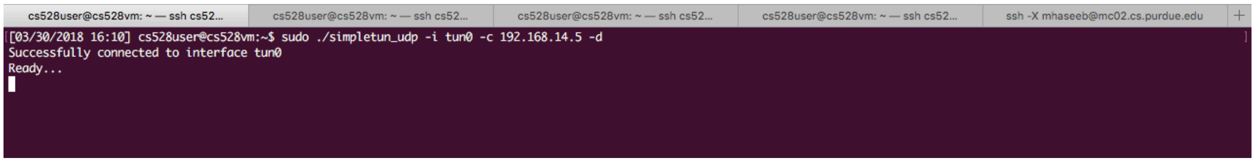
CS528 Lab 3 Report

Mohammad Haseeb
mhaseeb@purdue.edu

Task 1: Create a Host-to-Host Tunnel using TUN/TAP

- Host 1 has IP: 192.168.14.4
- Host 2 has IP: 192.168.14.5

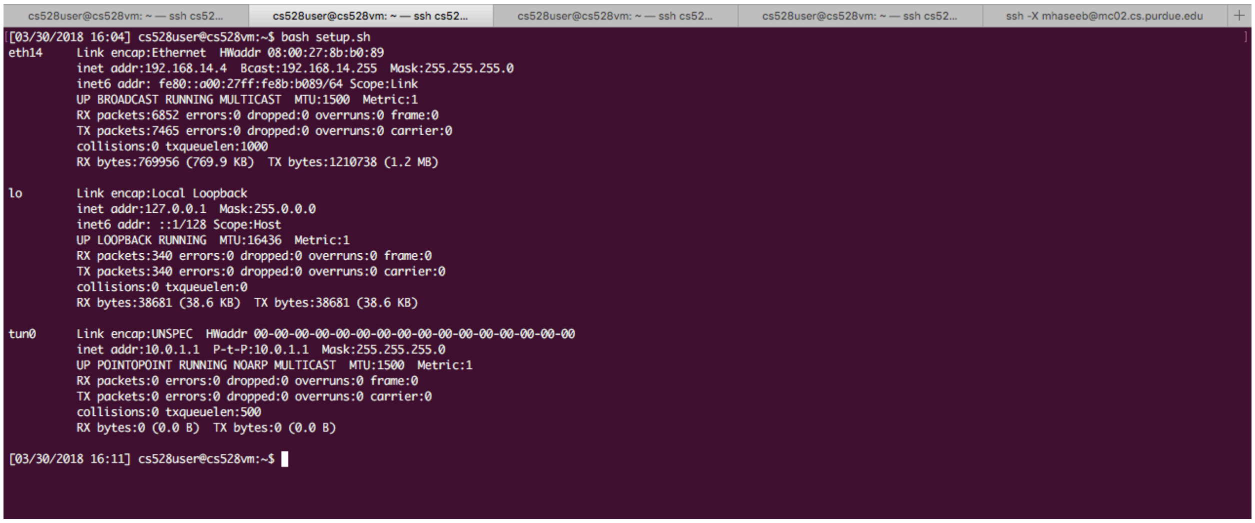
I run both the *simpletun* codes as clients so that I can pass the IP of the other host via the command line and not have to hard code it. Under the hood, the *simpletun_udp.c* code acts as both a server and a client since UDP is connectionless. The following screenshots show how we can *ssh* into the other host using the *simpleton_udp.c* program. They show the tunnel setup on both the hosts and how I can *ssh* into the other using the tunnel point.


cs528user@cs528vm: ~ — ssh cs52...
cs528user@cs528vm: ~ — ssh cs52...
cs528user@cs528vm: ~ — ssh cs52...
cs528user@cs528vm: ~ — ssh cs52...
ssh -X mhaseeb@mc02.cs.purdue.edu

```
[03/30/2018 16:10] cs528user@cs528vm:~$ sudo ./simpletun_udp -i tun0 -c 192.168.14.5 -d
Successfully connected to interface tun0
Ready...
```

Figure 1

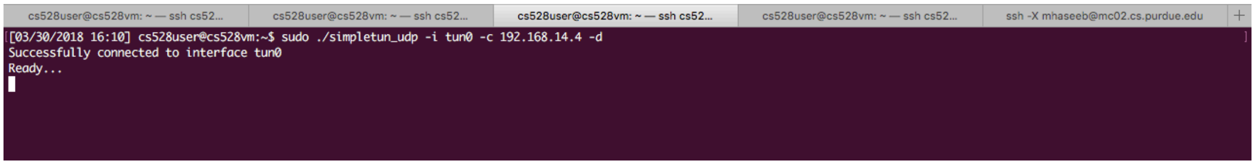
- Figure 1 shows the tunnel and the virtual network interface setup and ready on one of the hosts. Here the IP of the other VM is 192.169.14.5


cs528user@cs528vm: ~ — ssh cs52...
cs528user@cs528vm: ~ — ssh cs52...
cs528user@cs528vm: ~ — ssh cs52...
cs528user@cs528vm: ~ — ssh cs52...
ssh -X mhaseeb@mc02.cs.purdue.edu

```
[03/30/2018 16:04] cs528user@cs528vm:~$ bash setup.sh
eth14     Link encap:Ethernet  HWaddr 08:00:27:8b:b0:89
          inet addr:192.168.14.4  Bcast:192.168.14.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe8b:b089/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:6852 errors:0 dropped:0 overruns:0 frame:0
          TX packets:7465 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:769956 (769.9 KB)  TX bytes:1210738 (1.2 MB)

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:16436  Metric:1
          RX packets:340 errors:0 dropped:0 overruns:0 frame:0
          TX packets:340 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:38681 (38.6 KB)  TX bytes:38681 (38.6 KB)

tun0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          inet addr:10.0.1.1  P-t-P:10.0.1.1  Mask:255.255.255.0
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

[03/30/2018 16:11] cs528user@cs528vm:~$
```

Figure 2

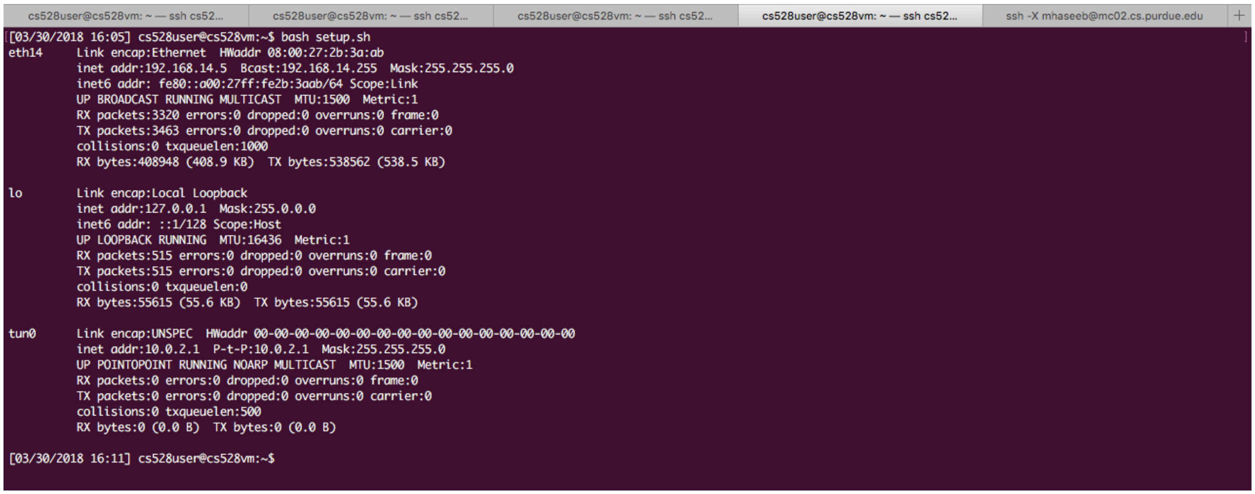
- Figure 2 shows the *ifconfig* of this host. As we can see the *tun0* virtual network interface has gotten an IP address as the one we mentioned in the *setup.sh* bash file and the one mentioned in the lab handout.


cs528user@cs528vm: ~ — ssh cs52...
cs528user@cs528vm: ~ — ssh cs52...
cs528user@cs528vm: ~ — ssh cs52...
cs528user@cs528vm: ~ — ssh cs52...
ssh -X mhaseeb@mc02.cs.purdue.edu
+

```
[03/30/2018 16:10] cs528user@cs528vm:~$ sudo ./simpletun_udp -i tun0 -c 192.168.14.4 -d
Successfully connected to interface tun0
Ready...
█
```

Figure 3

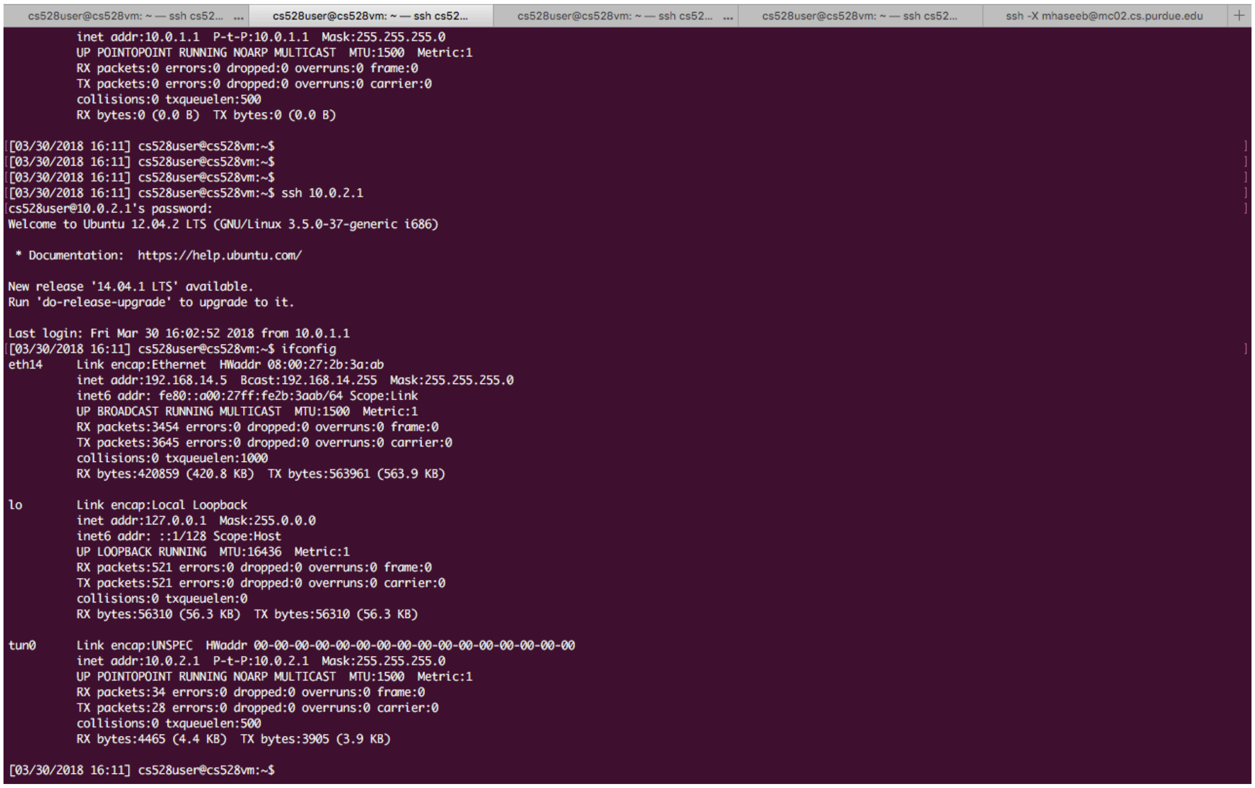
- Figure 3 shows the tunnel set up and ready on the other host.


cs528user@cs528vm: ~ — ssh cs52...
cs528user@cs528vm: ~ — ssh cs52...
cs528user@cs528vm: ~ — ssh cs52...
cs528user@cs528vm: ~ — ssh cs52...
ssh -X mhaseeb@mc02.cs.purdue.edu
+

```
[03/30/2018 16:05] cs528user@cs528vm:~$ bash setup.sh
eth14     Link encap:Ethernet  HWaddr 08:00:27:2b:3a:ab
          inet addr:192.168.14.5  Bcast:192.168.14.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe2b:3aab/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:3320 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3463 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:408948 (408.9 KB)  TX bytes:538562 (538.5 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:16436  Metric:1
          RX packets:515 errors:0 dropped:0 overruns:0 frame:0
          TX packets:515 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:55615 (55.6 KB)  TX bytes:55615 (55.6 KB)

tun0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          inet addr:10.0.2.1  P-t-P:10.0.2.1  Mask:255.255.255.0
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

[03/30/2018 16:11] cs528user@cs528vm:~$
```

Figure 4

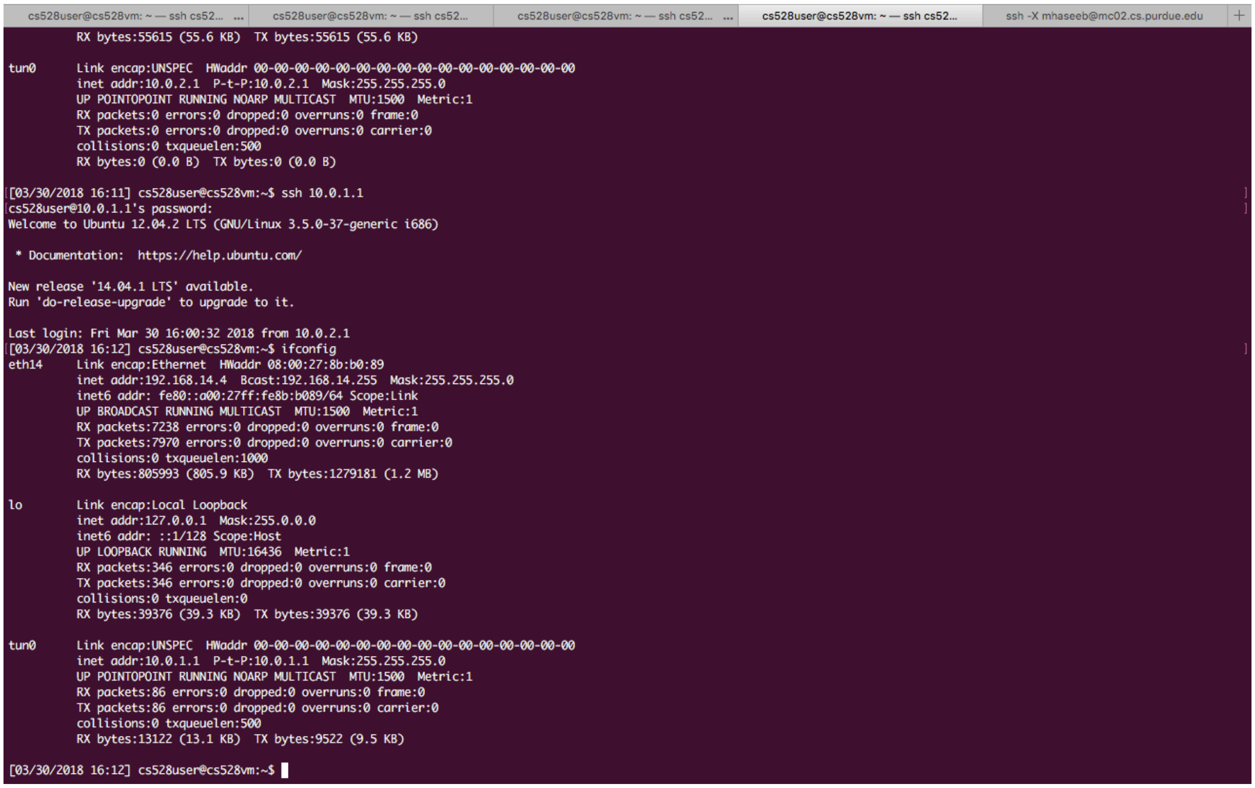
- Figure 4 shows the *ifconfig* of the newly setup machine and the corresponding information about the *tun0* virtual network interface.


cs528user@cs528vm: ~ — ssh cs52...
cs528user@cs528vm: ~ — ssh cs52...
cs528user@cs528vm: ~ — ssh cs52...
cs528user@cs528vm: ~ — ssh cs52...
ssh -X mhaseeb@mc02.cs.purdue.edu

```
          inet addr:10.0.1.1  P-t-P:10.0.1.1  Mask:255.255.255.0
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

[03/30/2018 16:11] cs528user@cs528vm:~$
[03/30/2018 16:11] cs528user@cs528vm:~$
[03/30/2018 16:11] cs528user@cs528vm:~$
[03/30/2018 16:11] cs528user@cs528vm:~$ ssh 10.0.2.1
cs528user@10.0.2.1's password:
Welcome to Ubuntu 12.04.2 LTS (GNU/Linux 3.5.0-37-generic i686)

 * Documentation:  https://help.ubuntu.com/

New release '14.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Fri Mar 30 16:02:52 2018 from 10.0.1.1
[03/30/2018 16:11] cs528user@cs528vm:~$ ifconfig
eth14     Link encap:Ethernet  HWaddr 08:00:27:2b:3a:ab
          inet addr:192.168.14.5  Bcast:192.168.14.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe2b:3aab/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:3454 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3645 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:420859 (420.8 KB)  TX bytes:563961 (563.9 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:16436  Metric:1
          RX packets:521 errors:0 dropped:0 overruns:0 frame:0
          TX packets:521 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:56310 (56.3 KB)  TX bytes:56310 (56.3 KB)

tun0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          inet addr:10.0.2.1  P-t-P:10.0.2.1  Mask:255.255.255.0
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:34 errors:0 dropped:0 overruns:0 frame:0
          TX packets:28 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:4465 (4.4 KB)  TX bytes:3905 (3.9 KB)

[03/30/2018 16:11] cs528user@cs528vm:~$
```

Figure 5

- Figure 5 shows that we can *ssh* into the second host from the first host. This is possible due to the existence of the tunnel we just set up.



cs528user@cs528vm: ~ — ssh cs52...
cs528user@cs528vm: ~ — ssh cs52...
cs528user@cs528vm: ~ — ssh cs52...
cs528user@cs528vm: ~ — ssh cs52...
ssh -X mhaseeb@mc02.cs.purdue.edu

```
          RX bytes:55615 (55.6 KB)  TX bytes:55615 (55.6 KB)

tun0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          inet addr:10.0.2.1  P-t-P:10.0.2.1  Mask:255.255.255.0
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

[03/30/2018 16:11] cs528user@cs528vm:~$ ssh 10.0.1.1
cs528user@10.0.1.1's password:
Welcome to Ubuntu 12.04.2 LTS (GNU/Linux 3.5.0-37-generic i686)

 * Documentation:  https://help.ubuntu.com/

New release '14.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Fri Mar 30 16:00:32 2018 from 10.0.2.1
[03/30/2018 16:12] cs528user@cs528vm:~$ ifconfig
eth14     Link encap:Ethernet  HWaddr 08:00:27:8b:b0:89
          inet addr:192.168.14.4  Bcast:192.168.14.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe8b:b089/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:7238 errors:0 dropped:0 overruns:0 frame:0
          TX packets:7970 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:805993 (805.9 KB)  TX bytes:1279181 (1.2 MB)

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:16436  Metric:1
          RX packets:346 errors:0 dropped:0 overruns:0 frame:0
          TX packets:346 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:39376 (39.3 KB)  TX bytes:39376 (39.3 KB)

tun0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          inet addr:10.0.1.1  P-t-P:10.0.1.1  Mask:255.255.255.0
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:86 errors:0 dropped:0 overruns:0 frame:0
          TX packets:86 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:13122 (13.1 KB)  TX bytes:9522 (9.5 KB)

[03/30/2018 16:12] cs528user@cs528vm:~$ 
```

Figure 6

- Figure 6 shows that we can *ssh* into the first host from the second host. This is possible due to the existence of the tunnel we just set up.

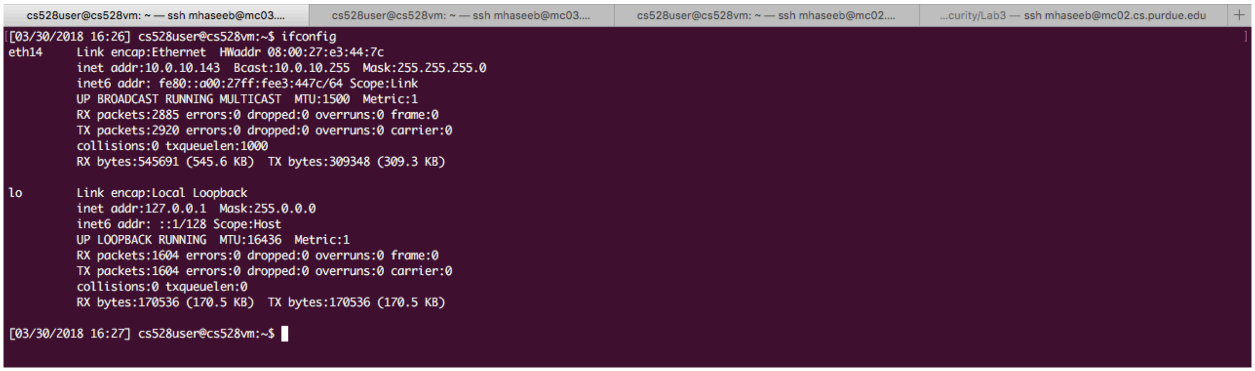
Q. Why it is better to use UDP in the tunnel, instead of TCP?

Notice that, depending on the application, whatever traffic a client sends into the tunnel is already sent over a TCP connection. As an example, consider *ssh*. When VM1 *ssh*'s into VM2, the *ssh* protocol creates a TCP connection 'under-the-hood' to ensure the reliable delivery of data between the hosts. The VPN tunnel's job is to only enable this *ssh* traffic to be transferred to the respective host. Therefore, it would not make sense for the tunnel to work on a TCP socket as that would just add extra overhead in terms of TCP headers and be a performance hit. The reliability is already provided by the application using this tunnel and so we do not need to do it again.

Task 2: Create a Private Network using a Gateway

In the following screenshots I first show the network configurations *ifconfig* of each of the client VMs. I then show that I was able to *ping* a remote internet host (google.com) from these VMs but they were not able to *ping* each other. This was because there was no Gateway-to-Gateway tunnel setup as of now.

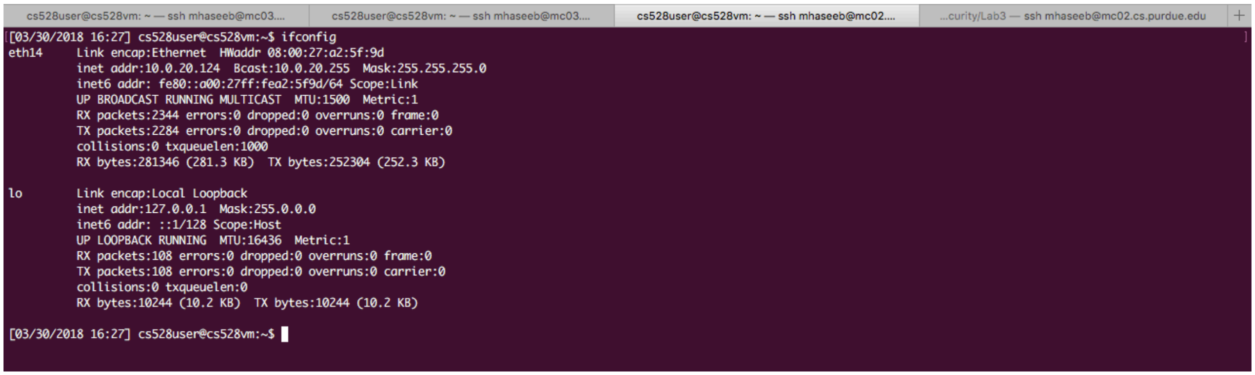
- VirtualHostA1 IP: 10.0.10.143
- VirtualHostB1 IP: 10.0.20.124


cs528user@cs528vm: ~ — ssh mhaseeb@mc03....
cs528user@cs528vm: ~ — ssh mhaseeb@mc03....
cs528user@cs528vm: ~ — ssh mhaseeb@mc02....
...curity/Lab3 — ssh mhaseeb@mc02.cs.purdue.edu

```
[03/30/2018 16:26] cs528user@cs528vm:~$ ifconfig
eth14     Link encap:Ethernet  HWaddr 08:00:27:e3:44:7c
          inet addr:10.0.10.143  Bcast:10.0.10.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fee3:447c/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:2885 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2920 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:545691 (545.6 KB)  TX bytes:309348 (309.3 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:16436  Metric:1
          RX packets:1604 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1604 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:170536 (170.5 KB)  TX bytes:170536 (170.5 KB)

[03/30/2018 16:27] cs528user@cs528vm:~$ 
```

Figure 7

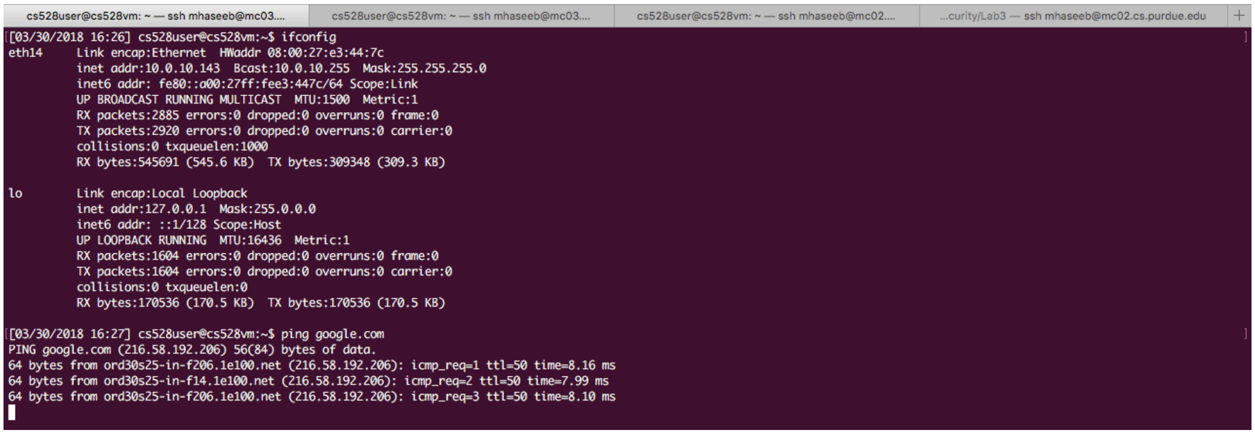
- Figure 7 shows the *ifconfig* on VirtualHostA1. Note that it does not have any virtual network interface set up and it's IP is the once assigned by GatewayA's DHCP configurations.


cs528user@cs528vm: ~ — ssh mhaseeb@mc03....
cs528user@cs528vm: ~ — ssh mhaseeb@mc03....
cs528user@cs528vm: ~ — ssh mhaseeb@mc02....
...curity/Lab3 — ssh mhaseeb@mc02.cs.purdue.edu

```
[03/30/2018 16:27] cs528user@cs528vm:~$ ifconfig
eth14     Link encap:Ethernet  HWaddr 08:00:27:a2:5f:9d
          inet addr:10.0.20.124  Bcast:10.0.20.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fea2:5f9d/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:2344 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2284 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:281346 (281.3 KB)  TX bytes:252304 (252.3 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:16436  Metric:1
          RX packets:108 errors:0 dropped:0 overruns:0 frame:0
          TX packets:108 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:10244 (10.2 KB)  TX bytes:10244 (10.2 KB)

[03/30/2018 16:27] cs528user@cs528vm:~$ 
```

Figure 8

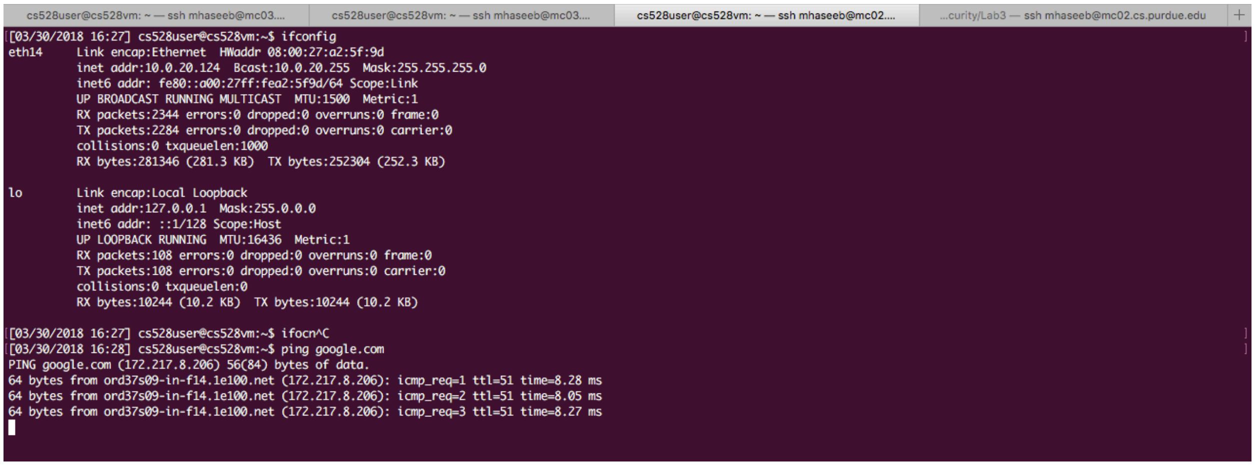
- Figure 8 shows the *ifconfig* on VirtualHostB1. Note that it does not have any virtual network interface set up and it's IP is the once assigned by GatewayB's DHCP configurations.


cs528user@cs528vm: ~ — ssh mhaseeb@mc03....
cs528user@cs528vm: ~ — ssh mhaseeb@mc03....
cs528user@cs528vm: ~ — ssh mhaseeb@mc02....
...curity/Lab3 — ssh mhaseeb@mc02.cs.purdue.edu

```
[03/30/2018 16:26] cs528user@cs528vm:~$ ifconfig
eth14     Link encap:Ethernet  HWaddr 08:00:27:e3:44:7c
          inet addr:10.0.10.143  Bcast:10.0.10.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fee3:447c/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:2885 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2920 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:545691 (545.6 KB)  TX bytes:309348 (309.3 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:16436  Metric:1
          RX packets:1604 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1604 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:170536 (170.5 KB)  TX bytes:170536 (170.5 KB)

[03/30/2018 16:27] cs528user@cs528vm:~$ ping google.com
PING google.com (216.58.192.206) 56(84) bytes of data.
64 bytes from ord30s25-in-f206.1e100.net (216.58.192.206): icmp_req=1 ttl=50 time=8.16 ms
64 bytes from ord30s25-in-f14.1e100.net (216.58.192.206): icmp_req=2 ttl=50 time=7.99 ms
64 bytes from ord30s25-in-f206.1e100.net (216.58.192.206): icmp_req=3 ttl=50 time=8.10 ms
```

Figure 9

- Figure 9 shows that the VirtualHostA1 is able to *ping* google.com

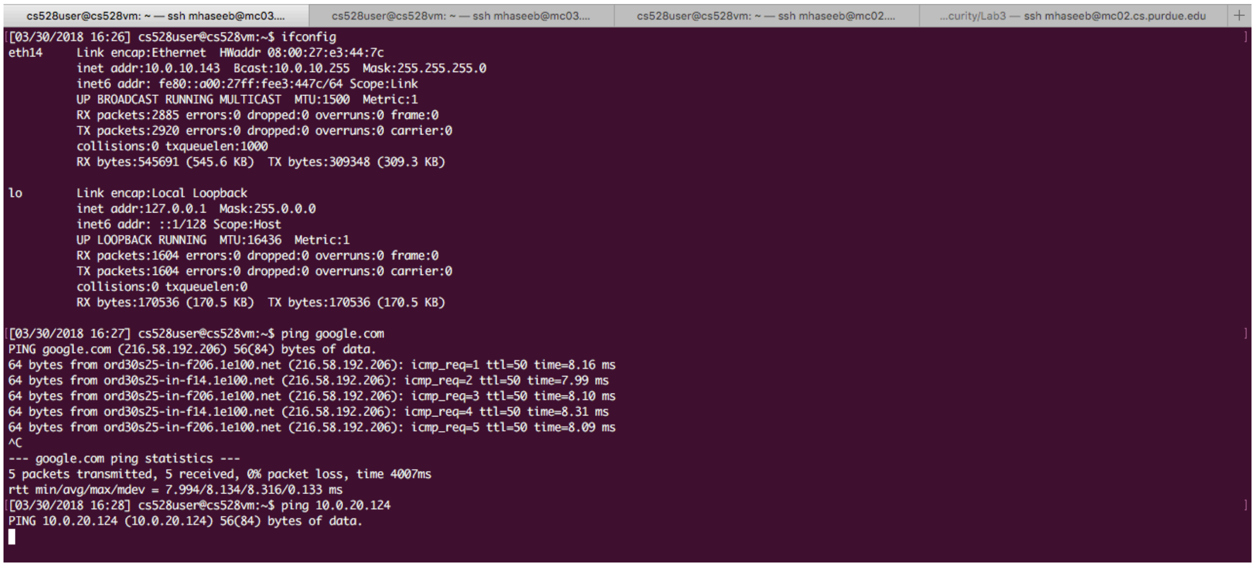


cs528user@cs528vm: ~ — ssh mhaseeb@mc03....
cs528user@cs528vm: ~ — ssh mhaseeb@mc03....
cs528user@cs528vm: ~ — ssh mhaseeb@mc02....
...curity/Lab3 — ssh mhaseeb@mc02.cs.purdue.edu

```
[03/30/2018 16:27] cs528user@cs528vm:~$ ifconfig
eth14     Link encap:Ethernet  HWaddr 08:00:27:a2:5f:9d
          inet addr:10.0.20.124  Bcast:10.0.20.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fea2:5f9d/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:2344 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2284 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:281346 (281.3 KB)  TX bytes:252304 (252.3 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:16436  Metric:1
          RX packets:108 errors:0 dropped:0 overruns:0 frame:0
          TX packets:108 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:10244 (10.2 KB)  TX bytes:10244 (10.2 KB)

[03/30/2018 16:27] cs528user@cs528vm:~$ ifocn^C
[03/30/2018 16:28] cs528user@cs528vm:~$ ping google.com
PING google.com (172.217.8.206) 56(84) bytes of data.
64 bytes from ord37s09-in-f14.1e100.net (172.217.8.206): icmp_req=1 ttl=51 time=8.28 ms
64 bytes from ord37s09-in-f14.1e100.net (172.217.8.206): icmp_req=2 ttl=51 time=8.05 ms
64 bytes from ord37s09-in-f14.1e100.net (172.217.8.206): icmp_req=3 ttl=51 time=8.27 ms
```

Figure 10

- Figure 10 shows the the VirtualHostB1 is also able to *ping* google.com

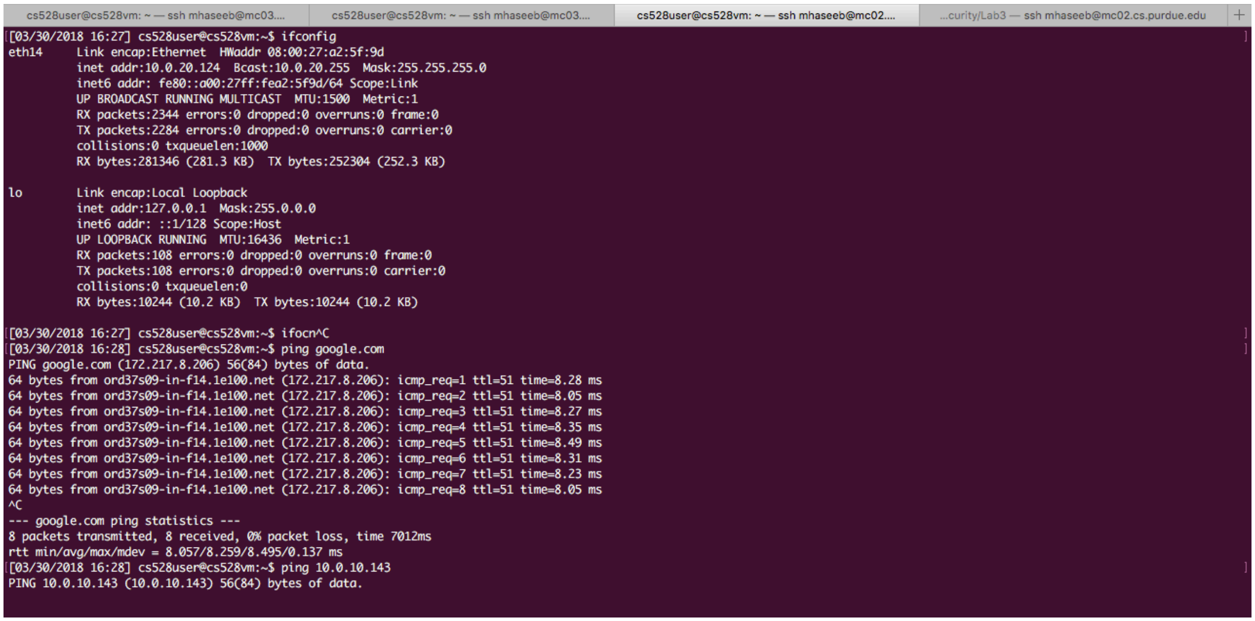


cs528user@cs528vm: ~ — ssh mhaseeb@mc03....
cs528user@cs528vm: ~ — ssh mhaseeb@mc03....
cs528user@cs528vm: ~ — ssh mhaseeb@mc02....
...curity/Lab3 — ssh mhaseeb@mc02.cs.purdue.edu

```
[03/30/2018 16:26] cs528user@cs528vm:~$ ifconfig
eth14     Link encap:Ethernet  HWaddr 08:00:27:e3:44:7c
          inet addr:10.0.10.143  Bcast:10.0.10.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fee3:447c/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:2885 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2920 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:545691 (545.6 KB)  TX bytes:309348 (309.3 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:16436  Metric:1
          RX packets:1604 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1604 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:170536 (170.5 KB)  TX bytes:170536 (170.5 KB)

[03/30/2018 16:27] cs528user@cs528vm:~$ ping google.com
PING google.com (216.58.192.206) 56(84) bytes of data.
64 bytes from ord30s25-in-f206.1e100.net (216.58.192.206): icmp_req=1 ttl=50 time=8.16 ms
64 bytes from ord30s25-in-f14.1e100.net (216.58.192.206): icmp_req=2 ttl=50 time=7.99 ms
64 bytes from ord30s25-in-f206.1e100.net (216.58.192.206): icmp_req=3 ttl=50 time=8.10 ms
64 bytes from ord30s25-in-f14.1e100.net (216.58.192.206): icmp_req=4 ttl=50 time=8.31 ms
64 bytes from ord30s25-in-f206.1e100.net (216.58.192.206): icmp_req=5 ttl=50 time=8.09 ms
^C
--- google.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4007ms
rtt min/avg/max/mdev = 7.994/8.134/8.316/0.133 ms
[03/30/2018 16:28] cs528user@cs528vm:~$ ping 10.0.20.124
PING 10.0.20.124 (10.0.20.124) 56(84) bytes of data.
```

Figure 11

- Figure 11 shows that the VirtualHostA1 is not able to *ping* VirtualHostB1 because a Gateway-to-Gateway tunnel has not yet been setup.


cs528user@cs528vm: ~ — ssh mhaseeb@mc03....
cs528user@cs528vm: ~ — ssh mhaseeb@mc03....
cs528user@cs528vm: ~ — ssh mhaseeb@mc02....
...curity/Lab3 — ssh mhaseeb@mc02.cs.purdue.edu

```
[03/30/2018 16:27] cs528user@cs528vm:~$ ifconfig
eth14     Link encap:Ethernet  HWaddr 08:00:27:a2:5f:9d
          inet addr:10.0.20.124  Bcast:10.0.20.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fea2:5f9d/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:2344 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2284 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:281346 (281.3 KB)  TX bytes:252304 (252.3 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:16436  Metric:1
          RX packets:108 errors:0 dropped:0 overruns:0 frame:0
          TX packets:108 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:10244 (10.2 KB)  TX bytes:10244 (10.2 KB)

[03/30/2018 16:27] cs528user@cs528vm:~$ ifocn^C
[03/30/2018 16:28] cs528user@cs528vm:~$ ping google.com
PING google.com (172.217.8.206) 56(84) bytes of data.
64 bytes from ord37s09-in-f14.1e100.net (172.217.8.206): icmp_req=1 ttl=51 time=8.28 ms
64 bytes from ord37s09-in-f14.1e100.net (172.217.8.206): icmp_req=2 ttl=51 time=8.05 ms
64 bytes from ord37s09-in-f14.1e100.net (172.217.8.206): icmp_req=3 ttl=51 time=8.27 ms
64 bytes from ord37s09-in-f14.1e100.net (172.217.8.206): icmp_req=4 ttl=51 time=8.35 ms
64 bytes from ord37s09-in-f14.1e100.net (172.217.8.206): icmp_req=5 ttl=51 time=8.49 ms
64 bytes from ord37s09-in-f14.1e100.net (172.217.8.206): icmp_req=6 ttl=51 time=8.31 ms
64 bytes from ord37s09-in-f14.1e100.net (172.217.8.206): icmp_req=7 ttl=51 time=8.23 ms
64 bytes from ord37s09-in-f14.1e100.net (172.217.8.206): icmp_req=8 ttl=51 time=8.05 ms
^C
--- google.com ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7012ms
rtt min/avg/max/mdev = 8.057/8.259/8.495/0.137 ms
[03/30/2018 16:28] cs528user@cs528vm:~$ ping 10.0.10.143
PING 10.0.10.143 (10.0.10.143) 56(84) bytes of data.
```

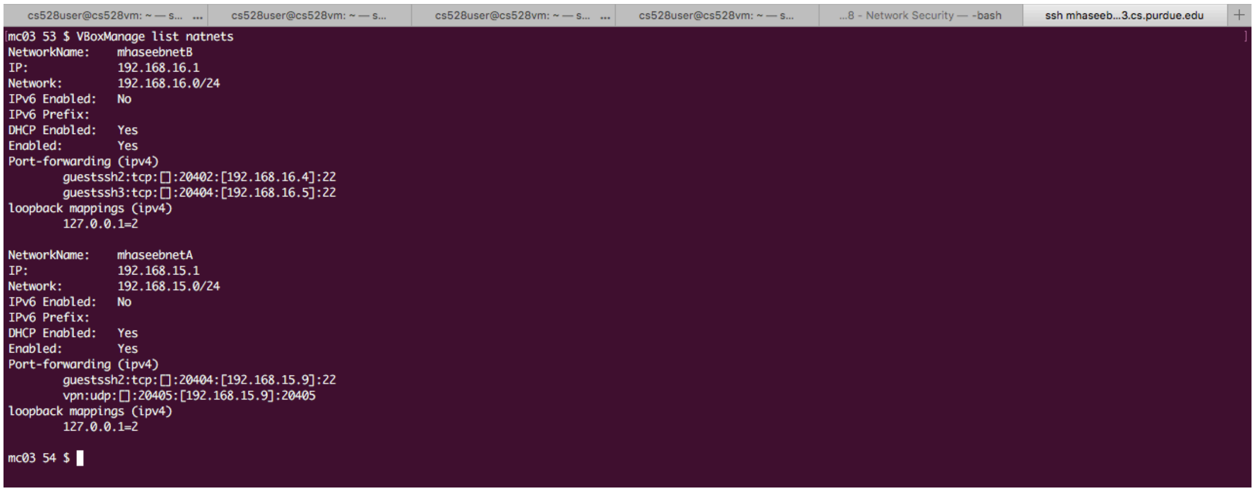
Figure 12

- Figure 11 shows that the VirtualHostB1 is not able to *ping* VirtualHostA1 because a Gateway-to-Gateway tunnel has not yet been setup.

Task 3 Create a Gateway-to-Gateway Tunnel

In this task, I first show the UDP port forwarding rules I added on each of my *mcoX* machines. This was to ensure that the tunnel made was running over UDP and not TCP and that my *simpleton_udp.c* would work on both the gateways. I then show the *ifconfig* for each of the gateways to verify that the *tun0* virtual network interface is up and running and has been assigned the correct *internal network* IP. Finally, I show that I am now able to *ssh* into VirtualHostB1 from VirtualHostA1. This was previously not possible but now is because we have now established a tunnel between the gateways that 'tunnel' the traffic between the virtual hosts on both sides.

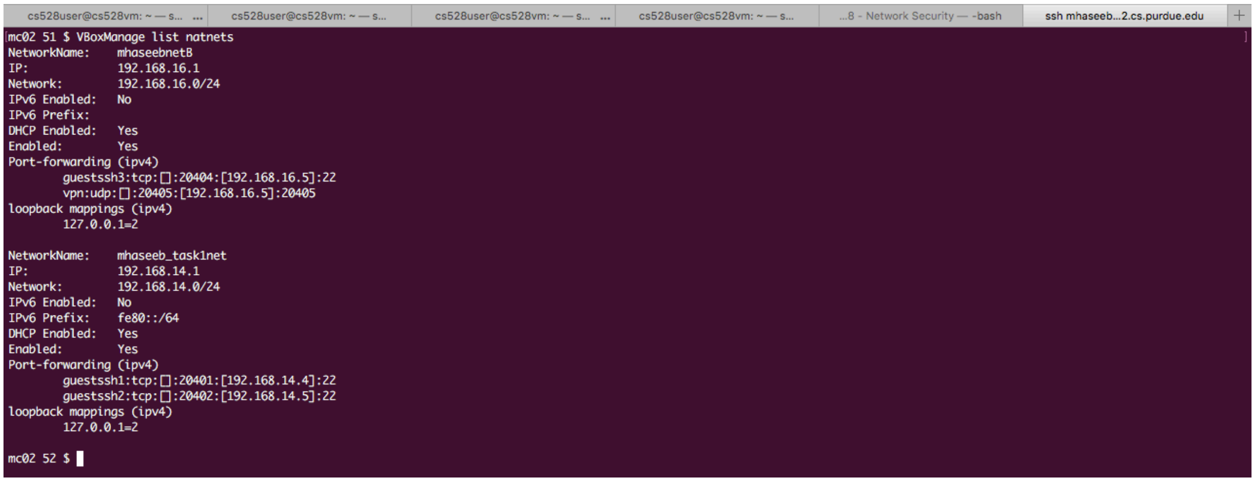
- Mc02 IP address: 128.10.12.202
- Mc03 IP address: 128.10.12.203
- GatewayA IP (was running on mc03): 192.168.15.9
- GatewayB IP (was running on mc02): 192.168.16.5
- VirtualHostA1 IP: 10.0.10.143
- VirtualHostB1 IP: 10.0.20.124


cs528user@cs528vm: ~ — s...
cs528user@cs528vm: ~ — s...
cs528user@cs528vm: ~ — s...
cs528user@cs528vm: ~ — s...
...8 - Network Security — -bash
ssh mhaseeb...3.cs.purdue.edu

```
mc03 53 $ VBoxManage list natnets
NetworkName:    mhaseebnetB
IP:             192.168.16.1
Network:        192.168.16.0/24
IPv6 Enabled:   No
IPv6 Prefix:
DHCP Enabled:   Yes
Enabled:        Yes
Port-forwarding (ipv4)
        guestssh2:tcp:□:20402:[192.168.16.4]:22
        guestssh3:tcp:□:20404:[192.168.16.5]:22
loopback mappings (ipv4)
        127.0.0.1=2

NetworkName:    mhaseebnetA
IP:             192.168.15.1
Network:        192.168.15.0/24
IPv6 Enabled:   No
IPv6 Prefix:
DHCP Enabled:   Yes
Enabled:        Yes
Port-forwarding (ipv4)
        guestssh2:tcp:□:20404:[192.168.15.9]:22
        vpn:udp:□:20405:[192.168.15.9]:20405
loopback mappings (ipv4)
        127.0.0.1=2

mc03 54 $ █
```

Figure 13

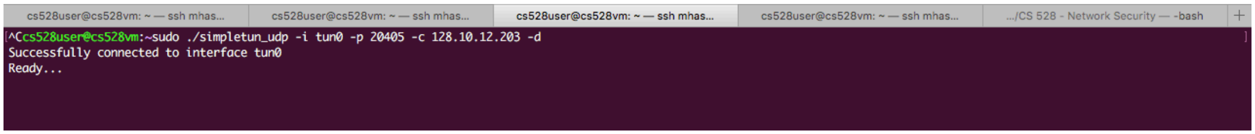
- Figure 13 shows the port forwarding rule I added at mc03 to the *natnetwork* named *mhaseebnetA*. The rule *vpn:udp* forwards all UDP traffic at port 20405 of the physical machine to the virtual machine with IP 192.168.15.9 and its port 20405. This is the IP of my GatewayA. My *simpleton_udp.c* program is running on port 20405 of GatewayA.


cs528user@cs528vm: ~ — s...
cs528user@cs528vm: ~ — s...
cs528user@cs528vm: ~ — s...
cs528user@cs528vm: ~ — s...
...8 - Network Security — -bash
ssh mhaseeb...2.cs.purdue.edu

```
mc02 51 $ VBoxManage list natnets
NetworkName:    mhaseebnetB
IP:             192.168.16.1
Network:        192.168.16.0/24
IPv6 Enabled:   No
IPv6 Prefix:
DHCP Enabled:   Yes
Enabled:        Yes
Port-forwarding (ipv4)
        guestssh3:tcp:□:20404:[192.168.16.5]:22
        vpn:udp:□:20405:[192.168.16.5]:20405
loopback mappings (ipv4)
        127.0.0.1=2

NetworkName:    mhaseeb_task1net
IP:             192.168.14.1
Network:        192.168.14.0/24
IPv6 Enabled:   No
IPv6 Prefix:    fe80::/64
DHCP Enabled:   Yes
Enabled:        Yes
Port-forwarding (ipv4)
        guestssh1:tcp:□:20401:[192.168.14.4]:22
        guestssh2:tcp:□:20402:[192.168.14.5]:22
loopback mappings (ipv4)
        127.0.0.1=2

mc02 52 $ █
```

Figure 14

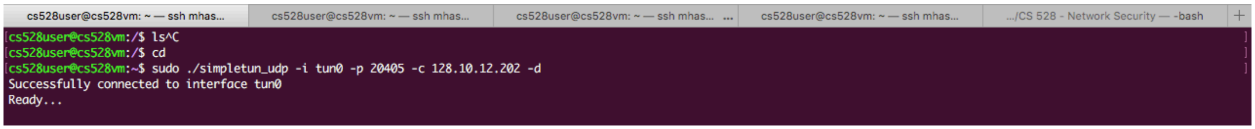
- Figure 14 shows the port forwarding rule I added at mc02 to the *natnetwork* named *mhaseebnetB*. The rule *vpn:udp* forwards all UDP traffic at port 20405 of the physical machine to the virtual machine with IP 192.168.16.5 and its port 20405. This is the IP of my GatewayB. My *simpleton_udp.c* program is running on port 20405 of GatewayB.


cs528user@cs528vm: ~ — ssh mhas...
cs528user@cs528vm: ~ — ssh mhas...
cs528user@cs528vm: ~ — ssh mhas...
cs528user@cs528vm: ~ — ssh mhas...
.../CS 528 - Network Security — -bash

```
^Ccs528user@cs528vm:~sudo ./simpletun_udp -i tun0 -p 20405 -c 128.10.12.203 -d
Successfully connected to interface tun0
Ready...
```

Figure 15

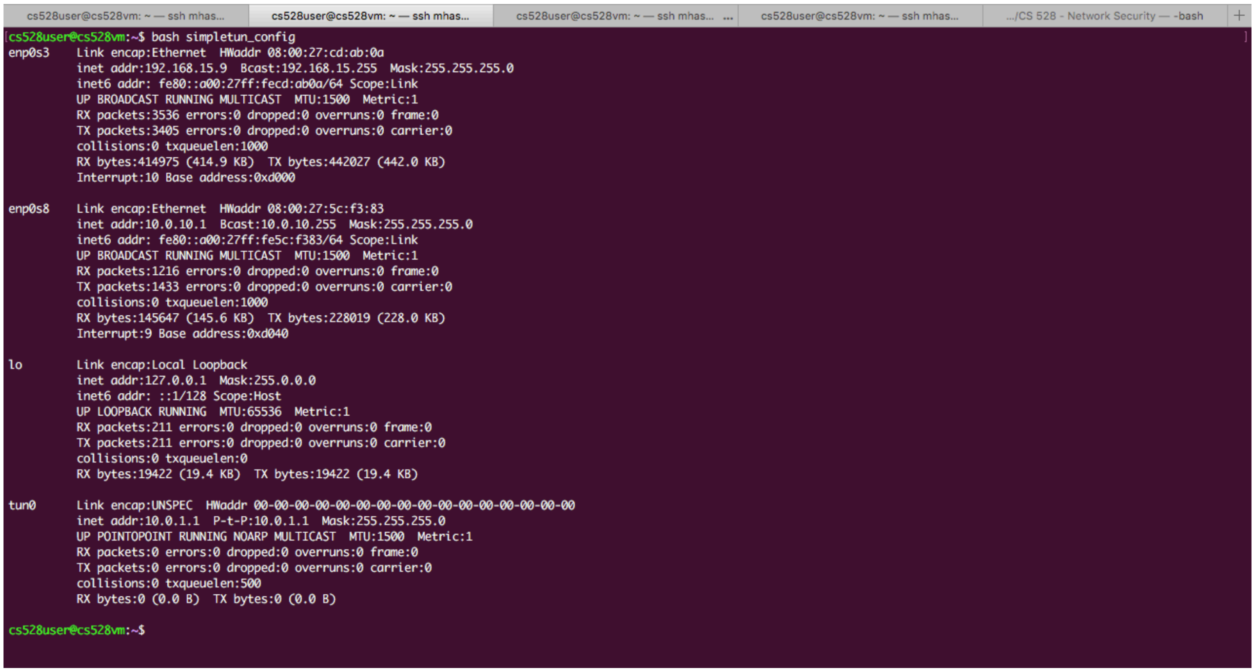
- Figure 15 shows that the GatewayB (my server machine) is up and running and is connecting to the VM running on machine with IP 128.10.12.203 (mc03) and port 20405. The UDP port forwarding rule added on the mc03 as shown in Figure 13 will forward this traffic to the GatewayA VM.


cs528user@cs528vm: ~ — ssh mhas...
cs528user@cs528vm: ~ — ssh mhas...
cs528user@cs528vm: ~ — ssh mhas... ...
cs528user@cs528vm: ~ — ssh mhas...
.../CS 528 - Network Security — -bash

```
cs528user@cs528vm:/$ ls^C
cs528user@cs528vm:/$ cd
cs528user@cs528vm:~$ sudo ./simpletun_udp -i tun0 -p 20405 -c 128.10.12.202 -d
Successfully connected to interface tun0
Ready...
```

Figure 16

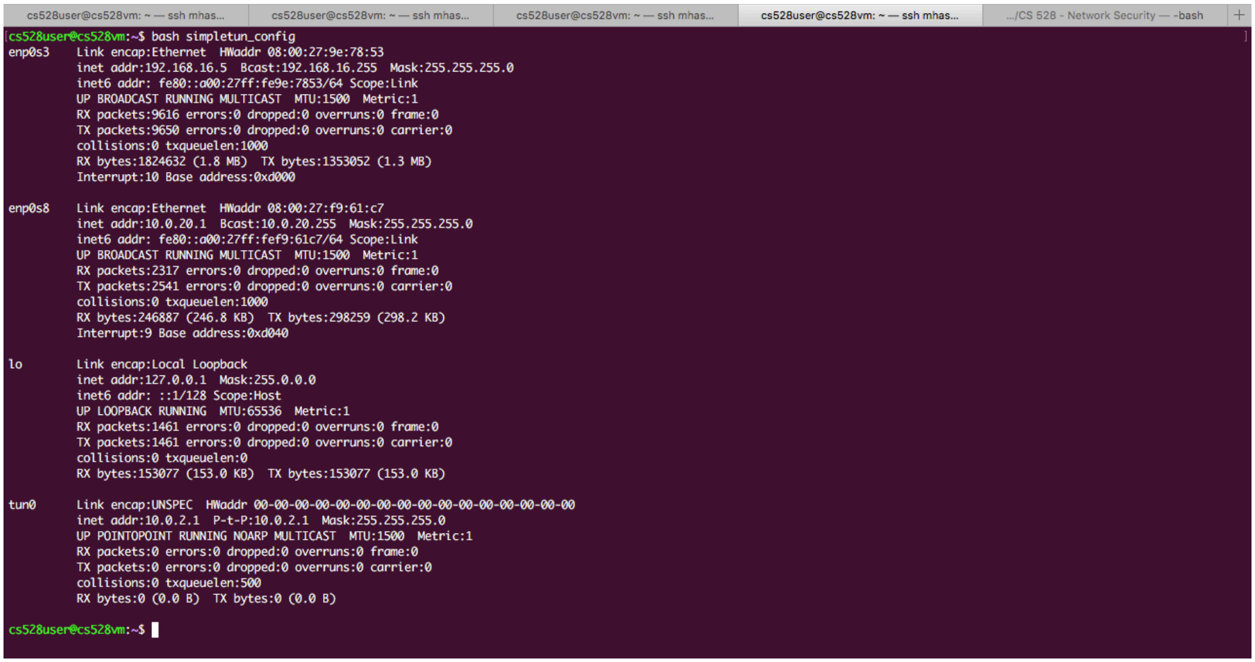
- Figure 15 shows that the GatewayA (my client machine) is up and running and is connecting to the VM running on machine with IP 128.10.12.202 (mc02) and port 20405. The UDP port forwarding rule added on the mc02 as shown in Figure 14 will forward this traffic to the GatewayB VM.


cs528user@cs528vm: ~ — ssh mhas...
cs528user@cs528vm: ~ — ssh mhas...
cs528user@cs528vm: ~ — ssh mhas... ...
cs528user@cs528vm: ~ — ssh mhas...
.../CS 528 - Network Security — -bash

```
cs528user@cs528vm:~$ bash simpletun_config
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:cd:ab:0a
          inet addr:192.168.15.9  Bcast:192.168.15.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fecd:ab0a/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:3536 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3405 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:414975 (414.9 KB)  TX bytes:442027 (442.0 KB)
          Interrupt:10 Base address:0xd000

enp0s8    Link encap:Ethernet  HWaddr 08:00:27:5c:f3:83
          inet addr:10.0.10.1  Bcast:10.0.10.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe5c:f383/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1216 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1433 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:145647 (145.6 KB)  TX bytes:228019 (228.0 KB)
          Interrupt:9 Base address:0xd040

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:211 errors:0 dropped:0 overruns:0 frame:0
          TX packets:211 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:19422 (19.4 KB)  TX bytes:19422 (19.4 KB)

tun0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          inet addr:10.0.1.1  P-t-P:10.0.1.1  Mask:255.255.255.0
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

cs528user@cs528vm:~$
```

Figure 17

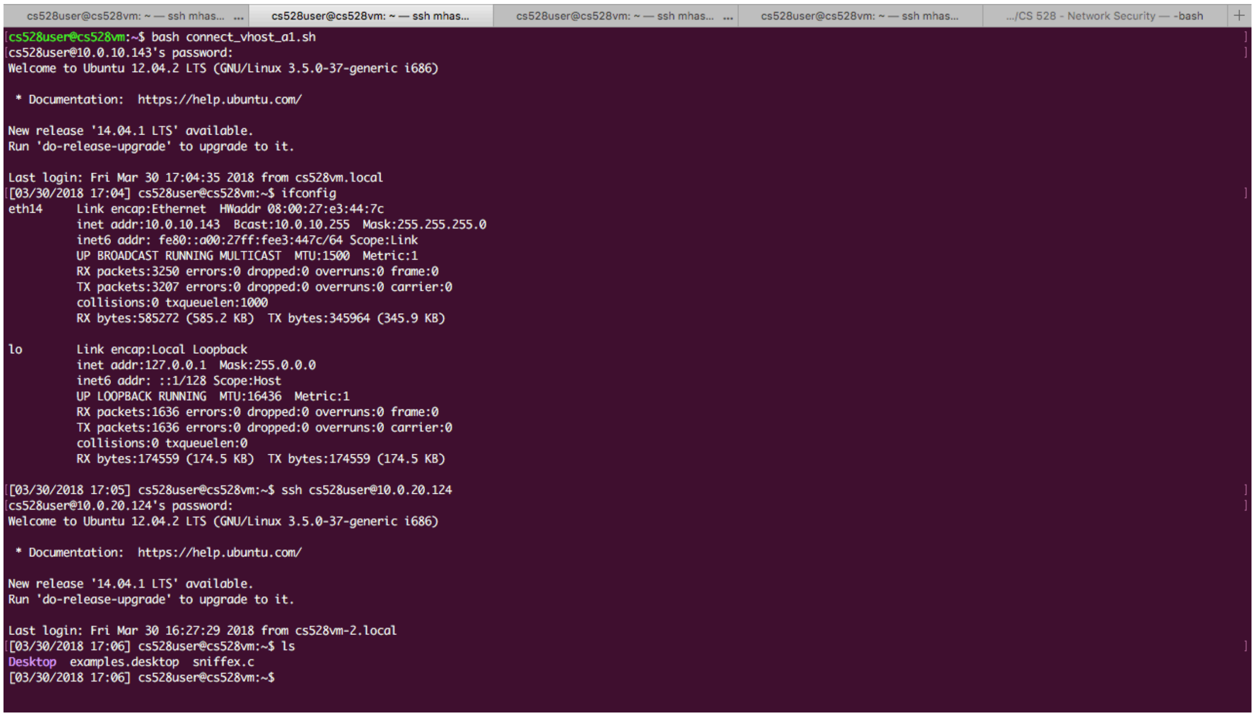
- Figure 17 shows that the *tun0* virtual network interface is up and has been assigned an IP as per the rules I added in the *simpleton_config* bash file. This is on GatewayA hence the IP of *tun0* is 10.0.1.1


cs528user@cs528vm: ~ — ssh mhas...
cs528user@cs528vm: ~ — ssh mhas...
cs528user@cs528vm: ~ — ssh mhas...
cs528user@cs528vm: ~ — ssh mhas...
.../CS 528 - Network Security — -bash

```
cs528user@cs528vm:~$ bash simpletun_config
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:9e:78:53
          inet addr:192.168.16.5  Bcast:192.168.16.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe9e:7853/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:9616 errors:0 dropped:0 overruns:0 frame:0
          TX packets:9650 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1824632 (1.8 MB)  TX bytes:1353052 (1.3 MB)
          Interrupt:10 Base address:0xd000

enp0s8    Link encap:Ethernet  HWaddr 08:00:27:f9:61:c7
          inet addr:10.0.20.1  Bcast:10.0.20.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fef9:61c7/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:2317 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2541 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:246887 (246.8 KB)  TX bytes:298259 (298.2 KB)
          Interrupt:9 Base address:0xd040

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:1461 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1461 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:153077 (153.0 KB)  TX bytes:153077 (153.0 KB)

tun0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          inet addr:10.0.2.1  P-t-P:10.0.2.1  Mask:255.255.255.0
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

cs528user@cs528vm:~$ 
```

Figure 18

- Figure 18 shows that the *tuno* virtual network interface is up and has been assigned an IP as per the rules I added in the *simpleton_config* bash file. This is on GatewayB hence the IP of *tuno* is 10.0.2.1


cs528user@cs528vm: ~ — ssh mhas... ...
cs528user@cs528vm: ~ — ssh mhas...
cs528user@cs528vm: ~ — ssh mhas... ...
cs528user@cs528vm: ~ — ssh mhas...
.../CS 528 - Network Security — -bash

```
cs528user@cs528vm:~$ bash connect_vhost_a1.sh
cs528user@10.0.10.143's password:
Welcome to Ubuntu 12.04.2 LTS (GNU/Linux 3.5.0-37-generic i686)

 * Documentation:  https://help.ubuntu.com/

New release '14.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Fri Mar 30 17:04:35 2018 from cs528vm.local
[03/30/2018 17:04] cs528user@cs528vm:~$ ifconfig
eth14     Link encap:Ethernet  HWaddr 08:00:27:e3:44:7c
          inet addr:10.0.10.143  Bcast:10.0.10.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fee3:447c/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:3250 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3207 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:585272 (585.2 KB)  TX bytes:345964 (345.9 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:16436  Metric:1
          RX packets:1636 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1636 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:174559 (174.5 KB)  TX bytes:174559 (174.5 KB)

[03/30/2018 17:05] cs528user@cs528vm:~$ ssh cs528user@10.0.20.124
cs528user@10.0.20.124's password:
Welcome to Ubuntu 12.04.2 LTS (GNU/Linux 3.5.0-37-generic i686)

 * Documentation:  https://help.ubuntu.com/

New release '14.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Fri Mar 30 16:27:29 2018 from cs528vm-2.local
[03/30/2018 17:06] cs528user@cs528vm:~$ ls
Desktop  examples.desktop  sniffex.c
[03/30/2018 17:06] cs528user@cs528vm:~$
```

Figure 19

- Figure 19 shows that I have logged into the VirtualHostA1 and am able to *ssh* into VirtualHostB1. This shows that my Gateway-to-Gateway tunnel is working fine and that the two virtual hosts behind their respective gateways can talk to each other.

Task 4: Create a Virtual Private Network (VPN)

Because tasks 4 and 5 are interrelated, I have included screenshots for both tasks together in the next section.

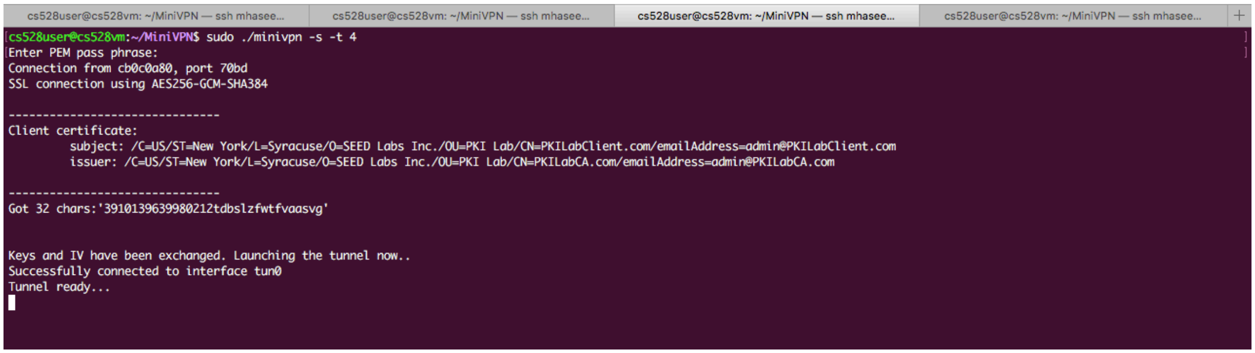
Task 5: Authentication and Key Exchange

Notes:

- For this task, my GatewayA image got deleted so I had to re-setup the A side of the tunnel. Therefore, the IP's for the A side are different from Task 3. I mention the IP's here again for convenience:
    - Mc02 IP address: 128.10.12.202
    - Mc03 IP address: 128.10.12.203
    - GatewayA IP (was running on mc03): 192.168.15.11
    - GatewayB IP (was running on mc02): 192.168.16.5
    - VirtualHostA1 IP: 10.0.10.172
    - VirtualHostB1 IP: 10.0.20.124
- Although I have used *ping* to test the functionality of the VPN, the same can be shown using *ssh* as will be demonstrated during the lab demo to the TA.
- In the following screenshots, the *-t* flag being passed while running the *minivpn* program indicates the test number to execute. I have written tests numbered 1-6 each testing a different feature of the VPN software.

Authentication

The following screenshots will show different parts of the authentication i.e. the exchange and display of each side's certificates at the other side, authentication at both the sides, and finally, no authentication in case of a forged certificate being presented by either party.

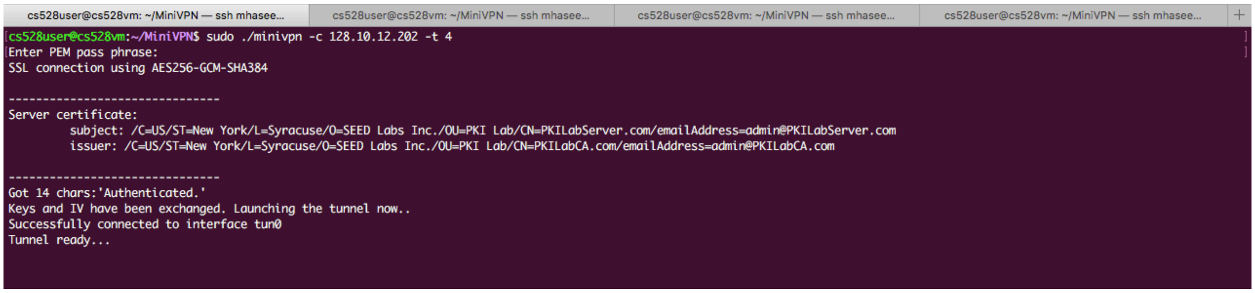


cs528user@cs528vm: ~/MiniVPN — ssh mhasee...
cs528user@cs528vm: ~/MiniVPN — ssh mhasee...
cs528user@cs528vm: ~/MiniVPN — ssh mhasee...
cs528user@cs528vm: ~/MiniVPN — ssh mhasee...

```
cs528user@cs528vm:~/MiniVPN$ sudo ./minivpn -s -t 4
Enter PEM pass phrase:
Connection from cb0c0a80, port 70bd
SSL connection using AES256-GCM-SHA384

------------------------------
Client certificate:
         subject: /C=US/ST=New York/L=Syracuse/O=SEED Labs Inc./OU=PKI Lab/CN=PKILabClient.com/emailAddress=admin@PKILabClient.com
         issuer: /C=US/ST=New York/L=Syracuse/O=SEED Labs Inc./OU=PKI Lab/CN=PKILabCA.com/emailAddress=admin@PKILabCA.com

------------------------------
Got 32 chars:'3910139639980212tdbslzfwtfvaasvg'


Keys and IV have been exchanged. Launching the tunnel now..
Successfully connected to interface tun0
Tunnel ready...
```

Figure 20

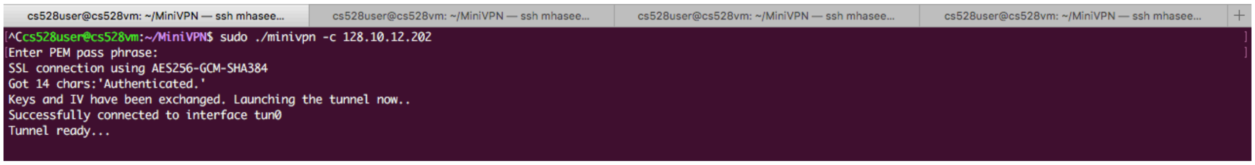
- Figure 20 shows the client's certificate as received and printed at the server.


cs528user@cs528vm: ~/MiniVPN — ssh mhasee...
cs528user@cs528vm: ~/MiniVPN — ssh mhasee...
cs528user@cs528vm: ~/MiniVPN — ssh mhasee...
cs528user@cs528vm: ~/MiniVPN — ssh mhasee...

```
cs528user@cs528vm:~/MiniVPN$ sudo ./minivpn -c 128.10.12.202 -t 4
Enter PEM pass phrase:
SSL connection using AES256-GCM-SHA384

------------------------------
Server certificate:
         subject: /C=US/ST=New York/L=Syracuse/O=SEED Labs Inc./OU=PKI Lab/CN=PKILabServer.com/emailAddress=admin@PKILabServer.com
         issuer: /C=US/ST=New York/L=Syracuse/O=SEED Labs Inc./OU=PKI Lab/CN=PKILabCA.com/emailAddress=admin@PKILabCA.com

------------------------------
Got 14 chars:'Authenticated.'
Keys and IV have been exchanged. Launching the tunnel now..
Successfully connected to interface tun0
Tunnel ready...
```

Figure 21

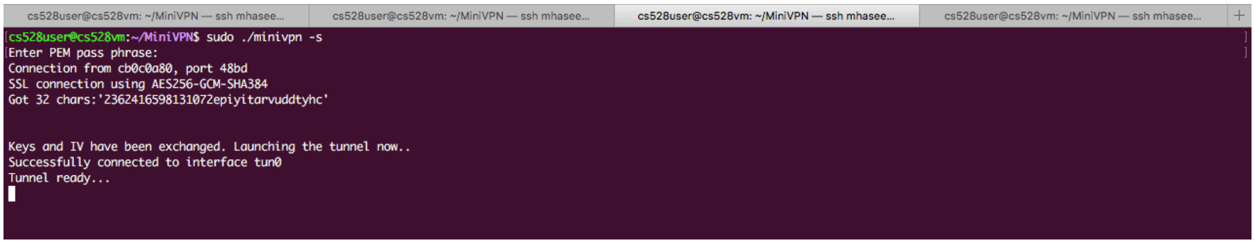
- Figure 21 shows the server's certificate as received and printed at the client.


cs528user@cs528vm: ~/MiniVPN — ssh mhasee...
cs528user@cs528vm: ~/MiniVPN — ssh mhasee...
cs528user@cs528vm: ~/MiniVPN — ssh mhasee...
cs528user@cs528vm: ~/MiniVPN — ssh mhasee...

```
^Ccs528user@cs528vm:~/MiniVPN$ sudo ./minivpn -c 128.10.12.202
Enter PEM pass phrase:
SSL connection using AES256-GCM-SHA384
Got 14 chars:'Authenticated.'
Keys and IV have been exchanged. Launching the tunnel now..
Successfully connected to interface tun0
Tunnel ready...
```

Figure 22

- Figure 22 shows the client has successfully verified the server.

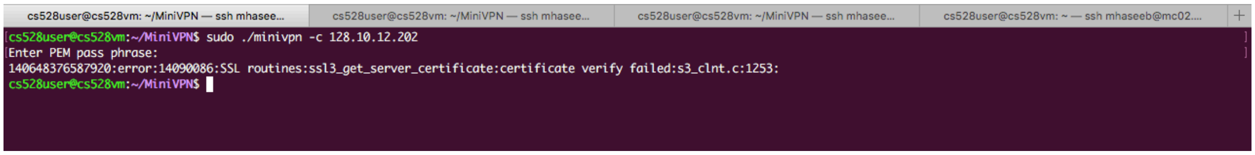

cs528user@cs528vm: ~/MiniVPN — ssh mhasee...
cs528user@cs528vm: ~/MiniVPN — ssh mhasee...
cs528user@cs528vm: ~/MiniVPN — ssh mhasee...
cs528user@cs528vm: ~/MiniVPN — ssh mhasee...

```
cs528user@cs528vm:~/MiniVPN$ sudo ./minivpn -s
Enter PEM pass phrase:
Connection from cb0c0a80, port 48bd
SSL connection using AES256-GCM-SHA384
Got 32 chars:'2362416598131072epiyitarvuddtyhc'


Keys and IV have been exchanged. Launching the tunnel now..
Successfully connected to interface tun0
Tunnel ready...
```

Figure 23

- Figure 23 shows the state of the server after it has verified the client's certificate using the client's public key that is available to the server.

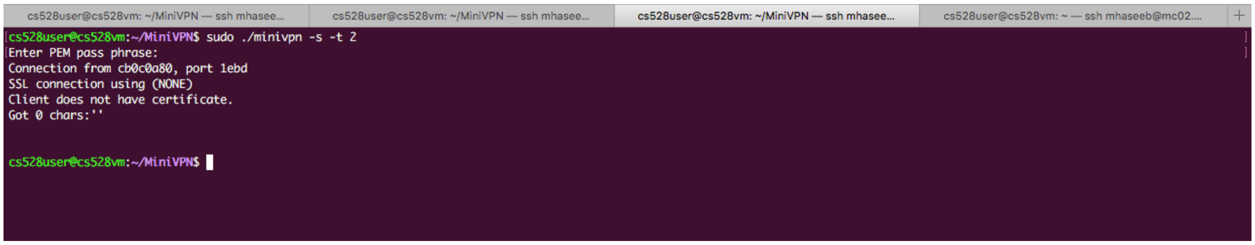
Server Authentication: Forged certificate from server


cs528user@cs528vm: ~/MiniVPN — ssh mhasee...
cs528user@cs528vm: ~/MiniVPN — ssh mhasee...
cs528user@cs528vm: ~/MiniVPN — ssh mhasee...
cs528user@cs528vm: ~ — ssh mhaseeb@mc02....

```
cs528user@cs528vm:~/MiniVPN$ sudo ./minivpn -c 128.10.12.202
Enter PEM pass phrase:
140648376587920:error:14090086:SSL routines:ssl3_get_server_certificate:certificate verify failed:s3_clnt.c:1253:
cs528user@cs528vm:~/MiniVPN$
```

Figure 24

- Figure 24 shows the client not being able to authenticate the server because the server presented it with a forged certificate.

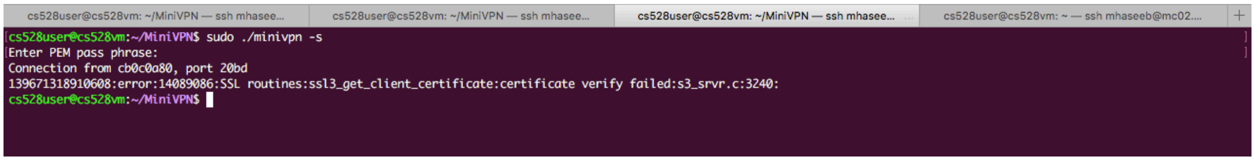

cs528user@cs528vm: ~/MiniVPN — ssh mhasee...
cs528user@cs528vm: ~/MiniVPN — ssh mhasee...
cs528user@cs528vm: ~/MiniVPN — ssh mhasee...
cs528user@cs528vm: ~ — ssh mhaseeb@mc02...

```
cs528user@cs528vm:~/MiniVPN$ sudo ./minivpn -s -t 2
Enter PEM pass phrase:
Connection from cb0c0a80, port 1ebd
SSL connection using (NONE)
Client does not have certificate.
Got 0 chars:''


cs528user@cs528vm:~/MiniVPN$ 
```

Figure 25

- Figure 25 shows that the UDP tunnel did not get setup because the authentication failed.

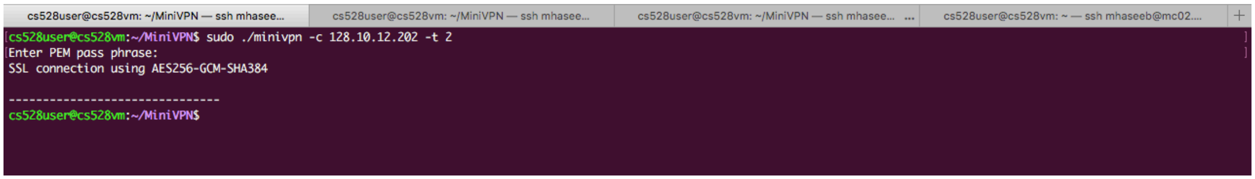
Client Authentication: forged certificate from client


cs528user@cs528vm: ~/MiniVPN — ssh mhasee...
cs528user@cs528vm: ~/MiniVPN — ssh mhasee...
cs528user@cs528vm: ~/MiniVPN — ssh mhasee...
cs528user@cs528vm: ~ — ssh mhaseeb@mc02...

```
cs528user@cs528vm:~/MiniVPN$ sudo ./minivpn -s
Enter PEM pass phrase:
Connection from cb0c0a80, port 20bd
139671318910608:error:14089086:SSL routines:ssl3_get_client_certificate:certificate verify failed:s3_srvr.c:3240:
cs528user@cs528vm:~/MiniVPN$ 
```

Figure 26

- Figure 26 shows the server not being able to authenticate the client because the client presented it with a forged certificate.

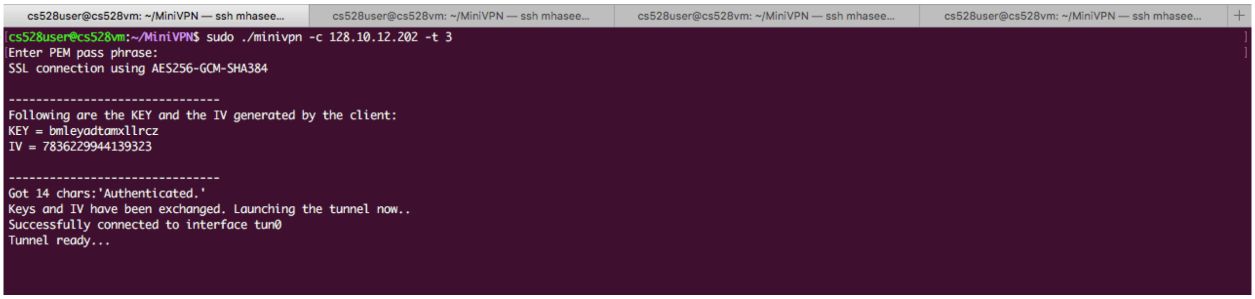

cs528user@cs528vm: ~/MiniVPN — ssh mhasee...
cs528user@cs528vm: ~/MiniVPN — ssh mhasee...
cs528user@cs528vm: ~/MiniVPN — ssh mhasee...
cs528user@cs528vm: ~ — ssh mhaseeb@mc02...

```
cs528user@cs528vm:~/MiniVPN$ sudo ./minivpn -c 128.10.12.202 -t 2
Enter PEM pass phrase:
SSL connection using AES256-GCM-SHA384

------------------------------
cs528user@cs528vm:~/MiniVPN$
```

Figure 27

- Figure 27 shows that the UDP tunnel did not get setup because the authentication failed.

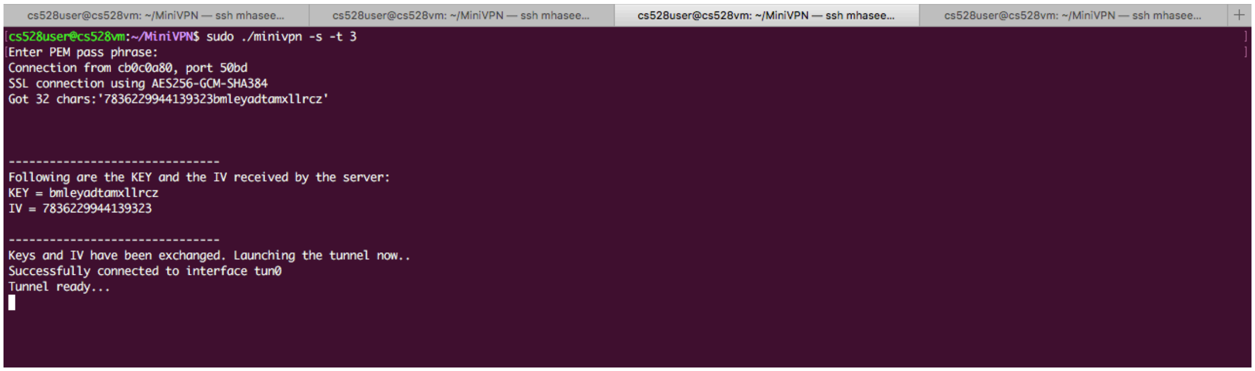
Key Generation and Exchange


cs528user@cs528vm: ~/MiniVPN — ssh mhasee...
cs528user@cs528vm: ~/MiniVPN — ssh mhasee...
cs528user@cs528vm: ~/MiniVPN — ssh mhasee...
cs528user@cs528vm: ~/MiniVPN — ssh mhasee...

```
cs528user@cs528vm:~/MiniVPN$ sudo ./minivpn -c 128.10.12.202 -t 3
Enter PEM pass phrase:
SSL connection using AES256-GCM-SHA384

------------------------------
Following are the KEY and the IV generated by the client:
KEY = bmleyadtamxllrcz
IV = 7836229944139323

------------------------------
Got 14 chars:'Authenticated.'
Keys and IV have been exchanged. Launching the tunnel now..
Successfully connected to interface tun0
Tunnel ready...
```

Figure 28

- Figure 28 shows the client generating its key and IV and exchanging them with the server over SSL after both parties have been authenticated.



cs528user@cs528vm: ~/MiniVPN — ssh mhasee...
cs528user@cs528vm: ~/MiniVPN — ssh mhasee...
cs528user@cs528vm: ~/MiniVPN — ssh mhasee...
cs528user@cs528vm: ~/MiniVPN — ssh mhasee...

```
cs528user@cs528vm:~/MiniVPN$ sudo ./minivpn -s -t 3
Enter PEM pass phrase:
Connection from cb0c0a80, port 50bd
SSL connection using AES256-GCM-SHA384
Got 32 chars:'7836229944139323bmleyadtamxllrcz'



------------------------------
Following are the KEY and the IV received by the server:
KEY = bmleyadtamxllrcz
IV = 7836229944139323

------------------------------
Keys and IV have been exchanged. Launching the tunnel now..
Successfully connected to interface tun0
Tunnel ready...
```

Figure 29

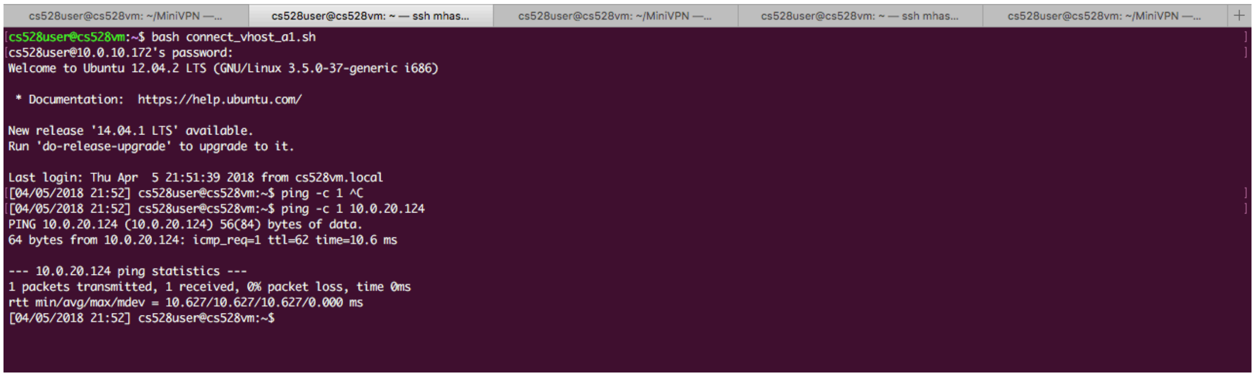
- Figure 29 shows the server receiving the same key and IV generated and sent by the client over SSL. This is done after both parties have authenticated each other. At this point both the client and server have established and exchanged their secret key, IV pair.

Providing Confidentiality (via Encryption)

The following screenshots will show the traffic being sent over the VPN is encrypted. This shows the confidentiality feature of my VPN.

Notes:

- For the purpose of the report, I am showing the encrypted traffic for one exchange of *ping* packets between VirtualHostA1 and VirtualHostB1. The same feature works for other protocols like *ssl* as well as will be shown to the TA in the demo.
- I have omitted screenshots that verify that the virtual network interface has been setup as that was shown in the previous sections.

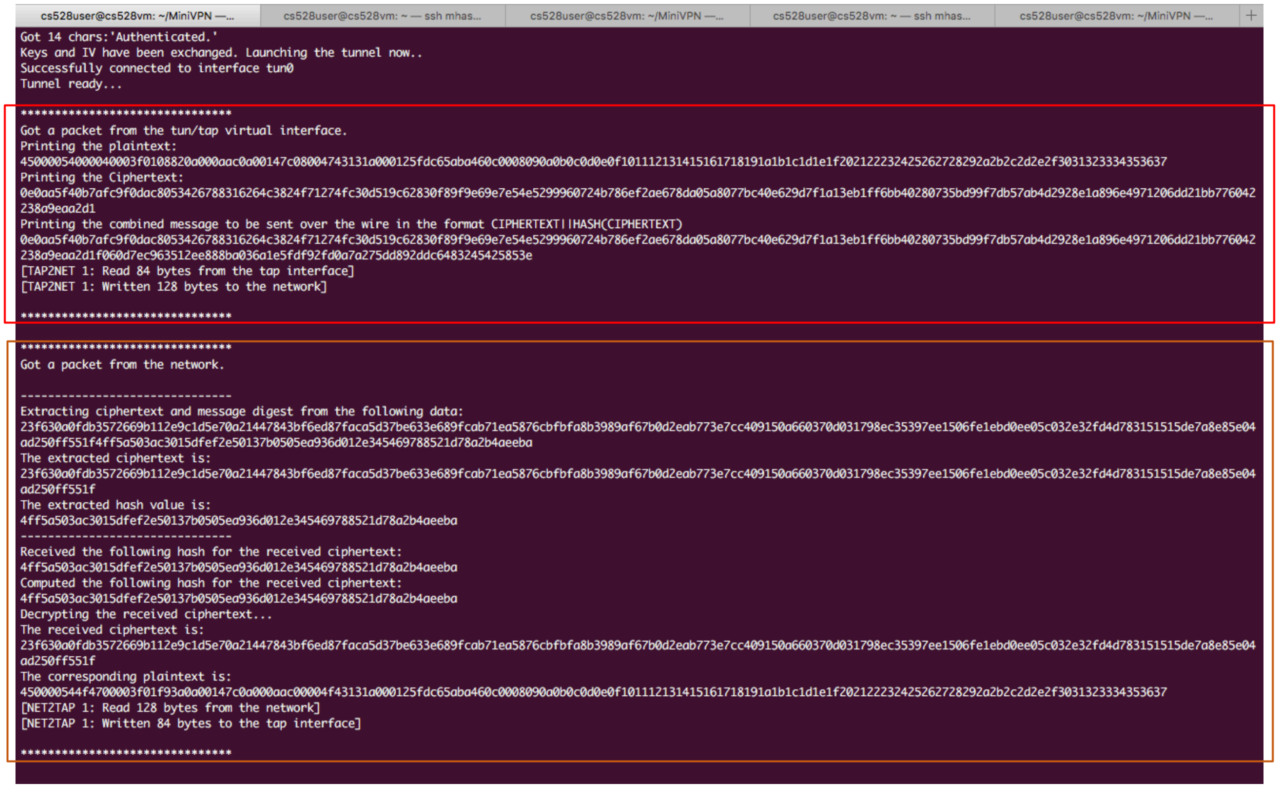


cs528user@cs528vm: ~/MiniVPN —...
cs528user@cs528vm: ~ — ssh mhas...
cs528user@cs528vm: ~/MiniVPN —...
cs528user@cs528vm: ~ — ssh mhas...
cs528user@cs528vm: ~/MiniVPN —...

```
cs528user@cs528vm:~$ bash connect_vhost_a1.sh
cs528user@10.0.10.172's password:
Welcome to Ubuntu 12.04.2 LTS (GNU/Linux 3.5.0-37-generic i686)

 * Documentation:  https://help.ubuntu.com/

New release '14.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Thu Apr  5 21:51:39 2018 from cs528vm.local
[04/05/2018 21:52] cs528user@cs528vm:~$ ping -c 1 ^C
[04/05/2018 21:52] cs528user@cs528vm:~$ ping -c 1 10.0.20.124
PING 10.0.20.124 (10.0.20.124) 56(84) bytes of data.
64 bytes from 10.0.20.124: icmp_req=1 ttl=62 time=10.6 ms

--- 10.0.20.124 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 10.627/10.627/10.627/0.000 ms
[04/05/2018 21:52] cs528user@cs528vm:~$
```

Figure 30

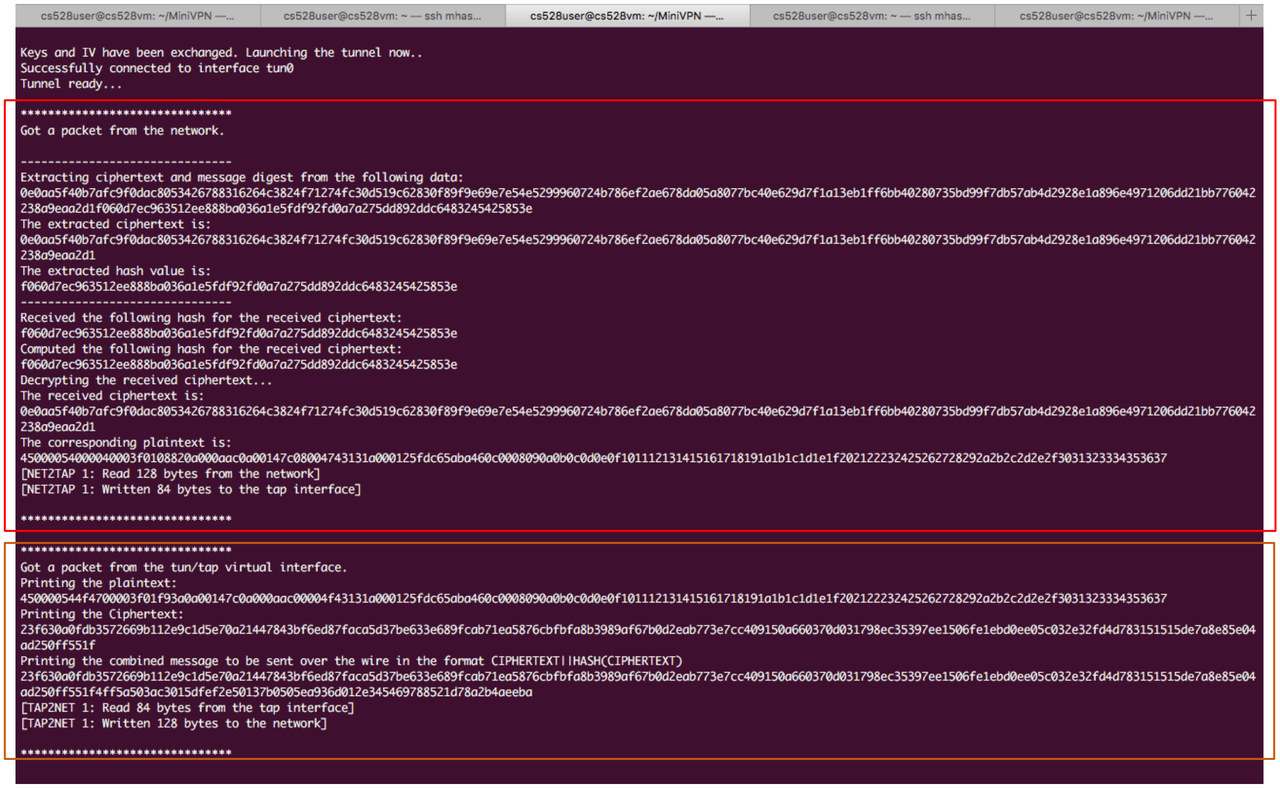
- Figure 30 shows one *ping* packet being sent from VirtualHostA1 to VirtualHostB1. The *ping* succeeded and there was 0% packet loss.


cs528user@cs528vm: ~/MiniVPN —...
cs528user@cs528vm: ~ — ssh mhas...
cs528user@cs528vm: ~/MiniVPN —...
cs528user@cs528vm: ~ — ssh mhas...
cs528user@cs528vm: ~/MiniVPN —...

```
Got 14 chars:'Authenticated.'
Keys and IV have been exchanged. Launching the tunnel now..
Successfully connected to interface tun0
Tunnel ready...

******************************
Got a packet from the tun/tap virtual interface.
Printing the plaintext:
45000054000040003f0108820a000aac0a00147c08004743131a000125fdc65aba460c0008090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f3031323334353637
Printing the Ciphertext:
0e0aa5f40b7afc9f0dac8053426788316264c3824f71274fc30d519c62830f89f9e69e7e54e5299960724b786ef2ae678da05a8077bc40e629d7f1a13eb1ff6bb40280735bd99f7db57ab4d2928e1a896e4971206dd21bb776042
238a9eaa2d1
Printing the combined message to be sent over the wire in the format CIPHERTEXT||HASH(CIPHERTEXT)
0e0aa5f40b7afc9f0dac8053426788316264c3824f71274fc30d519c62830f89f9e69e7e54e5299960724b786ef2ae678da05a8077bc40e629d7f1a13eb1ff6bb40280735bd99f7db57ab4d2928e1a896e4971206dd21bb776042
238a9eaa2d1f060d7ec963512ee888ba036a1e5fdf92fd0a7a275dd892ddc6483245425853e
[TAP2NET 1: Read 84 bytes from the tap interface]
[TAP2NET 1: Written 128 bytes to the network]

******************************

******************************
Got a packet from the network.

------------------------------
Extracting ciphertext and message digest from the following data:
23f630a0fdb3572669b112e9c1d5e70a21447843bf6ed87faca5d37be633e689fcab71ea5876cbfbfa8b3989af67b0d2eab773e7cc409150a660370d031798ec35397ee1506fe1ebd0ee05c032e32fd4d783151515de7a8e85e04
ad250ff551f4ff5a503ac3015dfef2e50137b0505ea936d012e345469788521d78a2b4aeeba
The extracted ciphertext is:
23f630a0fdb3572669b112e9c1d5e70a21447843bf6ed87faca5d37be633e689fcab71ea5876cbfbfa8b3989af67b0d2eab773e7cc409150a660370d031798ec35397ee1506fe1ebd0ee05c032e32fd4d783151515de7a8e85e04
ad250ff551f
The extracted hash value is:
4ff5a503ac3015dfef2e50137b0505ea936d012e345469788521d78a2b4aeeba
------------------------------
Received the following hash for the received ciphertext:
4ff5a503ac3015dfef2e50137b0505ea936d012e345469788521d78a2b4aeeba
Computed the following hash for the received ciphertext:
4ff5a503ac3015dfef2e50137b0505ea936d012e345469788521d78a2b4aeeba
Decrypting the received ciphertext...
The received ciphertext is:
23f630a0fdb3572669b112e9c1d5e70a21447843bf6ed87faca5d37be633e689fcab71ea5876cbfbfa8b3989af67b0d2eab773e7cc409150a660370d031798ec35397ee1506fe1ebd0ee05c032e32fd4d783151515de7a8e85e04
ad250ff551f
The corresponding plaintext is:
450000544f4700003f01f93a0a00147c0a000aac00004f43131a000125fdc65aba460c0008090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f3031323334353637
[NET2TAP 1: Read 128 bytes from the network]
[NET2TAP 1: Written 84 bytes to the tap interface]

******************************
```

Figure 31

- Figure 31 shows the *ping* packet being received at the tun/tap virtual network interface. This is the interface connected to the VirtualHostA1. The *minivpn* then prints the received plaintext's bytes in HEX representation. It then encrypts the plaintext and prints the ciphertext. Finally, it computes the hash of the ciphertext, appends it to the ciphertext and writes it to the network.
- This part of the message exchange screenshot has been outlined in **red** to show the same data as being received at the server (GatewayB). This is shown in the next screenshot.


cs528user@cs528vm: ~/MiniVPN —...
cs528user@cs528vm: ~ — ssh mhas...
cs528user@cs528vm: ~/MiniVPN —...
cs528user@cs528vm: ~ — ssh mhas...
cs528user@cs528vm: ~/MiniVPN —...

```
Keys and IV have been exchanged. Launching the tunnel now..
Successfully connected to interface tun0
Tunnel ready...

******************************
Got a packet from the network.

------------------------------
Extracting ciphertext and message digest from the following data:
0e0aa5f40b7afc9f0dac8053426788316264c3824f71274fc30d519c62830f89f9e69e7e54e5299960724b786ef2ae678da05a8077bc40e629d7f1a13eb1ff6bb40280735bd99f7db57ab4d2928e1a896e4971206dd21bb776042
238a9eaa2d1f060d7ec963512ee888ba036a1e5fdf92fd0a7a275dd892ddc6483245425853e
The extracted ciphertext is:
0e0aa5f40b7afc9f0dac8053426788316264c3824f71274fc30d519c62830f89f9e69e7e54e5299960724b786ef2ae678da05a8077bc40e629d7f1a13eb1ff6bb40280735bd99f7db57ab4d2928e1a896e4971206dd21bb776042
238a9eaa2d1
The extracted hash value is:
f060d7ec963512ee888ba036a1e5fdf92fd0a7a275dd892ddc6483245425853e
------------------------------
Received the following hash for the received ciphertext:
f060d7ec963512ee888ba036a1e5fdf92fd0a7a275dd892ddc6483245425853e
Computed the following hash for the received ciphertext:
f060d7ec963512ee888ba036a1e5fdf92fd0a7a275dd892ddc6483245425853e
Decrypting the received ciphertext...
The received ciphertext is:
0e0aa5f40b7afc9f0dac8053426788316264c3824f71274fc30d519c62830f89f9e69e7e54e5299960724b786ef2ae678da05a8077bc40e629d7f1a13eb1ff6bb40280735bd99f7db57ab4d2928e1a896e4971206dd21bb776042
238a9eaa2d1
The corresponding plaintext is:
45000054000040003f0108820a000aac0a00147c08004743131a000125fdc65aba460c0008090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f3031323334353637
[NET2TAP 1: Read 128 bytes from the network]
[NET2TAP 1: Written 84 bytes to the tap interface]

******************************

******************************
Got a packet from the tun/tap virtual interface.
Printing the plaintext:
450000544f4700003f01f93a0a00147c0a000aac00004f43131a000125fdc65aba460c0008090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f3031323334353637
Printing the Ciphertext:
23f630a0fdb3572669b112e9c1d5e70a21447843bf6ed87faca5d37be633e689fcab71ea5876cbfbfa8b3989af67b0d2eab773e7cc409150a660370d031798ec35397ee1506fe1ebd0ee05c032e32fd4d783151515de7a8e85e04
ad250ff551f
Printing the combined message to be sent over the wire in the format CIPHERTEXT||HASH(CIPHERTEXT)
23f630a0fdb3572669b112e9c1d5e70a21447843bf6ed87faca5d37be633e689fcab71ea5876cbfbfa8b3989af67b0d2eab773e7cc409150a660370d031798ec35397ee1506fe1ebd0ee05c032e32fd4d783151515de7a8e85e04
ad250ff551f4ff5a503ac3015dfef2e50137b0505ea936d012e345469788521d78a2b4aeeba
[TAP2NET 1: Read 84 bytes from the tap interface]
[TAP2NET 1: Written 128 bytes to the network]

******************************
```

Figure 32

- The **red** outline in Figure 32 shows the same message as it was received at the GatewayB. Note that I appended the ciphertext and hash of the ciphertext and so the server received an appended message. It then extracts the ciphertext and digest and prints each of these. Next, it recomputed the hash of the ciphertext to verify that the message integrity is intact. Once verified, it writes the packet to the tun/tap interface which then sends the packet to the VirtualHostB1.
- Once VirtualHostB1 received the *ping* request from VirtualHostA1, it sends a *ping* reponse which is received GatewayB's tun/tap interface. It carries out the same steps i.e. encrypt then hash and then writes the message to the network. This is outlined in **orange** on Figure 32. The received message at VirtualHostA1 is outlined in orange in Figure 31.

Q. Algorithm used for encryption

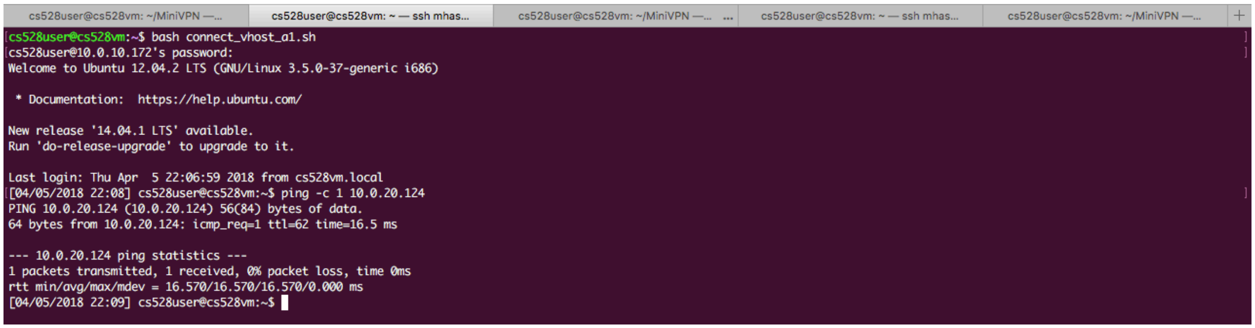
I used the AES-128-bit version with CBC for encrypting the data. This is a function call that is provided by OpenSSL. It is important to use CBC instead of EBC since the latter is insecure as we discussed in class.

Providing Integrity (via HMAC)

The following screenshots will show that the *MiniVPN* provides data integrity by computing and appending the HMAC of the data.

Notes:

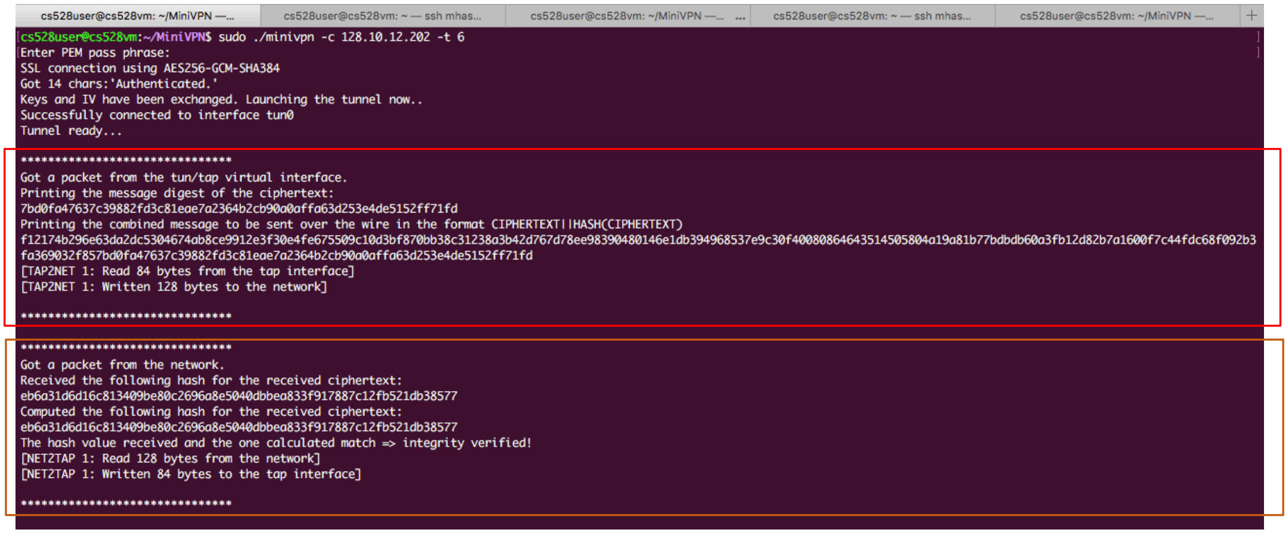
- For the purpose of the report, I am showing the traffic for one exchange of *ping* packets between VirtualHostA1 and VirtualHostB1. The same feature works for other protocols like *ssl* as well as will be shown to the TA in the demo.
- I have omitted screenshots that verify that the virtual network interface has been setup as that was shown in the previous sections.


cs528user@cs528vm: ~/MiniVPN —...
cs528user@cs528vm: ~ — ssh mhas...
cs528user@cs528vm: ~/MiniVPN —...
cs528user@cs528vm: ~ — ssh mhas...
cs528user@cs528vm: ~/MiniVPN —...

```
cs528user@cs528vm:~$ bash connect_vhost_a1.sh
cs528user@10.0.10.172's password:
Welcome to Ubuntu 12.04.2 LTS (GNU/Linux 3.5.0-37-generic i686)

 * Documentation:  https://help.ubuntu.com/

New release '14.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Thu Apr  5 22:06:59 2018 from cs528vm.local
[04/05/2018 22:08] cs528user@cs528vm:~$ ping -c 1 10.0.20.124
PING 10.0.20.124 (10.0.20.124) 56(84) bytes of data.
64 bytes from 10.0.20.124: icmp_req=1 ttl=62 time=16.5 ms

--- 10.0.20.124 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 16.570/16.570/16.570/0.000 ms
[04/05/2018 22:09] cs528user@cs528vm:~$ 
```

Figure 33

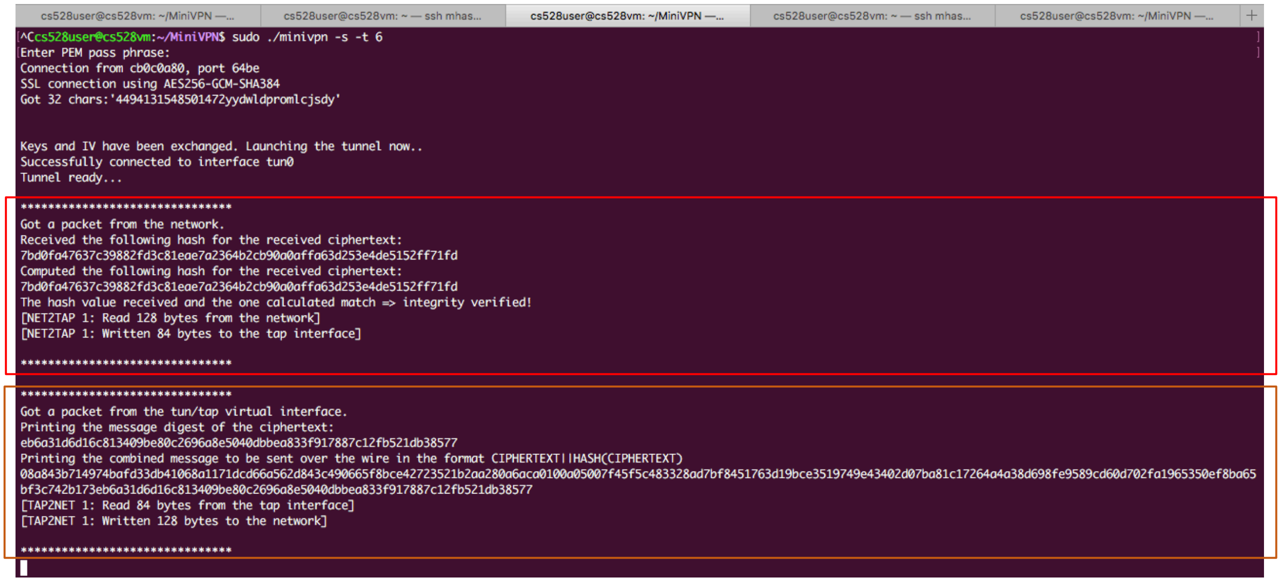
- Figure 33 shows one *ping* packet being sent from VirtualHostA1 to VirtualHostB1. The *ping* succeeded and there was 0% packet loss.


cs528user@cs528vm: ~/MiniVPN —...
cs528user@cs528vm: ~ — ssh mhas...
cs528user@cs528vm: ~/MiniVPN —...
cs528user@cs528vm: ~ — ssh mhas...
cs528user@cs528vm: ~/MiniVPN —...

```
cs528user@cs528vm:~/MiniVPN$ sudo ./minivpn -c 128.10.12.202 -t 6
Enter PEM pass phrase:
SSL connection using AES256-GCM-SHA384
Got 14 chars:'Authenticated.'
Keys and IV have been exchanged. Launching the tunnel now..
Successfully connected to interface tun0
Tunnel ready...

******************************
Got a packet from the tun/tap virtual interface.
Printing the message digest of the ciphertext:
7bd0fa47637c39882fd3c81eae7a2364b2cb90a0affa63d253e4de5152ff71fd
Printing the combined message to be sent over the wire in the format CIPHERTEXT||HASH(CIPHERTEXT)
f12174b296e63da2dc5304674ab8ce9912e3f30e4fe675509c10d3bf870bb38c31238a3b42d767d78ee98390480146e1db394968537e9c30f40080864643514505804a19a81b77bdbdb60a3fb12d82b7a1600f7c44fdc68f092b3
fa369032f857bd0fa47637c39882fd3c81eae7a2364b2cb90a0affa63d253e4de5152ff71fd
[TAP2NET 1: Read 84 bytes from the tap interface]
[TAP2NET 1: Written 128 bytes to the network]

******************************

******************************
Got a packet from the network.
Received the following hash for the received ciphertext:
eb6a31d6d16c813409be80c2696a8e5040dbbea833f917887c12fb521db38577
Computed the following hash for the received ciphertext:
eb6a31d6d16c813409be80c2696a8e5040dbbea833f917887c12fb521db38577
The hash value received and the one calculated match => integrity verified!
[NET2TAP 1: Read 128 bytes from the network]
[NET2TAP 1: Written 84 bytes to the tap interface]

******************************
```

Figure 34

- Figure 34 shows the message digest of the ciphertext corresponding to the *ping* request packet generated by VirtualHostA1 and received at GatewayA.
- The screenshot area outlined in **red** in the next screenshot shows how GatewayB extracts the HMAC and verifies the integrity of this packet.


cs528user@cs528vm: ~/MiniVPN —...
cs528user@cs528vm: ~ — ssh mhas...
cs528user@cs528vm: ~/MiniVPN —...
cs528user@cs528vm: ~ — ssh mhas...
cs528user@cs528vm: ~/MiniVPN —...

```
^Ccs528user@cs528vm:~/MiniVPN$ sudo ./minivpn -s -t 6
Enter PEM pass phrase:
Connection from cb0c0a80, port 64be
SSL connection using AES256-GCM-SHA384
Got 32 chars:'4494131548501472yydwldpromlcjsdy'


Keys and IV have been exchanged. Launching the tunnel now..
Successfully connected to interface tun0
Tunnel ready...

******************************
Got a packet from the network.
Received the following hash for the received ciphertext:
7bd0fa47637c39882fd3c81eae7a2364b2cb90a0affa63d253e4de5152ff71fd
Computed the following hash for the received ciphertext:
7bd0fa47637c39882fd3c81eae7a2364b2cb90a0affa63d253e4de5152ff71fd
The hash value received and the one calculated match => integrity verified!
[NET2TAP 1: Read 128 bytes from the network]
[NET2TAP 1: Written 84 bytes to the tap interface]

******************************
******************************
Got a packet from the tun/tap virtual interface.
Printing the message digest of the ciphertext:
eb6a31d6d16c813409be80c2696a8e5040dbbea833f917887c12fb521db38577
Printing the combined message to be sent over the wire in the format CIPHERTEXT||HASH(CIPHERTEXT)
08a843b714974bafd33db41068a1171dcd66a562d843c490665f8bce42723521b2aa280a6aca0100a05007f45f5c483328ad7bf8451763d19bce3519749e43402d07ba81c17264a4a38d698fe9589cd60d702fa1965350ef8ba65
bf3c742b173eb6a31d6d16c813409be80c2696a8e5040dbbea833f917887c12fb521db38577
[TAP2NET 1: Read 84 bytes from the tap interface]
[TAP2NET 1: Written 128 bytes to the network]

******************************
```

Figure 35

- The area outlines in **red** in Figure 35 shows the GatewayB receiving the message digest of the *ping* request sent by VirtualHostA1. It recomputes the HMAC of the data and verifies the integrity. It only writes to the tun/tap interface once the message integrity has been verified.
- The same chain of events occurs when the VirtualHostB1 sends the *ping* reply to VirtualHostA1. The area highlighted in **orange** in Figures 35 and 34 show this

Q. Algorithm used for hashing and why

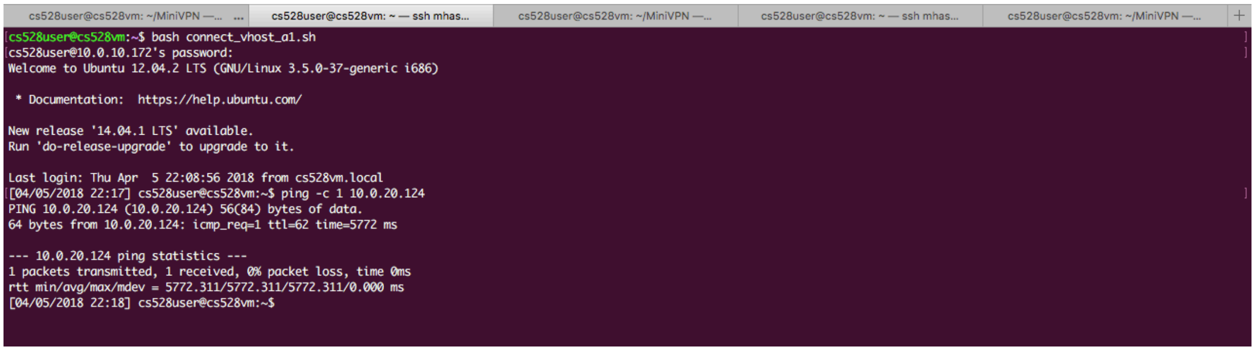
I used the sha256 algorithm for hashing. Although there were options like sha1 and sha512, I used sha256 mainly because sha1 is known to be broken and sha512 would just be a performance overhead. Sha256 is known to be secure for now and so, I used that. Sha256 was also mentioned in the lab handout.

Q. Security Flaws and Protection against Man in the Middle

There are some security flaws possible that we tried catering to in the implementation of the MiniVPN. For example, if the IV and the KEY were not being sent securely over SSL, it would be possible for an on-path adversary to sniff these two values and decrypt all traffic being sent which would effectively render the confidentiality and integrity of the system to zero. To cater to this, we transfer the IV and KEY over an SSL connection. Similarly, it is very important to always generate a random IV and not repeat it again. As we saw in assignment 1, it is possible to break a CBC-like encryption if the same IV is used again and again. To address this issue, our VPN generates a random IV every time a new session is created between the gateways. Another possible vulnerability that could be present is that if the client and server did not first authenticate each other using the certificates, it would have been possible for a man-in-the-middle attacker to intercept the initial messages for exchanging the KEY and IV and send his own pair of KEY' and IV' to the other party. The other party would accept this since there is no authentication being done and the man in the middle would be able to listen to all the communication between the two hosts. We address this problem by having both parties first authenticate themselves using PKI and digital certificates. Finally, note that we used the Encrypt-then-Authenticate scheme of sending the data, which, as we discussed in class provides security as compared to other schemes like Authenticate-then-Encrypt. This also prevents against a possible vulnerability.

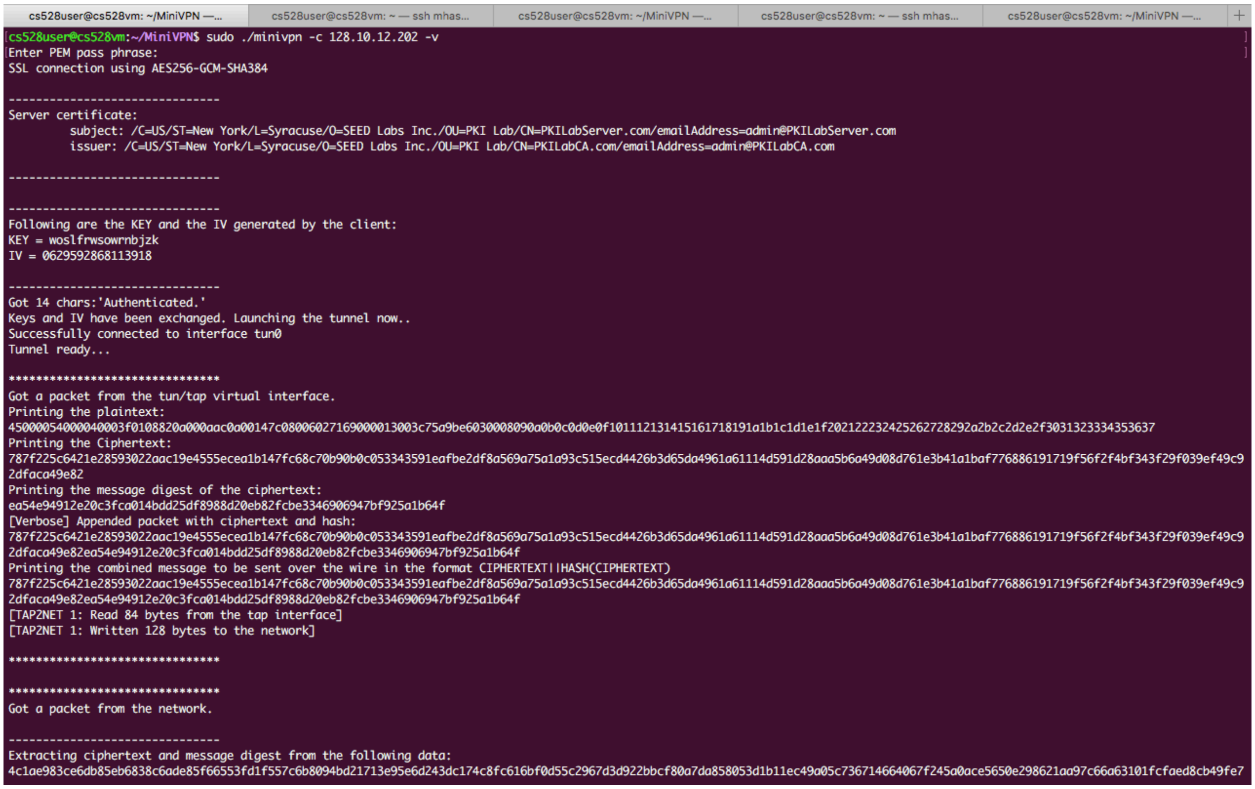
Appendix

Following is a complete flow of sending one *ping* request from VirtualHostA1 and receiving the reply from VirtualHostB1 with verbose printouts of the different steps.


cs528user@cs528vm: ~/MiniVPN —...
cs528user@cs528vm: ~ — ssh mhas...
cs528user@cs528vm: ~/MiniVPN —...
cs528user@cs528vm: ~ — ssh mhas...
cs528user@cs528vm: ~/MiniVPN —...

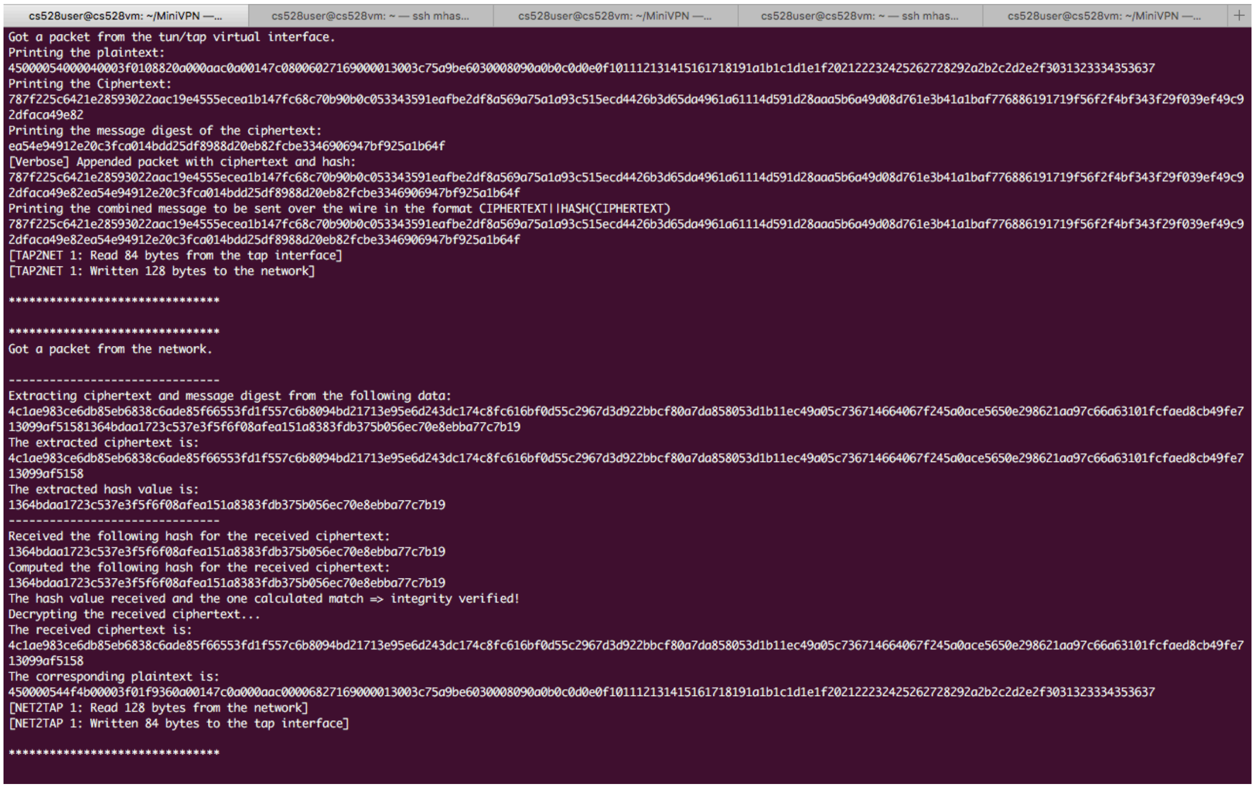
```
cs528user@cs528vm:~$ bash connect_vhost_a1.sh
cs528user@10.0.10.172's password:
Welcome to Ubuntu 12.04.2 LTS (GNU/Linux 3.5.0-37-generic i686)

 * Documentation:  https://help.ubuntu.com/

New release '14.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Thu Apr  5 22:08:56 2018 from cs528vm.local
[04/05/2018 22:17] cs528user@cs528vm:~$ ping -c 1 10.0.20.124
PING 10.0.20.124 (10.0.20.124) 56(84) bytes of data.
64 bytes from 10.0.20.124: icmp_req=1 ttl=62 time=5772 ms

--- 10.0.20.124 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 5772.311/5772.311/5772.311/0.000 ms
[04/05/2018 22:18] cs528user@cs528vm:~$
```

Figure 36: VirtualHostA1 pinging VirtualHostB1


cs528user@cs528vm: ~/MiniVPN —...
cs528user@cs528vm: ~ — ssh mhas...
cs528user@cs528vm: ~/MiniVPN —...
cs528user@cs528vm: ~ — ssh mhas...
cs528user@cs528vm: ~/MiniVPN —...

```
cs528user@cs528vm:~/MiniVPN$ sudo ./minivpn -c 128.10.12.202 -v
Enter PEM pass phrase:
SSL connection using AES256-GCM-SHA384

------------------------------
Server certificate:
         subject: /C=US/ST=New York/L=Syracuse/O=SEED Labs Inc./OU=PKI Lab/CN=PKILabServer.com/emailAddress=admin@PKILabServer.com
         issuer: /C=US/ST=New York/L=Syracuse/O=SEED Labs Inc./OU=PKI Lab/CN=PKILabCA.com/emailAddress=admin@PKILabCA.com

------------------------------

------------------------------
Following are the KEY and the IV generated by the client:
KEY = woslfrwsowrnbjzk
IV = 0629592868113918

------------------------------
Got 14 chars:'Authenticated.'
Keys and IV have been exchanged. Launching the tunnel now..
Successfully connected to interface tun0
Tunnel ready...

******************************
Got a packet from the tun/tap virtual interface.
Printing the plaintext:
45000054000040003f0108820a000aac0a00147c08006027169000013003c75a9be6030008090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f3031323334353637
Printing the Ciphertext:
787f225c6421e28593022aac19e4555ecea1b147fc68c70b90b0c053343591eafbe2df8a569a75a1a93c515ecd4426b3d65da4961a61114d591d28aaa5b6a49d08d761e3b41a1baf776886191719f56f2f4bf343f29f039ef49c9
2dfaca49e82
Printing the message digest of the ciphertext:
ea54e94912e20c3fca014bdd25df8988d20eb82fcbe3346906947bf925a1b64f
[Verbose] Appended packet with ciphertext and hash:
787f225c6421e28593022aac19e4555ecea1b147fc68c70b90b0c053343591eafbe2df8a569a75a1a93c515ecd4426b3d65da4961a61114d591d28aaa5b6a49d08d761e3b41a1baf776886191719f56f2f4bf343f29f039ef49c9
2dfaca49e82ea54e94912e20c3fca014bdd25df8988d20eb82fcbe3346906947bf925a1b64f
Printing the combined message to be sent over the wire in the format CIPHERTEXT||HASH(CIPHERTEXT)
787f225c6421e28593022aac19e4555ecea1b147fc68c70b90b0c053343591eafbe2df8a569a75a1a93c515ecd4426b3d65da4961a61114d591d28aaa5b6a49d08d761e3b41a1baf776886191719f56f2f4bf343f29f039ef49c9
2dfaca49e82ea54e94912e20c3fca014bdd25df8988d20eb82fcbe3346906947bf925a1b64f
[TAP2NET 1: Read 84 bytes from the tap interface]
[TAP2NET 1: Written 128 bytes to the network]

******************************

******************************
Got a packet from the network.

------------------------------
Extracting ciphertext and message digest from the following data:
4c1ae983ce6db85eb6838c6ade85f66553fd1f557c6b8094bd21713e95e6d243dc174c8fc616bf0d55c2967d3d922bbcf80a7da858053d1b11ec49a05c736714664067f245a0ace5650e298621aa97c66a63101fcfaed8cb49fe7
```

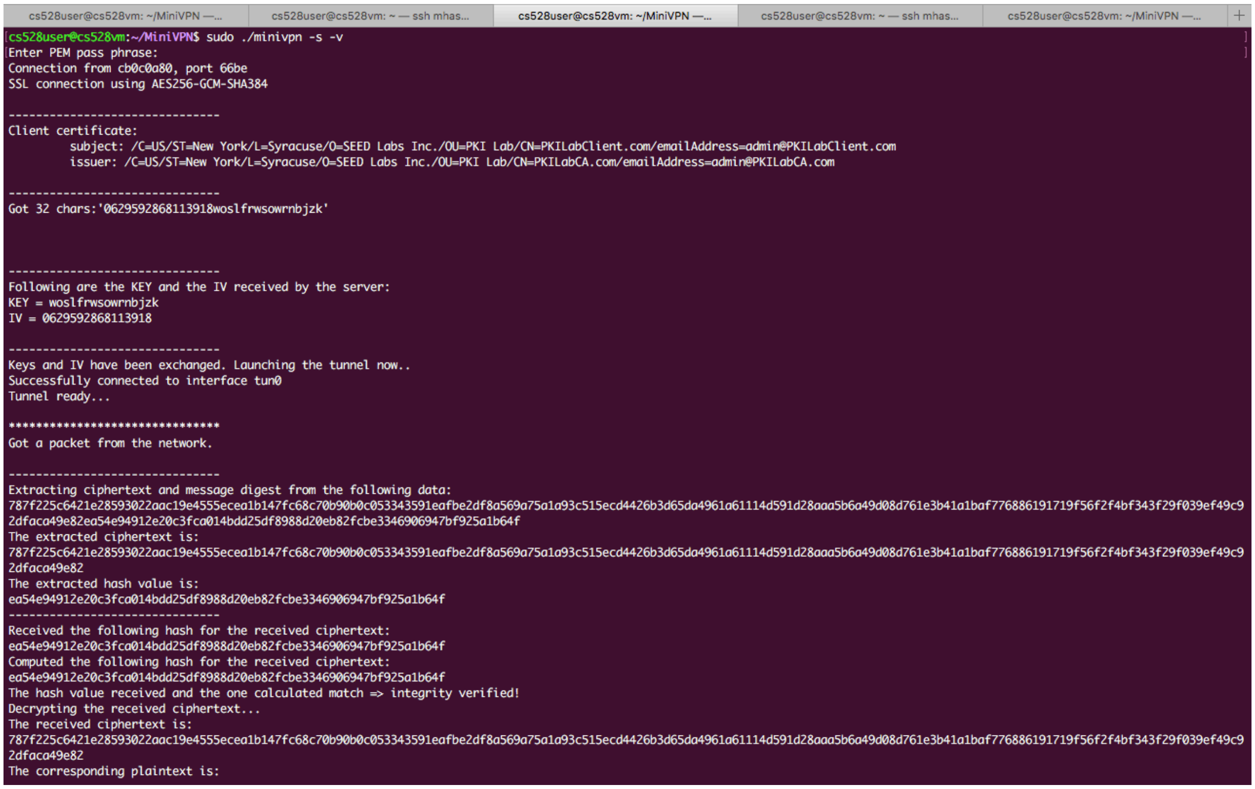
Figure 37: Chain of events at the client (GatewayA)


cs528user@cs528vm: ~/MiniVPN —...
cs528user@cs528vm: ~ — ssh mhas...
cs528user@cs528vm: ~/MiniVPN —...
cs528user@cs528vm: ~ — ssh mhas...
cs528user@cs528vm: ~/MiniVPN —...
+

```
Got a packet from the tun/tap virtual interface.
Printing the plaintext:
450000540000400003f0108820a000aac0a00147c0800602716900001300 3c75a9be6030008090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f3031323334353637
```

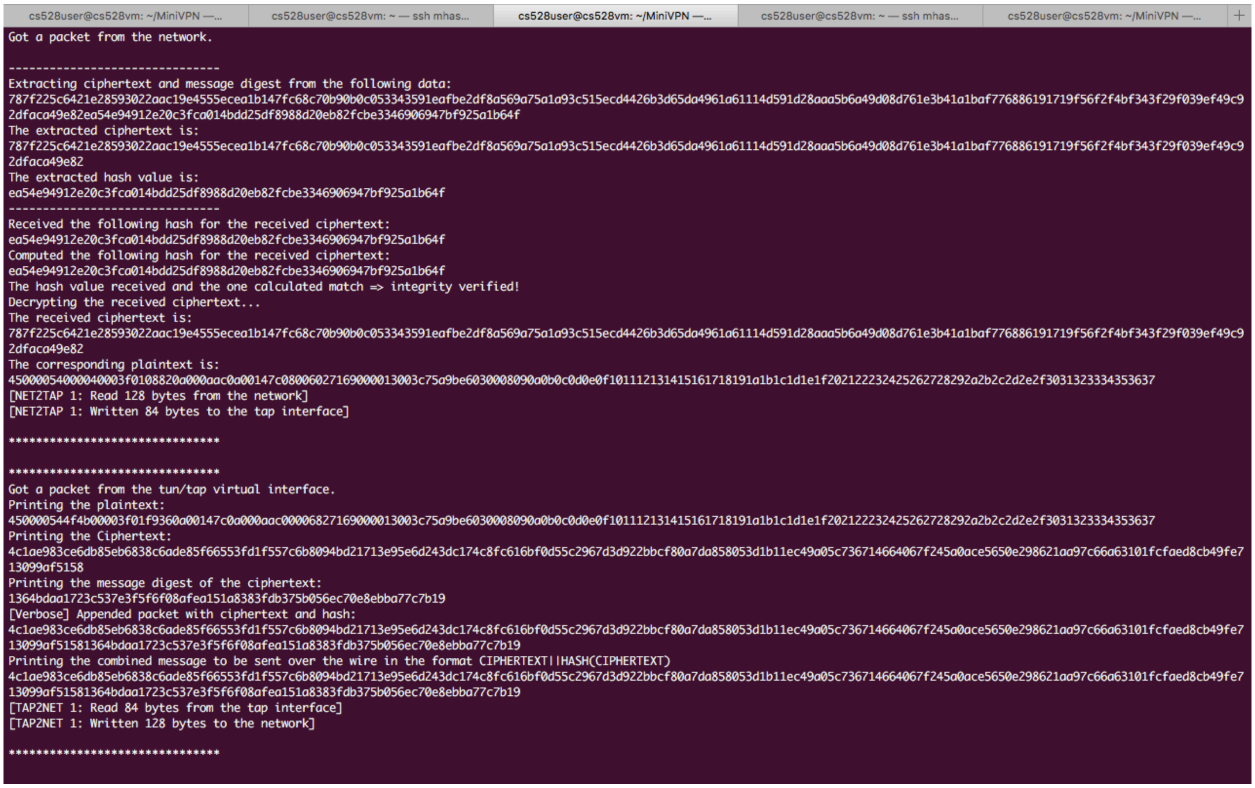
```
Got a packet from the tun/tap virtual interface.
Printing the plaintext:
45000054000040003f0108820a000aac0a00147c08000602716900001 3003c75a9be6030008090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f3031323334353637
```

Figure 38: Chain of events at the client (GatewayA)


cs528user@cs528vm: ~/MiniVPN —...
cs528user@cs528vm: ~ — ssh mhas...
cs528user@cs528vm: ~/MiniVPN —...
cs528user@cs528vm: ~ — ssh mhas...
cs528user@cs528vm: ~/MiniVPN —...
+

```
cs528user@cs528vm:~/MiniVPN$ sudo ./minivpn -s -v
Enter PEM pass phrase:
Connection from cb0c0a80, port 66be
SSL connection using AES256-GCM-SHA384

------------------------------
Client certificate:
        subject: /C=US/ST=New York/L=Syracuse/O=SEED Labs Inc./OU=PKI Lab/CN=PKILabClient.com/emailAddress=admin@PKILabClient.com
        issuer: /C=US/ST=New York/L=Syracuse/O=SEED Labs Inc./OU=PKI Lab/CN=PKILabCA.com/emailAddress=admin@PKILabCA.com

------------------------------
Got 32 chars:'0629592868113918woslfrwsowrnbjzk'



------------------------------
Following are the KEY and the IV received by the server:
KEY = woslfrwsowrnbjzk
IV = 0629592868113918

------------------------------
Keys and IV have been exchanged. Launching the tunnel now..
Successfully connected to interface tun0
Tunnel ready...

******************************
Got a packet from the network.

------------------------------
Extracting ciphertext and message digest from the following data:
787f225c6421e28593022aac19e4555ecea1b147fc68c70b90b0c053343591eafbe2df8a569a75a1a93c515ecd4426b3d65da4961a61114d591d28aaa5b6a49d08d761e3b41a1baf776886191719f56f2f4bf343f29f039ef49c9
2dfaca49e82ea54e94912e20c3fca014bdd25df8988d20eb82fcbe3346906947bf925a1b64f
The extracted ciphertext is:
787f225c6421e28593022aac19e4555ecea1b147fc68c70b90b0c053343591eafbe2df8a569a75a1a93c515ecd4426b3d65da4961a61114d591d28aaa5b6a49d08d761e3b41a1baf776886191719f56f2f4bf343f29f039ef49c9
2dfaca49e82
The extracted hash value is:
ea54e94912e20c3fca014bdd25df8988d20eb82fcbe3346906947bf925a1b64f
------------------------------
Received the following hash for the received ciphertext:
ea54e94912e20c3fca014bdd25df8988d20eb82fcbe3346906947bf925a1b64f
Computed the following hash for the received ciphertext:
ea54e94912e20c3fca014bdd25df8988d20eb82fcbe3346906947bf925a1b64f
The hash value received and the one calculated match => integrity verified!
Decrypting the received ciphertext...
The received ciphertext is:
787f225c6421e28593022aac19e4555ecea1b147fc68c70b90b0c053343591eafbe2df8a569a75a1a93c515ecd4426b3d65da4961a61114d591d28aaa5b6a49d08d761e3b41a1baf776886191719f56f2f4bf343f29f039ef49c9
2dfaca49e82
The corresponding plaintext is:
```

Figure 39: Chain of events at the server (GatewayB)


cs528user@cs528vm: ~/MiniVPN —...
cs528user@cs528vm: ~ — ssh mhas...
cs528user@cs528vm: ~/MiniVPN —...
cs528user@cs528vm: ~ — ssh mhas...
cs528user@cs528vm: ~/MiniVPN —...

```
Got a packet from the network.

------------------------------
Extracting ciphertext and message digest from the following data:
787f225c6421e28593022aac19e4555ecea1b147fc68c70b90b0c053343591eafbe2df8a569a75a1a93c515ecd4426b3d65da4961a61114d591d28aaa5b6a49d08d761e3b41a1baf776886191719f56f2f4bf343f29f039ef49c9
2dfaca49e82ea54e94912e20c3fca014bdd25df8988d20eb82fcbe3346906947bf925a1b64f
The extracted ciphertext is:
787f225c6421e28593022aac19e4555ecea1b147fc68c70b90b0c053343591eafbe2df8a569a75a1a93c515ecd4426b3d65da4961a61114d591d28aaa5b6a49d08d761e3b41a1baf776886191719f56f2f4bf343f29f039ef49c9
2dfaca49e82
The extracted hash value is:
ea54e94912e20c3fca014bdd25df8988d20eb82fcbe3346906947bf925a1b64f
------------------------------
Received the following hash for the received ciphertext:
ea54e94912e20c3fca014bdd25df8988d20eb82fcbe3346906947bf925a1b64f
Computed the following hash for the received ciphertext:
ea54e94912e20c3fca014bdd25df8988d20eb82fcbe3346906947bf925a1b64f
The hash value received and the one calculated match => integrity verified!
Decrypting the received ciphertext...
The received ciphertext is:
787f225c6421e28593022aac19e4555ecea1b147fc68c70b90b0c053343591eafbe2df8a569a75a1a93c515ecd4426b3d65da4961a61114d591d28aaa5b6a49d08d761e3b41a1baf776886191719f56f2f4bf343f29f039ef49c9
2dfaca49e82
The corresponding plaintext is:
45000054000040003f0108820a000aac0a00147c08006027169000013003c75a9be6030008090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f3031323334353637
[NET2TAP 1: Read 128 bytes from the network]
[NET2TAP 1: Written 84 bytes to the tap interface]

******************************

******************************
Got a packet from the tun/tap virtual interface.
Printing the plaintext:
450000544f4b00003f01f9360a00147c0a000aac000068271690000013003c75a9be6030008090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f3031323334353637
Printing the Ciphertext:
4c1ae983ce6db85eb6838c6ade85f66553fd1f557c6b8094bd21713e95e6d243dc174c8fc616bf0d55c2967d3d922bbcf80a7da858053d1b11ec49a05c736714664067f245a0ace5650e298621aa97c66a63101fcfaed8cb49fe7
13099af5158
Printing the message digest of the ciphertext:
1364bdaa1723c537e3f5f6f08afea151a8383fdb375b056ec70e8ebba77c7b19
[Verbose] Appended packet with ciphertext and hash:
4c1ae983ce6db85eb6838c6ade85f66553fd1f557c6b8094bd21713e95e6d243dc174c8fc616bf0d55c2967d3d922bbcf80a7da858053d1b11ec49a05c736714664067f245a0ace5650e298621aa97c66a63101fcfaed8cb49fe7
13099af51581364bdaa1723c537e3f5f6f08afea151a8383fdb375b056ec70e8ebba77c7b19
Printing the combined message to be sent over the wire in the format CIPHERTEXT||HASH(CIPHERTEXT)
4c1ae983ce6db85eb6838c6ade85f66553fd1f557c6b8094bd21713e95e6d243dc174c8fc616bf0d55c2967d3d922bbcf80a7da858053d1b11ec49a05c736714664067f245a0ace5650e298621aa97c66a63101fcfaed8cb49fe7
13099af51581364bdaa1723c537e3f5f6f08afea151a8383fdb375b056ec70e8ebba77c7b19
[TAP2NET 1: Read 84 bytes from the tap interface]
[TAP2NET 1: Written 128 bytes to the network]

******************************
```

Figure 40: Chain of events at the server (GatewayB)