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 15, 2023

Contents

List of Abbreviations

E^3 Extensible Evaluation Environment. 3

List of Figures

Chapter 1

Introduction

Chapter 2

Testing methodology

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. During the thesis assignment we will look at a number of solutions for ad hoc MPC sessions and compare them in terms of performance, security and usability.

2.1 Performance

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

2.2 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

2.3 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 3

Overview

In this chapter we will provide a high level overview of the solutions that will be analyzed in more detail in the following chapters.

3.1 Overlay Networking

The Open Systems Interconnection (OSI) model distinguishes 7 layers in computer networks:

- 7 - Application layer
- 6 - Presentation layer
- 5 - Session layer
- 4 - Transport layer
- 3 - Network layer
- 2 - Data link layer -
- 1 - Physical layer

3.2 Virtual Private Networks (VPN)

Chapter 4

Wireguard

Wireguard[Don17] is a VPN protocol built with the Noise Protocol Framework[**noiseProtocol**] that focuses on configuration simplicity. It considers issues such as peer discovery and key distribution as out of scope and a responsibility of a higher level system that uses Wireguard as a building block. 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 another.

```
# 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/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. The Address field specifies the virtual IP address that the local network interface will use, while the AllowedIPs 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 peer. 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.

4.1 Implementation

Chapter 5

Tailscale

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

5.1 Overview

5.2 Usability

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

5.3 Security

5.3.1 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

5.3.2 Identity

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

- Magic DNS

5.4 Performance

Bibliography

[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. 7).