Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 3, 2026

R.J. Coetzee
Violet Sky Security SEZC

March 3, 2026

Sovereign Policy Token Transactions (SPT-Txn) draft-coetzee-oauth-spt-txn-tokens-00

Abstract

This document specifies the Sovereign Policy Token Transaction (SPT-Txn) Framework, a token-based architecture for policy-bound, tamper-evident authorization across trust boundaries in agentic and multi-organizational systems. The framework extends IETF Transaction Tokens (RFC 9700) with Capability Acquisition Tokens (CATs) -- cryptographically scoped authorization tokens that bind capability grants to specific transaction contexts, propagate human-origin identity commitments across delegation chains, and support offline verification without issuer interaction.

SPT-Txn addresses the gap between coarse-grained bearer token authorization and the fine-grained, auditable, scope-limited authorization required in regulated financial infrastructure, AI agent systems, and cross-organizational transaction chains. The framework provides cryptographic guarantees that every hop in a transaction chain was authorized by a verified human principal, operated within a declared capability scope, and can be audited without exposing personally identifiable information.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 3, 2026.

Copyright Notice

Internet-Draft SPT-Txn Tokens March 2026

Table of Contents

1. Introduction 3
1.1. Terminology 5
1.2. Changes from -03 6
2. Problem Statement 7
2.1. Limitations of Current Txn-Tokens 7
2.2. Requirements 8
3. TBAC Capability Token Model 10
3.1. The ABAC-to-TBAC Boundary 10
3.2. Capability Token Structure 11
3.3. Eight-Step Enforcement Engine 13
3.4. Scope Invariants 15
3.5. Token Serialization Formats 16
4. SPT-Txn Token Structure 19
4.1. Base Claims (Inherited) 19
4.2. SPT-Txn Extension Claims 19
5. ZK Circuit Specification 22
5.1. Proof System 22
5.2. Circuit 1: Attribute Proof 22
5.3. Circuit 2: Compliance Claim 24
5.4. Circuit 3: Biometric Commitment 26
6. Multi-Chain Trust Registry Model 28
6.1. Chain-Agnostic Trust Registry 28
6.2. Chain-Specific Deployment Notes 29
6.2.1. Ethereum / EVM-Compatible Chains 29
6.2.2. XDC Network 30
6.2.3. Algorand 30
6.2.4. Hedera Hashgraph 31
7. Transaction Flow 32
8. Security Considerations 36
8.1. Capability Token Security 36
8.2. Delegation Chain Security 37
8.3. ZK Proof Security 37
8.4. Trust Registry Security 38
9. Privacy Considerations 39
10. Token Format Security Properties 40
10.1. Signature Algorithm Requirements 40
10.2. Key Management 40
10.3. Transport Security 41
10.4. Replay Prevention 41
10.5. Token Format Security Properties 41
10.6. ZK Proof System Security 42
10.7. Multi-Chain Trust Registry Integrity 43
11. IANA Considerations 44
12. References 46
12.1. Normative References 46
12.2. Informative References 47
Author's Address 48

1. Introduction

Authorization in distributed and agentic systems requires answering three questions with cryptographic certainty at every transaction hop:

(a) Was this action authorized by a verified human principal?
(b) Does the acting entity operate within an explicitly declared

capability scope?
(c) Can the full chain of authorization be audited without accessing personally identifiable information?

Existing approaches cannot answer all three simultaneously. Bearer tokens (OAuth 2.0 [RFC6749]) answer none of them at the cryptographic layer. Transaction Tokens (RFC 9700) address workload identity propagation but not capability scoping or privacy-preserving human traceability. Demonstrating Proof of Possession (DPoP, RFC 9449) binds tokens to a key pair but not to a transaction context or a human-origin commitment.

The consequence is that when an AI agent or automated service takes an action that causes harm, the audit record is a log. Logs are not proof. They are mutable, selectively produced, and unavailable to external auditors without system access. What regulated environments require is cryptographic proof of authorization scope at every hop -- immutable, verifiable by any party holding the appropriate public key, and auditable without PII exposure.

This document specifies the Sovereign Policy Token Transaction (SPT-Txn) Framework, which provides this capability through two coordinated mechanisms:

Capability Acquisition Tokens (CATs): Cryptographically signed authorization tokens issued by an Attribute-Based Access Control (ABAC) Policy Decision Point (PDP), binding a specific capability scope to a holder identity, transaction context, and delegation depth. CTs are verifiable offline and cannot be used outside their declared scope.

SPT-Txn Transaction Tokens: An extension of IETF Transaction Tokens that adds CT references, scope hashes, human-origin commitments (humanAnchor), and compliance proof references to every hop in the transaction chain. The chain is immutable and carries cryptographic evidence of its full authorization history.

Together, these mechanisms enable regulated financial systems, AI agent orchestration platforms, and cross-organizational transaction chains to provide cryptographic proof that every action was authorized by a verified human, within declared scope, with a complete and auditable chain of custody -- without exposing identity information to intermediate parties.

The framework is designed for direct alignment with IETF standards. CTs are normatively serialized as JSON Web Tokens [RFC7519] with JSON Web Signatures [RFC7515]. SPT-Txn Transaction Tokens extend the base claims defined in RFC 9700. The ZK proof system uses Groth16 over BN254, with circuit specifications at the normative level. Trust registries are chain-agnostic and support Ethereum, XDC, Algorand, and Hedera deployments.

The SPT-Txn Framework directly addresses the requirements of the EU AI Act Article 14 (human oversight of high-risk AI systems). The humanAnchor claim provides cryptographic traceability to the authorizing human; CT scope provides bounded authorization that the AI system cannot exceed; the revocation mechanism provides the override capability Article 14 requires; and the HCS audit chain provides monitoring evidence without PII exposure.

Formal cryptographic security definitions and proofs for the token binding security property are provided in the companion theory paper [SPT-THEORY]. The theory paper defines transaction binding security as a game-based security property and proves that the SPT-Txn construction achieves it under EUF-CMA assumptions in the random oracle model.

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

This document uses the following terms:

Capability Acquisition Token (CAT):
: A signed token issued by an ABAC Policy Decision Point, binding a capability scope to a holder identity and transaction context. Normative serialization: JWT [RFC7519]. SD-JWT, Biscuit, and CWT are alternative formats (Section 3.5).

Capability Token (CT):
: Synonym for Capability Acquisition Token. Used interchangeably in this document.

ABAC PDP:
: Attribute-Based Access Control Policy Decision Point. The issuing authority for Capability Tokens. Evaluates policy against subject attributes and issues scoped CTs.

TBAC:
: Token-Based Access Control. The enforcement model in which resource servers verify presented tokens rather than querying a central policy engine at each request.

humanAnchor:
: A zero-knowledge commitment to the zkDID of the human principal who authorized the root transaction. Propagated immutably through all delegation hops. Enables human traceability without PII exposure.

zkDID:
: A zero-knowledge Decentralized Identifier commitment that proves identity membership in a verified set without revealing the underlying identifier.

delegation_depth:
: The remaining number of sub-delegation levels permitted from the current CT holder. Decremented at each delegation. A CT with depth 0 MUST NOT issue further delegation CTs.

Trust Registry:
: An on-chain smart contract registry that maps ABAC PDP identities to capability types they are authorized to certify. Used by resource servers to verify CT issuer authority without contacting the issuer.

Transaction Token (Txn-Token):
: As defined in RFC 9700. A short-lived signed token representing a workload’s identity and context within a transaction chain. SPT-Txn extends the base Txn-Token claim set.

SPT-Txn Token:

An SPT-Txn-extended Transaction Token carrying the additional claims defined in Section 4.2.

1.2. Changes from -03

The following changes were made in this revision:

(a) Section 1.1 (Terminology): CT definition updated from "signed JWT" to "signed token" with a normative note: "Normative serialization: JWT. SD-JWT, Biscuit, and CWT are alternative formats (Section 3.5)."

(b) Section 3.2: Added "normatively" qualifier to the CT serialization statement.

(c) Section 3.3 Step 1: Signature verification now specifies JWS for JWT/SD-JWT, public-key block verification for Biscuit, and COSE verification for CWT.

(d) Section 3.5 (new): Token Serialization Formats. Full section with RFC 2119 normative language covering SD-JWT selective disclosure and data minimisation, Biscuit cryptographic attenuation with Ed25519 and Datalog, CWT for high-throughput agent-to-agent, and format selection criteria.

(e) Section 6.2.4: Hedera Hashgraph section numbering corrected (was duplicate 6.2.3 in -03).

(f) Section 10.5 (new): Token Format Security Properties. Ed25519 and COSE alongside ECDSA; post-quantum vulnerability note; Biscuit structural attenuation defence. Existing Sections 10.5 and 10.6 renumbered to 10.6 and 10.7.

(g) References: Added SD-JWT [I-D.ietf-oauth-selective-disclosure-jwt], RFC 8392 (CWT), and Biscuit [BISCUIT]. Updated SPT-WP reference from v3.0 to v6.0.

2. Problem Statement

2.1. Limitations of Current Txn-Tokens

RFC 9700 Transaction Tokens solve the workload identity propagation problem: a downstream service can verify that a call originated from a known, authenticated workload rather than an arbitrary external caller. This is a significant security improvement over bearer token architectures.

However, RFC 9700 Txn-Tokens have four limitations that make them insufficient for regulated agentic systems:

No capability scoping: A Txn-Token proves that a workload made a call; it does not prove that the workload had authorization to make that specific call with that specific scope. The workload’s authorization is evaluated separately by a policy engine at

each hop, creating a distributed policy evaluation problem in cross-organizational deployments.

No human traceability: Txn-Tokens propagate the "sub" claim identifying the originating user. In cross-organizational chains, this propagates PII to every service in the chain, creating GDPR data minimization violations. Alternatively, organizations strip the subject claim, losing human traceability entirely.

No cross-organizational trust: RFC 9700 assumes a shared authorization server within a single trust domain. When a transaction crosses organizational boundaries, services in different organizations cannot verify each other's Txn-Tokens without a pre-established trust relationship at the authorization server level.

No compliance propagation: Regulated transactions require compliance attestations (KYC, AML, jurisdiction verification) at the root. RFC 9700 provides no mechanism for propagating compliance evidence through the chain without re-verification at each hop.

2.2. Requirements

The SPT-Txn Framework is designed to satisfy the following requirements:

REQ-1: A resource server MUST be able to verify that the presenting workload holds a valid capability grant from an authorized issuer, scoped to the requested operation, without contacting the issuing authority at request time.

REQ-2: The capability grant MUST be cryptographically bound to a specific transaction context such that it cannot be reused in a different transaction without detection.

REQ-3: Every hop in a transaction chain MUST carry a cryptographic commitment to the human principal who authorized the root transaction. This commitment MUST NOT reveal the human's identity to intermediate services.

REQ-4: Capability scope MUST be monotonically non-increasing through the delegation chain. No delegatee MAY hold greater capability than its delegator.

REQ-5: The maximum delegation depth MUST be set at root issuance and enforced cryptographically at each delegation step.

REQ-6: Resource servers in different organizations MUST be able to verify capability grants issued by issuers in other organizations using a shared trust registry, without bilateral pre-configuration.

REQ-7: Compliance attestations established at transaction root MUST be propagatable to downstream services by reference, without re-exposing the underlying compliance data.

REQ-8: The complete authorization chain MUST be auditable by an authorized auditor without requiring access to PII.

3. TBAC Capability Token Model

3.1. The ABAC-to-TBAC Boundary

The SPT-Txn Framework operates at the boundary between Attribute-Based Access Control (ABAC) and Token-Based Access Control (TBAC).

ABAC evaluation is expensive: it requires policy retrieval, attribute collection, and real-time evaluation at each access decision. In high-throughput transaction chains, ABAC evaluation at every hop introduces unacceptable latency and creates availability dependencies on the ABAC Policy Decision Point.

The SPT-Txn approach performs ABAC evaluation once, at the transaction root, and materializes the result as a Capability Token -- a signed, scope-limited, cryptographically verifiable authorization artifact. Downstream enforcement is TBAC: verify the token, check the scope, make the decision. No policy engine call required.

This boundary is the fundamental architectural claim of SPT-Txn: ABAC at the root, TBAC at every subsequent hop.

3.2. Capability Token Structure

A Capability Token (CT) is normatively a JSON Web Token [RFC7519] signed with JSON Web Signature [RFC7515]. Alternative serialization formats are specified in Section 3.5.

CTs MUST contain the following claims:

iss: The DID or HTTPS identifier of the ABAC PDP that issued this CT. REQUIRED.

sub: The identity of the CT holder. SHOULD be a workload identity (SPIFFE SVID, DID, or equivalent). REQUIRED.

iat: Issued-at time. REQUIRED.

exp: Expiry time. REQUIRED. CTs MUST have a finite lifetime.

jti: JWT ID. A unique identifier for this specific CT. REQUIRED. Used for revocation registry lookups.

ct_type: The capability type granted. MUST be a URI identifying the capability. REQUIRED.

ct_scope: The capability scope granted. A structured object defining the specific operations, resources, and constraints permitted. REQUIRED.

human_anchor: A zero-knowledge commitment to the zkDID of the authorizing human principal. REQUIRED for CTs in agentic chains. Format: bytes32 (Poseidon hash of zkDID commitment).

delegation_depth: The remaining delegation depth permitted from this CT holder. REQUIRED. MUST be a non-negative integer. A value of 0 means the holder MAY NOT issue further delegation CTs.

max_depth: The maximum delegation depth set at root issuance. REQUIRED. MUST equal the delegation_depth of the root CT.

parent_ct: The hash of the parent CT in the delegation chain. REQUIRED for all non-root CTs. OPTIONAL for root CTs.

compliance_ref: A hash reference to the ZK compliance proofs established at transaction root. REQUIRED for regulated transaction contexts. Format: bytes32.

revocation_nonce: A nonce included in the revocation registry commitment. REQUIRED.

Example Capability Token (decoded JWT payload):

```
{
  "iss": "did:web:abac-pdp.example.org",
  "sub": "spiffe://org.example/workload/agent-a1",
  "iat": 1741017600,
  "exp": 1741021200,
  "jti": "ct-7f3a9b2c-4d1e-8f6a-b5c3-2e9d1a7b4f8e",
  "ct_type": "urn:example:capability:financial-transfer",
  "ct_scope": {
    "operations": ["transfer", "query"],
    "max_amount_usd": 50000,
    "currencies": ["USD", "EUR"],
    "jurisdictions": ["US", "EU"]
  },
  "human_anchor": "0x7f3a9b2c4d1e8f6ab5c32e9d1a7b4f8e...",
  "delegation_depth": 2,
  "max_depth": 3,
  "parent_ct": "0x1a2b3c4d5e6f7a8b9c0d1e2f3a4b5c6d...",
  "compliance_ref": "0x9a8b7c6d5e4f3a2b1c0d9e8f7a6b5c4d...",
  "revocation_nonce": "0x3c4d5e6f7a8b9c0d1e2f3a4b5c6d7e8f..."
}
```

3.3. Eight-Step Enforcement Engine

Resource servers MUST implement the following eight-step enforcement procedure when a CT is presented. The enforcement engine is format-agnostic; format-specific parsing is handled by a token adapter layer (Section 3.5).

Step 1: Token Signature Verification

For JWT/SD-JWT: Verify the JWS signature using the issuer's public key. The issuer key MUST be retrieved from the Trust Registry (Section 6), not from a URL in the token.
For Biscuit: Verify the public-key block signature using the Ed25519 public key from the Trust Registry.
For CWT: Verify the COSE_Sign1 structure using the issuer's key from the Trust Registry.

If signature verification fails, MUST reject with HTTP 401.

Step 2: Issuer Trust Verification

Retrieve the Trust Registry entry for ct.iss. Verify that the issuer is authorized to certify capability type ct.ct_type. If the issuer is not listed for this capability type, MUST reject with HTTP 403.

Step 3: Temporal Validity

```
   Verify ct.exp > now and ct.iat <= now.
   If either check fails, MUST reject with HTTP 401.

Step 4: Revocation Check

   Query the revocation registry using ct.jti and
   ct.revocation_nonce.  If the CT appears in the revocation
   registry, MUST reject with HTTP 401.

Step 5: Scope Verification

   Verify that the requested operation and parameters fall
   within ct.ct_scope.  Scope checking MUST be performed
   against the structured scope object, not a free-form string.
   If the operation is outside scope, MUST reject with HTTP 403.

Step 6: Delegation Depth Verification

   If this CT was issued by delegation (parent_ct is present),
   verify that ct.delegation_depth < parent.delegation_depth
   and ct.max_depth == parent.max_depth.
   If either check fails, MUST reject with HTTP 403.

Step 7: Human Anchor Verification

   Verify that ct.human_anchor is present and non-zero.
   Verify that ct.human_anchor matches the human_anchor claim
   in the SPT-Txn Transaction Token presented alongside the CT.
   If anchors do not match, MUST reject with HTTP 403.

Step 8: Scope Hash Binding

   Compute hash(ct.ct_scope) and compare with the
   spt_ct_scope_hash claim in the accompanying SPT-Txn Token.
   If hashes do not match, MUST reject with HTTP 403.
```


3.4. Scope Invariants

The following invariants MUST hold across the delegation chain. Implementations MUST enforce these at issuance time.

Invariant 1 (Scope Monotonicity):
For any CT_child delegated from CT_parent:
ct_child.ct_scope is a subset of ct_parent.ct_scope

No delegatee MAY hold a scope that is not a subset of its delegator's scope. Scope extension through delegation is explicitly prohibited.

Invariant 2 (Depth Monotonicity):
For any CT_child delegated from CT_parent:
ct_child.delegation_depth < ct_parent.delegation_depth
ct_child.max_depth == ct_parent.max_depth

Invariant 3 (Human Anchor Immutability):
For any CT in a delegation chain rooted at CT_root:
ct.human_anchor == ct_root.human_anchor

The human anchor MUST NOT be modified at any delegation step.

Invariant 4 (Compliance Reference Continuity):
For any CT_child delegated from CT_parent:
ct_child.compliance_ref == ct_parent.compliance_ref

Compliance proofs established at root propagate by reference and MUST NOT be replaced or omitted at delegation steps.

3.5. Token Serialization Formats

The TBAC enforcement engine (Section 3.3) operates through a format-agnostic verification interface. Format-specific parsing is isolated behind a token adapter layer. The eight enforcement steps operate identically regardless of serialization format.

3.5.1. Primary Format: JWT with Selective Disclosure (SD-JWT)

JWT [RFC7519] is the normative CT serialization for cross-organizational interoperability. All implementations MUST support JWT.

SD-JWT [I-D.ietf-oauth-selective-disclosure-jwt] extends JWT with selective disclosure of individual claims. Implementations SHOULD support SD-JWT where data minimization requirements apply.

SD-JWT selective disclosure enables a CT holder to present only the subset of claims required by the verifying resource server, without revealing the full CT payload. This is particularly relevant for the ct_scope claim, where a holder MAY disclose only the sub-scope relevant to the requested operation, while the issuer’s binding covers the full scope.

SD-JWT CTs MUST protect the following claims from selective omission (they MUST always be disclosed):
iss, sub, iat, exp, jti, human_anchor, delegation_depth, max_depth, revocation_nonce.

The ct_scope and compliance_ref claims MAY be selectively disclosed in SD-JWT presentations.

3.5.2. Alternative Format: Biscuit

Biscuit [BISCUIT] is an append-only capability token format using Ed25519 signatures and Datalog for policy expression. Biscuit provides cryptographic attenuation: a holder can create a more restricted version of a token by appending a new block signed with a fresh key, without interaction with the original issuer.

Implementations MAY support Biscuit where cryptographic attenuation without issuer interaction is required. Biscuit’s append-only structure enforces Invariant 1 (Scope Monotonicity) cryptographically at the format layer.

When using Biscuit:
- The root block MUST contain all REQUIRED CT claims as Datalog facts.
- The human_anchor and delegation_depth MUST be in the root block and MUST NOT be modifiable by attenuation blocks.
- The Trust Registry MUST list the Ed25519 public key of the root block issuer for the relevant capability type.
- Step 1 of the enforcement engine MUST verify the root block signature using the Trust Registry key, and verify all attenuation block signatures form a valid chain.

3.5.3. Alternative Format: CWT (CBOR Web Token)

CWT [RFC8392] provides the same semantic structure as JWT in CBOR encoding, with COSE [RFC9052] for signing. CWT is RECOMMENDED for high-throughput agent-to-agent transaction contexts where binary encoding reduces token size and parsing overhead.

When using CWT:
- All REQUIRED CT claims MUST be present using their CBOR integer claim keys as defined in the IANA CBOR Web Token Claims Registry, or as defined in Section 11 for SPT-Txn-specific claims.
- The COSE_Sign1 structure MUST be used for signing.
- Step 1 of the enforcement engine MUST verify using COSE signature verification procedures [RFC9052].

3.5.4. Format Selection Criteria

The following criteria SHOULD guide format selection:

JWT/SD-JWT: Cross-organizational deployments; human-readable debugging requirements; broad ecosystem library support; when GDPR data minimization is required (SD-JWT).

Biscuit: Deployments requiring cryptographic attenuation without issuer interaction; edge computing contexts; when Datalog policy expression is operationally preferred.

CWT: High-throughput agent-to-agent pipelines; IoT and constrained environments; when binary encoding provides measurable operational benefit.

Implementations MUST NOT mix formats within a single delegation chain (e.g., a JWT root CT MUST NOT have a CWT child CT). The format is set at root issuance and MUST be consistent through the chain.

4. SPT-Txn Token Structure

4.1. Base Claims (Inherited)

SPT-Txn Tokens inherit and MUST include all REQUIRED claims defined in RFC 9700 for Transaction Tokens:

iss: The authorization server that issued this token.
iat: Issued-at time.
exp: Expiry time.
txn: Transaction identifier from RFC 9700.
sub: The workload identity of the calling service at this hop.
azp: The authorized party (original requesting workload).
rctx: The request context as defined in RFC 9700.

4.2. SPT-Txn Extension Claims

SPT-Txn Tokens MUST include the following additional claims:

spt_ct_ref: bytes32. The SHA-256 hash of the Capability Token presented by the caller at this hop. Binds the transaction token to the specific capability grant used. REQUIRED.

spt_ct_scope_hash: bytes32. The SHA-256 hash of the ct_scope claim of the presented CT. Used by the enforcement engine in Step 8 for scope binding verification. REQUIRED.

spt_human_anchor: bytes32. The humanAnchor value from the presented CT. MUST equal ct.human_anchor. Propagated immutably. REQUIRED.

spt_delegation_depth: uint8. The delegation_depth value from the presented CT at this hop. REQUIRED.

spt_compliance_ref: bytes32. The compliance_ref from the root CT. MUST be identical at every hop. REQUIRED for regulated transaction contexts.

spt_parent_txn: The jti of the parent SPT-Txn Token in the chain. OPTIONAL for root tokens. REQUIRED for all non-root tokens.

Example SPT-Txn Token extension claims:

```
{
  "spt_ct_ref":
    "0x7f3a9b2c4d1e8f6ab5c32e9d1a7b4f8e9c0d1e2f...",
  "spt_ct_scope_hash":
    "0x1a2b3c4d5e6f7a8b9c0d1e2f3a4b5c6d7e8f9a0b...",
  "spt_human_anchor":
    "0x7f3a9b2c4d1e8f6ab5c32e9d1a7b4f8e9c0d1e2f...",
  "spt_delegation_depth": 2,
  "spt_compliance_ref":
    "0x9a8b7c6d5e4f3a2b1c0d9e8f7a6b5c4d3e2f1a0b...",
  "spt_parent_txn": "ct-4a5b6c7d-8e9f-0a1b-2c3d-4e5f6a7b8c9d"
}
```


5. ZK Circuit Specification

5.1. Proof System

SPT-Txn uses Groth16 [GROTH16] over the BN254 pairing-friendly elliptic curve. Proof generation is performed by the ABAC PDP at root transaction time. Proof verification is performed by resource servers and auditors.

The proof system provides:
- Completeness: Valid witnesses always produce valid proofs.
- Soundness: Invalid witnesses cannot produce valid proofs (computationally, under the BN254 discrete log assumption).
- Zero-knowledge: Proofs reveal nothing about witnesses beyond the verified public inputs.

Circuit parameters MUST be generated through a trusted setup ceremony. The SPT-Txn reference implementation uses the Hermez ceremony outputs for BN254 [HERMEZ].

5.2. Circuit 1: Attribute Proof

The Attribute Proof circuit proves that a principal holds specified attributes issued by a trusted issuer, without revealing which attributes or their values.

Public inputs:
- attr_root: Merkle root of the attribute set.
- issuer_pk: Public key of the trusted attribute issuer.
- attr_type_commitment: Commitment to the attribute type being proved.

Private inputs (witness):
- attr_value: The actual attribute value.
- attr_merkle_path: Merkle path proving inclusion.
- issuer_signature: Issuer's signature over the attribute.
- nullifier_key: Holder's nullifier key.

Outputs:
- nullifier: Hash(nullifier_key, attr_root). Prevents double-use of the same attribute proof.
- proof: Groth16 proof of the above relation.

The circuit proves:
1. attr_value is included in the Merkle tree with root attr_root (via attr_merkle_path).
2. issuer_signature is a valid signature over (attr_type, attr_value) under issuer_pk.
3. nullifier == Hash(nullifier_key, attr_root).

5.3. Circuit 2: Compliance Claim

The Compliance Claim circuit proves that a principal satisfies a compliance requirement (KYC, AML, jurisdiction) without revealing the underlying compliance data.

Public inputs:
- compliance_type: URI identifying the compliance requirement.
- compliance_issuer_pk: Public key of the compliance certifier.
- jurisdiction_set_root: Merkle root of permitted jurisdictions.
- timestamp_bound: Maximum age of the compliance attestation.

Private inputs (witness):
- compliance_credential: The underlying compliance credential.
- issuer_signature: Certifier's signature.
- jurisdiction_merkle_path: Proof of jurisdiction inclusion.
- credential_timestamp: Timestamp of the credential.

Outputs:
- compliance_commitment: Hash(compliance_credential, compliance_type). Used as compliance_ref in CTs.
- proof: Groth16 proof of the above relation.

The circuit proves:
1. compliance_credential is validly signed by compliance_issuer_pk.
2. The jurisdiction in compliance_credential is included in the set with root jurisdiction_set_root.
3. credential_timestamp > (now - timestamp_bound).
4. compliance_commitment == Hash(compliance_credential, compliance_type).

5.4. Circuit 3: Biometric Commitment

The Biometric Commitment circuit proves that a biometric

measurement matches a committed template, producing a zkDID commitment without revealing the biometric data.

Public inputs:
- template_commitment: Pedersen commitment to the biometric template.
- liveness_proof_root: Root of the liveness attestation tree.

Private inputs (witness):
- biometric_measurement: The raw biometric measurement.
- biometric_template: The enrolled template.
- template_randomness: Randomness in the commitment.
- liveness_attestation: Signed liveness attestation.
- liveness_merkle_path: Merkle path in attestation tree.

Outputs:
- zkdid_commitment: Poseidon(biometric_template, template_randomness). This is the humanAnchor.
- proof: Groth16 proof of the above relation.

The circuit proves:
1. biometric_measurement matches biometric_template within a defined distance threshold.
2. template_commitment == Pedersen(biometric_template, template_randomness).
3. liveness_attestation is valid and included in the liveness attestation tree.
4. zkdid_commitment == Poseidon(biometric_template, template_randomness).

6. Multi-Chain Trust Registry Model

6.1. Chain-Agnostic Trust Registry

The Trust Registry is an on-chain data store mapping issuer identities to the capability types they are authorized to certify. It is the root of trust for Step 2 of the enforcement engine.

Trust Registry entries MUST contain:

issuer_id: The DID or HTTPS identifier of the ABAC PDP.
capability_types: A list of capability type URIs the issuer is authorized to certify.
issuer_pubkey: The current public key of the issuer, used for CT signature verification.
valid_from: Unix timestamp from which this entry is valid.
valid_until: Unix timestamp at which this entry expires.
status: ACTIVE, SUSPENDED, or REVOKED.

Trust Registry entries MUST be signed by a Trust Anchor -- a higher-level authority that has authorized the issuer to operate for the specified capability types. The Trust Anchor public key MUST be distributed out-of-band at deployment time.

Resource servers MUST cache Trust Registry entries for performance. Cache TTL SHOULD be configured based on the valid_until field minus a safety margin. Implementations MUST treat expired Trust Registry entries as equivalent to revoked.

6.2. Chain-Specific Deployment Notes

6.2.1. Ethereum / EVM-Compatible Chains

Trust Registry SHOULD be implemented as an upgradeable proxy smart contract using the EIP-1967 transparent proxy pattern. Access control for Trust Registry updates MUST use a multi-signature scheme with a minimum threshold of 3-of-5.

Event logs (TrustRegistryEntry events) MUST be emitted for all Trust Registry updates to support auditability.

Gas optimization: Trust Registry reads are view functions and consume no gas for read-only verification.

6.2.2. XDC Network

XDC provides EVM compatibility with Delegated Proof of Stake consensus and native support for trade finance applications. Trust Registry deployment on XDC follows the same EVM contract specification as Section 6.2.1.

XDC’s 2-second block finality provides faster Trust Registry update propagation than Ethereum mainnet. Implementations SHOULD use XDC for deployments with high Trust Registry update frequency.

6.2.3. Algorand

Algorand’s AVM (Algorand Virtual Machine) supports Trust Registry implementation as an Algorand Smart Contract (ASC1). State storage uses global state for Trust Registry entries.

Algorand’s 3.9-second block finality and immediate transaction finality (no confirmation delay) make it suitable for Trust Registry deployments requiring low-latency updates.

Algorand Standard Assets (ASAs) MAY be used to represent Trust Registry authority grants in a transferable format.

6.2.4. Hedera Hashgraph

Hedera Consensus Service (HCS) provides a tamper-evident, timestamped message log suitable for Trust Registry audit trails. Trust Registry state SHOULD be maintained in Hedera Smart Contracts (compatible with Solidity/EVM), with all state-changing operations also submitted to an HCS topic for auditability.

HCS provides:
- Cryptographic proof of message ordering and timestamp.
- Gossip-about-gossip consensus with Byzantine fault tolerance.
- GDPR-compliant auditability (HCS messages are public but can reference encrypted off-chain data by hash).

The HCS audit topic MUST record:
- All Trust Registry entry additions and modifications.
- All CT issuance events (by hash reference, not content).
- All revocation events.

7. Transaction Flow

The following illustrates a complete SPT-Txn transaction flow for a regulated financial agentic system.

Step 1: Human Authentication and Root CT Issuance

Human H presents to ABAC PDP:
- ZK proof of biometric commitment (liveness attested)
- ZK compliance proofs (KYC, AML, jurisdiction)

ABAC PDP evaluates against Policy NFT and issues:

```
CT_H: {
  iss: "did:web:abac-pdp.org-a.example",
  sub: H.zkDID,
  ct_type: "urn:example:capability:financial-transfer",
  ct_scope: { full authorized scope },
  human_anchor: H.zkDID_commitment,
  delegation_depth: 3,
  max_depth: 3,
  compliance_ref: hash(ZK_compliance_proofs)
}

Txn-Token-root: {
  (RFC 9700 base claims),
  spt_ct_ref: hash(CT_H),
  spt_ct_scope_hash: hash(CT_H.ct_scope),
  spt_human_anchor: H.zkDID_commitment,
  spt_delegation_depth: 3,
  spt_compliance_ref: hash(ZK_compliance_proofs)
}
```

Step 2: Human Delegates to AI Agent A1

H requests a delegation CT for A1 from ABAC PDP:

```
CT_A1: {
  sub: A1.workload_identity,
  human_anchor: H.zkDID_commitment,  <- unchanged
  ct_scope: { reduced_scope },       <- subset of CT_H
  delegation_depth: 2,               <- decremented
  parent_ct: hash(CT_H)
}
```

H provides A1 with CT_A1 and an updated Txn-Token identifying A1 as the next hop workload.

Step 3: A1 Calls Service S1 (Organization B)

A1 requests a delegation CT for S1:

```
CT_S1: {
  sub: S1.workload_identity,
  human_anchor: H.zkDID_commitment,  <- unchanged
  ct_scope: { further_reduced_scope },
  delegation_depth: 1,
  parent_ct: hash(CT_A1)
}
```

A1 presents CT_S1 and SPT-Txn Token to S1.

Step 4: S1 Verifies CT_S1 (Offline, No Issuer Contact)

S1 runs the eight-step enforcement engine:
(a) JWS signature valid per Trust Registry key for Org A.

```
   (b) Org A's ABAC PDP authorized for this capability type.
   (c) CT_S1.exp > now.
   (d) CT_S1.jti not in revocation registry.
   (e) Requested operation within CT_S1.ct_scope.
   (f) CT_S1.delegation_depth (1) < CT_A1.delegation_depth (2).
   (g) CT_S1.human_anchor == Txn-Token.spt_human_anchor.
   (h) hash(CT_S1.ct_scope) == Txn-Token.spt_ct_scope_hash.

   All checks pass.  S1 processes the request.
   No call to Org A's authorization server required.
```

Step 5: S1 Calls S2 (Organization C)

```
   S1 requests a leaf CT for S2:

     CT_S2: {
       sub: S2.workload_identity,
       human_anchor: H.zkDID_commitment,
       ct_scope: { minimal_leaf_scope },
       delegation_depth: 0,  <- cannot delegate further
       parent_ct: hash(CT_S1)
     }
```

Audit Trail (verifiable by any authorized auditor):

```
   Txn-Token chain: H -> A1 -> S1 -> S2
   Each hop: CT reference, scope hash, depth, compliance ref.
   Any party can verify: human authorized root, scope only
   reduced at each step, compliance claims valid from root,
   chain of custody intact.
   No PII revealed to intermediate parties or auditors.
```


8. Security Considerations

8.1. Capability Token Security

CT Signature Verification: All CT signatures MUST be verified against the issuer's public key retrieved from the Trust Registry. Implementations MUST NOT accept issuer public keys embedded in the token itself or retrieved from a URL in the token. This prevents key substitution attacks.

CT Expiry: CTs MUST have a finite exp claim. Implementations MUST NOT accept CTs with exp in the past. The recommended maximum CT lifetime is 1 hour for transactional CTs. Long-lived CTs increase revocation exposure.

JTI Uniqueness: The jti claim MUST be globally unique across all CTs issued by a given issuer. Implementations MUST reject CTs with a jti that has been seen before (replay prevention).

CT Scope Precision: CT scopes MUST be as narrow as possible while permitting the required operations. Overly broad scopes reduce the security benefit of TBAC enforcement. Issuers MUST validate scope requests against the requesting principal's ABAC policy before issuing.

8.2. Delegation Chain Security

Scope Monotonicity Enforcement: Issuers MUST verify Invariant 1

(Section 3.4) before issuing delegation CTs. Implementations MUST reject delegation requests where the requested scope is not a strict subset of the delegator's CT scope.

Depth Enforcement: Issuers MUST verify that the delegator's CT has delegation_depth > 0 before issuing a delegation CT. Implementations MUST set the delegatee's delegation_depth to delegator.delegation_depth - 1.

Human Anchor Immutability: Issuers MUST copy the human_anchor from the parent CT without modification. Any CT presenting a human_anchor different from its parent MUST be rejected.

Parent CT Validity: Before issuing a delegation CT, the issuer MUST verify that the parent CT passes all eight enforcement steps (Section 3.3). An issuer MUST NOT issue delegation CTs on behalf of an expired, revoked, or out-of-scope parent CT.

8.3. ZK Proof Security

Trusted Setup: The Groth16 proof system requires a trusted setup. The toxic waste from the setup MUST be destroyed. Multi-party computation ceremonies SHOULD be used to distribute trust. The reference implementation uses the Hermez ceremony.

Proof Replay: ZK proofs are publicly verifiable. The nullifier mechanism in Circuit 1 MUST be used to prevent proof replay. Implementations MUST maintain a nullifier registry and reject proofs with previously seen nullifiers.

Circuit Audit: ZK circuits MUST be independently audited before production deployment. The circuit specification in Section 5 is normative; implementations MUST match the specified input/output structure exactly.

8.4. Trust Registry Security

Registry Immutability: Trust Registry updates MUST require multi-signature authorization. Single-key control of the Trust Registry is a critical security risk.

Registry Monitoring: Implementations MUST monitor Trust Registry events for unexpected entry additions or modifications. An unexpected Trust Registry update MAY indicate a key compromise at the Trust Anchor level.

Entry Expiry: Trust Registry entries MUST have a valid_until field. Perpetual Trust Registry entries create long-term exposure if the associated issuer is later compromised.

9. Privacy Considerations

Human Anchor Privacy: The humanAnchor is a zero-knowledge commitment to the authorizing human's zkDID. It does not reveal the human's identity, biometric data, or any other PII to intermediate services. Only the licensed issuer holding the escrow key can deanonymize the humanAnchor under lawful process.

Compliance Reference Privacy: The compliance_ref is a hash

reference to ZK compliance proofs established at root. The proofs themselves are not propagated in the transaction chain. Resource servers verify that compliance_ref matches a known proof commitment; they do not receive or store the underlying compliance data.

Selective Disclosure: When SD-JWT format (Section 3.5.1) is used, CT holders MAY limit claim disclosure to the specific claims required by the verifying resource server. This reduces the PII surface per hop.

Auditability vs. Privacy: The SPT-Txn audit chain (Section 7) is designed to be auditable by authorized auditors without revealing PII. The Txn-Token chain carries human_anchor (ZK commitment), ct_ref (token hash), and compliance_ref (proof hash) -- none of which reveal PII directly. Identity deanonymization requires access to the issuer's zkDID escrow, which MUST be subject to lawful process requirements.

GDPR Compliance: The humanAnchor mechanism is designed to satisfy GDPR Article 25 (Data Protection by Design) and Article 5(1)(c) (Data Minimization). Intermediate services receive only cryptographic commitments, not personal data. Implementations operating in EU jurisdictions SHOULD obtain legal review of their specific humanAnchor escrow arrangement.

10. Security Properties

10.1. Signature Algorithm Requirements

All CT signatures MUST use one of the following algorithms:

ES256: ECDSA with P-256 and SHA-256. REQUIRED to support.
ES384: ECDSA with P-384 and SHA-384. RECOMMENDED.
EdDSA (Ed25519): Edwards-curve DSA. RECOMMENDED.
Ed25519 is RECOMMENDED for new deployments due to its constant-time implementation properties and smaller signature size.

RS256 and weaker algorithms MUST NOT be used for CT signing.

Post-quantum note: ECDSA and EdDSA are vulnerable to quantum adversaries with access to Shor's algorithm. Deployments with long-term security requirements (>10 years) SHOULD monitor NIST post-quantum standardization and plan migration to PQ-safe signature schemes. The SPT-Txn claim structure is algorithm-agnostic and does not require changes to accommodate algorithm migration.

10.2. Key Management

ABAC PDP signing keys MUST be stored in hardware security modules (HSMs) or equivalent tamper-resistant hardware.

Key rotation MUST be supported. Trust Registry entries MUST be updated when issuer keys are rotated. CTs signed with rotated-out keys MUST be considered expired at rotation time unless a key overlap period is explicitly configured.

Key overlap period SHOULD NOT exceed the maximum CT lifetime

to bound exposure of the outgoing key.

10.3. Transport Security

All SPT-Txn Token and CT transmission MUST occur over TLS 1.3 [RFC8446] or later. TLS 1.2 MAY be permitted for legacy compatibility with explicit operator acknowledgment.

CT transmission in HTTP headers MUST use the Authorization header with a custom scheme:

```
Authorization: SPT-CT <base64url-encoded-CT>
```

10.4. Replay Prevention

CTs include a jti claim for unique identification. Resource servers MUST maintain a short-term jti cache to detect CT replay within the CT's validity window.

SPT-Txn Tokens inherit the replay prevention mechanisms defined in RFC 9700, including the txn claim binding.

10.5. Token Format Security Properties

This section specifies security properties specific to each token serialization format (Section 3.5).

JWT/SD-JWT Security:

> JWS algorithm confusion attacks MUST be prevented by explicitly specifying the expected algorithm at verification time. Implementations MUST NOT accept the "none" algorithm.
>
> SD-JWT holder binding MUST use Ed25519 or P-256 key binding. Unbound SD-JWT presentations MUST NOT be accepted for CT presentation.

Biscuit Security:

> Biscuit's append-only structure provides structural defence against scope escalation: attenuation blocks can only restrict, never expand, the token's authority. This cryptographically enforces Invariant 1 (Scope Monotonicity) at the format layer, independent of the enforcement engine.
>
> The root block's Ed25519 key MUST be registered in the Trust Registry. Attenuation block keys are ephemeral and MUST NOT be registered.
>
> Biscuit tokens MUST be treated as opaque by intermediate services that are not the intended verifier.

CWT/COSE Security:

> COSE_Sign1 MUST be used (not COSE_Sign for multi-signer). Algorithm parameters MUST be in the protected header. Unprotected header algorithm parameters MUST be ignored.
>
> The COSE kid (key ID) parameter MUST NOT be used to select verification keys. Verification keys MUST be retrieved from the Trust Registry based on the iss claim.

10.6. ZK Proof System Security

See Section 8.3 for ZK proof security considerations.

Additionally:

- Groth16 proofs are computationally sound under the BN254 discrete logarithm assumption.
- Proof size is constant (192 bytes for Groth16 on BN254) regardless of circuit complexity.
- Verification time is constant and fast (< 5ms on commodity hardware), making ZK proof verification practical in the enforcement engine hot path.

10.7. Multi-Chain Trust Registry Integrity

See Section 8.4 for Trust Registry security considerations.

Cross-chain deployments: When Trust Registries are deployed on multiple chains, implementations MUST ensure consistency between chain deployments. A Trust Registry entry on one chain MUST NOT be considered authoritative for resource servers configured to use a different chain.

Chain reorganizations: For EVM chains, Trust Registry reads SHOULD wait for a minimum of 12 block confirmations to protect against reorganization-based attacks on Trust Registry state. Hedera and Algorand provide immediate finality and do not require confirmation delays.

11. IANA Considerations

This document requests registration of the following JWT claims in the IANA JSON Web Token Claims Registry [IANA.JWT.Claims].

Claim Name: spt_ct_ref
Claim Description: SHA-256 hash of the Capability Token presented at this transaction hop.
Change Controller: IETF
Specification Document(s): Section 4.2 of this document.

Claim Name: spt_ct_scope_hash
Claim Description: SHA-256 hash of the ct_scope claim of the presented Capability Token.
Change Controller: IETF
Specification Document(s): Section 4.2 of this document.

Claim Name: spt_human_anchor
Claim Description: Zero-knowledge commitment to the zkDID of the authorizing human principal (humanAnchor).
Change Controller: IETF
Specification Document(s): Sections 3.2 and 4.2 of this document.

Claim Name: spt_delegation_depth
Claim Description: Remaining delegation depth from the Capability Token at this transaction hop.
Change Controller: IETF
Specification Document(s): Section 4.2 of this document.

Claim Name: spt_compliance_ref
Claim Description: Hash reference to the ZK compliance proofs established at transaction root.
Change Controller: IETF
Specification Document(s): Sections 3.2 and 4.2 of this document.

Claim Name: spt_parent_txn
Claim Description: JWT ID of the parent SPT-Txn Token in the transaction chain.
Change Controller: IETF
Specification Document(s): Section 4.2 of this document.

This document also requests registration of the following Capability Token claims (in a new SPT-Txn CT Claims Registry to be established by IANA, or in the JWT Claims Registry):

Claim Name: ct_type
Claim Description: URI identifying the capability type granted by this Capability Token.
Change Controller: IETF
Specification Document(s): Section 3.2 of this document.

Claim Name: ct_scope
Claim Description: Structured object defining the capability scope, operations, and constraints granted.
Change Controller: IETF
Specification Document(s): Section 3.2 of this document.

Claim Name: human_anchor
Claim Description: Zero-knowledge commitment to the zkDID of the authorizing human (humanAnchor), carried in the CT.
Change Controller: IETF
Specification Document(s): Section 3.2 of this document.

Claim Name: delegation_depth
Claim Description: Remaining delegation levels permitted from this CT holder.
Change Controller: IETF
Specification Document(s): Section 3.2 of this document.

Claim Name: compliance_ref
Claim Description: Hash reference to ZK compliance proofs at transaction root.
Change Controller: IETF
Specification Document(s): Section 3.2 of this document.

Claim Name: revocation_nonce
Claim Description: Nonce included in the revocation registry commitment for this CT.
Change Controller: IETF
Specification Document(s): Section 3.2 of this document.

12. References

12.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <https://www.rfc-editor.org/info/rfc7515>.

[RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <https://www.rfc-editor.org/info/rfc7519>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8392] Jones, M., Wahlstroem, E., Erdtman, S., and H. Tschofenig, "CBOR Web Token (CWT)", RFC 8392, DOI 10.17487/RFC8392, March 2018, <https://www.rfc-editor.org/info/rfc8392>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC9052] Schaad, J., "CBOR Object Signing and Encryption (COSE): Structures and Process", STD 96, RFC 9052, DOI 10.17487/RFC9052, August 2022, <https://www.rfc-editor.org/info/rfc9052>.

[RFC9449] Fett, D., Campbell, B., Bradley, J., Lodderstedt, T., Jones, M., and D. Waite, "OAuth 2.0 Demonstrating Proof of Possession (DPoP)", RFC 9449, DOI 10.17487/RFC9449, September 2023, <https://www.rfc-editor.org/info/rfc9449>.

[RFC9700] Tulshibagwale, A., Fletcher, G., Kasselman, P., Burgin, K., and D. Richer, "Transaction Tokens", RFC 9700, 2025, <https://www.rfc-editor.org/info/rfc9700>.

[I-D.ietf-oauth-selective-disclosure-jwt] Fett, D., Yasuda, K., and B. Campbell, "Selective Disclosure for JWTs (SD-JWT)", Work in Progress, Internet-Draft, draft-ietf-oauth-selective-disclosure-jwt-13, 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-oauth-selective-disclosure-jwt>.

12.2. Informative References

[BISCUIT] Felte, C., et al., "Biscuit: A Capability-Based Authorization Token", 2022, <https://github.com/biscuit-auth/biscuit>.

[GROTH16] Groth, J., "On the Size of Pairing-Based Non-interactive Arguments", Advances in Cryptology EUROCRYPT 2016, LNCS 9666, pp. 305-326, 2016, DOI 10.1007/978-3-662-49896-5_11.

[HERMEZ] Hermez Network, "Trusted Setup Ceremony", <https://ceremony.hermez.io/>.

[RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749, October 2012, <https://www.rfc-editor.org/info/rfc6749>.

[SPT-THEORY] Coetzee, R.J., "Transaction Binding Security for Policy-Bound Authorization Tokens: Formal Definitions and Proofs for the SPT-Txn Framework", Zenodo, DOI 10.5281/zenodo.18917439, 2025, <https://doi.org/10.5281/zenodo.18917439>.

[SPT-WP] Coetzee, R.J., "The SPT-Txn Framework v6.0: Sovereign Policy Token Transactions", Zenodo, 2025, <https://doi.org/10.5281/zenodo.18917439>.

[IANA.JWT.Claims] IANA, "JSON Web Token Claims", <https://www.iana.org/assignments/jwt>.

Author's Address

Rudolf Jacobus Coetzee
Violet Sky Security SEZC
Cayman Islands

Email: rudi@violetskysecurity.com
URI: https://violetskysecurity.com