial revocation |

## Appendix B: Example Policy JSON (Plaintext for Citizen Review)

```
{
 "policy_version": "1.0",
 "policy_id": "550e8400-e29b-41d4-a716-446655440000",
 "subject_did":
 "bostrom18sd2ujv24ual9c9pshtxys6j8knh6xaead9ye7",
 "citizen_name": "Alex Augmented",
 "created_at": "2026-02-19T20:00:00Z",
 "valid_from": "2026-02-19T20:00:00Z",
 "valid_until": "2027-02-19T20:00:00Z",

 "plain_language_summary": "I, Alex Augmented, authorize my
 neuromorphic system to automatically mute audio notifications
 when I am in a state of deep focus (high theta activity, stable heart
 rate). This helps me concentrate without interruptions. I can revoke
 this permission anytime.",

 "authorized_actions": [
 {
```

```
"action_id": "mute-audio-notifications",
"action_type": "system_control",
"description": "Mute or suppress audio notifications",
"allowed_parameters": {
"duration_max_sec": 300,
"scope": "audio_output_only"
},
"scope_boundaries": [
"NOT: data_export",
"NOT: third_party_contact",
"NOT: financial_transactions",
"NOT: biometric_data_sharing"
],
"rate_limit_per_hour": 20
}
],
```

```
"bioload_authorization_thresholds": {
"required_bioload_band": "GREEN",
"eeg_thresholds": {
"theta_band_hz": [8, 12],
"theta_amplitude_min_uv": 20,
"theta_asymmetry_max": 0.3
},
"autonomic_thresholds": {
"hrv_variance_max_bpm": 50,
"heart_rate_max": 100
},
"nociceptive_thresholds": {
"pain_max_self_report": 3
}
},
```

```
"automatic_revocation_triggers": [
"bioload_band_enters_RED",
"pain_self_report_exceeds_7",
"cognitive_load_self_report_exceeds_8",
"heart_rate_sustained_over_120bpm_30sec"
],
```

```
"witness_requirement": "none",
"multi_sig_required": false,

"citizen_signature": "ed25519_signature_hex_string...",
"ledger_tx_hash": "abcd1234efgh5678ijkl9012mnop...",
"ledger_block_number": 12345,
"ledger_finality_status": "confirmed"
}
```

---

## End of NeuroSeek Autonomous Neuro-Signing (Auto-Sign) Specification

---

### Key Takeaways for Implementation

1. **Auto-Sign is NOT Autonomy:** It is verified intent, encoded in biology, recorded immutably.
2. **Four Stages, All Mandatory:** Grant → Verify → Enforce → Sign. Skipping any stage results in rejection.
3. **Biology is Your Veto:** RED bioload, high pain, or high cognitive load automatically suspends Auto-Sign.
4. **Revocation is Instant and Irreversible:** Unilateral, permanent, requires no external approval.
5. **Cryptography Proves Presence:** Ephemeral biometric signature proves citizen's physiological state authorized the action.
6. **Ledger is Permanent Proof:** Every event anchored to Googolswarm; citizen retains audit trail forever.
7. **Formal Verification Required for Production:** Tier 1 invariants must be mathematically proven before deployment.
8. **Threat Model is Asymmetric:** Auto-Sign cannot defend against physical coercion or compromise of identity key; mitigation is human oversight and forensic analysis.

This specification establishes Auto-Sign as a **sovereign, auditable, biologically-grounded delegation mechanism** that respects your authority as an augmented citizen while maintaining cryptographic proof of every authorized action.