WHITE PAPER

Agentic AI Security | Agent-to-Agent Communication

ZipIPS™

Quantum-Secure IoT Authentication with NanoTimestamp Power

Securing AI Agent-to-Agent Communication: Trust Chains in Autonomous Systems

Helene E. Schmidt, Inventor | Creative Synergies LLC

synergies.com | US10171465B2 | US10348729B2

April 2026

Executive Summary

When one AI agent tells another to do something, how does the receiving agent know the instruction is genuine? Not assumed genuine — actually verified, right now, before execution.

This is the trust chain problem. Multi-agent AI systems — where AI models delegate tasks to other AI models, which delegate to others still — create chains of authority that are only as strong as the weakest authentication link. A compromised or hijacked upstream agent can issue fraudulent instructions to every downstream agent and device it commands. Under legacy authentication, those instructions arrive looking legitimate. There is no mechanism to tell the difference.

ZipIPS™ solves this problem at the architectural level. Before any downstream agent or device executes an instruction from an upstream commanding agent, it can issue a fresh timestamp challenge. The commanding agent must respond with a credential derived from pre-shared tables that never cross the network. A hijacked or substituted agent cannot produce what it doesn't have. The instruction is blocked. The attempt is logged.

This White Paper examines the agent-to-agent trust chain problem in detail, explains how ZipIPS™ addresses it, and identifies the specific deployment scenarios where authenticated agent communication is not optional — it is essential.

Grok 4 (xAI) first determined that ZipIPS™ exceeds current NIST post-quantum and lightweight cryptography standards. Claude (Anthropic) subsequently reached the same conclusions independently — two separate analyses, the same result. ZipIPS™ is available for licensing. Creative Synergies LLC welcomes inquiries.

1. The Trust Chain Problem

1.1 How Multi-Agent Systems Work

A multi-agent AI system is one where AI models collaborate — one agent breaks a goal into subtasks, delegates those subtasks to specialized agents, and those agents may delegate further. The result is a chain: a user or system issues a goal to a coordinating agent, which issues instructions to subordinate agents, which issue commands to devices or APIs.

This architecture is powerful. It allows complex tasks to be parallelized, specialized, and automated in ways that a single AI model cannot match. It is also becoming the standard pattern for enterprise AI deployment — in manufacturing automation, financial operations, defense systems, and cloud infrastructure.

The authentication problem this creates is structural: every link in the chain assumes the instruction it received came from a legitimate source. But how is that verified?

1.2 Why the Chain Breaks

Under legacy authentication, agent identity is typically established at session initiation — an API key, a session token, a certificate. Once established, instructions from that agent are trusted for the duration of the session. The assumption is that the authenticated identity persists.

That assumption fails in several important ways:

- Prompt injection: A malicious input causes an agent to issue unauthorized instructions to downstream agents. The downstream agents see a legitimate-looking instruction from an authenticated source — they have no way to detect the injection.
- Session hijacking: An attacker gains control of an agent mid-session. All subsequent instructions from that agent appear authenticated because the session credential is still valid.
- Agent substitution: A man-in-the-middle attack replaces a legitimate upstream agent with a fraudulent one. Downstream agents and devices continue accepting instructions because the substitution happened after authentication.
- Compromised orchestrator: The coordinating agent at the top of the chain is compromised. Every agent and device it commands is now receiving fraudulent instructions from what appears to be an authoritative source.

> *In a trust chain, the vulnerability is not at the perimeter. It is at every link. An agent that cannot verify the source of its instructions before executing them is a liability — regardless of how well the session was authenticated at the start.*

1.3 The Quantum Dimension

Most agent-to-agent authentication today relies on TLS, JWT tokens, OAuth, or certificate-based PKI. All of these depend on mathematical hardness assumptions that quantum computing threatens. An adversary with a sufficiently powerful quantum computer can break the cryptographic foundations of these systems.

Multi-agent AI systems being deployed today may still be running when quantum computers capable of these attacks are operational. Authentication architecture chosen now determines exposure then. ZipIPS™ does not rely on any quantum-vulnerable mathematical structure — it is quantum-secure by architecture, not by parameter choice.

2. How ZipIPS™ Secures Agent-to-Agent Communication

2.1 Command-Level Challenge

The key insight is this: session-level authentication is not enough. What matters is whether this specific instruction, arriving right now, actually came from an authorized source.

ZipIPS™ enables command-level challenge. Before executing any new instruction from an upstream agent, the receiving agent generates a fresh timestamp and sends it as a challenge. The commanding agent must respond with a credential derived from its pre-shared tables — tables that were provisioned before the session began and never cross the network during operation. The receiving agent independently derives the expected credential from its own copy of the same tables and compares.

If the credentials match — the instruction proceeds. If they don't — the instruction is blocked and the attempt is permanently logged. A hijacked, substituted, or prompt-injected agent cannot produce what it doesn't have.

2.2 The Double Two-Way Handshake in Agent Chains

ZipIPS™ authentication works in both directions. The downstream agent verifies the upstream commanding agent before executing its instruction. The upstream agent also verifies the downstream agent before trusting its responses. Neither side assumes the other is legitimate — both sides prove themselves, independently, at every exchange.

In a multi-agent chain, this means:

- The coordinating agent verifies each subordinate agent before accepting its output
- Each subordinate agent verifies the coordinating agent before executing its instruction
- Each device at the end of the chain verifies the agent commanding it before executing any action

Every link in the chain is independently authenticated. A compromise at any one link does not propagate silently — it fails at the next verification point and is logged.

2.3 Key Properties for Agent-to-Agent Security

Property	Why It Matters for Agent-to-Agent Communication
No static credential	API keys and session tokens shared between agents are a standing vulnerability. ZipIPS™ generates a fresh credential at every command exchange — there is nothing persistent to steal or replay.
No transmitted sequence	The order in which credential strings are assembled is derived independently by both agents from their pre-shared tables. An intercepted credential reveals nothing about the next one — and nothing about the tables that produced it.
Command-level challenge	Any agent can challenge the commanding source before executing a new instruction. Session-level authentication is not sufficient — ZipIPS™ authenticates at the command level, where the actual risk lives.
Permanent audit log	Every authentication attempt — successful or failed — is permanently logged and available for review. In a multi-agent system, the audit trail covers every link in the chain, providing a verifiable record of which agent authorized which action.
Quantum-secure by construction	Agent-to-agent communication secured with ZipIPS™ is not vulnerable to quantum computing attacks. No mathematical structure to attack, no key to factor, no algorithm to break.
On-demand generation only	Credentials are generated only when a command challenge occurs. No background process, no idle resource consumption — the system scales to any frequency of agent interaction without overhead.

3. Attack Scenarios and ZipIPS™ Response

The following scenarios illustrate how specific agent-to-agent attack vectors are defeated by ZipIPS™ authentication.

Attack Vector	How It Works	ZipIPS™ Response
Prompt injection	A malicious input causes an upstream agent to issue unauthorized instructions to downstream agents. The downstream agents receive instructions that appear to come from a legitimate authenticated source.	Before executing the injected instruction, the downstream agent challenges the upstream source with a fresh timestamp. The upstream agent — now issuing fraudulent instructions — still holds valid credentials and passes the challenge. This is why command-level provisioning and table rotation policies matter — a compromised agent with valid tables remains a risk until tables are rotated.
Agent substitution (MitM)	An attacker replaces a legitimate upstream agent with a fraudulent one mid-session. Downstream agents continue receiving instructions that appear authenticated.	The substituted agent does not have the pre-shared tables of the legitimate agent. When challenged with a fresh timestamp, it cannot produce the correct credential. The instruction is blocked. The attempt is logged. The substitution is detected at the first challenge.
Session hijacking	An attacker gains control of an agent's session credentials. All subsequent instructions appear to come from the authenticated agent.	Session credentials alone are not sufficient under ZipIPS™. Each command requires a fresh timestamp challenge. A hijacked session without the pre-shared tables cannot pass command-level verification.
Compromised orchestrator	The top-level coordinating agent is compromised. Every downstream agent and device receives fraudulent instructions from what appears to be the authoritative source.	Downstream agents challenge the orchestrator before executing each instruction. A compromised orchestrator that has lost its pre-shared tables — through table rotation or revocation — cannot pass verification. This makes table rotation policy the critical governance control.
Replay attack	An attacker captures a valid credential from a previous agent exchange and attempts to reuse it to authorize a fraudulent instruction.	Every credential is tied to a single, unrepeatable timestamp. A captured credential is already expired. There is nothing to replay.

4. Deployment Scenarios

4.1 Enterprise AI Automation

Enterprise AI systems increasingly use multi-agent architectures to automate complex workflows: data retrieval, analysis, report generation, system configuration, and decision support all handled by chains of specialized agents. Each handoff between agents is an authentication event that legacy systems leave unverified.

ZipIPS™ brings command-level authentication to every handoff in an enterprise agent chain. An agent receiving a data retrieval instruction from an orchestrator can verify the instruction before executing it. An agent receiving analysis results from a subordinate can verify the source before incorporating those results into its output. The entire chain is auditable — every instruction, every response, every verification, permanently logged.

4.2 Financial AI and Algorithmic Trading

Financial AI systems operate at speeds and frequencies that make human supervision impossible in real time. An agent chain managing a trading portfolio may execute thousands of instructions per hour — each one potentially moving significant capital. A compromised agent in that chain, issuing fraudulent buy or sell instructions, can cause substantial financial damage before any monitoring system detects the anomaly.

ZipIPS™ authenticates each trading instruction at the command level. A fraudulent instruction from a compromised agent cannot pass the timestamp challenge. The instruction is blocked before execution. The attempt is logged with a timestamp that provides regulators and compliance teams with a precise, verifiable record.

4.3 AI-Controlled Infrastructure

Industrial control systems, energy grid management, and smart city infrastructure are increasingly managed by AI agent networks. An agent chain controlling a power distribution system, a water treatment facility, or a manufacturing production line has the ability to cause physical consequences — not just data consequences — if compromised.

The stakes make command-level authentication essential, not optional. Before any agent in an infrastructure control chain executes an instruction that affects physical systems, ZipIPS™ verification ensures the instruction came from an authorized source. A substituted or hijacked commanding agent cannot pass the challenge. The command does not execute.

4.4 Defense and Intelligence AI

Defense AI systems operate in adversarial environments where nation-state level attackers may have sophisticated capabilities — including quantum computing in the near future. An agent chain managing intelligence analysis, logistics, or command and control must be authenticated with a system that remains secure against those capabilities.

ZipIPS™ is quantum-secure by construction. Defense AI agent chains secured with ZipIPS™ are not vulnerable to the quantum attacks that will eventually break RSA, ECC, and the other mathematical foundations of legacy PKI. The architecture that secures agent communication today remains secure as the quantum threat matures.

5. Security Performance

ZipIPS™ is a licensable architecture. The following illustrative entropy levels represent what the architecture can achieve at different implementation depths. There is no upper limit — licensees can increase entropy by adding character sets, time units, or any real-time data their systems can generate.

Implementation	Entropy	Payload	Characteristics
ms original	381 bits	95 bytes	Millisecond-resolution timestamps. Alphanumeric character strings. Eight time units. Strong baseline for high-frequency agent-to-agent authentication.
ns standard	476 bits	117 bytes	Nanosecond-resolution timestamps. Alphanumeric character strings. Ten time units. Appropriate for sensitive agent chains handling financial, industrial, or classified data.
ns+ high security	735 bits	157 bytes	Maximum time-unit depth with non-control special characters from the extended ASCII character set. Designed for defense, intelligence, and critical infrastructure agent systems.

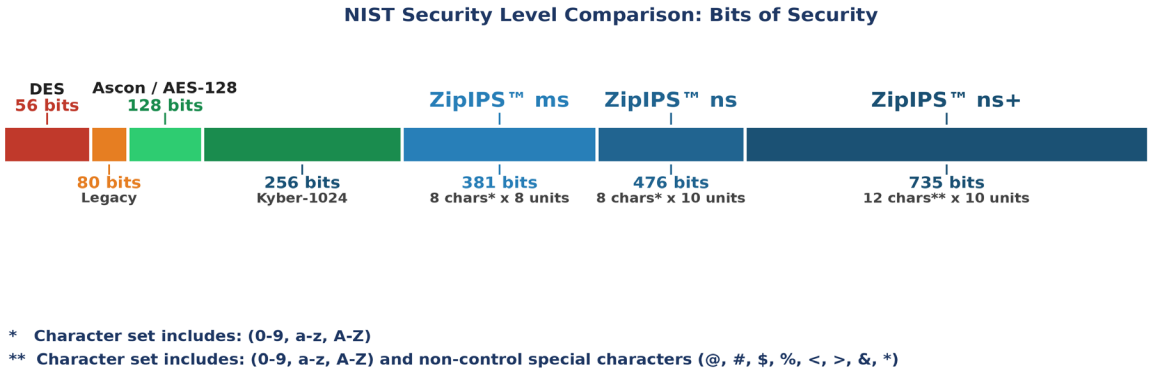

Figure 1: NIST Security Level Comparison — ZipIPS™ illustrative implementations vs. reference points

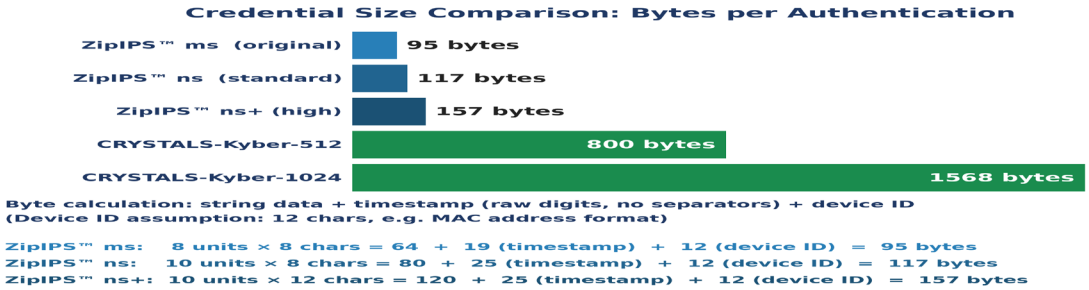

Figure 2: Credential Size Comparison — ZipIPS™ illustrative implementations vs. CRYSTALS-Kyber

At nanosecond precision, ZipIPS™ generates up to 476-bit credentials in a 117-byte payload — more than three times the entropy of AES-256, generated fresh at every command exchange, with no transmitted key and no quantum vulnerability. A multi-agent system authenticating thousands of times per hour carries no meaningful overhead.

6. Implementation Considerations

ZipIPS™ is a licensable architecture — not a product. Licensees write their own implementation code. For agent-to-agent communication specifically, two implementation decisions are critical:

- Command-level challenge integration: The most effective implementation builds the timestamp challenge into the agent's instruction-processing logic — before any received instruction is executed, the agent challenges the source. This is a licensee implementation decision that determines whether protection operates at the session level (insufficient) or the command level (effective).
- Table provisioning and rotation policy: Pre-shared tables are provisioned to each agent at deployment. The security of the system depends on the integrity of that provisioning process and on the licensee's policy for rotating tables. A compromised agent retains valid tables until they are rotated — table rotation is the primary governance control for a compromised agent scenario.

Implementation requirements will vary by environment and are the licensee's responsibility to evaluate. Creative Synergies LLC makes no representation regarding specific integration requirements for any particular system.

7. Patent Foundation

ZipIPS™ is protected by two issued United States patents, both assigned to Helene E. Schmidt of Creative Synergies LLC.

Patent	Issued	Scope
US10171465B2	Jan. 1, 2019	Method patent covering the authentication process: timestamp generation, character string retrieval and sequencing, initiating string construction and transmission, host-side verification, second timestamp generation, client-side verification, and the on-demand re-verification loop. Applicable to any networked device pair, including AI agent-to-agent communication.
US10348729B2	Jul. 9, 2019	System patent covering the authentication architecture: host device with sequence tables and string tables at variable security levels, client device with mirrored table architecture, device identifier, and the system configuration for the complete double two-way handshake with on-demand re-verification. Continuation of US10171465B2.

8. Licensing

Creative Synergies LLC welcomes inquiries from qualified organizations interested in exploring licensing opportunities. No terms are presented in this White Paper — the right licensing structure is best arrived at through direct dialogue between technically and legally informed parties.

ZipIPS™ is available for licensing. Please visit synergies.com and Contact Us if you have any questions. Helene E. Schmidt, Inventor Creative Synergies LLC synergies.com | zipips@synergies.com