atsign

WhitePaper

SECURING THE AGENTIC FUTURE

Using the Atsign Platform atRPC to Build Trusted Agentic AI Systems

How MCP, A2A, and ACP can operate over encrypted channels where API servers never see the data flow

github.com/pembrook-ai/Pembrook ↗

A TECHNICAL PAPER • PEMBROOK PROJECT • MARCH 2026

TABLE OF CONTENT

Using the Atsign Platform and atRPC to Build Trusted Agentic AI Systems

Abstract

As autonomous AI agents proliferate across enterprise workflows, the protocols that govern their communication have become critical infrastructure. The Model Context Protocol (MCP), Agent-to-Agent (A2A), and Agent Communication Protocol (ACP) have each emerged as foundational standards for connecting agents to tools and to one another. Yet all three share a fundamental architectural weakness: they require trusted intermediaries to see the data in transit. API servers, cloud relays, and proxy gateways sit in the path of every message, creating honeypots of sensitive information and single points of compromise.

This paper presents an alternative architecture, implemented in the open-source **Pembrook** project, that layers JSON-RPC 2.0 messaging over **atRPC** on the Atsign Platform. This approach delivers end-to-end encryption, cryptographic identity, and zero open inbound ports while remaining fully compatible with the semantics of MCP, A2A, and ACP. The result is a system where the infrastructure operators themselves, including the Atsign Platform relay, are cryptographically prevented from reading the data that flows through them.

01

Introduction:
The Agentic Protocol Landscape

The year 2025 marked a turning point for agentic AI. What had been a research curiosity, autonomous software agents that plan, reason, and act on behalf of humans, became an enterprise deployment reality. With this shift came an urgent need for standardised communication protocols. Three specifications now dominate the conversation.

Anthropic's Model Context Protocol (MCP)

MCP standardises how large language models connect to external tools, APIs, and data sources. Built on JSON-RPC 2.0, MCP provides a universal adapter that lets any AI application invoke functions, fetch resources, and use predefined prompts from external services through a structured client-server architecture. The protocol supports transport over stdio (for local processes) and Streamable HTTP (for remote servers), and has rapidly become the de facto standard for tool integration, with hundreds of MCP servers available for everything from file system access to database queries.

Google's Agent-to-Agent Protocol (A2A)

A2A addresses the complementary problem of inter-agent collaboration. Introduced in April 2025 (now at version 0.3 under the Linux Foundation, and backed by over 150 organisations), A2A defines how opaque agents—potentially built by different vendors on different frameworks—discover each other's capabilities via Agent Cards, negotiate interaction modalities, and manage task lifecycles. It uses JSON-RPC 2.0 over HTTPS with support for streaming via Server-Sent Events (SSE) and push notifications via webhooks.

IBM's Agent Communication Protocol (ACP)

ACP takes a workflow-oriented approach, providing REST-based conventions for task delegation, stateful sessions, and orchestration across multi-agent systems. Although ACP has since merged its efforts with A2A under the Linux Foundation umbrella, its contributions to stateful session management and observability remain influential in the combined specification.

These three protocols are often described as complementary layers: MCP connects agents to tools, A2A connects agents to other agents, and ACP orchestrates the resulting workflows. Together they promise an interoperable ecosystem where specialised agents collaborate freely. But there is a problem hiding in plain sight.

02

The Trust Problem in Current Agentic Protocols

Every major agentic protocol today relies on a transport architecture where intermediary servers can observe, log, and potentially modify the data in transit. This is not a bug. It is a design assumption inherited from the web's client-server model.

2.1 MCP's exposure model

MCP's remote transport (Streamable HTTP) sends JSON-RPC messages over standard HTTPS connections. The MCP server, which provides tools, resources, and prompts, sees every request and response in cleartext at the application layer. When an AI agent queries a database through an MCP server, the server sees the query, the results, and any sensitive data returned.

The recent addition of OAuth 2.0 authentication addresses who can connect, but not who can see the data once connected. The MCP specification itself acknowledges this gap, stating that security enforcement happens above the protocol level, not within it.

2.2 A2A's visibility assumptions

A2A transmits task messages over HTTPS using JSON-RPC 2.0. Transport-layer encryption (TLS) protects data from network eavesdroppers. However, the server endpoints themselves, specifically the agent platforms hosting the remote agents, have full visibility into the task content, message parts, and generated artefacts.

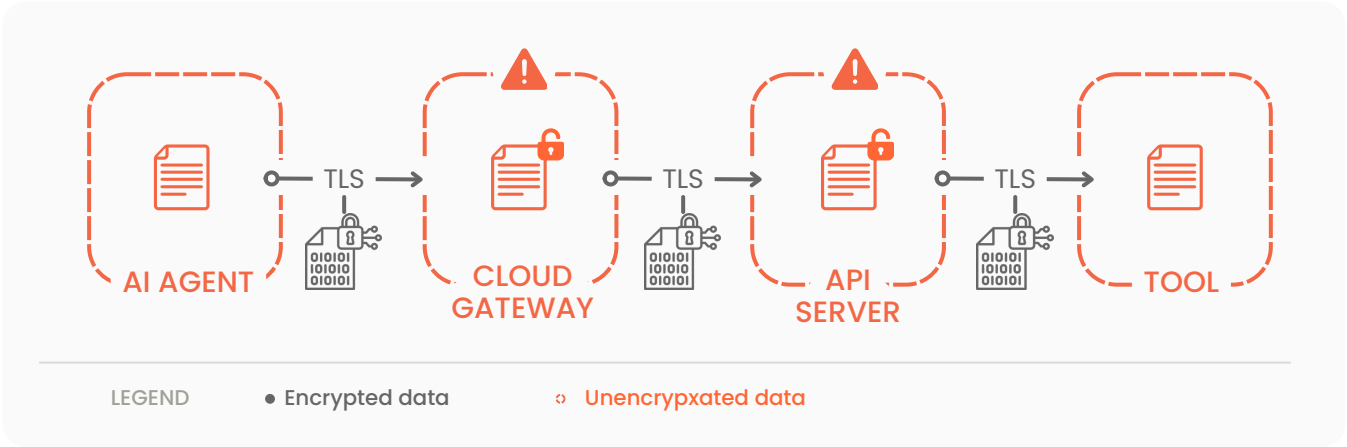
In enterprise deployments where agents operate across organisational boundaries, this means every intermediary platform in the chain can inspect the business data flowing through it.

The protocol supports authentication schemes aligned with OpenAPI (API keys, OAuth 2.0, OpenID Connect), but these verify identity rather than enforcing data confidentiality from the platform operator.

2.3 The cascading trust problem

The real danger emerges when these protocols are composed. Consider a typical enterprise workflow: a user's agent (via A2A) delegates a task to a specialist agent, which invokes several MCP tools, one of which triggers a further A2A call to a third-party supplier's agent. At each hop, a new server operator gains visibility into the data.

> **Each operator adds to the aggregate attack surface. A single compromised intermediary can exfiltrate data from every transaction passing through it.** This is precisely the pattern that the Cloud Security Alliance's Agentic Trust Framework and NIST 800-207 identify as the "**transaction boundary problem**"—implicit trust cascading across hops without explicit verification.



Characteristic	MCP	A2A	atRPC (Pembrook)
Wire format	JSON-RPC 2.0	JSON-RPC 2.0	JSON-RPC 2.0
Transport encryption	TLS (HTTPS)	TLS (HTTPS)	TLS + E2E (AES-256 + RSA)
Server sees cleartext?	Yes	Yes	**No** (zero-knowledge relay)
Open inbound ports	Required	Required	**Zero**
Identity model	OAuth 2.0 / API keys	OpenAPI auth schemes	Cryptographic Atsign (PKAM)
Discovery	Manual config	Agent Cards (/.well-known/)	atDirectory (global, no data)

Table 1: Comparison of agentic protocol security properties.

03

The Atsign Platform: An End-to-End Encrypted Foundation

The Atsign Platform is a networking technology built on the Atsign Protocol, an application-layer protocol operating over TCP/IP. It was originally designed for secure personal data exchange, but its architecture maps remarkably well onto the requirements of agentic AI communication. The platform's core properties are precisely the ones that current agentic protocols lack.

3.1 Cryptographic Identity with Atsigns

Every participant on the Atsign Platform, whether a person, device, or AI agent, is represented by a unique identifier called an Atsign (e.g., `@alice`, `@agent`, `@services`). Each Atsign owns a pair of asymmetric cryptographic keys (RSA 2048 or ECC).

- The **private keys** are generated and stored exclusively on the edge device; they never traverse the network.
- The **public keys** are made globally available through the Atsign Platform so that any participant can verify identity and establish encrypted channels. Authentication uses PKAM (Public Key Authentication Method), which is cryptographic rather than token-based.

Unlike OAuth tokens, which can be stolen and replayed, Atsign authentication requires proof of possession of a private key.

3.2 Zero-knowledge relay architecture

Each Atsign has a corresponding atServer, a personal data service that stores and relays encrypted data.

The critical architectural innovation is that atServers store only encrypted ciphertext and do not possess the decryption keys. Even the operators of the infrastructure—including Atsign themselves—cannot read private data passing through an atServer. Symmetric encryption keys used for data sharing are themselves encrypted with the recipient's public key before being transmitted via the Atsign Protocol. This means that only the intended sender and recipient possess the keys needed to decrypt any given piece of data. The relay is, by design, a zero-knowledge intermediary.

3.3 Zero open inbound ports

All connections in the Atsign Platform are outbound-only. No component—whether an AI agent, a tool server, or a messaging bridge—needs to open any inbound port on the network.

The atDirectory service (root.atsign.org) provides discovery by mapping Atsigns to their atServer's address, but the directory itself holds no data beyond this mapping. This design eliminates the entire class of vulnerabilities associated with exposed network services: port scanning, DDoS attacks, and exploitation of listening services. As a proven example of this architecture in production, Atsign's NoPorts application already leverages this exact principle to allow secure SSH and TCP connections without any open ports. Pembrook applies this same proven paradigm to MCP and agentic protocols.

3.4 atRPC: remote procedure calls over encrypted channels

atRPC is the remote procedure call mechanism built into the atSDK. It uses the Atsign Platform's notification system to deliver RPC requests and responses between Atsigns. Every atRPC message is end-to-end encrypted and cryptographically authenticated before it leaves the sender's device. The receiver's atServer is notified of a pending message, which the receiving client then decrypts locally.

Importantly, atRPC supports an **allowList mechanism**: the server component specifies exactly which Atsigns are permitted to call it. Any request from an Atsign not on the list is silently discarded. This provides a built-in, identity-level firewall that requires no network configuration.

04

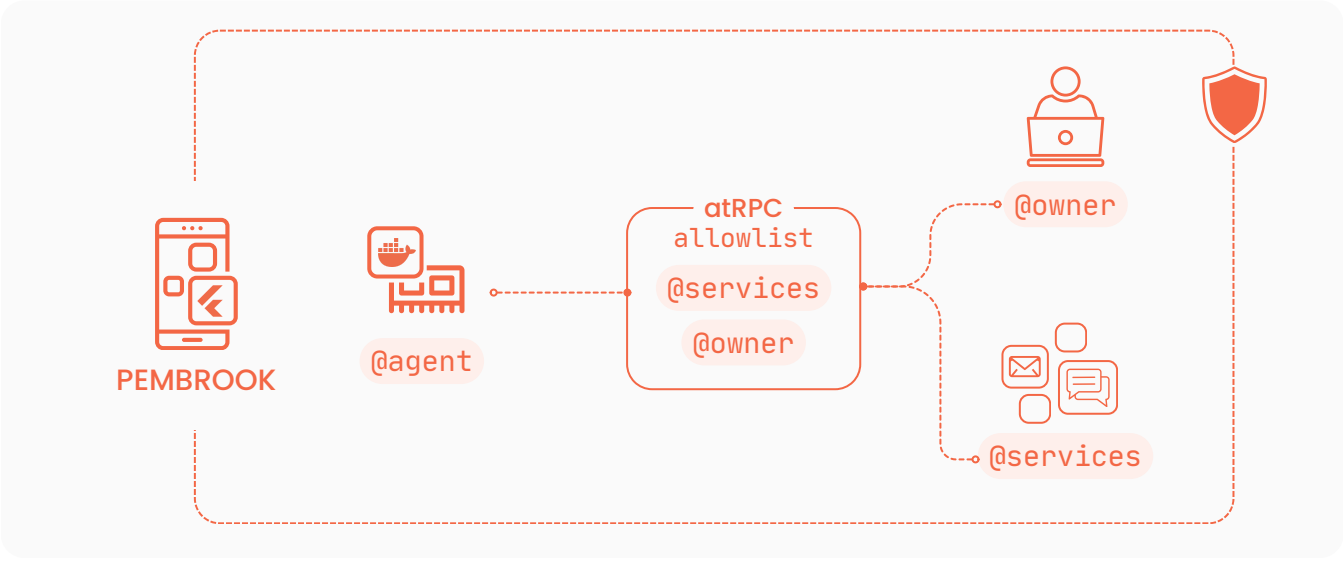
Pembrook:
JSON-RPC 2.0 Over atRPC in Practice

Pembrook is an open-source, privacy-first AI agent platform that demonstrates how to build a complete agentic system on the Atsign Platform. The project implements JSON-RPC 2.0 message semantics, the same wire format used by MCP and A2A, but transports them over atRPC instead of HTTPS.

This substitution is the key insight:
By changing only the transport layer while preserving the message format, Pembrook achieves full end-to-end encryption and zero-knowledge relay without sacrificing protocol compatibility.

4.1 Architecture Overview

The Pembrook system consists of three core Atsigns, each representing a distinct security domain. The `@owner` Atsign represents a person and is used by the Flutter cross-platform application on all their devices. The `@agent` Atsign represents the AI daemon that runs in Docker and manages skills, memory, and audit logs. The `@services` Atsign is shared by all bridges (email, messaging), MCP server wrappers, and skill containers. Communication between these identities flows exclusively through the Atsign Platform's encrypted relay. The agent's Gateway component implements an atRPC server with a strict allowList containing only `@owner` and `@services`. Every incoming request is verified against this list before any processing occurs.



4.2 How a chat message flows

When a message is sent through the Flutter app, the following sequence occurs:

1. The app creates a JSON-RPC 2.0 request containing the command, conversation ID, and timestamp.
2. The atSDK on the device encrypts this payload using the shared symmetric key negotiated with `@agent`, wraps it in an Atsign Protocol notification, and transmits it outbound to the Atsign Platform relay.
3. The relay stores the encrypted ciphertext on `@agent`'s atServer.
4. The agent daemon's Gateway, which maintains a persistent outbound connection to its atServer, receives the notification, decrypts it locally using the agent's private key, and passes the cleartext JSON-RPC request to the Policy Engine.

> At no point in this chain does the Atsign Platform relay—or any other intermediary—see the message content.

4.3 Streaming responses

The agent streams response tokens back to the user by batching approximately 80 characters per chunk and sending each as an atRPC notification. All authenticated `@owner` devices receive these chunks in real-time via Atsign Platform broadcast, enabling multi-device synchronisation. A final sentinel message of done:true signals stream completion.

This streaming model mirrors the Server-Sent Events pattern used by MCP and A2A, but with each event individually encrypted and authenticated.

4.4 Skill sandboxing and tool execution

Pembrook's skill system implements the tool-invocation pattern that MCP standardises, but adds multiple layers of isolation. Each skill runs in a disposable Docker container with strict resource limits (256 MB memory, 0.5 CPU). Skills that do not require network access run with network isolation disabled entirely.

- **The Policy Engine** evaluates every tool invocation against configurable rules, and can escalate sensitive operations (such as sending emails) to human-in-the-loop (HITL) approval.
- **Audit entries** for every skill invocation are stored as immutable AtKeys on the `@owner`'s atServer, meaning the agent cannot delete its own activity logs.

05

Running MCP, A2A, and ACP Semantics Over atRPC

The crucial technical observation is that MCP, A2A, and ACP all use JSON-RPC 2.0 as their wire format. This means their message semantics are transport-agnostic by design. The JSON-RPC specification itself states that the protocol is transport-independent; the message format is the same whether carried over HTTP, WebSocket, stdio, or any other channel. This opens the door to running all three protocol semantics over atRPC without modifying the protocols themselves.

5.1 MCP over atRPC

An MCP server that provides tools (say, a database query tool) can be wrapped so that its JSON-RPC 2.0 requests and responses are transported via atRPC instead of HTTP. The Pembrook project demonstrates this pattern with its mcp_servers/ directory, which contains wrappers for home automation, database, and web browser tools.

Each wrapper has its own Atsign (`@services`) and communicates with the agent via the Atsign Platform relay. The MCP message format is preserved exactly—tools/list, tools/call, and their responses are identical JSON-RPC structures—but the transport is now end-to-end encrypted.

> **The benefits are immediate:**
> The MCP server infrastructure operator never sees the queries or results. Even Atsign's own infrastructure, which relays the messages, sees only encrypted ciphertext. This directly addresses the vulnerability identified by Atsign's own analysis that MCP's current architecture is scaling a broken trust model.

5.2 A2A over atRPC

A2A's agent-to-agent messaging translates naturally to the atRPC model.

In standard A2A, a client agent discovers a remote agent via its Agent Card, then sends task messages over HTTPS.

> **In the atRPC variant**, agent discovery uses the atDirectory (which maps Atsigns to atServers) combined with public metadata stored on each agent's atServer—functionally equivalent to an Agent Card. Task messages (message/send, message/stream) are JSON-RPC 2.0 payloads transported over atRPC. Because atRPC supports both request-response and notification patterns, it accommodates A2A's synchronous, streaming, and asynchronous push models.

The critical difference is that the agent platform hosting the remote agent no longer needs to be trusted with the content. The client and remote agent negotiate a shared encryption key via the Atisgn Protocol's key exchange, and all subsequent task messages are encrypted with that key. The hosting platform routes encrypted traffic but cannot inspect it, achieving the "preserve opacity" principle that A2A espouses, but extending it from application-level opacity to cryptographic data-level opacity.

5.3 ACP orchestration over atRPC

ACP's workflow orchestration semantics—task delegation, stateful sessions, and progress tracking—map to the Pembrook Orchestrator's existing capabilities.

The Orchestrator:

- Manages Multi-step tool chains (up to 10 iterations)
- Maintains conversation state via encrypted AtKeys
- Coordinates with the Policy Engine for authorisation decisions.

These are precisely the workflow primitives that ACP defines. When the orchestrator delegates to a skill or sub-agent, it uses atRPC to deliver JSON-RPC task assignments, receives progress updates via atRPC notifications, and persists session state in encrypted AtKeys on the agent's atServer.

The orchestration layer never needs to expose an HTTP endpoint, maintaining the zero-port-exposure posture.

06

Security Analysis: What Changes and What Doesn't

Moving agentic protocols onto atRPC addresses a specific and well-defined set of threats. It is important to be precise about what it does and does not change.

6.1 Threats eliminated

Data exfiltration by infrastructure operators

In the standard model, every MCP server, A2A agent host, and API gateway can read the cleartext data flowing through it. With atRPC, these intermediaries see only ciphertext. Even a compromised relay yields no usable data.

Network exposure and port scanning

Standard deployments require open inbound ports for HTTP/HTTPS servers. Each open port is a potential entry point. Pembrook's zero-port architecture eliminates this entire attack surface. An nmap scan of a Pembrook deployment returns zero open ports.

Credential theft and replay

OAuth tokens and API keys can be stolen and reused. Atsign authentication requires cryptographic proof of key possession, which cannot be replayed from a different device.

Man-in-the-middle attacks

Even with TLS, a compromised certificate authority or a misconfigured proxy can intercept traffic. The Atsign Protocol's end-to-end encryption is independent of the transport layer's TLS, providing defence in depth.

6.2 Threats That Remain

Endpoint compromise

If the device running the agent is compromised, the attacker gains access to the private keys and can decrypt all messages. Endpoint security remains essential.

Prompt injection and reasoning attacks

The encryption of the transport does not protect against malicious content within the messages. Pembrook addresses this separately through prompt tagging and trust levels, but the Atsign Platform itself does not solve this class of problem.

LLM provider visibility

When the agent routes queries to an external LLM (via the LLM Router's sanitisation pipeline), the external provider sees the sanitised query. Fully private inference requires local models, which Pembrook supports via Ollama.

07

Comparative Architecture: Standard vs atRPC-Based Agentic Systems

Architectural Property	Standard (MCP/A2A/ACP over HTTPS)	atRPC-Based (Pembrook)
Message format	JSON-RPC 2.0	JSON-RPC 2.0 (identical)
Transport encryption	TLS at transport layer	TLS + AES-256/RSA E2E at application layer
Relay/server sees cleartext	Yes — full visibility	No — zero-knowledge relay
Identity model	Tokens, API keys, OAuth	Cryptographic key pairs per Atsign
Open inbound ports required	Yes (HTTP/HTTPS)	Zero (all outbound)
Agent discovery	Agent Cards over HTTPS, manual config	atDirectory + public atServer metadata
Credential theft risk	Token/key replay possible	PKAM requires key possession proof
Audit trail integrity	Server-controlled logs	Immutable AtKeys on owner's atServer
Multi-device sync	Application-level sync required	Built-in via Atsign Platform self-key propagation

Table 2: Architectural comparison between standard HTTPS-based and atRPC-based agentic systems.

08

Practical Implications for Enterprise Deployment

8.1 Zero-configuration networking

One of the most significant practical benefits is the elimination of network configuration as a deployment concern. Standard MCP and A2A deployments require firewall rules, security group configurations, DNS records, SSL certificate management, and often VPN tunnels.

A Pembrook-style deployment requires none of these. The docker-compose file starts the agent, Ollama (for local LLM inference), and any MCP server wrappers, and they begin communicating immediately via the Atsign Platform. This reduces deployment from a multi-team, multi-week infrastructure project to a single-developer, single-day task.

8.2 Compliance and data sovereignty

> For organisations operating under GDPR, HIPAA, or other data protection regulations, the zero-knowledge relay architecture provides a powerful compliance posture.

Because the Atsign Platform relay cannot access the data it transports, it does not need to be classified as a data processor under GDPR. The data controller (the organisation running the agent) retains full control of the encryption keys and thus full control of the data. This simplifies compliance architectures by eliminating the need to audit and contract with every intermediary in the communication chain.

8.3 Cross-organisational agent collaboration

The most compelling use case for A2A-over-atRPC emerges when agents need to collaborate across organisational boundaries.

In the standard model, both organisations must trust every intermediary platform.

> **With atRPC**, Organisation A's agent and Organisation B's agent negotiate a shared encryption key via the Atsign Protocol, and their subsequent task messages are opaque to both organisations' hosting infrastructure. Each organisation maintains its own Atsign-based access policies. This enables secure supply-chain automation, cross-company workflow orchestration, and multi-party AI collaboration without any party needing to trust the other's infrastructure.

09

Future Work and Open Questions

Several areas warrant further investigation.

First, formal standardisation of the atRPC transport binding for MCP and A2A would allow third-party agents and tools to interoperate with Pembrook-style systems without custom integration.

Second, the performance characteristics of atRPC versus direct HTTPS need rigorous benchmarking under production loads, particularly for high-throughput streaming scenarios.

Third, the agent discovery mechanism (currently using atDirectory lookups) could be enriched with A2A-style Agent Cards stored as public atKeys, combining the discoverability of A2A with the cryptographic identity of the Atsign Platform. Finally, the integration with emerging standards such as the Agentic Trust Framework (ATF) from the Cloud Security Alliance and Microsoft's Entra Agent ID merits exploration to ensure that atRPC-based systems can participate in the broader enterprise identity ecosystem.

10
Conclusion

The agentic AI protocols that are being adopted today—MCP, A2A, and ACP—solve real and important problems around tool integration, agent discovery, and workflow orchestration. But they share a fundamental architectural assumption that intermediary servers can be trusted with cleartext data. This assumption is increasingly untenable as agents handle sensitive enterprise data across organisational boundaries.

The Atsign Platform, through the Atsign Protocol and atRPC mechanism, provides a proven, production-ready alternative transport that delivers end-to-end encryption, cryptographic identity, and zero network exposure. Because MCP, A2A, and ACP all use JSON-RPC 2.0 as their message format, layering their semantics over atRPC requires no modification to the protocols themselves, only a change in the underlying transport.

The Pembrook project demonstrates that this is not merely theoretical. A fully functional, privacy-first agentic system—with streaming chat, multi-device sync, skill sandboxing, tool invocation, memory, audit, policy enforcement, and human-in-the-loop approval—can be built entirely on this architecture. The code is open source (BSD 3-Clause) and available at github.com/pembrook-ai/Pembrook.

The question for the agentic AI community is not whether to adopt MCP, A2A, or ACP—these protocols are already achieving critical mass. The question is whether the transport layer beneath them can be upgraded to match the sensitivity of the data they carry. The Atsign Platform offers one compelling answer: keep the protocols, change the plumbing, and make the intermediaries blind.

References

1. Anthropic. "Model Context Protocol Specification." modelcontextprotocol.io/specification, 2025.

2. Atsign. "Atsign Platform Documentation." docs.atsign.com/core, 2025.

3. Atsign. "Atsign Protocol White Paper." atsign.com/resources/white-papers/the-Atsign Protocol, 2024.

4. Atsign. "MCP NoPorts: Securing and Accelerating Private AI Deployments." Press release, June 2025.

5. Cloud Security Alliance. "The Agentic Trust Framework: Zero Trust Governance for AI Agents." February 2026.

6. Google. "Agent2Agent Protocol Specification (Release v0.3)." a2a-protocol.org/latest/specification, 2025.

7. Huang, K., et al. "A Novel Zero-Trust Identity Framework for Agentic AI." arXiv:2505.19301, May 2025.

8. IBM / BeeAI. "Agent Communication Protocol (ACP)." Merged with A2A under the Linux Foundation, 2025.

9. JSON-RPC Working Group. "JSON-RPC 2.0 Specification." jsonrpc.org, 2013.

10. NIST. "Zero Trust Architecture." Special Publication 800-207, 2020.

11. Pembrook Project. github.com/pembrook-ai/Pembrook. 2025–2026.

12. Red Hat. "Zero Trust for Autonomous Agentic AI Systems." Emerging Technologies Blog, February 2026.