Network Working Group
Internet-Draft
Expires: October 29, 2025

Guy Bary

Intended status: Standards Track
April 27, 2025

Secure Task-bound Agent Message Proof (STAMP) Protocol
draft-guy-bary-stamp-protocol-00

Abstract

The Secure Task-bound Agent Message Proof (STAMP) protocol provides a cryptographically verifiable delegation and proof framework for AI-driven multi-agent systems. STAMP introduces task-bound access tokens and message-level proofs for agentic environments, binding tokens to specific user intents, tasks, and agent identities. STAMP tokens ensure that delegation, execution, and agent-to-agent communication comply with least-privilege and zero-trust security principles. STAMP interoperates with OAuth 2.0, GNAP, and modern proof-of-possession (PoP) techniques, providing stronger security for dynamic, non-deterministic workflows involving autonomous agents.

Status of This Memo

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

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

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

Table of Contents

1. Introduction

The rise of autonomous agents, especially AI-driven multi-agent systems, introduces complex delegation, communication, and authorization challenges. Agents often act independently, call peer agents, and dynamically decompose tasks into sub-tasks across system boundaries.

Traditional delegation protocols such as OAuth 2.0 assume a static, client-server model and do not natively support task binding or dynamic delegation between autonomous entities. STAMP fills this gap by binding tokens not just to clients, but to *tasks*, and requiring proof-of-possession (PoP) for every resource request or agent-to-agent invocation.

STAMP enables secure, verifiable delegation and task management, ensuring agents can operate under strict least-privilege conditions while maintaining a complete audit trail across workflows. In AI systems where actions are non-deterministic, STAMP provides an essential security and governance framework.

2. Terminology

Agent:
: A software entity that performs tasks on behalf of a user or another agent.

Resource Owner (User):
: The human end-user authorizing tasks to be performed.

Authorization Server (AS):
: The entity that issues STAMP tokens to agents after user authorization and consent.

Resource Server (RS):
: The service or agent that validates STAMP tokens and processes secured resource requests.

Task:
: A user-defined or agent-defined unit of work. Each STAMP token is bound to a specific task context via a unique task identifier.

STAMP Token:
: A JWT or CBOR/COSE token that cryptographically binds agent identity, task intent, authorized scopes, and proof-of-possession material.

Proof-of-Possession (PoP) Key:
: An ephemeral asymmetric key pair associated with a STAMP token, used to prove possession during resource requests.

Task Integrity Digest (TID):
: A cryptographic checksum representing the original user request or task definition to detect tampering.

3. Motivation and Use Cases

Modern multi-agent systems exhibit dynamic task generation, subtask delegation, and high-volume agent-to-agent communication. Traditional bearer tokens are inadequate to secure these interactions because:

- Agents may act outside their authorized task scopes if unchecked.
- Tasks may mutate non-deterministically under LLM control.

- Delegation chains may be ambiguous or unverifiable.
- Revocation, replay prevention, and auditability are difficult.

STAMP provides:

- Task-bound authorization with explicit task IDs (TID).
- PoP key binding per token to prevent token theft/replay.
- Verifiable agent-to-agent delegation with audit chains.
- Short-lived tokens for least-privilege and reduced attack window.
- Support for OAuth 2.0 compatibility fallback when needed.

Example real-world applications:

- LLM-driven agent chains for travel planning, calendaring, procurement, where each sub-agent has task-specific rights.
- Enterprise automation where delegated AI assistants interact with financial, CRM, and HR systems under zero-trust policies.
- Multi-stage workflows with sensitive data (e.g., legal review, medical record access) requiring strict task-bound controls.

4. Protocol Overview

STAMP enhances existing delegation protocols by introducing the following primitives:

- **Task Binding**: Every token explicitly carries a unique task identifier (tid) and associated integrity digest (TID).

- **PoP Binding**: Every token carries a confirmation key (cnf.jwk) representing an ephemeral public key for proof-of-possession during each authorized operation.

- **Short- and Long-Task Modes**:
  Tokens can be short-lived (single operation) or support long-lived workflows with refresh or continuation proofs.

- **Delegation Model**:
  Agents can request new subordinate STAMP tokens for peer agents, inheriting the original task ID but with subsetted scopes.

- **Proof Mechanism**:
  All API or agent communication is accompanied by a cryptographic signature using the PoP key over standardized request components, preventing replay and enforcing freshness.

- **OAuth 2.0 / GNAP Compatibility**:
  STAMP tokens can be obtained via OAuth 2.0 Authorization Code Grant flows, Device Code flows, GNAP grants, or other secure grant/consent models.

STAMP defines the lifecycle of a task:

- A user initiates a task.
- The agent requests a STAMP token scoped to that task.
- Each API request proves task continuity.
- Task completion (or cancellation) ends token validity.

5. STAMP Token Format

A STAMP token is a signed JWT (JSON Web Token) or a CBOR/COSE object issued by the Authorization Server (AS). It includes the following claims:

- iss (Issuer):
  The AS that issued the token.

- sub (Subject):
  Identifier of the resource owner (user).

- aud (Audience):
  Intended recipient(s) (e.g., Resource Servers).

- tid (Task ID):
  Unique identifier for the task context. Example: UUID v4.

- agent (Agent ID):
  Identifier of the requesting agent (e.g., client_id, DID).

- scope:
  List of authorized scopes for the agent under this task.

- iat, nbf, exp:
  Issued-at, not-before, and expiration timestamps.

- jti (JWT ID):
  Unique identifier for the token instance.

- cnf (Confirmation Key):
  Public key information (JWK or COSE key) binding the token to the
  agent’s ephemeral PoP key.

- parent (optional):
  Identifier of the parent token if the token was obtained via
  delegation.

- tid_integrity (optional):
  A SHA-256 digest of the original user instruction (Prompt Integrity
  Digest), ensuring that the agent’s actions align with the authorized
  task.

Example (JWT payload):

```
{
  "iss": "https://auth.example.com",
  "sub": "user12345",
  "aud": "https://api.example.com",
  "tid": "550e8400-e29b-41d4-a716-446655440000",
  "agent": "agent-A",
  "scope": "calendar:write email:send",
  "iat": 1700000000,
  "exp": 1700000600,
  "jti": "abc123",
  "cnf": {
    "jwk": {
      "kty": "EC",
      "crv": "P-256",
      "x": "...",
      "y": "..."
    }
  },
  "parent": "xyz7890",
  "tid_integrity": "SHA-256 checksum of task prompt"
}
```

6. Task Handling (Short vs Long Tasks)

6.1 Short Tasks

Short tasks involve immediate or short-lived operations that complete quickly (e.g., a single API call, minor data retrieval).

For short tasks:

- Token lifetimes are very brief (minutes or seconds).
- Tokens are single-use or allow limited reuse.
- Refresh tokens are unnecessary; the agent must request a new STAMP token for a new task.

Best practices:

- Use one STAMP token per short task.
- Include strong timestamps and expiration (‘exp‘ within 5-15 minutes).

6.2 Long Tasks

Long tasks involve workflows spanning multiple steps or prolonged durations (e.g., background job orchestration, model training).

For long tasks:

- Token lifetime may be extended (hours), but short-lived tokens are preferable.
- Agents can refresh tokens by reauthenticating or presenting valid continuation proofs.
- Continuation tokens MUST maintain the same ‘tid‘ and ‘scope‘, with a refreshed ‘exp‘ and new ‘jti‘.
- Session binding mechanisms (e.g., nonce confirmation, re-signing) SHOULD be used for additional security.

Best practices:

- Enforce periodic refresh and re-authorization checkpoints.
- Invalidate all tokens related to a ‘tid‘ if the user revokes consent.

7. Agent-to-Agent Delegation

In dynamic multi-agent systems, an agent (Agent A) may need to delegate a task or subtask to another agent (Agent B). STAMP formalizes this process to maintain verifiable authorization chains without privilege escalation.

7.1 Delegation Process

1. Agent A holds a valid STAMP token (T_A) bound to Task ID (tid_A).
2. Agent A requests a new delegated token (T_B) from the Authorization Server (AS), presenting T_A as proof.
3. The AS verifies T_A, confirms delegation is permitted, and issues T_B:
   - Same ‘tid‘ (or subtask identifier if hierarchical).
   - New ‘agent‘ claim identifying Agent B.
   - Scoped to a subset of Agent A’s original ‘scope‘.
   - Includes ‘parent‘ field linking T_B to T_A’s ‘jti‘.
4. Agent B uses T_B to perform its operations, providing PoP proofs per standard STAMP rules.

7.2 Delegation Constraints

- The AS SHOULD enforce delegation policies such as:
  - Maximum delegation depth.
  - Approved agent identities.
  - Scope narrowing.

- Agents SHOULD NOT forward tokens directly without AS mediation.

- Audit systems MUST record delegation chains (‘parent‘ field).

8. Cryptographic Primitives

8.1 Proof-of-Possession (PoP)

- Agents MUST generate ephemeral ECDSA P-256 key pairs (or equivalent) for each STAMP token.
- The public key is included in the token (‘cnf.jwk‘).
- Each API call or message is signed using the private PoP key.

8.2 Message Signing

- The signature covers:
  - HTTP Method
  - Request URI
  - Timestamp
  - Nonce (optional for replay protection)
  - ‘tid‘ (task identifier)

- Signature methods SHOULD comply with [I-D.ietf-httpbis-message-signatures].

- Example:
  ‘‘‘
  Signature-Input: sig1=("@method" "@path" "tid" "date");alg=ecdsa-p256-sha256
  Signature: sig1=:base64(signature):
  ‘‘‘

8.3 Algorithms

- Mandatory to Implement:
  - ECDSA with SHA-256 (alg=ES256)
  - SHA-256 hashing
- Recommended:
  - EdDSA (Ed25519) where available.
- Deprecated:
  - RSA with SHA-1, HMAC without PoP binding.

8.4 Replay and Freshness

- Include timestamps (‘date‘) and optional nonces in signatures.
- Servers MUST reject requests with stale timestamps or reused nonces.

8.5 Key Management

- PoP keys MUST be rotated per task or per significant token refresh.
- Keys SHOULD be held in memory and cleared after use.

9. Message Flows

9.1 Token Acquisition Flow (via OAuth 2.0)

1. User initiates a task via an agent (Agent A).
2. Agent A requests authorization via OAuth 2.0 Authorization Code Grant or Device Code flow.
3. Authorization Server (AS) authenticates the user and obtains consent.
4. Agent A submits a PoP public key (cnf.jwk) during token request.
5. AS issues a STAMP token bound to:
   - Agent A’s identity
   - Task ID (tid)
   - Scope
   - PoP public key

9.2 Resource Access Flow

1. Agent A prepares an HTTP request to the Resource Server (RS).
2. Agent A signs the request using its PoP private key.
3. Agent A includes:

- Authorization: Stamp <STAMP Token>
- Signature-Input and Signature headers (per HTTP Message Signatures draft)

4. RS verifies:
   - Token validity (issuer, audience, scope, expiry)
   - Proof-of-Possession signature
   - Task ID match
   - Freshness (timestamp, optional nonce)
5. Upon successful validation, RS authorizes the request.

9.3 Delegation Flow

1. Agent A holds a valid STAMP token T_A.
2. Agent A requests a delegated token T_B from AS for Agent B.
3. AS issues T_B:
   - Same Task ID (tid)
   - Narrowed scopes
   - Parent link to T_A (parent field)
4. Agent B uses T_B with its own PoP key to access resources.

9.4 OAuth 2.0 Fallback Mode

- If a Resource Server does not recognize STAMP-specific proof headers, Agent A MAY present the STAMP token as a Bearer token.
- This fallback is discouraged unless absolutely necessary.

10. Comparison to OAuth2 + DPoP and GNAP

Feature	OAuth2 + DPoP	GNAP	STAMP
Token binding (PoP)	Optional	Yes	Mandatory
Task identifier (tid)	No	No	Yes
Agent-to-agent delegation	Limited	Partial	Full Chain
Non-determinism protection	No	No	Yes
Short/long task modes	Manual	Partial	Built-in
OAuth2 compatibility fallback	N/A	N/A	Supported

11. Security Considerations

- Tokens are short-lived and PoP-bound to limit exposure.
- All requests are cryptographically signed for authenticity.
- Task ID (tid) prevents token misuse across unrelated contexts.
- Delegation chains are auditable via ‘parent‘ claims.
- Revocation can occur at token or task level by revoking Task ID.

12. Privacy Considerations

- Tokens should avoid including sensitive user data directly.
- Task ID correlation across services should be minimized if not necessary.
- PoP keys should be ephemeral and task-specific to reduce fingerprinting risks.
- Servers should not expose token contents unnecessarily in errors.

13. IANA Considerations

This document has no actions for IANA at this time.

14. References

[RFC6749] D. Hardt, \"The OAuth 2.0 Authorization Framework\", October 2012.

[RFC7519] M. Jones, \"JSON Web Token (JWT)\", May 2015.

[RFC8705] B. Campbell, J. Bradley, N. Sakimura, \"OAuth 2.0 Mutual-

TLS Client Authentication and Certificate-Bound Access Tokens\",
February 2020.

[RFC9449] D. Fett, J. Bradley, M. Jones, \"OAuth 2.0 Demonstrating
Proof-of-Possession at the Application Layer (DPoP)\", November 2023.

[RFC9635] J. Richer, \"Grant Negotiation and Authorization Protocol
(GNAP)\", March 2024.

[I-D.ietf-httpbis-message-signatures] C. Thomson, \"HTTP Message
Signatures\", IETF Internet-Draft, work in progress.

Author’s Address

Guy Bary
Independent
Email: bgauryy@gmail.com