TSP: Trust Spanning Protocol (Rev2)

Wenjing Chu
Nov. 19, 2025

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

TSP: Trust Spanning Protocol (Rev2)

What’s new in TSP Rev 2 ?

Why is TSP different ?

Broad areas of applications

Demo of open source TSP SDK

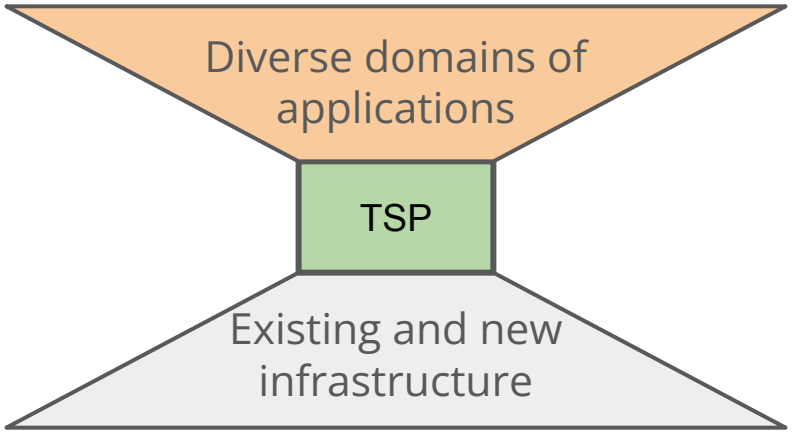
What is new in TSP Rev2?

- To access and contribute:
  - TSP Spec authored by Wenjing Chu and Sam Smith:
    - https://trustoverip.github.io/tswg-tsp-specification/
  - TSP SDK Code hosted in this OpenWallet repo:
    - https://github.com/openwallet-foundation-labs/tsp
- What is new in Rev2
  - Aligning with CESR code table version 2
  - Revising TSP relationship forming control messages
  - Adding experimental PQC support
  - Removing VID type, supporting DID and URN formats
  - Fixing editorial and other minor issues
  - Open source SDK from 0.8 to 0.9

Why is TSP different?

1. A minimalistic trust spanning layer
    - Broadening applicable application domains: target general purpose use
    - Maximizing opportunity for interoperability - adopting commonality without undue restriction
    - TSP is the spanning layer of the Trust over IP stack

AI agents/pipelines, webs, APIs, wallets, mobile, telco, messaging, social media, Git, supply chains, clouds, networks, industrial, digital twins, ...



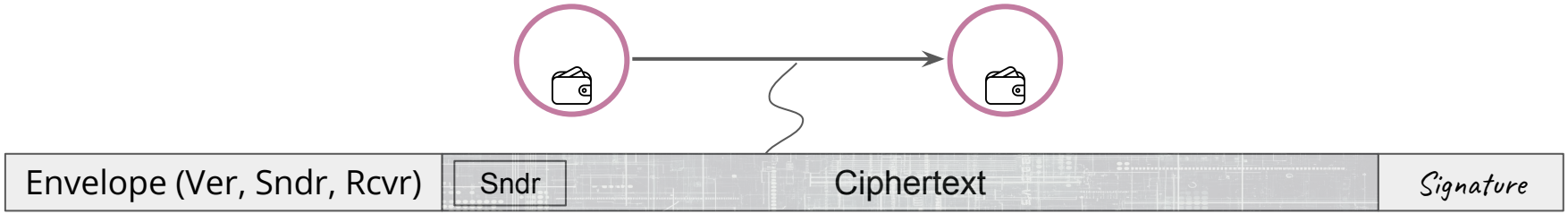
Existing internet and web infrastructure, emerging AI infrastructure, cloud scaling, hardware/gateway/device deployments...

Why is TSP different?

2. A compact & strong trust primitive for general purpose use

- Integrating *strong security and privacy primitives* in a compact efficient protocol
- Enabling us to answer the essential trust question: "*who said what to whom*".
- Overcoming shortcomings in the existing "*web stack*" with backward compatible evolution paths
- Offering a simple and easy to adopt trust framework for developers, providers, policy governance roles / parties

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

6

An Example VID: did:webvh
Also keri or did:webs, scid

Self-certifying identifier (SCID)

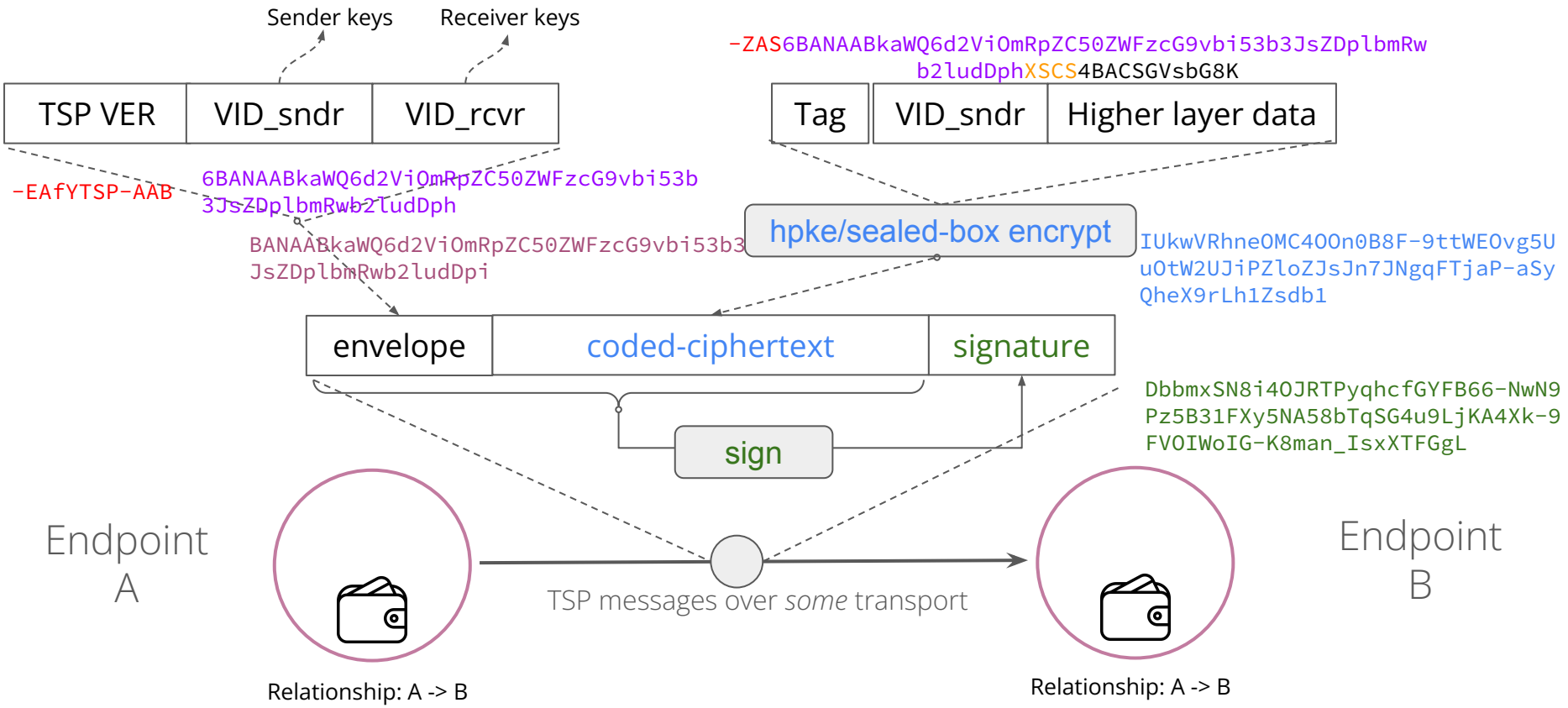
did:webvh:QmV9kqaJGL7aeS8YtxvT4yaTZfYLKqUvna61WfqNszVnEZ:did.teaspoon.world:endpoint:ddg-searchTmcpSseServer-5866f957-c21f-47f4-a7e4-62b61b8b3139

Resolves to

https://did.teaspoon.world/endpoint/ddg-searchTmcpSseServer-5866f957-c21f-47f4-a7e4-62b61b8b3139/did.jsonl

Which can be verified (with history) & resolved to transport mechanisms (e.g. URLs) etc.

```
echo “Hello\n” | tsp --verbose send -s a -r b
```

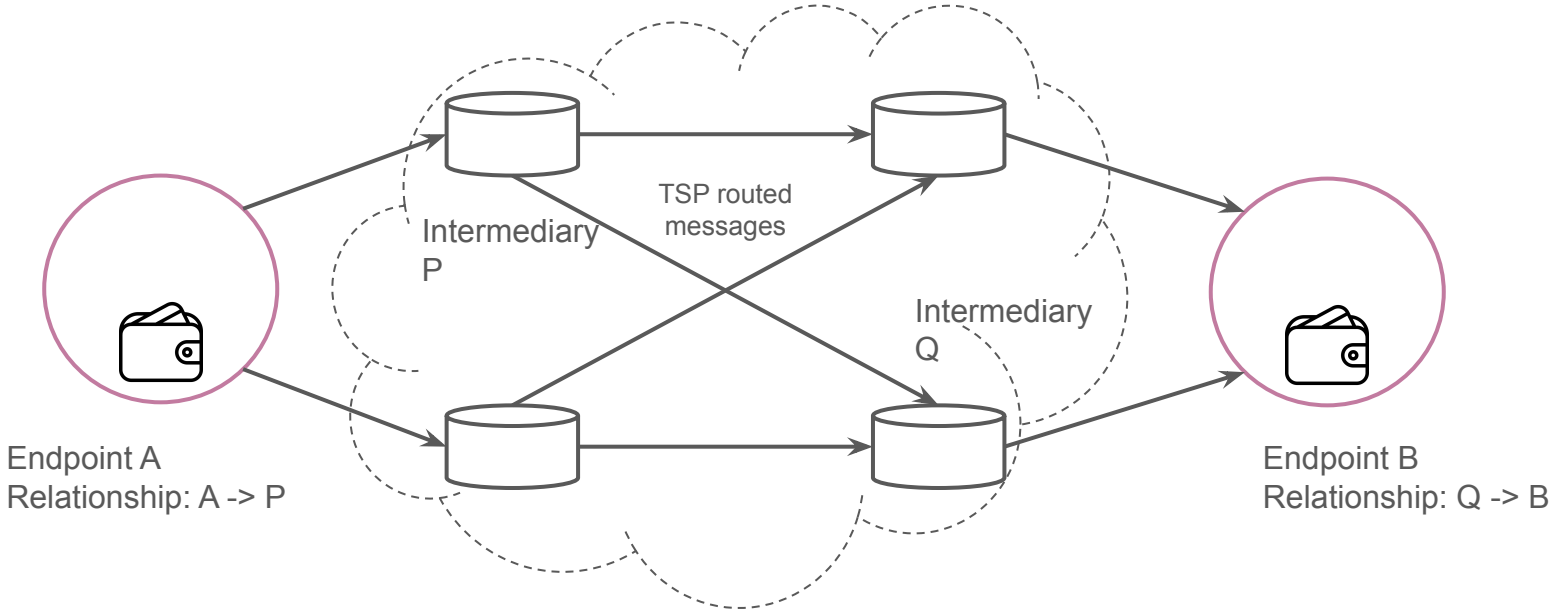



```
-EAfYTSP-AAB6BANAABkaWQ6d2ViOmRpZC50ZWFzcG9vbi53b3JsZDplbmRwb2ludDph6BANAABkaWQ6d2ViOmRpZC50ZWFzcG9vbi53b3
JsZDplbmRwb2ludDpi6CAhAAAhSTBVGGd44wLg46fQHwX7221YQ6-DlS461bZQmI9mWhkmwmfsk2CoVONo_5pLJCF5f2suHVmx1vWdWqPj
ChrABzavCQ6i_eWPzXcaZc74cXYeR7hHfYPpfGSBs7b5LZfE-CAW-KAW0BDbbmxSN8i4OJRTPyqhcfGYFB66-NwN9Pz5B31FXy5NA58bTq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

Why is TSP different?

3. A protocol that scales with the internet and strengthens metadata privacy

- Offering standardized nesting
- And standardized metadata privacy protecting message routing

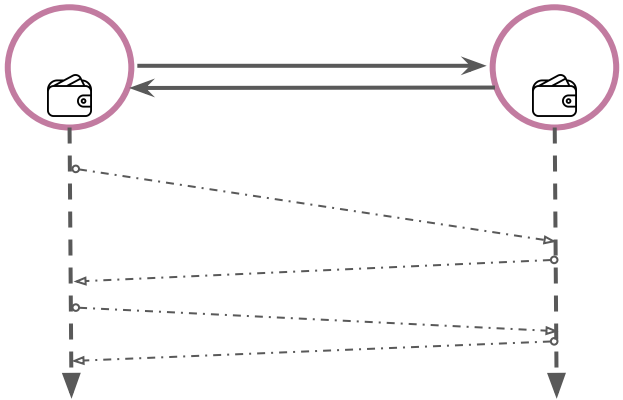


Why is TSP different?

4. A general purpose protocol in managing *long term trust relationships*

- Integrating *long term verifiable identifiers,* e.g. SCIDs, DID:WEBVH, KERI
- Standardized mechanisms with relationship forming protocol
- Enabling strong trust data/signal for reputation and context

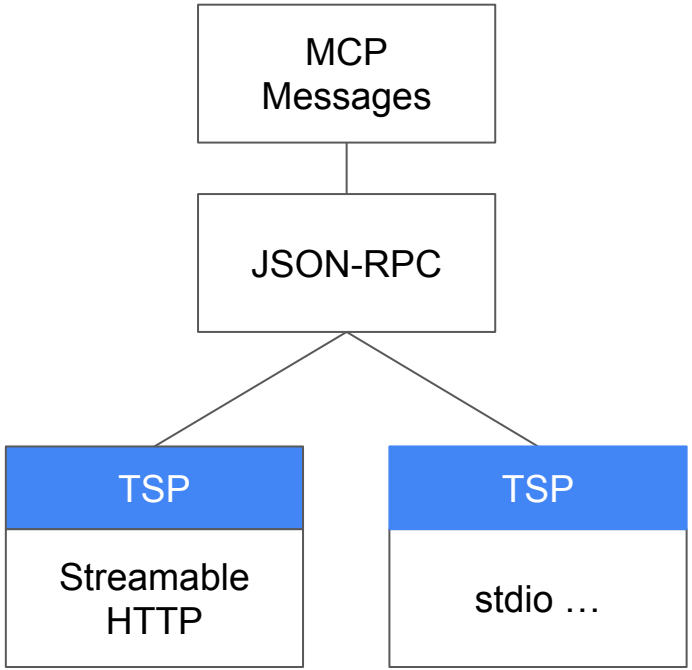
```
did:webvh:QmV9kqaJGL7aeS8YtxvT
4yaTZfYLKqUvna61WfqNszVnEZ:did
.teaspoon.world:endpoint:ddg-s
earchTmcpSseServer-5866f957-c2
1f-47f4-a7e4-62b61b8b3139
```



Why is TSP different?

5. A flexible trust solution for AI agent protocols

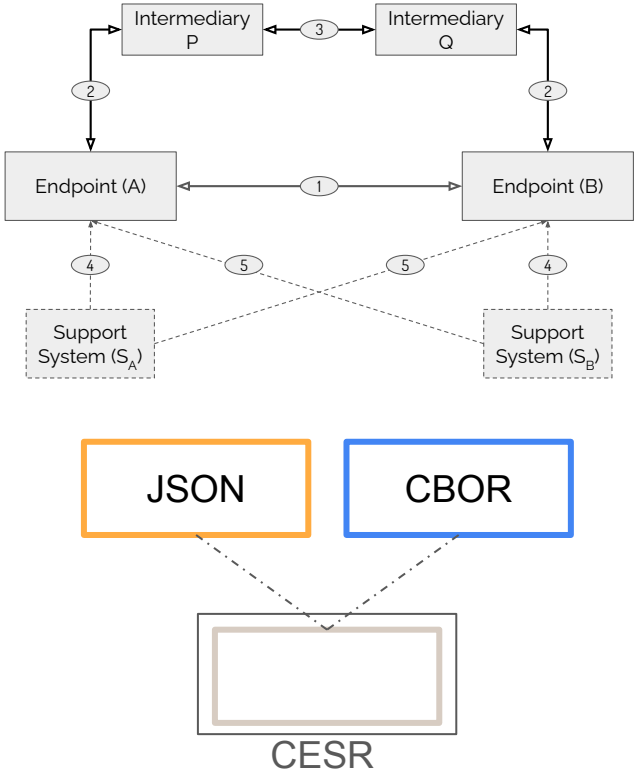
- TSP offers a foundation for managing long term trust relationships needed for autonomous functioning of AI agents
- Clean context and trust signal (‘who said what to whom’) supports reputation and accountability.
- Flexibility and simplicity to integrate with AI Agent protocols such as MCP and A2A



Why is TSP different?

6. A spanning protocol with careful considerations in efficiency, scalability, flexibility, agility

- Practical challenges of protocol design, implementation and deployment
- CESR: Improve data and compute efficiency
- Support multiple data encoding schemes: JSON, CBOR, MsgPack, CESR
- Separation of endpoint, intermediaries and support subsystems
- Rust implementation of core SDK, with multiple language bindings
- Self-encoding of cryptos for agility, e.g. rapid PQC support.

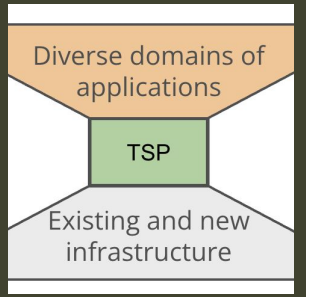


Why is TSP different?

What does TSP NOT do?

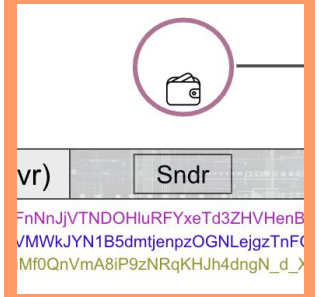
- It does not force a “controller” model (also called “agent”). The higher layer can choose to design any suitable ones based on application logic.
- It does not force a particular types of identifiers (or DIDs, or any particular style of identifiers) - does require it be “verifiable” - therefore VID.
- It does not force a particular application data encoding scheme (e.g. JSON, CBOR, MgPack, CESR) - does use CESR for the envelope
- It does not force a secure compute/storage element (e.g. wallet)
- It does not assume any kinds of networking layer
- It does not force a routing scheme - i.e. can adapt to cloud/mobile infra.
- It does not tie closely to any cryptographic algorithms - agility e.g. PQC

Why is TSP different?



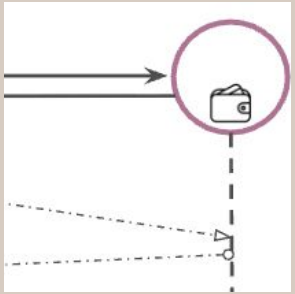
A minimalistic spanning layer

TSP is a general purpose protocol designed for wide use and interoperability.



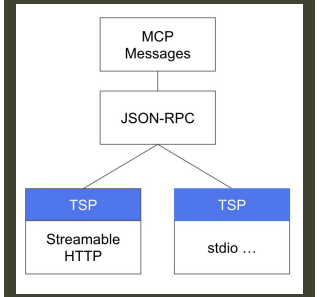
A compact & strong trust primitive

TSP does not compromise in authenticity, confidentiality and meta-data privacy.



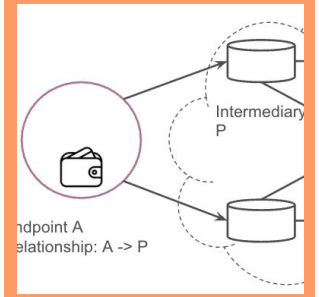
Managing long term trust relationships

TSP is natively designed for managing long term trust relationships.



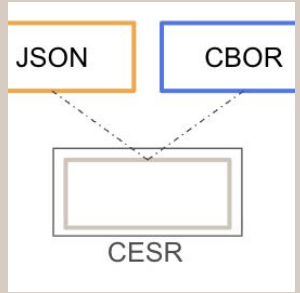
A trust layer for Agents

TSP is therefore very suitable to be the trust layer for AI agent protocols such as MCP or A2A.



Clean metadata privacy

TSP natively supports clean and strong meta-data privacy methods - an ever more challenging problem in the AI era.



Efficiency, scalability, flexibility & agility

TSP incorporates many practical features for application devs and deployment.

Broad Areas of Applications

- Control protocols, e.g. credentials, authentication, authorization
  - Linux kernel group (credentials)
  - Git
  - First Person Project
- Data protocols
  - Strengthening content authenticity together with C2PA for human creators and AI.
  - Matrix
- AI and Agents
  - TMCP, TA2A, Agent Protocols over TSP
- Transports
  - Networks, telcos, industrial devices

Thank you

- TSP SDK Code hosted in this OpenWallet repo:
  - https://github.com/openwallet-foundation-labs/tsp
- TSP Spec authored by Wenjing Chu and Sam Smith:
  - https://trustoverip.github.io/tswg-tsp-specification/

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

- 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 of TSP SDK

Testing environment setup