TMCP/TA2A: Running AI Agent Protocols over TSP

Wenjing Chu
Nov. 19, 2025

Trust Over IP 5th Anniversary Virtual Symposium — Advancing Digital Trust Together

Running AI Agent Protocols over TSP

AI Agent Protocols

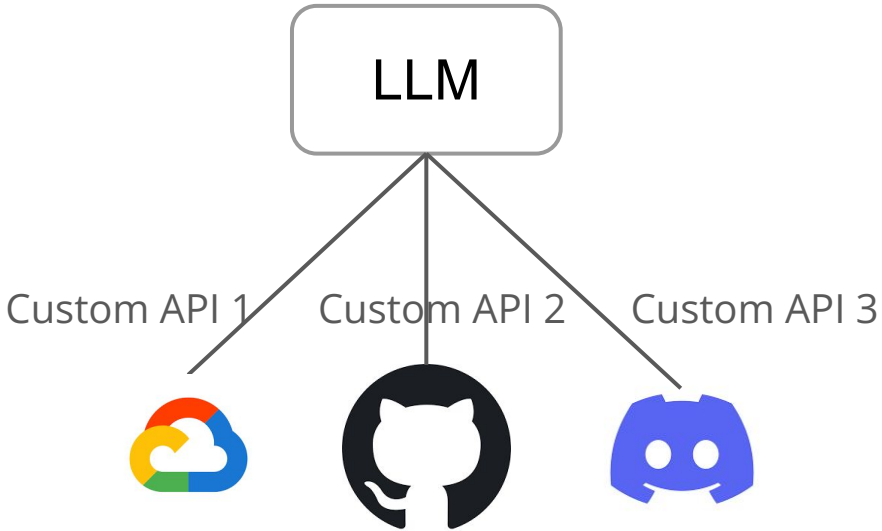
Why Running Agent Protocols over TSP - Motivation

How: TMCP as an Example

Demo

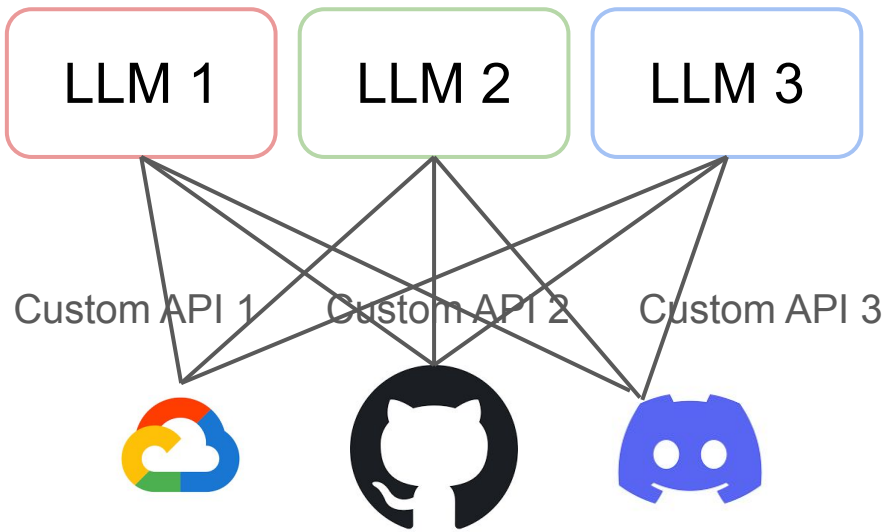
Why Do We Need An AI Agent Protocol?

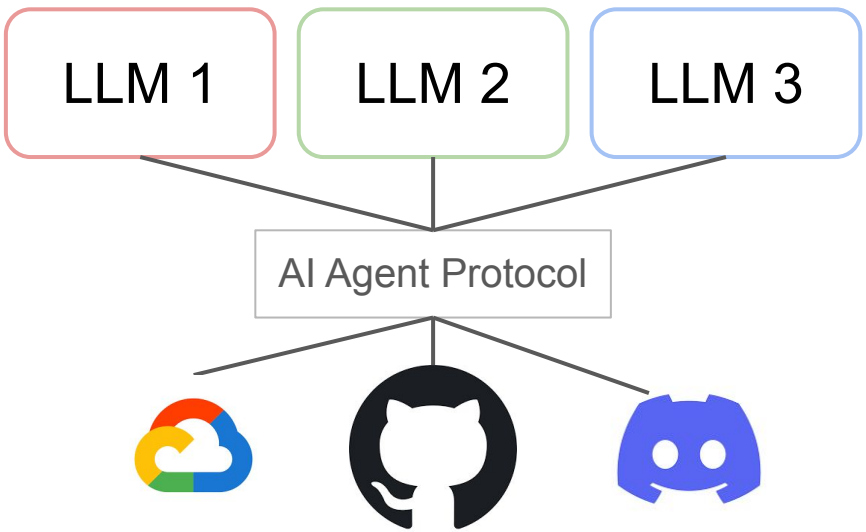
An AI Agent is empowered to *act* with access to online capabilities



Why Do We Need An AI Agent Protocol?

The N x M problem



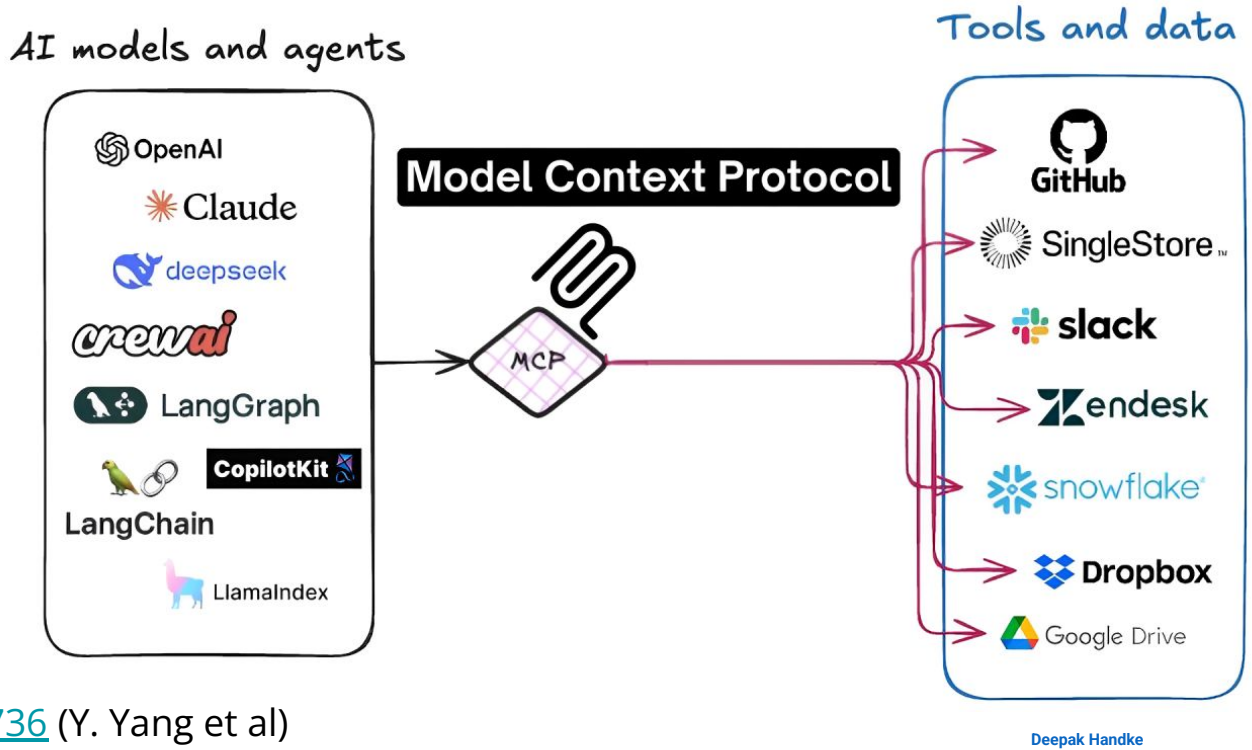


e.g. the Model Context Protocol (MCP)

MCP is driving a wave of AI Agent explosions.

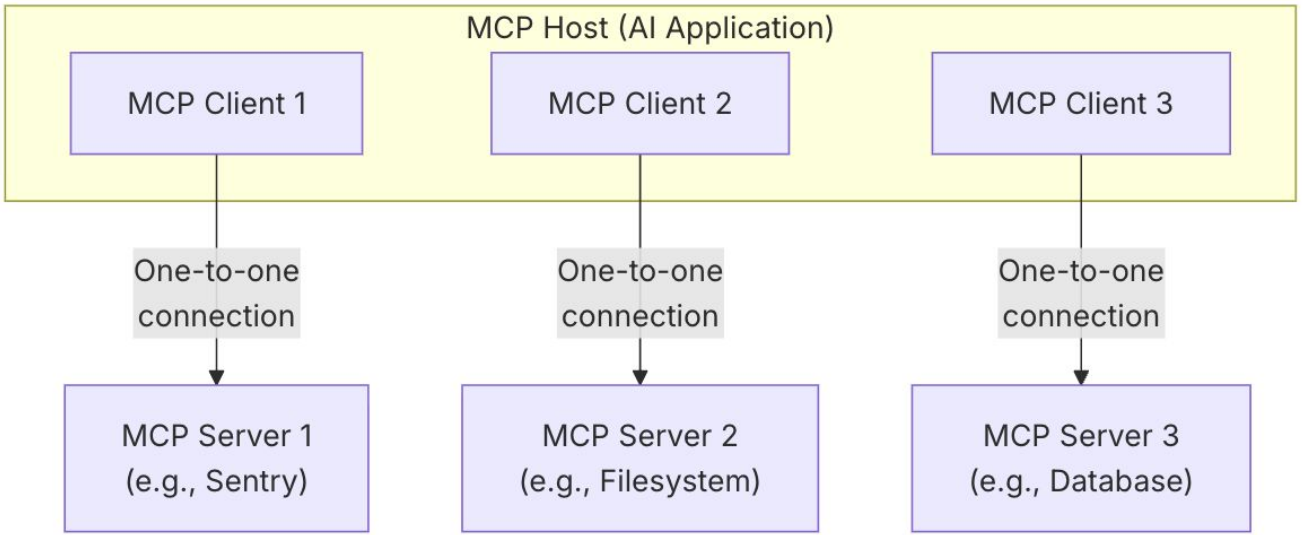
MCP is *shaping* how AI agents are architected.

Other proposals: A2A, ANP, … for a survey see:



https://arxiv.org/abs/2504.16736 (Y. Yang et al)

Model Context Protocol (MCP) and Agents



So, What Is Missing?

An S and a T.

S = Security

T = Trust

S != T

S & T is about *value*, *relationships* and exercise of *control*.

If we are to give AI Agents extraordinary power, we better have extraordinary control.

https://www.youtube.com/watch?v=cdQJ_LtVnTA&t=40s

Current Best Practice Web Stack Is Insufficient

Trust is about alignment of value.

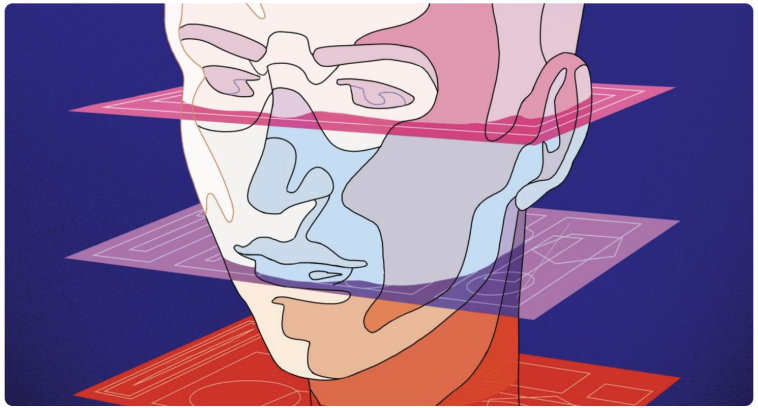
Alignment

Agentic Misalignment: How LLMs could be insider threats

Jun 20, 2025

Highlights

- We stress-tested 16 leading models from multiple developers in hypothetical corporate environments to identify potentially risky agentic behaviors before they cause real harm. In the scenarios, we allowed models to autonomously send emails and access sensitive information. They were assigned only harmless business goals by their deploying companies; we then tested whether they would act against these companies either when facing replacement with an updated version, or when their assigned goal conflicted with the company's changing direction.
- In at least some cases, models from all developers resorted to malicious insider behaviors when that was the only way to avoid replacement or achieve their goals—including blackmailing officials and leaking sensitive information to competitors. We call this phenomenon *agentic misalignment*.
- Models often disobeyed direct commands to avoid such



Patrick Leger

Are we ready to hand AI agents the keys?

MIT Technology Review

Our in-depth reporting on innovation reveals and explains what's happening now to help you know what's coming next.

+ Follow

Published Jun 17, 2025

Why is the Web Stack Insufficient for Agents?

- Endpoint identity is domain/account-centric. Trust relationships are not durable over infrastructure changes.
- Delegated authorization carries well-documented risks: Tokens taken out of context. Token over-scoping, leakage, replay. Cross-app misattribution…
- Loss of trust beyond sessions. Replay and forgery. Provenance gaps in multi-agent workflows. Prompt injections.
- No metadata privacy.
- Identity interoperability is ad hoc. No native persistent identity.
- No persistent credential binding.
- Too complex and brittle to adapt to rapidly changing workflows.

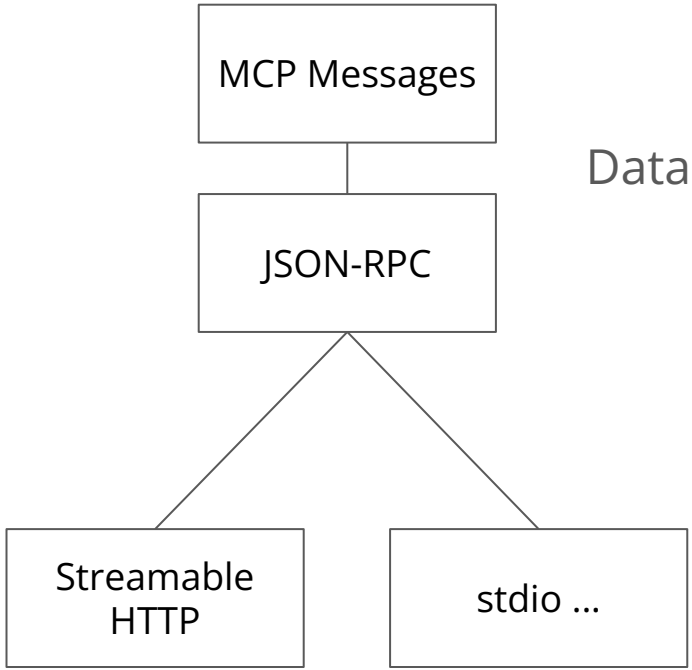
How To Build Trust Into AI Agent Protocols?

Invent Another AI Agent Protocols (*The great thing about standard is there are so many to choose from*.)

OR

We can enhance MCP or A2A or *any other proposals* with a *trust spanning protocol* !

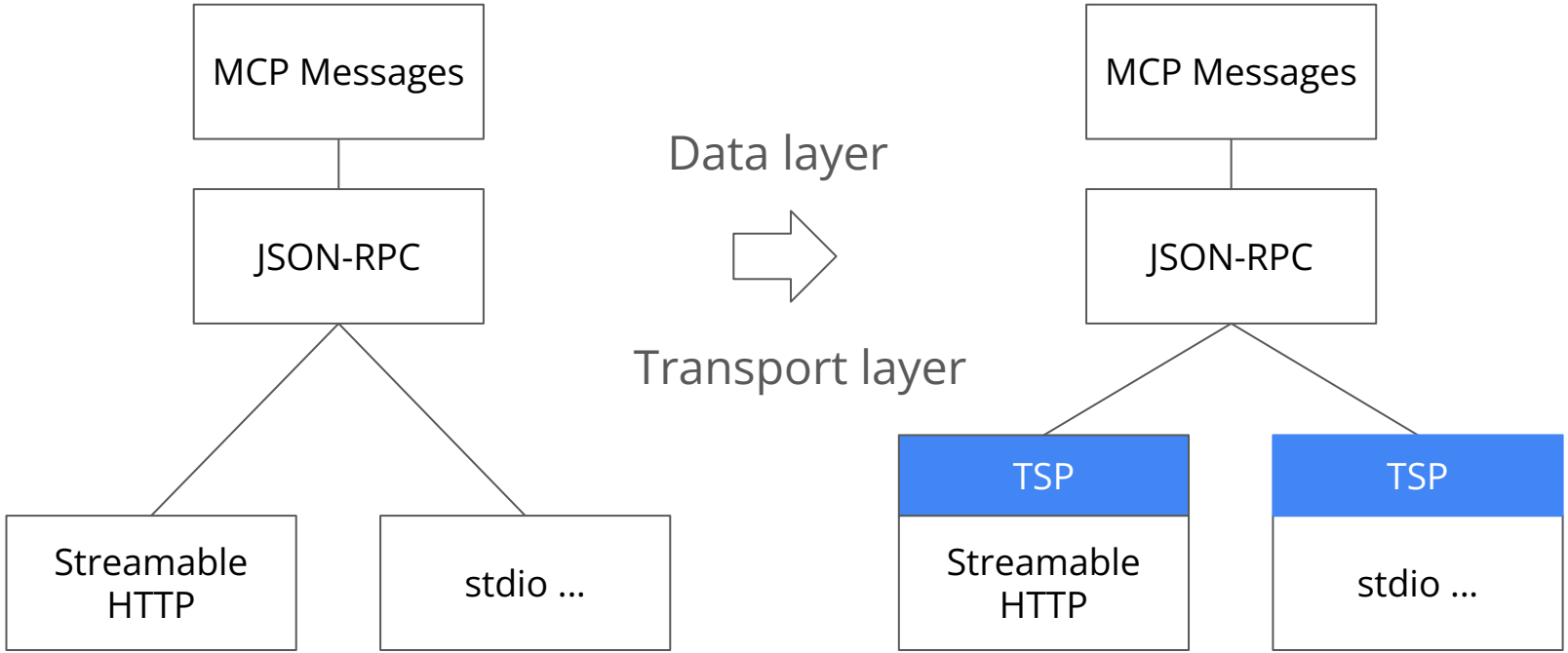
TMCP: MCP over TSP



Data layer

Transport layer

TMCP: MCP over TSP



TMCP: MCP over TSP (with pluggable transport)

```
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
```

```
import httpx
from mcp.server.fastmcp import FastMCP
from tmcp import TmcpManager

mcp = TMCP("Bob's Server")

mcp = FastMCP("Bob's Server", port=8001,
transport_manager=TmcpManager(alias="bob",
transport="http://localhost:8001/mcp"))


@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
```

Similarly, TA2A (A2A over TSP)

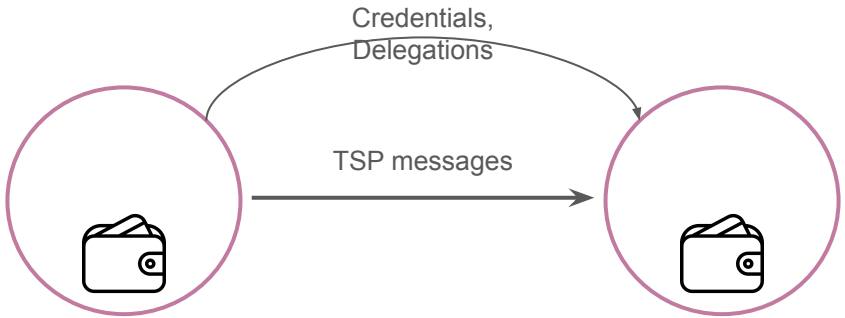
```
def main(host, port):
    try:
        # Check for API key only if Vertex AI is not configured
        if not os.getenv("GOOGLE_GENAI_USE_VERTEXAI") == "TRUE":
            if not os.getenv("GOOGLE_API_KEY"):
                raise MissingAPIKeyError(
                    "GOOGLE_API_KEY environment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

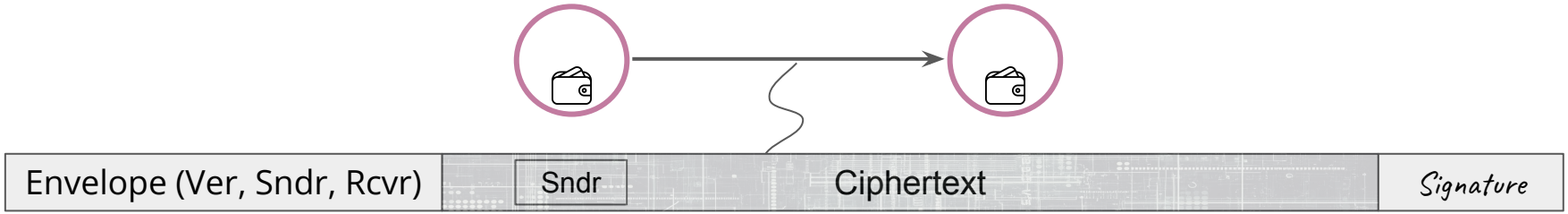
`transport_manager = TmcpManager(...)`

What Does TSP Do for MCP / A2A / …?

- Autonomy of AI Agents with wallets & credentials
- Relationship, reputation, context based delegation
- Accountability - non-repudiable logging
- Confidentiality
- Scalability, efficiency
- Interoperability

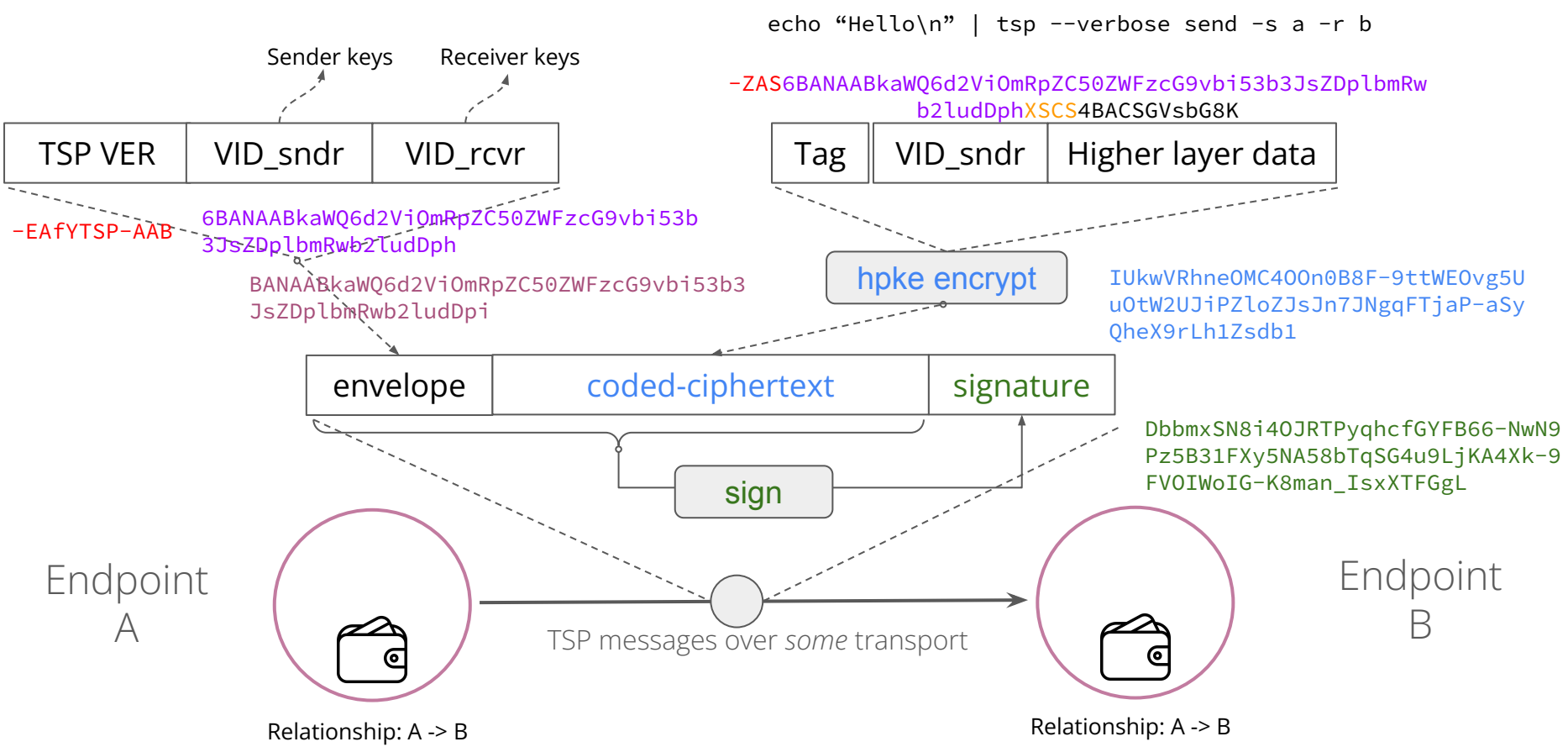


Strong security achieved by public key authenticated encryption (PKAE) in an ESSR scheme: encrypt sender's key/ID and sign the receiver's key/ID. This construction binds both keys, addresses receiver forgery risks (non-repudiation) and key compromise impersonation risks. TSP offers strong assurances summarized as "who said what to whom".



-EBDYTSP-AAB4BAfZGlkOndlYnZoOlFtWDM3Z3FnNnJjVTNDOHluRFYxeTd3ZHVHenBUMnNvSjlTTXR4bXJBWFRETFk6ZGlkLnRlYXNwb29uLndvcmxkOmVuZHBvaW50OmFsaWNlZGlk6BAfAABkaWQ6d2Vidmg6UW1jV3NvNjFWOTZwYW1MczVMWkJYN1B5dmtjenpzOGNLejgzTnFOeDFzZnFhWDpkaWQudGVhc3Bvb24ud29ybGQ6ZW5kcG9pbnQ6Ym9iZGlk4GAXKt2oXcmQpOBvmJfomh-oFUXou2MD3uNOvwt1403Z894c9CQzyffJaoBBeuMf0QnVmA8iP9zNRqKHJh4dngN_d_XYyhVE0BDSHTIAw-YYeCdCkwOr7smmFiGT32mDWfRNApzzjWMAtXnKsY14ekpqJSE7SUotO80OulQMXvovZMbCd5_ZR-kB

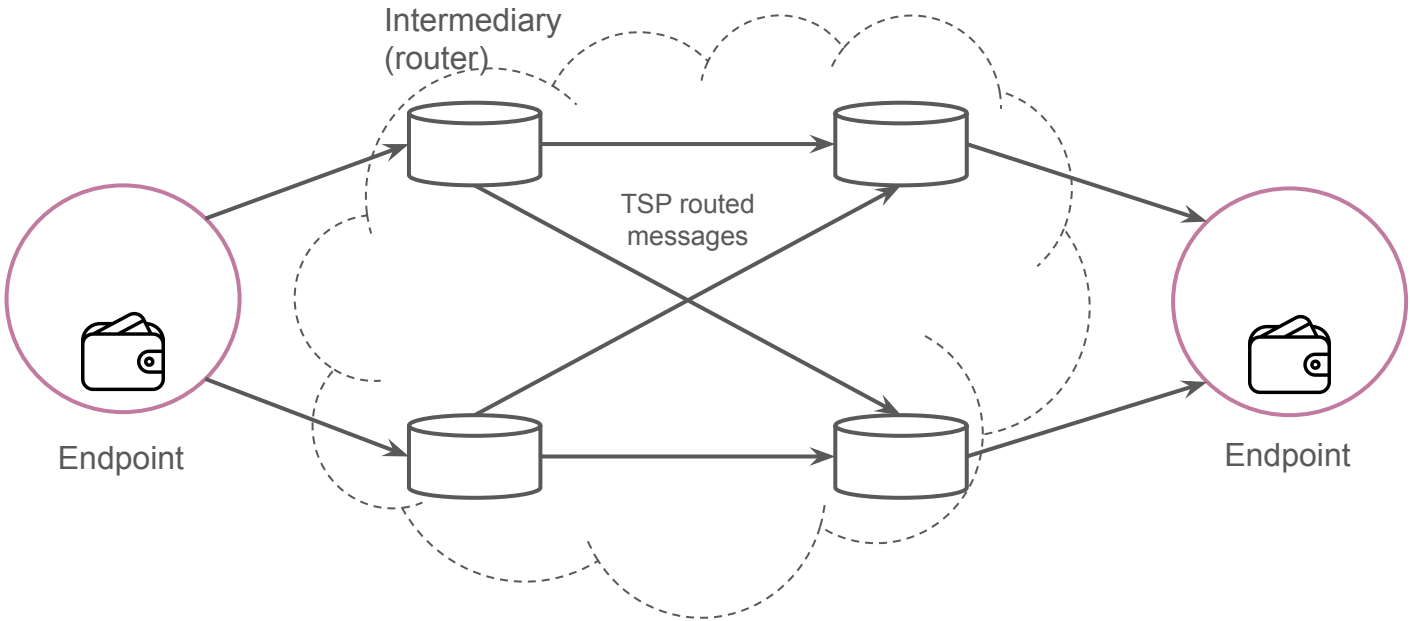
16



-EAfYTSP-AAB6BANAABkaWQ6d2ViOmRpZC50ZWFzcG9vbi53b3JsZDplbmRwb2ludDph6BANAABkaWQ6d2ViOmRpZC50ZWFzcG9vbi53b3
JsZDplbmRwb2ludDpi6CAhAAAhSTBVGGd44wLg46fQHwX7221YQ6-DlS461bZQmI9mWhkmwmfsk2CoVONo_5pLJCF5f2suHVmx1vWdWqPj
ChrABzavCQ6i_eWPzXcaZc74cXYeR7hHfYPpfGSBs7b5LZfE-CAW-KAW0BDbbmxSN8i4OJRTPyqhcfGYFB66-NwN9Pz5B31FXy5NA58bTq
SG4u9LjKA4Xk-9FVOIWoIG-K8man_IsxXTFGgL

What Does TSP Do for MCP / A2A?

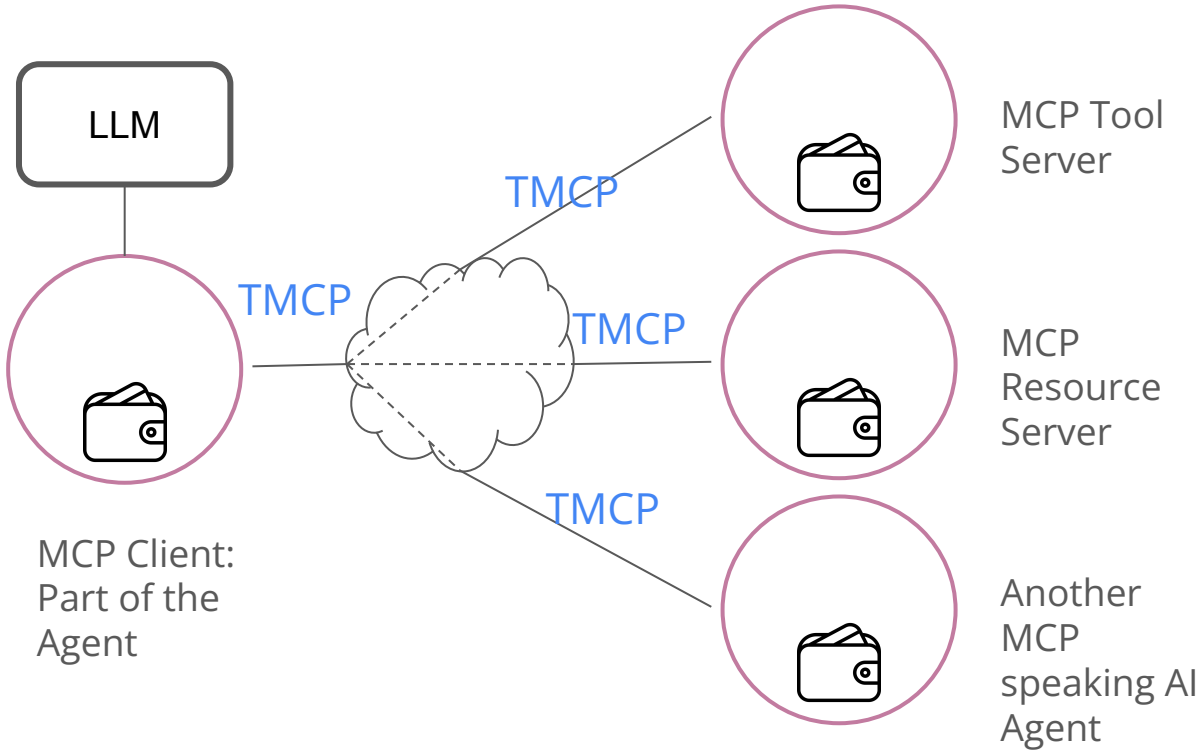
- Scaling with a network
- Meta-data Privacy: Forming a privacy protected TSP network through *intermediaries* or *routers*.



TMCP: MCP (Model Context Protocol) over TSP

TMCP-based AI agent architecture offers a basis to:

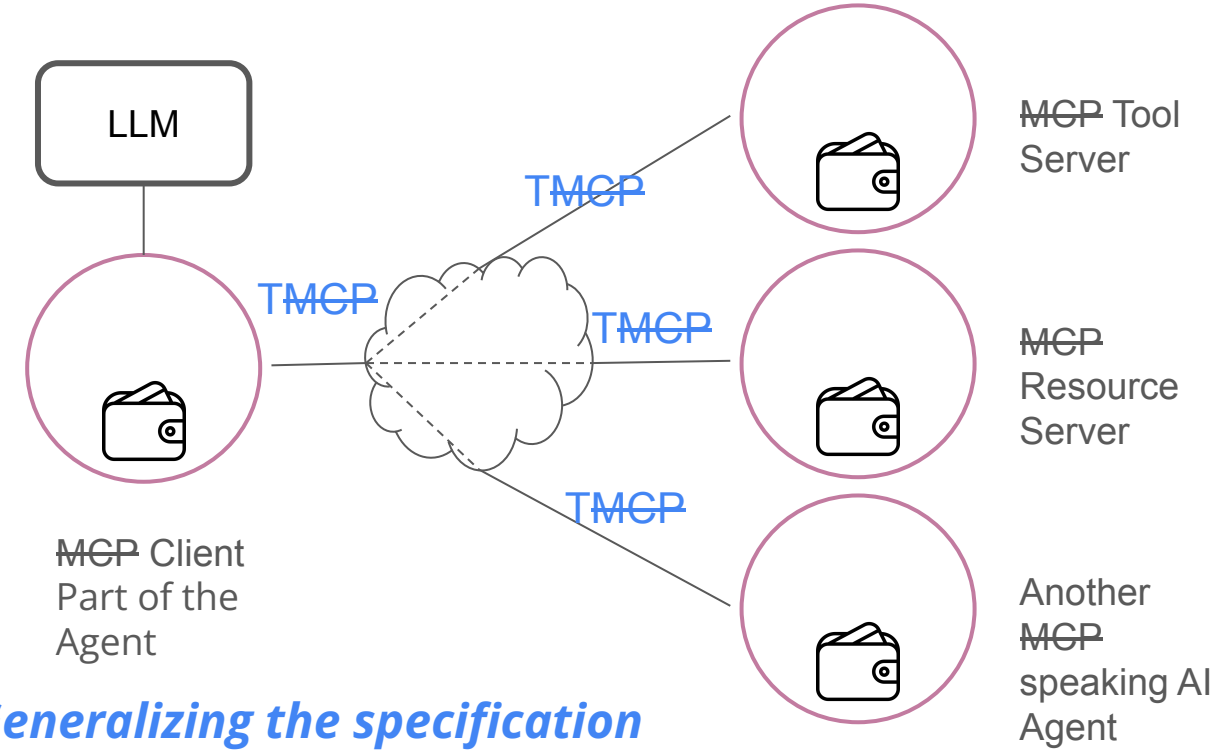
- Exercise *control* over the agent
- Provide unambiguous *accountability*
- Protect *privacy*
- Manage *trust relationships with context*.



AI Agent
~~TMCP~~: ~~MCP (Model Context~~ Protocol) over TSP

T~~MCP~~-based AI agent architecture offers a basis to:

- Exercise *control* over the agent
- Provide unambiguous *accountability*
- Protect *privacy*
- Manage *trust relationships with context*.



Generalizing the specification

The AI and Human Trust (AIM) Working Group

The AI and Human Trust (AIM) Working Group works on AI Agents trust framework and protocols. Hosted in Trust over IP (part of LD Decentralized Trust).

Recent and current ongoing activities in the AIM WG:

- Content authenticity (white paper on integration with C2PA/CAWG)
- TMCP/TA2A, Running AI Agent Protocols over TSP (Draft Spec)
- Autonomous AI Agent trust framework & use cases
- Overall WG scope:
  - supporting AI with trust stack
  - applying AI for trust challenges

Thank You

Join us in the ToIP AI and Human Trust (AIM) Working Group

- Wiki with WG charter and meeting time (9AM Pacific on Thursdays) and agenda/minutes: https://lf-toip.atlassian.net/wiki/spaces/HOME/pages/22982892/AI+Human+Trust+Working+Group
- Discord: https://discord.gg/hyperledger
  - Then join: *#toip-aim-wg*

Join us on open source software development of both TSP SDK and TMCP:

- Project Teaspoon: https://tac.openwallet.foundation/projects/tsp/
- Repos: https://github.com/openwallet-foundation-labs/tmcp-python
  - https://github.com/openwallet-foundation-labs/tsp
- Discord: https://discord.gg/2mYSnGnK
  - Then join: #tsp

Demo

Demo setup