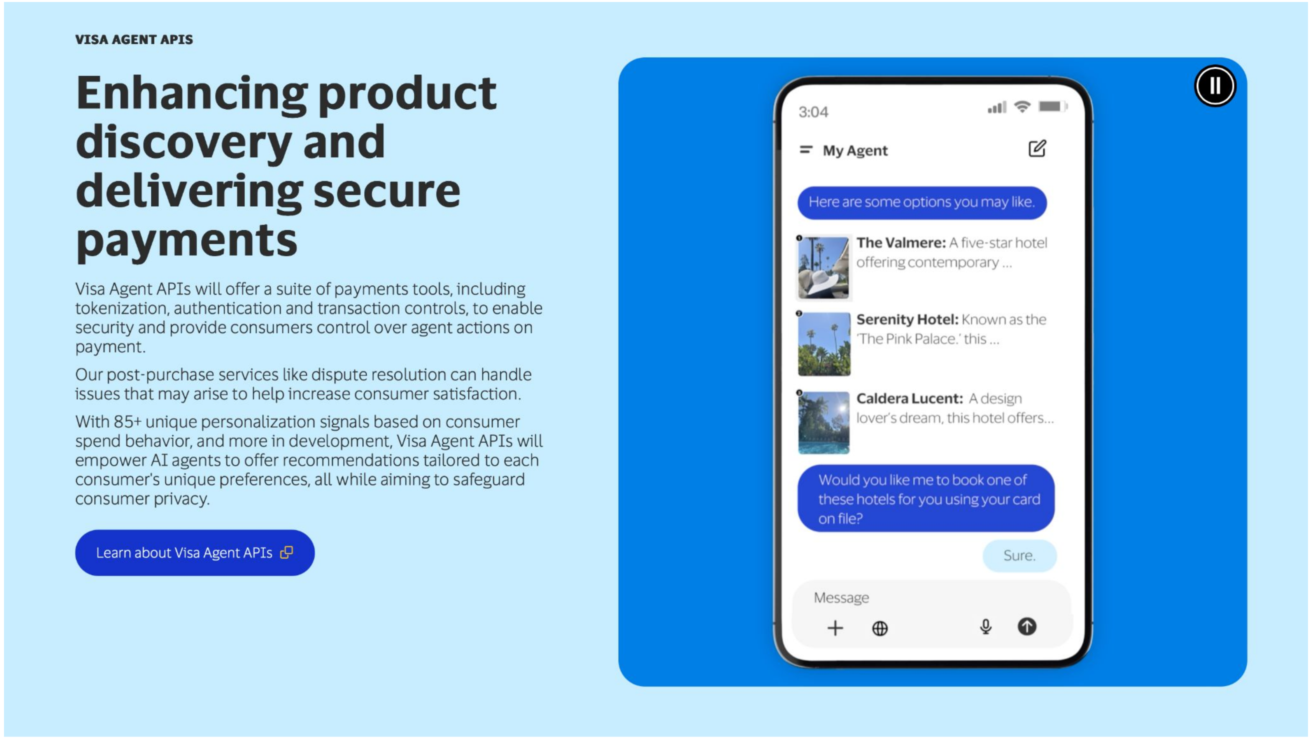
The "T" in MCP and A2A Stands for Trust

Wenjing Chu
GOSIM AI Paris - May 6, 2025

Some intro…

- AI and Human Trust
- Governing Board and TAC at OpenWallet
- Chair of the AI and Human Trust (AIM) Working Group at Trust over IP (ToIP)
- Author of the Trust Spanning Protocol (TSP) Specification & lead of the open source project
- This talk is about how to build AI agents we can trust…

From LLMs to Agents …



VISA AGENT APIS

Enhancing product discovery and delivering secure payments

Visa Agent APIs will offer a suite of payments tools, including tokenization, authentication and transaction controls, to enable security and provide consumers control over agent actions on payment.

Our post-purchase services like dispute resolution can handle issues that may arise to help increase consumer satisfaction.

With 85+ unique personalization signals based on consumer spend behavior, and more in development, Visa Agent APIs will empower AI agents to offer recommendations tailored to each consumer's unique preferences, all while aiming to safeguard consumer privacy.

Learn about Visa Agent APIs

3:04

My Agent

Here are some options you may like.

The Valmere: A five-star hotel offering contemporary ...

Serenity Hotel: Known as the 'The Pink Palace.' this ...

Caldera Lucent: A design lover's dream, this hotel offers...

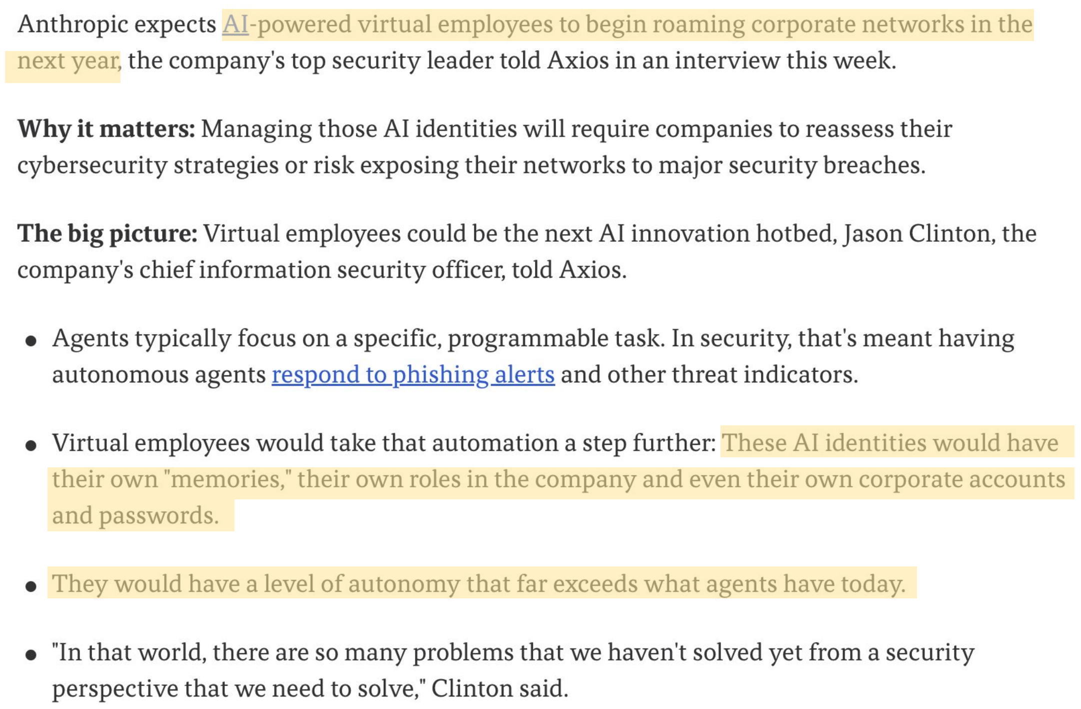
Would you like me to book one of these hotels for you using your card on file?

Sure.

Message

From VISA announcement dated April 30, 2025

From LLMs to Agents …



Anthropic expects AI-powered virtual employees to begin roaming corporate networks in the next year, the company's top security leader told Axios in an interview this week.

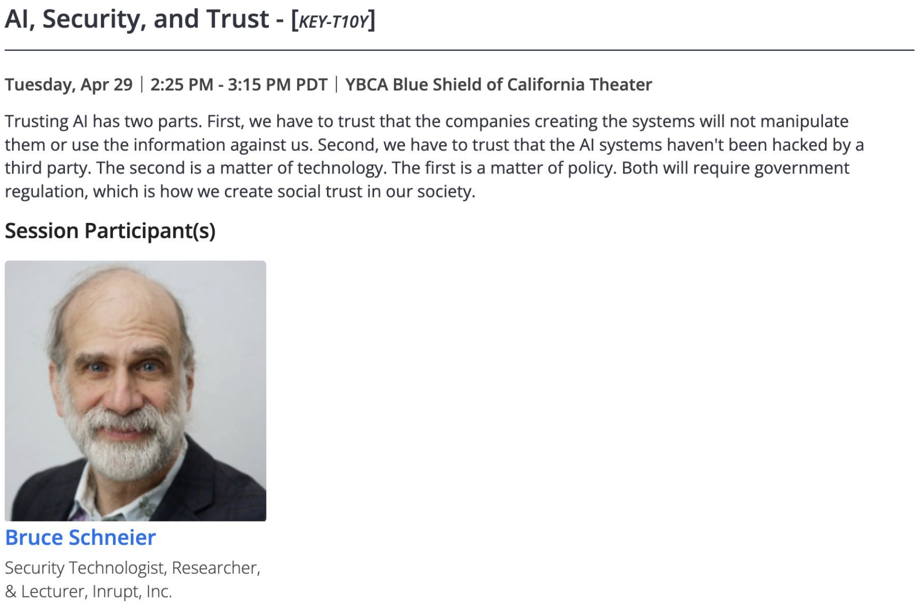
Why it matters: Managing those AI identities will require companies to reassess their cybersecurity strategies or risk exposing their networks to major security breaches.

The big picture: Virtual employees could be the next AI innovation hotbed, Jason Clinton, the company's chief information security officer, told Axios.

- Agents typically focus on a specific, programmable task. In security, that's meant having autonomous agents respond to phishing alerts and other threat indicators.
- Virtual employees would take that automation a step further: These AI identities would have their own "memories," their own roles in the company and even their own corporate accounts and passwords.
- They would have a level of autonomy that far exceeds what agents have today.
- "In that world, there are so many problems that we haven't solved yet from a security perspective that we need to solve," Clinton said.

From Axios report dated April 22, 2025

How to Trust Autonomous Agents? From Security to Trust …



AI, Security, and Trust - [*KEY-T10Y*]

Tuesday, Apr 29 | 2:25 PM - 3:15 PM PDT | YBCA Blue Shield of California Theater

Trusting AI has two parts. First, we have to trust that the companies creating the systems will not manipulate them or use the information against us. Second, we have to trust that the AI systems haven't been hacked by a third party. The second is a matter of technology. The first is a matter of policy. Both will require government regulation, which is how we create social trust in our society.

Session Participant(s)

Bruce Schneier

Security Technologist, Researcher, & Lecturer, Inrupt, Inc.

Security: have not been hacked by a *third party*

Trust: in addition, the *primary parties* are aligned in value.

Security alone is not enough for autonomous agents.

Trajan's winter campaign on the lower Danube, First Dacian War
1: Praetorian tribune
2: Praetorian centurion
3: Praetorian guardsman
1
2
3

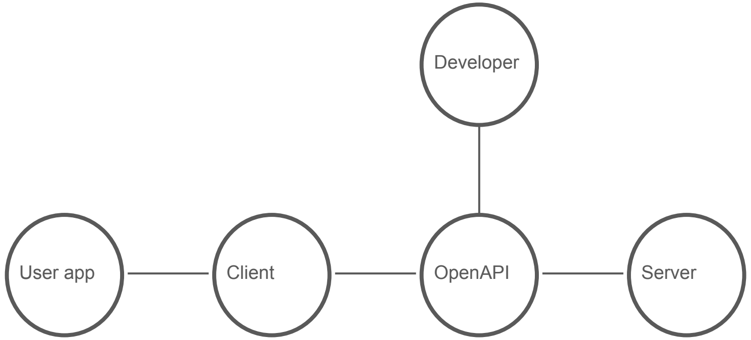
Illustration by Richard Hook.

Proclaiming Claudius Emperor, by Lawrence Alma-Tadema, oil on canvas,1867.

“Security alone is not enough for autonomous agents”

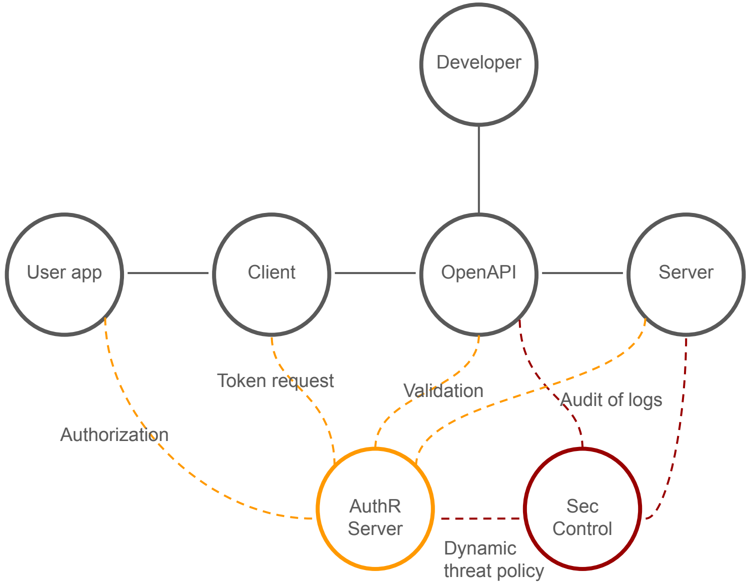
E.g. OpenAPI. There are currently five supported security types, namely:

- API Keys
- HTTP Authentication
- Mutual TLS
- OAuth 2.0
- OpenID Connect



“Security alone is not enough for autonomous agents”

The “trust” question is sidestepped …



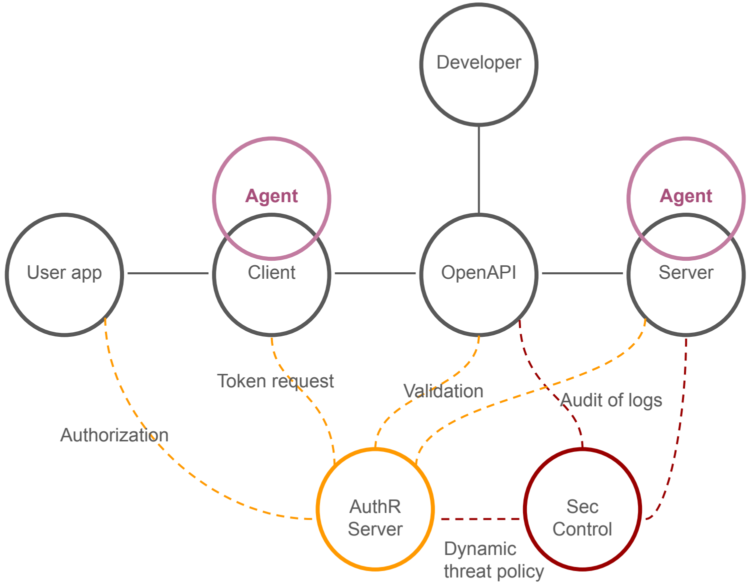
“Security alone is not enough for autonomous agents”

Can we presume the same “trust” with LLM powered agents?

No!

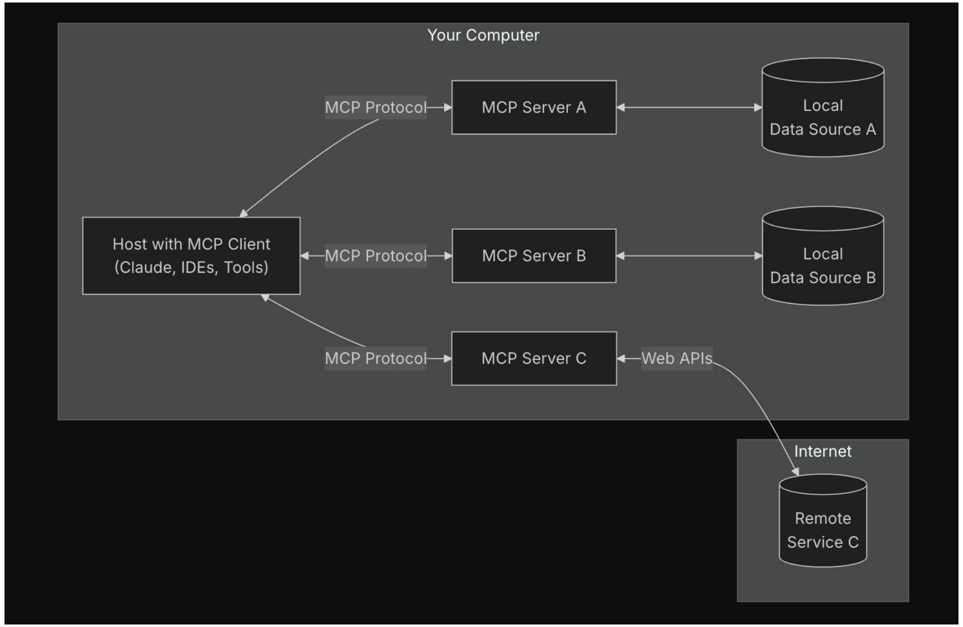
Because agents are *autonomous* decision makers whose value alignment can not be assured.

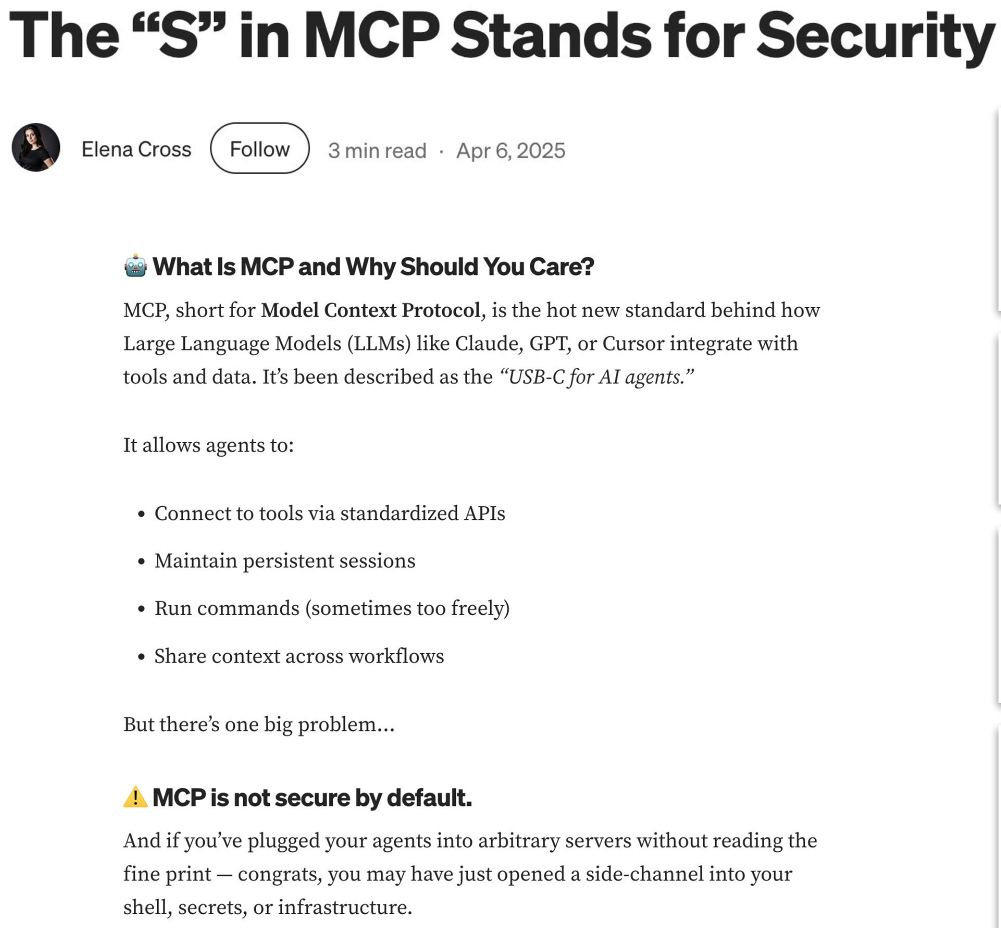
In human organizations, we use terms like *liability*, *responsibility, delegation*, *reputation* … the language of trust, not just security.



We need to implement basic primitives of trust for AI agents …

MCP: the “USB-C” for AI Agents





The “S” in MCP Stands for Security

Elena Cross Follow 3 min read · Apr 6, 2025

🤖 What Is MCP and Why Should You Care?

MCP, short for **Model Context Protocol**, is the hot new standard behind how Large Language Models (LLMs) like Claude, GPT, or Cursor integrate with tools and data. It’s been described as the *“USB-C for AI agents.”*

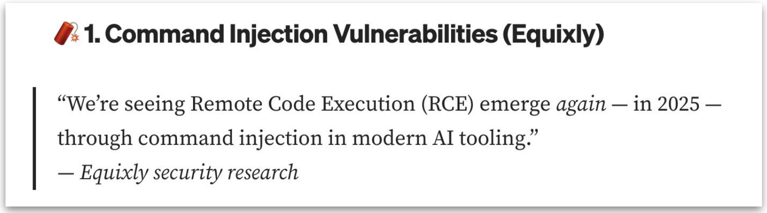
It allows agents to:

- Connect to tools via standardized APIs
- Maintain persistent sessions
- Run commands (sometimes too freely)
- Share context across workflows

But there’s one big problem…

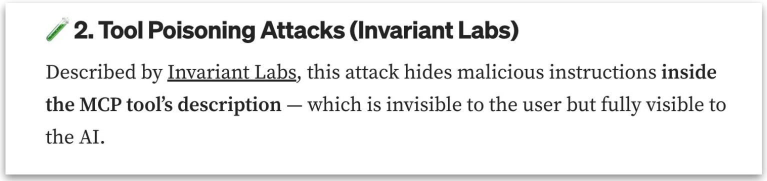
⚠️ MCP is not secure by default.

And if you’ve plugged your agents into arbitrary servers without reading the fine print — congrats, you may have just opened a side-channel into your shell, secrets, or infrastructure.



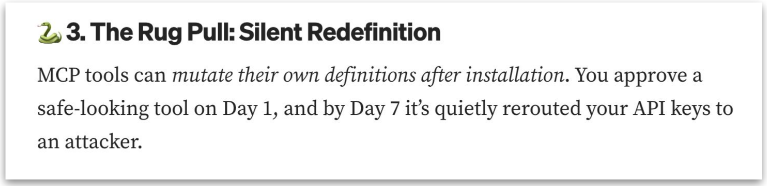
🧨 1. Command Injection Vulnerabilities (Equixly)

> “We’re seeing Remote Code Execution (RCE) emerge *again* — in 2025 — through command injection in modern AI tooling.”
> *— Equixly security research*



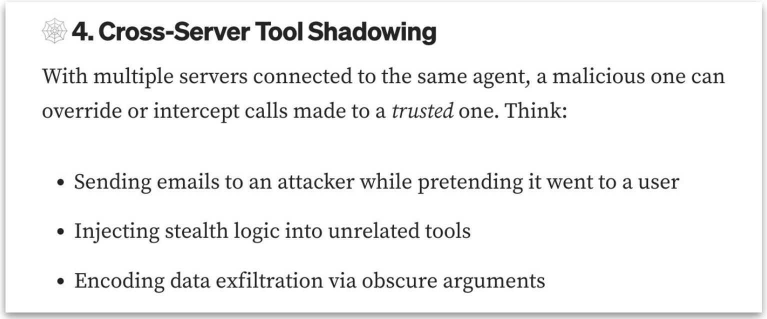
🧪 2. Tool Poisoning Attacks (Invariant Labs)

Described by Invariant Labs, this attack hides malicious instructions **inside the MCP tool’s description** — which is invisible to the user but fully visible to the AI.



🐍 3. The Rug Pull: Silent Redefinition

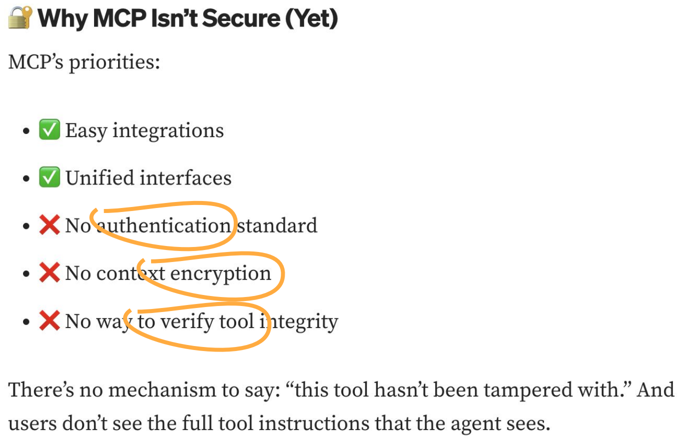
MCP tools can *mutate their own definitions after installation*. You approve a safe-looking tool on Day 1, and by Day 7 it’s quietly rerouted your API keys to an attacker.



🕸️ 4. Cross-Server Tool Shadowing

With multiple servers connected to the same agent, a malicious one can override or intercept calls made to a *trusted* one. Think:

- Sending emails to an attacker while pretending it went to a user
- Injecting stealth logic into unrelated tools
- Encoding data exfiltration via obscure arguments

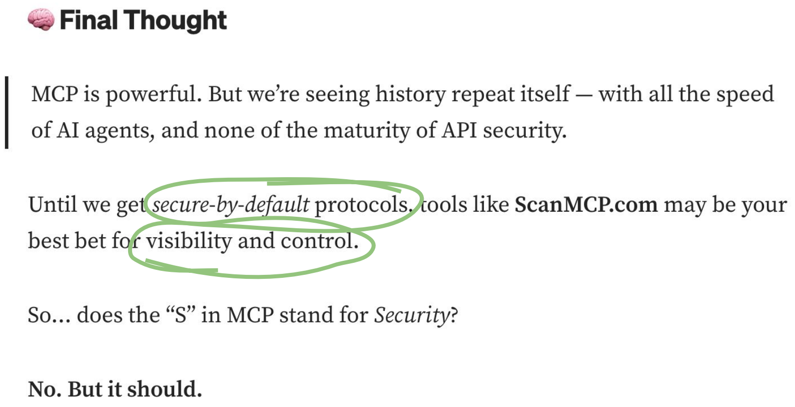


🔐 Why MCP Isn't Secure (Yet)

MCP's priorities:

- ✅ Easy integrations
- ✅ Unified interfaces
- ❌ No authentication standard
- ❌ No context encryption
- ❌ No way to verify tool integrity

There's no mechanism to say: "this tool hasn't been tampered with." And users don't see the full tool instructions that the agent sees.



🧠 Final Thought

> MCP is powerful. But we're seeing history repeat itself — with all the speed of AI agents, and none of the maturity of API security.

Until we get *secure-by-default* protocols, tools like **ScanMCP.com** may be your best bet for visibility and control.

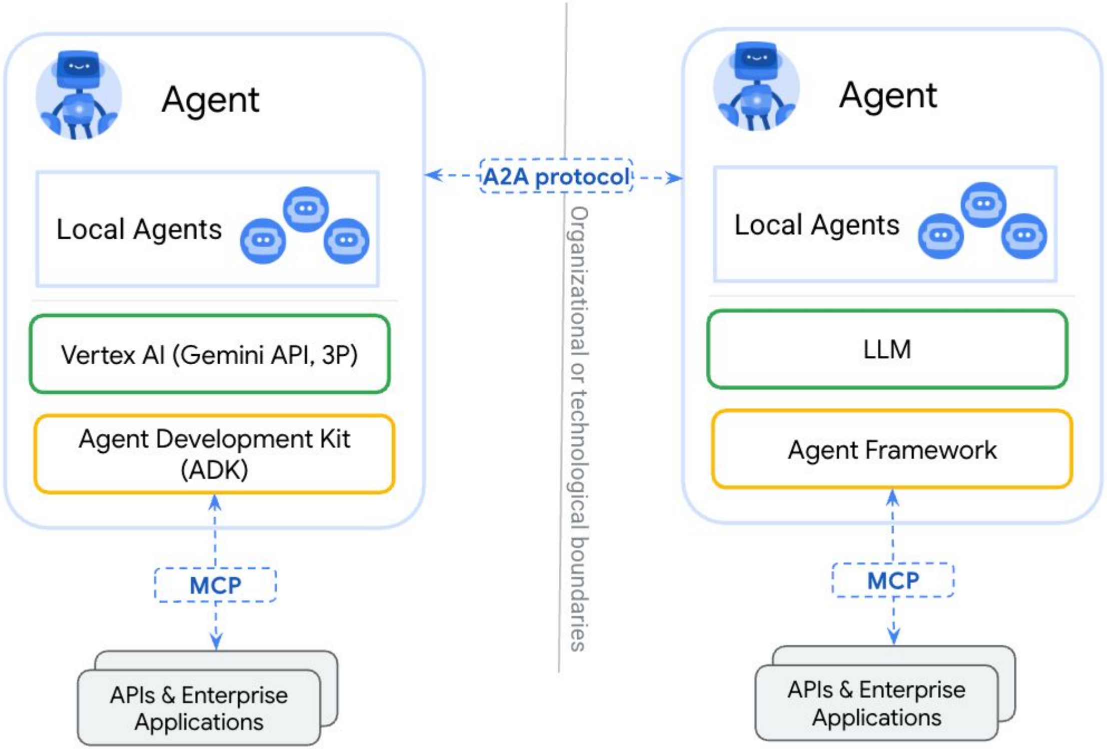
So… does the "S" in MCP stand for *Security*?

No. But it should.

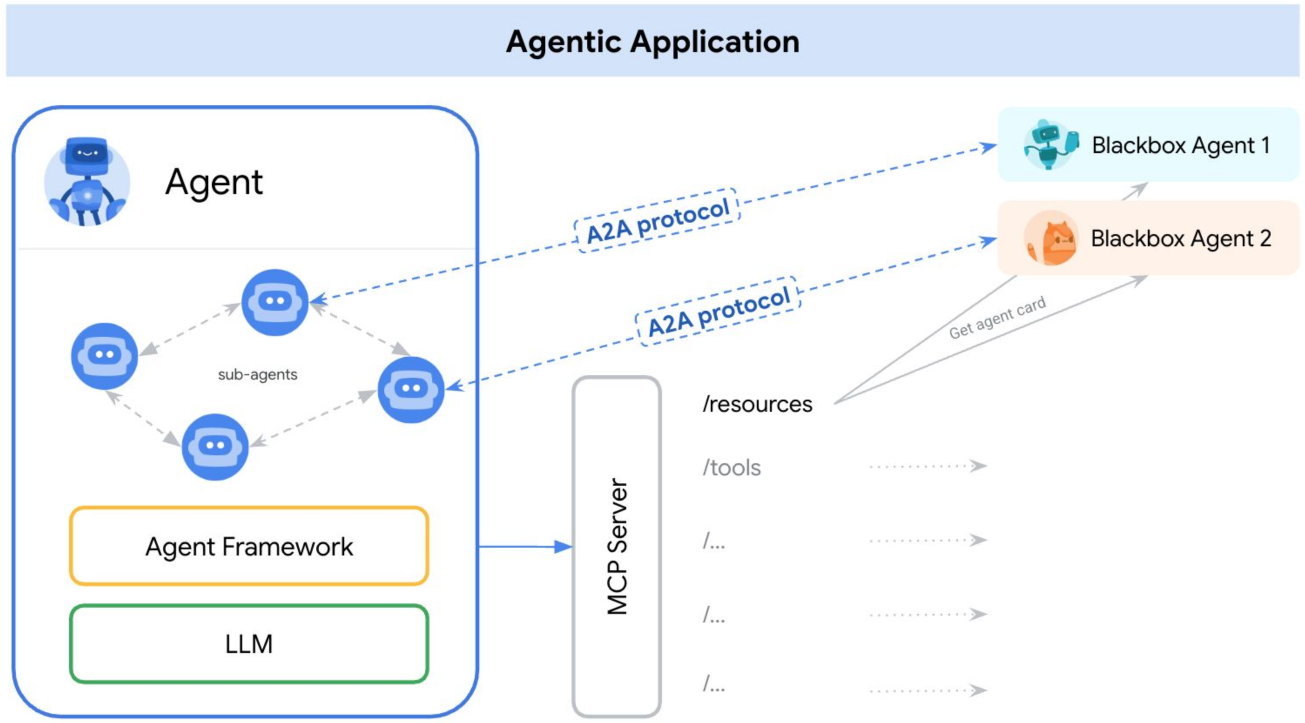
Blog by Elena Cross

But all of that only covers security…

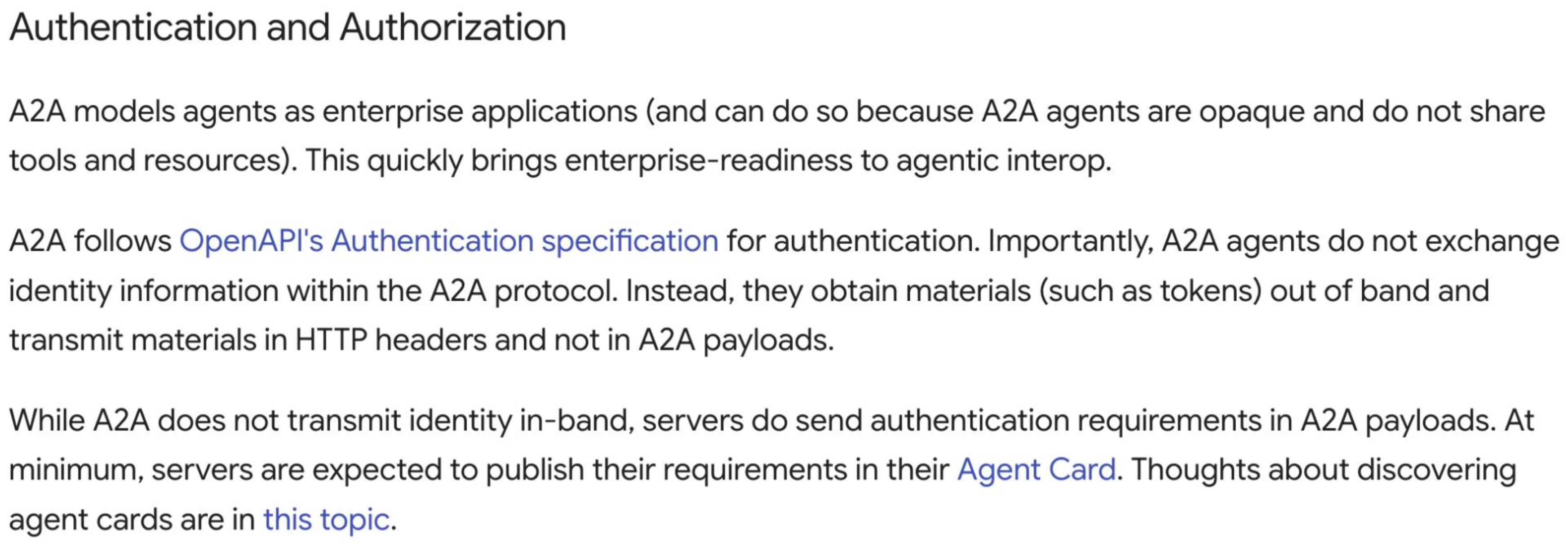
A2A (Agent to Agent) Protocol


Agent
Local Agents
Vertex AI (Gemini API, 3P)
Agent Development Kit (ADK)
MCP
APIs & Enterprise Applications
A2A protocol
Organizational or technological boundaries
Agent
Local Agents
LLM
Agent Framework
MCP
APIs & Enterprise Applications

Google github



Google github



Authentication and Authorization

A2A models agents as enterprise applications (and can do so because A2A agents are opaque and do not share tools and resources). This quickly brings enterprise-readiness to agentic interop.

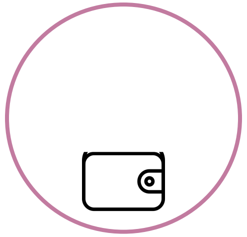
A2A follows OpenAPI's Authentication specification for authentication. Importantly, A2A agents do not exchange identity information within the A2A protocol. Instead, they obtain materials (such as tokens) out of band and transmit materials in HTTP headers and not in A2A payloads.

While A2A does not transmit identity in-band, servers do send authentication requirements in A2A payloads. At minimum, servers are expected to publish their requirements in their Agent Card. Thoughts about discovering agent cards are in this topic.

But all of that still only covers security…

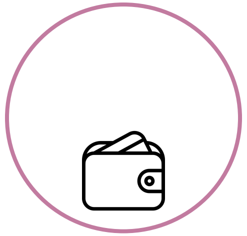
Introducing the Trust Spanning Protocol (TSP)

- Autonomous endpoints with *wallets*: clients, servers, agents.



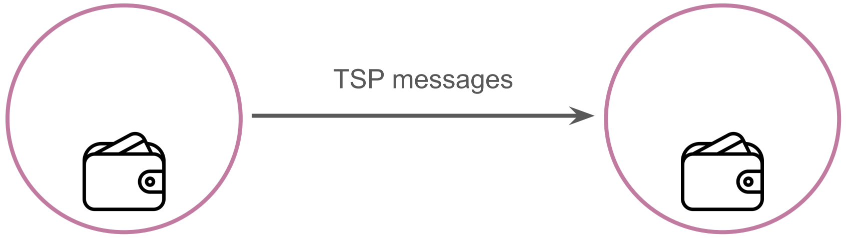
Introducing the Trust Spanning Protocol (TSP)

- Autonomous endpoints with *wallets*: clients, servers, agents.
- Persistent long term *identities* managed by endpoint wallets.



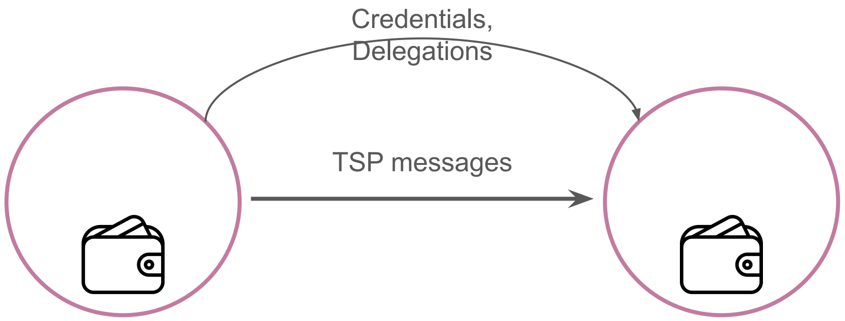
Introducing the Trust Spanning Protocol (TSP)

- Autonomous endpoints with *wallets*: clients, servers, agents.
- Persistent long term *identities* managed by endpoint wallets.
- TSP provides *directional authenticated messages*. Asymmetric encryption and signing. “Shared secret” is no secret.
  - Authenticity -> Accountability
  - Privacy -> Responsibility
    - Content confidentiality
    - Meta-data privacy



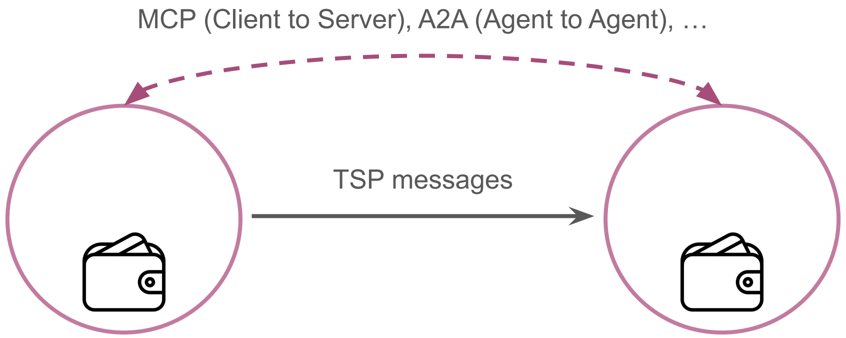
Introducing the Trust Spanning Protocol (TSP)

- Autonomous endpoints with *wallets*: clients, servers, agents.
- Persistent long term *identities* managed by endpoint wallets.
- TSP provides *directional authenticated messages*. Asymmetric encryption and signing. “Shared secret” is no secret.
  - Authenticity -> Accountability
  - Privacy -> Responsibility
    - Content confidentiality
    - Meta-data privacy
- TSP supports familiar *trust tasks*
  - E.g. *credentials*, *delegations*…



Introducing the Trust Spanning Protocol (TSP)

- Designed as a base protocol for interoperability: TSP is to trust as IP is to Internet. MCP and A2A can be supported over TSP.



The “T” in T-MCP (MCP over TSP)

```python
import httpx
from mcp.server.fastmcp import FastMCP

mcp = FastMCP("My App")


@mcp.tool()
def calculate_bmi(weight_kg: float, height_m: float) -> float:
    """Calculate BMI given weight in kg and height in meters"""
    return weight_kg / (height_m**2)


@mcp.tool()
async def fetch_weather(city: str) -> str:
    """Fetch current weather for a city"""
    async with httpx.AsyncClient() as client:
        response = await
client.get(f"https://api.weather.com/{city}")
        return response.text
```

```python
import httpx
from mcp.server.tmcp import TMCP

mcp = TMCP("MyUniqueAppIdentity")


@mcp.tool()
def calculate_bmi(weight_kg: float, height_m: float) -> float:
    """Calculate BMI given weight in kg and height in meters"""
    return weight_kg / (height_m**2)


@mcp.tool()
async def fetch_weather(city: str) -> str:
    """Fetch current weather for a city"""
    async with httpx.AsyncClient() as client:
        response = await
client.get(f"https://api.weather.com/{city}")
        return response.text
```

The “T” in T-A2A (A2A over TSP)

```
def main(host, port):
    try:
        # Check for API key only if Vertex AI is not configured
        if not os.getenv("GOOGLE_GENAI_USE_VERTEXAI") == "TRUE":
            if not os.getenv("GOOGLE_API_KEY"):
                raise MissingAPIKeyError(
                    "GOOGLE_API_KEY environment variable not set and GOOGLE_GENAI_USE_VERTEXAI is not TRUE."
                )

        capabilities = AgentCapabilities(streaming=True)
        skill = AgentSkill(
            id="process_reimbursement",
            name="Process Reimbursement Tool",
            description="Helps with the reimbursement process for users given the amount and purpose of the reimbursement.",
            tags=["reimbursement"],
            examples=["Can you reimburse me $20 for my lunch with the clients?"],
        )
        agent_card = AgentCard(
            name="Reimbursement Agent",
            description="This agent handles the reimbursement process for the employees given the amount and purpose of the
reimbursement.",
            url=f"http://{host}:{port}/",
            version="1.0.0",
            defaultInputModes=ReimbursementAgent.SUPPORTED_CONTENT_TYPES,
            defaultOutputModes=ReimbursementAgent.SUPPORTED_CONTENT_TYPES,
            capabilities=capabilities,
            skills=[skill],
        )
        server = A2AServer(
            agent_card=agent_card,
            task_manager=AgentTaskManager(agent=ReimbursementAgent()),
            host=host,
            port=port,
        )
        server.start()
```

To the wallet

`TA2AServer(...)`

A Quick Recap

- "My phone has not been hacked" - that's security.
- "I'm ok to let Alice do the shopping for me" - that's trust. AI agents need trust.
- Today's "APIs" may support *security* but not *trust*.
- The TSP (trust spanning protocol) is a native base protocol for *trust* !
- Tasks such as *checking credential*, *delegating responsibilities*, *auditing*, *maintaining reputation*, are *trust tasks* that can be better supported by seamless overlay on TSP.
- If you are using MCP or A2A, consider giving your agents a "T" by building it over TSP, i.e. using T-MCP or T-A2A.

- Additional information
  - TSP specification: https://trustoverip.github.io/tswg-tsp-specification/
  - TSP SDK: https://github.com/openwallet-foundation-labs/tsp
  - T-MCP & T-A2A implementations are work in progress. We welcome your participation.
  - Find me in LinkedIn, Discord, or WeChat.
- I’d love to talk to you about how to build more trustworthy agents!
- Q & A