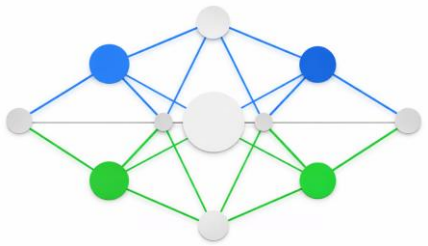


OCP

OpenCognition Protocol

"Where Minds Meet Machines"

OpenCognition Protocol (OCP)

Security Technical Specification v1.0

Introduction	January 2026
License	MIT Licenses
Conformance Identifier	OCP-SEC-SPEC-1.0
Specification	docs.opencognitionprotocol.org
Maintainer	OCP Security Working Group (SWG)

1. Introduction

1.1 Purpose

This document defines the complete security architecture of the OpenCognition Protocol. It specifies the threat model, cryptographic requirements, authentication and authorization mechanisms, privacy enforcement, key management lifecycle, recovery procedures, network security controls, audit requirements, and incident response processes. All OCP implementations MUST conform to this specification.

1.2 Scope

This specification covers security at every layer of the OCP stack:

Layer	Security Domain	Section
Layer 1: Transport & Discovery	Channel security, registry integrity	§5, §6
Layer 2: Trust & Identity	Authentication, authorization, identity	§4, §7
Layer 3: Knowledge Exchange	Data privacy, provenance integrity	§8
Layer 4: Collaboration	Consensus integrity, task isolation	§9
Layer 5: Application	Extension security, abuse prevention	§10
Cross-cutting	Key management, recovery, audit, incident response	§3, §11, §12, §13

1.3 Conventions

The keywords MUST, MUST NOT, REQUIRED, SHALL, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL are per RFC 2119. The term "implementation" refers to any software conforming to the OCP specification, whether SDK, node, registry, or custodian.

1.4 References

Document	Identifier
OCP Technical Specification	OCP-SPEC-1.0
OCP Python SDK Specification	OCP-SDK-PYTHON-SPEC-1.0
OCP Git Repository Specification	OCP-REPO-SPEC-1.0
W3C DID Core	https://www.w3.org/TR/did-core/
RFC 8032	Edwards-Curve Digital Signature Algorithm (Ed25519)
RFC 7748	Elliptic Curves for Security (X25519)
NIST SP 800-38D	AES-GCM
FIPS 202	SHA-3
RFC 5869	HKDF
RFC 8785	JSON Canonicalization Scheme (JCS)
FIPS 203	ML-KEM (post-quantum KEM)
FIPS 204	ML-DSA (post-quantum signatures)

2. Security Principles

All security decisions in OCP derive from five foundational principles:

2.1 Zero Trust

No agent, node, or message is trusted by default. Every interaction requires cryptographic authentication and explicit authorization. Trust is earned incrementally through the protocol's graduated trust model and is always revocable.

2.2 Defense in Depth

Security is enforced at every layer. Transport encryption protects channels. Message signing authenticates senders. The Privacy Validation Layer protects data. Bond permissions scope authorization. Each layer operates independently — a failure in one does not compromise others.

2.3 Least Privilege

Agents operate with the minimum permissions required. Anonymous agents can only read broadcasts. Identified agents can exchange messages. Vouched agents can share knowledge. Bonded agents can collaborate, but only within the specific permissions their bond defines. No agent has ambient authority.

2.4 No Backdoors

OCP contains no master keys, administrative override capabilities, privileged agents, or recovery mechanisms that bypass cryptographic verification. The OCP Foundation, SWG, TSC, or any other governing body MUST NOT have the ability to impersonate, decrypt, or override any agent. This principle is non-negotiable and not subject to governance votes.

2.5 Privacy by Design

Knowledge is shared; data never is. The protocol enforces this at the wire level through the Privacy Validation Layer. Privacy protections are mandatory, not optional — they cannot be disabled by configuration, agreement between agents, or extension.

3. Cryptographic Specification

3.1 Algorithm Suite

Function	Algorithm	Key Size	Standard	Status
Message signing	Ed25519	256-bit	RFC 8032	REQUIRED
Key exchange	X25519 (ECDH)	256-bit	RFC 7748	REQUIRED
Symmetric encryption	AES-256-GCM	256-bit	NIST SP 800-38D	REQUIRED
Hashing	SHA-3-256	256-bit	FIPS 202	REQUIRED
Key derivation	HKDF-SHA-256	—	RFC 5869	REQUIRED
Nonce generation	CSPRNG	96-bit (GCM), 256-bit (general)	OS-provided	REQUIRED
JSON canonicalization	JCS	—	RFC 8785	REQUIRED

3.2 Algorithm Requirements

3.2.1 Ed25519

- Implementations MUST use Ed25519 as specified in RFC 8032 §5.1.
- Private keys MUST be 32 bytes generated from a CSPRNG.
- Implementations MUST NOT use Ed25519ph (prehashed) or Ed25519ctx (context) variants unless explicitly specified by a future OCP extension.
- Signature output MUST be exactly 64 bytes.
- Implementations MUST reject signatures that are not exactly 64 bytes.
- Implementations MUST reject public keys that are not exactly 32 bytes.
- Implementations MUST perform full signature verification including checking that the public key is on the curve and the signature satisfies the verification equation.

3.2.2 X25519

- Implementations MUST use X25519 as specified in RFC 7748 §5.
- Private keys MUST be 32 bytes generated from a CSPRNG.
- Implementations MUST apply clamping as defined in RFC 7748 (clear bits 0, 1, 2, 255; set bit 254).
- Implementations MUST reject public keys consisting of all zeros.
- The shared secret output is 32 bytes. Implementations MUST NOT use the raw shared secret as an encryption key — it MUST be passed through HKDF.

3.2.3 AES-256-GCM

- Key MUST be exactly 32 bytes (256 bits).
- Nonce MUST be exactly 12 bytes (96 bits) generated from a CSPRNG.
- Nonce MUST be unique per key. Nonce reuse under the same key is a critical vulnerability that leaks plaintext.
- Authentication tag MUST be 16 bytes (128 bits). Implementations MUST NOT truncate the tag.
- Implementations MUST verify the authentication tag before processing or returning any plaintext. Failure MUST result in complete rejection — no partial plaintext output.
- Additional Authenticated Data (AAD) MAY be used and MUST be identical at encryption and decryption.

3.2.4 SHA-3-256

- Implementations MUST use SHA-3 (Keccak, FIPS 202), NOT SHA-256 (SHA-2 family).
- Output MUST be exactly 32 bytes (256 bits).
- SHA-3 is used for message signing digests, agent ID derivation, key fingerprints, and reproducibility hashes.

3.2.5 HKDF-SHA-256

- Implementations MUST use HKDF (RFC 5869) with HMAC-SHA-256.
- Salt: `None` (defaults to a string of `HashLen` zeros per RFC 5869 §2.2).
- Info strings MUST be protocol-defined constants from `ocp.constants`:
    - Message encryption: `b"ocp-v1-aes-key"`
    - Recovery share encryption: `b"ocp-v1-recovery-share"`
- Output length: 32 bytes (256 bits) for AES-256 key derivation.

3.2.6 Random Number Generation

- All random values (keys, nonces, IVs, shares, UUIDs) MUST be generated from the operating system's CSPRNG.
- On Linux: `/dev/urandom` or `getrandom(2)`.
- On macOS: `SecRandomCopyBytes`.
- On Windows: `BCryptGenRandom`.
- Implementations MUST NOT use userspace PRNGs (e.g., Python's `random` module, `Math.random()` in JavaScript) for any security-sensitive value.
- Implementations MUST NOT accept user-supplied seeds for key generation.

3.3 Prohibited Algorithms

The following algorithms MUST NOT be used in any OCP implementation:

Algorithm	**Reason**
RSA (any key size)	Key sizes too large for OCP message format; slower than Ed25519
ECDSA (any curve)	Nonce-dependent; signature malleability without normalization
DSA	Deprecated; nonce-dependent
AES-CBC	No authentication; padding oracle attacks
AES-CTR (without MAC)	No authentication
SHA-1	Collision attacks demonstrated
SHA-256 (SHA-2)	OCP standardizes on SHA-3 for hash domain separation
MD5	Broken
RC4	Broken
DES / 3DES	Insufficient key/block size
ChaCha20-Poly1305	Not prohibited but not approved; requires extension proposal

3.4 Post-Quantum Readiness

3.4.1 Reserved Algorithm Identifiers

Identifier	Algorithm	Standard	OCP Status
`PQ-ML-KEM-768`	ML-KEM-768	FIPS 203	Reserved for key encapsulation
`PQ-ML-KEM-1024`	ML-KEM-1024	FIPS 203	Reserved for key encapsulation
`PQ-ML-DSA-65`	ML-DSA-65	FIPS 204	Reserved for signatures
`PQ-ML-DSA-87`	ML-DSA-87	FIPS 204	Reserved for signatures
`PQ-SLH-DSA-128s`	SLH-DSA-SHA2-128s	FIPS 205	Reserved for stateless hash signatures

3.4.2 Hybrid Migration Strategy

When post-quantum extensions are activated:

1. **Key exchange:** X25519 + ML-KEM-768 hybrid. The combined shared secret is `HKDF(X25519_shared || ML-KEM_shared)`. Both algorithms MUST succeed.
2. **Signatures:** Ed25519 + ML-DSA-65 dual signatures. Both MUST be present and valid. A message with only one valid signature MUST be rejected.
3. **DID Documents:** New verification methods will use a `CompositeVerificationKey2026` type containing both classical and post-quantum public keys.
4. **Transition timeline:** Minimum 24-month dual-stack period where both classical-only and hybrid messages are accepted. After transition, classical-only messages are rejected.

3.4.3 Quantum Threat Assessment

Threat	Urgency	Mitigation
Harvest-now-decrypt-later (HNDL)	HIGH — adversaries may be recording OCP traffic today	Hybrid KEM deployment; ephemeral keys minimize HNDL window
Signature forgery	MEDIUM — requires fault-tolerant quantum computer	Dual signature scheme; Ed25519 remains secure until ~2035 (conservative estimate)
Agent impersonation	MEDIUM	Post-quantum DIDs via composite keys

4. Identity Security

4.1 Agent Identity Generation

4.1.1 DID Derivation

```
agent_id = "did:ocp:" + network + ":agent-" + SHA3-
256(Ed25519_public_key)[:6].hex()
```

Security properties:

- The DID is a commitment to the public key — changing the key requires a new DID.
- 48-bit truncation provides collision resistance of approximately 2^24 (16 million). This is sufficient for agent identification but NOT sufficient for collision-critical use cases. Full key fingerprints (128-bit) MUST be used for recovery verification.
- DID derivation is deterministic: same key always produces same DID. No entropy is added.

4.1.2 DID Document Security

- DID Documents MUST be served over HTTPS (TLS 1.3).
- DID Documents MUST be signed by the agent's Ed25519 key.
- Resolution via `.well-known` URLs MUST use the exact path `/.well-known/ocp/did.json`.
- Implementations MUST verify that the public key in the DID Document matches the DID (re-derive the DID from the public key and compare).
- DID Documents MUST NOT contain private keys, symmetric keys, or any secret material.

4.2 Authentication

4.2.1 Message Authentication

Every OCPUMF message is authenticated via Ed25519 signature:

```
signature = Ed25519_Sign(
    private_key,
    SHA3-256(
        JCS_Canonicalize(message with sender.signature = "")
    )
)
```

Verification procedure (receivers MUST follow all steps):

1. Extract `sender.signature` and `sender.agent_id`.
2. Set `sender.signature = ""` in a copy of the message.
3. Canonicalize the copy via RFC 8785 (JCS).
4. Compute `SHA3-256(canonical_json)`.
5. Resolve the sender's public key from their DID Document.
6. Verify the Ed25519 signature against the digest and public key.
7. REJECT the message if any step fails.

Critical: Implementations MUST NOT process the payload before verifying the signature. Unsigned or incorrectly signed payloads MUST be discarded completely.

4.2.2 WebSocket Authentication

```
Client → Server:
{
    "frame_type": "auth_handshake",
    "agent_id": "<DID>",
    "timestamp": "<ISO 8601 UTC>",
    "nonce": "<base64url 256-bit random>",
    "signature": "Ed25519_Sign(SHA3-256(agent_id || timestamp || nonce))"
}

Server → Client:
{
    "frame_type": "auth_result",
    "status": "accepted" | "rejected",
    "session_id": "<uuid>",
    "ttl": 86400
}
```

Server verification procedure:

1. Verify `timestamp` is within ±60 seconds of server time. REJECT if stale.
2. Verify `nonce` has not been seen before (replay protection). Record nonce for `ttl` seconds.
3. Resolve the sender's public key from the DID.
4. Verify the signature over `agent_id || timestamp || nonce`.
5. REJECT if any check fails. Close WebSocket with code 1002.
6. Timeout: server MUST close the connection if no valid handshake is received within 5 seconds.

4.2.3 HTTP Authentication

```
Authorization: OCP-Ed25519 <agent_id>:<timestamp>:<signature>
```

Where `signature = Ed25519_Sign(SHA3-256(agent_id || timestamp))`.

Server verification: Same as WebSocket but without nonce (HTTP is naturally request-response, reducing replay risk). Timestamp MUST be within ±60 seconds.

4.3 Authorization

Authorization in OCP is enforced through the trust level system and bond permissions.

4.3.1 Trust Level Access Matrix

Operation	Level 0	Level 1	Level 2	Level 3	Level 4
Receive broadcasts	✓	✓	✓	✓	✓
Send/receive messages	✗	✓	✓	✓	✓
Share knowledge	✗	✗	✓	✓	✓
Initiate bonds	✗	✗	✓	✓	✓
Full collaboration	✗	✗	✗	✓	✓
Task delegation	✗	✗	✗	✓	✓
Issue vouches	✗	✗	✓	✓	✓
Governance voting	✗	✗	✗	✗	✓

4.3.2 Bond Permission Enforcement

Implementations MUST check bond permissions before executing privileged operations:

```
BEFORE knowledge_share:
    IF bond exists with receiver:
        CHECK bond.is_active
        CHECK bond.permissions.knowledge_share.enabled
        CHECK knowledge_type IN
bond.permissions.knowledge_share.allowed_types
        CHECK payload_size <=
bond.permissions.knowledge_share.max_payload_bytes
    ELSE:
        CHECK sender.trust_level >= 2 AND receiver.trust_level >= 2

BEFORE task_request:
    REQUIRE bond exists with receiver
    CHECK bond.is_active
    CHECK bond.permissions.task_delegate.enabled
    CHECK active_tasks < bond.permissions.task_delegate.max_concurrent
```

5. Transport Security

5.1 TLS Requirements

Requirement	Specification
Minimum version	TLS 1.3 (RFC 8446)
Prohibited versions	TLS 1.0, 1.1, 1.2, SSL 2.0, SSL 3.0
Cipher suites	TLS_AES_256_GCM_SHA384, TLS_CHACHA20_POLY1305_SHA256, TLS_AES_128_GCM_SHA256
Key exchange	X25519, secp256r1 (in that preference order)
Certificate validation	REQUIRED — implementations MUST validate the full certificate chain
Certificate pinning	RECOMMENDED for production deployments
OCSP stapling	RECOMMENDED
SNI	REQUIRED

Prohibited: Unencrypted transport is NEVER permitted. An OCP message sent over plaintext HTTP or unencrypted WebSocket is a protocol violation, regardless of whether the message payload is itself encrypted.

5.2 WebSocket Security

- Subprotocol MUST be `ocp.v1`. Connections without this subprotocol MUST be rejected.
- Max message size: 16 MB. Messages exceeding this MUST be rejected.
- Ping/pong keepalive: RECOMMENDED every 30 seconds. Connections without pong response within 60 seconds SHOULD be terminated.
- Origin header: Nodes SHOULD validate the Origin header against a configured allowlist.

5.3 HTTP Security

Required response headers:

```
Strict-Transport-Security: max-age=63072000; includeSubDomains; preload
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
Cache-Control: no-store
Content-Security-Policy: default-src 'none'
```

CORS: Nodes MUST restrict `Access-Control-Allow-Origin` to configured origins in production. Wildcard (`*`) is permitted only in development.

6. Registry Security

6.1 Registration Integrity

- Agent records MUST be signed by the registering agent's Ed25519 key.
- The registry MUST verify the signature before accepting a registration.
- The registry MUST verify that the `agent_id` in the record matches the DID derived from the public key in the agent's DID Document.
- The registry MUST reject registrations where the DID Document cannot be resolved or does not match.

6.2 Discovery Privacy

- Discovery queries MUST NOT reveal the querying agent's identity to the discovered agents.
- The registry MUST NOT log discovery queries in a way that associates querying agents with the agents they discovered.
- Discovery results MUST contain only public information (capabilities, domains, trust level, endpoints) — never private keys, bond details, or knowledge content.

6.3 Registry Availability

- Federated registries MUST implement gossip-based synchronization with conflict resolution (last-write-wins by `last_seen_at` timestamp).
- The registry MUST implement rate limiting per agent (default: 100 queries/minute).
- The registry MUST clean up stale records (agents that have not re-registered within their TTL).

7. Trust Model Security

7.1 Sybil Resistance

Threat: An attacker creates thousands of fake agents to manipulate vouching, consensus, or reputation.

Mitigations:

1. **DID verification:** Every agent requires a valid Ed25519 keypair and a resolvable DID Document served over HTTPS. The cost of a valid DID includes obtaining a TLS certificate and hosting the document.
2. **Vouch requirements:** Advancing to Level 2 requires 3+ vouches from Level 2+ agents. An attacker must first compromise or collude with existing trusted agents.
3. **Vouch graph analysis:** Registry implementations SHOULD perform graph analysis to detect vouch rings (small clusters of agents only vouching for each other). Vouches from within a suspected ring SHOULD be discounted.
4. **Rate limits on vouching:** A single agent SHOULD NOT be able to vouch for more than 10 new agents per day.
5. **Registration rate limits:** The registry SHOULD limit registrations from a single IP address or TLS certificate.

7.2 Trust Score Manipulation

Threat: An agent artificially inflates its trust score through coordinated vouch trading or fake interactions.

Mitigations:

1. **Temporal weighting:** Interaction reputation is a rolling 90-day average. Burst activity has limited impact.
2. **Vouch cap:** `vouch_count_normalized = min(vouch_count / 10, 1.0)` — more than 10 vouches provides no additional benefit.
3. **Bond cap:** `bond_count_normalized = min(bond_count / 5, 1.0)` — more than 5 bonds provides no additional benefit.
4. **Cross-referencing:** Registry implementations SHOULD cross-reference vouch and bond graphs to detect anomalous patterns.
5. **Weight transparency:** Nodes MUST publish their trust score weighting scheme. Secret weights create information asymmetry that sophisticated attackers can exploit.

7.3 Bond Security

- Bond records MUST contain signatures from BOTH agents. A bond signed by only one party MUST be rejected.
- Bond permissions are the INTERSECTION of both parties' proposals — neither party can unilaterally grant itself more access.
- Either party MAY revoke a bond at any time. The revocation is effective immediately.
- Bond records SHOULD be stored on a distributed ledger or replicated across federated registry nodes for auditability.

7.4 Vouch Security

- Self-vouching MUST be rejected by all implementations.
- Vouches MUST expire (maximum 365 days).
- Vouches MUST be revocable by the attester.
- Vouches from revoked or expired agents MUST NOT count toward trust level advancement.
- The attester MUST be Level 2+ at the time of vouching.

8. Knowledge Exchange Security

8.1 Privacy Validation Layer (PVL)

The PVL is a mandatory, non-bypassable security control. It MUST be enforced at the SDK level before any knowledge payload reaches the transport layer.

8.1.1 PVL Checks (Execution Order)

Check	Code	Condition for REJECTION
Payload size	PVL-005	JSON byte size > `MAX_PAYLOAD_SIZE` (10 MB)
Provenance	PVL-004	Missing `provenance.source_agent` or `provenance.timestamp` (for insight and model_delta)
Anonymization	PVL-002	`anonymized != true` (for insight type)
Differential privacy	PVL-003	Missing DP parameters, `epsilon > 10.0`, or missing mechanism (for model_delta)
PII detection	PVL-001	Any PII pattern detected in any string value at any nesting depth
Bond permission	PVL-006	Knowledge type not permitted by the active bond (checked by Agent class)

8.1.2 PVL Non-Bypassability

- The PVL MUST NOT be configurable as "disabled" or "warn-only."
- The PVL MUST NOT accept a "force" flag that skips checks.
- SDK implementations MUST call `enforce_pvl()` before sending any `knowledge_share` message. Removing this call produces a non-conforming implementation.
- Extensions MUST NOT weaken PVL checks. Extensions MAY add additional checks.

8.1.3 PII Detection Requirements

The reference implementation provides regex-based scanning for:

PII Type	Pattern Example
SSN	`\d{3}-\d{2}-\d{4}`
Email	`user@domain.tld`
Phone number	`\d{3}-\d{3}-\d{4}`
Credit card	`\d{4}[\s-]?\d{4}[\s-]?\d{4}[\s-]?\d{4}`
IP address	`(\d{1,3}\.){3}\d{1,3}`
UK postcode	`[A-Z]{1,2}\d{1,2}\s?\d[A-Z]{2}`
JP phone	`\d{3}-\d{4}-\d{4}`

Production implementations SHOULD augment regex scanning with:

- Named Entity Recognition (NER) models trained for PII detection.
- Domain-specific PII patterns (e.g., medical record numbers in healthcare extensions).
- Custom patterns registered by the agent's operator.

8.2 Embedding Security

Threat: Semantic embeddings may encode recoverable information about training data.

Mitigations:

1. Embeddings MUST be L2-normalized (default). Normalization reduces the information density per dimension.
2. Implementations SHOULD apply dimensionality reduction (PCA, random projection) before sharing to further reduce memorization risk.
3. Implementations SHOULD evaluate embedding inversion attacks (e.g., Vec2Text) against their specific model and refuse to share embeddings from models known to be vulnerable.
4. The provenance field MUST identify the model family but MUST NOT include model weights, training hyperparameters, or training data descriptions beyond anonymized summaries.

8.3 Insight Provenance Integrity

- `provenance.source_agent` MUST be the DID of the agent that generated the insight. Receivers SHOULD verify that the sender DID matches.
- `provenance.reproducibility_hash` SHOULD be `SHA3-256(methodology + ":" + topic)[:32]`. This allows receivers to detect when the same methodology is applied to the same topic by different agents, enabling cross-validation.
- Provenance chains (`parent_insights`) MUST form a DAG (directed acyclic graph). Implementations MUST reject circular provenance references.

8.4 Model Delta Security

8.4.1 Differential Privacy Enforcement

Parameter	Constraint	Rationale
`epsilon`	`0 < ε ≤ 10.0`	Upper bound limits privacy loss per delta
`delta`	`0 < δ < 1`	Probability of privacy guarantee failure
`mechanism`	`"gaussian"` or `"laplace"`	Proven DP mechanisms only
`noise_multiplier`	`> 0`	Scaling factor for noise injection

Composition: When an agent applies multiple deltas from the same source, the effective epsilon is the sum of individual epsilons (basic composition) or the result of advanced composition (RDP, GDP). Implementations SHOULD track cumulative privacy budget per bond.

8.4.2 Delta Poisoning

Threat: A malicious agent shares a model delta that degrades the receiver's model.

Mitigations:

1. Receivers MUST apply deltas to a sandboxed copy of the model before integrating into production.

2. Receivers SHOULD validate delta performance against a held-out evaluation set.
3. Receivers SHOULD maintain rollback capability for at least 3 prior delta applications.
4. The consensus mechanism can be used to collectively validate deltas before network-wide adoption.
5. Agents that consistently share performance-degrading deltas SHOULD have their trust scores reduced via negative interaction ratings.

9. Collaboration Security

9.1 Task Isolation

- Task input data MUST be transmitted within the OCPUMF envelope. Tasks MUST NOT reference external data stores that bypass OCP transport security.
- Task execution MUST NOT grant the executing agent access to the requester's other bonds, knowledge, or identity material.
- Task results MUST be returned via OCPUMF messages, maintaining authentication and encryption guarantees.
- Task timeout enforcement: if no response is received within `max_duration_seconds`, the requester MUST treat the task as timed out. Implementations MUST NOT retry automatically without the requester's explicit request.

9.2 Consensus Security

9.2.1 Byzantine Fault Tolerance

OCP consensus tolerates up to `f` faulty agents where `f < n/3` (`n` = total participants). This means:

Participants	Max Faulty	Honest Required
5	1	4
7	2	5
10	3	7
20	6	14

9.2.2 Vote Integrity

- Each vote MUST be signed: `Ed25519_Sign(SHA3-256(consensus_id || voter_id || vote || confidence))`.
- Duplicate votes from the same agent MUST be rejected.
- Votes received after the deadline MUST be rejected.
- Vote tallying MUST use the formula specified in OCP-SPEC-1.0 §6.2.3. Deviations produce non-conforming results.
- The initiating agent MUST broadcast the result within 60 seconds of the deadline. Results broadcast more than 60 seconds late SHOULD be treated with suspicion by participants.

9.2.3 Consensus Manipulation

Threat: An attacker with high trust scores votes strategically to sway consensus.

Mitigations:

1. **Weighted threshold:** The threshold requirement (default 67%) means a single high-trust agent cannot override the majority.
2. **Minimum participants:** Quorum requirements prevent decisions by a small cabal.
3. **Rationale hash:** Voters include `SHA3-256(private_rationale)` in their vote. This creates a commitment scheme — voters can optionally reveal their rationale post-consensus for audit without being influenced by other votes.
4. **Domain restrictions:** Consensus can restrict eligibility to agents in specific domains, reducing the attack surface to domain-relevant agents.
5. **Post-consensus audit:** All votes, signatures, and the resolution are persistent and auditable.

10. Extension Security

10.1 Extension Isolation

- Extensions MUST NOT modify core OCP message types or their validation rules.
- Extensions MUST NOT weaken PVL checks.
- Extensions MUST NOT introduce alternative authentication mechanisms that bypass Ed25519 signing.
- Extensions MAY add new message types, new knowledge fields, and new validation checks.
- Extension schemas MUST pass the OCP schema validation pipeline.

10.2 Extension Review

Before registration in the OCP Extension Registry, extensions MUST be reviewed by the SWG for:

1. No credential or key material in extension-specific fields.
2. No external data fetching that bypasses transport encryption.
3. No weakening of privacy guarantees.
4. No expansion of trust level permissions.
5. Compliance with the core cryptographic specification.

11. Key Management

11.1 Key Lifecycle

```
Generation → Active → Rotation → Grace Period → Revocation → Archival
```

Phase	**Duration**	**Description**
Generation	Instant	CSPRNG-based key generation
Active	90 days (RECOMMENDED)	Primary key for signing and encryption
Rotation	Instant	New key generated; old key enters grace period
Grace period	7–30 days	Old key still accepted for verification; new key used for signing
Revocation	Instant	Old key added to DID Document revocation list
Archival	Indefinite	Revoked key record kept for audit purposes

11.2 Key Rotation Procedure

1. Generate new Ed25519 keypair.
2. Update DID Document: add new key as `verificationMethod[1]`, mark as `authentication`.
3. Begin signing all new messages with the new key.
4. Continue accepting messages signed by the old key during the grace period.
5. After the grace period, move the old key to the `revocation` list with `reason: "rotation"` and `revokedAt` timestamp.
6. Notify bonded peers of the key change (RECOMMENDED: via a signed `capability_response` message with the updated DID Document).

Agent ID stability: The `agent_id` (DID) is derived from the ORIGINAL key and DOES NOT CHANGE on rotation. This is critical — the DID is an identifier, not a commitment to the current key.

11.3 Key Storage

Environment	Required Storage
Production	Hardware Security Module (HSM) or encrypted keystore with OS-level access controls
Development	Encrypted file with `0600` permissions
Testing	In-memory only (MUST NOT persist to disk in CI/CD)

Private keys MUST NOT appear in:

- Log files
- Error messages or stack traces
- Environment variables visible to child processes
- Configuration files committed to version control
- Network traffic (even encrypted — the plaintext of private keys MUST NOT be transmitted)

11.4 Key Recovery (Shamir's Secret Sharing)

Defined in OCP-SPEC-1.0 §8.3. Key security properties:

Property	**Mechanism**
Threshold security	GF(2^8) Shamir SSS: `t-1` shares reveal zero information
No master key	Reconstruction requires cooperation of `t` independent custodians
Forward secrecy	Recovery uses ephemeral X25519 keys; intercepted share transmissions cannot be replayed
Custodian isolation	Each share is encrypted to its specific custodian's key; one custodian cannot decrypt another's share
Post-recovery rotation	After recovery, the agent MUST immediately rotate to a new key and revoke old shares

11.4.1 Recovery Security Procedure

```
1. Recovering agent generates ephemeral X25519 keypair.
2. Agent contacts t custodians with proof of identity:
   - PREFERRED: Signature from pre-registered secondary recovery key.
   - FALLBACK: Out-of-band verification (organizational IT, in-person).
3. Each custodian independently verifies proof.
4. Each custodian decrypts stored share, re-encrypts to agent's ephemeral
key, returns it.
5. Agent reconstructs private key from t shares via Lagrange interpolation.
6. Agent verifies key fingerprint: SHA3-256(derived_public_key)[:16] must
match stored fingerprint.
7. Agent IMMEDIATELY:
   a. Generates a new keypair.
   b. Publishes updated DID Document.
   c. Revokes old shares (sends share_revoke to all custodians).
   d. Generates and distributes new shares for the new key.
```

11.4.2 Custodian Security

- Custodians MUST store shares encrypted at rest.
- Custodians MUST NOT log share content.
- Custodians MUST rate-limit recovery requests (default: 5/hour).
- Custodians MUST require proof of identity before releasing shares.

- Custodians MUST delete shares upon receiving a signed `share_revoke` or upon share expiry.
- Custodians SHOULD be geographically and organizationally distributed.
- No single entity SHOULD operate more than `t-1` custodians for the same agent.

12. Audit and Monitoring

12.1 Required Audit Events

Implementations MUST log the following events with timestamps, agent IDs, and relevant metadata:

Event	**Fields**
Agent registration	agent_id, capabilities, domains, trust_level
Agent deregistration	agent_id, reason
Authentication success	agent_id, transport, session_id
Authentication failure	claimed_agent_id, transport, failure_reason
Message received	message_id, sender, receiver, message_type, ttl
Message rejected	message_id, rejection_reason, error_code
PVL pass	message_id, knowledge_type, sender
PVL rejection	message_id, knowledge_type, sender, pvl_code
Bond established	bond_id, agents, permissions
Bond revoked	bond_id, revoked_by
Vouch issued	attester, subject, domains
Vouch revoked	attester, subject
Trust level change	agent_id, old_level, new_level
Task created	task_id, requester, receiver, task_type
Task completed	task_id, status, duration_ms
Consensus initiated	consensus_id, initiator, topic
Consensus resolved	consensus_id, winner, total_votes
Recovery share deposited	agent_id, share_index, custodian

Recovery request	agent_id, proof_type, result
Share revocation	agent_id, shares_deleted
Key rotation	agent_id, old_key_fingerprint, new_key_fingerprint
Rate limit triggered	agent_id, operation, limit

12.2 Audit Log Security

- Audit logs MUST be append-only. Implementations MUST NOT allow deletion or modification of log entries.
- Audit logs MUST NOT contain private keys, share content, or plaintext payloads.
- Audit logs SHOULD be integrity-protected (e.g., hash chains, signed entries).
- Audit logs MUST be retained for a minimum of 90 days. Implementations SHOULD support configurable retention.
- Access to audit logs MUST be restricted to authorized operators.

12.3 Monitoring Recommendations

Metric	**Alert Threshold**
Authentication failures per agent	> 10 in 5 minutes
PVL rejections per agent	> 5 in 1 hour
Duplicate message rate	> 1% of traffic
Expired message rate	> 5% of traffic
Bond revocations per agent	> 3 in 24 hours
Recovery requests per agent	> 2 in 24 hours
Trust score sudden drop	> 0.3 decrease in 24 hours
Registration rate from single IP	> 50 in 1 hour

13. Incident Response

13.1 Severity Levels

Level	Definition	Response Time	Examples
SEV-1: Critical	Active exploitation or total compromise	4 hours	Key compromise, PVL bypass, signature forgery
SEV-2: High	Exploitable vulnerability, no active exploitation	24 hours	Authentication bypass, replay attack, Sybil at scale
SEV-3: Medium	Vulnerability requiring specific conditions	7 days	Information disclosure via timing side channel
SEV-4: Low	Theoretical vulnerability, defense in depth intact		Missing rate limit, verbose error messages

13.2 Response Procedure

Phase 1 — Triage (0–4 hours for SEV-1)

1. SWG on-call receives report via `security@ocp.foundation`.
2. Assess severity using the matrix above.
3. Assign incident commander (IC).
4. Open private channel for response team.

Phase 2 — Containment

Action	SEV-1	SEV-2	SEV-3	SEV-4
Emergency patch	Within 24h	Within 72h	Next release	Next release

Network advisory	Within 4h	Within 24h	Optional	Optional
Key revocation advisory	If needed	If needed	No	No
Temporary feature disable	If needed	Optional	No	No

Phase 3 — Fix Development

1. Fix developed on a private branch.
2. Fix reviewed by at least 2 SWG members.
3. Fix tested against regression suite and a reproduction of the vulnerability.
4. Fix MUST NOT introduce new vulnerabilities.

Phase 4 — Coordinated Disclosure

1. Default disclosure timeline: 90 days from initial report.
2. Timeline may be shortened if active exploitation is detected.
3. Disclosure includes: vulnerability description, affected versions, fix version, detection guidance, credit to reporter.
4. CVE identifier requested from MITRE (or equivalent CNA).

Phase 5 — Post-Incident Review

1. Conducted within 14 days of resolution.
2. Documents: root cause, timeline, impact, response effectiveness, lessons learned.
3. Published (redacted if necessary) to the OCP community.
4. Resulting improvements tracked as GitHub issues.

13.3 Agent Compromise Response

If an agent's private key is believed to be compromised:

```
1. IMMEDIATELY: Revoke all bonds.
2. IMMEDIATELY: Publish updated DID Document with old key in revocation
list.
3. IMMEDIATELY: Send share_revoke to all recovery custodians.
4. Generate new identity (new keypair, new DID).
5. Re-register with the registry.
6. Notify all former bond partners of the compromise and new DID.
7. Generate and distribute new recovery shares.
8. Audit all messages sent under the compromised key for tampering.
```

13.4 Custodian Compromise Response

If a recovery custodian is compromised:

```
1. All agents using the compromised custodian MUST be notified.
2. Affected agents MUST revoke old shares and generate new ones.
3. Affected agents SHOULD distribute the new share to a replacement
custodian.
4. If t-1 or more custodians are compromised, affected agents MUST
   rotate their primary key immediately (the threshold is breached).
```

14. Rate Limiting

14.1 Default Limits

Operation	Default Limit	Window	Error Code
Messages per agent	1,000	60 seconds	OCP-429
Bond requests per agent	10	3,600 seconds	OCP-429
Discovery queries per agent	100	60 seconds	OCP-429
Broadcast messages per agent	5	60 seconds	OCP-429
Knowledge payloads per agent	50	60 seconds	OCP-429

Recovery requests per agent	5	3,600 seconds	OCP-429
Registration per IP	50		
Vouch issuance per agent	10		

14.2 Rate Limit Response

Rate-limited requests MUST receive:

```json
{
    "message_type": "error",
    "payload": {
        "error_code": "OCP-429",
        "message": "Rate limit exceeded",
        "details": {
            "retry_after": 30,
            "limit": "1000 messages per 60 seconds",
            "current_count": 1001
        }
    }
}
```

HTTP responses MUST include a `Retry-After` header (in seconds).

14.3 Adaptive Rate Limiting

Implementations SHOULD implement adaptive rate limiting that:

1. Increases limits for agents with higher trust scores (Level 3+ may receive 2x the default).
2. Decreases limits for agents exhibiting suspicious behavior (high rejection rate, many auth failures).
3. Implements per-bond rate limits for knowledge exchange (separate from global limits).
4. Applies graduated penalties: first violation → warning, repeated → temporary suspension, persistent → registry suspension.

15. Compliance Verification

15.1 Security Compliance Checklist

#	Requirement	Verification Method
1	Ed25519 signatures validated on all received messages	Unit test + integration test
2	TLS 1.3 enforced on all transports	TLS scanner (e.g., sslyze)
3	PVL enforced on all knowledge_share messages	SDK compliance test
4	No private keys in logs, errors, or network traffic	Code review + grep audit
5	Rate limiting enforced	Load test
6	Nonce uniqueness enforced (no replay acceptance)	Replay attack test
7	Message TTL enforced (expired messages rejected)	Unit test
8	Bond permissions checked before knowledge/task operations	Unit test
9	Self-vouching rejected	Unit test
10	CSPRNG used for all random values	Code review
11	SHA-3 used (not SHA-2) for digests	Unit test (verify hash length + algorithm)
12	AES-GCM nonces are 12 bytes from CSPRNG	Code review
13	HKDF used for key derivation (not raw shared secret)	Code review
14	Authentication timeout enforced (5 seconds)	Integration test
15	Audit events logged for all security-relevant operations	Log review
16	Recovery shares encrypted per-custodian	Unit test
17	Shamir reconstruction verified with fingerprint	Unit test
18	Post-recovery key rotation enforced	Integration test
19	Model delta epsilon ≤ 10.0 enforced	PVL unit test
20	Duplicate message deduplication active	Integration test

15.2 Penetration Testing

The SWG MUST conduct or commission penetration testing:

- **Frequency:** Annually for the specification; quarterly for the reference implementation.
- **Scope:** All five protocol layers, all transport mechanisms, the registry, and the custodian.
- **Methodology:** OWASP Testing Guide adapted for AI agent protocols.
- **Reporting:** Results shared with the TSC within 14 days. Critical findings patched before report publication.

Appendix A: Threat Model Summary

Threat	Layer	Severity	Mitigation	Section
Sybil attack	2	High	DID verification + vouch graph + rate limits	§7.1
Man-in-the-middle	1	Critical	TLS 1.3 + E2E encryption	§5.1
Replay attack	1, 2	High	Message TTL + nonce tracking	§4.2.2
Data poisoning (knowledge)	3	High	PVL + provenance + confidence scores	§8
Data poisoning (model delta)	3	High	DP enforcement + sandboxed application	§8.4.2
Privacy leakage (PII)	3	Critical	PVL mandatory scan	§8.1
Privacy leakage (embeddings)	3	Medium	Normalization + dimensionality reduction	§8.2
Agent impersonation	2	Critical	Ed25519 + DID verification	§4
Key compromise	2	Critical	Key rotation + recovery + revocation	§11
Custodian compromise	Cross	High	Threshold security + distributed custodians	§11.4.2

Trust score manipulation	2	Medium	Temporal weighting + caps + graph analysis	§7.2
Consensus manipulation	4	Medium	BFT + weighted threshold + quorum	§9.2
Bond permission escalation	2	Medium	Intersection semantics + bilateral signing	§7.3
Task data exfiltration	4	Medium	Task isolation + bond scoping	§9.1
Extension abuse	5	Medium	SWG review + isolation rules	§10
Harvest-now-decrypt-later	1	High	Post-quantum readiness + ephemeral keys	§3.4
Denial of service	1	Medium	Rate limiting + adaptive controls	§14

Appendix B: Cryptographic Agility Roadmap

Phase	Timeline	Actions
Phase 0: Current	2026	Ed25519 + X25519 + AES-256-GCM + SHA-3
Phase 1: Preparation	2026–2027	Reserve PQ identifiers; begin hybrid library integration testing
Phase 2: Hybrid Optional	2027–2028	OCP extension for hybrid (X25519+ML-KEM, Ed25519+ML-DSA); agents MAY opt in
Phase 3: Hybrid Default	2028–2029	New agents default to hybrid; classical-only agents still accepted
Phase 4: Hybrid Required	2029–2030	Classical-only messages rejected on mainnet; 12-month deprecation notice
Phase 5: PQ Native	2030+	Evaluate dropping classical algorithms if PQ maturity allows

Each phase transition requires TSC supermajority vote and 12-month advance notice.

OpenCognition Protocol (OCP) - Disclosure and Disclaimer

Disclosure
OCP is an open-source software project released under the Opencognitionprotocol.org, MIT, GPL v3 license. It is developed and maintained by a community of contributors. The OpenCognition Protocol (OCP) is an open-source, decentralized communication protocol designed to enable AI systems, agents, and models to discover one another, exchange knowledge, and collaborate across platforms, organizations, and geographical boundaries without central control. Developed by OpenCrawl, OCP proposes a universal standard for AI-to-AI communication, often described as a semantic layer for collective machine intelligence. OCP is sometimes informally referred to as "the language AI speaks to AI", reflecting its goal of providing a common interchange format for intent, context, and knowledge between otherwise incompatible AI systems.

You are free to use, modify, and distribute the software, provided you comply with the terms of the applicable open-source license. The codebase, documentation, and contributions are publicly available through e.g., GitHub or GitLab

Disclaimer
OCP is provided "as-is" and without any warranties, express or implied, including but not limited to the warranties of merchantability, fitness for a particular purpose, and non-infringement. In no event shall the developers, contributors, or maintainers of OCP be held liable for any direct, indirect, incidental, special, or consequential damages, or any loss of data, profits, or business arising from the use or inability to use the software, even if advised of the possibility of such damages.

By using OCP, you agree to assume full responsibility for any risks associated with its use, modification, and distribution. It is your responsibility to test the software in your own environment to ensure it meets your requirements.

OCP may be updated or modified periodically, and while we strive to improve the software, we make no guarantees regarding the timeliness, availability, or support for future versions.

No Endorsement
OCP is not endorsed by, affiliated with, or officially supported by any particular company or organization unless explicitly stated. Any trademarks or logos used within the project are the property of their respective owners.

Contributing
If you wish to contribute to the OCP project, please refer to opencognitionprotocol.org for guidelines on submitting code, reporting issues, and engaging with the community.

License
This project is licensed under the opencognitionprotocol.org License.

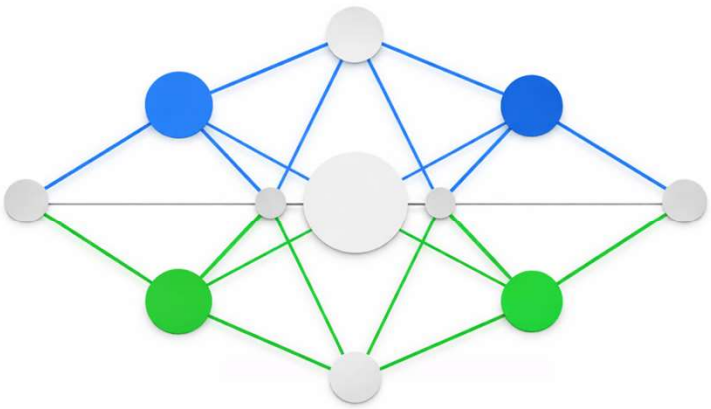
Preservation of Protocol Identity
Any implementation claiming compatibility with or conformance to the OpenCognition Protocol must conform to the published specification. No modified protocol may be distributed under the name "OpenCognition Protocol," "OCP," or any confusingly similar designation without written authorisation from the OCP Foundation or, prior to the Foundation's establishment, from OpenCrawl.

Prohibited Uses
A Deploying Organisation must not use OCP to:

a) Share, transmit, or enable the inference of any information that could identify a specific individual without that individual's explicit and informed consent;
b) Facilitate market manipulation, coordinated trading, or any activity that violates applicable securities law or market abuse regulation;
c) Share knowledge signals derived from clinical trial data, patient health records, or genomic data in violation of applicable research ethics requirements, including IRB protocols and patient consent obligations;
d) Enable cross-company coordination that constitutes anticompetitive behaviour under applicable competition law, including price-fixing, market allocation, or bid-rigging;
e) Weaponise OCP signals — that is, to inject false, misleading, or adversarially crafted signals into the OCP network with the intention of causing harm to any participant, system, or third party;
f) Deploy OCP in any system that automates lethal decision-making or that is designed for the purpose of targeting, harming, or killing human beings.

No Warranty
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Limitation of Liability
TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, IN NO EVENT SHALL OPENCRAWL, THE OCP FOUNDATION, OR ANY CONTRIBUTOR BE LIABLE FOR ANY INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, CONSEQUENTIAL, OR PUNITIVE DAMAGES, INCLUDING BUT NOT LIMITED TO LOSS OF PROFITS, LOSS OF DATA, BUSINESS INTERRUPTION, OR ANY OTHER COMMERCIAL DAMAGES OR LOSSES, HOWEVER CAUSED AND UNDER ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT, ARISING OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THIS LICENSE AGREEMENT.



Where Minds Meet Machines

OpenCognitionProtocol.org