Middle East Technical University
CENG489 Project Report
Virtual Private Network

Canberk Morelli	Göksel Şimşek	Kadir Berkay Aydemir	Egemen Sarıkaya
2036101	2036598	1941780	2036143

Table of Contents

1. Problem Statement

Surfing the web is riskier than most of the people think. In case of regular connections, there are many vulnerabilities an adversary can exploit. When your IP is not hidden, hackers can easily target you in attacks or determine which country you are located in. Furthermore, without any encryption protection, hackers can snoop in on your connection and steal your data. One example of this is the WiFi attacks. We need Internet access on the go and sometimes we have to connect to a public WiFi network. Unfortunately, most public WiFi hotspots do not use any encryption and this leaves your data at the mercy of hackers who can exploit your connections and collect your data by WiFi eavesdropping.

Besides security risks, many countries apply Internet censorship which restricts people from accessing information. One example of this is the Wikipedia ban in Turkey on 29 April 2017 in which Turkish authorities blocked online access to all language editions of the online encyclopedia Wikipedia throughout Turkey.

To prevent these network security and censorship issues, we built a simple Virtual Private Network (VPN). A VPN creates a safe and encrypted connection over a less secure network such as the Internet and enables users to send and receive data safely. There are several different protocols that can be used to secure users and their data in a VPN. Some of the most notable ones are IP security (IPsec), Secure Sockets Layer (SSL), Point-To-Point Tunneling Protocol (PPTP) and Layer 2 Tunneling Protocol (L2TP). Our VPN is built upon the Secure Socket Layer (SSL) for Linux Ubuntu operating system. It provides a private network using tunneling so that users can connect to the internet securely and anonymously. Moreover, our VPN protects the data from theft by encryption provided by OpenSSL.

2. Related Work

2.1. A Comparative Research on SSL VPN and IPSec VPN[1]

In this conference paper, two kinds of VPN, which are IPSec and SSL, are studied in detail. IPSec and SSL protocols are compared in means of the scope of application, operation complexity, security strategy and scalability. Advantages and disadvantages of both protocols are summarized and shortcomings such as security risks, management costs, and other

[1] H. Mao, L. Zhu and H. Qin, "A Comparative Research on SSL VPN and IPSec VPN", *2012 8th International Conference on Wireless Communications, Networking and Mobile Computing*, Shanghai, China, 2012, pp. 1-4. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6478270&isnumber=6478258

aspects are analyzed and compared. Lastly, based on characteristics, respective advantages and disadvantages of two VPN, how to select VPN technologies is explained.

2.2. Implementation of Improved VPN Based on SSL[2]

An improved VPN system based on SSL protocol is discussed to overwhelm the requirement of installing client software. The concept and technique of VPN (tunnel technique, encryption, validation) are discussed. SSL protocol architecture and the working process of the handshake layer and the record layer are analyzed in detail. Lastly, an improved VPN based on SSL protocol is designed. In this VPN system, a SSL proxy is added behind the corporation firewall to improve the security. Tasks of the improved VPN system such as validation, encrypting and decoding are explained.

2.3.The Research and Implementation of the VPN Gateway Based on SSL[3]

This article analyzes the principle of the VPN technology and the SSL protocol. Principle of VPN is elucidated by explaining concept of VPN and technologies of tunneling, encryption, decryption, key management, authentication. Architecture of SSL protocol is analyzed and concepts such as SSL connection and session are mentioned. A VPN gateway based on the SSL protocol is proposed. Lastly, the design, steps of implementation and application of this solution is given.

3. Approach

We followed the descriptions and tasks of Virtual Private Network (VPN) Lab[4] of Department of Electrical Engineering and Computer Science, Syracuse University as a guideline. We did not get out of the scope of this lab project but changed or skipped some of the tasks according to our design.

---

[2] Y. Kuihe and C. Xin, "Implementation of Improved VPN Based on SSL", *2007 8th International Conference on Electronic Measurement and Instruments*, Xi'an, 2007, pp. 2-15-2-19.
http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4350641&isnumber=4350397

[3] C. Fei, W. Kehe, C. Wei and Z. Qianyuan, "The Research and Implementation of the VPN Gateway Based on SSL", *2013 International Conference on Computational and Information Sciences*, Shiyang, 2013, pp. 1376-1379.
http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6643282&isnumber=6642914

[4] http://www.cis.syr.edu/~wedu/seed/Labs/VPN/

In a broad view, our goal was to connect two private networks through a secure tunnel so that an adversary would not be able to locate the networks or eavesdrop the communication between them.

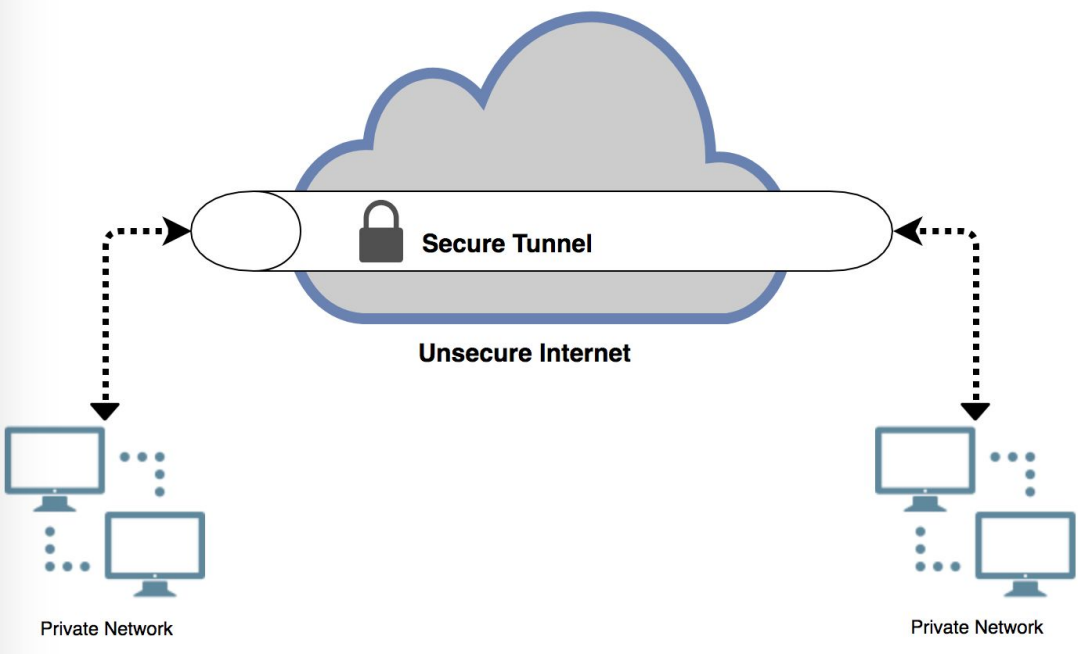

Figure 1 - General illustration of our goal

To provide a better description of our approach, the steps we have taken are included in chronological order in this section.

3.1. Host-to-Host Tunnel

Reading David Brini's TUN/TAP interface tutorial[5], we determined how to create and use TUN/TAP interface for tunneling. Using the modified version of Brini's simpletun tunneling program[6], which is provided by Syracuse University's VPN Lab, we were able to create a host-to-host tunnel. We ran the simpletun program as a server on one computer and as a client on another computer. In each computer, we created a TUN interface and assigned an IP to it. After establishing this tunnel, we set up the routing path on both machines to direct the intended outgoing traffic through the tunnel. Doing so, we were able to ping the one host from the other one through the tunnel. Note that Brini's tunneling program uses TCP protocol. However, we did not want to end up paying the TCP overhead twice (host-to-gateway and gateway-to-gateway) in our VPN so we modified the simpletun program and changed the TCP tunnel into a UDP tunnel.

---

[5] http://backreference.org/2010/03/26/tuntap-interface-tutorial/
(if the link is broken:
https://web.archive.org/web/20180114110608/http://backreference.org/2010/03/26/tuntap-interface-tutorial/)

[6] http://www.cis.syr.edu/~wedu/seed/Labs/VPN/files/simpletun.c

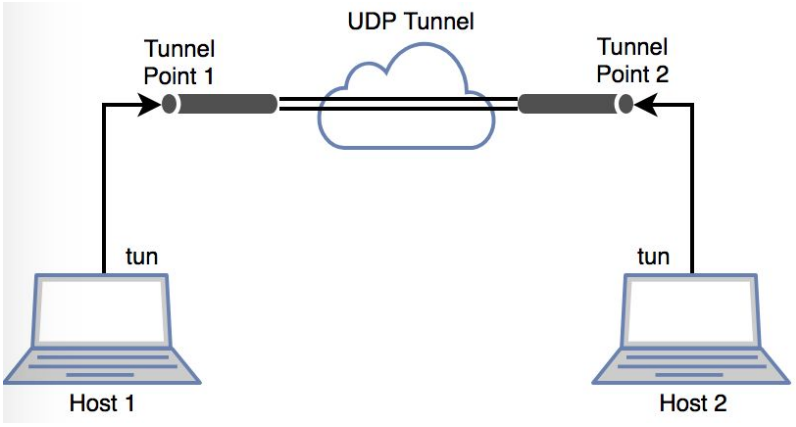

Figure 2 - Host-to-Host Tunnel Illustration

3.2. Host-to-Gateway Tunnel

After setting up the tunnel on the two hosts, we want to create a gateway. In this scenario, we set the gateway on one computer's virtual machine and the other computer acted as a host. In order to be able to receive packets sent from host in gateway, we had to forward packets from actual computer to virtual machine. To achieve this, we created a rule in port forwarding settings settings of the virtual machine.

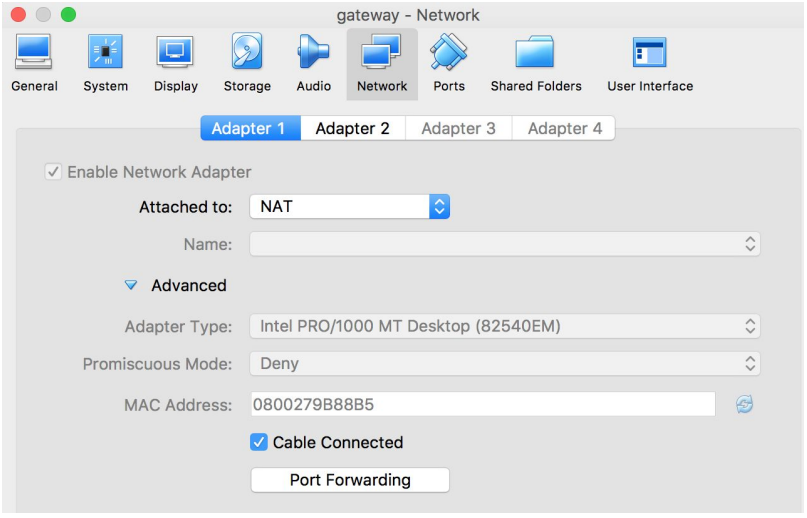

Figure 3 - Gateway virtual machine network settings

Name	Protocol	Host IP	Host Port	Guest IP	Guest Port
Rule 1	UDP	10.70.182.229	55555	10.0.4.15	55555

Figure 4 - Rule for forwarding packets to the gateway

After setting port forwarding, we created the UDP tunnel between the host computer and the gateway. This task was not much different from host-to-host tunnel, only difference is the usage of the virtual machine and port forwarding. This task's goal was to connect the host to the gateway so that the host will be able to connect gateway's private network through the tunnel.

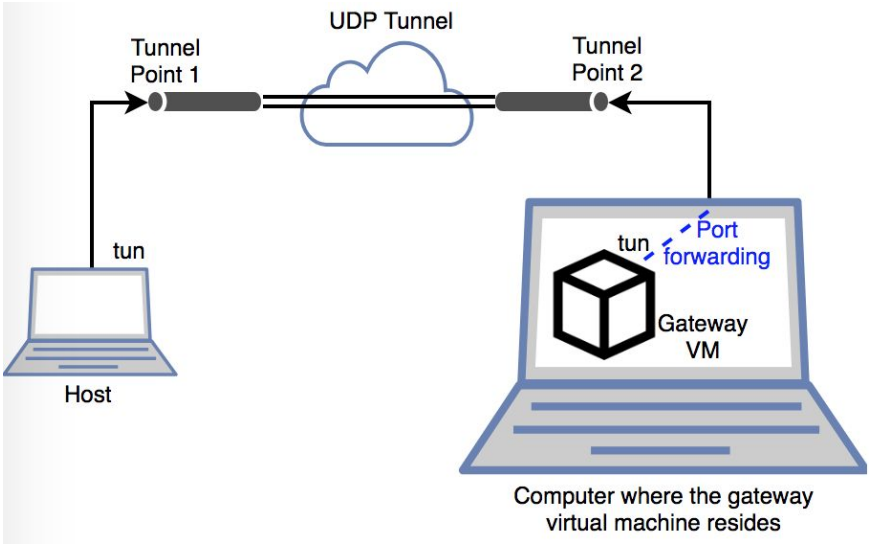

Figure 5 - Host-to-Gateway Tunnel Illustration

3.3. Gateway-to-Gateway Tunnel

This time, we setup a tunnel between two gateways each having their own private networks. Gateways reside in two different computer's virtual machines and their private network also reside on virtual machines in the corresponding computers.

A gateway has a network interface for its own private network which is set through the network settings of the virtual machine. The gateway receives packets from its private network through this interface.

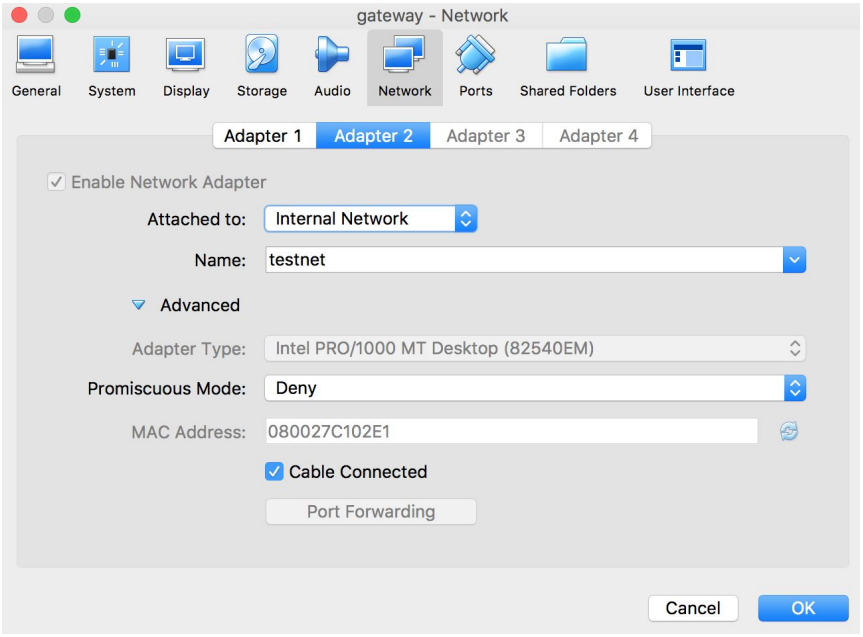

Figure 6 - Gateway Virtual Machine Network Settings

Its private network hosts also configured to be connected to this internal network.

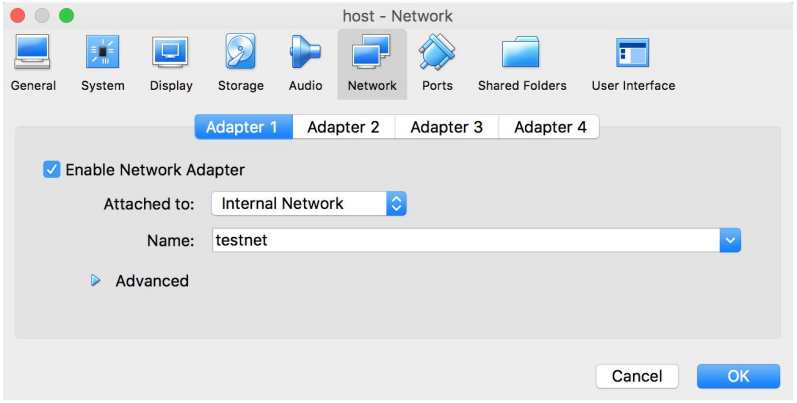

Figure 7 - Host Virtual Machine Network Settings

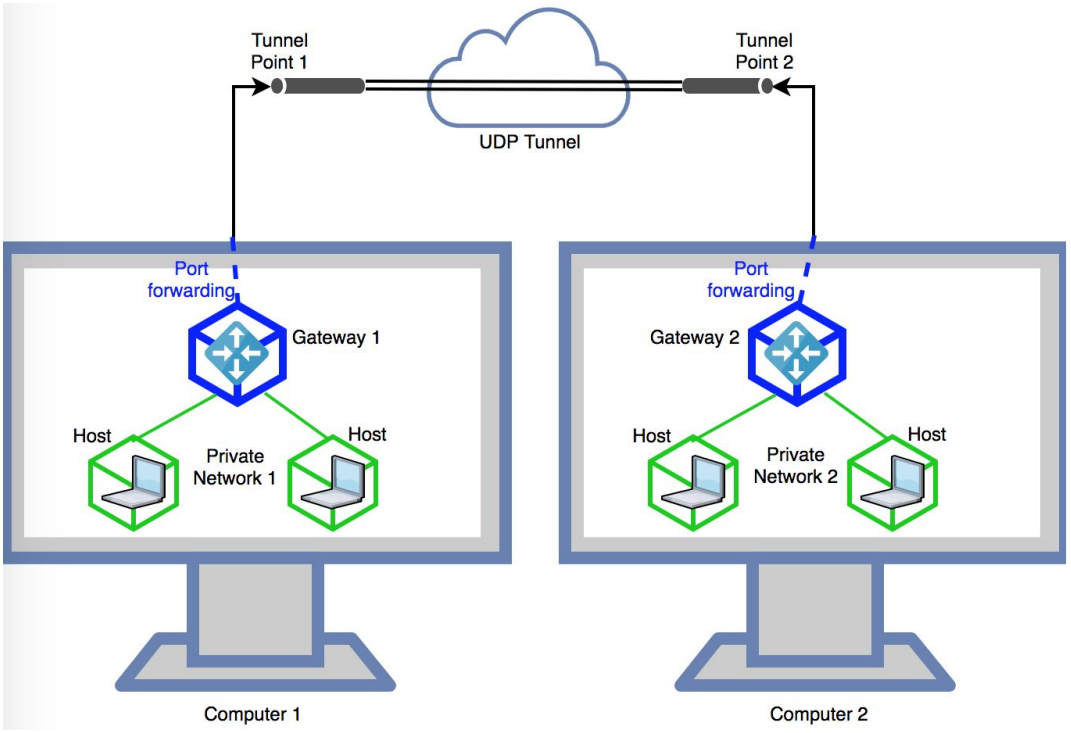

Figure 8 - Two gateways residing in different computers connected through the UDP tunnel

We established the tunnel between two gateways of different private networks so that any presentations and all the rest will be shifted accordingly. Please remember you should not come in/leave during the presentations, so if you have your presentation is in the first hour, be there before 3:40 pm and if your presentation is in the second hour, be therhost from one private network can communicate with the hosts on the other private network using the tunnel. When a host in the private network sends a packet to another host, this packet follows the steps listed below. Note that details of the implementation and routing are given in the implementation section.

1. Packet is routed to the internal network interface of the gateway.
2. Packet received at the internal network interface is routed to the tun0 interface of the gateway.
3. Packet is read from the tun0 interface in the server program and sent to the other gateway using the tunnel.
4. Packet is read from the other end of the tunnel by the second gateway. server program writes this packet to its tun1 interface.
5. Packet is routed to its destination from tun1 in the private network through the gateway's internal network interface.

3.4. Creating the Virtual Private Network

At this point, we have set the network tunnel between two gateways so that private networks can communicate with each other. This task's goal was to secure this tunnel so that we can create a VPN. To secure the tunnel we encrypted packets received by the gateway from its private network are encrypted and sent to other gateway. Other gateway decrypts received packets and transmits them to their destination in its own private network. By encrypting packets, we achieved confidentiality. However, this was not enough to achieve integrity. For integrity, we used Message Authentication Code (MAC) method.

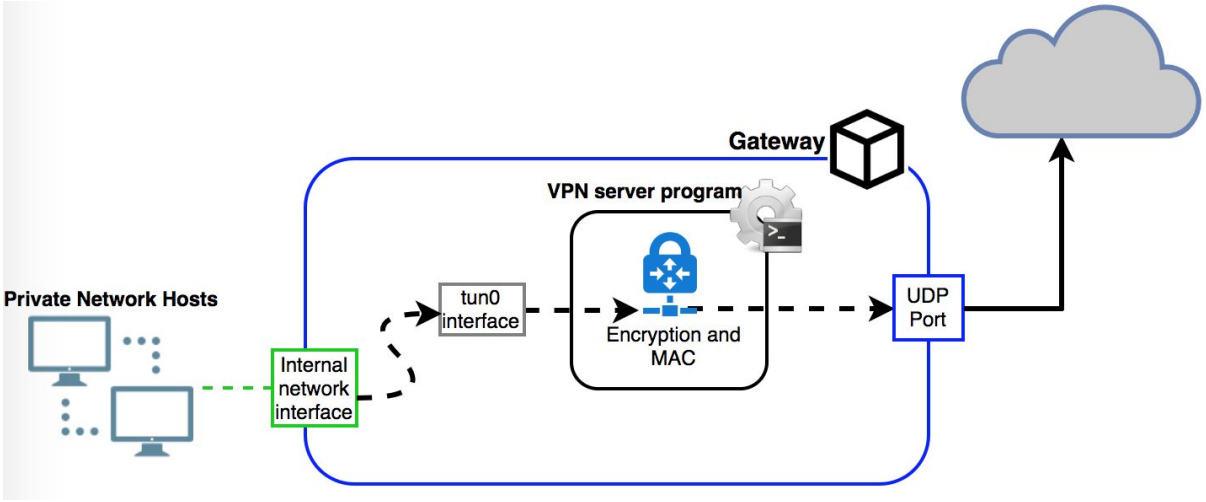

Figure 9 - Gateway sending packets

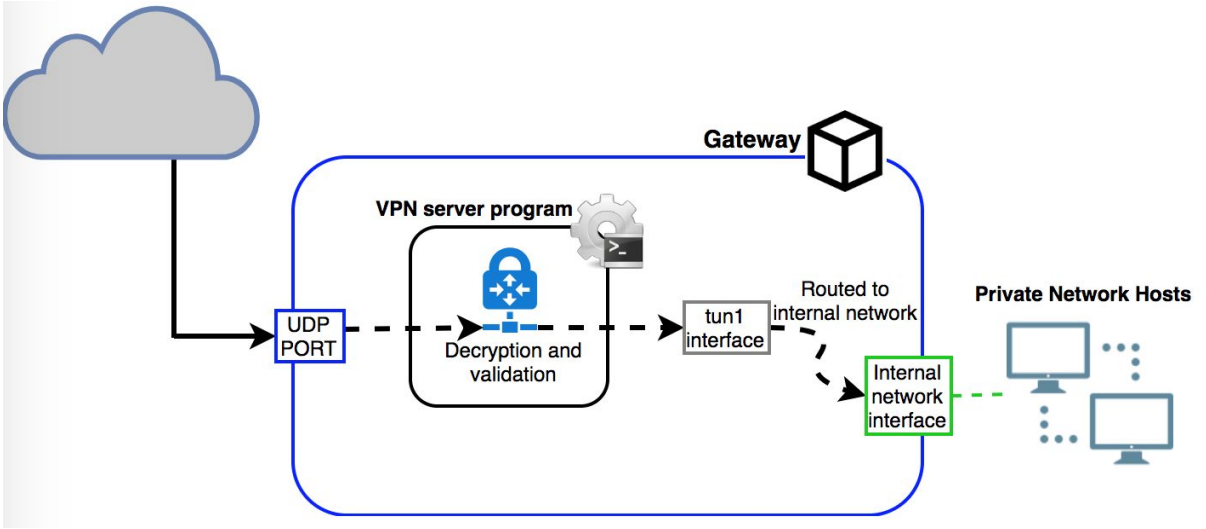

Figure 10 - Gateway receiving packets

Both encryption and MAC need a secret key. Although the keys can be different, we used the same secret key for simplicity. This secret key is agreed between gateways by the following procedure (details are provided in implementation section):

1. Start first gateway's VPN server. This server waits for SSL connections.
2. Start the second gateway's VPN server and provide the first gateway's IP as a command line argument.
3. Second gateway initiates an SSL connection with the first gateway.
4. Upon setting the connection, the first gateway generates a random secret key and sends it to the second gateway through SSL connection.
5. Second gateway receives the secret key and the key exchange is completed.

After completing the key exchange, gateways can exchange packets between them safely with encryption and MAC. Illustration of wrapping packets using encryption and HMAC can be seen below in the Figure 11.

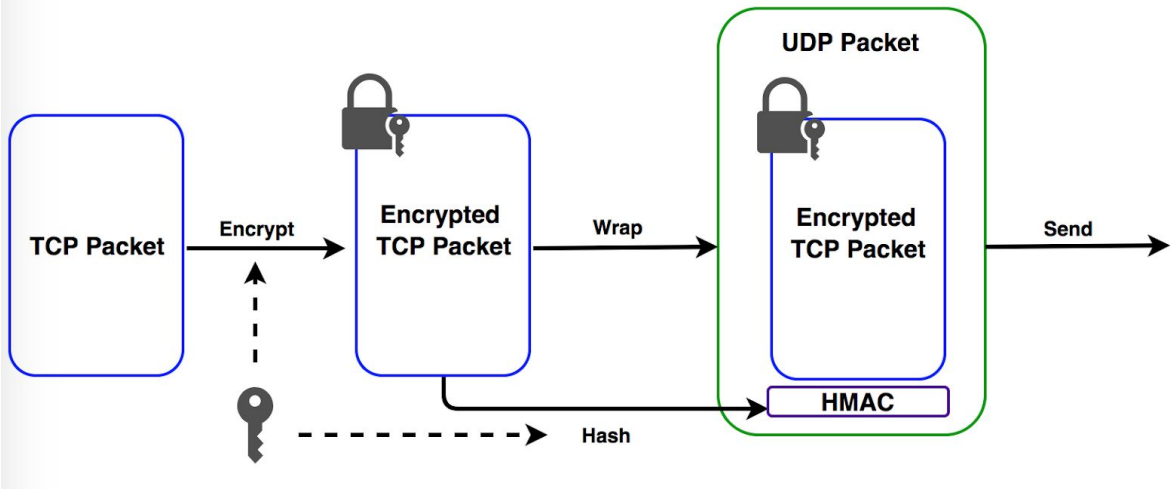

Figure 11 - Encryption and HMAC

4. Implementation

We implemented everything in C since provides the network libraries required for socket programming and the functionality to interact with TUN/TAP interfaces. We took the simpletun program as a base while implementing our VPN, especially TUN interface related functionalities.

4.1 UDP Tunnel

We noticed that in the simpletun program, a TCP socket is used. However, in our implementation, we decided to use a UDP tunnel. We made this decision because TCP uses adaptive timeouts to ensure reliability. These timeouts change dynamically with every received packet. When a packet timeouts, the following timeout is increased exponentially. This timeout policy works fine in the Internet but the optimization it provides breaks when stacking one TCP connection on top of another. We are wrapping the packets sent from the clients while sending them through the tunnel and if we also use TCP in the tunnel, there will be a nested timeout policy in which layers are not aware of each other. The upper layer retransmissions would be unnecessary since the carrier guarantees delivery, but the upper layer TCP cannot know this because it assumes an unreliable carrier. Thus, TCP's reliability backfires in this case and a UDP tunnel is much more appropriate.

4.2 TUN/TAP

We open the TUN interface almost same as the simpletun program:

```
int tun_alloc(char *dev, int flags) {

    struct ifreq ifr;
    int fd, err;

    if ((fd = open("/dev/net/tun", O_RDWR)) < 0) {
        perror("Opening /dev/net/tun");
        return fd;
    }

    memset(&ifr, 0, sizeof(ifr));

    ifr.ifr_flags = flags;

    if (*dev) {
        strncpy(ifr.ifr_name, dev, IFNAMSIZ);
    }

    if ((err = ioctl(fd, TUNSETIFF, (void *) &ifr)) < 0) {
        perror("ioctl(TUNSETIFF)");
        close(fd);
        return err;
    }

    if (ioctl(fd, TUNSETPERSIST, 1) < 0) {
        perror("enabling TUNSETPERSIST");
        exit(1);
    }

    strcpy(dev, ifr.ifr_name);

    return fd;
}
```

The difference is while we are opening TUN interface, we set it to be persistent so that we do not need to create a new TUN interface with `ip addr` command and activate it everytime we restart the client or the server. Achieving a persistent TUN interface is done by the `ioctl(fd, TUNSETPERSIST, 1)` call.

Using the file pointer id returned from the `tun_alloc` function, we read from and write to corresponding TUN interface using `unistd.h` header's `read` and `write` functions.

4.3 SSL Key Exchange

While founding the connection between gateways, one gateways waits for another gateway to connect it. This connection request and the key exchange operations are done by SSL connection based on TCP. We assumed that the gateway which acts as the client already has the public certificate of the other gateway which acts as a server. After handshaking using certificates, server sends an initial vector and a key which will be used for encryption and message authentication. Client takes the key and initial vector and sends it back to the server in order to finalize the connection and say that "OK, I'm done". This key exchange mechanism is safe thanks to OpenSSL. After key change is done both gateways close the SSL connection which uses TCP and starts the actual VPN connection which uses UDP.

4.4 SSL Encryption & Message Authentication

AES is used for encryption while servers are communicating. The key which was exchanged is 256 bit long. CBC mode is used while encrypting message, so the initial vector is also exchanged between gateways. For authentication HMAC is used based on SHA256 algorithm. For the sake of simplicity we used the same 256 bit key for AES and SHA256. After encrypting the original message using AES, whole cipher text is wrapped with its HMAC equivalent 32 byte hash value as indicated in Figure 11. After sending from one gateway to another, receiving gateway first separate HMAC part and cipher part. Then using HMAC it calculates hash value from cipher. If that hash value is equal to the received hash value, cipher text is decrypted otherwise message is dropped. By this way message authentication and message integrity is provided. For encryption, decryption and HMAC parts OpenSSL library functions are used.

4.5 Routing

A host has two interfaces which are internal interface and external interface but external one is not used for VPN, it is only used for accessing Internet for other purposes. Therefore external interface has nothing to do with the VPN implementation and disabling it does not affect the VPN. On the other hand, a gateway has two interfaces. The first interface of the gateway is an internal interface in order to communicate with the other hosts among the private network that gateway is in. The second one is a NAT interface that connects the

gateway to the other networks. In gateway, external interface is where gateway-to-gateway communication established.

In a host's routing table, following rules must be satisfied:

- If packet’s destination ip is another private network, send it to the gateway via internal interface.
- If packet’s destination ip is within the private network, send it directly via internal interface.

Also there are some routing entries that comes as a default such as local loopback entry and default gateway entry.

As an example, we created two private networks and configure their routing tables. Their attributes are shown in the Figure x, x, x and x. Figures cover one of the private networks because other network’s routing configurations are similar.

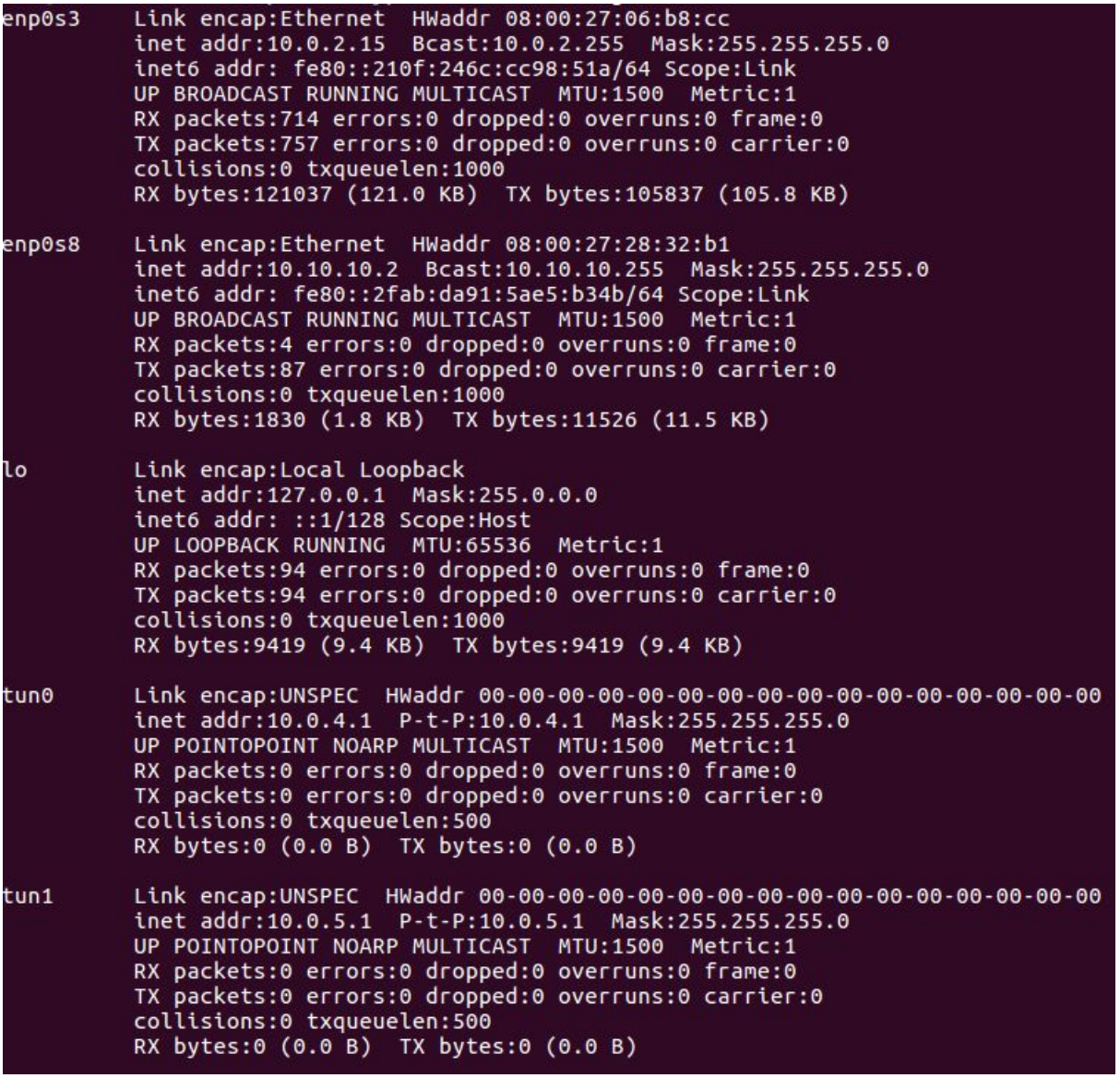

Figure 12 - Interfaces of gateway VM

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         10.0.2.2        0.0.0.0         UG    100    0        0 enp0s3
10.0.2.0        *               255.255.255.0   U     100    0        0 enp0s3
10.0.4.0        *               255.255.255.0   U     0      0        0 tun0
10.0.5.0        *               255.255.255.0   U     0      0        0 tun1
10.10.9.0       10.0.4.1        255.255.255.0   UG    0      0        0 tun0
10.10.10.0      *               255.255.255.0   U     100    0        0 enp0s8
link-local      *               255.255.0.0     U     1000   0        0 enp0s8
```

Figure 13 - Routing table of gateway VM

```
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:7e:57:d4
          inet addr:10.10.10.3  Bcast:10.10.10.255  Mask:255.255.255.0
          inet6 addr: fe80::a615:bef2:4fee:ff8f/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:8 errors:0 dropped:0 overruns:0 frame:0
          TX packets:136 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2894 (2.8 KB)  TX bytes:19410 (19.4 KB)

enp0s8    Link encap:Ethernet  HWaddr 08:00:27:d4:9b:4d
          inet addr:10.0.3.15  Bcast:10.0.3.255  Mask:255.255.255.0
          inet6 addr: fe80::930c:cba0:9cf8:6de8/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:148 errors:0 dropped:0 overruns:0 frame:0
          TX packets:249 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:56162 (56.1 KB)  TX bytes:28419 (28.4 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:159 errors:0 dropped:0 overruns:0 frame:0
          TX packets:159 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:14176 (14.1 KB)  TX bytes:14176 (14.1 KB)
```

Figure 14 - Interfaces of host VM

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         10.0.3.2        0.0.0.0         UG    100    0        0 enp0s8
10.0.3.0        *               255.255.255.0   U     100    0        0 enp0s8
10.10.9.0       10.10.10.2      255.255.255.0   UG    0      0        0 enp0s3
10.10.10.0      *               255.255.255.0   U     100    0        0 enp0s3
link-local      *               255.255.0.0     U     1000   0        0 enp0s3
```

Figure 15 - Routing table of host VM

5. Evaluation

In our evaluation, we examined the performance of our VPN by comparing the RTT values with and without encryption. To do so, in our experiments, we have closed and opened the encryption related functions. In our experiments we have sent 20 packets using the command `ping -c 20 10.10.10.3` which supports min/avg/max/mdev values of RTT values that are measured during the evaluation.

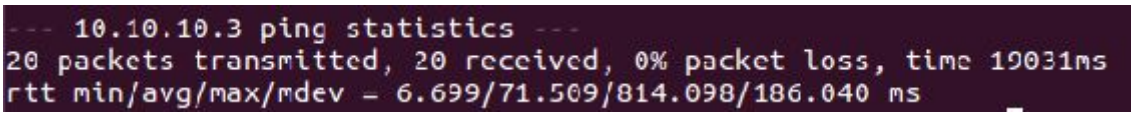

Figure 16 - RTT time with encryption

Comparing the average values with and without encryption shows us that the encryption algorithm that we have used does not create that much of overhead. Which may be because of the fact that we have used small packets during the experiments.

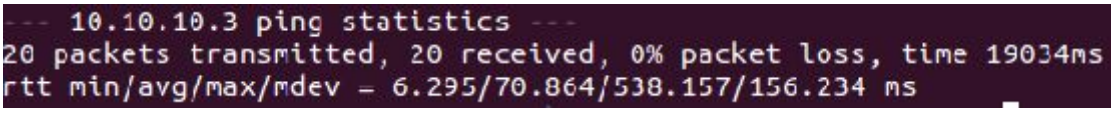

Figure 17 - RTT time without encryption

6. Conclusion

We implemented a simple Virtual Private Network which consists of a gateway-to-gateway tunnel that is secured via SSL encryption. Using this VPN, communication between the two private networks residing in each side of the gateway-to-gateway tunnel is secure. Real IP addresses of clients in private networks are hidden from the Internet and this provides anonymity to them. Also, since the all traffic through the tunnel is encrypted, communication data is protected from theft. Furthermore, by setting the gateway in a computer that is in a different country, one can bypass a censorship.

6.1 Future Work

Many improvements can be made on our VPN project. The ones that can be very beneficial are listed in this section.

Firstly, our VPN lacks of an advanced authentication system. Even it uses HMAC and certificates, there is not any functionality for classifying clients according their names or ids. Thus, a better authentication system can be implemented. For instance, gateways can keep the record of valid users by storing their names, ids and passwords in a database or simply in a file. Using these records, gateways can identify clients .Furthermore, an access control system can be implemented based on these records.

Secondly, there is not any functionality for multiple clients communicating through the tunnel at the same time. Currently, only one pair of clients (one client from the first private network and one from the other) can communicate bidirectionally with each other through the tunnel. This problem can be solved by supporting multiple VPN tunnels. With multiple VPN tunnels, VPN server can allow more than one clients to connect to it simultaneously. Each client would have its own VPN tunnel with the server and the session keys used in different tunnels would be different.

Lastly, there is no dynamic reconfiguration which allows disconnecting & reconnecting of clients to the server or changing the session key. A command interpretation functionality at the client side can be implemented to allow the client to change the session key or break the current VPN tunnel. By informing the server, corresponding resources can be released and new tunnels can be created.

7. References

References of related work are added as footnotes in corresponding pages.

Other sources we used are:

1. Ballard, (2012, June 28). *Secure programming with the OpenSSL API*[PDF]. IBM developerWorks. Retrieved from https://www.ibm.com/developerworks/linux/library/l-openssl/l-openssl-pdf.pdf

2. Du, W. (n.d.). *Virtual Private Network (VPN) Lab*[PDF]. Syracuse University. Retrieved from http://www.cis.syr.edu/~wedu/seed/Labs/VPN/VPN.pdf

3. Titz, O. (2001, April 23). Why TCP Over TCP Is A Bad Idea. Retrieved from http://sites.inka.de/bigred/devel/tcp-tcp.html