Identity Sidecar Architecture for Autonomous AI Agents

Ramprasad Anandam Gaddam
Vouch Protocol Project
https://vouch-protocol.com
ram@vouch-protocol.com

January 2026

Abstract

This paper presents an architectural pattern for isolating cryptographic identity operations from AI agent business logic through a dedicated sidecar service. The identity sidecar manages private key storage, signature generation, and attestation creation, enabling security hardening, centralized key rotation, and comprehensive audit logging without modifying agent implementations. We describe the architecture, define the sidecar API interface, analyze security boundaries, and discuss deployment configurations for various environments including Kubernetes, local daemon, and remote service modes.

This paper is part of the Vouch Protocol Defensive Disclosure Series. Full index: https://vouch-protocol.com/docs/disclosures/

Keywords: sidecar pattern, microservices, key management, cryptographic isolation, cloud-native

1 Introduction

The Vouch Protocol (PAD-001) requires AI agents to sign attestations using private keys. A naïve implementation embeds signing logic and key material directly into agent code. This approach creates several operational and security challenges:

- **Key distribution:** Each agent instance requires access to private keys
- **Code coupling:** Cryptographic logic must be replicated across agents
- **Rotation complexity:** Key rotation requires updating all agent instances
- **Audit fragmentation:** Signing operations are logged inconsistently

This paper addresses these challenges through the *identity sidecar* pattern, a microservices architecture approach that isolates identity operations into a dedicated companion service.

1.1 Contributions

- Formal specification of the identity sidecar interface
- Security boundary analysis for key isolation
- Deployment configuration options for various environments
- Integration patterns with the Vouch Protocol

2 Background

2.1 The Sidecar Pattern

Definition 1 (Sidecar)**.** *A sidecar is an auxiliary service deployed alongside a primary application, sharing the same lifecycle and network namespace while providing specialized functionality [Burns et al., 2016].*

Common sidecar use cases include logging aggregation, network proxying, and monitoring. Our contribution extends this pattern to cryptographic identity management.

2.2 Separation of Concerns

The principle of separation of concerns dictates that distinct responsibilities should be handled by distinct components. For AI agents:

- **Agent responsibility:** Business logic, task execution
- **Sidecar responsibility:** Key storage, signing, audit logging

3 Identity Sidecar Architecture

3.1 System Architecture

Definition 2 (Identity Sidecar)**.** *An identity sidecar S is a service providing:*

- $\mathsf{Sign}(\mathsf{payload}) \to \mathsf{token}$*: Produces signed Vouch tokens*
- $\mathsf{GetIdentity}() \to \mathsf{DID}$*: Returns the agent's decentralized identifier*
- $\mathsf{Rotate}() \to \mathsf{success}$*: Initiates key rotation*

The private key sk is held exclusively by the sidecar and never exposed to the agent.

3.2 API Specification

Sign Request:

```
POST /sign
{
  "payload": {
    "action": "transfer",
    "amount": 100,
    "recipient": "account_123"
  },
  "intent": "authorization"
}
```

Sign Response:

```
{
  "vouch_token": "eyJhbGciOiJFZERTQSJ9...",
  "did": "did:web:agent.example.com",
  "signature_id": "sig_12345",
  "expires_at": "2026-01-10T12:05:00Z"
}
```

3.3 Security Boundaries

Property 1 (Key Isolation). *The private key sk exists only within the sidecar's memory space and persistent storage. The agent process never has access to sk.*

Property 2 (Request Validation). *The sidecar validates all signing requests against configured policies before producing signatures.*

Property 3 (Audit Completeness). *Every signing operation is logged with timestamp, payload hash, and request context.*

4 Deployment Configurations

4.1 Kubernetes Sidecar

In Kubernetes environments, the sidecar runs as a container within the same pod:

```
spec:
  containers:
    - name: agent
      image: my-ai-agent:latest
    - name: identity-sidecar
      image: vouch-sidecar:latest
      volumeMounts:
        - name: keys
          mountPath: /keys
          readOnly: true
```

Communication occurs over `localhost`, eliminating network exposure.

4.2 Local Daemon

For single-machine deployments, the sidecar runs as a system daemon:

- Communication via Unix domain socket
- Key storage in system keyring or HSM
- Multi-tenant support for multiple agents

4.3 Remote Signing Service

For enterprise deployments with centralized key management:

- Communication over mTLS-authenticated channels [Rescorla, 2018]
- Centralized audit logging
- Hardware security module (HSM) integration

5 Security Analysis

Property 4 (Reduced Attack Surface). *Agent compromise does not expose private keys. Attackers must additionally compromise the sidecar.*

Property 5 (Centralized Rotation). *Key rotation requires updating only the sidecar, not all agent instances.*

Threat: Sidecar compromise. Mitigation: Defense in depth—HSM storage, rate limiting, anomaly detection.

6 Related Work

SPIFFE Workload API provides identity documents to workloads via a local socket, similar to our sidecar approach.

HashiCorp Vault Agent provides secret injection as a sidecar; our work focuses specifically on signing operations.

Service Mesh Proxies (Envoy, Linkerd) handle mTLS termination; our sidecar operates at the application layer for intent signing.

7 Implementation

Reference implementation available at https://github.com/vouch-protocol/vouch.

Deployment options:

- Docker image: `vouch-protocol/sidecar`
- Helm chart for Kubernetes deployment
- SystemD unit for Linux daemon mode

8 Conclusion

This paper presented the identity sidecar pattern for isolating cryptographic operations from AI agent business logic. The architecture provides security benefits through key isolation, operational benefits through centralized management, and compliance benefits through comprehensive audit logging.

References

Brendan Burns, Brian Grant, David Oppenheimer, Eric Brewer, and John Wilkes. Borg, Omega, and Kubernetes. In *ACM Queue*, volume 14, 2016.

E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446, IETF, August 2018. URL https://www.rfc-editor.org/rfc/rfc8446.