Routerlab

Summer semester 2018

Worksheet 6
Group 08

Valentin Franck, 364066
Nikhil Singh, 387694

Pages: 23

Submission Date: June 14, 2018

Question 1

1a

VPNs like OpenVPN and IPSec are used to establish a secure virtual private network between the two or more(!) networks over the public network like Internet while TLS/SSL (Transport Layer Security/Secure Socket Layer) is used for the secure web communication between the client (web browser) and web server through end to end encryption to avoid the malicious attacks such as session hijack. Both offer confidentiality, integrity and authenticity. IPsec and Open VPN operate at network layer of the OSI model while TLS above transport layer because it runs on top of TCP. VPNs are used to secure the network while TLS to secure the applications using the secured web tunnel. IPSec provides DES encryption to secure the network layer between two parties. OpenVPN uses the OpenSSL library to encrypt both data and control channels. It can also use HMAC packet authentication feature to append an extra layer of security. In TLS/SSL their is no permanent connectivity. Applications open a session and the server responds and when the application or client quits, the server quits the session. IPSec and OpenVPN can run over UDP and TCP both. However, TLS/SSL is not supporting UDP.

1b

IPSec includes multiple protocols like AH(Authentication Header), ESP(Encapsulated Security Protocol). AH mode provides authenticity by cryptographic checksum of packet's contents. It protects the non-changeable fields of an IP packet. However, when using NAT, it alters authenticated header fields. AH mode is scarcely used because it doesn't provide encryption. ESP on the other side encrypts IP packets and provides authenticity as well. It is used in two modes. Transport mode and tunnel mode. In transport mode it encrypts and authenticate payload of IP packet and it doesn't alter original IP header. It runs on end hosts. However, in tunnel mode it encrypts and authenticate the entire IP packet by encapsulating it within a new outer IP packet. It is generally used between gateways with ESP enabled. It supports NAT traversal. IPSec tunnel mode has 52 Bytes of ESP overhead and 24 Bytes of GRE overhead which means total overhead in tunnel mode is 76 Bytes. However, in transport mode ESP overhead is 52 Bytes and GRE Overhead is 4 Bytes which means total overhead is 56 Bytes.

1c

Generally IPSec in transport mode is used between end hosts. However, we will choose IPSec ESP in tunnel mode because it provides authenticity, integrity and confidentiality by encapsulating the entire original IP packet with a new packet header. It encrypts and authenticates entire IP packet and supports NAT traversal as well. Is is usually used for gateways. Transport mode is more common between end-hosts. However, this means that the IP header is not protected by ESP.

1d

Yes, IPSec can be run without encryption with authentication support using IPSec AH (Authentication Header) protocol. It uses a hash function and a pre-shared secret key to create a cryptographic checksum of the packet's contents thus providing integrity and authenticity. It uses sequence number and a sliding window to prevent replay attacks.

1e

One difference between openVPN and IPSec is that IPSec always works at the network layer (see above) while openVPN can encapsulate either Ethernet frames or IP packets. This means it can operate at different layers. In Bridging openVPN encypsulates Ethernet frames which will integrate the client host into the remote network (e.g. assignind an IP address and using other network protocols). Routing on the other hand works at layer 3 and creates a virtual subnet. It does not allow immediate access to the remote network, but relies on an IP gateway that forwards traffic. OpenVPN traffic is hard to distinguish from standard TLS encrypted traffic, because the complete payload is encrypted using SSL OpenVPN is best used with UDP over TCP (because TCP over TCP will slow things down unnecessarily). This means if an openVPN server runs on port 443 it is almost impossible to block this traffic, because it appears to be normal HTTPS traffic. This is important for censorship circumvention. IPSec on the other hand is easily recognized because of the headers it introduces in both transport and tunnel mode. In tunnelling we connect networks with each other not just point to point connections between hosts. The tunnel therefore usually exists between two gateways. As mentioned above we can use the routing table and firewall settings of the gateways to decide which traffic to forward where, i.e. which traffic should go through the tunnel.

1f

In TAP mode openVPN encypsulates traffic at layer 2. This allows us two bridge entire networks and not just point-to-point links as in TUN mode. Therefore it is best suitable to bridge two LANs across a layer 3 network. Using TAP mode clients will integrate completely into the remote network, allowing them to receive remote IP addresses using Broadcast etc. In TUN mode on the other hand only a point to point connection is established and traffic from the remote networks has to be routed through the points. Therefore no direct access to the remote networks is possible.

1g

IPSec uses an authenticated Diffie-Hellmann Key Exchange to get the keys required to provide integrity and confidentiality. This requires either a pre-shared key or certificates for authentication. The latter also requires a PKI. (This is a problem, which is not always easy to solve!)

IPSec uses a hash function and the shared key to create a cryptographic checksum of the packet's contents thus providing integrity and authenticity. It uses sequence numbers and a sliding window to prevent replay attacks. IPSec uses DES encryption of the payload to provide confidentiality. Note, that in ESP the header is not part of the data over which the integrity check value is computed. I AH it is secured (paritally), but AH does not provide confidentiality.

Since communicating parties have to perform the IKE (to exchange keys) and keep state of the derived key material, this can be considered a violation of the connectionlessness of IP.

1h

No, there are also other mechanisms, that do not rely on cryptography. One example is MPLS and the Label Stack feature, which allows end point to use predefined routes by providing a stack of labels from which one label is popped at every hop. The idea is that, with such predefined routes it is possible to send packets only via trusted networks. However, it requires support from the routers and it is not extremely secure to send the payload in plaintext, if something goes wrong or traffic is intercepted (between the trusted routers). MPLS in our opinion therefore should only be used to separate traffic (e.g. from different customers) and not to protect confidentiality. Another problem with this circuit switching like behavior is that it will easily mean that connections are impossible if only one link fails because the route is predefined and not dynamically changed for each packet as in packet switching.

Question 2

2a

We are using the topology as shown in Figure 1. We have configured the loadgens and lev-sc1 using the addresses specified in the Figure 1. We have configured the interfaces in group08-lg2 and group08-lg3 to operate with VLAN tags by using the below mentioned commands.

```
root@group08-lg2:~# ip addr add 10.8.1.102/24 dev eth1.1
root@group08-lg2:~# ip addr add 10.8.100.102/24 dev eth1.2
root@group08-lg3:~# ip addr add 10.8.100.103/24 dev eth1.2
root@group08-lg3:~# ip addr add 10.8.2.103/24 dev eth1.3
```

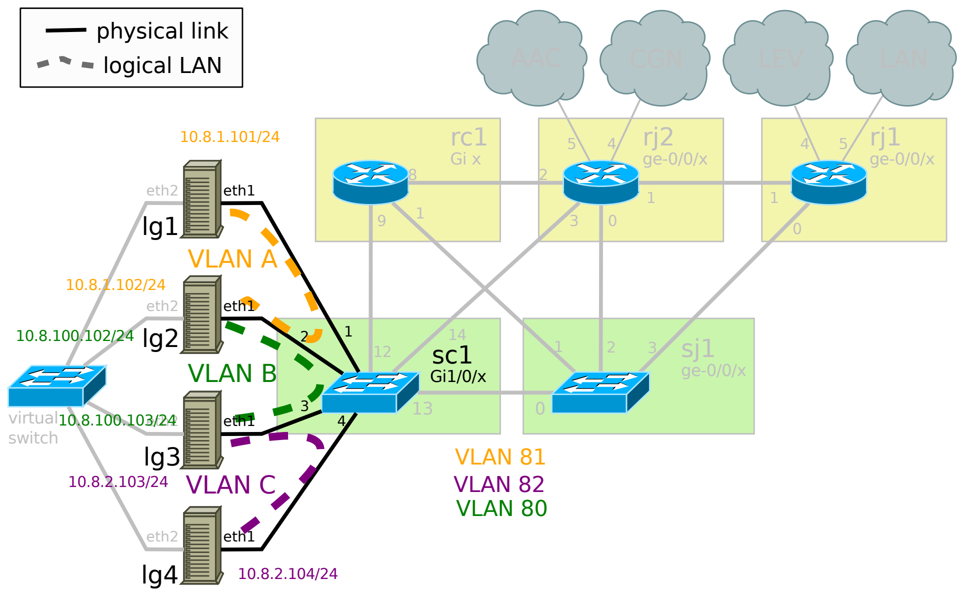

Figure 1: The topology we use for Question 2.

Please see the details below.

```
root@group08-lg1:~# ip a s
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    group default
     link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
     inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
     inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
    state UP group default qlen 1000
     link/ether 00:16:3e:af:08:10 brd ff:ff:ff:ff:ff:ff
     inet 172.16.0.208/20 brd 172.16.15.255 scope global eth0
        valid_lft forever preferred_lft forever
     inet6 fe80::216:3eff:feaf:810/64 scope link
        valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
    state UP group default qlen 1000
     link/ether 00:16:3e:af:08:11 brd ff:ff:ff:ff:ff:ff
     inet 10.8.1.101/24 scope global eth1
        valid_lft forever preferred_lft forever
     inet6 fe80::216:3eff:feaf:811/64 scope link
        valid_lft forever preferred_lft forever
4: eth2: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group
    default qlen 1000
     link/ether 00:16:3e:af:08:12 brd ff:ff:ff:ff:ff:ff

root@group08-lg2:~# ip a s
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    group default
     link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
     inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
     inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
    state UP group default qlen 1000
     link/ether 00:16:3e:af:08:20 brd ff:ff:ff:ff:ff:ff
     inet 172.16.0.209/20 brd 172.16.15.255 scope global eth0
        valid_lft forever preferred_lft forever
     inet6 fe80::216:3eff:feaf:820/64 scope link
        valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
    state UP group default qlen 1000
     link/ether 00:16:3e:af:08:21 brd ff:ff:ff:ff:ff:ff
     inet6 fe80::216:3eff:feaf:821/64 scope link
        valid_lft forever preferred_lft forever
4: eth2: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group
    default qlen 1000
     link/ether 00:16:3e:af:08:22 brd ff:ff:ff:ff:ff:ff
6: eth1.1@eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
    noqueue state UP group default
     link/ether 00:16:3e:af:08:21 brd ff:ff:ff:ff:ff:ff
     inet 10.8.1.102/24 scope global eth1.1
        valid_lft forever preferred_lft forever
     inet6 fe80::216:3eff:feaf:821/64 scope link
```

```
        valid_lft forever preferred_lft forever
7: eth1.2@eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
    noqueue state UP group default
     link/ether 00:16:3e:af:08:21 brd ff:ff:ff:ff:ff:ff
     inet 10.8.100.102/24 scope global eth1.2
        valid_lft forever preferred_lft forever
     inet6 fe80::216:3eff:feaf:821/64 scope link
        valid_lft forever preferred_lft forever

root@group08-lg3:~# ip a s
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    group default
     link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
     inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
     inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
    state UP group default qlen 1000
     link/ether 00:16:3e:af:08:30 brd ff:ff:ff:ff:ff:ff
     inet 172.16.0.210/20 brd 172.16.15.255 scope global eth0
        valid_lft forever preferred_lft forever
     inet6 fe80::216:3eff:feaf:830/64 scope link
        valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
    state UP group default qlen 1000
     link/ether 00:16:3e:af:08:31 brd ff:ff:ff:ff:ff:ff
     inet6 fe80::216:3eff:feaf:831/64 scope link
        valid_lft forever preferred_lft forever
4: eth2: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group
    default qlen 1000
     link/ether 00:16:3e:af:08:32 brd ff:ff:ff:ff:ff:ff
5: eth1.2@eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
    noqueue state UP group default
     link/ether 00:16:3e:af:08:31 brd ff:ff:ff:ff:ff:ff
     inet 10.8.100.103/24 scope global eth1.2
        valid_lft forever preferred_lft forever
     inet6 fe80::216:3eff:feaf:831/64 scope link
        valid_lft forever preferred_lft forever
6: eth1.3@eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
    noqueue state UP group default
     link/ether 00:16:3e:af:08:31 brd ff:ff:ff:ff:ff:ff
     inet 10.8.2.103/24 scope global eth1.3
        valid_lft forever preferred_lft forever
     inet6 fe80::216:3eff:feaf:831/64 scope link
        valid_lft forever preferred_lft forever

root@group08-lg4:~# ip a s
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    group default
     link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
     inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
     inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group
    default qlen 1000
     link/ether 00:16:3e:af:08:40 brd ff:ff:ff:ff:ff:ff
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
    state UP group default qlen 1000
     link/ether 00:16:3e:af:08:41 brd ff:ff:ff:ff:ff:ff
     inet 10.8.2.104/24 scope global eth1
        valid_lft forever preferred_lft forever
     inet6 fe80::216:3eff:feaf:841/64 scope link
        valid_lft forever preferred_lft forever
4: eth2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
    state UP group default qlen 1000
     link/ether 00:16:3e:af:08:42 brd ff:ff:ff:ff:ff:ff
     inet6 fe80::216:3eff:feaf:842/64 scope link
        valid_lft forever preferred_lft forever

lev-sc1#show running-config
Building configuration...
interface GigabitEthernet1/0/1
 description lg1
 switchport access vlan 81
 switchport mode access
!
interface GigabitEthernet1/0/2
 description lg2
 switchport mode trunk
!
interface GigabitEthernet1/0/3
 description lg3
 switchport mode trunk
!
interface GigabitEthernet1/0/4
 description lg4
 switchport access vlan 82
 switchport mode access
```

2b

We have configured the switch and interfaces of group08-lg2 and group08-lg3 so they could communicate and pinged the devices. Please check 2(a) for the configuration output and below for the output after pinging the devices.

```
root@group08-lg2:~# ping 10.8.100.103
PING 10.8.100.103 (10.8.100.103) 56(84) bytes of data.
64 bytes from 10.8.100.103: icmp_seq=1 ttl=64 time=0.484 ms
64 bytes from 10.8.100.103: icmp_seq=2 ttl=64 time=0.519 ms
64 bytes from 10.8.100.103: icmp_seq=3 ttl=64 time=0.669 ms
64 bytes from 10.8.100.103: icmp_seq=4 ttl=64 time=0.507 ms
^C
--- 10.8.100.103 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2998ms
rtt min/avg/max/mdev = 0.484/0.544/0.669/0.078 ms

root@group08-lg3:~# ping 10.8.100.102
PING 10.8.100.102 (10.8.100.102) 56(84) bytes of data.
64 bytes from 10.8.100.102: icmp_seq=1 ttl=64 time=0.613 ms
64 bytes from 10.8.100.102: icmp_seq=2 ttl=64 time=0.529 ms
64 bytes from 10.8.100.102: icmp_seq=3 ttl=64 time=0.551 ms
```

```
64 bytes from 10.8.100.102: icmp_seq=4 ttl=64 time=0.595 ms
^C
--- 10.8.100.102 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2997ms
rtt min/avg/max/mdev = 0.529/0.572/0.613/0.033 ms
```

2c

First we create the pre-shared key and transfer it to both loadgens:

```
openvpn --genkey --secret static.key
```

Then we configure both server and client:

```
root@group08-lg2:~# cat /etc/openvpn/tun0.conf
dev tun
ifconfig 10.8.10.102 10.8.10.103
secret /etc/openvpn/static.key

root@group08-lg3:~# cat /etc/openvpn/tun0.conf
remote 10.8.100.102
dev tun
ifconfig 10.8.10.103 10.8.10.102
secret /etc/openvpn/static.key
```

And finally run them successfully:

```
...
Mon Jun 11 16:49:25 2018 Peer Connection Initiated with [AF_INET
    ]10.8.100.102:1194
Mon Jun 11 16:49:26 2018 Initialization Sequence Completed
...
```

2d

We perform the ping on lg2 using the virtual interface like this:

```
root@group08-lg2:~# ping 10.8.10.103
PING 10.8.10.103 (10.8.10.103) 56(84) bytes of data.
64 bytes from 10.8.10.103: icmp_seq=1 ttl=64 time=0.987 ms
64 bytes from 10.8.10.103: icmp_seq=2 ttl=64 time=1.27 ms
```

On lg2 we can see the openvpn packets which are encrypted:

```
root@group08-lg3:~# tcpdump -i eth1 net 10.8.100.0/24
[30462.952701] device eth1 entered promiscuous mode
tcpdump: verbose output suppressed, use -v or -vv for full protocol
    decode
listening on eth1, link-type EN10MB (Ethernet), capture size 262144
    bytes
17:17:59.151002 IP 10.8.100.102.openvpn > 10.8.100.103.openvpn: UDP,
    length 124
17:17:59.151298 IP 10.8.100.103.openvpn > 10.8.100.102.openvpn: UDP,
    length 124
17:18:00.152414 IP 10.8.100.102.openvpn > 10.8.100.103.openvpn: UDP,
    length 124
17:18:00.152753 IP 10.8.100.103.openvpn > 10.8.100.102.openvpn: UDP,
    length 124
```

And we can see the packets decrypted at the virtual interface:

```
root@group08-lg3:~# tcpdump -i tun0
[30671.235312] device tun0 entered promiscuous mode
tcpdump: verbose output suppressed, use -v or -vv for full protocol
    decode
listening on tun0, link-type RAW (Raw IP), capture size 262144 bytes
17:15:13.401283 IP 10.8.10.102 > 10.8.10.103: ICMP echo request, id
    6715, seq 1, length 64
17:15:13.401325 IP 10.8.10.103 > 10.8.10.102: ICMP echo reply, id 6715,
     seq 1, length 64
17:15:14.402933 IP 10.8.10.102 > 10.8.10.103: ICMP echo request, id
    6715, seq 2, length 64
17:15:14.402971 IP 10.8.10.103 > 10.8.10.102: ICMP echo reply, id 6715,
     seq 2, length 64
```

What happens is, that the encapsulated packet arrives at eth1, where it is only recognized as a UCP packet which is directed to openvpn. However it is then preocessed and sent to the virtual interface tun0 where we can see it again in the second capture. This time it is decrypted and we can see the ICMP packet. Then the applciation generates the reply which will take the same path in reverse order.

2e

Here are the routes we configured:

```
root@group08-lg1:~# ip route
default via 10.8.1.102 dev eth1
10.8.1.0/24 dev eth1 proto kernel scope link src 10.8.1.101
172.16.0.0/20 dev eth0 proto kernel scope link src 172.16.0.208
172.16.255.0/24 via 172.16.0.2 dev eth0

root@group08-lg2:~# ip route
10.8.1.0/24 dev eth1.1 proto kernel scope link src 10.8.1.102
10.8.2.0/24 via 10.8.10.103 dev tun0
10.8.10.103 dev tun0 proto kernel scope link src 10.8.10.102
10.8.100.0/24 dev eth1.2 proto kernel scope link src 10.8.100.102
172.16.0.0/20 dev eth0 proto kernel scope link src 172.16.0.209
172.16.255.0/24 via 172.16.0.2 dev eth0

root@group08-lg3:~# ip route
10.8.1.0/24 via 10.8.10.102 dev tun0
10.8.2.0/24 dev eth1.3 proto kernel scope link src 10.8.2.103
10.8.10.102 dev tun0 proto kernel scope link src 10.8.10.103
10.8.100.0/24 dev eth1.2 proto kernel scope link src 10.8.100.103
172.16.0.0/20 dev eth0 proto kernel scope link src 172.16.0.210
172.16.255.0/24 via 172.16.0.2 dev eth0

root@group08-lg4:~# ip route
default via 10.8.2.103 dev eth1
10.8.2.0/24 dev eth1 proto kernel scope link src 10.8.2.104
172.16.0.0/20 dev eth0 proto kernel scope link src 172.16.0.211
172.16.255.0/24 via 172.16.0.2 dev eth0
```

And here we provide some traceroutes and pings to prove everything works and uses the openvpn on LG1:

```
root@group08-lg1:~# traceroute 10.8.2.104
traceroute to 10.8.2.104 (10.8.2.104), 30 hops max, 60 byte packets
 1  10.8.1.102 (10.8.1.102)  0.820 ms  0.774 ms  0.742 ms
 2   10.8.10.103  (10.8.10.103)   2.792  ms   2.780  ms   2.748  ms
 3   10.8.2.104  (10.8.2.104)   2.736  ms   2.758  ms   2.728  ms

root@group08-lg1:~# ping 10.8.2.104
PING 10.8.2.104 (10.8.2.104) 56(84) bytes of data.
64 bytes from 10.8.2.104: icmp_seq=1 ttl=62 time=2.30 ms
64 bytes from 10.8.2.104: icmp_seq=2 ttl=62 time=1.75 ms
64 bytes from 10.8.2.104: icmp_seq=3 ttl=62 time=1.87 ms
^C
--- 10.8.2.104 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 1.758/1.979/2.307/0.239 ms
```

On LG4:

```
root@group08-lg4:~# traceroute 10.8.1.101
traceroute to 10.8.1.101 (10.8.1.101), 30 hops max, 60 byte packets
 1   10.8.2.103  (10.8.2.103)   0.749  ms   0.700  ms   0.668  ms
 2   10.8.10.102  (10.8.10.102)   2.569  ms   2.547  ms   2.579  ms
 3   10.8.1.101  (10.8.1.101)   2.662  ms   2.627  ms   2.588  ms

root@group08-lg4:~# ping 10.8.1.101
PING 10.8.1.101 (10.8.1.101) 56(84) bytes of data.
64 bytes from 10.8.1.101: icmp_seq=1 ttl=62 time=2.01 ms
64 bytes from 10.8.1.101: icmp_seq=2 ttl=62 time=1.92 ms
64 bytes from 10.8.1.101: icmp_seq=3 ttl=62 time=2.10 ms
^C
--- 10.8.1.101 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 1.927/2.014/2.106/0.081 ms
```

For the dumps see question 2f.

2f

Here we provide a few lines of tcpdump to show no unencrypted traffic is passing vlan80:

```
root@group08-lg2:~# tcpdump -i eth1.2 net 0.0.0.0/0
[107838.988168] device eth1.2 entered promiscuous mode
[107838.988188] device eth1 entered promiscuous mode
tcpdump: verbose output suppressed, use -v or -vv for full protocol
    decode
listening on eth1.2, link-type EN10MB (Ethernet), capture size 262144
    bytes
14:41:27.764848 IP 10.8.100.102.openvpn > 10.8.100.103.openvpn: UDP,
    length 124
14:41:27.765894 IP 10.8.100.103.openvpn > 10.8.100.102.openvpn: UDP,
    length 124
14:41:28.766048 IP 10.8.100.102.openvpn > 10.8.100.103.openvpn: UDP,
    length 124
14:41:28.767447 IP 10.8.100.103.openvpn > 10.8.100.102.openvpn: UDP,
    length 124
14:41:29.767503 IP 10.8.100.102.openvpn > 10.8.100.103.openvpn: UDP,
    length 124
14:41:29.768626 IP 10.8.100.103.openvpn > 10.8.100.102.openvpn: UDP,
    length 124
14:41:32.765241 ARP, Request who-has 10.8.100.103 tell 10.8.100.102,
    length 28
```

```
14:41:32.765862 ARP, Reply 10.8.100.103 is-at 00:16:3e:af:08:31 (oui
    Unknown), length 46
14:41:40.528195 IP 10.8.100.102.openvpn > 10.8.100.103.openvpn: UDP,
    length 100
14:41:40.528325 IP 10.8.100.102.openvpn > 10.8.100.103.openvpn: UDP,
    length 100
14:41:40.528374 IP 10.8.100.102.openvpn > 10.8.100.103.openvpn: UDP,
    length 100
14:41:40.528418 IP 10.8.100.102.openvpn > 10.8.100.103.openvpn: UDP,
    length 100
14:41:40.528462 IP 10.8.100.102.openvpn > 10.8.100.103.openvpn: UDP,
    length 100
14:41:40.528505 IP 10.8.100.102.openvpn > 10.8.100.103.openvpn: UDP,
    length 100
14:41:40.528547 IP 10.8.100.102.openvpn > 10.8.100.103.openvpn: UDP,
    length 100
14:41:40.529473 IP 10.8.100.103.openvpn > 10.8.100.102.openvpn: UDP,
    length 132
14:41:40.529509 IP 10.8.100.103.openvpn > 10.8.100.102.openvpn: UDP,
    length 132
14:41:40.529523 IP 10.8.100.103.openvpn > 10.8.100.102.openvpn: UDP,
    length 132
14:41:40.529666 IP 10.8.100.103.openvpn > 10.8.100.102.openvpn: UDP,
    length 132
14:41:40.529689 IP 10.8.100.103.openvpn > 10.8.100.102.openvpn: UDP,
    length 132
14:41:40.529767 IP 10.8.100.103.openvpn > 10.8.100.102.openvpn: UDP,
    length 132
14:41:40.529778 IP 10.8.100.103.openvpn > 10.8.100.102.openvpn: UDP,
    length 132
14:41:40.531011 IP 10.8.100.102.openvpn > 10.8.100.103.openvpn: UDP,
    length 100
14:41:40.531070 IP 10.8.100.102.openvpn > 10.8.100.103.openvpn: UDP,
    length 100
14:41:40.531114 IP 10.8.100.102.openvpn > 10.8.100.103.openvpn: UDP,
    length 100
14:41:45.539233 ARP, Request who-has 10.8.100.102 tell 10.8.100.103,
    length 46
14:41:45.539263 ARP, Reply 10.8.100.102 is-at 00:16:3e:af:08:21 (oui
    Unknown), length 28
14:41:55.889227 IP 10.8.100.103.openvpn > 10.8.100.102.openvpn: UDP,
    length 100
14:41:55.889302 IP 10.8.100.103.openvpn > 10.8.100.102.openvpn: UDP,
    length 100
14:41:55.889355 IP 10.8.100.103.openvpn > 10.8.100.102.openvpn: UDP,
    length 100
14:41:55.889366 IP 10.8.100.103.openvpn > 10.8.100.102.openvpn: UDP,
    length 100
14:41:55.889385 IP 10.8.100.103.openvpn > 10.8.100.102.openvpn: UDP,
    length 100
14:41:55.889552 IP 10.8.100.103.openvpn > 10.8.100.102.openvpn: UDP,
    length 100
14:41:55.889562 IP 10.8.100.103.openvpn > 10.8.100.102.openvpn: UDP,
    length 100
14:41:55.891609 IP 10.8.100.103.openvpn > 10.8.100.102.openvpn: UDP,
    length 100
```

```
14:41:55.891638 IP 10.8.100.103.openvpn > 10.8.100.102.openvpn: UDP,
    length 100
14:41:55.891644 IP 10.8.100.103.openvpn > 10.8.100.102.openvpn: UDP,
    length 100
14:42:03.680785 IP 10.8.100.103.openvpn > 10.8.100.102.openvpn: UDP,
    length 124
14:42:03.681651 IP 10.8.100.102.openvpn > 10.8.100.103.openvpn: UDP,
    length 124
14:42:04.682176 IP 10.8.100.103.openvpn > 10.8.100.102.openvpn: UDP,
    length 124
14:42:04.683099 IP 10.8.100.102.openvpn > 10.8.100.103.openvpn: UDP,
    length 124
14:42:05.683931 IP 10.8.100.103.openvpn > 10.8.100.102.openvpn: UDP,
    length 124
14:42:05.684913 IP 10.8.100.102.openvpn > 10.8.100.103.openvpn: UDP,
    length 124
14:42:16.531639 IP 10.8.100.103.openvpn > 10.8.100.102.openvpn: UDP,
    length 124
14:42:16.532090 IP 10.8.100.102.openvpn > 10.8.100.103.openvpn: UDP,
    length 124
14:42:17.533824 IP 10.8.100.103.openvpn > 10.8.100.102.openvpn: UDP,
    length 124
14:42:17.534315 IP 10.8.100.102.openvpn > 10.8.100.103.openvpn: UDP,
    length 124
14:42:18.534821 IP 10.8.100.103.openvpn > 10.8.100.102.openvpn: UDP,
    length 124
14:42:18.535336 IP 10.8.100.102.openvpn > 10.8.100.103.openvpn: UDP,
    length 124
14:42:21.533245 ARP, Request who-has 10.8.100.103 tell 10.8.100.102,
    length 28
14:42:21.533796 ARP, Reply 10.8.100.103 is-at 00:16:3e:af:08:31 (oui
    Unknown), length 46
^C
51 packets captured
74 packets received by filter
23 packets dropped by kernel
[108189.314252] device eth1.2 left promiscuous mode
[108189.314271] device eth1 left promiscuous mode
```

Question 3

3a

We are using the topology as shown in Figure 2. We have configured two VLAN's on lev-sc1. VLAN 81 using access mode on interface Gi1/0/1. VLAN 84 using access mode on interface Gi1/0/4 and both of them using trunk mode on interface Gi1/0/14. And pinged lg4 from lg1 and vice versa. Please find below the appropriate configuration output and response after pinging the loadgens.

```
root@lev-rj2> show configuration
    ge-0/0/3 {
        description lev-sc1;
        vlan-tagging;
        unit 81 {
            vlan-id 81;
            family inet {
                address 10.8.1.1/24;
```

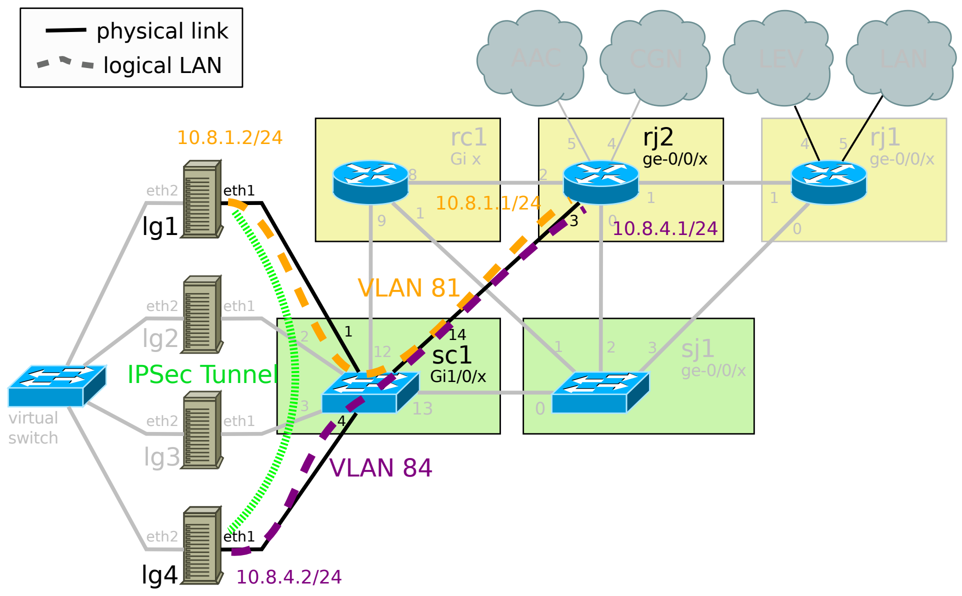

Figure 2: The topology we use for Question 3.

```
            }
        }
        unit 84 {
            vlan-id 84;
            family inet {
                address 10.8.4.1/24;
            }
        }
    }

lev-sc1#show running-config
!
interface GigabitEthernet1/0/1
 description lg1
 switchport access vlan 81
 switchport mode access
 !
interface GigabitEthernet1/0/4
 description lg4
 switchport access vlan 84
 switchport mode access
 !
interface GigabitEthernet1/0/14
 description lev-rj2
 switchport mode trunk

root@group08-lg1:~# ip a s
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
    state UP group default qlen 1000
     link/ether 00:16:3e:af:08:11 brd ff:ff:ff:ff:ff:ff
     inet 10.8.1.2/24 scope global eth1
        valid_lft forever preferred_lft forever
     inet6 fe80::216:3eff:feaf:811/64 scope link
        valid_lft forever preferred_lft forever

root@group08-lg4:~# ip a s
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
    state UP group default qlen 1000
     link/ether 00:16:3e:af:08:41 brd ff:ff:ff:ff:ff:ff
     inet 10.8.4.2/24 scope global eth1
        valid_lft forever preferred_lft forever
     inet6 fe80::216:3eff:feaf:841/64 scope link
        valid_lft forever preferred_lft forever

root@group08-lg1:~# ip route add 10.8.4.0/24 via 10.8.1.1 dev eth1
root@group08-lg4:~# ip route add 10.8.1.0/24 via 10.8.4.1 dev eth1



root@group08-lg1:~# ping 10.8.4.2
PING 10.8.4.2 (10.8.4.2) 56(84) bytes of data.
64 bytes from 10.8.4.2: icmp_seq=1 ttl=63 time=1.11 ms
64 bytes from 10.8.4.2: icmp_seq=2 ttl=63 time=1.28 ms
64 bytes from 10.8.4.2: icmp_seq=3 ttl=63 time=1.34 ms
64 bytes from 10.8.4.2: icmp_seq=4 ttl=63 time=1.06 ms
^C
```

```
--- 10.8.4.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 1.062/1.203/1.349/0.122 ms

root@group08-lg4:~# ping 10.8.1.2
PING 10.8.1.2 (10.8.1.2) 56(84) bytes of data.
64 bytes from 10.8.1.2: icmp_seq=1 ttl=63 time=1.18 ms
64 bytes from 10.8.1.2: icmp_seq=2 ttl=63 time=1.27 ms
64 bytes from 10.8.1.2: icmp_seq=3 ttl=63 time=1.29 ms
64 bytes from 10.8.1.2: icmp_seq=4 ttl=63 time=1.40 ms
^C
--- 10.8.1.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 1.187/1.290/1.406/0.089 ms
```

3b

- /etc/ipsec.conf is used to configure IPSec related parameters, which are not directly security related. In our case it configures, which hosts should be connecting based on ID, IP addresse etc. and which authentication keys should be used. It also defines that ikev2 is used and the ikev1 daemon is not started, no fresh CRL needs to be loaded before each connection...
- /etc/ipsec.secrets is used to configure the type of the secret s for authentication (RSA key etc.) and path to files where they are stored. It may also contain some directives and should be protected.
- /etc/strongswan.conf is used to configure strongswan applications especially other than IPSec related ones. In our case it configures, which cryptographic algorithms should be loaded and that multiple authentication is not possible.
- ipsec statusall - This command gives detailed information either on a connection or all connections. It is implemented by calling the ipsec stroke command where stroke is used to control and monitor IPSec connections.
- ipsec listall - It provides all the information generated by individual list commands such as ipsec listpubkeys, ipsec listplugins, ipsec listgroups etc.
- ip -s xfrm policy - ip-xfrm is used for transforming packets, which is obviously needed with the encapsulation happening when using IPSec. The given command gives detailed stats on what the security policy for these transformations is.
- ipsec up <conection-name> - tells the IKE daemon to start the named connection
- service ipsec restart - terminates all connections and the IKE daemon, then starts them again by using the parameters in the ipsec.conf file

3c

The relevant configuration we used (it is identical on both except for the ipsec.conf file, where we switched left and right on lg4):

```
root@group08-lg1:~# cat /etc/strongswan.conf
# strongswan.conf - strongSwan configuration file
#
# Refer to the strongswan.conf(5) manpage for details
#
# Configuration changes should be made in the included files

charon {
        load_modular = yes
        plugins {
                include strongswan.d/charon/*.conf
        }
}

include strongswan.d/*.conf

root@group08-lg1:~# cat /etc/ipsec.secrets
10.8.1.2 10.8.4.2 : PSK sv+NkxY9LLZvwj4qCC2o/gGrWDF2d21jL

root@group08-lg1:~# cat /etc/ipsec.conf
# /etc/ipsec.conf - strongSwan IPsec configuration file

conn %default
        ikelifetime=60m
        keylife=20m
        rekeymargin=3m
        keyingtries=1
        authby=secret
        keyexchange=ikev2
        mobike=no

conn host-host
        left=10.8.1.2
        leftfirewall=yes
        right=10.8.4.2
        type=transport
        auto=add
```

This is how we launched the IPSec connection (again the same on both loadgens):

```
root@group08-lg1:~# ipsec restart
Stopping strongSwan IPsec...
Starting strongSwan 5.5.1 IPsec [starter]...
root@group08-lg1:~# ipsec up host-host
initiating IKE_SA host-host[1] to 10.8.4.2
generating IKE_SA_INIT request 0 [ SA KE No N(NATD_S_IP) N(NATD_D_IP) N
    (FRAG_SUP) N(HASH_ALG) N(REDIR_SUP) ]
sending packet: from 10.8.1.2[500] to 10.8.4.2[500] (1300 bytes)
received packet: from 10.8.4.2[500] to 10.8.1.2[500] (592 bytes)
parsed IKE_SA_INIT response 0 [ SA KE No N(NATD_S_IP) N(NATD_D_IP) N(
    FRAG_SUP) N(HASH_ALG) N(MULT_AUTH) ]
authentication of '10.8.1.2' (myself) with pre-shared key
establishing CHILD_SA host-host
generating IKE_AUTH request 1 [ IDi N(INIT_CONTACT) IDr AUTH N(
    USE_TRANSP) SA TSi TSr N(MULT_AUTH) N(EAP_ONLY) ]
sending packet: from 10.8.1.2[500] to 10.8.4.2[500] (384 bytes)
received packet: from 10.8.4.2[500] to 10.8.1.2[500] (240 bytes)
parsed IKE_AUTH response 1 [ IDr AUTH N(USE_TRANSP) SA TSi TSr N(
    AUTH_LFT) ]
authentication of '10.8.4.2' with pre-shared key successful
IKE_SA host-host[1] established between
    10.8.1.2[10.8.1.2]...10.8.4.2[10.8.4.2]
scheduling reauthentication in 3352s
maximum IKE_SA lifetime 3532s
connection 'host-host' established successfully
```

3d

For some reason tcpdump at rj2 would not show the ping packets (only VSTP packets were observed). So we captured the traffic on lg4 as explained in Theresa's email:

```
root@group08-lg4:~# tcpdump -i eth1 net 0.0.0.0/0
[ 5040.551934] device eth1 entered promiscuous mode
tcpdump: verbose output suppressed, use -v or -vv for full protocol
    decode
listening on eth1, link-type EN10MB (Ethernet), capture size 262144
    bytes
11:05:25.102028 IP 10.8.1.2 > 10.8.4.2: ESP(spi=0xc9b7b956,seq=0x76),
    length 120
11:05:25.102148 IP 10.8.4.2 > 10.8.1.2: ESP(spi=0xc15e0e0c,seq=0x80),
    length 120
11:05:25.463303 IP 10.8.1.2 > 10.8.4.2: ESP(spi=0xc9b7b956,seq=0x77),
    length 120
11:05:25.463430 IP 10.8.4.2 > 10.8.1.2: ESP(spi=0xc15e0e0c,seq=0x81),
    length 120
11:05:26.103493 IP 10.8.1.2 > 10.8.4.2: ESP(spi=0xc9b7b956,seq=0x78),
    length 120
11:05:26.103616 IP 10.8.4.2 > 10.8.1.2: ESP(spi=0xc15e0e0c,seq=0x82),
    length 120
11:05:26.464574 IP 10.8.1.2 > 10.8.4.2: ESP(spi=0xc9b7b956,seq=0x79),
    length 120
11:05:26.464649 IP 10.8.4.2 > 10.8.1.2: ESP(spi=0xc15e0e0c,seq=0x83),
    length 120
11:05:27.104773 IP 10.8.1.2 > 10.8.4.2: ESP(spi=0xc9b7b956,seq=0x7a),
    length 120
11:05:27.104945 IP 10.8.4.2 > 10.8.1.2: ESP(spi=0xc15e0e0c,seq=0x84),
    length 120
11:05:27.465710 IP 10.8.1.2 > 10.8.4.2: ESP(spi=0xc9b7b956,seq=0x7b),
    length 120
11:05:27.465838 IP 10.8.4.2 > 10.8.1.2: ESP(spi=0xc15e0e0c,seq=0x85),
    length 120
^C
12 packets captured
12 packets received by filter
0 packets dropped by kernel
[ 5043.818552] device eth1 left promiscuous mode
```

Because we use encrypted ESP we cannot see that this is ICMP traffic, but only see that it is traffic using IPSec ESP. However we can observe the traffic pattern (request-reply) and we can see the metadata such as length, sequence numbers, which also might give away some information, as this is a typical pattern for ping etc.).

3e

In the traceroute we can see that only one hop appears and that is the destination:

```
oot@group08-lg4:~# traceroute 10.8.1.2
traceroute to 10.8.1.2 (10.8.1.2), 30 hops max, 60 byte packets
 1  * * *
 2  10.8.1.2 (10.8.1.2)  0.632 ms  0.606 ms  0.557 ms
```

In the previous question we were able to see all hops and also see the tunnel in action (because of the changed IP addresses from the virtual interface). This time we are unable to see the exact path, but only see the destination hop. This is because of the encapsulation of the IP packet generated by traceroute in IPSecs ESP packet.

Question 4

4a

There are different protocol to tunnel IPv6 over an IPv4 network, e.g. 6to4 and 6in4. As explained in RFC 4213 the IPv6 packet is encapsulated in an IPv4 packet, i.e. at the beginning of the packet an IPv4 header is added. The encapsulation can happen between routers and/ or hosts, that create the tunnel and address each other with the added IPv4 header. Note, that the reply does not necessarily have to take the same path! 6to4 on top of that also assigns IPv6 addresses to all globally reachable IPv4 hosts and is able to route traffic to native IPv6 networks.

The most important difference of automatic tunnels is that they are no point to point tunnels as manually configured ones are. This is because the underlying IPv4 network is treated as a non-broadcast multiaccess network. The benefit of automatic 6to4 tunneling is that scales better when the number of IPv6 sites to be reachable in the IPv4 network is large, because there is no need to configure each of these tunnels manually.

Configured tunneling works by manual setup and supports the following kind of links: host-to-router, router-to-host, host-to-host and router-to-router (the most common one to be manually configured). Tunnel brokers are services that provide the IPv6 tunnel, typically in order to allow end users to access IPv6 sites over the IPv4 network. 6in4 is used as part of the 6to4 technology and refers to the encapsulation of IPv6 traffic in IPv4 datagrams

4b

The problem is that IPv4 addresses are assigned dynamically, while one of the advantages of IPv6 is that there is no address shortage, which would in theory allow private customers to get static IPv6 addresses. With the way IPv6 address are derived from IPv4 addresses to support tunneling (see 4c) it is not possible to get a static IPv6 address. Also the tunnel would have to be reconfigured frequently due to the change of the customers IPv4 address. Therefore in such a scenario automatic configuration should be used. Manual configuration is better suitable for a scenario where IPv4 addresses are static (e.g. routers, servers etc.).

4c

6to4 addresses are derived by using 2002::/16 as a prefix. Then the IPv4 address is used. Then the 16bit Service Level Agreement ID (corresponding to a subnet ID) and the 64 interface identifier derived from the MAC address is appended. This means after appending the IPv4 address to the 16 bit prefix we basically get a normal /48 address.

4d

The default IPv4 address for a 6to4 relay is 192.88.99.1 (and the default IPv6 router address is 2002:c058:6301:: for a 6to4 site). There is a whole number of problems mentioned in the RFC.

Empirically the 6to4 connections have proven to be unreliable and frequently fail (up to 20% rate). The RFC discusses possible reasons for these problems such as firewalls/ACLs dropping the respective traffic, falsely configured routing tables and misconfigurations leading to embed private IPv4 addresses in 6to4, which will make any reply impossible.

The outbound black hole occurs when the ISP accepts user traffic to 192.88.99.0/24, but it actually is not configured to go anywhere useful or traffic is dropped.

The inbound black hole occurs when traffic to 192.88.99.0/24 is correctly routed, but the reply is dropped at some point (usually at the network from which the initial datagram was sent). This is a common problem seen at IPv6 servers, which will only see the SYN, because their SYNACK will never reach the counterpart host.

The No-Return-Relay occurs when the traffic reaches the IPv6 destination but the reply is not routed by the relay on the way back.

Due to complex paths it is possible to experience large RTTs.

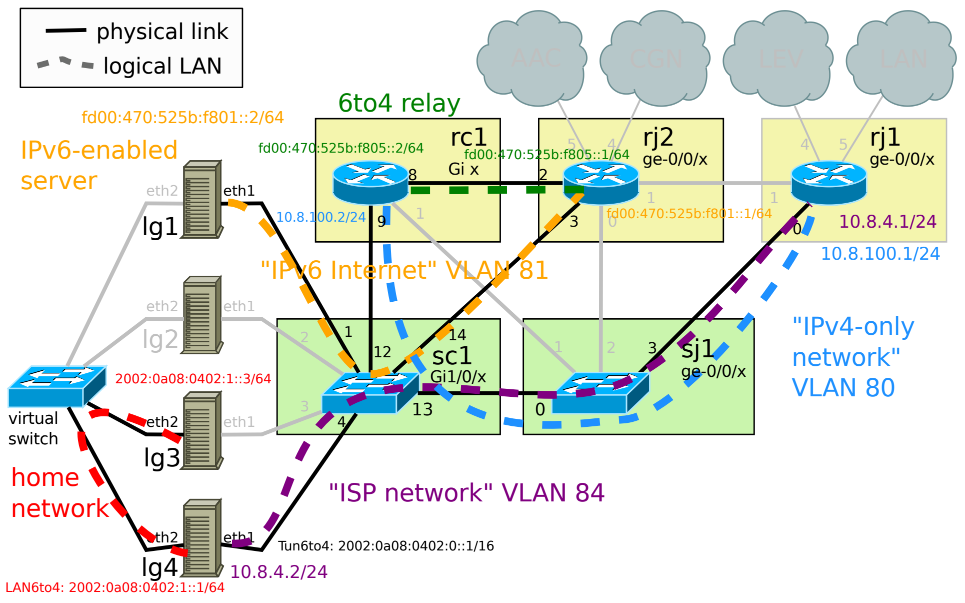

Figure 3: The topology we use for Question 5.

Sometimes Path MTU Discovery does not work and servers will sent packets back with too large MTU exceeding the 1500 bytes because of encapsulation header. Such traffic will be dropped.

Reverse DNS will fail for 6to4 hosts typically.

Sometimes routers falsely announce 2002::/16 prefixes.

The RFC mentions a few more similar problems. However, we point out that these problems become more severe by the fact that it is hard to diagnose the origin of the failures.

Question 5

Please note that we had to do parts of this question on cloud cgn instead of our assigned cloud lev, because it was not available.

5a

We are using the topology as shown in Figure 3.

5b

```
root@group08-lg3:~# ping -I eth2 -c 1 fe80::216:3eff:feaf:842
PING fe80::216:3eff:feaf:842(fe80::216:3eff:feaf:842) from fe80::216:3
    eff:feaf:832%eth2 eth2: 56 data bytes
64 bytes from fe80::216:3eff:feaf:842%eth2: icmp_seq=1 ttl=64 time
    =0.398 ms
```

5c

We configure the devices like this (assigned IP addresses are indicated in Figure 3):

```
root@group08-lg1:~# ip -6 route add default via fd00:470:525b:f801::1
    dev eth1

root@group08-lg4:~# ip route add default via 10.8.4.1 dev eth1

lev-rc1(config)#ip route 10.8.4.0 255.255.255.0 10.8.100.1

root@group08-lg4:~# traceroute 10.8.100.2
traceroute to 10.8.100.2 (10.8.100.2), 30 hops max, 60 byte packets
 1  10.8.4.1 (10.8.4.1)  0.563 ms  0.517 ms  0.531 ms
 2  10.8.100.2 (10.8.100.2)  1.131 ms * *
```

5d

```
lev-rc1(config)#ipv6 route fd00:470:525b:f801::0/64 fd00:470:525b:f805
    ::1

root@group08-lg1:~# ip route add fd00:470:525b:f805::0/64 via fd00
    :470:525b:f801::1 dev eth1

root@group08-lg1:~# traceroute6 fd00:470:525b:f805::2
traceroute to fd00:470:525b:f805::2 (fd00:470:525b:f805::2), 30 hops
    max, 80 byte packets
 1  fd00:470:525b:f801::1 (fd00:470:525b:f801::1)  0.649 ms  0.559 ms
     0.539 ms
 2  fd00:470:525b:f805::2 (fd00:470:525b:f805::2)  1.222 ms  1.186 ms
     1.159 ms
```

5e

```
root@group08-lg4:~# ./script.sh 10.8.4.2
IPv4 address: 10.8.4.2
Tunnel6to4: 2002:0a08:0402:0::1/16, LAN6to4: 2002:0a08:0402:1::1/64
```

The script computes the V6PREFIX by converting the given WAN IPv4 addres to hexadecimal and appending it to "2002:". It will then append the respective it will then append the default subnet IDs and interface identifiers including the prefix-length to provide the complete Tunnel6t04 and LAN6to4 addresses.

5f

We configured everything as explained in the worksheet using Loopback interface 100 on rc1 and the following IP addresses for the tunnel: 192.88.99.1/32 and 2002:C058:6301::1/128.

We set the following routes:

```
root@group08-lg1:~# ip -6 route add 2002::/16 via fd00:470:525b:f801::1
     dev eth1

root@cgn-rj2# set routing-options rib inet6.0 static route 2002::/16
    next-hop FD00:470:525B:F805::2

root@cgn-rj1# set routing-options static route 192.88.99.0/24 next-hop
    10.8.100.2

root@group08-lg4:~# ip route add 192.88.99.0/24 via 10.8.4.1 dev eth1
```

5g

See the explanations in the comments we added:

```
ip tunnel add tun6to4 mode sit ttl 255 remote any local 10.8.4.2
# adds the 6to4 tunnel interface, sets it to mode sit (which is 6to4
    tunneling), set ttl to 255 hops, sets remote address to any and
    local exterior address to 10.8.4.2 (which is for outgoing traffic
    sent into the tunnel)

ip link set tun6to4 mtu 1280
# sets mtu for the tunnel to 1280 (which is important because a too
    large mtu might cause packet drops due to the encapsulation which
    will further increase the packet size

ip link set tun6to4 up
# brings the tunnel interface up

ip addr add 2002:0a08:0402:0::1/16 dev tun6to4
# adds the 6to4 address to the tunnel interface

ip addr add 2002:0a08:0402:1::1/64 dev eth2
# adds the interior 6to4 address to the eth2 interface to provide
    access to the tunnel to the internal network (in this case just lg3
    )

ip -6 route add ::/96 dev tun6to4 metric 256
# adds the ip route for ipv6 on the linux machine which routes ::/96
    addresses via interface tun6to4 (using the tunnel)

ip -6 route add ::/0 via ::192.88.99.1
# sets the IPv6 default route to go through the tunnel until the
    endpoint at rc1 IPv4 tunnel endpoint address
```

Here is the output from the given commands to show that we set up the tunnel correctly:

```
root@group08-lg4:~# ip tunnel show tun6to4
tun6to4: ipv6/ip remote any local 10.8.4.2 ttl 255 6rd-prefix 2002::/16

root@group08-lg4:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    group default
     link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
     inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
     inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
    state UP group default qlen 1000
     link/ether 00:16:3e:af:08:40 brd ff:ff:ff:ff:ff:ff
     inet 172.16.0.211/20 brd 172.16.15.255 scope global eth0
        valid_lft forever preferred_lft forever
     inet6 fe80::216:3eff:feaf:840/64 scope link
        valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
    state UP group default qlen 1000
     link/ether 00:16:3e:af:08:41 brd ff:ff:ff:ff:ff:ff
     inet 10.8.4.2/24 scope global eth1
```

```
       valid_lft forever preferred_lft forever
    inet6 2002:a08:402::1/16 scope global
       valid_lft forever preferred_lft forever
    inet6 fe80::216:3eff:feaf:841/64 scope link
       valid_lft forever preferred_lft forever
4: eth2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
   state UP group default qlen 1000
    link/ether 00:16:3e:af:08:42 brd ff:ff:ff:ff:ff:ff
    inet6 2002:a08:402:1::1/64 scope global
       valid_lft forever preferred_lft forever
    inet6 fe80::216:3eff:feaf:842/64 scope link
       valid_lft forever preferred_lft forever
5: sit0@NONE: <NOARP> mtu 1480 qdisc noop state DOWN group default
    link/sit 0.0.0.0 brd 0.0.0.0
6: tun6to4@NONE: <NOARP,UP,LOWER_UP> mtu 1280 qdisc noqueue state
   UNKNOWN group default
    link/sit 10.8.4.2 brd 0.0.0.0
    inet6 2002:a08:402::1/16 scope global
       valid_lft forever preferred_lft forever
    inet6 ::10.8.4.2/96 scope global
       valid_lft forever preferred_lft forever
```

5h

The tunnel endpoint is reachable from both loadgens:

```
root@group08-lg4:~# ping 192.88.99.1
PING 192.88.99.1 (192.88.99.1) 56(84) bytes of data.
64 bytes from 192.88.99.1: icmp_seq=1 ttl=254 time=1.32 ms

root@group08-lg1:~# ping 2002:C058:6301::1
PING 2002:C058:6301::1(2002:c058:6301::1) 56 data bytes
64 bytes from 2002:c058:6301::1: icmp_seq=1 ttl=63 time=1.22 ms
```

The requested pings of rc1, rj2 and lg1 from lg4:

```
root@group08-lg4:~# ping 10.8.100.2
PING 10.8.100.2 (10.8.100.2) 56(84) bytes of data.
64 bytes from 10.8.100.2: icmp_seq=1 ttl=254 time=1.31 ms

root@group08-lg4:~# ping 10.8.4.1
PING 10.8.4.1 (10.8.4.1) 56(84) bytes of data.
64 bytes from 10.8.4.1: icmp_seq=1 ttl=64 time=0.673 ms

root@group08-lg4:~# ping fd00:470:525b:f801::2
PING fd00:470:525b:f801::2(fd00:470:525b:f801::2) 56 data bytes
64 bytes from fd00:470:525b:f801::2: icmp_seq=1 ttl=62 time=1.98 ms
```

We include a traceroute, showing the way the packet takes (in the IPv6 network:

```
root@group08-lg4:~# traceroute fd00:470:525b:f801::2
traceroute to fd00:470:525b:f801::2 (fd00:470:525b:f801::2), 30 hops
    max, 80 byte packets
 1  2002:c058:6301::1 (2002:c058:6301::1)  1.443 ms  1.306 ms  1.303 ms
 2  fd00:470:525b:f805::1 (fd00:470:525b:f805::1)  1.254 ms  1.256 ms
     1.203 ms
 3  fd00:470:525b:f801::2 (fd00:470:525b:f801::2)  1.771 ms  1.721 ms
     1.670 ms
```

5i

We assign the IP addresses from the topology to lg3 and lg4 on eth2 and add a route to lg3:

```
root@group08-lg3:~# ip -6 route add ::/0 via 2002:a08:402:1::1 dev eth2
```

We enable forwarding on lg4 on all interfaces:

```
root@group08-lg4:~# echo 1 > /proc/sys/net/ipv6/conf/all/forwarding
```

The ping from lg3 to lg1 using the tunnel:

```
root@group08-lg3:~# ping fd00:470:525b:f801::2
PING fd00:470:525b:f801::2(fd00:470:525b:f801::2) 56 data bytes
64 bytes from fd00:470:525b:f801::2: icmp_seq=1 ttl=61 time=2.45 ms
```

And an additional traceroute showing the path:

```
root@group08-lg3:~# traceroute6 fd00:470:525b:f801::2
traceroute to fd00:470:525b:f801::2 (fd00:470:525b:f801::2), 30 hops
    max, 80 byte packets
 1  2002:a08:402:1::1 (2002:a08:402:1::1)  0.472 ms  0.414 ms  0.379 ms
 2  2002:c058:6301::1 (2002:c058:6301::1)  2.035 ms  2.056 ms  2.026 ms
 3  fd00:470:525b:f805::1 (fd00:470:525b:f805::1)  1.729 ms  1.666 ms
     1.635 ms
 4  fd00:470:525b:f801::2 (fd00:470:525b:f801::2)  2.077 ms  1.992 ms
     1.942 ms
```

Included Files

q02-config-rj2.txt, q03-config-sc1.txt, q05-config-rj1.txt, q05-config-sc1.txt, q02-config-sc1.txt, q05-config-rc1.txt, q05-config-rj2.txt, q05-config-sj1.txt