VPN Simulation and Performance Analysis using OpenVPN

Design Document by Group 16
Version 1.0
October 15, 2024

Anvit Gupta
22114009
anvit_g@cs.iitr.ac.in

Ayush Ranjan
22114018
ayush_r@cs.iitr.ac.in

Indranil Das
22114037
indranil_d@cs.iitr.ac.in

Sarvasva Gupta
22114086
sarvasva_g@cs.iitr.ac.in

Souvik Karmakar
22114096
souvik_k@cs.iitr.ac.in

Vineet Kumar
22114107
vineet_k@cs.iitr.ac.in

1. Abstract

This project simulates a Virtual Private Network (VPN) using OpenVPN software and examines its impact on network performance and security. The VPN setup involves encrypting communications between clients and the server, ensuring secure data transmission. A detailed performance analysis is conducted by measuring latency and throughput to understand the overhead introduced by the VPN. Security features such as encryption and client authentication are evaluated, ensuring confidentiality and integrity of data.

2. Introduction

Background: Virtual Private Networks (VPNs) allow secure communication over a public or untrusted network by encrypting traffic between a client and a server. VPNs are widely used in both corporate and personal environments to ensure privacy and data security. They provide confidentiality by encrypting all data, making it unreadable to unauthorized parties, and can also offer authentication to ensure only trusted users can access network resources.

Purpose: The purpose of this project is to simulate a VPN using OpenVPN, a popular open-source VPN solution. The project focuses on encrypting communication between clients and the server while evaluating the impact of using a VPN on network performance, specifically in terms of latency and throughput. Additionally, the project ensures secure client-server communication using certificates and ta.key for enhanced security.

Scope: This project includes the setup and configuration of a VPN server and clients using OpenVPN, the implementation of encryption to secure data transmission, and performance analysis in terms of network latency and throughput. The security of the VPN is analyzed through the use of certificates, keys, and additional security features.

Network Setup: The network consists of a VPN server hosted on a Windows machine and a client connecting remotely using OpenVPN. Port forwarding is configured on the router to allow external access to the VPN server. The server and client exchange data over an encrypted channel using OpenVPN's tunneling protocol.

3. Tools and Technologies

OpenVPN: OpenVPN is an open-source software that provides a robust and flexible way to establish secure point-to-point or site-to-site connections. It uses the OpenSSL library to provide encryption, authentication, and certificate management, allowing secure communication between remote clients and servers.

Client-Server Authentication: OpenVPN uses X.509 certificates to authenticate both the server and the clients. These certificates, along with private keys, establish mutual trust between the VPN server and clients. Additionally, the use of ta.key (TLS-auth) adds an

extra layer of security, protecting against certain types of attacks, such as DDoS.

Performance Measurement Tools:
ping: Used to measure network latency (the time it takes for data to travel from the client to the server and back).
iperf: A network tool used to measure bandwidth (throughput) between the client and the server, helping analyze the impact of the VPN on performance.

4. System Design and Architecture

VPN Topology: The VPN architecture follows a client-server model where multiple clients connect to a single OpenVPN server. The server authenticates each client using certificates, and once authenticated, the clients and server establish a secure, encrypted communication channel.

Port Forwarding: The router is configured to forward UDP port 1194 (default OpenVPN port) to the internal IP address of the VPN server. This ensures that all incoming VPN traffic reaches the server and is handled appropriately.
1. The internal IP address of the VPN server is identified (e.g., 192.168.1.10).
2. In the router's port forwarding settings, port 1194 (UDP) is forwarded to this IP.
3. The VPN server is configured to listen on port 1194 for incoming connections.

Client Authorization: Authorization is handled using client certificates. Each client is issued a unique certificate signed by the VPN server's certificate authority (CA). During the connection process, the server verifies the client certificate before allowing access to the VPN. Additionally, ta.key is used for TLS authentication, providing an extra security measure. This prevents unauthorized clients from initiating any connection unless they possess the correct ta.key file.

Data Flow:
1. Client Initialization: The client initiates a connection to the VPN server using OpenVPN Connect, sending its certificate for authentication.
2. Server Authentication: The server validates the client's certificate against its own CA. If the certificate is valid, the server accepts the connection.
3. Key Exchange: The client and server negotiate encryption keys using TLS, and ta.key is verified.

5. Installing OpenVPN and Configuring Certificates

1. Installing OpenVPN on the VPN Server (Windows)

- On the VPN server and client machines install OpenVPN from the official community downloads.

2. Generating Server Certificates

- On the server, generate a Certificate Authority (CA) and server certificates to authenticate itself to clients.
- Run the following commands to generate the certificates:

```
./easyrsa init-pki
./easyrsa build-ca
./easyrsa gen-req server nopass
./easyrsa sign-req server server
```

- This process generates the server's certificate and key.

3. Generating Client Certificates

- Similarly, generate the client certificates by running the following commands:

```
./easyrsa gen-req client1 nopass
./easyrsa sign-req client client1
```

- The `client1` certificate is then transferred to the client device for authentication.

4. Configuring ta.key for TLS Authentication

- Generate the `ta.key` file for TLS authentication:

```
openvpn --genkey --secret ta.key
```

- This key is used for securing the control channel.

5. OpenVPN Server Configuration

- Create the server configuration file `server.conf`, which includes the generated certificates, `ta.key`.
- Place the files, server.ovpn, server.key, server.crt, ca.crt in the directory

```
C:\Program Files\OpenVPN\config
```

- Example configuration:

```
port 1194
proto udp
dev tun
ca ca.crt
cert server.crt
key server.key
dh dh.pem
tls-auth ta.key 0
cipher AES-256-CBC
```

- Save the configuration file in the OpenVPN configuration directory on the server.
- On the client side (Windows), create the `client.ovpn` configuration file.
- The client configuration includes the server's public IP, port, client certificates, and `ta.key`.
- Place the files, client.ovpn, client.key, client.crt, ca.crt in the directory

  ```
  C:\Program Files\OpenVPN\config
  ```

- Example configuration:

```
client
dev tun
proto udp
remote SERVER_PUBLIC_IP 1194
ca ca.crt
cert client1.crt
key client1.key
tls-auth ta.key 1
cipher AES-256-CBC
```

7. Windows Commands for Client-Side VPN Connection

- Open the Command Prompt on the client machine.
- Run the following command to start the OpenVPN client:

  ```
  openvpn --config "C:\Program Files\OpenVPN\config\client.ovpn"
  ```

- This command initiates the VPN connection using the configuration specified in the `client.ovpn` file.

8. Verifying VPN Connection

- After starting the OpenVPN client, verify the VPN connection by checking the client logs or by running a `ping` command to the VPN server:

```
ping SERVER_IP
```

- This verifies the connection and ensures that the VPN is functioning properly.

6. Some Screenshots of the Setup

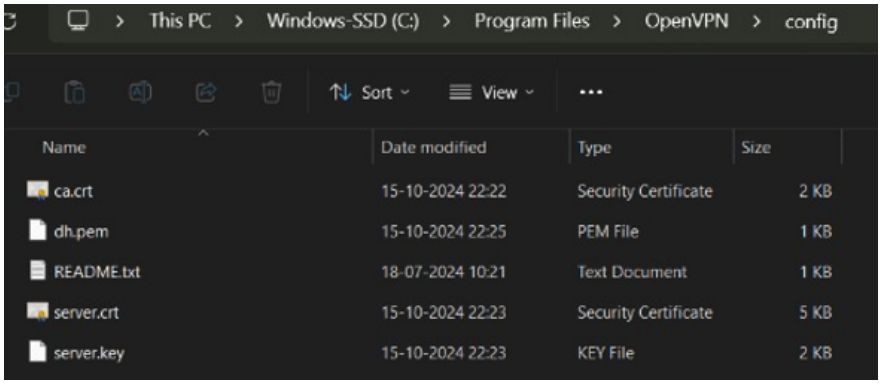

Figure 1: Server

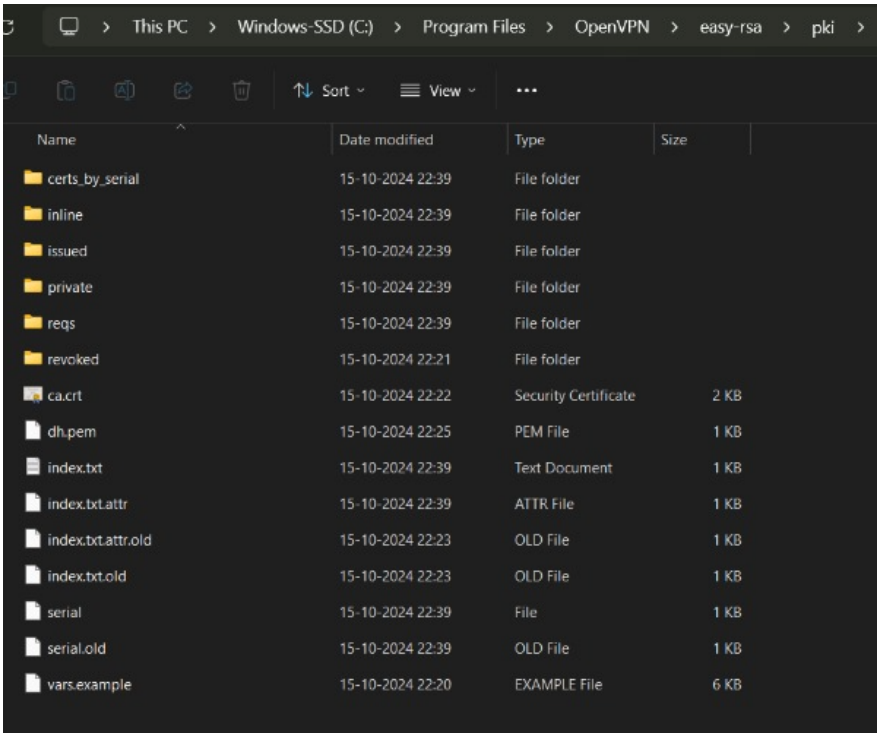

Figure 2: Server

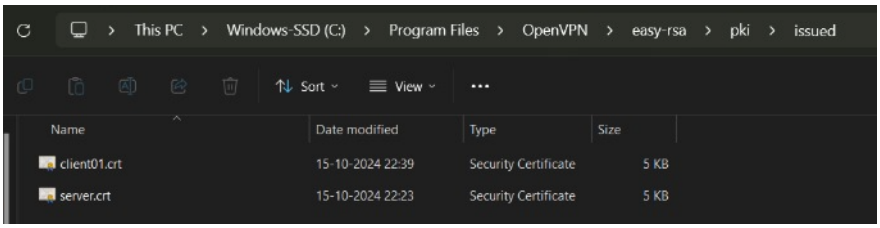

Figure 3: Server

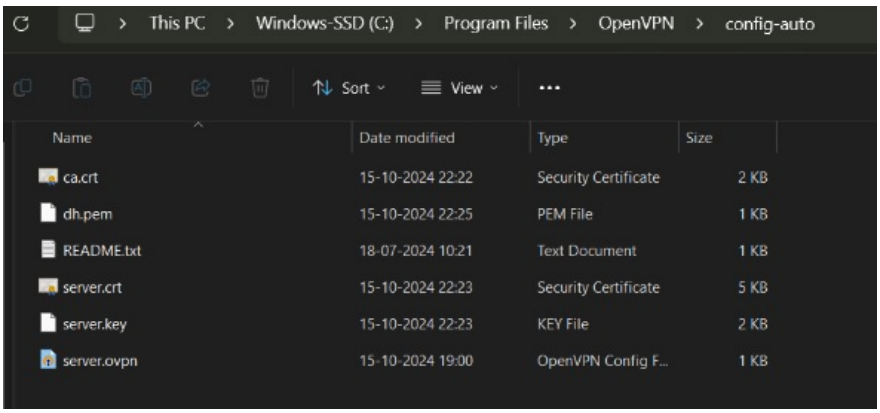

Figure 4: Server

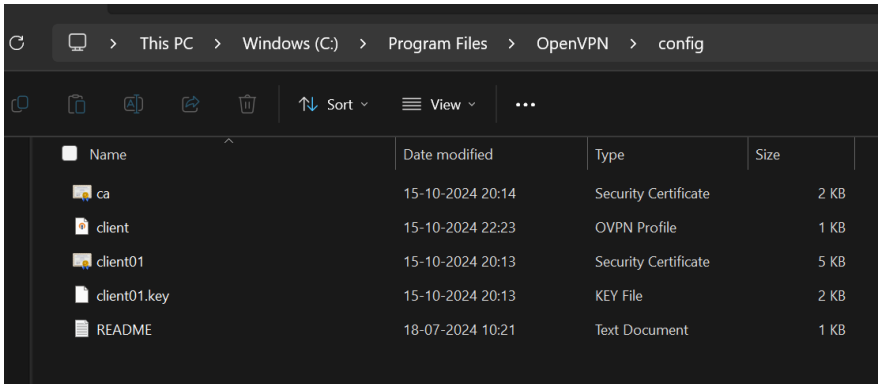

Figure 5: Client

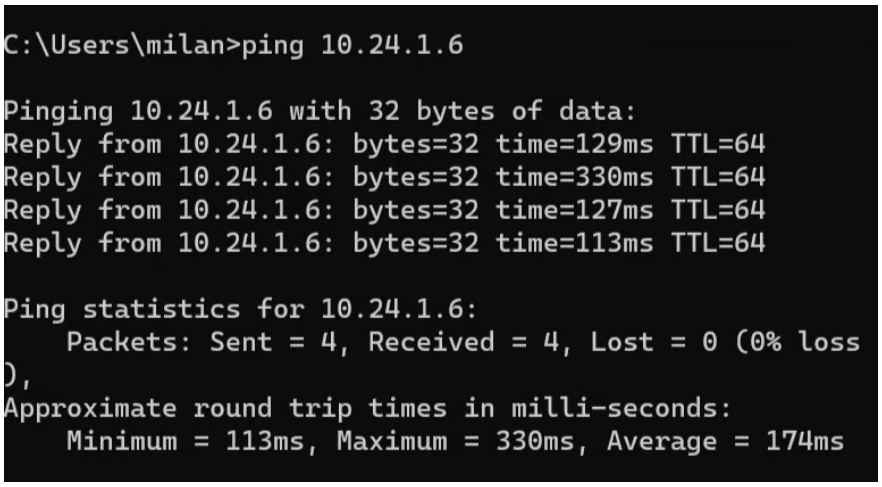

Figure 6: Performance

7. Conclusion

In this project, we successfully set up a Virtual Private Network (VPN) using OpenVPN, allowing secure communication between a client and a server. By creating and using certificates for authentication, we ensured a secure and private connection.

This implementation provided valuable hands-on experience with VPN setup, certificate management, and secure network communication. The project demonstrates how a VPN can be used to protect data during transmission and can be expanded for larger or more complex environments.