Chapter 1

Intro

- Emphasize on the difference between the Solutions Test Framework (E3) and the actual Solutions
- The Test Framework is an engineering problem of its own which deals with the scaffolding necessary to demonstrate the candidate solutions in realistic scenarios

Chapter 2

Landscape Overview

- How should the thesis be layed out?
    - A chapter for each solution with subsections for Overview, Security, Usability, Performance analysis
    - A chapter for Overview, Security, Usability, Performance analysis with subsections for each solution

2.1 WebRTC

- Peer to peer communications for browsers
    - can also work without a browser
    - Mainly used for multimedia communications - Peer-to-peer Audio/Video/VoIP
- Spec - https://www.w3.org/TR/webrtc/
- Uses nat STUN/TURN/ICE
- Data is encrypted
- Identity
    - Session Description Protocol (SDP)
    - peer certificates are generated and announced over SDP
    - ICE Candidates are negotiated for STUN/TURN connections
- Not a VPN
    - I think it can't serve as a TCP/IP network overlay that other applications can use
- Does not require additional plugins or native apps
- There seem to be many publicly available services that can be used as ICE servers for WebRTC
    - stun.l.google.com:19302
    - https://gist.github.com/zziuni/3741933
- Examples
    - https://github.com/pion/webrtc
        - Go
    - https://github.com/pojntfx/weron
- Resources
    - https://webrtcforthecurious.com/
    - https://temasys.io/guides/developers/webrtc-ice-sorcery/

  - https://web.dev/webrtc-basics/

2.2 Network Address Translation (NAT) Traversal

https://bford.info/pub/net/p2pnat/

https://www.jordanwhited.com/posts/wireguard-endpoint-discovery-nat-traversal/

2.2.1 test caption

2.2.2 Session Traversal Utilities for NAT (STUN)

- Uses a STUN server for discovery and UDP hole-punching
- Communications are peer-to-peer
- Examples:
  - https://github.com/shawwwn/Gole
  - https://github.com/malcolmseyd/natpunch-go ### Traversal Using Relays around NAT (TURN)
- Peers use a relay server as a mediator to route traffic ### Interactive Connectivity Establishment (ICE)
- Umbrella term covering STUN/TURN and other related techniques

2.2.3 Designated Encrypted Relay for Packets (DERP)

- TURN-like protocol by Tailscale
- Relaying encrypted Wireguard traffic over HTTP
- Routing based on the Peer's public key
- Overview - https://tailscale.com/kb/1232/derp-servers/
- Source code
  - https://github.com/tailscale/tailscale/tree/main/cmd/derper
  - https://github.com/tailscale/tailscale/blob/main/derp/derp.go

2.3 Wireguard

- VPN Protocol
- Used by Tailscale
- Whitepaper - https://www.wireguard.com/papers/wireguard.pdf
- Built with the Noise Protocol Framework
- Typically used as a building block in more complicated systems
- OSI Layer
  - Kernel mode - Network Layer 3 (IP)
  - Userspace mode - Transport Layer 4 (UDP, TCP)
- Simple configuration
  - Each peer has a public/private key pair for authentication and traffic encryption
  - Each peer has a config file:

```
[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 = 142.93.135.154:38133
PersistentKeepalive = 25
```

- Creates a virtual network interface in the operating system that looks like an additional network card and can be used for TCP/IP communications
- Handles the encryption of traffic
  - deals with handshakes and generating symmetric session keys
- Cryptokey routing - associates public/private key pairs with IP addresses
- Out of scope:
  - key distribution - managed manually or via other software that builds on top of wireguard
  - peer discovery - for each pair of peers, one needs to have an endpoint that can be reached by the other peer
- Resources
  - https://www.jordanwhited.com/posts/wireguard-endpoint-discovery-nat-traversal/

2.4 Noise Protocol Framework

- Framework for building protocols
- Spec - http://www.noiseprotocol.org/noise.pdf
- Suite of channel establishment protocols
- Similar to TLS
- Based on Elliptic-curve Diffie–Hellman (ECDH) Handshakes
- Used in WhatsApp, Signal, Wireguard
- Resources:
  - Crypto Layers
    - Low-level primitives - AES, ChaCha20.
    - Usefully combined primitives - AES-OCB, NaCl secretbox...
    - High-level protocols - TLS, Noise.
    - (sometimes) crypto spoken over the encrypted protocols, often for E2E crypto e.g. GPG over SMTPS, or CloudFlare blinded CAPTCHA tokens over HTTPS
  - https://noiseexplorer.com/
  - https://duo.com/labs/tech-notes/noise-protocol-framework-intro
  - https://www.youtube.com/watch?v=ceGTgqypwnQ

2.5 Tailscale

- Mesh VPN
- Built on top of [[0203-wireguard|Wireguard]]
-
- Coordination service
  - Closed source
  - Facilitates STUN/TURN for peer discovery

  - Distributes wireguard public keys
- Client
  - Open source
  - Interacts with the Coordination service
  - Configures Wireguard

2.6 Nebula

- Mesh VPN
- Similar to Tailscale
- Does not use Wireguard
- Built using the [[0204-noise|Noise Protocol Framework]] (used in Wireguard)
- Uses a Certificate Authority that needs to sign each Peer's certificate
  - Certificates contain
    - Peer's Virtual IP address
    - Peer's public key

2.7 OpenZiti

- Network Overlay
- No STUN
- Works by relaying traffic through intermediaries
- Focused on Services
- Allows embedding into apps via an SDK

2.8 DIDComm v2

- Protocol for communicating that uses DID for identity management

- Spec: https://identity.foundation/didcomm-messaging/spec/

- No sessions

  - messages are always encrypted with the public keys of the peers

- Routing happens via mediators

sources:
- https://newsletter.impervious.ai/decentralized-identifiers-implications-for-your-data-payments-and-commu

-

2.9 ngrok

- Paid service for creating public URLs for local services
-

2.10 BitTorrent

- No concept of identity based communication or peer discovery

2.11 Ethereum's P2P Protocol

2.12 IPFS

- Content Addressable Storage
- Network is for discovering data, not for executing computations
-

2.13 TOR

2.14 Friend2Friend Networks

- Peer to Peer Dark Web network
- Network that directly connects the machines of people who know each other
- Services and files accessible only by other people on the network
-

2.15 Teleport

Chapter 3

Solution notes

3.1 Headscale

- Docker seems to have a 50-100% performance penalty (possibly due to docker's internal NAT) which makes the performance results of the headscale setup worse than they should be
- Modify the deployment setup to
  - not depend on all machines being on the same tailscale network
  - use the `*.demo.mpyc.tech` hostnames
  - the nodes should switch between the tailscale and headscale network depending on the demo script
  -

Chapter 4

Brainstorm for custom solutions

- Initial state
  - Here's my identity, here are the identities of the other parties
- Desired result
  - Executed MPC

With tailscale we'd need to

- Each party:
  - registers a Tailscale account
  - Downloads and installs tailscale on the machine they want to run the MPC on
  - Runs tailscale on their machine and logs into their account in order to link it to their own Tailnet
  - Shares their Tailscale machine with the Tailnets of each of the other parties
  - Downloads 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

- we don't need to use the same route for the communication Party A → Party B and Party B → Party A
- we can have something like an asynchronous STUN
- Party A sends a QR code/public URL/json object/DID document containing
  - Party A's public key
  - Party A's Mediator URL
- The mediator is a STUN/TURN/DERP server
  - other parties can either use it as a STUN server to find out how to access the hole in party A's NAT punched by the mediator
  - or use it as a relayer so that Party B can send encrypted packets to party A via the mediator
- Instead of a QR code, the information could be stored on a public ledger and could be resolved via DIDs

---

- There is a generic MPC wrapper program that deals with supporting tasks like generating identities and pulling MPC demos
- One party creates a “lobby” for an MPC session in their program and get a session id/public URL/QR code that can be shared with the other parties
-

- Public Website is visited by everyone
- They prove their identities using a SSI wallet or OIDC
- They get a QR code that serves as an invitation
    - contains their STUN based endpoint and identity
- Somehow everyone needs to scan each other’s qr codes

- One party creates a “lobby” for an MPC session by visiting a public website
    - They provide their identity via OIDC/SSI wallet
- They get a public link/QR code that can be shared with the other parties
- The parties visit the URL and also provide their identities
- The parties obtain the MPC program they want to run
    - MPC program distribution could be done separately via cloning the github repo?
    - They could choose a DEMO and download it from the website?
    - There could also be a program running on the host machines that deals with the source code distribution. Similar to downloading custom maps for warcraft 3 or dota 2?
    - They could specify the source code when creating the MPC session in the website?
    - If the demo is not symmetrical where different parties have different roles and need to execute different programs, the roles could be assigned by the host or the people could choose their preferred role themselves?
- The parties download a configuration file that contains information on how to connect to the other parties
- They run the demos with the downloaded config file
- A temporary Wireguard mesh VPN is created between all parties

Chapter 5

Writing

- Ask questions and answer them
- Don't try to get the writing perfect on the first pass
- Write anything first and then iterate on it later

Chapter 6

Preparation phase report outline

6.1 Introduction

-
  - What is the purpose of this report?
  - How does it relate to the thesis assignment?
- Background information
  - What do I need to know to in order to be able to understand the problem that this thesis is trying to solve and its importance?
    - What is MPC?
    - What does MPC achieve?
    - When is MPC useful?
    - How does MPC work?
    - What is MPyC?
- Problem description
  - What is the issue with MPyC that this assignment is trying to solve?
- Research questions
- Scope

6.2 Technical survey

- Deployment
- Connectivity
- Summary

6.3 Implementation details

- Reproducible development
- NixOS image
- Provisioning

- Deployment
- Runtime execution

6.4 Conclusions

Chapter 7

Thesis outline

7.1 Introduction

- Background information
- Problem description
- Research questions
- Scope

7.2 Related work / Literature review

- Are there any existing solutions that can be applied to the problem in some capacity?

7.3 Methodology

- How are the solutions going to be evaluated?
- Performance
  - Which performance characteristics are relevant to multiparty computations?
  - How are the chosen solutions going to be implemented and deployed?
  - How are the performance metrics going to be measured?
- Security
  - What security aspects are relevant to multiparty computations?
  - How are they going to be analyzed?
- Usability
  - How is the usability of a solution going to be evaluated? ## Solutions
- For each solution
  - What is the solution?
  - How was it implemented?
  - What are the security implications?
  - How can it be used by the parties?
  - What is the performance of the solution?

7.4 Conclusions

- How do the proposed solutions compare to each other?
- Which solution is preferred under which circumstances?

Chapter 8

Latex

- use \input instead of \include to avoid issues with missing directories and .aux files