Network Working Group
Internet-Draft
Intended status: Informational
Expires: 29 November 2026

F. Serafini, Ed.
Vauban Research

28 May 2026

x402 Cryptographic Receipts: Format, Post-Quantum Discipline, and Starknet Anchor

draft-vauban-x402-consolidated-00

Abstract

The x402 V2 protocol defines HTTP-native payment flows but leaves three gaps that block compliance use cases. First, the PAYMENT-RESPONSE carries a facilitator-issued reference rather than a self-contained, offline-verifiable cryptographic receipt; an auditor cannot validate a retained receipt without contacting the facilitator. Second, classical signatures alone (ES256K) do not satisfy the post-quantum migration horizon set by NIST SP 800-208, ANSSI, BSI, and EU eIDAS 2.0 for high-value or long-retention material. Third, off-chain receipts alone do not provide ledger-anchored finality required by frameworks such as MiCA Art. 76 (settlement record-keeping) and EU AI Act Art. 12 (transparency-and-documentation).

This document consolidates three previously separate extensions into a single specification. It defines (a) a negotiable receipt-format extension with three variants (Stwo Circle STARK proof, hybrid ES256K + ML-DSA-65 dual-signature, classical ES256K fallback) over a JCS canonical preimage discipline grounded in RFC 8785; (b) a two-axis post-quantum discipline mapping hash-based proving and hybrid signatures to the NIST PQC migration roadmap; and (c) a Starknet on-chain anchor format with a canonical anchor tuple, Cairo event layout, RPC endpoint convention, and block explorer reference for human-readable audit.

Topics under active coalition discussion are explicitly out of scope: the VPSF composability claim algebra, the payment lifecycle finite state machine, and the delegation binding extension are deferred to companion documents not included in this consolidated submission.

Status of This Memo

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

This Internet-Draft will expire on 29 November 2026.

Copyright Notice

Table of Contents

1. Introduction

1.1. Problem Statement

The x402 protocol [X402-V2] defines HTTP-native payment flows using three messages: PAYMENT-REQUIRED (402 response), PAYMENT-SIGNATURE (client request), and PAYMENT-RESPONSE (facilitator confirmation). The PAYMENT-RESPONSE carries a payment_hash and a facilitator-issued settlement reference. Three gaps prevent the base protocol from satisfying audit-grade and post-quantum compliance use cases:

* Off-line verifiability gap: a verifier must contact the facilitator to confirm that the conditions of a specific payment were met. EU AI Act Art. 12 (transparency-and-documentation) and MiCA Art. 76 (settlement record-keeping) require evidence that is self-contained at audit time.

* Post-quantum migration gap: ES256K signatures rely on the discrete-logarithm hardness assumption and become forgeable under Shor's algorithm once a sufficient quantum capability becomes available. NIST [NIST-PQC-MIGRATION], ANSSI [ANSSI-PQC], BSI [BSI-PQC], and EU eIDAS 2.0 [EIDAS-2] guidance converge on a 2030-2035 horizon for high-value or long-retention material.

* On-chain finality gap: an off-chain receipt does not confirm that a settlement was committed to a public ledger. A regulator examining a retained receipt cannot independently verify that the settlement reached canonical confirmation at a specific block height.

1.2. Scope

This document folds three previously separate Internet-Drafts into one specification, addressing the three gaps in a single audit surface:

* A negotiable receipt-format extension with three variants over a JCS canonical preimage discipline (Section 3).

* A two-axis post-quantum discipline (proof-system layer and signature layer) aligned with the NIST PQC migration roadmap (Section 4).

* A Starknet on-chain anchor format with a canonical anchor tuple, Cairo event layout, RPC endpoint convention, and block explorer reference (Section 5).

1.3. Out of Scope

The following work, under active coalition discussion at the time of writing, is deferred to companion documents and is not included in this consolidated submission:

* Composability grammar for payment claims, including the algebra of operators for combining cryptographic evidence across payment lifecycle states. See [VPSF-ALGEBRA].

* Payment lifecycle finite state machine, including state-transition validity rules and chain-reconstruction algorithm. See [LIFECYCLE-FSM].

* Delegation binding extension, including cryptographic scope of delegation receipts to granting principals. See [DELEGATION-BINDING].

These topics are referenced in the present document only where necessary for cross-context clarity; this document defines no normative content for them.

1.4. Status of This Document

This document is an Independent Submission. It is not the product of an IETF Working Group. It is published for community review and to establish a stable reference for implementors. The three folded extensions have been the subject of public coalition review at [X402-2326], [X402-2357], [X402-2398], [X402-2411], [X402-2412], [X402-2413], [X402-2434], [X402-2440], and [X402-2459].

2. Conventions and Definitions

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.

Receipt format: A string token identifying which cryptographic receipt variant a facilitator produces. See Section 3.1.

JCS: JSON Canonicalization Scheme, defined in [RFC8785]. Produces a deterministic byte representation of a JSON value by applying recursive key sorting and value normalisation.

action_ref: A 32-byte opaque digest binding a payment receipt to a

work-layer event, derived by SHA-256 over the UTF-8 JCS encoding of a canonical preimage object. See Section 3.11.

payment_hash: A hash of the payment conditions committed to by the payer, as defined in the x402 V2 base specification [X402-V2].

NFC: Unicode Normalization Form C, as defined in [UAX15].

Hash-based proof system: A succinct proof system whose soundness reduces to the collision and second-preimage resistance of an underlying cryptographic hash function rather than to algebraic assumptions (pairings, discrete logarithms, factorisation). STARK proof systems are hash-based. See Section 4.2.

Hybrid signature: A composite signature scheme that produces two or more independent signatures over the same canonical message under disjoint cryptographic assumptions. A hybrid signature is valid only if every component signature verifies independently. See Section 4.3.

Anchor: An on-chain record that commits a payment settlement to a public ledger, identified by a canonical anchor tuple (Section 5.1) and verifiable by any party with read access to the chain.

Conforming emitter: A smart contract that satisfies the settlement event layout defined in Section 5.3 and emits that event for each settled payment.

3. Receipt Format and Canonical Preimage

This section defines the negotiable receipt_format extension, its three variants, the HTTP negotiation surface, and the JCS canonical preimage discipline that binds a receipt to a work-layer event. It harmonises material from the three folded source drafts into a single normative surface.

3.1. Receipt Format Enum

The receipt_format field identifies which cryptographic receipt variant a facilitator has produced. The value is a string token from the registry described in Section 7:

Token	Description	Approx. size
stark-vauban-pay-v1	Stwo Circle STARK over payment-condition witness (amount, currency, payer attestation, nullifier). Post-quantum sound at the proof layer. Offline verifiable.	~100 KB
hybrid-pqc	ES256K + ML-DSA-65 ([FIPS204]) dual-signature over JCS-canonical receipt core. Receipt survives Q-Day if either single algorithm is broken.	~3.3 KB
classical-es256k	ES256K signature only. Default fallback for clients that do not require post-quantum or zero-knowledge properties.	~0.5 KB

Table 1

Each variant corresponds to evidenceType: "cryptographic" in the [X402-2322] compliance taxonomy. The signature_algorithm discriminator used in the bazaar evidenceShape maps as follows:

receipt_format token	signature_algorithm
stark-vauban-pay-v1	"stark-m31-stwo"
hybrid-pqc	"es256k+ml-dsa-65"
classical-es256k	"es256k"

Table 2

Facilitators MAY introduce additional tokens by registering them in the x402 Receipt Format registry. Tokens not present in the registry MUST be treated as "classical-es256k" by verifiers that do not recognise them.

The three variants correspond to three orthogonal properties; a deployment selects the property it requires:

* proof-of-payment-conditions: the stark-vauban-pay-v1 receipt proves amount, currency, payer attestation, and nullifier without revealing them. The STARK proof is the deliverable.

* receipt-integrity-under-quantum: the hybrid-pqc receipt carries two independent signatures; the receipt remains verifiable if one signature algorithm is broken.

* work-receipt-binding: all three variants carry an optional action_ref field (32-byte opaque digest per Section 3.11). Binding to the work layer is a property of the preimage derivation, not of the receipt format itself.

3.2. HTTP Header Conventions

A resource server or facilitator SHOULD include X-Payment-Options in the 402 Payment Required response to advertise which receipt_format variants it can produce:

```
X-Payment-Options: receipt_format="stark-vauban-pay-v1, hybrid-pqc, classical-es256k"
```

The value is a comma-separated ordered list of receipt_format tokens from most-preferred to least-preferred. The server declares its capabilities; the client selects from this list. If X-Payment-Options is absent, the client MUST assume the facilitator can produce only classical-es256k.

The facilitator MUST include X-Receipt-Format in the PAYMENT-RESPONSE to state which variant it emitted:

```
X-Receipt-Format: stark-vauban-pay-v1
```

The verifier uses this header to select the correct receipt parser before attempting to deserialise or verify the receipt body. If X-Receipt-Format is absent, the verifier MUST assume classical-es256k.

A facilitator that lists stark-vauban-pay-v1 in X-Payment-Options MUST also be capable of producing classical-es256k as a fallback. Listing a variant without fallback capability is a protocol violation.

3.3. Negotiation Semantics

Normal negotiation path:

1. Client receives a 402 response with X-Payment-Options.

2. Client selects the highest-preference variant it can verify from the list.

3. Client SHOULD signal its selection by including receipt_format in the PAYMENT-SIGNATURE request payload.

4. Facilitator produces the receipt in the selected variant and sets X-Receipt-Format on the PAYMENT-RESPONSE.

If the client cannot verify any variant in X-Payment-Options, or if X-Payment-Options is absent, the client MUST fall back to classical-es256k. The facilitator MUST honour this fallback.

If a client REQUIRES a specific receipt format (for example, a regulator requiring an offline-verifiable STARK receipt for EU AI Act Art. 12 purposes), it MUST include receipt_format in the PAYMENT-SIGNATURE request extensions with required: true. If the facilitator cannot produce the requested variant, the facilitator MUST return HTTP 402 with error UnsupportedReceiptFormat rather than silently downgrading. The downgrade-attack mitigation in Section 6.5 requires this discipline.

3.4. Canonical Preimage Discipline

The canonicalisation rules used by this extension are grounded in [RFC8785] (JCS), as exercised and cross-validated through the x402 canonicalisation conformance set [X402-CANON]. The digest construction is:

JCS_hash = SHA-256(UTF-8(JCS(object)))

The wire-format version marker jcs-rfc8785-v1 identifies this rule set in the canon_version field. Four normative rules apply:

1. Timestamps are encoded as integers (timestamp_ms, milliseconds since Unix epoch). The field name timestamp carrying an RFC 3339 string MUST NOT be used.

2. Field names are compared as byte strings, sorted in [RFC8785] lexicographic order.

3. Arrays preserve their order; reordering produces a different digest.

4. Type validation is performed before canonicalisation; coercion is forbidden.

3.5. Preimage Schema and Worked Digest

The canonical action_ref preimage object contains the following fields, sorted lexicographically per [RFC8785] Section 3.2.3:

```
{
  "action_type":  "<string>",
  "agent_id":     "<string>",
  "scope":        "<string>",
  "timestamp_ms": <integer>
}
```

The JCS output for the shared coalition preimage is:

```
{"action_type":"sanctions_screen","agent_id":"did:web:agent-7.example.com","scope":"count
erparty-due-diligence","timestamp_ms":1747728000000}
```

The action_ref digest is:

```
action_ref = SHA-256(UTF-8(JCS(preimage)))
           = 10d8a38c01d8672176aa6e5209a368fde3e1831640d69e15283142b35880c2c1
```

This worked example is byte-identical across the eight-language conformance matrix described in Section 3.10.

3.6. Type Validation Requirements

Implementations MUST reject preimage objects containing:

* Float values for timestamp_ms: 1747728000000.0 MUST be rejected; only integer JSON numbers are valid. Conformance vector 0002-ms-precision-trap demonstrates the failure mode where a float representation produces no parsing error but an incorrect digest. Rejection MUST occur before canonicalisation.

* Missing canonical fields: if any of action_type, agent_id, scope, or timestamp_ms is absent, the preimage MUST be rejected before canonicalisation. Vector 0012-missing-canonical-field covers this case.

* Duplicate keys: if the JSON object contains duplicate key names, the preimage MUST be rejected at parse time. Accepting last-wins or first-wins semantics creates interoperability ambiguity. Vector 0010-duplicate-keys covers this case.

3.7. Unicode Normalisation Profile

[X402-CANON] deliberately applies no Unicode normalisation in the canonicaliser, so that NFC and NFD are distinct canonical inputs. This extension imposes a stricter profile on the action_ref preimage: it requires NFC and rejects non-NFC input. This is a profile narrowing, not a contradiction of [X402-CANON].

Implementations MUST normalise all string values in the preimage to Unicode Normalization Form C (NFC) per [UAX15] BEFORE JCS canonicalisation. Implementations MUST reject input where any string value in the preimage is encoded in NFD, NFKC, or NFKD form. Acceptance of non-NFC input is a conformance violation. Verifiers MUST NOT perform implicit re-normalisation; the input MUST already be NFC for the digest to be reproducible across runtimes.

Rationale: a verifier receiving an NFD-encoded preimage would compute a different digest than one receiving the NFC equivalent, silently breaking receipt verification without a parsing error. macOS HFS+ historically stored filenames in NFD decomposed form, and database collations vary across deployments; this is not a hypothetical edge case.

Vector 0011-unicode-normalisation demonstrates the NFD divergence failure mode.

3.8. Field-Level CBOR Layout for stark-vauban-pay-v1

The receipt body for the stark-vauban-pay-v1 variant is CBOR-encoded per [RFC8949] canonical CBOR (Section 4.2.1). The top-level CBOR map contains the following fields:

Key	Major type	Description
canon_version	text string	Canonicalisation rule identifier; "jcs-rfc8785-v1" for this version.
proof	byte string	The serialised Stwo Circle STARK proof bytes.
public_inputs	byte string	The serialised public inputs to the proof verifier.
payment_hash	byte string	SHA-256 of the canonical payment conditions, 32 bytes.
action_ref	byte string	OPTIONAL. 32-byte opaque digest per Section 3.11. Omitted (not zero-valued) when no work-receipt binding is asserted.
nullifier	byte string	32-byte nullifier preventing double-spend.
circuit_id	text string	Identifier for the circuit version; "vauban-pay-v1" for this version.

Table 3

The proof attests to a circuit that verifies the following predicates:

* Payment amount matches the signed PaymentRequirements.amount.

* Currency asset address matches PaymentRequirements.asset.

* Payer attestation is valid (subject to merchant attestation-gate configuration).

* Nullifier is unique (no replay for this nullifier in the settlement ledger).

The serialised receipt is base64url-encoded (no padding) per [RFC4648] Section 5 when transmitted in the PAYMENT-RESPONSE extension envelope. The canonical CBOR bytes layout is fixed by the conformance vectors at [VAUBAN-CONFORMANCE]; implementations MUST be byte-for-byte compatible with the published vectors.

When a stark-vauban-pay-v1 receipt is anchored on Starknet, the felt252 masking operation described in Section 5.4 MUST be applied before writing or comparing the action_ref value in keys[1] of the on-chain event.

3.9. Variant Bodies for hybrid-pqc and classical-es256k

The hybrid-pqc receipt body is a JSON object [RFC8259] with the following top-level fields, JCS-canonical [RFC8785]:

* receipt_core: JSON object; the canonical payload signed by both algorithms.

* signature: ES256K signature over SHA-256(UTF-8(JCS(receipt_core))), base64url-encoded.

* pqc_signature: ML-DSA-65 ([FIPS204]) signature over the identical canonical bytes, base64url-encoded.

* kid_es256k: Key identifier for the ES256K key in the facilitator’s JWKS endpoint.

* kid_mldsa65: Key identifier for the ML-DSA-65 key in the facilitator’s JWKS endpoint.

Both signatures MUST cover the byte-identical canonical byte string. Verifiers MUST confirm both signatures independently. A verifier that can only verify ES256K SHOULD still check pqc_signature length and structure for plausibility.

The classical-es256k receipt body is a JWS Compact Serialization string [RFC7515]. The JWS payload contains the canonical payment_hash and optionally action_ref. This is the default fallback for clients that do not require zero-knowledge or post-quantum properties.

3.10. Cross-Implementation Conformance Matrix

The JCS canonical preimage discipline of this extension is exercised through eight independent runners across eight languages. Five are validated byte-for-byte against published upstream JCS RFC 8785 libraries at the time of publication; three are pure-stdlib reference runners published in the conformance vectors repository with CI execution pending.

Language	Library or runner	Validation status
Python	rfc8785@0.1.4 (Trail of Bits)	validated 11/11 byte-for-byte
JavaScript	canonicalize@3.0.0 (Erdtman + Rundgren)	validated 11/11 byte-for-byte
Go	gowebpki/jcs v1.0.1	validated 11/11 byte-for-byte
Java	cyberphone/json-canonicalization (RFC 8785 reference)	validated 11/11 byte-for-byte
Rust	serde_jcs 0.2.0 (l1h3r) via vauban-x402-jcs-conformance	validated 11/11 byte-for-byte
PHP	runners/runner.php (pure stdlib, PHP 8.0+)	reference, CI-pending
Ruby	runners/runner.rb (pure stdlib, Ruby 3.0+)	reference, CI-pending
C#	runners/runner.cs (pure stdlib, .NET 8)	reference, CI-pending

Table 4

Aggregate cross-validation: 15/15 byte-for-byte agreements across the three ALLOW/DENY/REFER fixture vectors published at [X402-2434], established against the first five runners.

The conformance vector suite is published at https://github.com/vauban-org/x402-stark-receipts-conformance under Apache 2.0. The Vauban Research-maintained crates vauban-x402-jcs-conformance@0.1.0, vauban-x402-canonical@0.1.0, and vauban-x402-wire@0.1.0 (crates.io);

the Python package vauban-x402-stark-receipt@0.1.0 (PyPI); and the npm package @vauban-pay/substrate@0.1.0 are published as Apache 2.0 reference implementations.

Implementers SHOULD NOT introduce a new runner without first executing all eleven published vectors byte-for-byte against an existing validated runner.

3.11. Wire-Level Binding (action_ref)

The action_ref field is a 32-byte opaque digest that binds a payment receipt to a work-layer event. It is the seam between the payment layer (this document) and the work layer (the application that consumes the payment). The digest is derived by SHA-256 over the UTF-8 JCS encoding of the canonical preimage object defined in Section 3.5.

The encoding of action_ref on the wire depends on the carrier:

* In JSON payloads (hybrid-pqc, classical-es256k), action_ref is base64url-encoded with no padding per [RFC4648] Section 5.

* In CBOR payloads (stark-vauban-pay-v1), action_ref is a 32-byte byte string (Section 3.8).

* On the Starknet anchor, action_ref is felt252-masked per Section 5.4 before being written to keys[1] of the settlement event.

The action_ref field is OPTIONAL in all three variants. When omitted, the receipt asserts payment without asserting a work-layer binding. A receipt MUST NOT carry a zero-valued action_ref to signal absence; absence is signalled by omission.

Where a deployment also participates in the composite trust-query layer [X402-2440], the same action_ref digest doubles as the trust-query anchor. The trust-query layer is referenced here only for cross-context clarity; this document defines no normative content for it.

3.12. Compliance Receipt and RiskCheckResult Layering

Two distinct layers exist around a payment receipt and MUST NOT be conflated:

* The internal compliance_receipt carries the facilitator's compliance decision over the payment (ALLOW, DENY, REFER). It is an internal artefact of the facilitator's risk pipeline. Its fixture form is defined at [X402-2434].

* The consumer-facing RiskCheckResult is the wire-format signal returned to the client. It conveys the outcome necessary for the client to proceed, without exposing the facilitator's internal compliance reasoning.

A facilitator MUST NOT place internal compliance_receipt material in the consumer-facing RiskCheckResult surface. A verifier MUST NOT treat the presence of a RiskCheckResult as equivalent to a cryptographic payment receipt; the two answer different questions (was the payment permitted under policy, versus did the payment settle cryptographically).

3.13. Error Taxonomy

A facilitator or verifier rejecting a receipt or a PAYMENT-SIGNATURE SHOULD signal the reason via the X-Receipt-Reject-Reason response header. Seven tokens are defined, each mapped to an HTTP status code [RFC9110]:

Token	HTTP status	Meaning
MalformedClaim	400	The receipt or preimage is syntactically invalid.
PaymentRequired	402	No valid payment is associated with the request.
UnsupportedReceiptFormat	402	A required: true receipt format cannot be produced (Section 3.3).
NullifierReplay	409	The nullifier is already present in the settlement ledger (Section 6.2).
Expired	410	The receipt timestamp_ms exceeds the configured age threshold.
StructuralInvalid	422	The receipt parses but fails schema or canonical-preimage validation.
HumanityRequired	403	The deployment requires a humanity attestation the request did not satisfy.

Table 5

A facilitator MUST NOT return a generic 400 without an X-Receipt-Reject-Reason token when one of the defined tokens applies. The token set is closed for this version; new tokens require registration alongside the receipt-format registry.

3.14. Extension Schemas in x402 Messages

The receipt_format extension appears in each of the three x402 messages. The following fields are defined for the extension object:

* In PAYMENT-REQUIRED (the 402 response): supported (ordered array of tokens the facilitator can produce) and default (the token used when the client expresses no preference).

* In PAYMENT-SIGNATURE (the client request): receipt_format (the token the client selects) and required (boolean; when true, the facilitator MUST either honour the token or reject with UnsupportedReceiptFormat).

* In PAYMENT-RESPONSE (the facilitator confirmation): receipt_format (the token emitted), receipt (the serialised receipt body in the variant encoding of Section 3.8 or Section 3.9), and action_ref (OPTIONAL, the wire-encoded binding digest per Section 3.11).

The X-Payment-Options and X-Receipt-Format headers (Section 3.2) carry the same information at the HTTP layer for clients that negotiate before parsing the message body. Where both the header and the body extension are present, they MUST agree; a verifier encountering disagreement MUST reject the message with StructuralInvalid.

4. Post-Quantum Cryptographic Discipline

This section defines a two-axis post-quantum discipline. The proof-system axis (Section 4.2) governs the soundness assumption of any proof carried by a receipt; the signature axis (Section 4.3) governs the integrity of any signature over a receipt. The two axes are independent: a deployment may satisfy one without the other, and the migration profile (Section 4.4) states which combination is required for which risk class.

4.1. Threat Model: Quantum Adversary

This document assumes an adversary capable of executing Shor’s algorithm against classical public-key primitives and Grover’s algorithm against unstructured search. The adversary holds the harvest-now-decrypt-later capability: every receipt published today is recordable and can be attacked later, once a sufficient quantum capability becomes available. A receipt emitted under a statutory retention obligation may therefore have to survive an adversary that did not exist at issuance time.

Three forgery vectors are relevant:

* ES256K signature break under Shor’s algorithm. The security of ES256K reduces to the hardness of the elliptic-curve discrete logarithm, which Shor’s algorithm solves in polynomial time. A classical-only receipt becomes forgeable once this capability exists.

* Hash collision under Grover's algorithm. Grover provides only a quadratic speedup against unstructured search; SHA-256 retains an effective 128-bit pre-image and collision resistance against a quantum adversary, which remains adequate for the digest constructions in this document.

* Proof-system soundness break. A proof system whose soundness reduces to an algebraic assumption (pairings, discrete logarithms, factorisation) may lose soundness under a quantum adversary independently of the signature layer.

4.2. Proof System Layer Axis

A receipt variant that claims post-quantum soundness at the proof layer MUST use a proof system whose soundness reduces to the collision and second-preimage resistance of a cryptographic hash function rather than to an algebraic assumption. STARK proof systems satisfy this requirement; the stark-vauban-pay-v1 variant uses such a system.

Implementations MUST NOT present a receipt as post-quantum sound at the proof layer when the proof is produced by Groth16, by PLONK, or by any pairing-based SNARK construction over BN254, BLS12-381, or any other pairing-friendly curve. These constructions rely on assumptions that do not survive the threat model of Section 4.1, and labelling such a proof post-quantum sound is a conformance violation.

The hash-based property is preserved across faithful adapter implementations of a reference prover. It breaks under naive substitution of an algebraic proof system; see Section 6.9 for the auditability considerations.

4.3. Signature Layer Axis

A receipt variant that claims post-quantum soundness at the signature layer MUST carry two independent signatures over the identical JCS canonical preimage: one classical (ES256K) and one post-quantum (ML-DSA-65 per [FIPS204]). This is the hybrid-pqc construction.

A verifier MUST verify both component signatures independently and MUST reject the receipt if either component fails. Both signatures MUST cover the byte-identical canonical byte string produced by the discipline of Section 3.4. A construction in which the two component schemes sign different canonical messages does not provide the intended security property and MUST NOT be presented as a hybrid signature; see Section 6.6.

The hybrid construction provides defence in depth: the receipt remains unforgeable as long as at least one component scheme remains unbroken. It is not a transitional measure to be discarded at the moment ML-DSA-65 is universally deployed; retaining the classical component guards against an undiscovered weakness in the lattice assumption, and retaining the post-quantum component guards against Shor.

4.4. Migration Window Profile

This document profiles the 2025-2030 migration window. Within this window both verification paths SHOULD be supported, providing a five-year overlap during which counterparties migrate at their own pace.

* For deployments under a statutory retention obligation (for example MiCA Art. 80, AMLR Art. 56, or DORA Art. 14), hybrid-pqc is REQUIRED for new receipts. Long-retention material is exactly the class exposed to the harvest-now-decrypt-later adversary.

* For deployments without such an obligation, hybrid-pqc is RECOMMENDED.

* After 2030, classical-es256k SHOULD be deprecated for new receipt issuance. The exact deprecation date follows applicable NIST, ANSSI, BSI, and eIDAS guidance (Section 4.5); this document does not fix a single date, because the authoritative horizon is set by the named bodies and may move.

A facilitator in a regulated sector SHOULD publish its migration plan in advance of any deprecation step that affects external counterparties, so that those counterparties are not excluded without notice.

4.5. NIST and Jurisdictional Alignment

The discipline of this section maps to published migration guidance:

* NIST PQC migration roadmap [NIST-PQC-MIGRATION] and the hybrid composition principles for hash-based signature schemes in SP 800-208 [SP800-208], adapted here to the dual-signature hybrid-pqc construction.

* ANSSI guidance on post-quantum migration [ANSSI-PQC], which recommends hybrid classical-plus-post-quantum constructions during the transition.

* BSI Technical Guideline TR-02102 [BSI-PQC].

* EU eIDAS 2.0 [EIDAS-2], which sets the trust-services horizon for the European single market.

ML-DSA-65 is selected as the post-quantum signature component because it is the NIST-standardised lattice signature at the security category appropriate for long-retention financial material. ML-KEM [FIPS203] is referenced for completeness as the companion key-encapsulation standard; key establishment is out of scope for this document, which addresses receipt integrity rather than transport confidentiality.

5. Starknet On-Chain Anchor

This section defines the format and verification of an on-chain settlement anchor on Starknet. The anchor closes the on-chain finality gap of Section 1: it lets a verifier confirm, without facilitator cooperation, that a settlement reached canonical confirmation at a specific block height. The anchor is OPTIONAL; a deployment that does not require ledger-anchored finality MAY omit it.

5.1. Canonical Anchor Tuple

An on-chain settlement record is identified by a five-field tuple embedded in the PAYMENT-RESPONSE extension envelope when on-chain submission has been confirmed:

Field	Type	Required	Meaning
chain_id	string	yes	Starknet chain identifier (Section 5.2).
tx_hash	string	yes	Transaction hash, felt252 hex (Section 5.2).
event_index	integer	yes	Zero-based index of the settlement event within the transaction receipt.
block_number	integer	no	Block height of inclusion; omitted while the transaction is pending.
kind	string	yes	Settlement class discriminant: "settlement", "refund", or "delegation".

Table 6

The anchor tuple identifies the transaction but not the emitting contract; a verifier MUST additionally check the emitter address per Section 5.8.

5.2. Anchor Format Discipline

chain_id is one of the Starknet-native identifiers "SN_MAIN" (mainnet) or "SN_SEPOLIA" (Sepolia testnet). These are documented in [STARKNET-DOCS] and are not IANA-registered tokens (Section 7).

tx_hash is a felt252 value encoded as a 66-character string: the prefix 0x followed by 64 lowercase hexadecimal characters, zero-padded on the left. Verifiers MUST compare tx_hash values as normalised lowercase hex; an uppercase or unpadded form MUST be normalised before comparison, not rejected.

event_index is zero-based. The kind discriminant maps to the three settlement classes; the values derive from the lifecycle state names of [LIFECYCLE-FSM] but require no separate registration in this document.

The anchor object is embedded as an extension field of the PAYMENT-RESPONSE only after on-chain submission has been confirmed. A facilitator MUST NOT embed an anchor tuple for a transaction it has merely broadcast; the tuple asserts inclusion, not submission.

5.3. Settlement Event Layout

A conforming emitter emits, for each settled payment, a Starknet event with the following two-key, three-data structure:

Slot	Content
keys[0]	Selector for the PaymentSettled event (or the equivalent ABI event name).
keys[1]	The felt252-masked action_ref digest (Section 5.4).
data[0]	The kind discriminant: 0x1 (settlement), 0x2 (refund), 0x3 (delegation).
data[1]	The felt252-masked low bits of payment_hash.
data[2]	Reserved. Emitters MUST write 0x0. Verifiers MUST NOT reject an anchor whose data[2] is non-zero.

Table 7

The reserved-field discipline (data[2]) preserves forward compatibility: a future revision may assign meaning to the slot, and existing verifiers must not treat a populated slot as a validation failure.

5.4. felt252 Encoding Mask

Starknet field elements are bounded below 2^252. A SHA-256 digest is 256 bits and therefore does not always fit in a single felt252. This document fixes a deterministic truncation applied to any SHA-256 digest written to or compared against an on-chain felt252 slot:

```
hash_felt252 = sha256_bigint & ((1 << 251) - 1)
```

The mask retains the low 251 bits. Implementations MUST apply the identical mask at both emission time (when writing keys[1] and data[1]) and verification time (when recomputing the expected values). The collision probability introduced by retaining 251 of 256 bits is negligible (on the order of 2^-251) and does not weaken the binding for any practical adversary.

An implementation that omits the mask at verification time produces false-negative verifications for receipts whose digest has non-zero bits in positions 251 through 255; see Section 6.16.

5.5. RPC Endpoint Convention

Anchor verification requires read access to a Starknet JSON-RPC endpoint. The methods used are starknet_getTransactionByHash, starknet_getTransactionReceipt, and starknet_getBlockWithTxHashes, as defined in [STARKNET-DOCS].

Verifiers MUST use TLS-authenticated endpoints and SHOULD prefer endpoints they operate themselves (Section 6.10). The Vauban Pay reference deployment operates a self-hosted Pathfinder [PATHFINDER] full node at https://sepolia.rpc.vauban.tech/rpc/v0_10 (Sepolia, Starknet JSON-RPC spec v0.10.2) and https://rpc.vauban.tech/rpc/v0_10 (mainnet). These endpoints are cited as one conformant implementation; this specification mandates no specific endpoint operator.

5.6. Reference Implementation: PaymentDemoEmitter

The PaymentDemoEmitter Cairo contract is deployed on Starknet Sepolia at 0x044dd87a94a801cf775d4c5e4b6703102d4e97e1cd1d0a8879341219ae4f19ff. It emits the settlement event layout of Section 5.3 and is the reference against which the anchor verification algorithm (Section 5.8) is exercised.

The contract is a demonstration surface only. It MUST NOT be used as a production payment processor; it performs no value transfer and applies no access control suitable for production. Its purpose is to provide a stable, publicly browsable on-chain event that conforms to Section 5.3, so that an independent verifier can reproduce the anchor verification end to end.

The contract is browsable on Voyager (Section 5.7) at https://sepolia.voyager.online/contract/0x044dd87a94a801cf775d4c5e4b6703102d4e97e1cd1d0a8879341219ae4f19ff.

5.7. Block Explorer Canonical Reference

Voyager [VOYAGER] is the canonical human-readable audit surface for Starknet mainnet and Sepolia in this document. Transaction and contract URLs follow the patterns https://voyager.online/tx/<tx_hash> and https://voyager.online/contract/<address> for mainnet, with the sepolia.voyager.online host for the testnet.

The Voyager URL is supplementary. The normative anchor check is the RPC verification of Section 5.8, not a browser visit. Implementations MUST NOT cite any block explorer other than Voyager as the canonical visual audit surface for Starknet; the specification endorses no alternative explorer for this role. The spoofing risk of relying on a browser visit alone is treated in Section 6.12.

5.8. Anchor Verification Algorithm

A verifier confirms an anchor by executing the following steps in order. A verifier MUST NOT skip a step.

1. Validate chain_id against the deployment environment the verifier is configured for; reject on mismatch (Section 6.15).

2. Retrieve the transaction receipt via starknet_getTransactionReceipt for tx_hash.

3. Validate the emitter: confirm that events[event_index].from_address equals a known-good emitter address from an implementation-controlled list (Section 6.13).

4. Validate block_number against the receipt, where block_number is present.

5. Compute the expected keys[1] by applying the felt252 mask (Section 5.4) to the receipt’s action_ref, and compare it against the on-chain keys[1].

6. Compare data[1] against the felt252-masked low bits of payment_hash as an OPTIONAL cross-check.

7. Validate the kind discriminant in data[0] against the tuple’s kind field.

8. Check the block finality status against the verifier’s configured threshold (Section 6.14); a verifier under a regulatory finality obligation MUST require L1 acceptance rather than L2 acceptance alone.

6. Security Considerations

6.1. Privacy Class Threat Model

The three receipt variants defined in Section 3 expose different amounts of information to a verifier or third-party observer. Implementations MUST select the variant whose disclosure profile matches the deployment's privacy class.

The stark-vauban-pay-v1 variant proves payment conditions (amount, currency, payer attestation, nullifier) without revealing the witness. A verifier confirms that the conditions held without learning their values. However, the action_ref digest is deterministic given the preimage. If the preimage is known to an adversary, the adversary can confirm whether a given payment was bound to a specific work event. Implementations that require unlinkability MUST use opaque, non-guessable agent_id and scope values in the preimage. The nullifier set, when exposed to third parties, allows correlation of repeated nullifier checks against unrelated transactions; nullifier sets SHOULD be exposed only under access controls aligned with the deployment's privacy policy.

The hybrid-pqc and classical-es256k receipts do not provide zero-knowledge properties. They carry the receipt_core in cleartext as JCS-canonical JSON. Parties sharing these receipts with third-party verifiers SHOULD treat the full payment conditions as visible. A regulator examining a hybrid-pqc receipt sees the full settlement detail; a third-party logging service receiving a classical-es256k receipt for delivery confirmation similarly sees the full detail.

The three variants do not represent a single security hierarchy; they represent orthogonal disclosure profiles. A deployment requiring both proof-condition privacy and post-quantum signature integrity MAY combine a stark-vauban-pay-v1 proof with a hybrid outer signature in the same envelope, provided both cover the byte-identical canonical preimage.

6.2. Replay Nullification

The NullifierReplay error code, returned with HTTP 409, is the primary mechanism for preventing double-spend. Facilitators MUST maintain a nullifier store indexed by the settlement ledger and MUST reject any PAYMENT-SIGNATURE whose nullifier is already present. The store MUST be durable across facilitator restarts; an in-memory-only store creates a replay window across process boundaries.

The nullifier is bound to the receipt via the canonical preimage and the underlying STARK proof (for stark-vauban-pay-v1) or via the JCS-canonical receipt_core (for hybrid-pqc and classical-es256k). An adversary attempting to replay a settled payment MUST produce a fresh nullifier; if the underlying cryptographic scheme is sound, the adversary cannot do so without the witness material.

The timestamp_ms field in the action_ref preimage further bounds the replay window. Facilitators SHOULD enforce a maximum timestamp_ms age aligned with the maxTimeoutSeconds field in the PaymentRequired response. Receipts whose timestamp_ms is older than the configured threshold MUST be rejected with HTTP 410 Expired regardless of nullifier state.

On-chain anchoring (Section 5) adds a second-layer replay defence. A settled payment whose anchor tuple has been emitted on the canonical chain is verifiable by any party; an adversary attempting to issue a duplicate settlement after on-chain anchoring would produce a divergent (payment_hash, action_ref) pair detectable by any verifier with read access to the chain.

6.3. Canonical Preimage Validation Order

Implementations MUST validate the canonical preimage discipline of Section 3.4 BEFORE acting on any receipt content. Accepting a receipt whose action_ref was derived from a non-canonical preimage (float timestamp_ms, missing fields, duplicate keys, non-NFC strings) creates a binding gap: the receipt may appear structurally valid while the work-layer binding is silently broken. A verifier that processes receipt content before checking preimage validity cannot assert the causal link between the payment and the work event.

The validation order is fixed:

1. Parse the receipt envelope and extract the canonical preimage object.

2. Apply type validation per Section 3.4 (reject float timestamp_ms, missing fields, duplicate keys).

3. Apply Unicode NFC validation per Section 3.4 (reject non-NFC strings).

4. Compute the JCS canonical bytes and the SHA-256 digest.

5. Compare the computed digest against the action_ref field of the receipt.

6. Only after digest match: verify cryptographic signatures or STARK proof.

7. Only after signature or proof verification: act on receipt content.

A verifier MUST NOT short-circuit any of these steps. Acceptance of a non-canonical preimage by a verifier propagates the violation into every downstream system that trusts the verifier's output.

6.4. Delegation Chain Bounds

The cryptographic scope of delegation receipts to granting principals is defined in [DELEGATION-BINDING] (deferred to a companion document). This consolidated document does not specify normative content for delegation binding. A receipt produced by a delegated agent under a DelegationGrant SHOULD include the granting principal's identifier in the agent_id field of the action_ref preimage, and the delegation scope in the scope field, so that a verifier with access to the delegation policy can confirm the bounds. The exact delegation policy schema, the verifier-side bounds-check algorithm, and the cryptographic linkage between the delegation grant and the derived receipt are out of scope for this document and are addressed in the companion work.

A verifier of a non-delegated receipt MAY ignore the delegation policy machinery entirely. A verifier of a delegated receipt without access to the companion delegation specification MUST treat the delegation chain as unbound and SHOULD NOT accept the receipt as evidence of delegated authority.

6.5. Downgrade Attack

An adversary in a network position between the client and the facilitator MAY attempt to strip hybrid-pqc from the X-Payment-Options header, forcing the client to fall back to classical-es256k. The resulting receipt is not post-quantum sound, leaving long-term integrity exposed.

Mitigation: a client requiring post-quantum soundness MUST send receipt_format with required: true in the PAYMENT-SIGNATURE extension. A facilitator that cannot honour the request MUST return HTTP 402 with UnsupportedReceiptFormat rather than silently downgrading. Verifiers SHOULD cross-check the X-Receipt-Format header on the PAYMENT-RESPONSE against the client's required variant and reject mismatches.

6.6. Insecure Hybrid Composition

A naive hybrid signature implementation that signs different canonical messages with the two component schemes does not provide the intended security property. An adversary who breaks ES256K can produce a forged classical signature, and the verifier may accept the receipt if the verification logic treats the two signatures as covering disjoint content.

Mitigation: implementations MUST follow the composition principles described in Section 4.3; both signatures MUST cover the byte-identical JCS canonical preimage. The reference encoder produces a single canonical byte string consumed by both signing operations.

6.7. ML-DSA-65 Side-Channel Exposure

ML-DSA-65 signing operations involve rejection sampling, secret-dependent control flow, and arithmetic operations whose timing may leak secret-key material to a co-located adversary. Side-channel analysis of lattice-based signatures is an active research area.

Mitigation: ML-DSA-65 signing implementations used in production facilitator infrastructure MUST be constant-time and SHOULD be selected from libraries that have undergone public side-channel analysis since the publication of [FIPS204].

6.8. Migration Timing Risk

Premature deprecation of classical signature support may exclude legitimate counterparties that have not yet migrated to post-quantum infrastructure. Conversely, late migration leaves long-retention receipts exposed to retroactive forgery.

Mitigation: deployments SHOULD align their deprecation schedule with applicable jurisdictional guidance ([NIST-PQC-MIGRATION], [ANSSI-PQC], [BSI-PQC], [EIDAS-2]). The 2025-2030 hybrid deployment window provides a five-year overlap during which both verification paths SHOULD be supported. Facilitators in regulated sectors SHOULD publish their migration plan in advance of any deprecation step affecting external counterparties.

6.9. STARK Proof System Auditability

The post-quantum soundness claim for the stark-vauban-pay-v1 variant relies on the collision and second-preimage resistance of the underlying hash function and on the correctness of the prover-verifier implementation. A flaw in the proof system implementation may produce false-positive verifications independent of the hash function's strength.

Mitigation: implementations SHOULD use a reference STARK prover that has undergone public audit. The Stwo Circle STARK M31 prover [STWO], deployed operationally on Starknet mainnet since 2025, is the reference for this document. Implementations that fork or modify the reference prover SHOULD re-audit the modifications against the original soundness analysis before production deployment. The hash-based property of the underlying proof system is preserved across faithful adapter implementations; the property breaks under naive substitution of an algebraic SNARK (Section 4.2 forbids this substitution under the post-quantum soundness claim).

The post-quantum soundness claim is a security ceiling, not a security floor; it states that no known quantum algorithm reduces the proof system's soundness, not that the proof system is immune to future cryptanalysis. This document makes no claim about the long-term security of any specific proof system configuration beyond the current state of the published analysis.

6.10. Self-Hosted Infrastructure Assertion

The reference Vauban Pay deployment operates a self-hosted Pathfinder [PATHFINDER] Starknet full node at https://sepolia.rpc.vauban.tech/rpc/v0_10 (Sepolia testnet, JSON-RPC spec v0.10.2) and https://rpc.vauban.tech/rpc/v0_10 (mainnet). The on-chain anchor format of Section 5 can be verified against these endpoints or against any conformant Starknet JSON-RPC implementation operated by a party the verifier trusts.

The specification does not mandate the use of any specific endpoint operator. A verifier MAY operate its own Starknet full node, use a third-party endpoint of its choice, or any combination of the two. The protocol's anchor verification algorithm (Section 5.8) is independent of the endpoint provider; it requires only that the endpoint implement the Starknet JSON-RPC specification and that the TLS chain be authenticated.

Verifiers SHOULD NOT depend on any single third-party endpoint operator for anchor verification under regulatory audit obligations. The audit-resilience property of an on-chain anchor (that an auditor

can re-verify at year N without facilitator cooperation) is undermined if the verifier depends on a third-party endpoint that may withdraw service, impose access controls incompatible with audit timelines, or otherwise constrain re-verification. Verifiers operating under MiCA Art. 76, DORA Art. 14, or comparable obligations SHOULD operate their own Starknet read infrastructure or contract with multiple independent endpoint operators with cross-checked responses.

6.11. RPC Endpoint Authenticity

An anchor verification that relies on a network RPC endpoint is subject to man-in-the-middle attack if the transport is not authenticated. An adversary that controls the network path between the verifier and the RPC endpoint may return fabricated event data, causing the verifier to falsely confirm an anchor that was never committed to the canonical chain.

Mitigation: verifiers MUST use TLS-authenticated RPC endpoints. Verifiers SHOULD prefer endpoints they operate themselves or whose TLS certificate chain they have independently pinned. Verifiers MUST NOT accept RPC responses over unencrypted HTTP for anchor verification purposes.

6.12. Block Explorer Spoofing

A Voyager URL included in an audit submission is a supplementary human-readable convenience. An adversary controlling a domain similar to voyager.online or sepolia.voyager.online may display fabricated transaction data. Auditors relying solely on a browser visit to a Voyager URL, without independently verifying the anchor tuple via the Starknet JSON-RPC, cannot distinguish canonical chain data from spoofed presentation.

Mitigation: the normative audit check is the anchor tuple verification via the Starknet JSON-RPC, not the Voyager URL. Audit frameworks MUST perform RPC verification as the authoritative step. Voyager URLs are RECOMMENDED as a supplementary aid for human reviewers; they MUST NOT be the sole verification artefact. Implementations MUST NOT cite any block explorer other than Voyager as the canonical visual audit surface for Starknet; the specification does not endorse any alternative explorer for this role.

6.13. Event Collision and Selector Ambiguity

If two Cairo contracts deployed at different addresses emit events with the same keys[0] selector and the same keys[1] digest, a verifier that does not check the emitting contract address may accept the wrong anchor. This is especially relevant when a chain hosts multiple conforming emitters.

Mitigation: verifiers MUST validate that the event was emitted by the expected contract address. The anchor tuple (Section 5.1) identifies the transaction but not the emitting contract; verifiers MUST cross-reference the events[i].from_address field in the starknet_getTransactionReceipt response against the known emitter address before accepting the event as a valid anchor. Conforming emitters MUST be registered in an implementation-controlled list; events from unregistered emitters MUST be rejected.

6.14. Chain Reorganisation and Finality Window

Starknet blocks achieve L2 finality when their state update is proven and posted to the Ethereum L1 settlement layer. Prior to L1 finalisation, a block may in principle be reorganised. A verifier that confirms a Starknet anchor at a block that has not yet reached L1 finality accepts a non-final commitment.

Mitigation: for settlement records that require finality guarantees under regulatory frameworks, verifiers SHOULD wait for the Starknet block’s state update to be confirmed on Ethereum L1 before treating the anchor as final. The Starknet JSON-RPC exposes block status fields that allow verifiers to distinguish pending, accepted-on-L2, and accepted-on-L1 states [STARKNET-DOCS]. Verifiers MUST NOT treat a pending or L2-only block as providing the same finality guarantee as an L1-confirmed block. For non-regulatory use cases (for example, low-value instant payments), L2 acceptance MAY be sufficient; implementations MUST document their chosen finality threshold.

6.15. Chain Identifier Confusion

An anchor tuple carrying chain_id: "SN_SEPOLIA" references a testnet where tokens have no economic value. A verifier that does not validate the chain_id field against its expected deployment environment may accept a testnet anchor as proof of mainnet settlement.

Mitigation: verifiers MUST reject anchors whose chain_id does not match the deployment environment they are configured to verify. Production payment processors MUST be configured to accept only chain_id: "SN_MAIN". Test environments that accept chain_id:

"SN_SEPOLIA" MUST be segregated from production verification pipelines. The facilitator MUST include the chain_id in the anchor tuple; a missing chain_id MUST be treated as a malformed anchor and MUST be rejected.

6.16. felt252 Mask and On-Chain Comparison

The felt252 masking operation described in Section 5.4 is a deterministic transformation, not a cryptographic weakening. Implementations that omit the mask at verification time will silently produce false-negative verification results for receipts whose SHA-256 digest has non-zero bits in positions 251 through 255. Such false negatives do not create a security vulnerability (they deny valid receipts rather than accept invalid ones), but they create an operational failure mode in production payment pipelines. Implementations SHOULD test the mask against a receipt whose action_ref digest has a known non-zero high bit before deploying to production.

6.17. Facilitator Trust Boundary

The security model of this extension assumes facilitator integrity for receipt issuance. A compromised facilitator can issue false receipts regardless of the cryptographic variant. The extension does not provide a mechanism to verify that the facilitator was itself uncompromised at the time of issuance. Verifiers that require a trust-minimised settlement proof SHOULD combine the stark-vauban-pay-v1 receipt with a Starknet on-chain anchor per Section 5. The combination reduces the trust assumption from "facilitator integrity at issuance" to "facilitator integrity at the single-step anchor emission to the canonical ledger". An auditor re-verifying in year N then relies on the chain’s canonical history, not on the facilitator’s continued availability.

6.18. Retention-Property Determinism

The canonical preimage discipline in Section 3.4 produces a digest that two observers verifying at the same instant compute identically. For receipts emitted under a regulatory framework with a statutory retention obligation, the property MUST also hold across time: a supervisor or auditor re-verifying in year N against a retained off-VM manifest of the receipt MUST reproduce the same digest as the original verifier did at issuance time.

This raises a versioning concern. If the canonical rule (JCS, type-validation, field-name canonicalisation, Unicode normalisation policy) is revised between issuance and re-verification, the retained receipt becomes unreproducible under the then-current rule.

Verifier-side coercion of non-conforming inputs (rather than rejection) further breaks re-verifiability because the coercion step is verifier-local and is not replayed at audit time.

Producers emitting receipts under frameworks with statutory retention obligations (MiCA Art. 80, AMLR Art. 56, DORA Art. 14) MUST therefore:

* Include the canonicalisation version marker in a canon_version field in the receipt preimage at emission time. The value MUST be a registered x402 canonicalisation version identifier; the JCS discipline validated in [X402-CANON] is identified by the wire-format marker jcs-rfc8785-v1. A future verifier selects the contemporaneous rule by this marker rather than applying the then-current rule.

* Reject (not coerce) non-conforming inputs at the parse or schema-validation step, preserving re-verifiability against the raw retained object.

Producers emitting receipts outside such frameworks SHOULD include canon_version as a good-discipline default; the field becomes a MUST under any framework that imposes a statutory retention horizon.

6.19. Timing Side-Channel Risks in STARK Proof Generation

STARK proof generation for the stark-vauban-pay-v1 variant is computationally intensive and its duration correlates with the witness size. On shared infrastructure, an adversary observing proof generation latency MAY be able to infer approximate witness content (for example, payment amount magnitude or attestation complexity).

Mitigation: implementors SHOULD run proof generation in isolated compute environments (dedicated VMs or sandboxed processes) and SHOULD add randomised padding to the proof generation timeline where latency is observable by external parties. Implementations operating in regulated sectors SHOULD document their timing-side-channel posture in the operational deployment documentation.

6.20. Open Research Items

The composition properties of payment claims across receipt formats, lifecycle states, and delegation chains are under active investigation in the companion documents [VPSF-ALGEBRA], [LIFECYCLE-FSM], and [DELEGATION-BINDING]. The composition algebra is not in scope for this document; implementations of the consolidated receipt-format, post-quantum discipline, and on-chain anchor SHOULD treat composition properties as working hypotheses

pending the publication of the companion specifications. A companion paper covering the formal treatment is planned for submission to IACR ePrint; implementors are directed there for the formal analysis once published.

7. IANA Considerations

This document makes no request for IANA action.

The registry described as x402 Receipt Format is an internal registry maintained by the x402 Foundation Technical Steering Committee, not an IANA registry. The registration procedure is Specification Required [RFC8126], with Designated Experts selected by the Technical Steering Committee. No allocation in this document requires IETF Review or Standards Action.

The chain identifier values "SN_MAIN" and "SN_SEPOLIA" are Starknet-native identifiers documented in [STARKNET-DOCS]; they are not IANA-registered tokens. The kind discriminant values ("settlement", "refund", "delegation") derive from the lifecycle state names defined in [LIFECYCLE-FSM]; they do not require separate IANA registration in this specification.

Future per-chain adapter specifications that introduce new chain_id values SHOULD define a registration mechanism within the x402 Foundation registry structure rather than reserving values through IANA.

8. Conformance Checklist

An implementation claiming conformance to this consolidated specification MUST satisfy the following, organised by section.

Receipt format (Section 3):

* Produce and parse all three receipt format tokens.
* Set X-Payment-Options on 402 responses listing supported tokens.
* Set X-Receipt-Format on all PAYMENT-RESPONSE messages.
* Fall back to classical-es256k when the client sends no receipt_format preference.
* Return HTTP 402 with UnsupportedReceiptFormat when a client requests a variant with required: true that the facilitator cannot produce.

Preimage discipline (Section 3.4):

* Reject float values for timestamp_ms before JCS canonicalisation.

* Reject preimage objects with missing canonical fields before JCS canonicalisation.

* Reject preimage objects with duplicate keys at parse time.

* Normalise all preimage string values to NFC per [UAX15] before JCS canonicalisation, and reject non-NFC input.

* Pass every vector in the conformance suite at [VAUBAN-CONFORMANCE].

Post-quantum discipline (Section 4):

* Cover the byte-identical JCS canonical preimage with both component signatures of any hybrid-pqc receipt.

* Reject hybrid-pqc receipts where either component signature fails to verify.

* Use a hash-based proof system for any receipt variant claiming post-quantum proof-layer soundness.

On-chain anchor (Section 5):

* Produce a well-formed anchor tuple for each submitted settlement.

* Include chain_id, tx_hash, event_index, and kind in every anchor tuple.

* Apply the felt252 mask consistently at both event emission time and verification time.

* Validate the emitter contract address against the known-good emitter address before accepting an event as a valid anchor.

* Reject anchors whose chain_id does not match the configured deployment environment.

* Distinguish pending, L2-accepted, and L1-accepted block states and apply the appropriate finality threshold for the use case.

* Cite Voyager as the canonical visual audit surface for Starknet, and not cite any other explorer in audit submissions.

Known Adopters and Reference Implementations

This appendix documents reference implementations and adopters of this specification confirmed at the time of publication. The list is informational and will be updated in subsequent revisions as additional implementations are reported.

Vauban Pay (https://pay.vauban.tech) maintains the reference specification, the published conformance vectors (https://github.com/vauban-org/x402-stark-receipts-conformance), the multi-language runner matrix described in Section 3.10, and the on-chain reference deployment. The Vauban Pay live demonstration [VAUBAN-DEMO] emits compliant payloads at https://demo.pay.vauban.tech. The on-chain anchor PaymentDemoEmitter is deployed on Starknet Sepolia at contract 0x044dd87a94a801cf775d4c5e4b6703102d4e97e1cd1d0a8879341219ae4f19ff (Voyager-verified at https://sepolia.voyager.online/contract/0x044dd87a94a801cf775d4c5e4b6703102d4e97e1cd1d0a8879341219ae4f19ff). The Starknet RPC reference endpoint is a self-hosted Pathfinder validator [PATHFINDER] at https://sepolia.rpc.vauban.tech/rpc/v0_10.

The published Vauban Pay packages across three ecosystems are:

* crates.io: vauban-x402-jcs-conformance@0.1.0, vauban-x402-canonical@0.1.0, vauban-x402-wire@0.1.0
* PyPI: vauban-x402-stark-receipt@0.1.0
* npm: @vauban-pay/substrate@0.1.0

All packages are released under Apache 2.0.

Implementers SHOULD notify the contact at research@vauban.tech when adopting this specification in production. Adoption notifications include the production endpoint or product identifier, the emitted canon_version value, the JCS implementation library and version, the deployed emitter contract address with chain identifier (where applicable), the RPC endpoint provenance (self-hosted versus third-party), and a contact for follow-on coordination. Reported adopters will be listed in the next revision of this appendix following a verification step against the conformance vector matrix.

Acknowledgments

This consolidated document folds material developed across the x402 Linux Foundation V2 coalition review tracks [X402-2357], [X402-2398], [X402-2411], [X402-2412], [X402-2413], [X402-2434], [X402-2440], and [X402-2459]. The editor thanks the following contributors by name:

* FeedOracle, for the hybrid-PQC reference implementation and the receipt-core fixture set landed at [X402-2411].

* andysalvo, for the action-ref-verify v0.3.0 conformance vectors landed at [X402-2398], which materially shaped the JCS preimage discipline profiled in Section 3.

* egoriklok, for the x402 maintainer review summary at [X402-2459], which consolidated the substrate-and-axes architecture and clarified the boundary between the receipt-format extension and the deferred composability layer.

* The Vauban Research engineering team, for the collective drafting, conformance-vector publication, multi-language runner matrix, on-chain demo surface, and editorial discipline across the three folded source drafts.

The JCS canonicalisation discipline referenced in Section 3.4 is grounded in [RFC8785] and validated by independent runner implementations against published conformance suites [X402-CANON]. The Stwo Circle STARK M31 prover [STWO] is the open-source reference for the hash-based proving system underlying the stark-vauban-pay-v1 variant. The Starknet JSON-RPC specification [STARKNET-DOCS] and the Pathfinder full node [PATHFINDER] provide the on-chain infrastructure that the anchor format targets. The Voyager block explorer [VOYAGER] provides the canonical human-readable audit surface.

References

Normative References

[FIPS204] National Institute of Standards and Technology, "Module-Lattice-Based Digital Signature Standard (ML-DSA)", August 2024, <https://doi.org/10.6028/NIST.FIPS.204>.

[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/rfc/rfc2119>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/rfc/rfc4648>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/rfc/rfc8126>.

[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/rfc/rfc8174>.

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/rfc/rfc8259>.

[RFC8785] Rundgren, A., Jordan, B., and S. Erdtman, "JSON Canonicalization Scheme (JCS)", RFC 8785, DOI 10.17487/RFC8785, June 2020, <https://www.rfc-editor.org/rfc/rfc8785>.

[RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <https://www.rfc-editor.org/rfc/rfc8949>.

[RFC9110] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <https://www.rfc-editor.org/rfc/rfc9110>.

[SP800-208] National Institute of Standards and Technology, "Recommendation for Stateful Hash-Based Signature Schemes", October 2020, <https://doi.org/10.6028/NIST.SP.800-208>.

Informative References

[ANSSI-PQC] Agence nationale de la securite des systemes d’information, "ANSSI Position Paper on Post-Quantum Cryptography Migration", n.d., <https://cyber.gouv.fr/en/publications/anssi-views-post-quantum-cryptography>.

[BSI-PQC] Bundesamt fur Sicherheit in der Informationstechnik, "BSI Technical Guideline TR-02102 (cryptographic mechanisms)", n.d., <https://www.bsi.bund.de/EN/Themen/Unternehmen-und-Organisationen/Standards-und-Zertifizierung/Technische-Richtlinien/TR-nach-Thema-sortiert/tr02102/tr02102_node.html>.

[DELEGATION-BINDING]
"x402 V2 Delegation Binding (companion work, out of scope)", n.d., <https://datatracker.ietf.org/doc/draft-vauban-x402-delegation-binding/>.

[EIDAS-2] European Parliament and Council, "Regulation (EU) 2024/1183 amending Regulation (EU) No 910/2014 (eIDAS 2.0)", April 2024, <https://eur-lex.europa.eu/eli/reg/2024/1183/oj>.

[FIPS203] National Institute of Standards and Technology, "Module-Lattice-Based Key-Encapsulation Mechanism Standard (ML-KEM)", August 2024, <https://doi.org/10.6028/NIST.FIPS.203>.

[LIFECYCLE-FSM]
"x402 V2 Payment Lifecycle Finite State Machine (companion work, out of scope)", n.d., <https://datatracker.ietf.org/doc/draft-vauban-x402-lifecycle-fsm/>.

[NIST-PQC-MIGRATION]
National Institute of Standards and Technology, "Post-Quantum Cryptography Migration Roadmap", n.d., <https://csrc.nist.gov/projects/post-quantum-cryptography>.

[PATHFINDER]
"Pathfinder Starknet full node (eqlabs)", n.d., <https://github.com/eqlabs/pathfinder>.

[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/rfc/rfc7515>.

[STARKNET-DOCS]
"Starknet Documentation and JSON-RPC Specification", n.d., <https://docs.starknet.io>.

[STWO] "Stwo Circle STARK Prover (StarkWare Industries)", n.d., <https://github.com/starkware-libs/stwo>.

[UAX15] Unicode Consortium, "Unicode Normalization Forms", Unicode Standard Annex #15, 2023, <https://www.unicode.org/reports/tr15/>.

[VAUBAN-CONFORMANCE]
"Vauban STARK receipt conformance vectors (public)", n.d., <https://github.com/vauban-org/x402-stark-receipts-conformance/blob/main/manifest.json>.

[VAUBAN-DEMO]
"Vauban Pay reference demonstration (Starknet Sepolia)", n.d., <https://demo.pay.vauban.tech>.

[VOYAGER] "Voyager Starknet Block Explorer", n.d., <https://voyager.online>.

[VPSF-ALGEBRA]
"VPSF Claim Algebra for x402 Payment Receipts (companion work, out of scope)", n.d., <https://datatracker.ietf.org/doc/draft-vauban-x402-vpsf-algebra/>.

[X402-2322]
"x402 evidenceType taxonomy and composite trust-query alignment (discussion thread)", n.d., <https://github.com/x402-foundation/x402/issues/2322>.

[X402-2326]
"x402 shared canonicalisation discipline (coalition discussion thread)", n.d., <https://github.com/x402-foundation/x402/issues/2326>.

[X402-2357]
"x402 STARK Receipts Extension Proposal", n.d., <https://github.com/x402-foundation/x402/issues/2357>.

[X402-2398]
"action-ref-verify v0.3.0 conformance vectors", n.d., <https://github.com/x402-foundation/x402/pull/2398>.

[X402-2411]
"Hybrid-PQC receipt-core fixture set (Axis 2)", n.d., <https://github.com/x402-foundation/x402/pull/2411>.

[X402-2412]
"Canonicalisation substrate v0 fixtures (Axis 0, 53 vectors)", n.d., <https://github.com/x402-foundation/x402/pull/2412>.

[X402-2413]
"stark-vauban-pay-v1 v0 fixtures (Axis 1)", n.d., <https://github.com/x402-foundation/x402/pull/2413>.

[X402-2434]
"risk-check-attestation-sample v0 (compliance-receipt fixture, ALLOW/DENY/REFER)", n.d., <https://github.com/x402-foundation/x402/pull/2434>.

[X402-2440]
"Composite trust-query normative layer (Axis 4)", n.d., <https://github.com/x402-foundation/x402/pull/2440>.

[X402-2459]
"x402 maintainer review summary (egoriklok)", n.d., <https://github.com/x402-foundation/x402/pull/2459>.

[X402-CANON]
"x402 canonicalisation substrate v0 conformance vectors (53 vectors, Axis 0)", n.d., <https://github.com/x402-foundation/x402/pull/2412>.

[X402-V2] "x402 Linux Foundation V2 Working Group", n.d., <https://github.com/x402-foundation/x402>.

Author's Address

F. Serafini (editor)
Vauban Research
Email: research@vauban.tech
URI: https://pay.vauban.tech