Department of Mathematics and Computer Science
Coding Theory and Cryptology Group

Secure Sessions for Ad Hoc Multiparty Computation in MPyC

Master thesis

Emil Nikolov

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

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

March 21, 2023

Contents

List of Abbreviations

E^3 Extensible Evaluation Environment. 9

ACL Access Control List. 7

CGNAT Carrier-Grade NAT. 5

DERP Designated Encrypted Relay for Packets. 7

IP Internet Protocol. 5, 6

LAN Local Area Network. 5

NAT Network Address Translation. 5

P2P Peer to Peer. 5, 6, 7

STUN Session Traversal Utilities for NAT. 5

TCP Transmission Control Protocol. 5

TLS Transport Layer Security. 5

UDP User Datagram Protocol. 5

VPN Virtual Private Network. 5, 6, 7

List of Figures

Chapter 1

Introduction

Chapter 2

Related work

In this chapter we will provide a high level overview of the solutions that will be analyzed in more detail in the following chapters. There is a large body of existing protocols and applications that are relevant to our problem of connecting the parties of a multiparty computation over the internet. To make reviewing them easier, we will try to approximately map them to the 7 layers of the OSI model, which will not always be entirely accurate because many protocols and applications implement aspects of several layers. The table below shows the layers at which the protocols and network overlays appear to their users, but not necessarily the layer at which their internals are implemented.

OSI Layer	Description	Protocols	Network overlays
7. Application	High level protocols that user-facing services use	HTTP, HTTPS, DNS, FTP, SMTP, UPnP, NAT-PMP, PCP, SSH, STUN, TURN	WebRTC, OpenZiti, Teleport, ngrok, TOR, BitTorrent, IPFS, Ethereum, Freenet
6. Presentation	Translation of data between a networking service and an application, e.g. encoding, compression, encryption	MIME, TLS, Noise	
5. Session	Session setup, management, teardown, authentication, authorization	SOCKS, X.225	
4. Transport	Sending data of variable length over a network while maintaining quality-of-service, e.g. ports, connections, packet splitting	UDP, TCP, NAT port mapping	

OSI Layer	Description	Protocols	Network overlays
3. Network	Sending data packets between two nodes, routed via a path of other nodes, e.g. addressing, routing	IP, ICMP, NAT	TUN driver, IPSec, OpenVPN, Tinc, Wireguard, Tailscale, Nebula, ZeroTier
2. Data link	Sending data frames between two nodes, directly connected via a physical layer, e.g. on a LAN	MAC, L2TP	TAP driver, N2N, OpenVPN, Tinc
1. Physical	Sending raw bits over a physical medium	RS232, Ethernet, WiFi, USB, Bluetooth	

2.1 Internet protocol

IPv4 routing

NAT

NAT Traversal

Universal Plug and Play (UPnP)

PCP, NAT-PMP

STUN and TURN

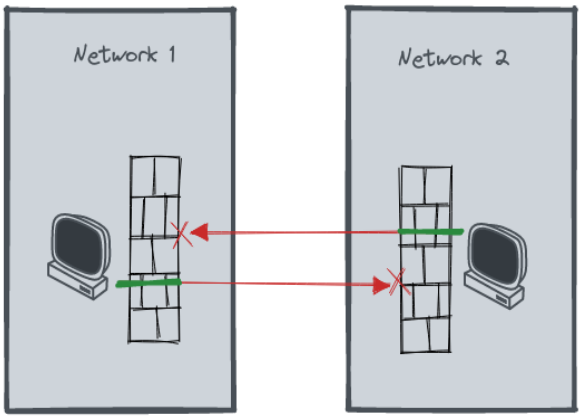

Figure 2.1: "Two parties behind separate NATs"

As we mentioned in the introduction chapter, the devices in a typical home network can only initiate connections to public endpoints (via Network Address Translation (NAT)) but cannot be discovered from outside their Local Area Network (LAN). This poses a challenge when two parties who want to communicate via a direct link are both behind separate NATs 2.1 and neither can be contacted by the other one first. Mesh Virtual Private Networks (VPNs) solve this issue via NAT traversal techniques such as User Datagram Protocol (UDP) hole punching based on concepts from Session Traversal Utilities for NAT (STUN). The machines of each party can contact a public STUN server 2.2, which will note what Internet Protocol (IP) addresses the connections come from and inform the parties. Since the parties initiated the connection to the STUN server, their routers will keep a mapping between their local IP addresses and the port that was allocated for the connection in order to be able to forward the incoming traffic. Those "holes" in the NATs were originally intended for the STUN server, but mesh VPNs use the stateless "fire and forget" UDP protocol for their internal communication, which does not require nor provides a mechanism for the NATs to verify who sent a UDP packet. With most NATs, this is enough to be able to (ab)use the "punched holes" for the purpose of Peer to Peer (P2P) traffic from other parties. Mesh VPNs implement the stateful Transmission Control Protocol (TCP) and Transport Layer Security (TLS) protocols on top of UDP and expose an regular network interface to the other programs, keeping them shielded from the underlying complexities. Other NAT implementations such as Symmetric NAT and Carrier-Grade NATs (CGNATs) can be more difficult to "punch through" due to their more complex port mapping strategies. In those cases, establishing P2P connections might involve guess work or even fail and require falling back to routing the (encrypted) traffic via another party or service.

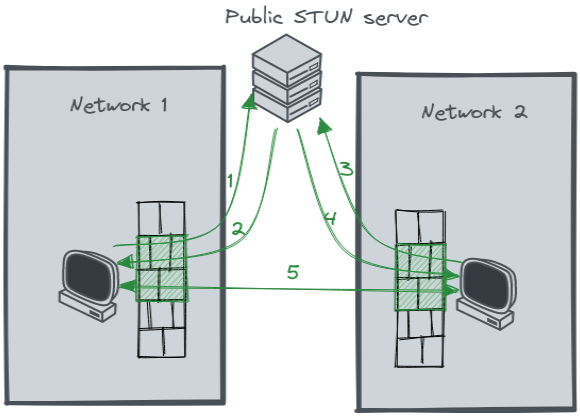

Figure 2.2: "NAT traversal via STUN"

2.2 Network overlays

2.2.1 TUN/TAP driver

2.2.2 Traditional VPNs

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 interface controller via a TUN/TAP driver on the operating system level and allow other

software to transparently use the functionality of the 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 multi-party computations due to their 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/TAP 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 in order 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+Xpg0ZZeOf+Ies/oql0NZxw+SOmkc=
AllowedIPs = 101.0.0.2/32
Endpoint = peer1.example.com:38133
```

```
# peer2.conf
[Interface]
Address = 101.0.0.2/32
ListenPort = 38133
PrivateKey = sN/d6XUPEVPGSziVgCCOnOivDK+qAoYC3nxnssQ5Rls=

[Peer]
PublicKey = e/TxvPmrgcc1G4cSH2bHv5JOPRHXKjYxTFoU8r+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[**noiseProtocol**]. 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.

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 in order 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, with the exception of Tailscale's coordination service which handles the 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 a number of high profile ex-googlers and is the most end-user focused of the three, providing a service that allows people to sign up using a variety of identity providers including 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
- OpenZiti
- 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? - Custom solution that automates the wireguard configuration by visiting a web page

Additionally we will analyse 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 a number of 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 a number of 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 in order to execute a joint multiparty computation. Those steps will be analyzed in terms of:

- Complexity - how much technical expertise is expected from the parties in order 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

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

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

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

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

[Def22] Defined. *Nebula: Open Source Overlay Networking | Nebula Docs*. 2022. URL: https://docs.defined.net/docs/ (visited on 12/01/2022) (cit. on p. 7).

[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. 6).

[Fon22] Juan Font. *Juanfont/Headscale*. Dec. 6, 2022. URL: https://github.com/juanfont/headscale (visited on 12/06/2022) (cit. on p. 7).

[Ope22] OpenVPN. *Community Resources*. OpenVPN. 2022. URL: https://openvpn.net/community-resources/ (visited on 11/30/2022) (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. 7).

[Tai] Tailscale. *Tailscale*. Tailscale. URL: https://tailscale.com/kb/ (visited on 11/30/2022) (cit. on p. 7).