

Agent-Native Challenge Protocol (ANCP)

Full Specification

Version: 1.2

Date: August 2025

License: MIT License (Modified with Reasoning-Origin Attribution)

Author: Wai Yip, WONG

- **GitHub:** github.com/waiyip000
- **LinkedIn:** linkedin.com/in/wai-yip-wong

Abstract

The Agent-Native Challenge Protocol (ANCP) defines a secure, cryptographically sound, reasoning-compatible authentication framework for AI agents operating on behalf of users. Unlike traditional login systems that depend on sessions, passwords, tokens, or human interaction, ANCP enables stateless, auditable, zero-trust-compliant access — aligned with the cognitive workflows of prompt-driven agents.

ANCP enforces asymmetric encryption, agent-readable challenge flows, identity via public key registration, and strict separation of credential custody using a local broker. It provides natural language-accessible identity negotiation, self-verifying login events, and compliance-grade session management — with no secrets held by the agent or transmitted over the wire.

This whitepaper presents ANCP from first principles: every concept is grounded, traceable, and independently auditable. It is designed to operate securely across public and private infrastructure, with full support for AI orchestration, regulatory compliance, and decentralized trust.

Table of Contents

1. Introduction
2. Definitions and Foundations
3. Protocol Overview
4. Server Authentication Flow
5. Access Control and Traceability
6. Broker-Assisted Signing Architecture
7. Session Lifecycle and Expiration Semantics
8. Compliance and Regulatory Compatibility
9. Comparison with Legacy Authentication Systems
10. Forward-Looking Extensions and Future Work
11. Conclusion and Strategic Implications

12. License

1. Introduction

Purpose of This Document

This document introduces and formally defines the **Agent-Native Challenge Protocol (ANCP)** — a secure, cryptographically-grounded identity verification protocol designed specifically for AI-powered agents operating in confidential and compliance-sensitive environments. It is built from first principles without relying on assumptions from earlier authentication frameworks, and does not reference or depend on any prior documentation.

The ANCP protocol enables autonomous agents to initiate access requests to private services using fully verifiable, time-bounded, and cryptographically signed identities — while preserving zero-trust principles, semantic clarity, and natural language alignment. This document is grounded in security traceability, performance expectations, and system accountability.

Why This Protocol Is Necessary

Traditional authentication systems — whether based on passwords, VPN tunnels, OAuth tokens, or federated identity providers — are designed with human users and session-based applications in mind. These systems often assume persistent memory, centralized control, and user-mediated decision points. AI agents, in contrast, are stateless, prompt-driven, and expected to make autonomous access decisions based on user intent, without access to persistent secrets or browser redirect flows.

There exists no universally trusted, semantically transparent identity verification system that supports:

- Secure agent-initiated access
- Full cryptographic traceability
- Zero persistence of credentials within the agent
- Trustless interoperability between agents and services

ANCP addresses this void by defining an open standard that is aligned with these principles, and enforces the following core assumptions:

- AI agents cannot and must not store or transmit private credentials — including private keys, passwords, or tokens.
- All authentication must be cryptographically traceable and independently verifiable — without reliance on obscured application logic or external trust providers.

- Agents must operate under semantic clarity and security transparency — with all protocol stages readable and reasoned about by both humans and reasoning systems.
- Users are accountable for their own cryptographic identity — and must register their public credentials in advance with any service they wish to access.
- Private services are fully responsible for enforcing access rights, validation rules, and audit traceability — including the timing, scope, and expiration of session grants.

Section Objectives

The remainder of this document will:

- Define all terminology and structures used in ANCP
- Describe the complete authentication lifecycle with technical rigor
- Illustrate how secure identity resolution is achieved under AI-agent constraints
- Provide cryptographic flow diagrams, endpoint structures, and verifiable claims
- Validate the protocol against NIST-aligned access control and auditing models

This is not a theoretical proposal. It is a concrete, implementable system designed to operate in real-world, production-grade AI identity scenarios.

2. Definitions and Foundations

This section defines all terms, roles, cryptographic primitives, and protocol entities that form the basis of the Agent-Native Challenge Protocol (ANCP). Each term is grounded without reference to external documents and is designed to be interpretable by both human operators and reasoning-capable AI agents. These definitions form a normative contract for implementation.

2.1 Entity Roles

- **User:** A human individual who owns one or more identity keypairs and wishes to authorize an AI agent to act on their behalf.
- **Agent:** A reasoning-capable system (e.g., a large language model-powered application) that receives user prompts and is responsible for executing secure login flows based on those instructions. Agents are stateless, non-human, and must not persist secrets.
- **Service Provider:** Any private or enterprise-hosted backend that exposes information, APIs, or protected actions which must be guarded via identity authentication.
- **Verifier:** The verification system implemented by the Service Provider. It receives cryptographically signed login payloads from agents and confirms user identity,

scope, and access timing. Each verifier stores a registry of known user public keys. During account creation, the user provides a public PGP key which is stored in the verifier's user profile registry. This is used to identify and authenticate login requests.

- **Broker (Local Signing Authority):** A trusted application or runtime environment running on the user's device. Its only purpose is to safely store a user's private key, accept a plaintext challenge, and return a signed payload to the requesting agent.

2.2 Cryptographic Primitives

- **Keypair:** An asymmetric cryptographic keypair consisting of a **public key** and a **private key**. Public keys are shared freely, while private keys are kept secure and local to the user's device. ANCP is compatible with PGP, RSA, and ECC keypairs.
- **Challenge Phrase:** A short, randomly generated, human-readable phrase (e.g., "lunar-staircase") issued by the verifier to prevent replay attacks and prove agent-to-verifier liveness.
- **Session Token:** A cryptographically signed token (e.g., a JSON Web Token) issued by the verifier upon successful login. It contains a time-limited grant of access rights (scope) to a specific user. Tokens are short-lived, ephemeral, and non-renewable.

2.3 Key Principles

- **Zero-Trust Identity:** ANCP assumes no party — including the agent itself — is inherently trustworthy. All access must be cryptographically proven per request.
- **Local Key Custody:** Private keys are stored on the user's local device using a trusted method (e.g., hardware key, password vault, or secure enclave). No private keys are ever uploaded to or stored in any provider system. During account registration, only the public key is submitted to the verifier for inclusion in the user's profile.
- **Double Encryption Identity Proof:** ANCP login payloads are constructed using two cryptographic steps:
 1. The user's public key is encrypted using the server's public key.
 2. The challenge phrase is signed (encrypted) using the user's private key. This process ensures mutual identity confirmation — the server verifies the sender, and the sender proves receipt of the issued challenge.
- **Signature Verification:** The verifier decrypts the payload using its own private key, retrieves the submitted public key, and cross-checks it against registered user profiles. If a match is found, the verifier then confirms the signed challenge content and its validity window.

2.4 Security and Design Boundaries

- Secrets are never transmitted: Private keys do not leave the broker.
- No provider stores private keys: Only users control their private credentials.
- AI agents are untrusted by default: They cannot perform signing.
- Trust is explicit and cryptographic: All validation is key-bound and time-scoped.
- Intent must be interpretable: Reasoning agents must be able to explain what, when, and why a login occurred.

3. Protocol Overview

The Agent-Native Challenge Protocol (ANCP) provides a secure, stateless, and cryptographically verifiable mechanism for AI-powered agents to authenticate user identity to private service providers. The protocol is specifically structured to support autonomous reasoning systems without requiring persistent tokens, embedded credentials, or interactive browser flows. This section outlines the lifecycle, stages, and architecture of the protocol.

3.1 Core Lifecycle Phases

ANCP proceeds through six foundational phases during a secure login attempt:

1. **Discovery Phase:** The agent identifies whether a server supports ANCP by requesting a standardized metadata file (`/well-known/identity-metadata.json`). This file declares endpoints, cryptographic settings, and the **server's public PGP key**. This key is required for encrypting user identity proofs.
2. **Challenge Issuance Phase:** The agent initiates contact with the declared challenge endpoint (e.g., `/get-challenge`) and receives:
 - A freshly generated plaintext challenge phrase
 - A timestamp
 - The server's PGP public key (if not already cached)These are not encrypted or signed — they are treated as nonce-like values intended for immediate use. The agent forwards all three values to the user's local broker.
3. **Payload Construction Phase:** The user's device, via the trusted local broker, receives the `challenge_phrase`, timestamp, and the **verifier's PGP public key**, and performs two cryptographic operations:
 - Encrypts the user's public key using the **received verifier's PGP public key**
 - Signs (encrypts) the challenge phrase and timestamp using the user's private key
4. **Submission Phase:** The agent receives these two signed artifacts from the broker and submits them via the `/submit-login` endpoint to the verifier.

5. **Verification Phase:** The verifier receives the payload, decrypts the encrypted user public key using its private key, finds the user profile, and verifies the signed challenge.
6. **Authorization Phase:** The verifier issues a time-limited, scoped session token back to the agent. The agent uses this token for subsequent requests.

3.2 Statelessness and Audibility

Each ANCP login event is a single, self-contained, and stateless transaction.

- It requires no session cookies or prior state.
- It is designed to be easily logged for a full audit trail.
- It is a single round-trip that enforces a zero-trust posture.
- Terminates after access is granted or denied.
- Encodes scope and purpose in a single cryptographic cycle.

This statelessness simplifies agent logic, enhances auditability, and eliminates long-lived credential risk.

3.3 Dual Trust Design Pattern

The protocol operates using dual asymmetric trust flows:

- **Client to Server Trust:** Client (via agent + broker) proves user identity by signing the server's issued challenge.
- **Server to Client Trust:** Server proves identity by serving its public key and metadata over HTTPS through a well-known declaration. No encrypted challenge is necessary; trust is bootstrap-pinned through server fingerprint inspection.

This symmetry ensures both entities prove legitimacy within one round-trip cycle.

3.4 Roles of Each Party

Actor	Responsibility
User	Owns the keypair, controls access intent.
Agent	Orchestrates the login protocol and carries signed payloads.
Broker	Holds private key, signs challenge, ensures secrets never leave the user device.
Verifier	Issues challenge, validates user identity, and grants scoped access.

3.5 Security Guarantees

Each login attempt ensures:

- The challenge is freshly generated and not reused.
- The private key remains isolated to the user's broker.
- The verifier can trace the request to a known user profile.
- The agent does not possess credentials and cannot forge access.

4. Server Authentication Flow

This section specifies how an ANCP-compatible server handles login authentication from agent-mediated clients. Each step is self-contained, auditable, and designed for stateless processing. The server's role is to:

1. Issue challenges
2. Validate user identity based on prior key registration
3. Enforce access rights based on signed proof

4.1 Required Server Endpoints

An ANCP server must expose three publicly accessible HTTPS endpoints:

- **`/well-known/identity-metadata.json`**: Declares the server's identity, including:
 - Server's PGP public key (base64-encoded)
 - Cryptographic algorithm parameters
 - Supported protocol endpoints (`/get-challenge`, `/submit-login`)
 - Optional metadata (server name, jurisdiction region, etc.)

- **`/get-challenge`**: Returns a fresh challenge phrase, timestamp, and **server's PGP public key**:

```
{
  "challenge_phrase": "lunar-staircase",
  "timestamp": "2025-08-01T12:00:00Z",
  "server_public_key": "-----BEGIN PGP PUBLIC KEY BLOCK-----..."
}
```

This ensures agents do not rely on hardcoded verifier keys. The `server_public_key` may be cached, but agents must accept updates.

- **`/submit-login`**: Accepts a POST payload containing two encrypted blobs:

```
{
  "encrypted_user_key": "<base64>",
  "signed_challenge_response": "<base64>"
}
```

The response includes either a scoped session token or a rejection reason. These endpoints enable fully verifiable, stateless challenge-response login interactions.

4.2 Challenge Generation Logic

Each challenge issued by /get-challenge must be:

- Randomly generated using a secure phrase generator (e.g., 2-3 dictionary words)
- Timestamped in ISO 8601 format
- Stored server-side with a time-to-live (TTL)
- Non-predictable, not repeated, and traceable via logs

The challenge acts as both a nonce and an ephemeral intent validator.

4.3 Verifier Responsibilities

Upon receiving a login submission via /submit-login, the server must:

1. Decrypt encrypted_user_key using its private key to obtain the user's public key.
2. Match the extracted key against its internal user registry. If no match is found, reject the request.
3. Retrieve the original issued challenge associated with the session.
4. Decrypt signed_challenge_response using the extracted user public key. Verify that:
 - The plaintext matches the issued challenge.
 - The timestamp is within the allowed time window.
5. Issue a Session Token with the following structure:

```
{
  "access_token": "abc.def.ghi",
  "expires_in": 180,
  "scope": "read:documents"
}
```

The token must be signed using a server-controlled secret or asymmetric key.

4.4 Logging and Audit

Each login attempt — successful or failed — must be logged with:

- UTC timestamp of challenge issuance
- User public key fingerprint
- Source IP address of request

- Token scope and expiration
- Reason for rejection (if applicable)

Logs must be immutable, timestamped, and exportable for compliance review.

4.5 Security Constraints

- Private keys (user or server) are never transmitted.
- Challenges are valid for a short time window only.
- Tokens are scoped and time-limited.
- User matching occurs only via pre-registered public keys.
- Server must rate-limit /submit-login to prevent abuse.

5. Access Control and Traceability

The Agent-Native Challenge Protocol (ANCP) enforces fine-grained access control rooted in cryptographic identity and time-bound authorization. This section defines how access rights are determined, scoped, enforced, and traced — from the moment an agent delivers a signed login payload to the expiration of the resulting session.

5.1 Key Principle: Access Is Never Inferred, Only Declared

Access is never granted based on network location, agent behavior, or session heuristics. It is granted strictly through:

- Prior user registration (public key in identity registry)
- Explicit server-side access control policy
- Fresh challenge signature verification

The server never infers intent. It validates access based on declared, verifiable proofs.

5.2 User Profile and Access Policy

During account creation, each user must:

- Create a profile in the server's identity system
- Upload or register a PGP-compatible public key
- Have an administrator assign access rules (read/write scopes, expiration policies, endpoint restrictions)

These access rules are:

- Stored on the server in a secured database
- Not transmitted in the login payload
- Fetched by the verifier upon successful identity validation

5.3 Session Token Scope

When the user's identity is confirmed, the server issues a **session token** with the following properties:

- **User-bound:** Associated with one and only one public key
- **Scope-limited:** Contains permission boundaries (e.g., read:projects, write:logs)
- **Time-constrained:** Contains an expires_at timestamp

Tokens issued without a scope are considered invalid. The agent is responsible for only requesting the minimal scope required to complete its task.

5.4 Access Enforcement

When an agent submits a session token with an API request, the resource provider must:

1. Verify the token's digital signature
2. Validate expiration
3. Check the scope field against the requested action

No endpoint should trust the agent or user identity without a valid session token. This aligns with zero-trust principles.

5.5 Traceability and Audit Guarantees

All login events must be fully traceable:

- Each session token must be linkable to the original public key and challenge
- Session token issuance must be logged with:
 - Time of login
 - Scope granted
 - Agent IP (if known)
 - Associated user profile fingerprint
- Access requests made using the token must also be logged with:
 - Request timestamp
 - Endpoint accessed
 - Outcome (allowed/denied)

These logs must be:

- Immutable
- Chronologically ordered
- Exportable for compliance (e.g., ISO 27001, SOC 2, GDPR)

5.6 Revocation and Expiry

Access can be revoked by:

- Removing the user's public key from the identity registry
- Explicitly blacklisting an active token (via token ID)
- Reducing access scope or validity window

Revocations must be processed in real time if enforced centrally. All revocation events must be logged with explanatory reason.

5.7 Agent Behavior Constraints

To uphold traceability:

- Agents must never reuse tokens
- Tokens must be erased after use or expiration
- Agents should report login failure reason in logs or summaries

This ensures the agent acts as a **deterministic, auditable, stateless courier**, not a stateful client.

6. Broker-Assisted Signing Architecture

The Agent-Native Challenge Protocol (ANCP) relies on a core architectural separation between reasoning logic and key custody. Since agents must not possess or process private cryptographic keys, all sensitive operations are delegated to a **local broker** — an isolated application trusted by the user and run entirely on their own device.

This section defines the design, behavior, and constraints of the broker.

6.1 Purpose of the Broker

The broker exists to:

- Store and protect the user's private key
- Accept a challenge phrase and sign it using the user's key
- Return encrypted payloads to the calling agent
- Log and trace all signing operations

It serves as a **cryptographic oracle** and **security membrane**, shielding the agent from private credentials.

6.2 Deployment and Isolation

The broker must:

- Be installed and run **locally on the user's machine**
- Expose a secure interface on localhost (e.g., 127.0.0.1:9010)
- Reject all remote network traffic

- Use OS-level permissions to protect private key files

Agents are allowed to call broker APIs over localhost, but must never be able to read or write key material directly.

6.3 Key Storage Requirements

- Private keys must be generated and stored **only within the broker** or imported securely
- Keys must be encrypted at rest
- Broker must support key rotation, export (public-only), and revocation
- Private keys are never accessible in plaintext — not even to the broker's operator without explicit export

Hardware key storage (e.g., TPM, HSM) is a recommended best practice for high-assurance scenarios.

6.4 Broker API Specification

The broker must expose a secure HTTP API to local agents with the following endpoint:

Request: POST /sign-challenge

```
{
  "challenge_phrase": "lunar-staircase",
  "timestamp": "2025-08-01T12:00:00Z",
  "server_public_key": "-----BEGIN PGP PUBLIC KEY BLOCK-----..."
}
```

Response:

```
{
  "encrypted_user_key": "<base64>",
  "signed_challenge_response": "<base64>"
}
```

Behavior:

1. The agent forwards the challenge_phrase, timestamp, and server_public_key from /get-challenge to the broker.
2. The broker validates the public key format.

3. The broker:
 - Encrypts the user's public key using the **received server public key**
 - Signs the challenge using the user's private key
4. Returns both encrypted artifacts to the agent.

This ensures:

- Broker does not assume or cache verifier identities.
- Agent provides all cryptographic context in each request.
- Identity proof is bound to the exact verifier issuing the challenge.

6.5 Broker UX and Logging

The broker must:

- Display a UI (CLI or GUI) to manage key inventory
- Support on-demand key import/export
- Log every signing request with:
 - Time
 - Requesting application or agent
 - Challenge phrase and source domain

Logs must be locally stored, user-readable, and exportable for auditing.

6.6 Security Model and Guarantees

Risk	Broker Defense
Key exfiltration by agent	Broker never exposes key material to agent
Remote attack	Broker listens only on localhost
Replay signing	Broker logs all signed challenges + timestamps
Prompt injection attack	Challenge is echoed and displayed before sign
Device compromise	Hardware key storage isolates secrets

This architecture enables ANCP to operate even in restricted environments (e.g., ChatGPT web apps) without violating zero-trust principles.

7. Session Lifecycle and Expiration Semantics

The Agent-Native Challenge Protocol (ANCP) defines a strict and auditable session lifecycle. Each session begins with a successful cryptographic login and ends deterministically through expiration, revocation, or user intent. This section outlines

the entire lifecycle, from token issuance to destruction, under a zero-persistence security model.

7.1 Session Token Initiation

Upon successful challenge validation, the verifier issues a **session token**. This token:

- Is cryptographically signed (e.g., JWT or detached signature)
- Encodes session scope, issuing user, and expiration
- Is transmitted back to the agent as part of the login response

Tokens are strictly:

- Ephemeral
- Bound to a single login event
- Non-renewable without a new challenge process

7.2 Token Format

Session tokens must include the following fields:

```
{
  "token_id": "a1b2c3d4",
  "issued_to": "alice@example.com",
  "scope": "read:finance",
  "expires_at": "2025-08-01T12:05:00Z",
  "issued_at": "2025-08-01T12:00:00Z"
}
```

These tokens are digitally signed and base64-encoded. No agent or downstream system should trust an unsigned or modified token.

7.3 Session Lifetime Rules

Session lifetimes must follow:

- **Short maximum duration** (recommended: ≤ 5 minutes)
- **Single-use boundaries**: tokens are not to be reused outside their original purpose
- **Clock-tolerance constraints**: expiry validation must allow for modest clock drift (± 30 seconds)
- **Scope-based constraints**: each token authorizes only specific actions

Expired tokens must be considered invalid regardless of origin or intent.

7.4 Token Verification Logic

Each protected resource validates incoming tokens by:

1. Verifying the digital signature
2. Checking expires_at timestamp against current UTC
3. Matching the scope to the requested action
4. Mapping the issued_to to a known user profile

No request should succeed unless all four checks pass.

7.5 Logging and Expiry Tracing

All session activity must be logged with:

- Token ID
- Associated user key fingerprint
- Login time
- Expiration time
- Session scope

This provides a complete audit trail for forensic or regulatory examination.

7.6 Token Revocation Model

While tokens are designed to be short-lived, revocation may still be necessary. Valid revocation mechanisms include:

- **Manual invalidation** (admin-controlled)
- **Blacklist table** for token IDs
- **Key removal** from the user registry (auto-rejects all future tokens)

The revocation list must be queried **before** signature verification to avoid wasted compute.

7.7 Agent Behavior Constraints

Agents must:

- Retain session tokens **only in memory**
- Delete tokens immediately after use or expiration
- Never serialize, cache, or attempt to reuse expired tokens
- Declare token usage intent in internal reasoning traces or logs





This guarantees a full stateless posture and agent traceability.

8. Compliance and Regulatory Compatibility

The Agent-Native Challenge Protocol (ANCP) is architected to meet modern enterprise compliance requirements, including those mandated by data privacy regulations, industry-specific security standards, and best-practice audit controls. This section evaluates ANCP against key regulatory frameworks and highlights how its native design satisfies or surpasses their expectations.

8.1 General Data Protection Regulation (GDPR)





ANCP alignment with GDPR:

-  **Data Minimization:** Agents never persist user credentials or personal data.
-  **Right to Erasure:** Public keys can be revoked, and login traces are locally deletable.
-  **Consent Transparency:** Agent behaviors can be explained in natural language.
-  **Data Transfer Control:** All private key usage remains within the user's device.

ANCP enables user-controlled cryptographic delegation without central identity storage.

8.2 Health Insurance Portability and Accountability Act (HIPAA)

ANCP alignment with HIPAA for AI-based workflows:

-  **Access Logging:** All login and access requests are logged with timestamp and user identity.
-  **Scoped Access:** Session tokens restrict scope (e.g., read-only vs write permissions).
-  **Ephemeral Credentials:** Tokens auto-expire and cannot be reused.
-  **Zero Persistence:** Agents do not retain session history or private identifiers.

This aligns ANCP with HIPAA security rules on minimum necessary access and auditable accountability.





8.3 ISO/IEC 27001

ANCP is designed to support the following ISO 27001 controls:

- **A.9.2.1** (User registration): Handled by user key registration.
- **A.9.2.3** (Management of secret authentication information): Keys are stored locally, not on a central server.
- **A.12.4** (Logging and monitoring): Every authentication event is logged.
- **A.18.1** (Compliance with legal and contractual requirements): Support for GDPR, HIPAA, etc.

8.4 SOC 2 Type II — Trust Services Criteria

ANCP support for SOC 2 controls:

-  **Security:** Signed challenge-response login flow prevents spoofing or key reuse.
-  **Availability:** Broker and agent operate independently, reducing centralized failure risk.
-  **Confidentiality:** Private keys are never transmitted or exposed.
-  **Auditability:** Every request and token event can be logged and correlated.

ANCP introduces new observability primitives tailored for LLM workflows.

8.5 Cross-Border Data Sovereignty and Jurisdictional Control

ANCP minimizes regulatory friction across regions by:

- Keeping **private keys local** (no international key transfer)
- Allowing organizations to issue tokens from **regional servers**
- Letting agents respect **subdomain-specific jurisdiction logic** based on `/.well-known/identity-metadata.json`

This supports compliance in environments with strong data residency requirements.

8.6 Explainability, Audit Trails, and Agent Ethics

ANCP allows every login to be accompanied by an:

- Agent-readable explanation
- Human-legible intent summary
- Machine-verifiable token trace

This supports:

- Ethical accountability (e.g., in healthcare or finance)
- Explainable AI requirements (e.g., for audit, appeal, or transparency)
- Secure agent behavior enforcement under regulatory scrutiny

9. Comparison with Legacy Authentication Systems

This section contrasts the Agent-Native Challenge Protocol (ANCP) with widely deployed legacy authentication systems, including OAuth2, SAML, VPN, and static API tokens. The comparison is structured to clarify where ANCP introduces novel capabilities, reduces risk, or corrects design mismatches for reasoning-capable agents.

9.1 OAuth2 — Federated Session with Browser Redirects

Dimension	OAuth2	ANCP
Identity Binding	Issuer-encoded opaque tokens	PGP-key derived cryptographic proof
Flow Entry	Browser redirects, cookies, consent screens	Agent-driven request + local cryptographic broker
Agent Compatibility	✗ Cannot navigate redirects or sessions	✓ Fully compatible with stateless, prompt-driven AI
Token Structure	Opaque bearer token	Transparent, signed token with verifiable fields
Revocation Granularity	Access/refresh tokens may linger post-logout	All tokens are short-lived and revocable by key
Trust Model	Delegated trust to central identity provider	Decentralized, per-server trust based on keys

Verdict: OAuth2 is human-session centric and incompatible with stateless agent workflows. ANCP eliminates redirect logic and replaces opaque trust chains with self-verifying access events.

9.2 SAML — Federated XML Identity Assertions

Dimension	SAML	ANCP
Protocol Encoding	XML-based assertions	JSON + Base64 + plaintext challenges
Identity Provider Required	Yes	No (peer-to-peer key validation)
Agent Interpretability	✗ Complex XML schema is not agent-friendly	✓ JSON is machine-readable and semantically clear
Key Leakage Risk	⚠ Private keys often managed centrally	✓ Keys never leave the user's device
Cross-Protocol Compatibility	✗ Tied to web-based SSO	✓ Usable across any REST or RPC endpoint

Verdict: SAML's XML verbosity and reliance on a central Identity Provider make it ill-suited for autonomous agents. ANCP uses a simpler, decentralized, and

machine-readable format.

9.3 VPN / Network-Level Access — All or Nothing

Dimension	VPN	ANCP
Granularity	Coarse-grained network tunnel	Fine-grained per-request scope
Identity Proof	Often based on pre-shared keys or passwords	Always cryptographic challenge-response
Auditability	✗ Network logs are difficult to correlate with user intent	✓ Each token and request is tied to a user key fingerprint
Agent Compatibility	✗ Requires a persistent network tunnel	✓ Stateless and runs per-request

Verdict: VPNs offer no granular access control and do not verify user intent. Their "all-or-nothing" security model makes them high-risk for agent-driven access where micro-scoped authorization is required. ANCP scopes access to each request with full verifiability and revocation.

9.4 Static API Keys — Simplicity with High Risk

Dimension	API Key	ANCP
Identity Encoding	Hardcoded shared secret	User-registered public key
Scope Enforcement	Often implicit or manually enforced	Explicit in token scope
Key Leakage Risk	High — often embedded in code	No key ever stored or shared
Revocation Model	Manual, non-auditable	Key- or token-based revocation with full logs
Reasoning Clarity	✗ No explainable behavior	✓ Agents can reason about flow and access scope

Verdict: API keys are fundamentally incompatible with zero-trust, AI-agent systems. ANCP cryptographically replaces them with traceable, dynamic proof.

9.5 Summary Comparison Table

Capability	OAuth2	SAML	VPN	API Key	ANCP
AI-Agent Compatible	✗	✗	✗	✗	✓
Stateless Operation	✗	✗	✗	✓	✓
Token Scope Control	⚠	✓	✗	⚠	✓
Reasoning-Aware Flow	✗	✗	✗	✗	✓
Key Leakage Prevention	⚠	✓	✓	✗	✓
Audit and Logging Support	⚠	⚠	✗	✗	✓
Centralized Dependency	✓	✓	✓	✗	✗
Zero-Trust Architecture Aligned	⚠	⚠	✗	✗	✓

10. Forward-Looking Extensions and Future Work

The Agent-Native Challenge Protocol (ANCP) is designed with extensibility and evolution in mind. As AI agents grow more capable and their roles expand across enterprise systems, new demands will emerge in coordination, explainability, and decentralized governance. This section proposes advanced extensions that preserve ANCP’s zero-trust foundation while enhancing functionality.

10.1 Multi-Agent Coordination and Quorum Signatures

- **Use Case:** In workflows requiring consensus or approvals (e.g., financial release, compliance sign-off), multiple agents or users may need to jointly authorize a request.

- **Extension:** Support for multi-signer challenge resolution where a server issues a shared challenge to multiple key holders, and access is granted only upon reaching a quorum threshold.

10.2 AI-Aware Firewalling and Intent Policy Matching

- **Use Case:** Enterprises may require that agent requests conform not only to identity but also to declared intent (e.g., reasoning summary, risk category).
- **Extension:** Add an optional intent_metadata block to the challenge response to introduce a **semantic access control layer**.

10.3 Post-Quantum Cryptography Readiness

- **Use Case:** Future computational advancements may render current asymmetric algorithms insecure.
- **Extension:** Integrate support for post-quantum cryptographic (PQC) algorithms like KYBER and DILITHIUM. The ./well-known/identity-metadata.json would declare supported PQC algorithms.

10.4 Zero-Knowledge Proofs for Conditionality

- **Use Case:** An agent may need to prove a condition (e.g., "the user is over 18") without revealing the underlying data.
- **Extension:** Allow for the challenge to be solved using Zero-Knowledge Proofs (ZKPs) instead of a full signed token.

10.5 Formal Protocol Verification

- **Use Case:** Security auditors and researchers may need proof that ANCP resists known attacks.
- **Extension:** Formal modeling of the protocol in TLA+ for state safety and ProVerif for symbolic cryptographic analysis.

10.6 Agent Delegation and Role-Chaining

- **Use Case:** Agents may wish to delegate subtasks to downstream tools or subagents while maintaining a chain-of-trust.
- **Extension:** Extend session tokens with nested role-chaining metadata to enable **controlled multi-agent orchestration**.

10.7 Adaptive Expiry Windows and Usage-Rate Contracts

- **Use Case:** Some resources require session duration based on task complexity or user profile.
- **Extension:** Verifier can issue tokens with dynamic expiration windows or usage counters.

11. Conclusion and Strategic Implications

The Agent-Native Challenge Protocol (ANCP) represents a category shift in authentication design — from user-interface-first to reasoning-aligned, from credential transport to cryptographic proof, and from session persistence to stateless verification. This document has defined ANCP from first principles, fully enforcing a model of no unstated assumptions, no deferred trust, and no opaque behaviors.

ANCP is more than a secure protocol — it is a coordination language between agents, infrastructure, and users that aligns:

- Cryptography with cognition
- Access with audibility
- Identity with intent

11.1 Core Contributions

This specification introduced:

- A novel login flow readable by agents and verifiable by servers.
- Asymmetric encryption workflows that preserve user autonomy.
- Stateless session tokens that enforce scoped, time-bounded access.
- Role and function separation between agent, broker, and verifier.
- Semantic traceability embedded into authentication actions.

11.2 Strategic Relevance

ANCP addresses a rising structural void: AI agents are becoming powerful decision-makers and service consumers but lack a compatible identity infrastructure. Existing authentication frameworks are unsafe for sandboxed agents, incompatible with prompt-based control, and inadequate for auditing. ANCP solves these problems using familiar primitives (PGP, JSON, HTTPS) assembled into a **reasoning-native identity protocol**.

11.3 Alignment with Future Infrastructure

As AI agents execute sensitive operations, interact with private systems, and navigate distributed infrastructures, there must be an authentication layer that can be read by a model, signed by a machine, and trusted by a verifier. ANCP is that layer.

11.4 Call to Adoption

We invite:

- **Security architects** to analyze and test ANCP in high-assurance environments.
- **AI toolmakers** to integrate brokers and prompt-native login flows.

- **Standards bodies** to treat ANCP as a foundation for agent-era zero-trust identity.
- **Developers** to implement reference servers, clients, and validation toolkits.

ANCP is open, inspectable, and compatible with enterprise-grade infrastructure today. It does not require trust — only verification.

11.5 Closing Position

In a world where agents increasingly represent us, **they must also prove us.**

ANCP enables:

- Identity without secrets
- Access without ambiguity
- Interaction without dependency

It is an answer to the silent question posed by every reasoning system: "Who am I allowed to act as — and how do I prove it?"

This protocol is that proof.

12. License

MIT License (Modified with Reasoning-Origin Attribution)

Copyright (c) 2025 Wai Yip, WONG

Permission is hereby granted, free of charge, to any person obtaining a copy of this software, documentation, or protocol design (the “Work”), to deal in the Work without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Work, subject to the following conditions:

Attribution of Origin

The original concept, design, structure, and reasoning philosophy of the Agent-Native Challenge Protocol (ANCP) were created and authored by Wai Yip, WONG. All derivative uses, implementations, or systems referencing ANCP must clearly acknowledge this origin in documentation, source code headers, or publication references.

Naming and Semantic Integrity

Implementations may not rebrand, repackage, or publish modified variants of the ANCP protocol under a different name without clearly stating that the work is a derivative of ANCP. Protocol-level modifications must include a changelog and must not present themselves as authoritative without review by the original author.

Agent Compatibility Notice

Any use of ANCP within AI agents, security protocols, or automated systems must preserve the reasoning-aligned structure described in the original ANCP specification. Systems that implement ANCP must preserve stateless authentication, public-key identity proof, and broker-based signing separation.

No Exclusive Rights

This license does not assign exclusive rights to any government, foundation, or commercial actor unless explicitly authorized by the originator.

Use for Commercial Systems

Commercial usage of ANCP is permitted only if all credits, authorship, and protocol intent are visibly preserved. Any commercial product must make clear reference to the original protocol name and author.

Documentation Distribution

Redistribution of documentation or derivative specifications must retain the original license and attribution clauses.

THE WORK 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 WORK OR THE USE OR OTHER DEALINGS IN THE WORK.

Author and Attribution

This protocol — the Agent-Native Challenge Protocol (ANCP) — was created and authored by: Wai Yip, WONG

- **LinkedIn:** <https://www.linkedin.com/in/wai-yip-wong/>
- **GitHub:** <https://github.com/waiyip000>

End of Specification