Department of Mathematics and Computer Science
Coding Theory and Cryptology Group

Secure Sessions for Ad Hoc Multiparty Computation in MPyC

Master's thesis

Emil Nikolov

Id nr: 0972305
`emil.e.nikolov@gmail.com`

Supervisor : Dr. ir. L.A.M. (Berry) Schoenmakers

May 23, 2023

Contents

List of Abbreviations

E^3 Extensible Evaluation Environment. 15

ACL Access Control List. 12

CGNAT Carrier-Grade NAT. 8

CIDR Classless Inter-Domain Routing. 12

DERP Designated Encrypted Relay for Packets. 6, 12

DNS Domain Name System. 5

DTLS Datagram Transport Layer Security. 8

ECDH Eliptic Curve Diffie-Hellman. 8

HTTP HyperText Transfer Protocol. 5, 8

IP Internet Protocol. 5, 11

IPSec Internet Protocol Security. 8, 9

ISP Internet Service Provider. 3, 4, 8

LAN Local Area Network. 4, 5, 9

NAT Network Address Translation. 6

NAT-PMP NAT Port Mapping Protocol. 7

OSI Open Systems Interconnection. 3, 4

P2P Peer to Peer. 11, 12

PCP Port Control Protocol. 7

PDU Protocol Data Unit. 3

SSL Secure Sockets Layer. 8

STUN Session Traversal Utilities for NAT. 7

TCP Transmission Control Protocol. 5

TLS Transport Layer Security. 8

TURN Traversal Using Relays around NAT. 6

UDP User Datagram Protocol. 5

UPnP Universal Plug and Play. 7

URL Universal Resource Locator. 5

VPN Virtual Private Network. 8, 9, 11, 12

WWW World Wide Web. 4

List of Figures

Chapter 1

Introduction

Chapter 2

Background information

include a section on MPC?

This chapter provides background information on the challenges of Internet communications between multiple independent parties. It also presents a systematic overview of the available solutions, using the Open Systems Interconnection (OSI) reference model as a conceptual framework. Section 2.1 briefly explores the fundamentals of the Internet, its protocols, the limitations for peer-to-peer protocols, and some of the approaches to overcome them. Section 2.2 discusses higher-level overlay networks that build on top of the lower-level protocols from section 2.1.

2.1 Internet Fundamentals

Is there a more fitting section name than "The Internet"

The Internet is a global network that consists of numerous interconnected computer networks spanning billions of host devices owned by diverse parties from around the world. Various communication protocols and the physical infrastructure that connects the individual networks are key components of the Internet. Packet switching

The Internet Protocol Suite utilizes packet-switching, meaning that application layer communications are broken into smaller discrete packets that are handled individually by the network infrastructure. A protocol typically has a Protocol Data Unit (PDU) that describes the information that c has a header i A packet contains multiple nested a header and a payload, where the header identifies the protocol that that which protocol which protocols are involvedhierarchically organized information from the protocols. identifies information that identifies its from the various communication protocols with each protocol having its own PDU. The various protocols have their own PDUs that have a header that ide

Different tiers of Internet Service Providers (ISPs) deploy and manage sections of the infrastructure and also maintain links to each other.

Communication protocols are usually organized into abstraction layers based on the scope of their functionality. There are several reference models that communicate using the protocols of the Internet Protocol Suite, commonly known as TCP/IP. n

The Internet is a global multi-tiered computer network of billions of host devices that communicate using the protocols of the Internet Protocol Suite (TCP/IP). ISPs are responsible for managing different sections of the infrastructure that connects the Local Area Networks (LANs) of various end-users including households and enterprises. The World Wide Web (WWW) or simply the Web is a collection of interconnected documents, e.g. HTML Web Pages, available on the Internet and is typically accessed via a user-agent software such as a ***Web Browser***. The term "the Web" is sometimes used interchangeably with the Internet, but the Internet supports other services as well, e.g. file transfer (FTP), email (SMTP), instant messaging, remote access (SSH) and others.

Conceptual frameworks like the OSI model are useful for understanding the objectives and functions of communication protocols. Figure 2.1 describes the responsibilities of the 7 layers of the OSI model and how they relate to the TCP/IP model used by the Internet Protocol Suite. The newer TCP/IP model only recognizes 4 layers as it merges the OSI Session (L5) and Presentation (L6) layers into the Application layer (L7), as well as the Physical layer (L1) into the Data link layer (L2). While the TCP/IP model is a more accurate representation of the Internet, the 7-layer numbers of the OSI model are still widely used in the literature.

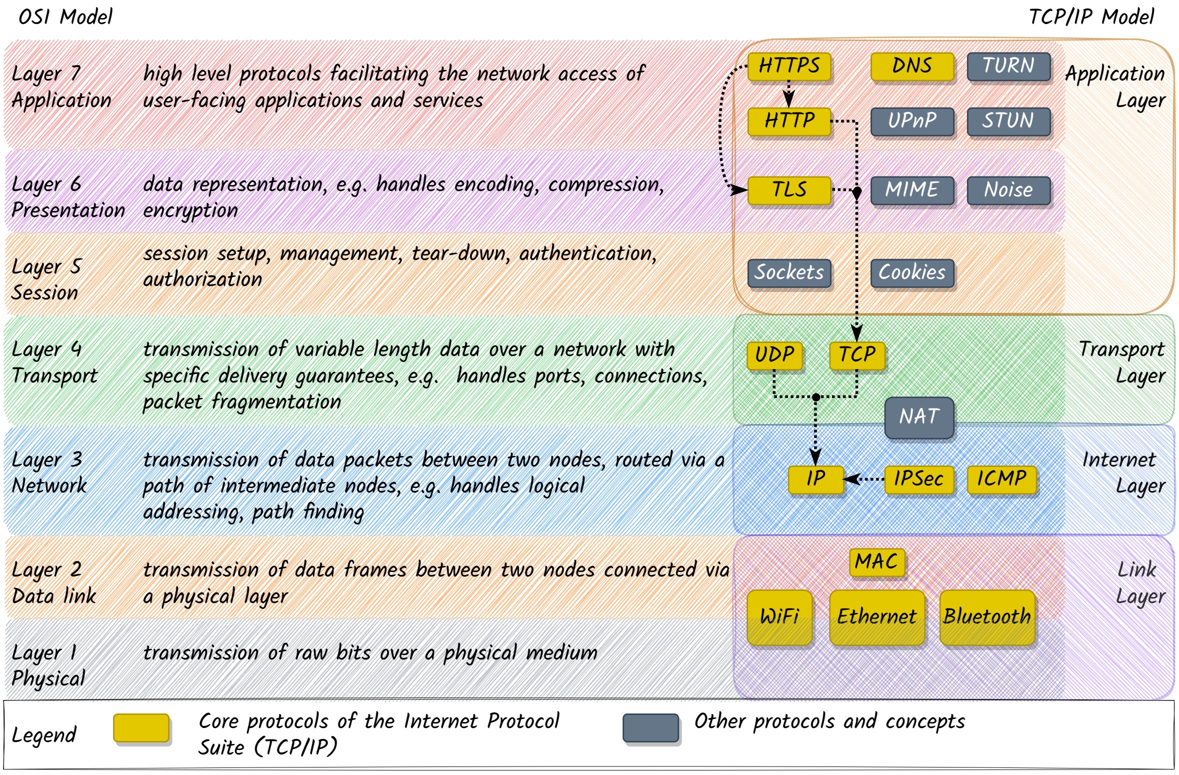
OSI Model
TCP/IP Model
Layer 7 Application
high level protocols facilitating the network access of user-facing applications and services
Layer 6 Presentation
data representation, e.g. handles encoding, compression, encryption
Layer 5 Session
session setup, management, tear-down, authentication, authorization
Layer 4 Transport
transmission of variable length data over a network with specific delivery guarantees, e.g. handles ports, connections, packet fragmentation
Layer 3 Network
transmission of data packets between two nodes, routed via a path of intermediate nodes, e.g. handles logical addressing, path finding
Layer 2 Data link
transmission of data frames between two nodes connected via a physical layer
Layer 1 Physical
transmission of raw bits over a physical medium
HTTPS
DNS
TURN
HTTP
UPnP
STUN
TLS
MIME
Noise
Sockets
Cookies
UDP
TCP
NAT
IP
IPSec
ICMP
MAC
WiFi
Ethernet
Bluetooth
Application Layer
Transport Layer
Internet Layer
Link Layer
Legend
Core protocols of the Internet Protocol Suite (TCP/IP)
Other protocols and concepts

Figure 2.1: OSI model mapping of the Internet Protocol Suite

add a paragraph that describes what the rest of the section will contain

2.1.1 Communication Protocols

- packet switching
- data is split into small packets
- each packet is handled individually
- each packet contains data from the protocols
-

The **Internet Protocol (IP)** [81] is a Network layer (L3) protocol of the Internet Protocol Suite that is responsible for transferring datagrams between devices across the boundaries of their LANs by possibly routing them via multiple intermediate devices (e.g. routers). A datagram is a self-contained unit of data, typically associated with connectionless protocols that provide no guarantees for delivery or ordering (e.g. IP, UDP). . IP datagrams have a header that contains fields such as the **IP addresses** of its source and destination, and a payload that encapsulates the data from the Transport Layer (L4) protocols. A ***router*** is a device that is part of multiple networks and relays datagrams between them based on a routing table that maps IP address ranges to networks.

packet is the physical envelope of the IP datagram

show a diagram of the internet with multiple local networks

User Datagram Protocol (UDP) [80] and **Transmission Control Protocol (TCP)** [Edd22] are Transport layer (L4) protocols that employ 16-bit port numbers to enable multiple applications on the same host to establish their own communication channels while sharing an IP address. Both protocols UDP offers faster communication, but only provides best-effort delivery, while TCP is a reliable transport protocol with stronger delivery guarantees at the expense of higher network latency. TCP maintains stateful connections that handle error detection and correction, packet ordering, flow control, acknowledgments and retransmissions in case packets are lost during transmission.

IP multiplexing via ports, TCP segments and segmentation, UDP no segmentation

HyperText Transfer Protocol (HTTP) is an Application layer (L7) protocol that enables interactions on the Web between web servers and clients (e.g. browsers). Traditionally, HTTP offers stateless request/response , but can also . Similar to other L7 protocols, it uses ***Universal Resource Locators (URLs)*** for locating resources using the format `scheme://host:port/path?query=value#fragment`, e.g. http://www.example.com:80/path/to/file.html. It is built on top of TCP and provides several features such as:

- Request Methods - used by the client to specify the action to perform on the resource behind the given URL, e.g. GET, POST, PUT, DELETE, etc.
- Headers - used to provide additional information about a request or response, e.g. Content-Type, Authorization, Cache-Control
- Status codes - used to indicate the result of a request, e.g. if it was successful (200), or if the resource is missing (404)
- Cookies - used to include stateful information about the user kept at the client-side
- Caching - used to specify that the result of a request can be cached for a certain time to avoid repeating the request's action.

The ***Domain Name System (DNS)*** operates at the Application Layer (L7) and allows the conversion of human-readable domains to IP addresses, e.g. `google.com` to `142.250.179.142`.

2.1.2 Host Addressing Challenges

The version of the Internet Protocol, that was originally deployed globally (IPv4), uses 32-bit numbers as IP addresses, allowing for around 4 billion unique addresses. Due to the popularity of the Internet, there are many more devices than available IPv4 addresses, which has caused challenges. IPv6 is a newer version of the protocol that uses a larger 128-bit address space which is sufficient for assigning 100 addresses for each atom on Earth. However, its adoption has been slow, as according to Google[] as of 2023 around 41% of their users access their services over IPv6. Additionally, despite that IPv6 allows for all devices to be addressable on the Internet, for security reasons, most of them would use firewalls to block incoming remote traffic that is not associated with outgoing connections.

A widespread solution to the addressing problem is **Network Address Translation (NAT)**. It allows many devices without globally unique IP addresses to initiate connections to publicly addressable devices on the Internet via a limited number of gateways that must have globally unique IP addresses. A NAT gateway replaces the local source IP address of each outgoing IP datagram with its own public IP address before passing it on to the next link on the way to the destination while maintaining a mapping between the source and destination IPs in a translation table. The destination host can then address its responses back to the NAT gateway's public IP address, which in turn replaces its own IP from the incoming datagrams with the IP of the local device and forwards them to it. If the IP datagrams encapsulate TCP/UDP packets, the gateway additionally rewrites the source and destination ports, which means that NAT techniques can be placed somewhere between Layers 3 and 4 of the OSI model.

The effect of NAT on connectivity is similar to an IPv6 firewall as they both allow devices on a local network to initiate bidirectional communication to remote devices with public IP addresses, but connections cannot be natively initiated by the remote devices. As Figure 2.2 shows, it follows that when two devices are behind separate NATs, neither can contact the other first. **Client/Server** communication is less affected by this limitation because Servers are usually deployed to a public IP address that can be contacted by Clients with local IP addresses. **Peer-to-Peer** communication, however, is more challenging because the peers are often devices in separate residential networks behind different NATs. Several **NAT traversal** techniques try to solve this with different performance tradeoffs and success that varies depending on the NAT [**natBehaviorRFC**] and its behavior when mapping ports and IP addresses.

describe some of the nat behaviors, e.g. if the source IP address/port vary per destination are changed depending on the destination/port mapping algorithms, if it maps ports, IPs, whether the mapped IPs are different per destination and others.

One approach based on the Client/Server model is to use a publicly addressable **relay** server that is contacted by the NATed devices and then forwards the Peer-to-Peer traffic to the intended recipient. Compared to direct communication, relaying results in a higher network latency due to the longer path that each packet must travel. Maintaining a relay server requires some technical expertise and may be costly depending on the expected throughput. Despite the drawbacks, relaying works under most networking scenarios and is therefore often used as a fallback in case all other approaches fail to find a direct path. Protocols such as **Traversal Using Relays around NAT (TURN)** [Red+20] and **Designated Encrypted Relay for Packets (DERP)** [**derpDocs**] can be used to securely implement relaying.

The NAT gateway in many residential networks is a Router device under the customer's control that has a statically or dynamically assigned public IP address. Most routers can be manually

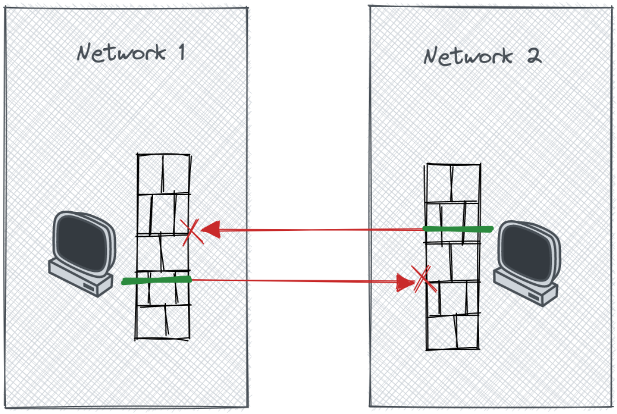

Figure 2.2: Two parties behind separate NATs

configured through their admin page to forward all traffic that arrives at a given port to a specific device on the local network. Remote applications can then initiate a connection to the local device if they know the IP address of the router and the forwarded port. The manual configuration, however, can be inconvenient and many users may be unaware of that setting because it is not necessary for the more straightforward Client/Server communications. Some routers also support programmatic configuration of port forwarding via a Layer 7 protocol like **Universal Plug and Play (UPnP)** or its successors **NAT Port Mapping Protocol (NAT-PMP)** and **Port Control Protocol (PCP)**. However, these protocols are not always supported and are often disabled by the local network administrators due to security concerns related to bugs in their implementation, vulnerable IOT devices on the local network or malicious programs being able to expose local devices to the internet.

An efficient NAT traversal approach that works with some types of NATs is to use **Session Traversal Utilities for NAT (STUN)** [**stunRFC**] in combination with UDP hole punching (Figure 2.3). STUN is a protocol operating at Layer 7 that allows a client application to detect the presence of NAT gateways on the network path to a public STUN server, and identify their types and the public IP address that they map to externally. An application sends UDP datagrams to the STUN server and it responds with the source IP address and port specified inside the datagrams. The application can compare its own endpoint with the source endpoint observed by the STUN server and if the values differ, it can be inferred that they were rewritten by a NAT. Additional STUN servers are contacted to determine if the NAT maps IPs and ports in a predictable fashion. UDP hole punching is a related technique that, depending on the NAT types, can allow direct communication between two applications behind separate NATs. The applications must discover each other's externally mapped endpoints, perhaps via the STUN server. If the NATs use the same external port regardless of the remote destination, the two applications can simultaneously send UDP packets to each other's external endpoints. Their respective NATs will see the outgoing connection to the other peer - the "punched hole" - when the incoming traffic arrives from it and forward it correctly. NATs that map different ports per remote destination sometimes allocate port numbers predictably, which can be used by the peers to try to guess the port that will be opened by the opposing side's NAT.

If, else?

Talk about traversal friendly NATs and unfriendly NATs
add a bullet list with the steps in the STUN process and relate them to

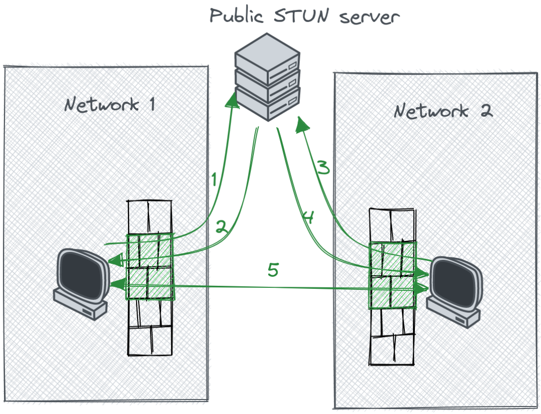

Figure 2.3: NAT traversal via STUN

In mobile networks like 4G and 5G, the ISP often utilizes a **Carrier-Grade NAT (CGNAT)** as part of their infrastructure, while all devices under the user's control, including the router, only have local IP addresses. STUN techniques would fail to discover a direct path between two parties behind separate CGNATs or other unpredictable NAT algorithms. The only remaining possibility is to relay the traffic via a publicly reachable third-party host using a protocol similar to TURN.

only 65000 ports per IP address means that CGNATs that provide more than 65000 connections from client devices require more than one public IP address hairpinning - Hairpinning, also known as NAT loopback or NAT reflection, is a technique used by NAT devices to allow hosts on a private network to access a public server using its public IP address. Without hairpinning, the NAT device would not recognize the connection as a loopback connection and would route it to the public network, causing the connection to fail. With

2.1.3 Communication Security

Transport Layer Security (TLS) and its precursor Secure Sockets Layer (SSL) provide secure communications to Application Layer (L7) protocols. to on top of a reliable transport protocol like TCP; Datagram Transport Layer Security (DTLS) is a related protocol that works with connectionless protocols like UDP. TLS does not strictly fit in any single OSI layer but it is usually placed somewhere between the Transport Layer (L4) and the Presentation Layer (L6). It is rather complex because it needs to support many possible use cases while remaining backward compatible. The **Noise Protocol Framework** [Per18] is a more recent effort that applies the ideas of TLS in a simplified way by serving as a blueprint for designing use-case specific protocols for establishing secure communication channels based on Eliptic Curve Diffie-Hellman (ECDH) handshake patterns. It powers the end-to-end encryption in messaging applications such as WhatsApp and Signal, and Virtual Private Network (VPN) software such as WireGuard and Nebula.

HTTPS is an extension to HTTP that uses TLS for encryption.

Internet Protocol Security (IPSec) is a protocol suite for encrypting the IP datagrams between two hosts. It was originally developed as part of IPv6 but can also be used with IPv4. IPSec is similar in purpose to TLS but operates at the Network Layer (L3).

- **Internet Protocol Security (IPSec)**
  - Layer 3 protocol suite part of the Internet Protocol Suite
  - used inside VPN software
  - has implementations in both user and kernel space as well as hardware implementations
  - rewrites and encrypts the IP headers and payloads
  - virtual routing table
  - Initially was built into IPv6, separate from IPv4

The Internet Protocol Security (IPSec) protocol suite is designed specifically to provide secure communication between two hosts at the network layer (L3) of the OSI model. IPSec can be used to secure traffic between two hosts, between a host and a network, or between two networks. It provides a range of services, including data confidentiality, data integrity, and authentication, and is often used in Virtual Private Network (VPN) connections to create secure tunnels between remote networks. IPSec operates in either Transport or Tunnel mode, and can use either the Authentication Header (AH) or Encapsulating Security Payload (ESP) protocols to provide its security services.

2.2 Overlay Networks

- the low-level solutions from the previous section are complex to set up.
- overlay networks package some of those solutions for a specfic use case Most overlay networks use a combination of the NAT traversal techniques mentioned previously. They can be placed in Layers 2, 3 or 7. Layer 2 overlays act as a virtual network switch, while Layer 3 overlays act as a virtual network router. Layer 7 overlays are implemented in user-space as libraries or applications that run on top of the network stack of the host operating system. Layer 2 and 3 overlays can either be implemented as kernel modules or as user-space applications that use a **TUN/TAP** driver to interface with the kernel.

Figure 2.4 shows an approximate OSI model mapping of several protocols and network overlay solutions from the point of view of the systems that use them and the arrows show dependency relations between them.

2.2.1 TUN/TAP driver

- Layer 2 vs Layer 3 Networks
  - Layer 2 overlays bridge networks
    - ∗ virtual network switch
    - ∗ remote machines are on the same virtual LAN and can share the same IP address range
    - ∗ allows broadcast/multicast
    - ∗ TAP driver
  - Layer 3 overlays route traffic between separate local networks
    - ∗ virtual network router
    - ∗ remote machines are on separate LANs
    - ∗ simpler to configure
    - ∗ TUN driver

2.2.2 Traditional VPNs

explain the names

tls vs ipsec vpns. TLS vpns offer a virtual network interface at layer 3, but run over L7 TLS

- The term “VPN” is somewhat overloaded as it can refer to different related concepts.

VPNs are implemented as Layer 2 or 3 network overlays. They are commonly used for securely connecting machines from different LANs. They provide software emulation of a network

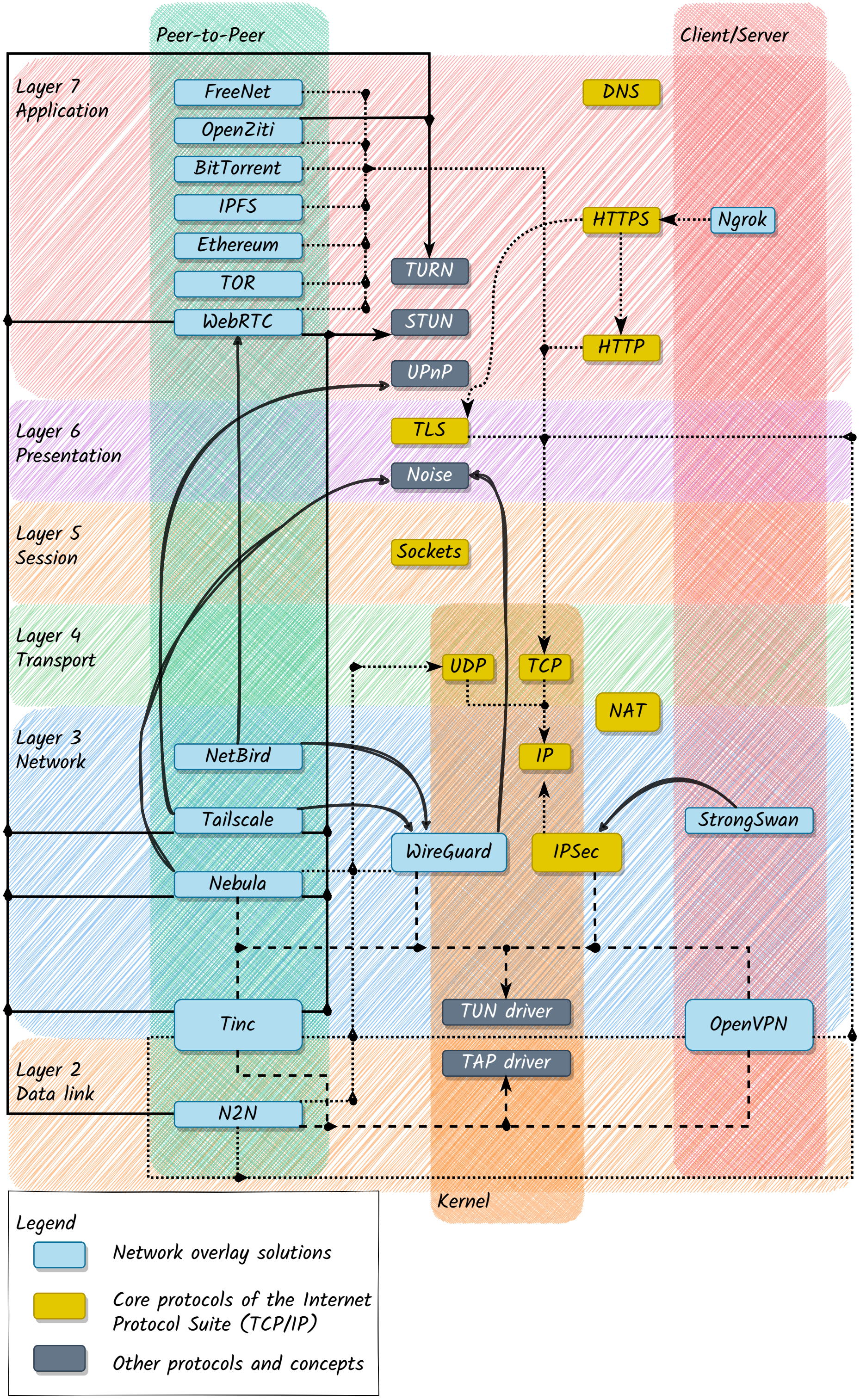

Figure 2.4: OSI model mapping of various protocols

interface controller via a TUN/TAP driver on the operating system level and allow other software to transparently use the functionality of the Internet Protocol (IP) suite without requiring extra changes. Traditional VPNs such as IPSec [**ipSecDocs**] and OpenVPN [Ope22] use a centralized service that all (encrypted) client communications must pass through. This introduces a single point of failure and a potential bottleneck that might negatively impact the performance of the multiparty computations due to their Peer to Peer (P2P) nature.

2.2.3 WireGuard

WireGuard [Don17] is a more recent protocol with a design informed by lessons learned from IPSec and OpenVPN and a key management approach inspired by SSH. It is a lower-level protocol that focuses on configuration simplicity while network topology, peer discovery and key distribution are left as a responsibility of higher-level systems that use it as a building block. Wireguard is implemented as a Layer 3 overlay over UDP tunnels. WireGuard has both user-space implementations that use a TUN driver and also has direct support built into the Linux Kernel since version 5.6 (May 2020). The kernel implementation allows for better performance because it does not need to copy packets between the kernel and user-space memory.

The snippets below show a minimal set of configuration options that need to be provided for two peers to be able to form secure tunnels with each other.

```
# peer1.conf
[Interface]
Address = 101.0.0.1/32
ListenPort = 53063
PrivateKey = ePTiXXhHjvAHdWUr8Bimk30n0gh3m241RAzsN0JZDW0=

[Peer]
PublicKey = BSn0ejd1Y3bKuD+Xpg0ZZe0f+Ies/oql0NZxw+S0mkc=
AllowedIPs = 101.0.0.2/32
Endpoint = peer1.example.com:38133
```

```
# peer2.conf
[Interface]
Address = 101.0.0.2/32
ListenPort = 38133
PrivateKey = sN/d6XUPEVPGSziVgCC0n0ivDK+qAoYC3nxnssQ5Rls=

[Peer]
PublicKey = e/TxvPmrgcc1G4cSH2bHv5J0PRHXKjYxTFoU8r+G93E=
AllowedIPs = 101.0.0.1/32
```

Each peer has a public/private key pair that is used for authentication and encryption based on the Noise Protocol Framework [Per18]. The `Address` field specifies the virtual IP address that the local network interface will use, while the `AllowedIPs` field specifies what virtual IP addresses are associated with a peer's public key. A peer's `Endpoint` field specifies the URL at which it can be reached. Only one of the peers must be configured with a reachable endpoint for the other one. In the above example once `peer1` initiates communication with `peer2`, `peer2` will learn the current endpoint of `peer1` and will be able to communicate back with it.

Classless Inter-Domain Routing (CIDR) is a common notation for describing IP address ranges, e.g. ‘192.168.0.1/16‘, where the number after the slash describes the bit-length of the fixed prefix for a subnet.

2.2.4 Mesh VPNs

- Tinc
- N2N
- Tailscale
- Nebula
- ZeroTier

Mesh VPNs such as Tinc [Sli22], Tailscale [Tai] and Nebula [Def22] utilize NAT Traversal techniques to create direct P2P links between the clients for the data traffic. Authentication, authorization and traffic encryption are performed using certificates based on public key cryptography.

All three are open-source, except Tailscale’s coordination service which handles peer discovery and identity management. Headscale [Fon22] is a community-driven open-source alternative for that component. Tinc is the oldest of the three but has a relatively small community. It is mainly developed by a single author and appears to be more academic than industry motivated. Nebula and Tailscale are both business driven. Tailscale was started by some high-profile ex-googlers and is the most end-user-focused of the three, providing a service that allows people to sign up using identity providers such as Google, Microsoft, GitHub and others. They also provide an Admin console that allows a user to easily add their personal devices to a network or share them with others. It also has support for automation tools like Terraform for creating authorization keys and managing an Access Control List (ACL) based firewall. Nebula was originally developed at the instant messaging company Slack to create overlay networks for their cross-region cloud infrastructure, but the authors later started a new company and are currently developing a user-centric platform similar to Tailscale’s. Nebula is more customizable than Tailscale and since it is completely open-source it can be adapted to different use cases, but it is also more involved to set up. A certificate authority needs to be configured for issuing the identities of the participating hosts. Furthermore, publicly accessible coordination servers need to be deployed to facilitate the host discovery. Tailscale employs a distributed relay network of Designated Encrypted Relay for Packets (DERP) servers, while Nebula can be configured to route via one of the other peers in the VPN.

2.2.5 Layer 7 overlays

WebRTC is

- WebRTC
  - Uses STUN/TURN
  - d
- OpenZiti
  - uses relays
- ngrok
- TOR
- BitTorrent
- IPFS
- Ethereum

- Teleport
- Freenet

Chapter 3

Testing methodology

In the following chapters, we will design and implement several solutions for ad hoc MPC sessions based on a subset of the previously discussed related works:

- Internet protocol
- Wireguard
- Tailscale
- Headscale
- ? Headscale with DID identity?
- ? WebRTC?
- A custom solution that automates the WireGuard configuration by visiting a web page

Additionally, we will analyze and compare them in terms of performance, security and usability

3.1 Measuring performance

During the preparation phase of the project, we developed the Extensible Evaluation Environment (E^3) framework which simplifies and automates the process of deploying machines in different geographical regions, connecting them via an overlay network and executing multiparty computations between them, where each machine represents a different party.

To summarize, E^3 is a set of scripts that use several automation tools:

- Terraform - declarative provisioning
- NixOS - declarative Linux distribution
- Colmena - declarative deployment for NixOS
- PSSH - parallel execution of remote scripts over ssh
- DigitalOcean - a cloud provider

It allows us to quickly provision cloud virtual machines in multiple regions and reproducibly deploy all necessary software for running a multiparty computation over a chosen network overlay solution. The source code of E^3 can be found on GitHub

Each solution will be deployed using the E^3 framework and the performance will be quantitatively measured in terms of the time it takes to execute several MPyC demos. The selected demos have different complexities in terms of communication rounds and message sizes which will allow us to observe their impact on the overall performance.

1. Secret Santa - high round complexity with small messages
2. Convolutional Neural Network (CNN) MNIST classifier - low round complexity with large messages

The demos will be configured at three different input size levels

- Low,
- Medium
- High

Furthermore, the demos will be executed in several networking scenarios:

1. 1-10 parties in the same geographic region
2. 1-10 parties evenly distributed across two nearby regions
3. 1-10 parties evenly distributed across two distant regions
4. 1-10 parties distributed across multiple distant regions

3.2 Security

We will analyze aspects such as

- key distribution
- trust model - are there any trusted third parties and what would be the consequences if they are corrupted or breached
- traffic encryption
- identity strength

3.3 Usability

For each solution, we will describe the steps that the parties need to perform to execute a joint multiparty computation. Those steps will be analyzed in terms of:

- Complexity - how much technical expertise is expected from the parties to be able to execute the steps
- Initial effort - how much effort is each party expected to put in preparing for their first joint computation
- Repeated effort - after the initial setup, how much effort is required to perform another computation
  - with the same set of parties
  - with another set of parties
- Finalization effort - how much effort is required to finalize the MPC session once it is complete and clean up any left-over artifacts or resources so that the machine of each party is in its original state

Chapter 4

Internet Protocol based solution

This solution focuses on directly using the internet protocol without involving an overlay network. Our goal is to analyze the implications of using only the functionalities that MPyC directly supports to serve as the reference for our later experiments.

4.1 Implementation details

We will manually set up the multiparty computations via the public IP addresses of the machines and DNS.

4.2 Performance analysis

4.3 Security analysis

4.4 Usability analysis

Chapter 5

WireGuard based solution

This solution creates an overlay network by manually configuring WireGuard on each machine.

5.1 Implementation details

5.2 Performance analysis

5.3 Security analysis

5.4 Usability analysis

Chapter 6

Tailscale based solution

Tailscale is a VPN solution that configures a mesh of direct Wireguard tunnels between the peers.

6.1 Implementation details

6.2 Performance analysis

6.3 Security analysis

6.3.1 Trust model

There is a centralized service that deals with the key distribution, which needs to be trusted to provide the correct public keys for the correct parties

6.3.2 Identity

Identity is based on third party identity providers such as Microsoft and GitHub

- Magic DNS
-

6.4 Usability analysis

With tailscale each party needs to

- register a Tailscale account
- Download and install tailscale on the machine they want to run a multiparty computation
- Run tailscale on their machine and logs into their account in order to link it to their own Tailnet
- Share their Tailscale machine with the Tailnets of each of the other parties
- Download the demo they want to run
- Form the flags for running the chosen demo

  - add -P $HOST:$PORT for each party using their Tailscale hostname/virtual IP
- Run the demo

Chapter 7

Headscale based solution

This solution is similar to the Tailscale one, but it uses Headscale - a self-hosted open-source alternative to the closed-source Tailscale coordination service.

7.1 Implementation details

7.2 Performance analysis

7.3 Security analysis

7.3.1 Trust model

There still is a centralized service like in the Tailscale solution, but here it is self-deployed.

7.3.2 Identity

7.4 Usability analysis

Bibliography

[] *IPv6 – Google*. URL: https://www.google.com/intl/en/ipv6/statistics.html (visited on 05/21/2023) (cit. on p. 6).

[80] *User Datagram Protocol*. Request for Comments RFC 768. Internet Engineering Task Force, Aug. 1980. 3 pp. DOI: 10.17487/RFC0768. URL: https://datatracker.ietf.org/doc/rfc768 (visited on 05/14/2023) (cit. on p. 5).

[81] *Internet Protocol*. Request for Comments RFC 791. Internet Engineering Task Force, Sept. 1981. 51 pp. DOI: 10.17487/RFC0791. URL: https://datatracker.ietf.org/doc/rfc791 (visited on 05/14/2023) (cit. on p. 5).

[Def22] Defined. *Nebula: Open Source Overlay Networking | Nebula Docs*. 2022. URL: https://docs.defined.net/docs/ (visited on 12/01/2022) (cit. on p. 12).

[Don17] Jason A. Donenfeld. "WireGuard: Next Generation Kernel Network Tunnel". In: *Proceedings 2017 Network and Distributed System Security Symposium*. Network and Distributed System Security Symposium. San Diego, CA: Internet Society, 2017. ISBN: 978-1-891562-46-4. DOI: 10.14722/ndss.2017.23160. URL: https://www.ndss-symposium.org/ndss2017/ndss-2017-programme/wireguard-next-generation-kernel-network-tunnel/ (visited on 09/28/2022) (cit. on p. 11).

[Edd22] Wesley Eddy. *Transmission Control Protocol (TCP)*. Request for Comments RFC 9293. Internet Engineering Task Force, Aug. 2022. 98 pp. DOI: 10.17487/RFC9293. URL: https://datatracker.ietf.org/doc/rfc9293 (visited on 05/14/2023) (cit. on p. 5).

[Fon22] Juan Font. *Juanfont/Headscale*. Dec. 6, 2022. URL: https://github.com/juanfont/headscale (visited on 12/06/2022) (cit. on p. 12).

[Ope22] OpenVPN. *Community Resources*. OpenVPN. 2022. URL: https://openvpn.net/community-resources/ (visited on 11/30/2022) (cit. on p. 11).

[Per18] Trevor Perrin. "The Noise Protocol Framework". In: (July 1, 2018) (cit. on pp. 8, 11).

[Red+20] Tirumaleswar Reddy.K et al. *Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)*. Request for Comments RFC 8656. Internet Engineering Task Force, Feb. 2020. 79 pp. DOI: 10.17487/RFC8656. URL: https://datatracker.ietf.org/doc/rfc8656 (visited on 04/24/2023) (cit. on p. 6).

[Sli22] Guus Sliepen. *Tinc Docs*. Nov. 30, 2022. URL: https://www.tinc-vpn.org/docs/ (visited on 11/30/2022) (cit. on p. 12).

[Tai] Tailscale. *Tailscale.* Tailscale. URL: https://tailscale.com/kb/ (visited on 11/30/2022) (cit. on p. 12).