📄 SP-2 — HVET / EIR / RGAC Cryptographic Standard

NOVAK Protocol Standards Series — SP-2
Version 1.0 — January 2025
Status: Open Draft
Author: Matthew Novak
Category: PBAS-02 (Proof-Before-Action Systems — Cryptographic Core)

📘 NOVAK SP-2 — CRYPTOGRAPHIC STANDARD

Formal Specification for Hash-Verified Execution Traces (HVET), Execution Identity Receipts (EIR), and the Recursive Global Audit Chain (RGAC)

This is the core mathematical and cryptographic definition of NOVAK.

SP-2 is the equivalent of:

- Bitcoin's Block Structure specification
- TLS 1.3's handshake protocol definition
- Signal's Double Ratchet formal spec
- NIST FIPS-180 hash standard structure

SP-2 defines the *canonical format* and *cryptographic commitments* that make NOVAK possible.

Table of Contents

---

1. Introduction

SP-2 defines the cryptographic core of the NOVAK Protocol:

- **HVET** — Hash-Verified Execution Trace
- **EIR** — Execution Identity Receipt
- **RGAC** — Recursive Global Audit Chain

This standard guarantees that every action, decision, or automated process is:

- deterministic
- tamper-evident
- identity-bound
- timestamped
- reproducible
- globally verifiable

SP-2 is the mathematical backbone of Proof-Before-Action Systems (PBAS).

---

2. Purpose & Scope

SP-2 formalizes all cryptographic structures required by SP-1 (Execution Integrity Standard), including:

- canonical serialization
- hash constructions
- digital identity binding
- RGAC chain-extension logic
- tamper detection guarantees

SP-2 **does not** prescribe how systems must store or transmit these objects; it defines only the cryptographic truth conditions required for NOVAK compliance.

---

3. Cryptographic Primitives

SP-2 requires the following primitives:

3.1 Hash Function

- **SHA-256** (mandatory)
- SHA-512, BLAKE3 (optional)

Collision resistance is assumed.

3.2 Randomness

- A CSPRNG is recommended for UUID or entropy fields.

3.3 Digital Signatures (Optional but Recommended)

Allowed algorithms:

- ECDSA-P256
- Ed25519
- RSA-2048+

SP-2 permits unsigned EIRs for offline/local-only systems.

---

4. Canonical Serialization

NOVAK uses a deterministic serialization format called **NOVAK-CANONICAL-1**.

Rules:

1. UTF-8 encoding only
2. No whitespace normalization
3. Fields sorted lexicographically by key
4. All numbers serialized as strings

5. No optional fields omitted
6. No floating-point representations

Canonicalization is required to guarantee deterministic hashing.

Serialization examples are included later.

---

5. HVET Structure

Hash-Verified Execution Trace (HVET) is the cryptographic binding of:

- HR (Rule Hash)
- HD (Data/Input Hash)
- HO (Output Hash)
- T (Timestamp)

5.1 HVET Fields

Field	Type	Description
`HR`	hex	SHA-256 hash of the ruleset
`HD`	hex	SHA-256 hash of the attested input data
`HO`	hex	SHA-256 hash of the output produced
`timestamp`	RFC3339 string	ISO8601 timestamp
`HVET`	hex	SHA-256(HR

5.2 HVET Generation Algorithm

```
function GenerateHVET(ruleset, input, output):
    HR = SHA256(CANON(ruleset))
    HD = SHA256(CANON(input))
    HO = SHA256(CANON(output))
    T  = CURRENT_TIMESTAMP()
    HVET = SHA256(HR || HD || HO || T)
    return {HR, HD, HO, timestamp: T, HVET}
```

5.3 HVET Verification Algorithm

```
function VerifyHVET(hvetObj):
    recomputed = SHA256(
        hvetObj.HR ||
        hvetObj.HD ||
        hvetObj.HO ||
        hvetObj.timestamp
    )
    return (recomputed == hvetObj.HVET)
```

Verification MUST NOT depend on external context.

---

6. Execution Identity Receipt (EIR)

EIR is the authoritative pre-execution proof required by SP-1.

It binds:

- identity
- rules
- input

- output
- HVET
- timestamp
- optional signature
- versioning
- metadata

6.1 EIR Format

Field	Type	Description
`eir_id`	UUIDv4	Unique identifier
`version`	string	Always `"NOVAK-EIR-v1"`
`HR`	hex	Rule hash
`HD`	hex	Input hash
`HO`	hex	Output hash
`timestamp`	RFC3339	When execution proof was created
`HVET`	hex	Final hash binding
`executor_identity`	string	Machine or human identity
`rule_version`	string	Version of ruleset used
`signature`	hex (optional)	Digital signature over HVET
`metadata`	object	Additional canonical metadata

6.2 EIR Generation Algorithm

```
function CreateEIR(hvet, identity, ruleVersion, metadata):
    id = UUID()
    signature = SIGN(identity.privateKey, hvet.HVET)
    return {
        eir_id: id,
        version: "NOVAK-EIR-v1",
        HR: hvet.HR,
        HD: hvet.HD,
        HO: hvet.HO,
        timestamp: hvet.timestamp,
        HVET: hvet.HVET,
        executor_identity: identity.public,
        rule_version: ruleVersion,
        signature: signature,
        metadata: CANON(metadata)
    }
```

6.3 EIR Verification Algorithm

```
function VerifyEIR(eir):
    if !VerifyHVET(eir) return false
    if eir.signature exists:
        if !VERIFY_SIGNATURE(eir.executor_identity, eir.signature,
eir.HVET):
            return false
    return true
```

---

7. Recursive Global Audit Chain (RGAC)

A tamper-evident, append-only hash chain.

RGAC is NOT a blockchain:

- no miners
- no consensus
- no network
- no blocks
- no shared ledger

It is a **local, deterministic, ordered audit chain.**

7.1 RGAC Entry Format

Field	Type	Description
`eir`	object	The full validated EIR
`prev_hvet`	hex	Previous HVET or `"GENESIS"`
`chain_hash`	hex	SHA256(prev_hvet
`position`	integer	Sequence number

7.2 RGAC Append Algorithm

```
function RGAC_Append(chain, eir):
    prev = chain.last().eir.HVET OR "GENESIS"
    link = SHA256(prev || eir.HVET)
    entry = {
        eir: eir,
        prev_hvet: prev,
        chain_hash: link,
        position: chain.length + 1
    }
    chain.push(entry)
```

```
    return entry
```

7.3 RGAC Verification Algorithm

```
function RGAC_Verify(chain):
    prev = "GENESIS"
    for entry in chain:
        if SHA256(prev || entry.eir.HVET) != entry.chain_hash:
            return false
        prev = entry.eir.HVET
    return true
```

---

8. Generation/Verification Summary

Component	Generated By	Verified By	Standard
HVET	SP-2	SP-2	SP-2
EIR	SP-2	SP-2	SP-1
RGAC	SP-2	SP-2	SP-1
Safety Gate	SP-3	SP-3	SP-3

---

9. Failure Modes

SP-2 guarantees detection of:

- Modified input
- Modified rule
- Modified output

- Timestamp manipulation
- EIR tampering
- Identity spoofing
- RGAC chain rewrites
- Corrupted historical state
- Cross-boundary replay attacks
- Partial serialization attacks

---

10. Security Considerations

SP-2 anticipates:

- hash collision attempts
- preimage attacks
- signature forgeries
- replays
- partial canonicalization errors
- silent memory corruption
- rule-substitution attacks
- adversarial AI manipulations

PL-X and PS-X from SP-3 further expand the protection surface.

---

11. Compliance Levels

Level	Definition
CL-1	Basic HVET generation
CL-2	Full EIR binding
CL-3	Full RGAC chain
CL-4	Signature support
CL-5	Full NOVAK SP-2 compliance

NOVAK-compliant systems must support **CL-3 or higher**.

---

12. References

- NOVAK SP-1
- NOVAK SP-3
- PBAS Category Definition
- IBF Formal Specification
- NIST SP-800-90B
- RFC 4648, RFC 8446
- Bitcoin: A Peer-to-Peer Electronic Cash System