Universidade de Aveiro
Departamento de Electrónica, Telecomunicações e Informática,
2023

Vasco Regal Sousa

# Multiple Client Wireguard Based Private and Secure Overlay Network

# DOCUMENTO PROVISÓRIO

"An idiot admires complexity,
a genius admires simplicity."

— Terry A. Davis

**o júri / the jury**

presidente / president

**ABC**
Professor Catedrático da Universidade de Aveiro (por delegação da Reitora da Universidade de Aveiro)

vogais / examiners committee

**DEF**
Professor Catedrático da Universidade de Aveiro (orientador)

**GHI**
Professor associado da Universidade J (co-orientador)

**KLM**
Professor Catedrático da Universidade N

**agradecimentos / acknowledgements**

Ágradecimento especial aos meus gatos...

Desejo também pedir desculpa a todos que tiveram de suportar o meu desinteresse pelas tarefas mundanas do dia-a-dia, ...

**Abstract**

An overlay network is a group of computational nodes that communicate with each other through a virtual or logic channel, built on top of another network. Although there are already numerous services and protocols implementing this mechanic, scalibility and administration agility are among the most desired characteristics of such a network topology. Hence, this document presents a centralized solution for the creation and control of secure overlay networks for multiple nodes - from client management to operation auditing. In the University of Aveiro, namely the autonomous robot ecosystem residing in the IRIS lab, supporting such a networking architecture would prove to be particular interesting, both for development and project organization. Therefore, this context is used as a validation environment. ...

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Motivation

Network security has become a topic of evergrowing interest among any information system. Companies strive to ensure their communications follow principles of integrity and confidentially while minimizing attack vectors that could compromise services and data. With such goals in mind, network topologies are subjected to policies which apply rules and conditions to inbound and outbound traffic. One such mechanism is the use of Virtual Private Network (VPN) .

Traditional VPN services consist in the establishment of a secure, encrypted channel between a client and a network, through an insecure communication medium.

The University of Aveiro (UA)'s Intelligent Robotics and Systems Laboratory (IRIS-Lab) conducts research projects using autonomous mobile robots, which communicate through a Wi-Fi network. Currently, this network is confined to the premises of the IRIS-Lab, preventing the robots from operating in the remaining UA's buildings. Although the ua's Wi-Fi infrastructure covers most of its edifices, which can be used by the robots, due to security mechanisms, this network proves to be highly restraining, not allowing Peer to Peer (P2P) communications through the Robot Operating System (ROS) [13] - the operating system the robots run on - middleware without additional network equipments. Moreover, these constraints keep developers from being able to interact with the robots through their personal machines, which, if otherwise possible, would be of great interest.

## 1.2 Objectives

The main goal of this dissertation is to implement a private overlay network manager to be used exclusively by UA's clients. The concept of a manager entails both the definition of a network's client universe (which nodes should be allowed to connect to a certain network) and its respective identification and authentication mechanisms.

In the IRIS-Lab scenario, the management platform should provide operations to achieve communication between a team of robots, regardless of their physical location within the campus. Moreover, the authentication and connection to a desired overlay network by the robots must be a seemingless operation, requiring little to no manual configuration.

Finally, all traffic must be encrypted and properly authenticated, to ensure the privacy of the communication.

## 1.3 Document Structure

This document presents an implementation proposal of such an overlay network manager. With this goal in mind, it is structured in two main chapters - State of The Art and Methodology. The former consists in an exploration of the background and current state of the art, providing an analysis not only of potential tools, protocols and frameworks suitable for the scope of the dissertation but also of published research conducted covering similar topics and scenarios while the latter establishes the work methodology to be taken for the development and results gathering process.

# Chapter 2

# State of the Art

## 2.1 Encrypted Peer to Peer Communications / VPNs

VPNs have become a mature technology, with widespread usage among the Internet. With such a wide range of products offering VPN capabilities, this section aims to analyze some of its most notable providers, focusing on the processes involved on their respective data planes - which refers to the subsection of a network communication responsible for carrying data between devices. This implies not only the quality of its device identification, encryption suite and protocol security but also of its features regarding concepts such as mobility - how the service behaves when clients change their physical locations and IP adresses - and overcoming constraints associated with networks using NAT mechanisms and secured with firewall rules. Finally, since operations taken in the data plane are necessarily associated with a computation overhead, namely authentication, traffic encryption and session management, protocol performance is also perceived as a valued dimension.

#### NAT Traversal

It is very likely that devices on the internet reside in a network behind both NAT mechanisms and Firewall rules, with no open ports. Also, believing nodes will have a consistent static IP is a very naive assumption, specially when considering mobile devices. NAT Traversal is a networking technique that enables the establishing and maintaining of P2P connections between two peers, no matter what's standing between them, making communication possible without the need for firewall configurations or public-facing open ports. There's no one solution to achieve this functionality. In fact, there are various developments effectively implementing a NAT Traversal solution, such as ICE [5] and STUN [11]. Hence, each VPN service can have its own way of supporting NAT Traversal. Each case is explored seperatly in its own subsection.

### 2.1.1 IPSec

### 2.1.2 OpenVPN

### 2.1.3 Wireguard

Wireguard [3] is an open-source layer 3 network tunnel implemented as a kernel virtual network interface. Wireguard offers both a robust cryptographic suite and transparent

session management, based on the fundamental principle of secure tunnels: peers in a Wireguard communication are registred as an association between a public key - analogous to the OpenSSH keys mechanism - and a tunnel source IP address.

One of Wireguard's selling points is its simplicity. In fact, compared to similar protocols, which generally support a wide range of cryptographic suites, Wireguard settles for a singular one. Although one may consider the lack of cipher agility as a disadvantage, this approach minimizes protocol complexity, increasing security robustness by avoiding SSL/TLS vulnerabilities commonly originated from such protocol negotiation.

### Routing

Peers in a Wireguard communication maintain a data structure containing its own identification - both the public and private keys - and interface listening port. Then, for each known peer, an entry is present containing an association between a public key and a set of allowed source ips.

This structure is queried both for outgoing and incoming packets. To encrypt packets to be sent, the structure is consulted and, based on the destination address, the desired peer's public key is retrieved. As for receiving data, after decryption (with the peer's own keys), the structure is used to verify the validity of the packet's source address, which, in other words, means checking if there's a match between the source address and the allowed addresses present on the routing structure.

Optionally, Wireguard peers can configure one aditional field, an internet endpoint, defining the listening address where packets should be sent. If not defined, the incoming packets' source address is used instead.

### Cipher Suite

As aforementioned, Wireguard offers a single cipher suite for encryption and authentication mechanisms in its ecosystem. The peers' pre-shared keys consist in Curve25519 points [1], an implementation of an eliptic-curve-Diffie-Hellman function, characterized by its strong conjectured security level - presenting the same security standards as other algorithms in public key cryptography - while achieving record computational speeds.

Regarding payload data cryptography, a Wireguard message's plain text is encrypted with the sender's public key and a nounce counter, using ChaCha20Poly1305, a Salsa20 variation [2]. The ChaCha cryptographic family offers robust resistance to cryptoanalytic methods [12], without sacrificing its state-of-the-art performance.

Finally, before any encrypted message exchange actually happens, Wireguard enforces a 1-RTT handshake for symmetric key exchange (one for sending, and one for receiving). The messages involved in this handshake process follow a variation of the Noise [10] protocol - essentially a state machine controlled by a set of variables maintained by each party in the process.

### Security

On top of its robust cryptographic specification, Wireguard includes in its design a set of mechanisms to further enhance protocol security and integrity.

With such a scope in mind, Wireguard presents itself as a silent protocol. In other words, a Wireguard peer is essentially invisible when communication is attempted by an illegitimate

party. Packets coming from an unknown source are just dropped, with no leaks of information to the sender.

Additionally, a cookie system is implemented in an attempt to mitigate DDOS attacks. Since, to determine the authenticity of an handshake message, a Curve25519 multiplication must be computed, an operation requiring considerable CPU usage, a CPU-exhaustion attack vector could be exploited. Cookies are introduced as a response message to handshake initiation. These cookie messages are used as a peer response when under high CPU load, which is then in turn attached to the sender's message, allowing the requested handshake to proceed later.

#### Sessions and Key Rotation

### 2.1.4 Performance Comparison

The concept of performance in VPN applications entails both protocol overhead on communication throughput and bandwidth usage minimization. These dimensions can be empirically measured, by calculating communication latency / ping time and throughput. The performance claims on [3], where, when compairing Wireguard to its alternatives like OpenVPN and IPsec, presents results in favor of Wireguard in both metrics. This conclusion is backed by more extensive research [6], [7], where communication is tested in a wide range of different environments and CPU architectures.

Wireguard, due to its kernel implementation (compared to, for example, OpenVPN's user space implementation) and efficient multi-threading usage contribute greatly to such performance benchmarks. Moreover, its relatively small codebase (around 4000 lines) creates a very auditable, maintainable VPN protocol.

## 2.2 Control Platforms

Although Wireguard proves itself as a robust, performant and maintainable protocol for encrypted communication, it still presents some complexity regarding administration agility and scalibility. New clients added to a standalone Wireguard network imply the manual reconfiguration of every other peer already present, a process with added complexity and prone to errors, as more nodes join the system. With this in mind, this section explores applications and implementations of control platforms built, or with the pontential to be built, on top of Wireguard, aiming to create a seamless peer orchestration and configuration process, minimizing human intervention.

First, it is mandatory to define what a control platform is. The main goal should be to overcome the limitations previously mentioned, by supporting:

- A centralized server storing peers' identification (public key and tunnel IP address).
- Establishment of secure channels between peers and such a centralized server.
- On-demand retrieving of information regarding any peer in network.

### 2.2.1 OOR Map Server Implementation

An implementation with said requirements is proposed in [8]. The core architecture of this solution consists in a centralized Open Overlay Router Map Server, containing peer

identification data, which provides devices with on-demand information regarding any other peer in the network to setup a direct connection. From a client prespective, a peer wanting to communicate with another should first establish a secure Wireguard connection to this server and request a connection with a destination node. The server, with the source IP and public key of the requesting client, redirects this data to the destination node, reaching a state where both peers contain all necessary information to begin the Wireguard tunnel.

This prototype successfully tackles one of the main limitations of Wireguard, offering a mechanism capable of dynamically configuring peers, without the need to reconfgiure every device everytime a new client joins the network. Also, it reduces routing table complexity, as peers are not required to keep all other peers' information locally. However, the addition of such a centralized entity also introduces a new attack vector. Efectively, if the private key of the central server, crucial in creating the first secure channel between a peer and the server, is compromised, a man-in-the-middle attack could be mounted, since an attacker could impersonate the centralized server.

Regarding performance, there is, as expected, an overhead compared to native OOR benchmarks, as requests to OOR Map Server are themselves conducted through a Wireguard channel.

### 2.2.2 TailScale

TailScale is a VPN service operating with a golang user-space Wireguard variant as its data plane [9]. Traditional VPN services operate under a hub-and-spoke architecture, a model composed by one or more VPN Gateways - devices accepting incoming connections from client nodes and forwarding the traffic to their final destination. Hub-and-spoke architectures carry some limitations. First, it implies increased latency associated with geographical distance between a client to the nearest hub. Also, regarding scalibility and dynamic configuration, adding new clients to the network requires the distribution of its keys to all hubs. With these constraints in mind, TailScale offers an hybrid model. TailScale's central entity, refered to as a coordination server, functions as a shared repository of peer information, used by clients to retrieve information regarding other nodes and establishing on-demand P2P connections among each other.

This control plane approach differs from traditional hub-and-spoke since the coordination server carries nearly no traffic - it only serves encryption keys and peer information. TailScale's architecture can be perceived as an hybrid model, benefitting from the advantages of control plane centralization without bottlenecking its data plane performance.

In pratical terms, a TailScale client will store, in the coordination server, its own public key and where it can currently be found. Then, it downloads a list of public keys and addresses that have been stored in the server previously by other clients. With this information, the client node is able to configure its Wireguard interface and start communicating with any other node in its domain.

#### HeadScale

While TailScale's client is open source, its control server isn't. There is, however, an open-source, self-hosted alternative to TailScale's control server, HeadScale. HeadScale [4] provides a narrow scope implementation (with a single TailScale private network) of the aforementioned control server which, in the authors' words, mostly suitable for personal user

and small organizations.

## 2.3 Automation and Configuration

# Chapter 3

# Methodology

# Bibliography


[1] Daniel J Bernstein. Curve25519: new diffie-hellman speed records. In *Public Key Cryptography-PKC 2006: 9th International Conference on Theory and Practice in Public-Key Cryptography, New York, NY, USA, April 24-26, 2006. Proceedings 9*, 2006.

[2] Daniel J Bernstein et al. Chacha, a variant of salsa20. In *Workshop record of SASC*, 2008.

[3] Jason A Donenfeld. Wireguard: next generation kernel network tunnel. In *NDSS*, 2017.

[4] Juan Font and Kritoffer Dalby. Headscale. `https://headscale.net/`, 2023.

[5] Ari Keränen, Christer Holmberg, and Jonathan Rosenberg. Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal. RFC 8445, 2018.

[6] Steven Mackey, Ivan Mihov, Alex Nosenko, Francisco Vega, and Yuan Cheng. A performance comparison of wireguard and openvpn. In *Proceedings of the Tenth ACM Conference on data and application security and privacy*, 2020.

[7] Lukas Osswald, Marco Haeberle, and Michael Menth. Performance comparison of vpn solutions. 2020.

[8] Jordi Paillisse, Alejandro Barcia, Albert Lopez, Alberto Rodriguez-Natal, Fabio Maino, and Albert Cabellos. A control plane for wireguard. In *2021 International Conference on Computer Communications and Networks (ICCCN)*, 2021.

[9] Avery Pennarun. How tailscale works. `https://tailscale.com/blog/how-tailscale-works/`, 2020.

[10] Trevor Perrin. The noise protocol framework. 2018.

[11] Marc Petit-Huguenin, Gonzalo Salgueiro, Jonathan Rosenberg, Dan Wing, Rohan Mahy, and Philip Matthews. Session Traversal Utilities for NAT (STUN). RFC 8489, 2020.

[12] Gordon Procter. A security analysis of the composition of chacha20 and poly1305. 2014.

[13] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, Andrew Y Ng, et al. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, 2009.