Computer Networks — 2023/24, Q2		**Assignment:**	Lab 8
Virtual Private Networks		**Version:**	1.0
Issued:	2023-12-04	**Submission Due:**	2023-12-10

Submission note. You will need to submit all the answers to this lab's questions in Moodle, under "Lab 8: VPN".

1 Goals

- Learn the basics of Virtual Private Networks (VPNs).
- Use unencrypted tunnels to implement a VPN (GRE).
- Use encrypted tunnels to implement a VPN (OpenVPN).

2 Exercises

Virtual Private Networks, or VPNs, are a means of extending a private network (say your home network) using a public network infrastructure (such as the Internet). Moreover, VPNs enable sending and receiving data across networks as if hosts were directly connected to the private network, providing access to resources that, otherwise, would remain inaccessible from the public network. Figure 1 shows the operating principle of common VPNs.

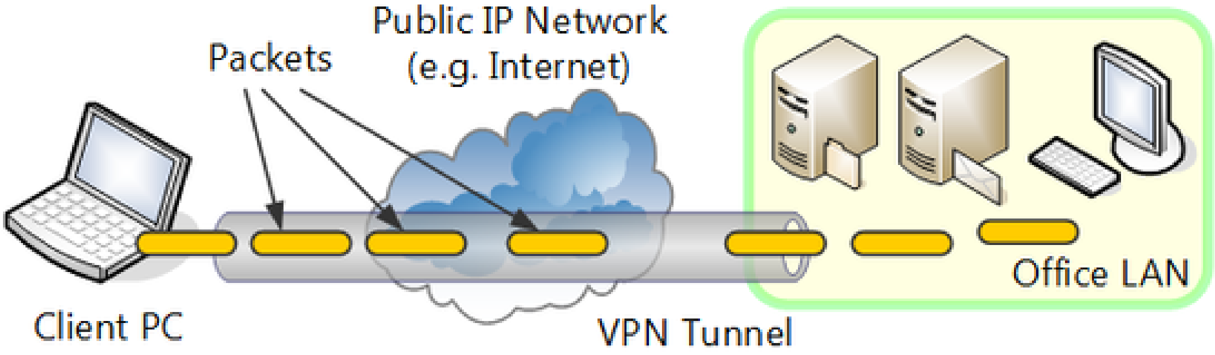

Figure 1: Operating principle of common VPNs

To create a VPN, virtual point-to-point connections are established with tunneling protocols over existing networks, and different security mechanisms can be used to enforce different properties, depending on the security model:

- Encryption (for data confidentiality, in such a way that packet sniffers will not have access to the plaintext payload);
- Message integrity (used to detect packets that have been tampered with);
- Sender authentication (to prevent unauthorized parties from accessing the VPN).

2.1 Unencrypted tunnels

In some scenarios, where data confidentiality is not necessary, VPNs may use tunneling protocols without encryption. For instance, Generic Routing Encapsulation (GRE) is a protocol that encapsulates packets (such as IPv4/IPv6, and unicast/multicast) in order to route other protocols over IP networks, creating a private point-to-point connection. As shown in Figure 2, tunnel endpoints are assigned specific addresses (`10.10.10.1` and `10.10.10.2` in the example).

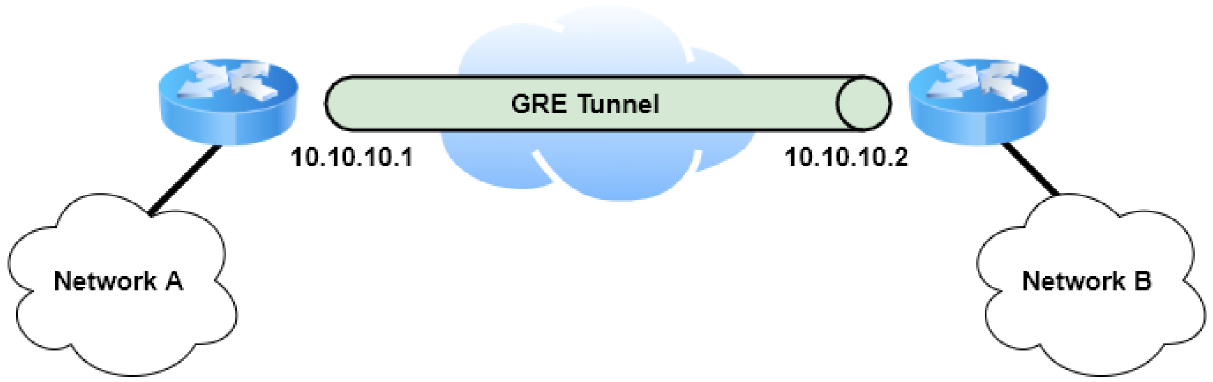

Figure 2: GRE tunnel

GRE works by encapsulating a payload – an inner packet that needs to be delivered to a destination network – inside an outer IP packet. GRE tunnel endpoints send payloads through GRE tunnels by routing encapsulated packets through intervening IP networks. Other IP routers along the way do not parse the payload (the inner packet); they only parse the outer IP packet as they forward it towards the GRE tunnel endpoint. Upon reaching the tunnel endpoint, GRE encapsulation is removed and the payload is forwarded along to its ultimate destination. Figure 3 explains how packets are encapsulated in GRE tunnels.

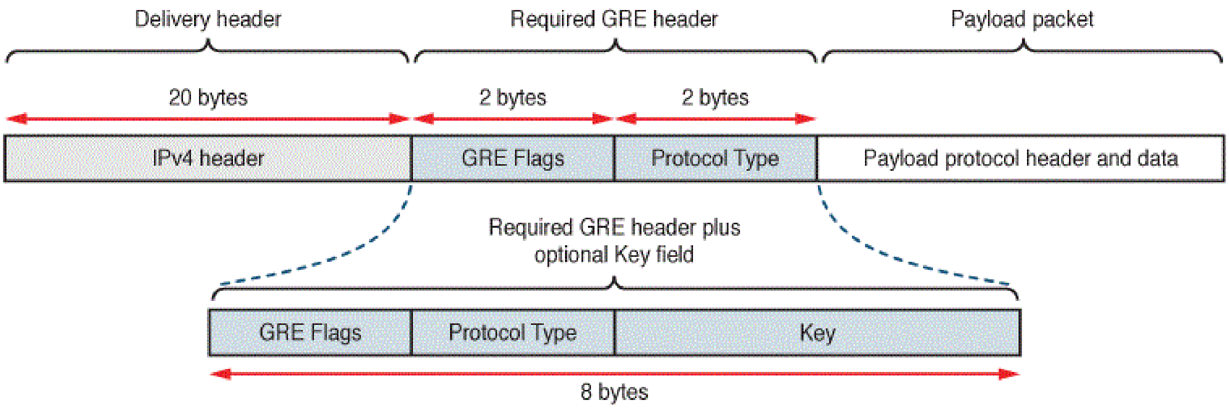

Figure 3: Packet encapsulation in GRE

GRE provides a stateless, private connection, but is not a secure protocol because it does not use encryption, as secure VPN technologies – like IPsec or OpenVPN – do.

Suppose that you are on a laptop connected to your office network, but need to access a custom server running on your desktop, which is connected to your home network. To that end, you will configure a GRE tunnel in CORE to implement a simple VPN that emulates the connection between your home network and your office network, granting you remote access from your office.

Start by opening CORE and loading the network topology from file `vpn-1.imn`. Before starting the session, take your time to analyze the network topology, and mind the two distinct private networks, in purple and green. In particular, you should:

- Check which services are running on routers `r1` and `r4`;
- Confirm that `desktop` is running a simple HTTP server in Python on port `8080` (you can, for instance, use `nmap` from router `r1` to check that port `8080` is open).

  Hint. To check which services are running on a given node on CORE, stop the session, right-click the node, select `Services...` at the bottom of the list, and look for active services, in white.

Q1. Try to ping the `desktop` at home (`10.0.3.10`) from the `laptop` at the office (`10.0.4.20`). Why does ping fail?

To establish the GRE tunnel, the two endpoints, one in each network, need to be configured separately. Starting with the endpoint in the home network, open a terminal on router `r1`, and run the following commands:

```
:~$ ip tunnel add gre1 mode gre local 10.0.0.1 remote 10.0.2.2
:~$ ip link set gre1 up
:~$ ip addr add 10.10.10.1/24 dev gre1
```

This will create a GRE tunnel endpoint called `gre1` on router `r1`, and set its remote address to `10.0.2.2` (the IPv4 address of interface `eth0` of router `r4`, i.e. the remote endpoint). Tunneling packets will be originating from `10.0.0.1` (i.e. the local endpoint), and the tunnel device will be assigned the IPv4 address `10.10.10.1/24`.

Q2. Configure the other endpoint of the tunnel on router `r4` (the endpoint should be assigned the IPv4 address `10.10.10.2`). Include all relevant commands.

The GRE tunnel should now be configured, and endpoints should be able to ping one another (try to ping `10.10.10.2` from router `r1`, and then try to ping `10.10.10.1` from router `r4`). If that does not work, the tunnel may be misconfigured.

> **Hint.** Tunnel configuration is not persistent: every time the CORE session restarts, tunnel configuration is lost, and you will have to reconfigure the endpoints. It is possible to persist the configuration, but that is out of the scope of this lab.

Q3. Start Wireshark on interface `eth0` of router `r3`. Open a terminal on router `r1` and ping the GRE tunnel endpoint on the other side. Do you notice anything different about these ICMP packets?

Even though the tunnel is configured, pinging `desktop` directly from `laptop` will not work (try it out!). That is because routers `r1` and `r4` still need to be configured to tunnel packets to/from specific subnetworks through the respective GRE tunnel endpoints.

To configure tunneling in the home network, open a terminal on router `r1`, and run the following command:

```
:~$ ip route add 10.0.4.0/24 via 10.10.10.1
```

This will tell the router to tunnel all packets going to the office network (`10.0.4.0/24`) through the local GRE tunnel endpoint (`10.10.10.1`).

Q4. Configure tunneling in the office network by including the relevant command.

Make sure to validate your configuration before proceeding.

Q5. Start Wireshark on interface `eth0` of router `r3`. Open a terminal on `laptop` and fetch the webpage from the HTTP server by running `curl 10.0.3.10:8080`. Is the HMTL content encrypted? Why, or why not?

Q6. If you ping the `www` server (`10.0.5.10`) from the `laptop`, will these ICMP packets be GRE-encapsulated?

2.2 Encrypted tunnels

In the previous exercise, you used GRE, a tunneling protocol without encryption, to implement a simple VPN. For some specific cases, that approach might suffice, but in most real-world scenarios stronger security properties may be required (such as the ones described in the first section of the handout).

OpenVPN is an open source VPN system that implements techniques that allow to create a secure point-to-point tunnel over an untrusted network (such as the Internet). Typically, VPN clients will connect to a OpenVPN server (multiple clients can connect to the same server).

Node authentication can be done with a username and password, or using a public key infrastructure with certificates when stronger security is needed. Figure 4 illustrates a point-to-point OpenVPN tunnel.

Consider the same scenario from the previous exercise, where you require your desktop to be accessible from your office network. This time, you will configure OpenVPN in CORE to implement a simple VPN that connects your home network to your office network (with stronger security garanties, though).

Computer Networks — 2023/24, Q2		**Assignment:**	Lab 8
Virtual Private Networks		**Version:**	1.0
Issued:	2023-12-04	**Submission Due:**	2023-12-10

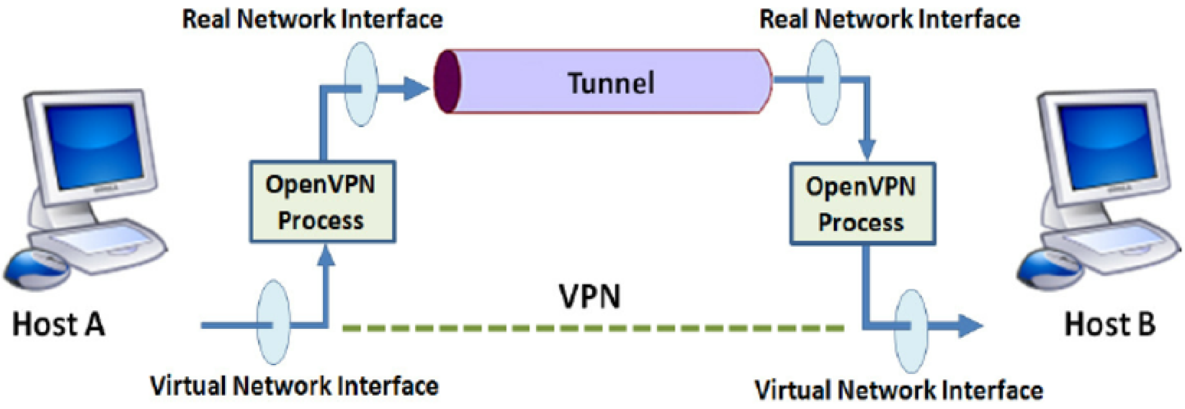

Figure 4: Point-to-point OpenVPN tunnel

Start by opening CORE and loading the network topology from file `vpn-2.imn`. Once again, before starting the session, take your time to analyze the network topology. This time, your home network looks a bit different: a `vpnserver` is managing connections from VPN clients to control remote accesses to your home network. That server has been assigned a public IPv4 address, meaning that your `laptop` should be able to reach it (try to ping it!).

Q7. Try to ping the desktop at home (`10.0.6.10`) from the laptop at the office (`10.0.0.20`). Why does ping fail?

To configure OpenVPN, let us start with the server. Open a terminal on `vpnserver` and list the contents of the directory (use `ls` for that). You will find multiple files there, but focus your attention on `server.conf` (you may want to analyze its contents). This file provides the configuration parameters used by OpenVPN to configure and launch the server, which should run over UDP at `10.0.1.10:1194`. Similarly to GRE, the OpenVPN server will create a dedicated tunnel. Differently from GRE, the tunnel is secure.

Try to start the server by running this on the terminal:

```
:~$ openvpn --config server.conf
```

Q8. The server fails to start because the configuration file is incomplete. What seems to be missing? Replace all TODOs with the appropriate configuration parameters (you can use `vim` to edit the file).

> **Hint.** Server-related operations will be logged under `server.log`. You may want to inspect the log if you suspect something is not working correctly.

Once you have the configuration working, start the server by repeating the command above (the program should seem to stall, meaning that the server is running; if it exits abruptly, it may be misconfigured).

Computer Networks — 2023/24, Q2		**Assignment:**	Lab 8
Virtual Private Networks		**Version:**	1.0
Issued:	2023-12-04	**Submission Due:**	2023-12-10

Q9. What is the IPv4 address of the server in the VPN?

Now it is time to configure the `laptop`, i.e. the VPN client. Open a terminal on `laptop` and list the contents of the directory. Like before, you will find multiple files there, but focus your attention on `client.conf`, which, you guessed, is used by OpenVPN to configure and launch the client.

Try to start the client by running this on the terminal:

```
:~$ openvpn --config client.conf
```

Q10. The client fails to start because the configuration file is incomplete. What seems to be missing? Replace all TODOs with the appropriate configuration parameters.

> **Hint.** Client-related operations will be logged under `client.log`. You may want to inspect the log if you suspect something is not working correctly.

Finally, start the client by repeating the command above (the program should seem to stall, meaning that the client is running; if it exits abruptly, it may be misconfigured). You can test if the tunnel has been successfully established by checking the connection between the client and the server (try to ping the server from your `laptop`).

Q11. Even though the OpenVPN tunnel is configured, pinging `desktop` directly from `laptop` will not work (try it out!). Try to fix this problem, and respond by including the relevant command you used.

Q12. Start Wireshark on interface `eth2` of router `r3`. Open a terminal on `laptop` and fetch the webpage from the HTTP server by running `curl 10.0.6.10:8080`. This time, can you see the contents of the HTML page? Why, or why not?