

# AVA PROTOCOL

AI Visibility Anonymizer

**Gerald Enrique Nelson Mc Kenzie**

*A Technical Specification for Privacy-Preserving AI  
Reversible Tokenization with Audit-Ready Transparency*

**DOI: 10.5281/zenodo.19111004**

Published on PyPI: [pypi.org/project/ava-protocol/](https://pypi.org/project/ava-protocol/)

Version 0.1.0 | March 2026

## Abstract

---

The rapid adoption of Large Language Models (LLMs) in enterprise environments has created an unprecedented privacy challenge: how do organizations leverage AI capabilities while protecting sensitive personal information? Current solutions either block AI adoption entirely or rely on irreversible redaction that destroys data utility.

This paper introduces AVA Protocol (AI Visibility Anonymizer), an open standard for privacy-preserving AI middleware. AVA provides a reversible tokenization system that sanitizes sensitive data before it reaches AI systems, maintains cryptographic audit trails, and enables faithful restoration of original values in AI outputs.

Unlike point solutions that lock users into specific detection engines or cloud providers, AVA defines a protocol abstraction layer that works with any PII detection technology—from Microsoft's Presidio to AWS Macie to custom enterprise classifiers. This engine-agnostic approach ensures organizations maintain sovereignty over their privacy infrastructure while gaining the benefits of standardized interoperability.

The protocol specification includes: (1) a manifest-based audit format with cryptographic integrity, (2) pluggable detection engine adapters with policy-driven sensitivity levels, (3) secure token vault implementations with configurable retention, and (4) client libraries supporting both embedded and gateway deployment modes.

**Key Innovation: Reversible anonymization that preserves both privacy AND data utility**

# 1. Introduction

---

## 1.1 The Privacy-AI Tension

Organizations today face a frustrating dilemma. On one hand, Large Language Models offer transformative capabilities for customer service, document analysis, code generation, and decision support. On the other hand, these same models create massive compliance risks when fed unprotected personal data.

Consider a typical healthcare scenario: A hospital wants to use an LLM to summarize patient records. The records contain names, medical record numbers, addresses, and diagnoses. Sending this to a third-party AI service—even one with SOC-2 certification—violates HIPAA's minimum necessary standard and creates audit nightmares.

Current approaches to this problem fall into three categories:

1. BLOCK: Prohibit AI use entirely for sensitive data. Safe but stifles innovation.
2. REDACT: Permanently remove sensitive values. Safe but destroys context and utility.
3. IGNORE: Hope nothing bad happens. Common but catastrophically risky.

AVA Protocol introduces a fourth approach: TRANSFORM. Replace sensitive values with reversible tokens that preserve context, maintain audit trails, and enable complete restoration when the AI response returns.

## 1.2 Why Existing Solutions Fall Short

The PII detection landscape is fragmented. Microsoft Presidio provides excellent open-source detection but no standard protocol. Cloud providers (AWS Macie, Azure PII, Google DLP) offer powerful services but lock you into their infrastructure. Point solutions like Gretel.ai (acquired by Nvidia for \$320M in March 2025) focus on synthetic data generation rather than reversible anonymization.

What doesn't exist: A vendor-neutral, protocol-based approach that lets you:

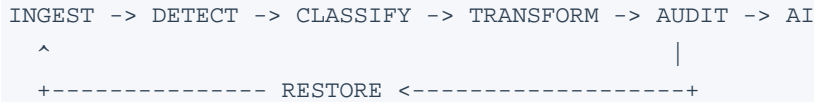
- Swap detection engines without changing your code
- Maintain audit trails in a standard format
- Self-host or use cloud services interchangeably
- Comply with GDPR right to explanation requirements

This gap represents both a technical opportunity and a market opportunity. The LLM privacy middleware market is projected to grow from \$12.4M in 2024 to \$189M by 2034 at 49% CAGR. AVA Protocol positions itself as the interoperability layer that makes this ecosystem work together.

## 2. Core Concepts

### 2.1 The Six-Stage Pipeline

AVA implements a six-stage processing pipeline that operates on every text input:



INGEST: Receive raw text from application

DETECT: Identify sensitive entities using configured engine

CLASSIFY: Map entities to sensitivity levels per policy

TRANSFORM: Replace entities with format-preserving tokens

AUDIT: Generate cryptographically-signed manifest

AI: Send sanitized text to LLM

RESTORE: Swap tokens back to original values in AI output

### 2.2 Token Design Philosophy

AVA tokens follow three principles: visibility, reversibility, and non-malleability.

**VISIBILITY:** Tokens are human-readable and indicate entity type. An email like 'john@example.com' becomes 'AVA\_EMAI\_UfhwZS\_2'-immediately recognizable as an email token without revealing the original.

**REVERSIBILITY:** The token vault maintains a cryptographically secure mapping between tokens and original values. This is not hash-based (which would be irreversible) but rather encrypted storage with automatic expiration.

**NON-MALLEABILITY:** Tokens include embedded checksums. Attempts to modify tokens are detected and rejected during restoration.

The token format is: AVA\_{TYPE}\_{RANDOM}\_{CHECKSUM}

- TYPE: 4-character entity type code (EMAI, PHON, SSN, etc.)
- RANDOM: 8-character base64url-safe random string
- CHECKSUM: Single character integrity verification

**Tokens preserve context: "Contact AVA\_EMAI\_UfhwZS\_2" is readable; "Contact [REDACTED]" is not**

## 3. System Architecture

### 3.1 Protocol Layer Design

AVA uses a layered architecture that separates concerns:

**APPLICATION LAYER:** Your code. Uses AVA client library. No knowledge of underlying detection engines.

**CLIENT LAYER:** Unified interface (`ava.Client`) supporting both embedded and gateway modes. Handles session management, configuration, and error recovery.

**PROTOCOL LAYER:** Core abstractions-`AVAManifest`, `TokenVault`, `DetectionEngine`. These define interfaces, not implementations.

**ADAPTER LAYER:** Pluggable detection engines. Each adapter implements the `DetectionEngine` interface for a specific backend (Presidio, AWS, Azure, etc.).

**ENGINE LAYER:** The actual PII detection technology. AVA does not implement detection-it orchestrates it.

This design means you can start with the `MockEngine` (regex-based, no dependencies), switch to `Presidio` (ML-based, local), then migrate to `AWS Macie` (cloud-based) without changing your application code.

### 3.2 Deployment Modes

AVA supports two deployment modes for different organizational needs:

**EMBEDDED MODE:** Everything runs in your process. The detection engine, token vault, and manifest generator are all local. Best for: air-gapped environments, strict data residency requirements, low-latency applications.

**GATEWAY MODE:** Your application connects to a remote AVA service via HTTP. Detection happens on the server. Best for: centralized policy management, multi-service consistency, reduced client complexity.

The beauty is identical code works for both.

```
# This code works in BOTH modes
import ava

# Embedded mode (local)
client = ava.Client(engine="presidio", vault_type="memory")

# Gateway mode (remote) - same API!
client = ava.Client(
    gateway_url="https://ava.company.com",
    api_key="secret123"
)

# Identical session usage
```

```
with client.session() as session:  
    clean = session.sanitize(text)
```

## 4. Technical Specification

### 4.1 The AVA Manifest Format

Every sanitization operation produces an AVAManifest-a JSON document that serves as a cryptographically verifiable audit trail.

```
{
  "schema_version": "1.0",
  "manifest_id": "ava-7d3f-9a2b-8e1c",
  "session_id": "sess-550e-8400-e29b-41d4",
  "created_at": "2026-03-19T12:34:56Z",
  "policy": "healthcare_strict",
  "detection_engine": "presidio-2.2.361",
  "input_hash": "sha256:a3f5...9c2b",
  "reversibility": true,
  "entities": [
    {
      "type": "PERSON_NAME",
      "original_value_hash": "sha256:7b3f...2d1a",
      "token": "AVA_PERS_xK9mP2nQ",
      "position": [12, 21],
      "confidence": 0.94
    },
    {
      "type": "EMAIL_ADDRESS",
      "original_value_hash": "sha256:9e4c...5f8b",
      "token": "AVA_EMAIL_UfhwZS_2",
      "position": [28, 46],
      "confidence": 0.99
    }
  ]
}
```

### 4.2 Entity Type System

AVA defines a standardized taxonomy of entity types with four-digit codes:

IDENTITY: PERS (person name), DOCU (document ID), SSN\_ (social security)  
 CONTACT: EMAI (email), PHON (phone), ADDR (address)  
 FINANCIAL: CRED (credit card), BANK (bank account), TAX\_ (tax ID)  
 HEALTHCARE: MEDI (medical record), DIAG (diagnosis), PRE\_ (prescription)  
 LOCATION: GPS\_ (coordinates), IPAD (IP address)  
 TEMPORAL: DATE (specific date), TIME (specific time)

Each entity type maps to sensitivity levels (1-5) based on the active policy.

## 5. Implementation Details

---

### 5.1 Token Vault Security

The TokenVault maintains the sensitive token-to-original mapping:

**MEMORY VAULT:** In-process storage using Python dict with threading.Lock. Data never touches disk. Automatic TTL expiration. Best for: ephemeral processing, single-session workloads.

**SQLITE VAULT:** File-based storage with AES-256 encryption for data at rest. Session isolation via foreign keys. Best for: audit requirements, recovery scenarios.

**REDIS VAULT:** Distributed storage with cluster support. Automatic key expiration via Redis TTL. Best for: microservices, load-balanced deployments.

All vault implementations use constant-time comparison for token lookups to prevent timing side-channels.

### 5.2 Session Management

AVA uses context manager-based sessions that guarantee cleanup:

```
with client.session(reversibility=True, ttl=1800) as session:
    clean = session.sanitize(text)
    restored = session.restore(ai_response)
# Token mapping automatically purged on exit
```

The session pattern ensures tokens do not leak between requests-even if exceptions occur, the `__exit__` handler purges the session. This implements the 'zero-retention by default' principle.

### 5.3 Confidence Scoring

Not all detections are equal. AVA surfaces confidence scores from underlying engines:

**HIGH CONFIDENCE (0.9+):** Automatic replacement

**MEDIUM CONFIDENCE (0.7-0.9):** Replace with flag for review

**LOW CONFIDENCE (<0.7):** Include in manifest but do not replace

## 6. Policy Configuration

### 6.1 Domain-Specific Policies

AVA ships with pre-configured policies for common domains:

HEALTHCARE\_STRICT: HIPAA-aligned. All 18 PHI identifiers at maximum sensitivity. No retention beyond session.

FINANCIAL\_PARANOID: PCI-DSS level 1. Credit cards tokenized with one-time-use tokens.

LEGAL\_CONFIDENTIAL: Attorney-client privilege protection. Enhanced logging for privilege review.

GENERAL\_MODERATE: Balanced approach for typical business use. Names and emails protected, dates preserved.

RESEARCH\_ANONYMIZED: Scientific data sharing. Irreversible hashing for true anonymization.

### 6.2 Custom Policy Definition

Policies are YAML/JSON files that specify: entity type sensitivities, confidence thresholds, retention periods.

```
# custom_policy.yaml
name: "enterprise_gdpr"
entity_sensitivity:
  PERS: 5      # Names always protected
  EMAI: 5      # Emails always protected
  PHON: 4      # Phone numbers usually protected
  DATE: 2      # Dates contextual

thresholds:
  min_confidence: 0.85

retention:
  session_ttl: 3600 # 1 hour
```

**Policies are composable: start with HEALTHCARE\_STRICT and override specific settings**

## 7. Use Cases & Security Analysis

---

### 7.1 Real-World Deployment Patterns

#### Pattern 1: AI Customer Service

A bank uses AVA to enable AI-powered chat for customer service. Customer messages with account numbers are sanitized to AVA\_ACCT\_XXXXXXXX before reaching the LLM. The AI responds with "I will check account AVA\_ACCT\_XXXXXXXX" which gets restored to the actual account number before display.

#### Pattern 2: Healthcare Documentation

A medical practice uses AVA to generate patient summaries with GPT-4. Clinical notes with patient names, MRNs, and diagnoses are tokenized. The AI produces a summary mentioning "Patient AVA\_PERS\_XXXX with AVA\_DIAG\_YYYY". Original values are only restored when viewed by authorized clinicians.

#### Pattern 3: Legal Document Review

A law firm processes discovery documents through AVA before AI analysis. Attorney names are protected (privilege) but case citations are preserved (public record). The manifest becomes part of the privilege log.

### 7.2 Security Considerations

Threat: Token vault compromise

Mitigation: Vault data encrypted at rest; memory vaults never touch disk; automatic expiration minimizes exposure.

Threat: Replay attacks using stolen tokens

Mitigation: Tokens are session-scoped; cross-session tokens rejected; checksums detect tampering.

Threat: AI provider model memorization

Mitigation: Unique tokens per session; no stable identifiers across requests.

## 8. Conclusion & Future Work

AVA Protocol represents a new approach to AI privacy-one that does not force a choice between protection and utility. By standardizing reversible tokenization as a protocol rather than a product, we enable an ecosystem where organizations choose their detection engines, deployment modes, and policies while maintaining interoperability.

The PyPI release (ava-protocol) provides production-ready implementations for Python developers. The protocol specification is designed for porting to other languages-Go, Rust, and TypeScript implementations are planned for 2026.

Key takeaways:

- Reversible anonymization preserves both privacy and context
- Protocol abstraction enables vendor independence
- Audit trails satisfy regulatory requirements without friction
- Session-scoped tokens minimize exposure windows

### Future Work

Near-term (2026 Q2-Q3):

- IETF standardization submission for entity type taxonomy
- Go and Rust reference implementations
- Kubernetes operator for gateway deployments

Medium-term (2026 Q4-2027):

- Differential privacy integration
- Federated learning adapter
- Hardware security module (HSM) support

Research directions:

- Homomorphic encryption for computation on encrypted tokens
- Zero-knowledge proofs for manifest verification

**AVA is open source and seeking contributors: [github.com/ava-protocol](https://github.com/ava-protocol)**

### References

- [1] Microsoft. Presidio - Data Protection and De-identification SDK. <https://github.com/microsoft/presidio>
- [2] General Data Protection Regulation (GDPR), Articles 17, 25, and 32. 2016.
- [3] HIPAA Privacy Rule, 45 CFR Part 160 and Subparts A and E of Part 164.
- [4] Nvidia Corporation. Nvidia to Acquire Gretel.ai. Press Release, March 2025.
- [5] NIST. Privacy Framework: A Tool for Improving Privacy through Enterprise Risk Management. 2020.

Author: Gerald Enrique Nelson Mc Kenzie

DOI: 10.5281/zenodo.19111004

PyPI: <https://pypi.org/project/ava-protocol/>