TSP信任互联协议入门

2024年6月 24日 （v2.0）

AUTHOR: WENJING CHU
Trust over IP Foundation & OpenWallet Foundation

什么是TSP信任互联协议?

TSP是英文Trust Spanning Protocol的缩写，翻译为信任互联协议。其中 'Spanning' 也可以理解为 '跨越'，即跨越不同的信任域的协议。互联网起源于不同的物理网络之间需要 '跨越' 的诉求，比如当时的局域网有以太网（Ethernet）、Token Bus、Token Ring等多种，各自有自己的地址格式、物理介质、和控制协议，加上局域网之间需要另一种广域网来把它们联接到一起，这就衍生了Inter-network 网络之间的网络的概念和协议（即 Internetworking Protocol 缩写为IP），简称Internet。

如果说IP协议联接了不同的网络，那么TSP协议可以理解为联接了不同的信任域。TSP的作者最初把它叫做"信任域互联协议"（Inter-Trust Domain Protocol or ITDP[1]）。什么是信任域呢？信任域可以定义为一个采用统一的数字标识符和信任认证机制的计算机系统的集合。比如采用公开密钥架构（PKI）下的数字证书认证机构（CA）来给每个系统发放X.509格式证书，系统之间通过对X.509证书的认证来验明真实身份，那么这样的一群系统之间就可以互相"信任"了。在这个例子里，所谓的"信任"的基础是大家都可以信任这些CA，尤其是根CA，我们把这样的一群系统叫做一个信任域，信任域之内的系统间的信任是可以有统一标准可以遵循的。

信任域可以有无数多个。即使是大家都采用同样的PKI标准，每个不同的根CA可以派生出一个不同的信任域。同时不少应用场景可能会采用不同的技术标准，比如证书的内容、格式、线下的治理条例、甚至法律规章等都在造就不同的信任域。

单一的信任域不能完全包含整个互联网里无数的系统和多元的应用场景。比如，企业之间可能没有上下附属关系；国际之间可能无法统一对治理规则的认可；针对每个个人的系统（或未来的AI Agent）的数量级会太大，导致集中由CA发布与管理证书的方式不现实成本高；另外单一的CA的

[1] https://github.com/wenjing/Inter-Trust-Domain-Protocol

安全风险可能太高，等等。这些因数导致互联网中实际上采用PKI的系统常常是有限的，而且问题很多。互联网的成功在于其分布式体系结构，PKI的缺点则在于它的集中性，集中化的系统难以规模化，难以接受多元化的应用需求，系统维护成本不断提高，而且有单一系统单点故障的风险。

这些新的挑战需要用更加分布式的信任技术来克服。除去单一的中心化模块才能让整个系统满足互联网级别的规模化要求，满足无数种应用场景的多元化要求，满足区域之间数据治理的合规要求，以及克服单点故障的鲁棒性要求，等等。

但是如果只是采用多种多个信任域，把互联网分割成了一个个的孤岛，那就违背的互联网的初心和优点了。所以我们必须同时发展一个可以让不同的信任域能够互联又保持信任的机制，这就是信任互联协议（Trust Spanning Protocol, or TSP）。

TSP要解决什么问题?

互联网的原始设计本身不包含一个内涵的信任机制，比如IP的地址和数据包内容很容易被篡改，所以发送与接收双方都不能确认IP数据包是不是真的从源地址来，到目标地址去，也不能确认收到的数据是真实数据。因为互联网遍布全球的任何角落，无法在物理网络层面克服这些问题，我们必须在上层协议上来解决。我们把这一类问题归结为真实性问题（Authenticity），包括发送和接收方的身份真实性（Identity Authenticity），也包括数据的真实性（Message Authenticity）。对应地讲，就是我们需要一套身份标识符及其认证机制，加上数据签名（或MAC）及其认证机制。

第二类的问题是数据保密的问题（Confidentiality）。这类问题大部分开发人员和用户都比较熟悉，即所谓的加密机制以保证没有其他第三方可以读到数据包中的内容。也有人把这类问题叫做保护个人隐私，但只包括对数据内容的隐私保护（Content Privacy）。为了准确起见，我们采用私密性（Confidentiality）来表示这一类问题。我们熟悉的TLS协议就是主要用来支持私密性的。私密性一般必须首先满足真实性，没有真实性的话私密性也就无从谈起，比如说A送一个私密信息给B，如果第三方C能够冒充是B，那么什么加密都是没有用的。所以我们需要把Authenticity和Confidentiality绑在一起使用。比如TLS采用X.509格式的证书来验证身份。

那么为什么需要设计一个新的协议而不直接使用成熟的TLS呢？首先，TLS（和相关的HTTPS协议等）采用PKI和基于X.509的证书。如上所述，这样的系统无法满足现代和未来互联网的需求，我们希望采用分布式无中心点（即去中心化）的可认证标识符。其次，现有的TLS实现都只要求服务器端提供证书，不包括客户端，从而导致客户的认证只能依靠微弱的密码方式或损害隐私的联邦认证方式来实现。另外，即使采用双向TLS，对客户端和服务器都发证书，PKI管理的效率很低

，所以证书数据的质量非常微弱，实时性差，安全效益很低而部署成本很高。这也是X.509证书的使用没有遍布互联网的根本原因。我们认为真正有效的解决方案需基于分布式系统。更重要的是，即使在某些应用场景规模不大可以采用PKI下的X.509证书，它所提供的安全系数还是不强。比如TLS在认证了证书之后通过协议产生一个对称的密钥，所以从根本上讲无法区分发送方和接收方，也就不能满足不可抵赖性。这些问题严重地限制可以实现的安全性和隐私保护的要求。

所以，我们需要一套新的更强的安全协议，基于分布式无中心的架构，充分保证真实性和私密性，来作为未来应用的安全基础。这就是TSP协议设计的出发点。

TSP还提供了一套全新的工具来解决第三类问题：元数据的私密性（Meta-data Privacy）。这方面的问题在以前（包括TLS）的技术方案中考虑很少，但现在越来越影响到用户的隐私。由于互联网的底层（比如TCP/IP）没有私密性，即使我们的应用采用了端到端的加密，其数据流的TCP端口和IP地址等元数据还是可以在网上观察到的，并且可以与其他信息联合在一起从而了解到很多个人信息。TSP协议为解决这第三类问题提供了基于中间点（Intermediaries）的解决方案，即可保护元数据的私密性，又可以实现TSP系统的规模化，满足整个互联网规模的需求。

TSP的主要技术特点有哪些?

综上所述，在TSP协议中，真实性是首要保证的，同时可根据实际应用需求选择数据私密性和/或元数据私密性。TSP的主要技术特点可以总结为以下几个方面：

1）多种多类可认证标识符（Verifiable Identifiers），尤其包括去中心化标识符（Decentralized Identifiers，DID）

作为信任域之间的桥梁，TSP是为多种类可认证标识符（Verifiable Identifier）设计的，包括去中心化的标识符（DID）但不仅限于此。比如我们正在开发基于传统X.509格式的did:x509，基于WEB的did:web，以及基于自认证标识符（Self-Certifying Identifier，SCID）的did:webs、did:tdw (trusted web)、和KERI AID等。未来可以出现更多的可认证标识符以满足多元的应用需求，只要它们满足TSP定义的“可认证性”的要求。

2）有方向性的信任关系（Directional Relationship）

A信任B并不代表B信任A，这是个常理，也常常是许多应用的安全和隐私保障机制里重要的区分，但现有的安全体系一般都是不完全区分通讯的双方的。TSP的基础信任架构是一种有方向性的信任关系（Directional Relationship）：A -> B，其中A和B以Verifiable Identifier来代表，而他们之间的通讯通过非对称密钥算法来实现。

3）采用基于非对称密钥的真实性和私密性保障（Public Key Authenticated Encryption）

同时保障真实性和私密性是大部分应用场景的公共要求，但是以前的实现通常最终归结为发送方和接收方之间的共享对称密钥的，从而在信任关系上无法区分他们，也不能满足“不可抵赖性”（Non-Repudiation）。以前这样的对称性设计是因为效益原因，对称性算法远比非对称算法成本低，但也是源于Client/Server模式的信任关系：一般都是服务器方在主导从而私密性不是主要考虑。在TSP架构下，通讯的双方是对称平行的。TSP采用公开密钥认证加密算法（Public Key Authenticated Encryption, PKAE），基于RFC9180（HPKE）定义的格式，同时也支持开源软件中常用的Nacl/Libsodium实现的Sealed Box。这两种实现是基本一致的，我们希望将来他们会最终统一为一种兼容的实现。

PKAE本身仍然有一些值得重视的安全和隐私缺陷，比如在密钥泄露时可能被第三方冒充的弱点（Key Compromise Inpersonation, KCI），TSP通过发送方签名来克服这个弱点。另外TSP采用ESSR算法来加强某些配置下的接收方不可冒充性的功能（Receiver UnForgeability, RUF）等。

结合这些非对称性算法和协议技术，TSP为互联网通讯提供了最安全又最私密的保障。

4）采用嵌套式信息包来加强元数据私密性 （Nested Messages）

随着互联网应用的普及，很多原先不太重要的设计细节的缺点浮出水面。元数据的泄露就是其中之一，包括IP、TCP、HTTPS等等协议的Header中的诸多信息。把这些元信息和其他个人信息交叉索引，尤其是采用AI算法，可以非常准确地还原许多个人信息。所以个人信息的保护常常不能停留在数据内容层面，也需要在元数据层面采取保护措施。TSP的嵌套式信息包可以隐蔽内部真正使用的VID，避免被交叉索引算法所用。

5）采用路由信息包，通过中间点系统保证规模化实现，并同时加强元数据私密性 （Routed Messages）

虽然点对点的信任关系是TSP协议的基础，在实际部署中因为多种原因，引入高性能的中间服务节点是至关重要的：几乎所用的现代互联网应用都采用大型数据中心作为服务器；移动手机应用必须解决间断性联网的问题，某个服务器必须替不能永久联网的节点作为信息存储器；我们也需要解决用户间互相发现的问题，大型中间服务可以帮助这方面的实现；同时网路的路由，以避开出问题的节点和不可信的资源等，仍然需要某种路由协议经过中间节点来解决。

TSP不仅仅引入的中间点的结构，同时设计了可以通过中间点加强对用户元数据私密性保护的路由机制。TSP的路由机制为上层应用提供在互联网基础上的端对端路由信息包（Routed

Messages）通讯，其中的中间节点无需知道确切完整的路由途径，从而为两端的用户提供又高性能又安全私密的通用信息包服务。如果有必要，上层应用还可以将前一节里介绍的嵌套式信息包和路由结合在一起，更加加强路由协议下的元信息私密性。

6）支持最现代的加密和认证算法，支持后量子计算加密和认证算法（Post-Quantum Cryptography, or PQC）

HPKE定义了一套指定加密和认证算法的统一规格，因此TSP可以不仅支持现阶段最新最现代的算法，而且可以在必要时快速地采用新的算法，无需TSP协议本身的改动。这个优点也包括了对后量子密码算法的支持。TSP的另一个重要特点是每个TSP的数据都采用自编码单元（Self Encoding），这更是对协议的未来演变的高度保障（Future-Proof）。

7）采用高效益编码系统

非常高效益的编码系统又是另一个TSP的技术特点。TSP的数据包结构（Message Structure，即用户内容除外的包的结构部分）采用可组合事件流表达式（Composible Event Streaming Representation, CESR）来实现数据结构的线性转化（Serialization）。“可组合”意味着在“编码”和“串联”两个操作之间的可交换性，也就是说，先编码再串联与先串联在编码会得到完全同样的结果。可组合性为软件实现提供很多的方便，包括支持高性能的流处理实现。CESR同时设计了高效的编码表，在TSP这样高程度使用密码操作的协议里，这种效益的成本优势非常明显。另外，如上一节所述，CESR是一个完全自编码的数据格式，它支持更好的未来可扩张可演变性。TSP协议的内容部分（即用户数据主体）则可以是任何常见的协议，包括JSON、CBOR、MsgPak等。这也是自定义格式的一个优点，数据单元可以采用每个应用所需的格式而不必强求统一。

8）与网络载体解耦

TSP可以完全与网络载体解耦，它可以用于任何底层传输协议之上，也可以用在非互联网载体中，比如云存储、Bluetooth、NFC、甚至QR或信件等等。这样的设计让它的应用面非常广，同时开发者可以根据应用要求作合适的调整，又不妨碍协议的信任互联可互操作性。

9）为多种多元的上层应用设计

TSP的设计遵循最小化的原则。作为一个互联协议，TSP对应用层提出的限制越小越好，TSP才能越被普遍地采用。TSP为确实需要的信任任务提供必须的互操作性服务，其他功能一律排除在外，留给应用按具体情况来实现。这样的设计理念将TSP和许多其他类似的协议区分开来，TSP只是一个工具，真正的应用将会是应用层软件和TSP的结合。

10）开源协议实现和编程界面

TSP的开源实现也是一个重要的特性。TSP的SDK使用Rust语言编程，大幅提高加密和认证代码的可靠性和安全性，防止常见的C语言的内存安全性弱点，同时又支持高效高性能。为了给常见的开发场景提供方便，TSP的开发社区也将提供相关的编程环境Bindings，比如JS、Python、WASM、C、Android、iOS等等。TSP的SDK为上层软件提供方便又简易安全的编程界面，让原先非常复杂又高风险的信任协议的编程变得简单清晰。

哪里可以找到关于TSP协议的学习材料？如何参与TSP协议的开发和应用社区？

- TSP协议的标准草案在Trust over IP (ToIP) Foundation的GitHub上发布

W. Chu, S. Smith, “Trust Spanning Protocol (TSP) Specification”， Implementer’s Draft, https://trustoverip.github.io/tswg-tsp-specification。

有针对标准草案的问题的话可以参加该GitHub上的讨论。

- TSP协议的Rust语言实现和其他相关的软件则在OpenWallet Foundation的TSP Lab项目中

项目GitHub： https://github.com/openwallet-foundation-labs/tsp。

欢迎大家参与GitHub上的代码讨论与开发工作。这个项目包含TSP协议的各个方面，不仅仅是协议本身，也包括多种可认证的DID实现，不同的传输协议支持，不同的编程语言的Bindings，多种应用，多种安全算法和编码等等多个方面。

- 开放鸿蒙（OpenHarmony）开源社区的WEB3 TSG（技术指导小组）（中文）

欢迎对TSP的研究、开发、与应用感兴趣的同业朋友与大学师生参与，有关信息可参见: https://www.openharmony.cn/techCommittee/aboutTSG/。

其它进一步的技术信息请参照后面附录里的参考材料。

附录-参考材料：

【1】W. Chu, S. Smith, “Trust Spanning Protocol (TSP) Specification”, Implementer’s Draft, https://trustoverip.github.io/tswg-tsp-specification.

【2】The OpenWallet Foundation TSP Lab Project, https://github.com/openwallet-foundation-labs/tsp.

【3】R. Barnes, K. Bhargavan, B. Lipp, C. Wood, RFC 9180, “Hybrid Public Key Encryption”, https://datatracker.ietf.org/doc/rfc9180/. Feb 2022.

【4】Libsodium (Nacl) Documentation， https://doc.libsodium.org/.

【5】J. H. An, “Authenticated Encryption in the Public-Key Setting: Security Notions and Analyses”, Cryptology ePrint Archive, Paper 2001/079, https://eprint.iacr.org/2001/079.

【6】M. Spony, A. Guy, M. Sabadello, D. Reed, et al, “Decentralized Identifiers (DIDs) v1.0”, https://www.w3.org/TR/did-core/. W3C Recommendation, 19 July 2022.

【7】S. Smith, K. Griffin, “Composable Event Streaming Representation (CESR)”, v1.0, https://trustoverip.github.io/tswg-cesr-specification/.

【8】D. Hardman, S. Curran, S. Curren, et al, “ Peer DID Method Specification”, v1.0, https://identity.foundation/peer-did-method-spec/.

【9】S. Curran, J. Jordan, A. Whitehead, B. Richter, “Trust DID Web - did:tdw”, Draft, https://bcgov.github.io/trustdidweb/.

【10】P. Feairheller, D. Hardman, S. Smith, L. Byrd, et al, “ToIP did:webs Method Specification v0.9.15”, https://trustoverip.github.io/tswg-did-method-webs-specification/.

【11】M. Prorock, O. Steele, O. Terbu et al, “did:web Method Specification”, 06 May 2023, https://w3c-ccg.github.io/did-method-web/.

【12】E. Scouten, W. Chu (co-Chairs), et al, “did:x509 Method Specification”, Work in progress in the Trust over IP (ToIP) X.509 based DID Task Force X5VTF. https://github.com/trustoverip/tswg-did-x509-method-specification.