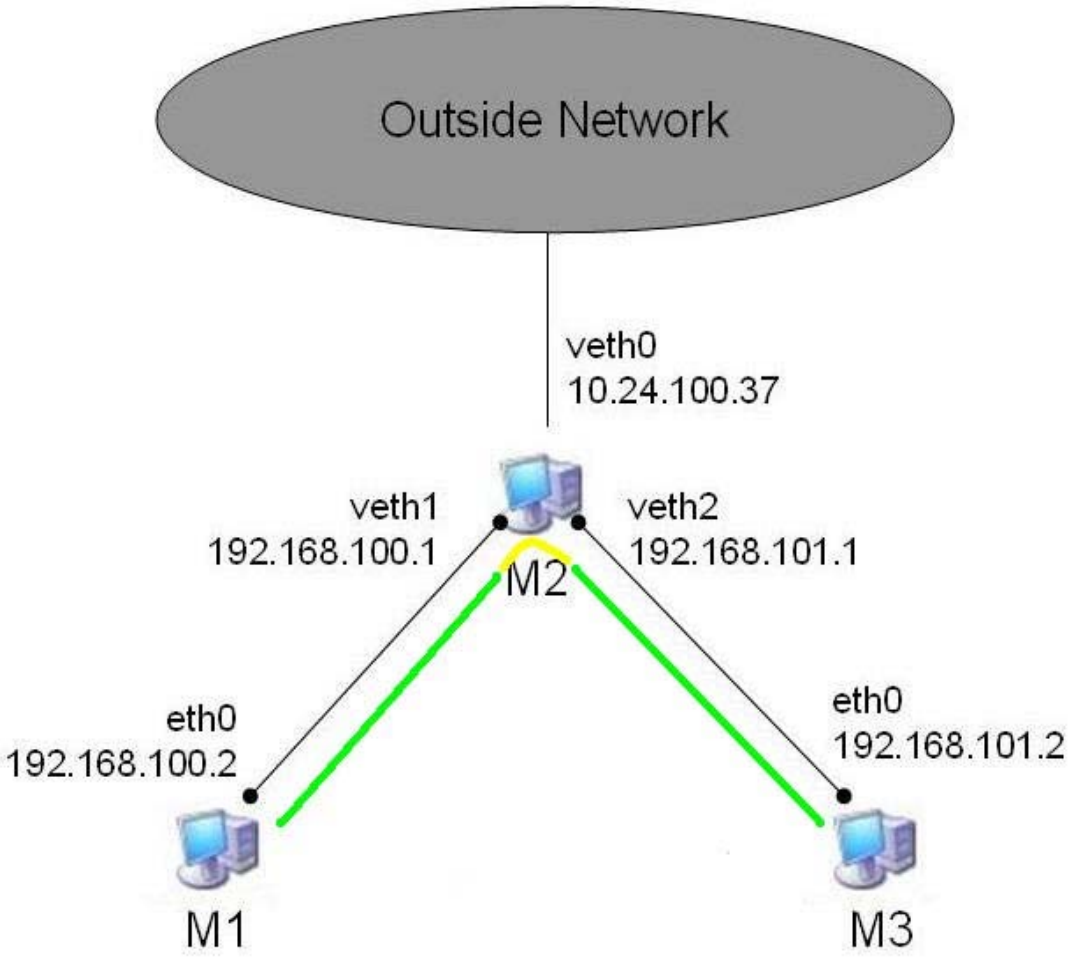
IPSEC LAB

Part 1:IPSec connection with Manual Keying in the same subnet (Transport mode)

The goal of this part of the lab is to establish an IPSec connection between M1(192.168.100.2) and M3 (192.168.101.2) within the same (inner) subnet. The topology of the network is shown below.



Setkey is a small utility used to read and write to the security policy. The configuration for each of the machines requires creating and setkey.conf file. This configuration file contains the Security Association Database (SAD) and the Security Policy Database (SPD) entries.

1. Steps taken to configure the hosts:

Below is the *setkey.conf* file for M1. The *setkey.conf* file for M3 is identical, except for the *security policies* section, which had "–P out" instead of "–P in" and "–P in" instead of "–P out" indicating the change in direction for the policies.

```
# Configuration for 192.168.100.2 (M1)                          ------>   1

# Flush the SAD and SPD                                         ------>   2

        flush;                                                  ------>   3
        spdflush;                                               ------>   4

# ESP SAs using 192 bit long keys (168 + 24 parity)             ------>   5

        add 192.168.100.2 192.168.101.2 esp 0x201 -E 3des-cbc   ------>   6
        0x7aeaca3f87d060a12f4a4487d5a5c3355920fae69a96c831       ------>   7
        -A hmac-md5 0xc0291ff014dccdd03874d9e8e4cdf3e6;         ------>   8

        add 192.168.101.2 192.168.100.2 esp 0x301 -E 3des-cbc   ------>   9
        0xf6ddb555acfd9d77b03ea3843f2653255afe8eb5573965df       ------>   10
        -A hmac-md5 0x96358c90783bbfa3d7b196ceabe0536b;         ------>   11

# Security policies                                             ------>   12

        spdadd 192.168.100.2 192.168.101.2 any -P out ipsec     ------>   13
        esp/transport//require;                                 ------>   14
        spdadd 192.168.101.2 192.168.100.2 any -P in ipsec      ------>   15
        esp/transport//require;                                 ------>   16
```

setkey.conf file for M1 (192.168.100.2)

The setkey file has two blocks. The first part deals with configuring the IPSec security associations while the second part is used for configuring the security policies.

We first flushed the previous configuration settings (if any) and configured the Security Association Database parameters as follows.

We configured IPSec to use the ESP protocol to set up the SA and we chose Triple DES in Cipher Block Chaining mode for encryption. We also set the security parameter index (SPI) for this connection and this setting is shown in Line 6 of the configuration file. Line 7 contains the associated key for it.

Similarly, for integrity and authentication, we used HMAC as the protocol of choice with MD5 as its underlying hash function. This and the associated key is displayed in Line 8 of the configuration file.

Lines 9-11 contain the required configuration settings for the SA in the other direction. The encryption and authentication keys for this SA are different from those used for the previous SA. Observe that we have 4 different keys, two keys for encryption and the other two for integrity.

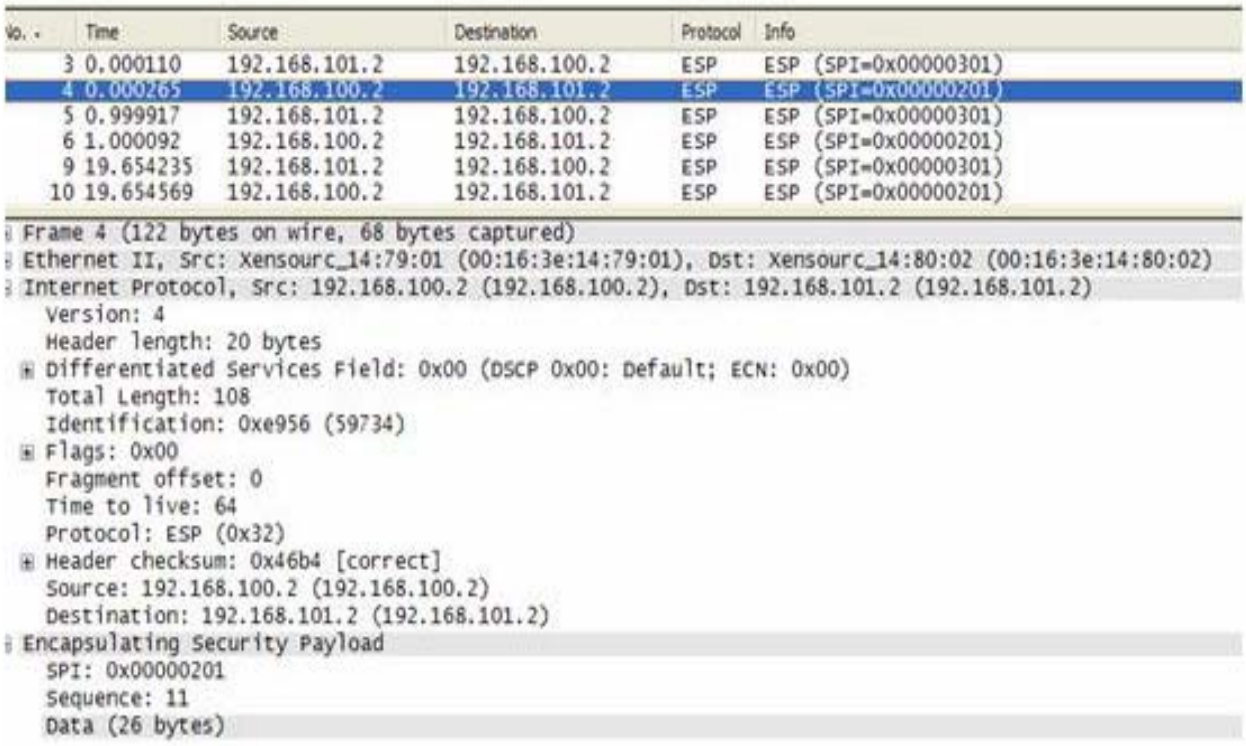
We configured the Security Policies in the following way

Using the keyword spdadd we specified the two end hosts which would be taking part in the IPSec connection and gave the SA and direction (OUT since the configuration file is on M1). Also we specified that the SA should be set up using the transport mode and the ESP protocol. This is shown in Lines 13 and 14. Lines 15 and 16 specify the other direction (IN).

2. IPSec provides the following services in this example:

- Confidentiality, since we are using the ESP protocol which encrypts the IP payload unlike the AH protocol. We set the encryption key in the setkey.conf file
- Integrity and Authentication, since we used the HMAC protocol. A hash of certain fields is created and verified at the other end . If they do not match then the message is discarded.
- The cryptographic protocols that are being used are described in the configuration file.

Once the setkey.conf file is ready, we used the setkey -f /etc/setkey.conf command to load the SAs and SPs into the databases . All packets sent between M1 and M3 will henceforth be IPSec protected. We initiated a connection from M1 (192.168.100.2) to M3 (192.168.101.2) using the PING command. The following is a screenshot of one of the captured packets.



No.	Time	Source	Destination	Protocol	Info
3	0.000110	192.168.101.2	192.168.100.2	ESP	ESP (SPI=0x00000301)
4	0.000265	192.168.100.2	192.168.101.2	ESP	ESP (SPI=0x00000201)
5	0.999917	192.168.101.2	192.168.100.2	ESP	ESP (SPI=0x00000301)
6	1.000092	192.168.100.2	192.168.101.2	ESP	ESP (SPI=0x00000201)
9	19.654235	192.168.101.2	192.168.100.2	ESP	ESP (SPI=0x00000301)
10	19.654569	192.168.100.2	192.168.101.2	ESP	ESP (SPI=0x00000201)

```
Frame 4 (122 bytes on wire, 68 bytes captured)
Ethernet II, Src: Xensourc_14:79:01 (00:16:3e:14:79:01), Dst: Xensourc_14:80:02 (00:16:3e:14:80:02)
Internet Protocol, Src: 192.168.100.2 (192.168.100.2), Dst: 192.168.101.2 (192.168.101.2)
    Version: 4
    Header length: 20 bytes
  ⊞ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
    Total Length: 108
    Identification: 0xe956 (59734)
  ⊞ Flags: 0x00
    Fragment offset: 0
    Time to live: 64
    Protocol: ESP (0x32)
  ⊞ Header checksum: 0x46b4 [correct]
    Source: 192.168.100.2 (192.168.100.2)
    Destination: 192.168.101.2 (192.168.101.2)
Encapsulating Security Payload
    SPI: 0x00000201
    Sequence: 11
    Data (26 bytes)
```

3. No,the IP header is not encrypted. This is because we are using the Transport mode which runs on top of IP. The length of the IP header is 20 bytes. All of the IP payload is encrypted while the IP header itself is left untouched.

4. The protocol number for ESP is 0x32 in hex or 50 in decimal

5. No. Since the entire IP payload is encrypted, there is no way in which we can determine if the higher layer protocol is TCP or UDP or ICMP.

6. The SPI for the SA from M1 to M3 is 0x00000201 while that for the SA from M3 to M1 is 0x00000301.

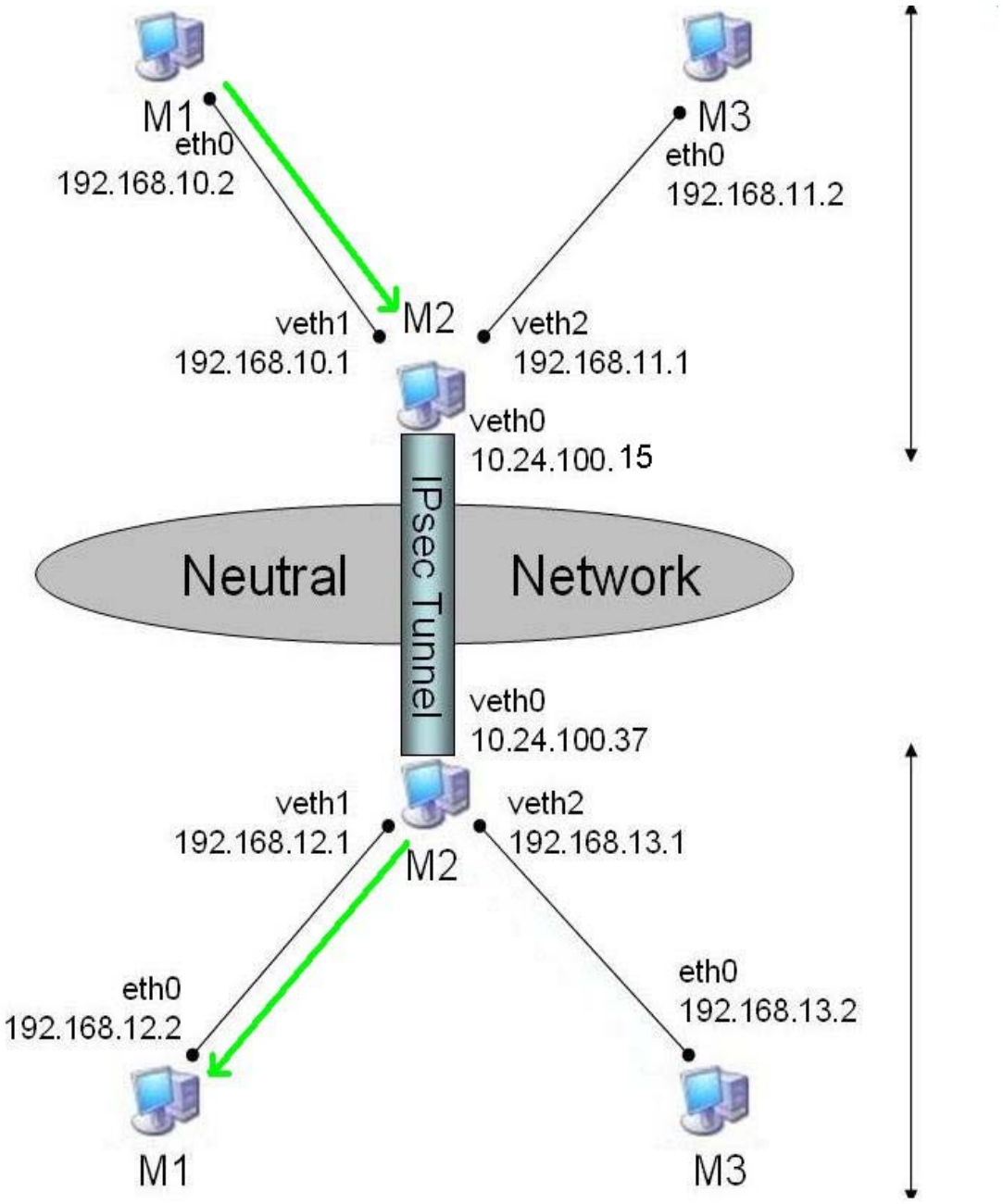
7. The sequence numbers are incremented by 1 in each of the SAs independent of each other.

8. Manual keying is useful if the encryption and authentication keys can be securely installed in the communicating hosts. Hence, this is feasible only in small networks and fails completely in the case of two unknown hosts requiring to communicate with each other.

Part 2: IPSec connection with Manual Keying between hosts in different subnets (Tunnel Mode):

In this part we have established an IPSec SA in the tunnel mode between 10.24.100.15 and 10.24.100.37 gateways. NAT was overcome using the given instructions. After the initial setup we sent packets between 192.168.10.2 and 192.168.12.2 which were behind the 10.24.100.15 and 10.24.100.37 gateways respectively.

Tunnel mode guarantees security for information flowing within the tunnel which was encrypted using the ESP protocol. Shown below is the network topology that we used.



1. Shown below is the *setkey.conf* file that we used on 10.24.100.37 (M2). The *setkey.conf* file for M2 of our partner group is identical, except for the *security policies* section, which had "–P out" instead of "–P in" and "–P in" instead of "–P out" indicating the change in direction for the policies.

Configuration for 10.24.100.37 (M2) ------>1

Flush the SAD and SPD ------>2

flush; ------>3

```
        spdflush;                                                    ------>4

# ESP SAs using 192 bit long keys (168 + 24 parity)                  ------>5

        add 10.24.100.15 10.24.100.37 esp 0x015ed985 -m tunnel -E 3des-cbc------>6
        0x7aeaca3f87d060a12f4a4487d5a5c3355920fae69a96c831            ------>7

        add 10.24.100.37 10.24.100.15 esp 0x04786b7e -m tunnel -E 3des-cbc------>8
        0xf6ddb555acfd9d77b03ea3843f2653255afe8eb5573965df            ------>9

# Security policies                                                  ------>10

        spdadd 192.168.12.0/24 192.168.10.0/24 any -P out ipsec      ------>11
        esp/tunnel/10.24.100.37-10.24.100.15/require;                ------>12

        spdadd 192.168.10.0/24 192.168.12.0/24 any -P in ipsec       ------>13
        esp/tunnel/10.24.100.15-10.24.100.37/require;                ------>14
```

Setkey.conf file on 10.24.100.37

The setkey file has two blocks in it and the first part deals with configuring the IPSec Security Associations while the second part is used for configuring the Security Policies. We first flushed the previous configuration settings (if any) and configured the Security Association Database parameters as follows.

We configured IPSec in the tunnel mode to use the ESP protocol to set up the SA and we chose Triple DES in Cipher Block Chaining mode for encryption. We also set the security parameter index (SPI) for this connection and this setting is shown in Line 6 of the configuration file. Line 7 contains the associated key for it.

Lines 8-9 contain the required configuration settings for the SA in the other direction. The encryption key for this SA are different from the previous SA.

We configured the Security Policies in the following way

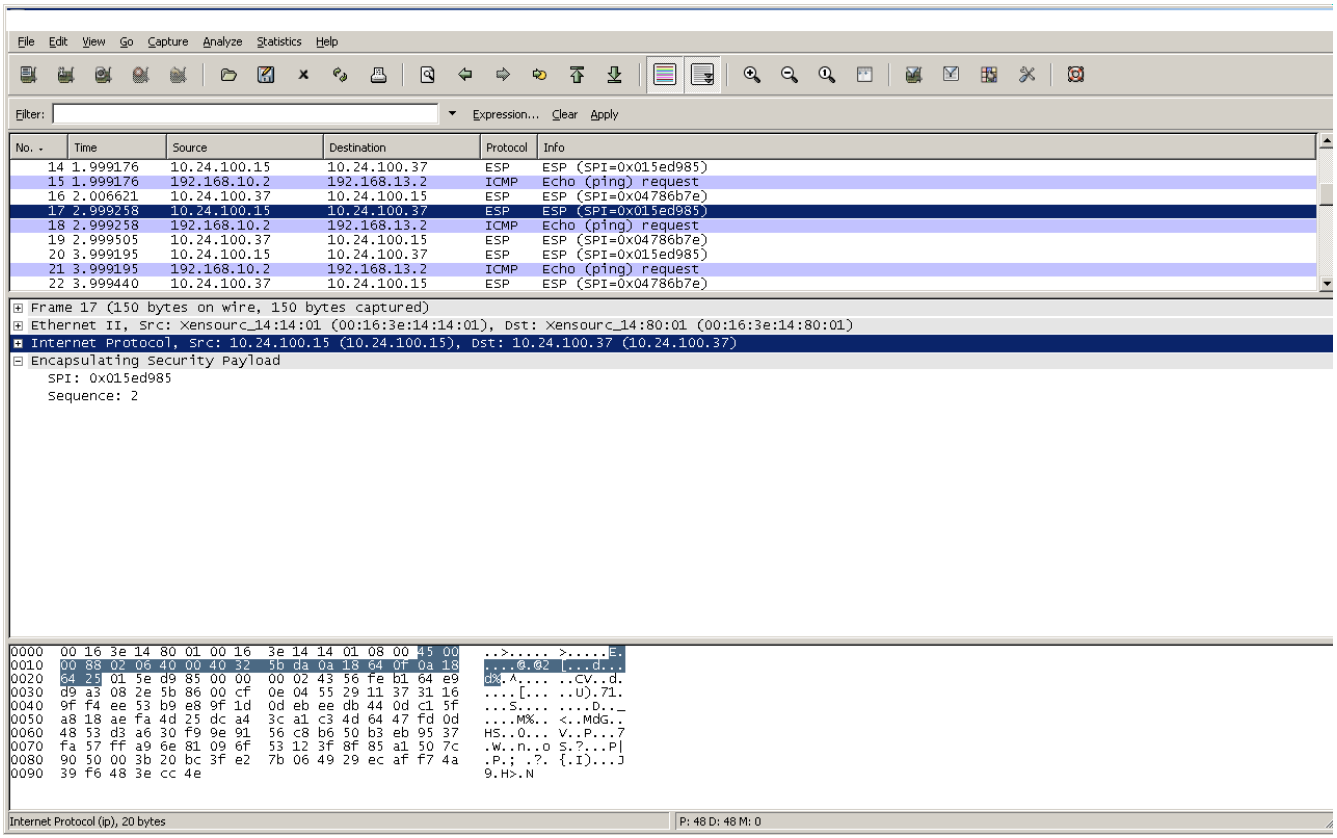
Using the keyword spdadd we specified the two end hosts which would be taking part in the IPSec connection and gave the SA and direction (OUT since the configuration file is on M1). Also we specified that the SA should be set up using the transport mode and the ESP protocol. This is shown in Lines 13 and 14. Lines 15 and 16 specify the other direction (IN).

We initiated the connection between 192.168.10.2 and 192.168.12.2 using PING.

2. IPSec provides only confidentiality in this case. We are neither using AH nor using authentication/integrity services provided by ESP. This is not a very secure way to setup an

IPSec connection. The cryptographic protocol used in this setup is 3DES-CBC with a 192 bit long key for encryption. We are using only the encryption option.

3. Since we are using the tunneling mode the original IP header is encrypted and a new IP header is created which has the end points of the IPSec tunnel as the source and destination addresses. The connection was initiated between 192.168.10.2 and 192.168.12.2, but the IP header shows 10.24.100.15 and 10.24.100.37 as the source and destinations respectively. The new IP header is 20 bytes long. The entire original IP payload is encrypted. Following is a screenshot showing the same.



4. Yes, we are able to see ESP packets in both directions but only the incoming ICMP echo request packet and not the response packet. This is because the incoming packet is decrypted at the external interface M2 (10.24.100.37) while the outgoing packet is IPSec enabled at the internal interface 192.168.12.1 and so by the time it reaches 10.24.100.37, it is already IPSec enabled.

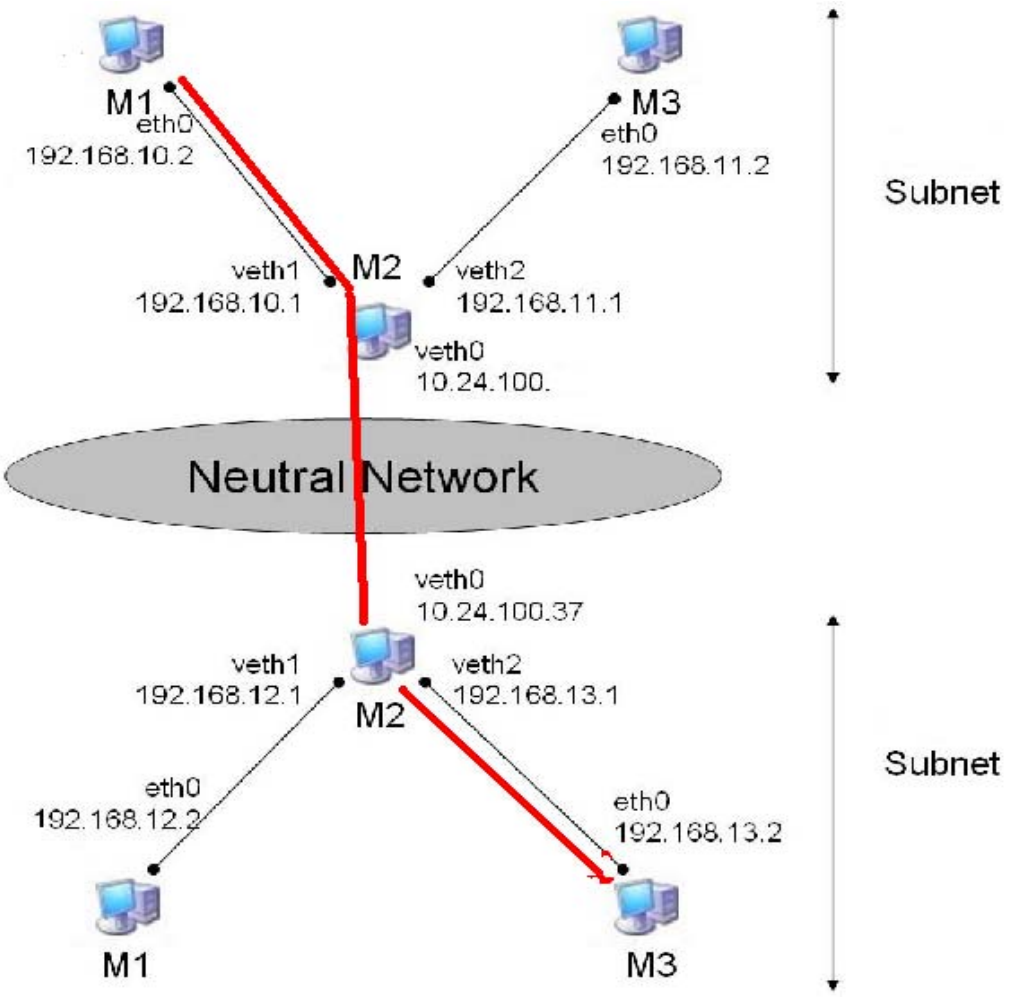
5. If in the tunnel mode ESP authentication was used , then there would be no difference in the extent of authentication provided by either of the protocols. AH authenticates both the IP header as well as IP payload in both the transport and tunnel mode. In the tunnel mode, ESP authentication would cover the original IP header as well as the IP payload.

Part 3: IPSec connection with Manual Keying between hosts in different subnets (Transport Mode):

In this part we have established an IPSec SA in the transport mode between 192.168.10.2 and 192.168.13.2 which are behind the 10.24.100.15 and 10.24.100.37 gateways. NAT was overcome using the given instructions.

Transport mode guarentees end to end security since all the information between the hosts 192.168.10.2 and 192.168.13.2 was encrypted using the IPSec ESP protocol.

Shown below is the network topology over which this was done.



2. Shown below is the *setkey.conf* file that we used on 192.168.13.2 (M3). The *setkey.conf* file for M1 of our partner group is identical, except for the *security policies* section, which had "–P out" instead of "–P in" and "–P in" instead of "–P out" indicating the change in direction for the policies.

#Setkey configuration for 192.168.10.2: ------>1

flush the SAD and SPD ------>2

flush; ------>3

```
        spdflush;                                                    ------>4

#ESP SAs usin g 192 bit long keys                                    ------>5

        add 192.168.13.2 192.168.10.2 esp 0x201 -E 3des-cbc          ------>6
        0x7aeaca3f87d060a12f4a4487d5a5c3355920fae69a96c831;          ------>7

        add 192.168.10.2 192.168.13.2 esp 0x301 -E 3des-cbc          ------>8
        0xf6ddb555acfd9d77b03ea3843f26532558fe8eb5573965df;          ------>9

#Security Policies                                                   ------>10

        spdadd 192.168.10.2 192.168.13.2 any -P in ipsec             ------>11
        esp/transport//require;                                      ------>12

        spdadd 192.168.13.2 192.168.10.2 any -P out ipsec            ------>13
        esp/transport//require;                                      ------>14
```

Setkey.conf file on 192.168.13.2

The setkey file has two blocks in it and the first part deals with configuring the IPSec security associations while the second part is used for configuring the Security Policies. We first flushed the previous configuration settings(if any) and configured the Security Association Database parameters as follows.

We configured IPSec to use the ESP protocol to set up the SA. We chose Triple DES in Cipher Block Chaining mode for encryption. We also set the security parameter index (SPI) for this connection and this setting is shown in Line 6 of the configuration file. Line 7 contains the associated key for it.

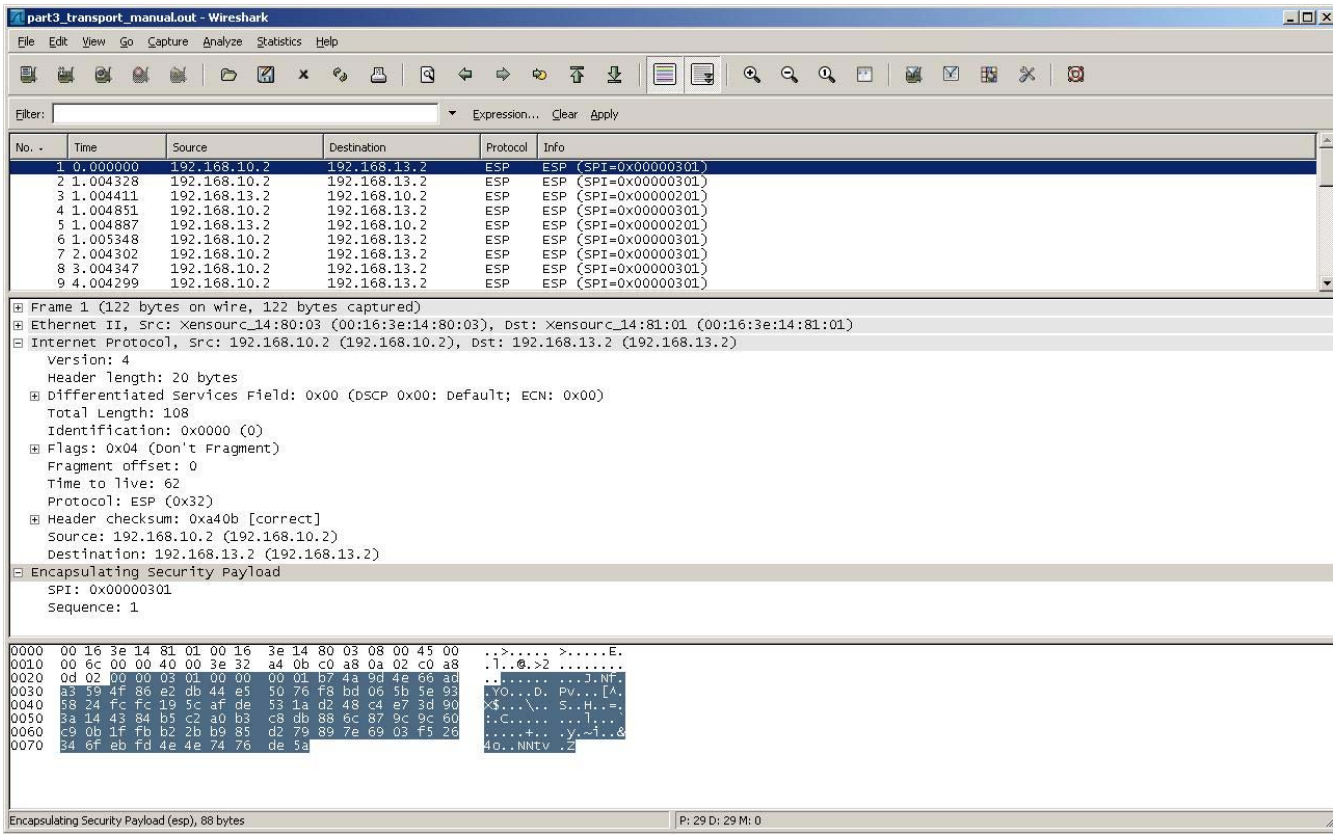
Lines 8 and 9 contain the required configuration settings for the SA in the other direction. The encryption key for this SA is different from the one used for the previous SA.

We configured the Security Policies in the following way

Using the keyword spdadd we specified the two end hosts which would be taking part in the IPSec connection and gave the SA a direction (IN since the configuration file is on M3). Also we specified that the SA should be set up using the transport mode and the ESP protocol. This is shown in Lines 11 and 12. Lines 13 and 14 specify the other direction (OUT).

2. IPSec provides only confidentiality in this case. We are neither using AH nor using authentication/integrity services provided by ESP. This is not a very secure way to setup an IPSec connection. The cryptographic protocol used in this setup is 3DES-CBC with a 192 bit long key for encryption. We are using only the encryption option.

3. No,the IP header is not encrypted. This is because we are using the Transport mode which runs on top of IP. The length of the IP header is 20 bytes. All of the IP payload is encrypted while the IP header itself is left untouched. This is shown in the following screenshot.



4. No. Since the entire IP payload is encrypted, there is no way in which we can determine if the higher layer protocol is TCP or UDP or IP.

5. Some of the difference between the transport mode and the tunnel mode are:

- The original IP header is protected in the tunnel mode while it is not in the transport mode
- Transport mode provides host to host security while transport mode provides gateway to gateway security
- In the tunnel mode, the end systems are not required to be IPSec aware and only the machines on the end points of the IPSec tunnel (gateways) need be. In transport mode, both end hosts

need to be IPSec aware.

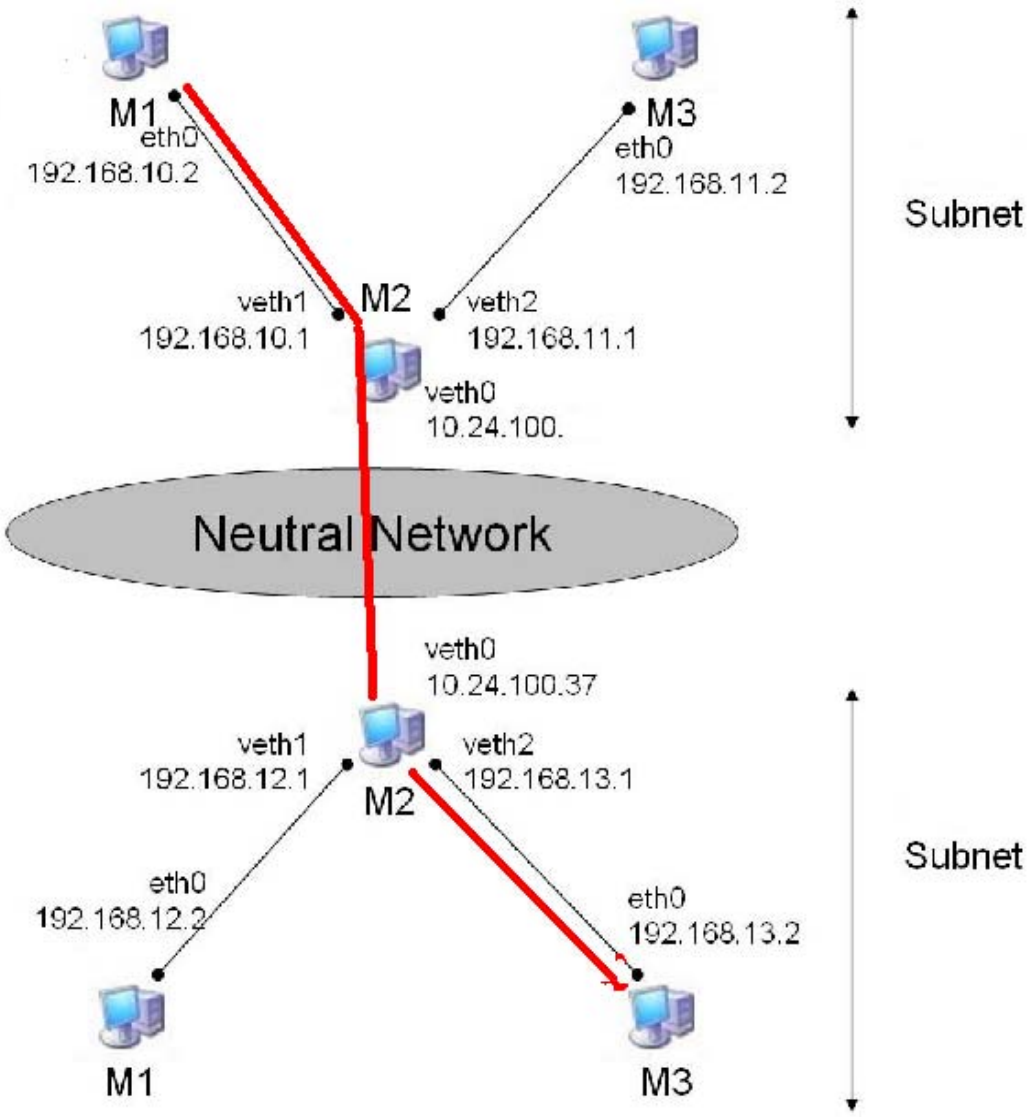
6. The source and destination addresses of the IP Packets are the end hosts 192.168.10.2 and 192.168.13.2 . No, these are not the same as the source and destination of the tunnel mode in part 2. NAT traversal allows visibility from one end host to the other.

7. "Use" level is used if an SA has already been established between the hosts. The packets are hence IPSec protected if a session was existing else they will be sent in the clear. The "require" lever requires an SA to be setup each time a new communication session begins, irrespective of whether an earlier SA existed or not. We used the require level in out setkey.conf file. Other levels are default and unique.

Part 4: Automatic Keying with IKE with pre-shared keys (Transport Mode)

In this part of the lab we used the method of pre-shared keys to set up the IPSec SA in the Transport mode. The SA was setup between host M1 (192.168.10.2) of the network with external IP address 10.24.100.15 and host M3 (192.168.13.2) of the network with external IP address 10.24.100.37.

Shown below is the network topology used by us.



1 a). The Configuration file for the IPSEC daemon Racoon : *racoon.conf*

```
path pre_shared_key "/etc/psk.txt";                                    ------>1

remote 192.168.10.2 {                                                  ------>2
                        exchange_mode main;                            ------>3
                        proposal {                                     ------>4
                                encryption_algorithm 3des;             ------>5
                                hash_algorithm md5;                    ------>6
                                authentication_method_pre_shared_key;>7
                                dh_group modp1024;                     ------>8
```

```
                        }                                   ------>9
                    }                                       ------>10

sainfo anonymous{                                           ------>11
        pfs_group modp768;                                  ------>12
        encryption_algorithm 3des;                          ------>13
        authentication_algorithm hmac_md5;                  ------>14
        compression_algorithm deflate;                      ------>15
        }                                                   ------>16
```

The configuration file contains details about various parameters that are used while setting up an IPSec SA between the two communicating hosts. Following is a line by line desctiption of the file.

Line 1: Specifies the method that we are using to set up the IPSec SA (preshared key) and the path where the preshared keys file will be found on the system.

Line 2: Identifies the remote host and requires the settings specified inside the block to be used to set up the SA with the remote host.

Line 3: This option specifies that the main mode should be used. Optionally, we can also use the aggressive mode.

Line 4-9: This block is used to negotiate the cryptographic techniques that will be used to communicate with the peer. Encryption is needed to provide confidentiality and the encryption to be used is 3DES in this case. Integrity is ensured using the MD5 hashing algorithm and finally authentication is done using an RSA signature. Line 8 specifies the group to be used for Diffie-Hellman exponentiations.

Lines 11-16: This block is used during Phase 2 of IKE. Line 11 specifies that the block remains anonymous. Instead of setting this for a specific host, the anonymous parameter is used to specify that these settings should be used for all hosts that do not have a specific configuration [1]. This is sufficient for simple connections. The peer is identified using the presharedsecret.

Line 12: PFS stands for perfect forward secrecy. This is a property of a protocol in which some who sniffs encrypted traffic cannot later decrypt the conversation. This is achieved by using another round of (less heavy) cryptographic techniques in Phase 2. This again refers to the Diffie Hellman exponentiation group to be used.

Line 13-14: Encryption to be used is 3des and authentication is to be achieved using HMAC with MD5 as the underlying hashing algorithm.

Line 15: IPSec tries to reduce the traffic to be carried over the network and compresses the IP payload. The currently used algorithm used for compression is deflate.

1 b) The configuration file for Setkey: setkey.conf

Setkey is a small utility which allows us to change and configure the IPSEC key management in intuitive ways. The following is a typical setkey.conf file to enable this setup.

```
#!/usr/sbin/setkey -f

#Config for 192.168.13.2

# flush the SAD and SPD

flush;
spdflush;

#security policies

spdadd 192.168.13.2 192.168.10.2 any -P out ipsec
esp/transport//require;

spdadd 192.168.10.2 192.168.13.2 any -P in ipsec
esp/transport//require;
```

We need only to define the directions over which these policies are to be applied. In this setup, there are two associations, one for each direction.The first IP address specifies the origin of the packets while the second one specifies the destination and hence, packets originating from M3 (192.168.13.2) of one subnet, destined to M1 (192.168.10.2) of the other subnet have to follow the OUT policy. The esp protocol in the transport mode is required. The setkey.conf file for the other host will be the same except for the IN and OUT directions which are opposite to the one shown.

1.c). Finally, the file from which the preshared key is derived: psk.txt

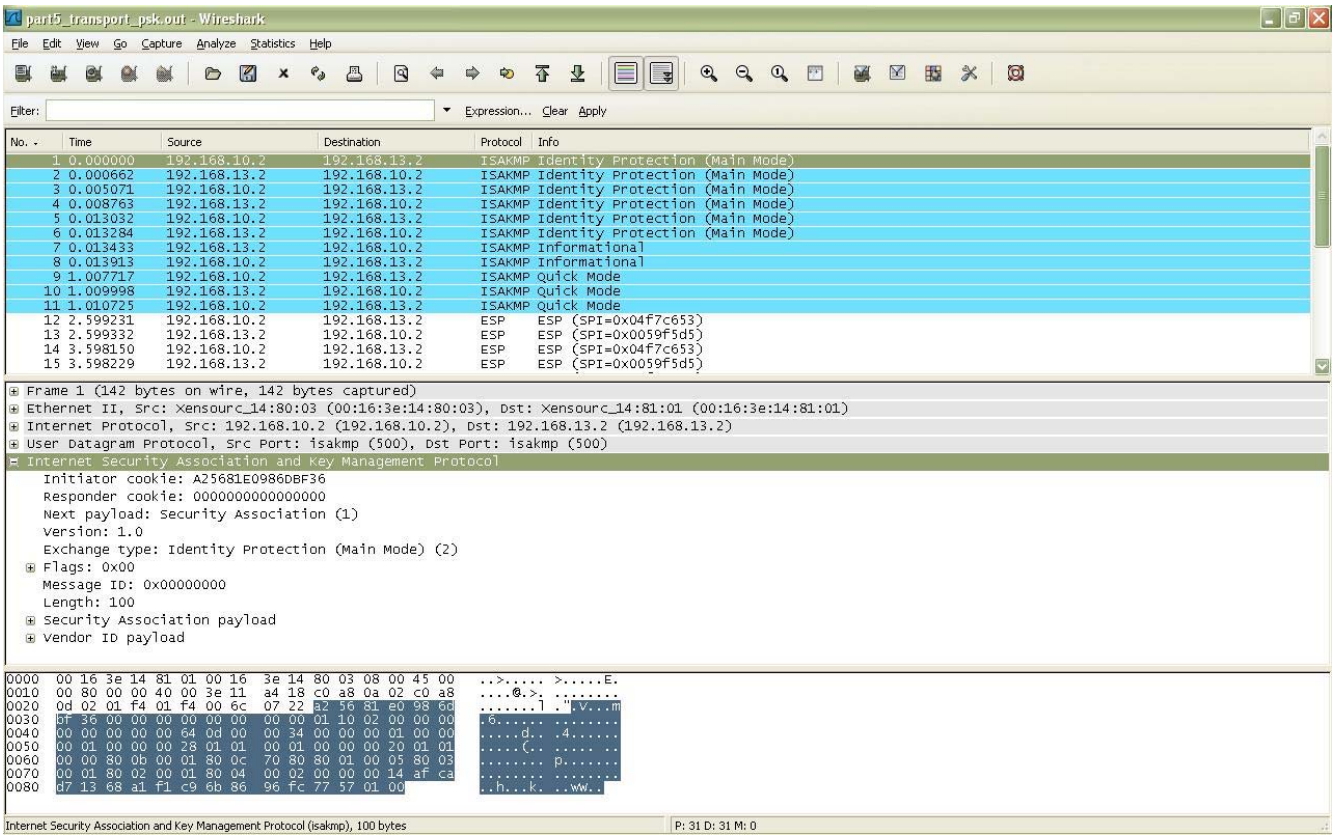
```
#This file contains the preshared key

192.168.10.2            presharedkey.
```

This is a simple text file in which the first column specifies the identifier of the remote host while the second column specifies the pre shared key. The identifier could be an IP address, an email address or a website. The file can have different identifiers for different hosts.

2. In this case IPSec provides, confidentiality through encryption and both integrity and authentication using HMAC. We are using

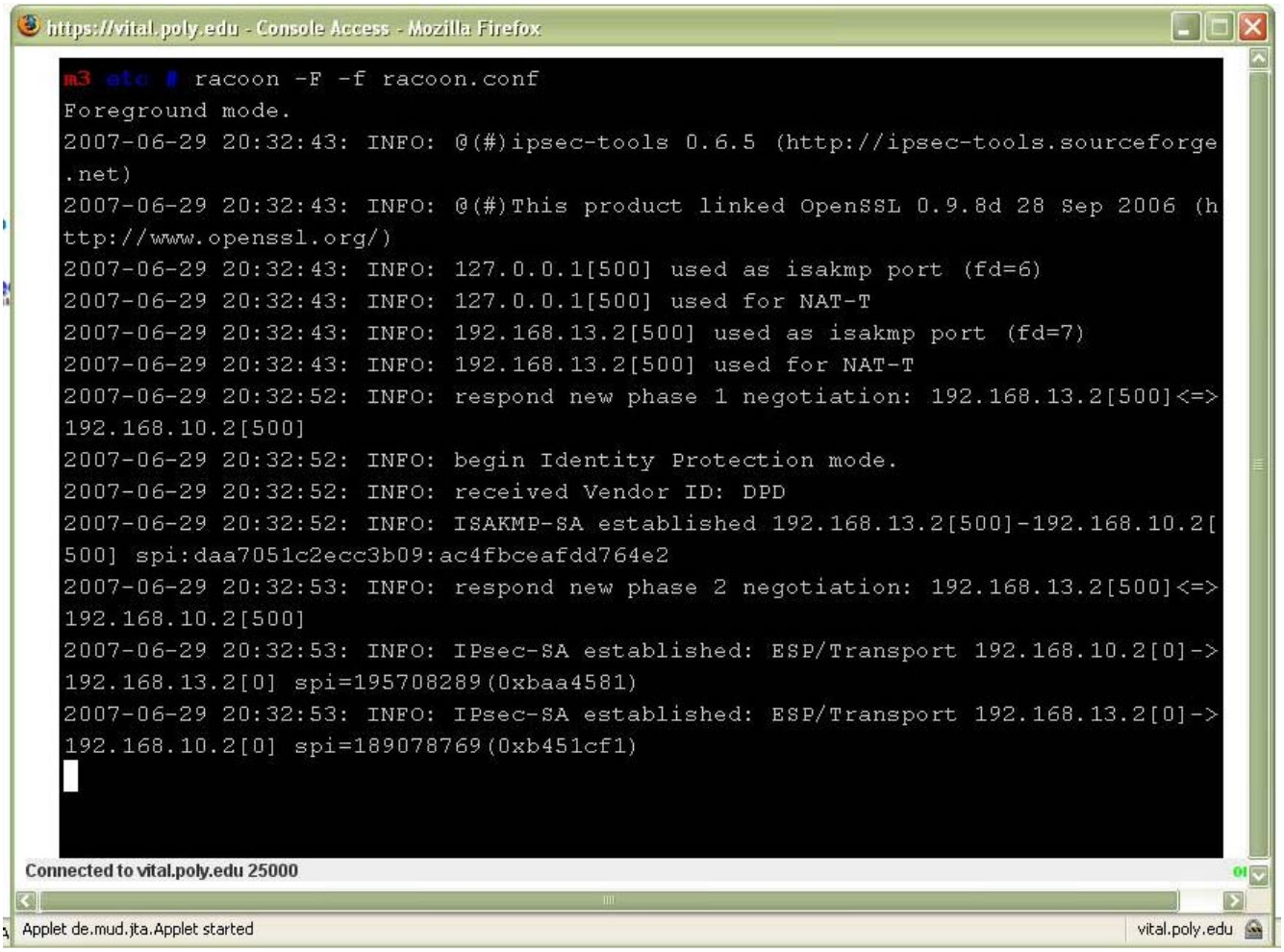
3. Description of IKE with preshared keys in the transport mode.



The IKE has two phases. Phase 1 is used for mutual authentication and to establish session keys and at the end of this phase, two session keys are established, an integrity key and an encryption key. It takes the first 4 packets to create these keys and they are used to encrypt the rest of phase 1 and all of the phase 2 messages. Phase 1 can be done in either the main mode or aggressive mode. Here we use the main mode for our purposes.

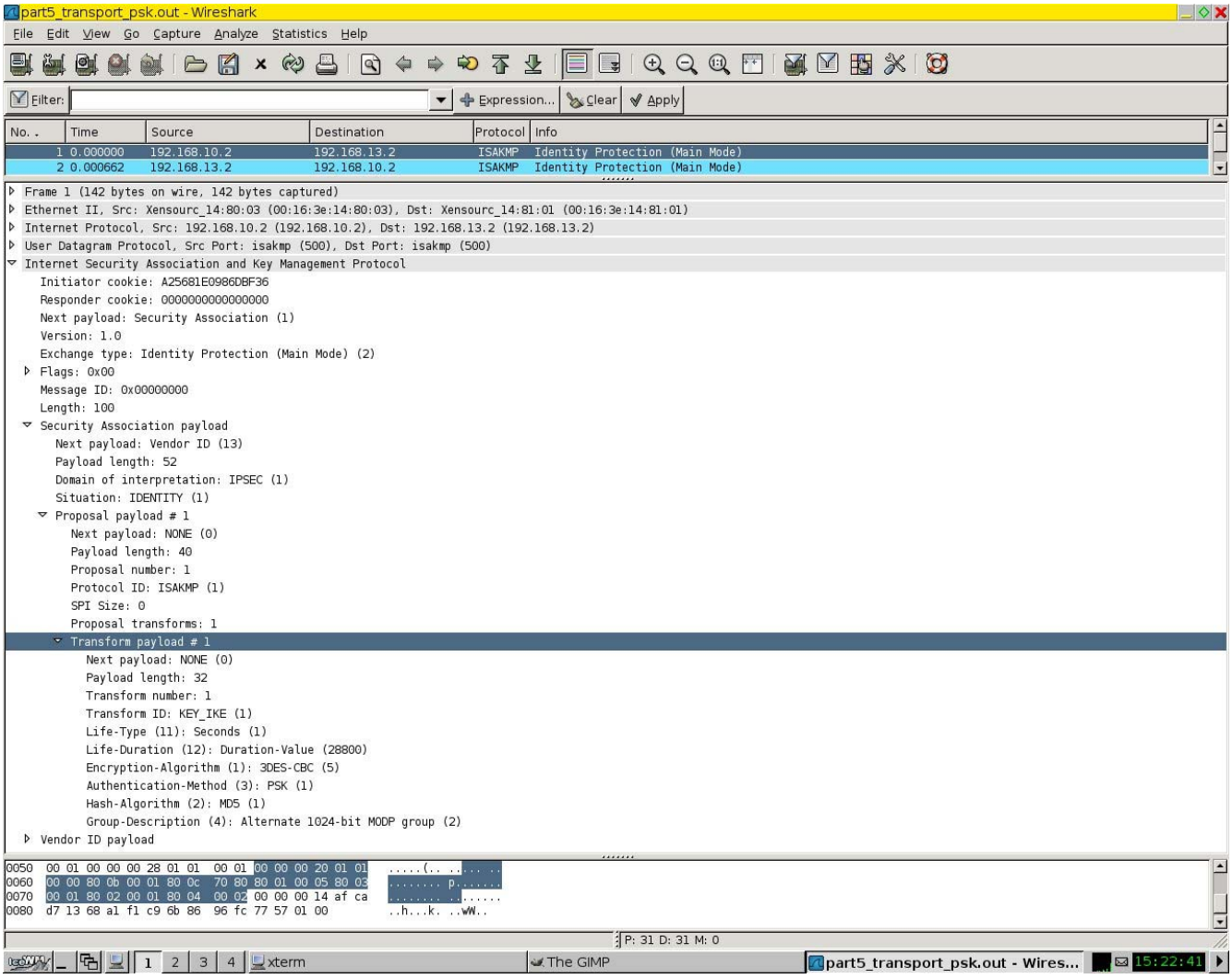
Phase 2 of the IKE builds up on the IKE / ISAKMP security association that was created in phase 1 to create sessions between the two host machines. Once the IKE SA is setup, either of the hosts can initialte an IPSec SA through Phase 2. This mode is also called the quick mode. We can establish this SA as either an ESP or/and AH SA .

Phase 1: The ISAKMP SA establishment (Main Mode):



The screen shot above was taken while running racoon in the foreground or diagnostic mode using the command racoon -F . It shows a successful, key exchange between the participating hosts 192.168.13.2 and 192.168.10.2. Notice that the ISAKMP SA is first established and the IPSec SA establishment follows it. As required, the ESP protocol is running in the transport mode.

Messages 1 and 2.

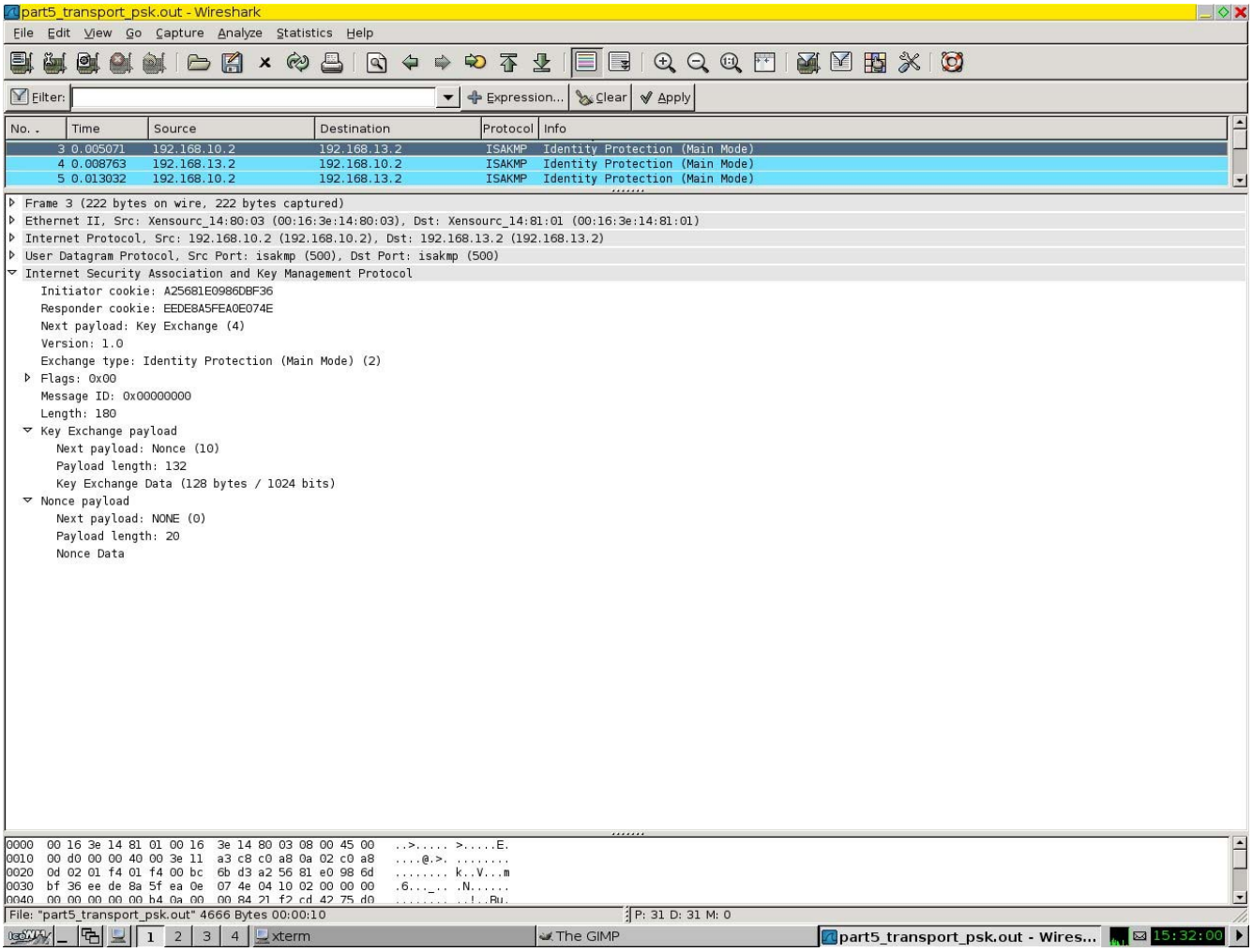


The first message of the ISAKMP protocol begins with an Initiator cookie. In this case the IP address of the initiator is 192.168.10.2 and the packet is destined for 192.168.13.2. The value of the initiator cookie is A25681E0986. The Responder cookie value is set to all 0s as it is not known yet.

This message also sends the Crypto Proposal from the first machine to the second machine to agree upon, for encryption and authentication purposes. In this case the the encryption algorithm will be 3DES-CBC, Authentication is by using preshared keys and the hash algorithm for integrity check is MD5. The proposal also contains the diffie-hellman key exchange modp group.

The second message is almost exactly the same as the first except that the source and destination are now reversed. The responder cookie is now chosen and set. These two cookies will be used as the session identifiers during IKE. Once the initiator and responder cookies are set and there is agreement over the Crypto proposal, by both the machines, the next step is to establish the session keys, this occurs in the following 4 packets.

Messages 3 and 4.



These two messages concern themselves with the Diffie-Hellman key exchange. Each host independently computes and exponent and a nonce which is then sent to the other host. This is done one after the other in messages 3 and 4. Once both the hosts have these values, they compute the Diffie Hellman key.

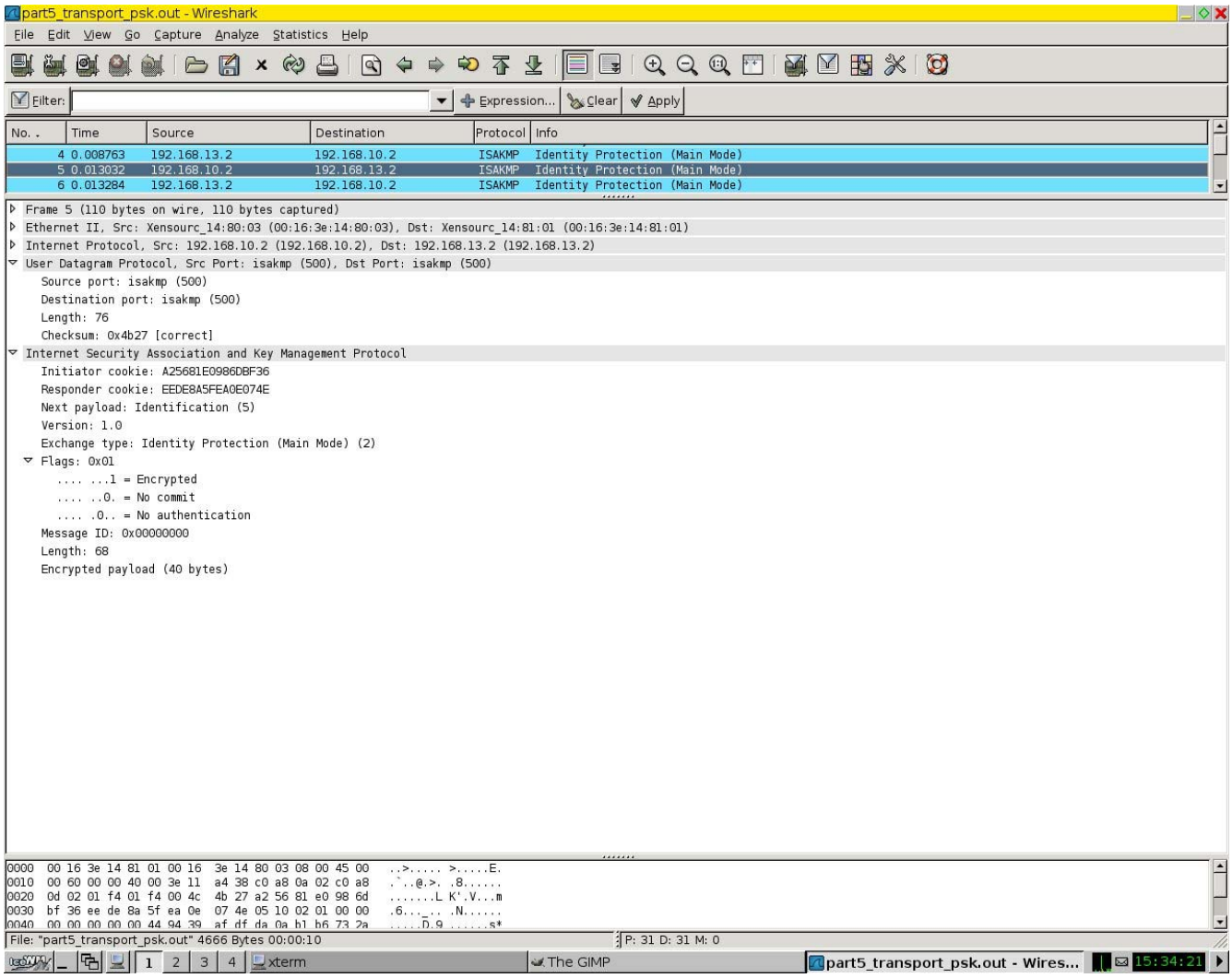
Using this they then create a session key which is a function of the Diffie Hellman key, the nonce sent by the host 1,the nonce sent by host 2, the initiator cookie, the responder cookie and the shared secret, which in this case is 'presharedkey'.

Messages 5 and 6

The newly minted session key is put to test in the final two steps in Phase 1 of the IKE. The 5th message is needed for proof of identity. Alice should know that she is talking to Bob and not anyone else. Similarly Bob needs to know if he really is talking to Alice.

More formally , the proof of identity proves that the sender knows the key associated with the identity (the preshared key) and it also serves as integrity protection for the previous messages. This is so because the session key that was generated in the previous message is used to compute the

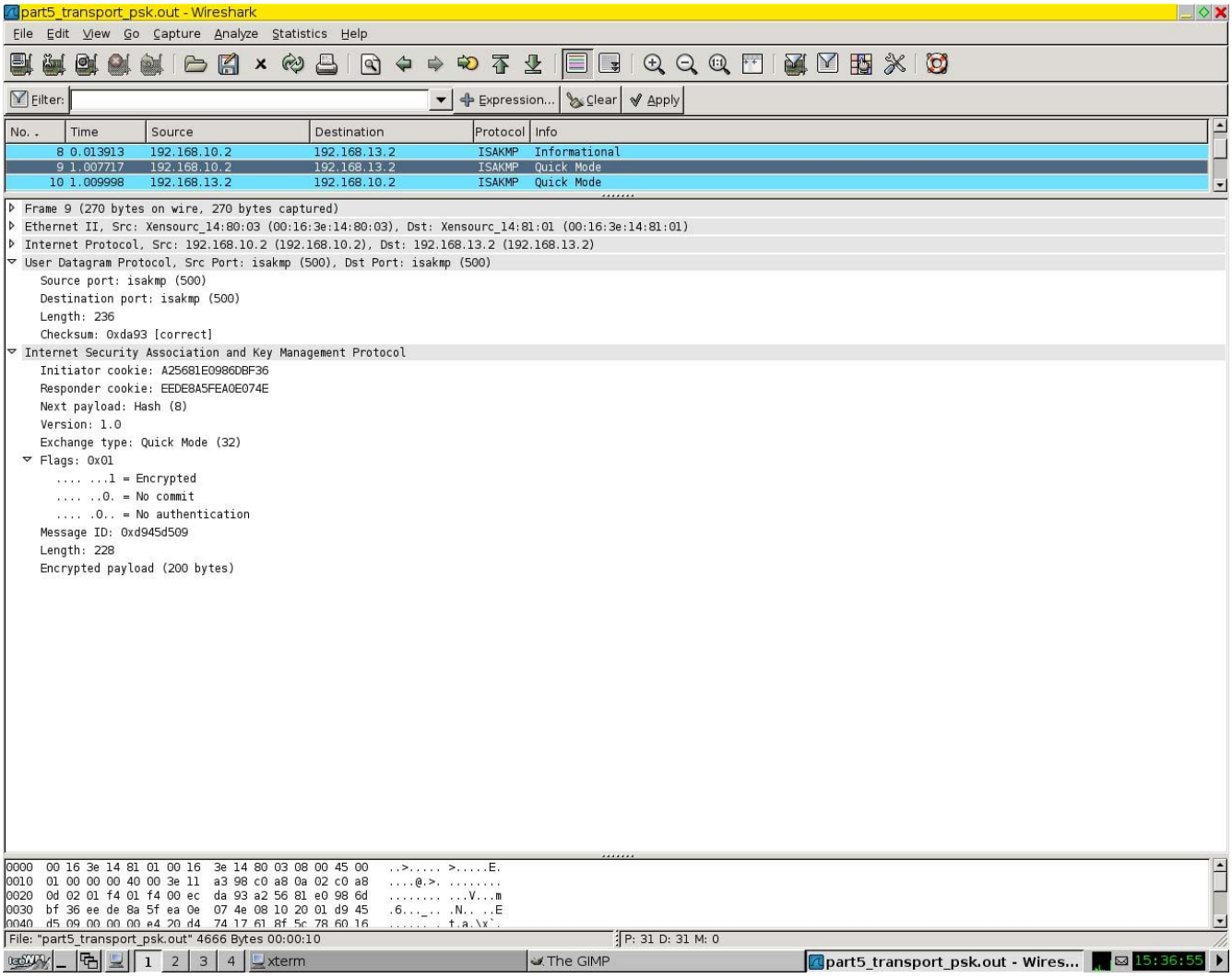
has of the preshared key and the person associated with it.



We can notice in the screen shot that the encryption flag is set this time which means that the encryption of IKE packets begins now. Message 6 is exactly the same as message 5 and the only differences are in the check sum and source and destination addresses (which are reversed).

We should notice that the IPSec SA establishment has not started yet and none of the policies that we defined in the setkey.conf file are in use. The next three messages constitute the informational section, followed by the second phase of IKE which relates to IPSec SA establishment.

Phase 2: IPSEC SA establishment (Quick Mode).



The Quick Mode is a 3- message protocol which negotiates parameters for the Phase2 SA, including cryptographic parameters and the SPI with which the Phase-2 SA will be identified. The SPI is set automatically and we do not include it in the configuration files.

Messages 1 and 2 of Phase 2 :

The second phase need not be initiated by the same pair of hosts that initiated the first pair. The first message constitutes of the pair of cookies agreed upon in Phase 1, a new 32 bit number chosen by the initiator to distinguish this phase 2 setup and some encrypted traffic which consists of the new crypto proposal for the IPSec SA, a nonce and the first Diffie-Hellman exponentiation.

The cookie pair serves as the identifier for Phase 1 and this can be common for multiple IPSec SAs. Similarly the new 32 bit number serves as the identifier for each of the individual IPSec SAs, allowed by the policy, as they can share the same Phase 1 information.

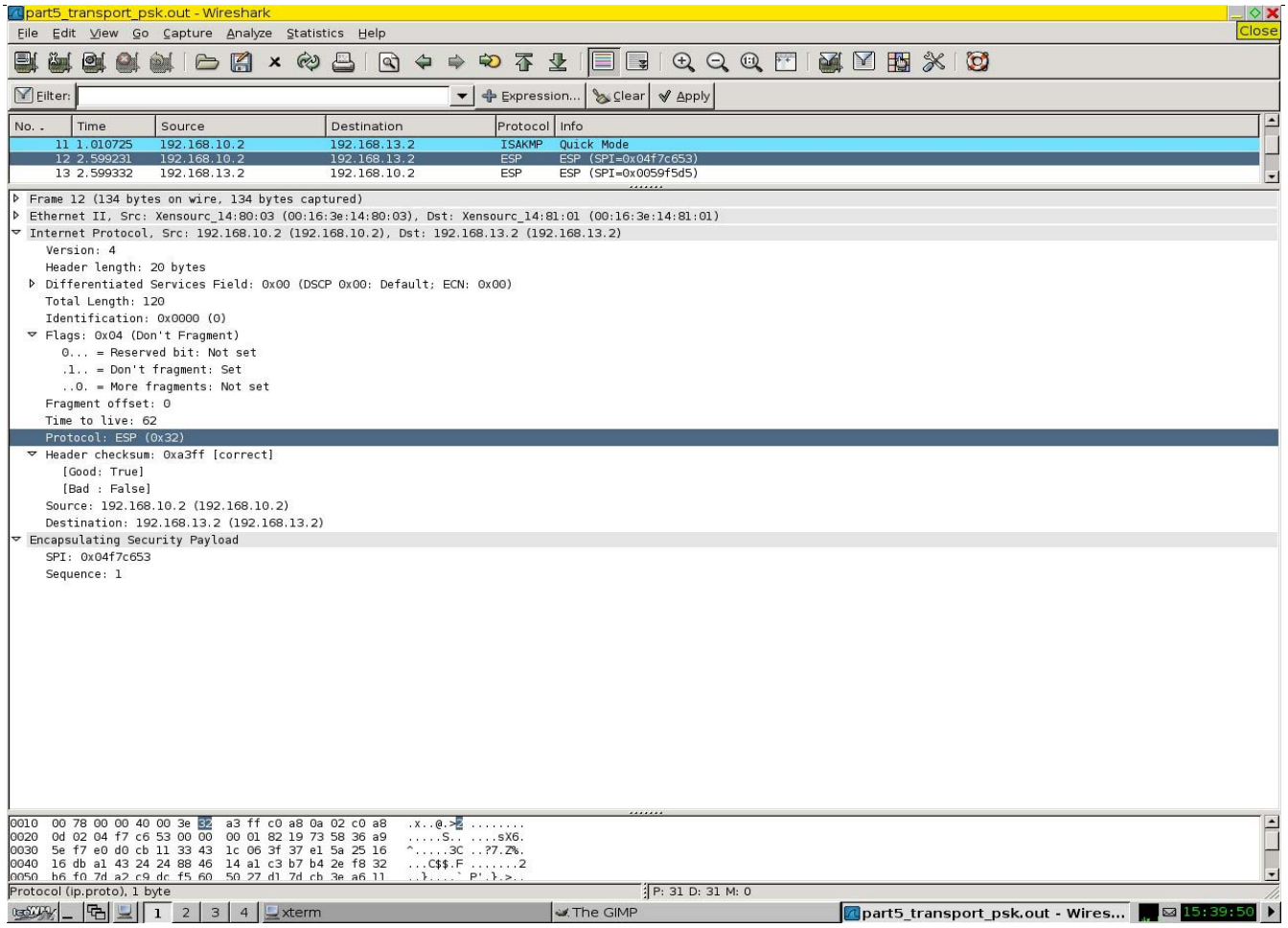
The second message is very similar to the first and apart from all the usual identifiers it contains the accepted crypto proposal and the second Diffie Hellman exponentiation. The DH key exchange is to allow Perfect Forward Secrecy (PFS), but the parameters are not negotiated on the fly.

Hence there is a different DH group specified in the racoon.conf for the quick mode in the line,

pfs_group modp768;

Message 3: The final message of IKE is the acknowledgment from the initiator of the quick mode along with both the identifiers of Phase 1 and Phase 2 ie, the pair of cookies and the 32 bit identifier for Phase 2. Once this is done, the actual traffic flows between each of the hosts over the specified protocol. Throughout our lab we will be using the ESP protocol for encryption.

The following capture shows the successful establishment of the IPSec connection with the Esp protocol



4. The ISAKMP SA is setup first and the keys generated here are used to set up the IPSec SA. Multiple IPSec SAs can be setup using the same pair of ISAKMP keys.

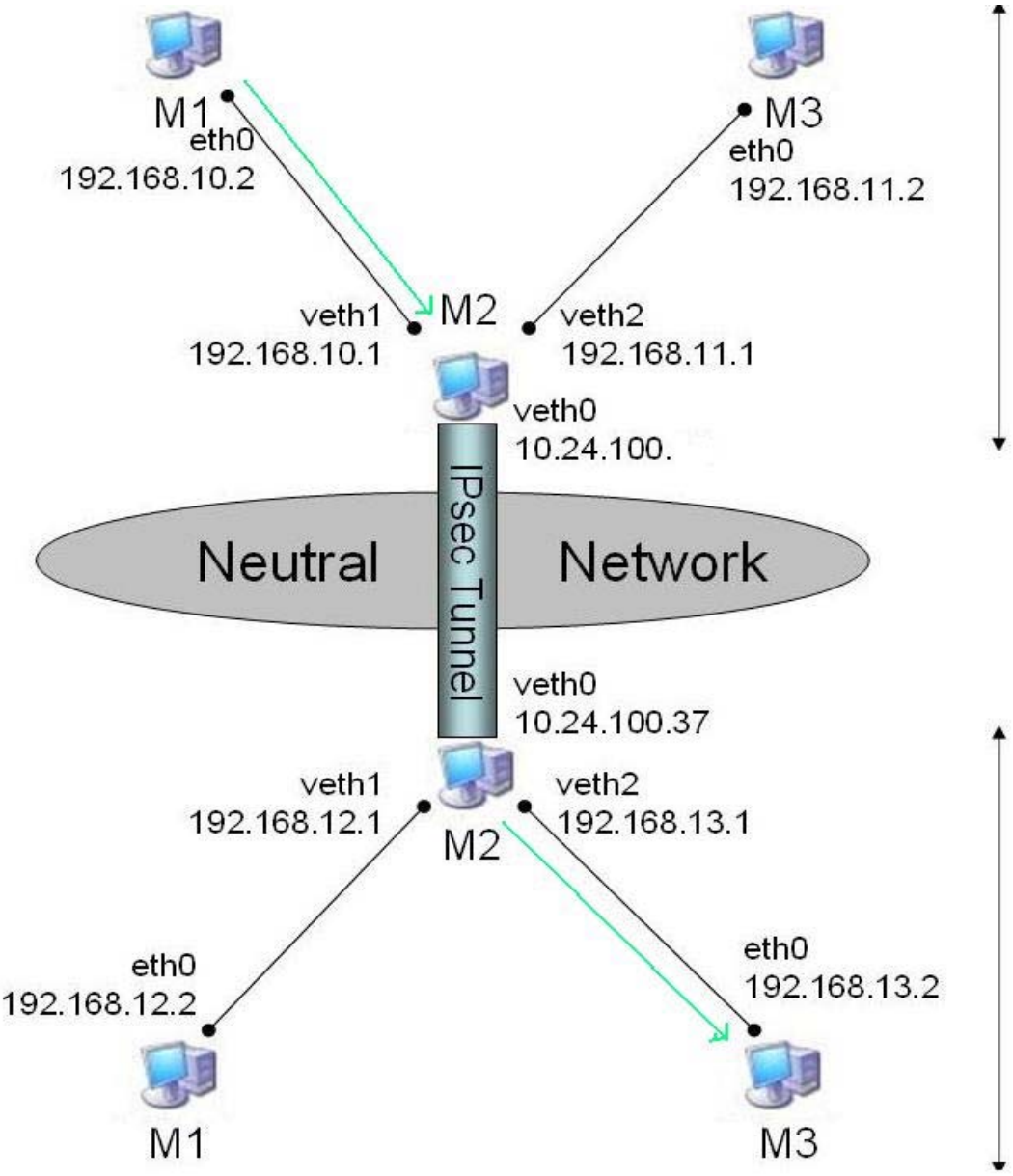
5. The SPI for the SA from 192.168.10.2 and 192.168.13.2 is 0x04f7c653 while that for the SA from 192.168.13.0 to 192.168.10.2 is 0x0059f5d5.

Part5 Automatic Keying in the IPSec tunnel mode using certificates:

In this part of the lab we used certificates to set up the IPSec SA in the Tunnel mode with IKE. Shown below is the map of the network over which the IPSec SA was established. Traffic was

generated across two networks NATted behind the gateways 10.24.100.15 and 10.24.100.37 between which the IPSec tunnel was established. Traffic was captured on one of the gateways.

Nodes 192.168.10.2 and 192.168.13.2 were our communicating hosts.



We used the IKE daemon called Racoon. We configured racoon using the racoon.conf file shown below.

1 a) The racoon configuration file

```
path certificate "/etc/certs";                                          ------>    1

remote 10.24.100.15 {                                                   ------>    2
        exchange_mode main;                                             ------>    3
        certificate_type x509 "cacert.pem" "privkey.pem";               ------>    4
        verify_cert off;                                                ------>    5
        my_identifier asn1dn;                                           ------>    6
        peers_identifier asn1dn;                                        ------>    7
        proposal {                                                      ------>    8
                encryption_algorithm 3des;                              ------>    9
                hash_algorithm md5;                                     ------>    10
                authentication_method rsasig;                           ------>    11
                dh_group modp1024;                                      ------>    12
        }                                                               ------>    13
}                                                                       ------>    14

sainfo address 192.168.13.0/24 any address 192.168.10.0/24 any{         ------->   15
        pfs_group modp768;                                              ------>    16
        encryption_algorithm 3des;                                      ------>    17
        authentication_algorithm hmac_md5;                              ------>    18
        compression_algorithm deflate;                                  ------>    19
}                                                                       ------>    20
```

The configuration file contains details about various parameters that are used while setting up an IPSec SA between the two communicating hosts.

Line 1: Specifies the method that we are using to set up the IPSec SA (certificate) and the path where the certificate will be found on the system.

Line 2: Identifies the remote host and requires the settings specified inside the block to be used to set up the SA with the remote host.

Line 3: This option specifies that the main mode should be used. Optionally, we can also use the aggressive mode.

Line 4: Specifies the type of certificate (x509) to be used for authentication purposes and in quotes specifies the name of the certificate file (cacert.pem) and name of the file containing the private key (privkey.pem). Both these files would be looked up in the path specified in Line 1.

Line 5: This gives us the option of verifying if the certificate is signed by a trusted authority or not. We turn this off as we are signing the certificate ourselves.

Line 6: Contains the identifier for the host machine. This identifier could also be an IP Address, a fqdn (fully qualified domain name), and so on. We have used asn1dn as our identifier of choice. It stands for Abstract Syntax Notation 1 (ASN.1) domain name. The actual syntax is my_identifier asn1dn [*string]* , where the string stands for the domain name, but it is not necessary to specify it. If

this field is left blank, then the DN identifier will be taken from Subject field of the certificate.

Line 7 : Similar to Line 6, corresponds to the peer.

Line 8-13: This block is used to negotiate the cryptographic techniques that will be used to communicate with the peer. Encryption is needed to provide confidentiality and the encryption to be used is 3DES in this case. Integrity is ensured using the MD5 hashing algorithm and finally authentication is done using an RSA signature. Line 13 specifies the group to be used for Diffie-Hellman exponentiations.

Lines 15-20: This block is used during Phase 2 of IKE. Line 15 specifies the end points of the hosts which will use the IPSec SA. The first address is the source address and the second address is the destination address. Here we specify the subnets. 'any' stand for any kind of traffic.

Line 16: PFS stands for perfect forward secrecy. This is a property of a protocol in which some who sniffs encrypted traffic cannot later decrypt the conversation. This is achieved by using another round of (less heavy) cryptographic techniques in Phase 2. This again refers to the Diffie Hellman exponentiation group to be used.

Line 17-18: Encryption to be used is 3des and authentication is to be achieved using HMAC with MD5 as the underlying hashing algorithm.

Line 19: IPSec tries to reduce the traffic to be carried over the network and compresses the IP payload. The currently used algorithm used for compression is deflate.

Similarly we setup the setkey.conf file

1 b) The Setkey Configuration File:

```
#!/usr/sbin/setkey -f                                   ------>1
#                                                       ------>2
# Flush SAD and SPD                                     ------>3

        flush;                                          ------>4
        spdflush;                                       ------>5

# Create policies for racoon

        spdadd 192.168.13.0/24 192.168.10.0/24 any -P out ipsec     ------>6
        esp/tunnel/10.24.100.37-10.24.100.15/require;               ------>7

        spdadd 192.168.10.0/24 192.168.13.0/24 any -P in ipsec      ------>8
        esp/tunnel/10.24.100.15-10.24.100.37/require;               ------>9
```

The configuration file adds one policy in each direction using the keyword sdpadd.

Lines 4 and 5 erase the previous SAD and SPD entries

Lines 6 specifies the direction of the SA. If the source is any host in 192.168.13.0/24 subnet and the destination is any host in 192.168.10.0/24 subnet then the policy should be applied in the OUT direction.

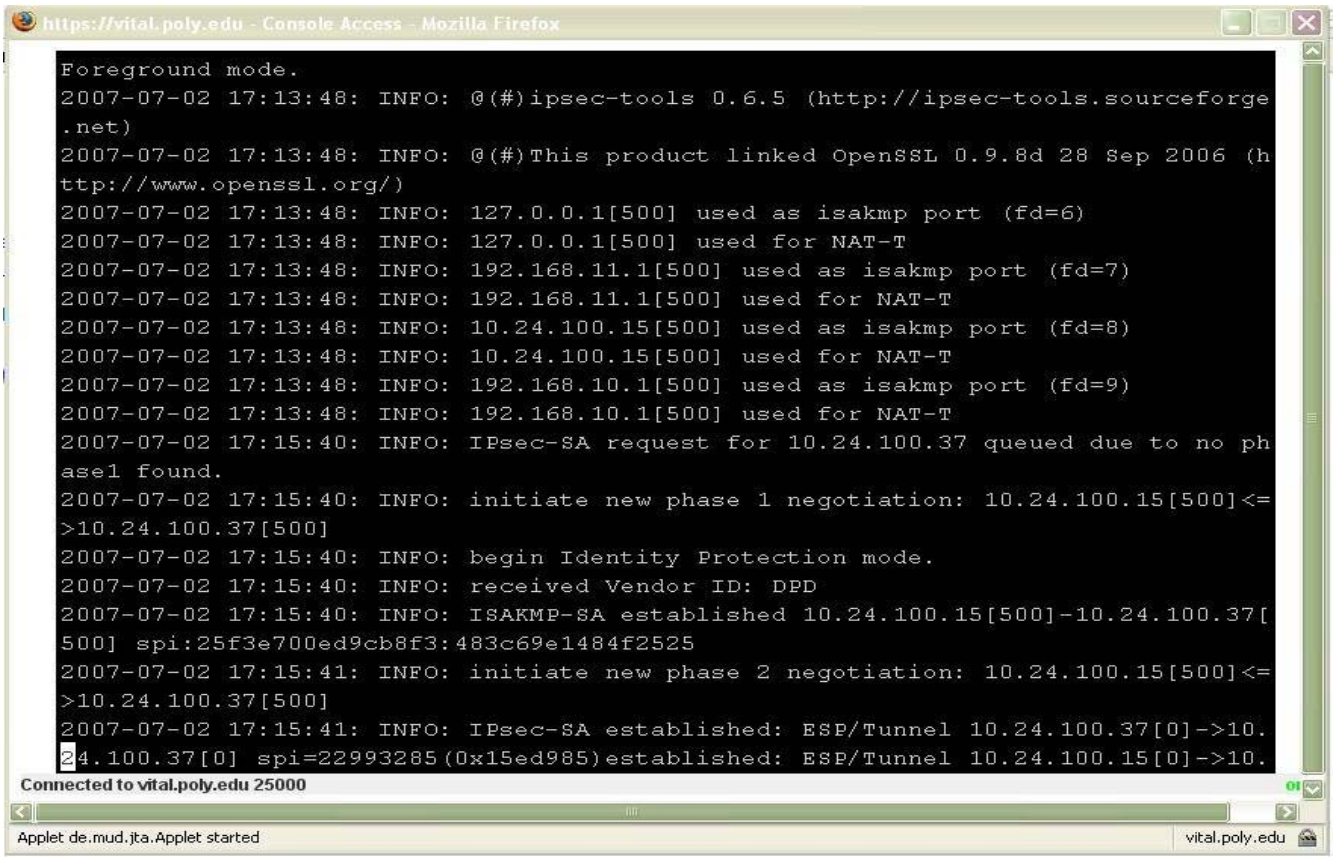
Line 7 informs that we will be using the ESP protocol and that tunnel mode should be used. Also, the tunnel should be between 10.24.100.37 and 10.24.100.15.

Line 8 and 9 specify the other direction.

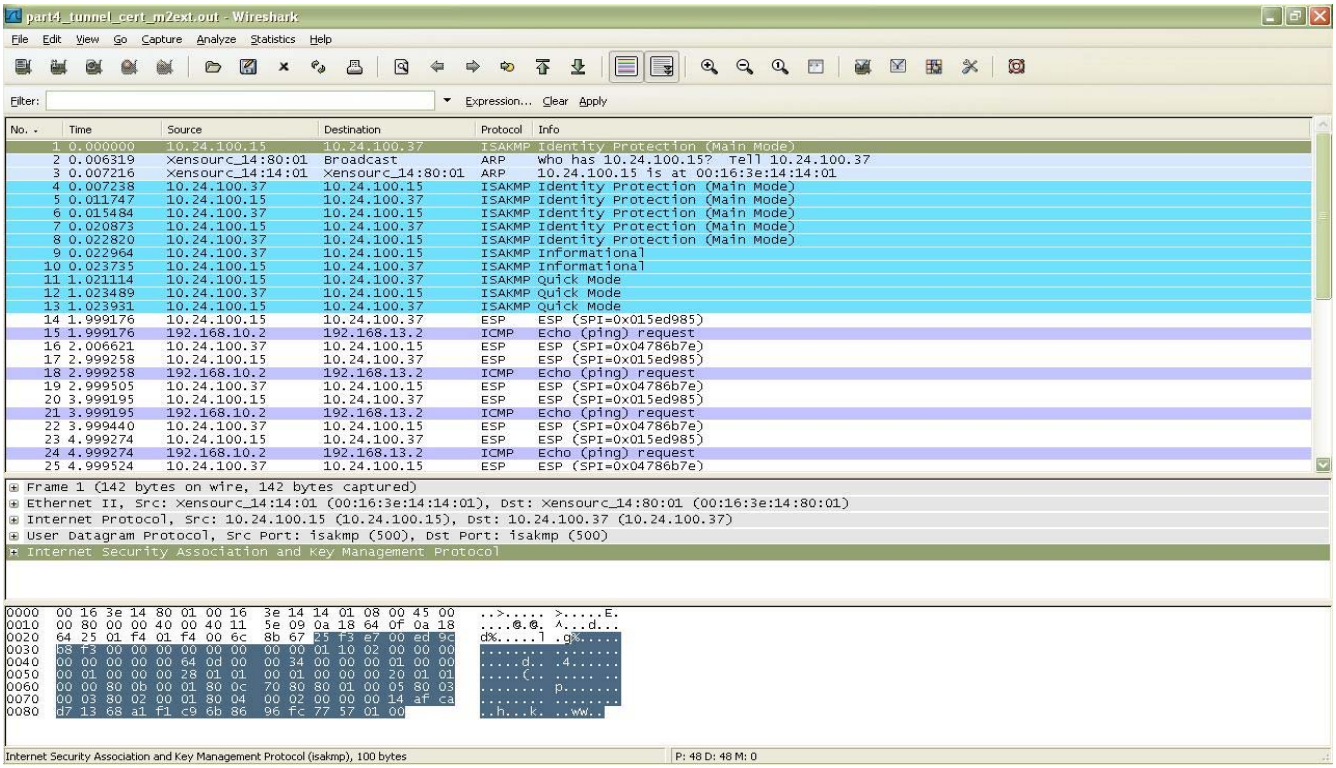
Description of IKE using Certificates in the tunnel mode.

As said earlier, the IKE has two phases. Phase 1 is used for mutual authentication and to establish session keys and at the end of this phase, two session keys are established, an integrity key and an encryption key. Phase 2 of the IKE builds up on the IKE / ISAKMP security association that was created in phase 1 to create sessions between the two host machines. We can establish this SA as either an ESP or/and AH SA .

The screen shown below was taken while running racoon in the foreground using the command *racoon -F* . It shows a successful, key exchange between the participating hosts 192.168.13.2 and 192.168.10.2. Notice that the ISAKMP SA is first established and the IPSec SA establishment follows it. As required, the ESP protocol is running in the tunnel mode and the tunnel is setup in each direction between 10.24.100.15 and 10.24.100.37. We can also see that port 500 is being used for setting up the ISAKMP SA (Phase 1).



Corresponding to the messages on the screen, the following screen shot shows all these steps in the form of packets



Given below is a message-by-message analysis of the IKE using certificates. We will discuss preshared keys in the next section.

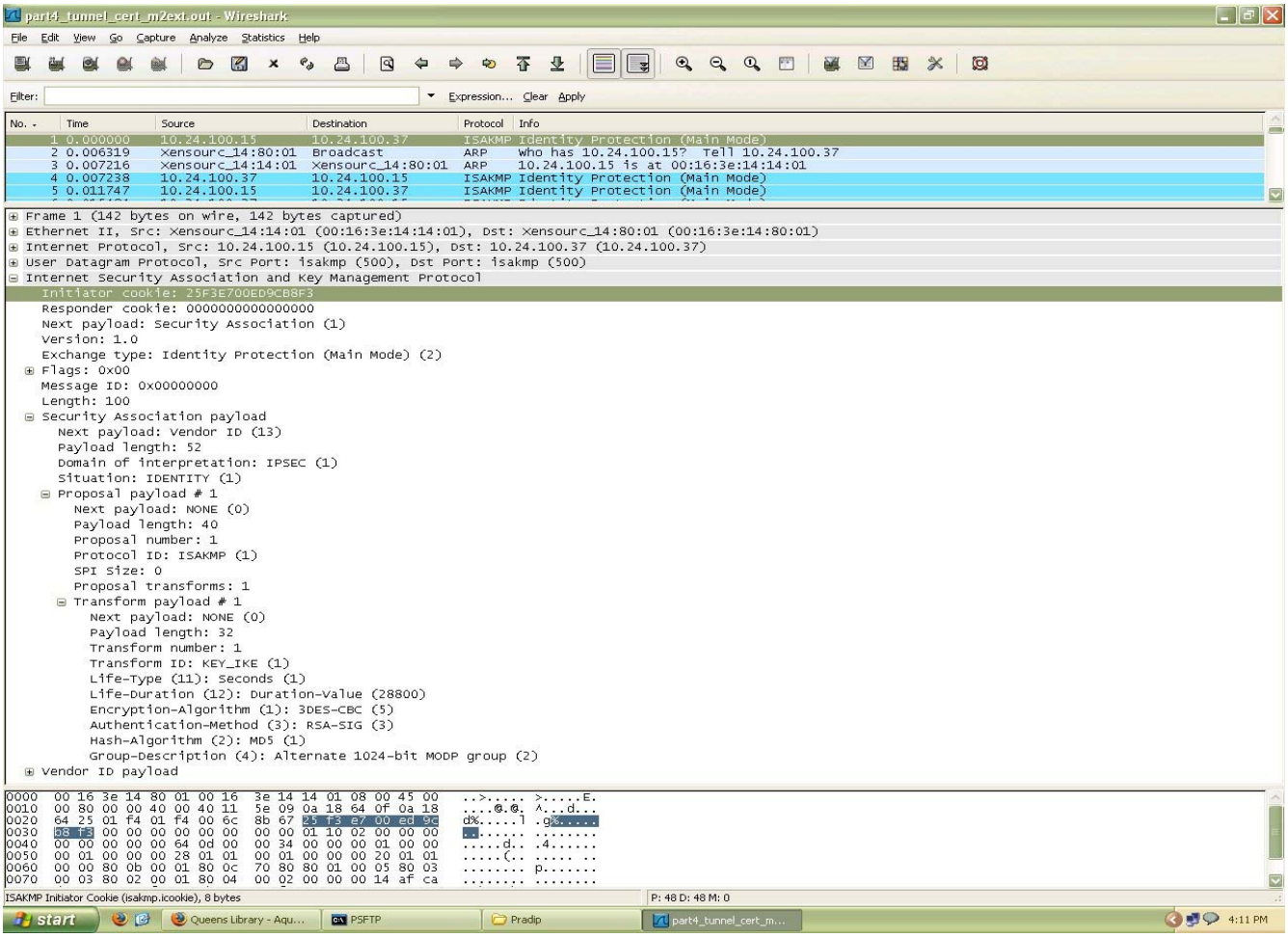
Phase 1 : Identity Protection

To initiate the traffic we just ping the host 192.168.13.2 from the host 192.168.10.2. In the tunnel mode using ESP, the original IP header is encrypted and in its place a new IP header is created. This IP header has the source and destination addresses as the two end points of the IPSec tunnel that we have set up. This is reflected in the packet captures taken at gateway 10.24.100.37.

We should note that the tunnel is set up between the gateways and hence we cannot see the original IP header which contained the source and destination address as 192.168.10.2 and 192.168.13.2 .

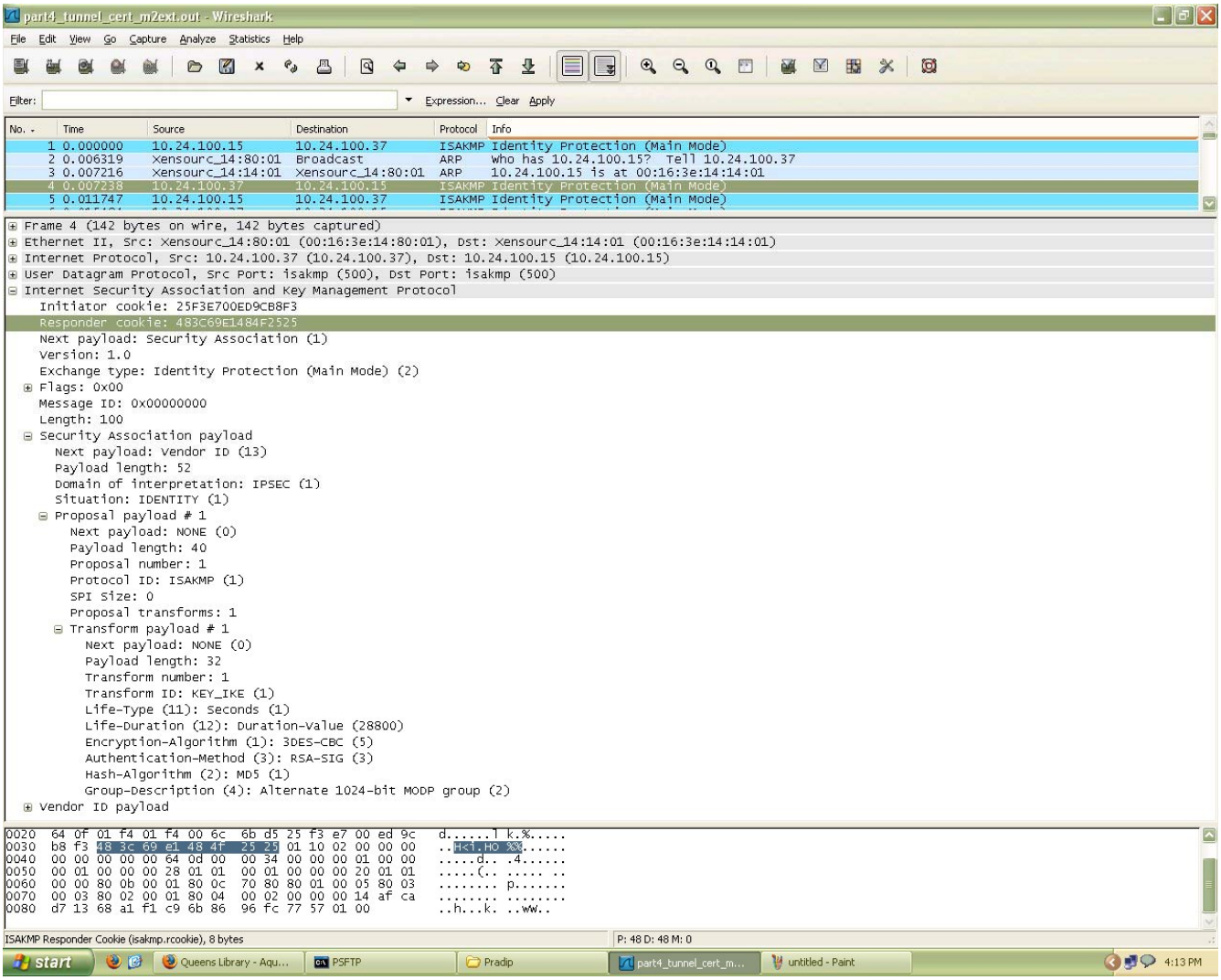
Messages 1 and 2.

The first message of the ISAKMP protocol begins with an Initiator cookie. In this case the IP address of the initiator is 192.168.10.2 and the packet is destined for 192.168.13.2. The value of the initiator cookie is 25F3E700ED9CB8F3. The Responder cookie value is set to all 0s as it is not known yet. The following screen shot is a capture at the 10.24.100.37 gateway.



The major purpose of this message is to send a Crypto Proposal from the first machine to the second machine to agree upon, for encryption and authentication purposes. In this case the the encryption algorithm will be 3DES-CBC, Authentication is by using rsa signatures and the hash algorithm for integrity check is MD5. The proposal also contains the diffie-hellman key exchange modp group.

The second message indicates that the Crypto Proposal is Accepted and it is almost exactly the same as the first except that the source and destination are now reversed. The responder cookie is now chosen and set. These two cookies will be used as the session identifiers during IKE. The values of the initiator and responder cookies will appear in the same order for each of the messages that do the negotiations from now on. After negotiating the cryptographic suite, the actual key exchange process starts from the third message.

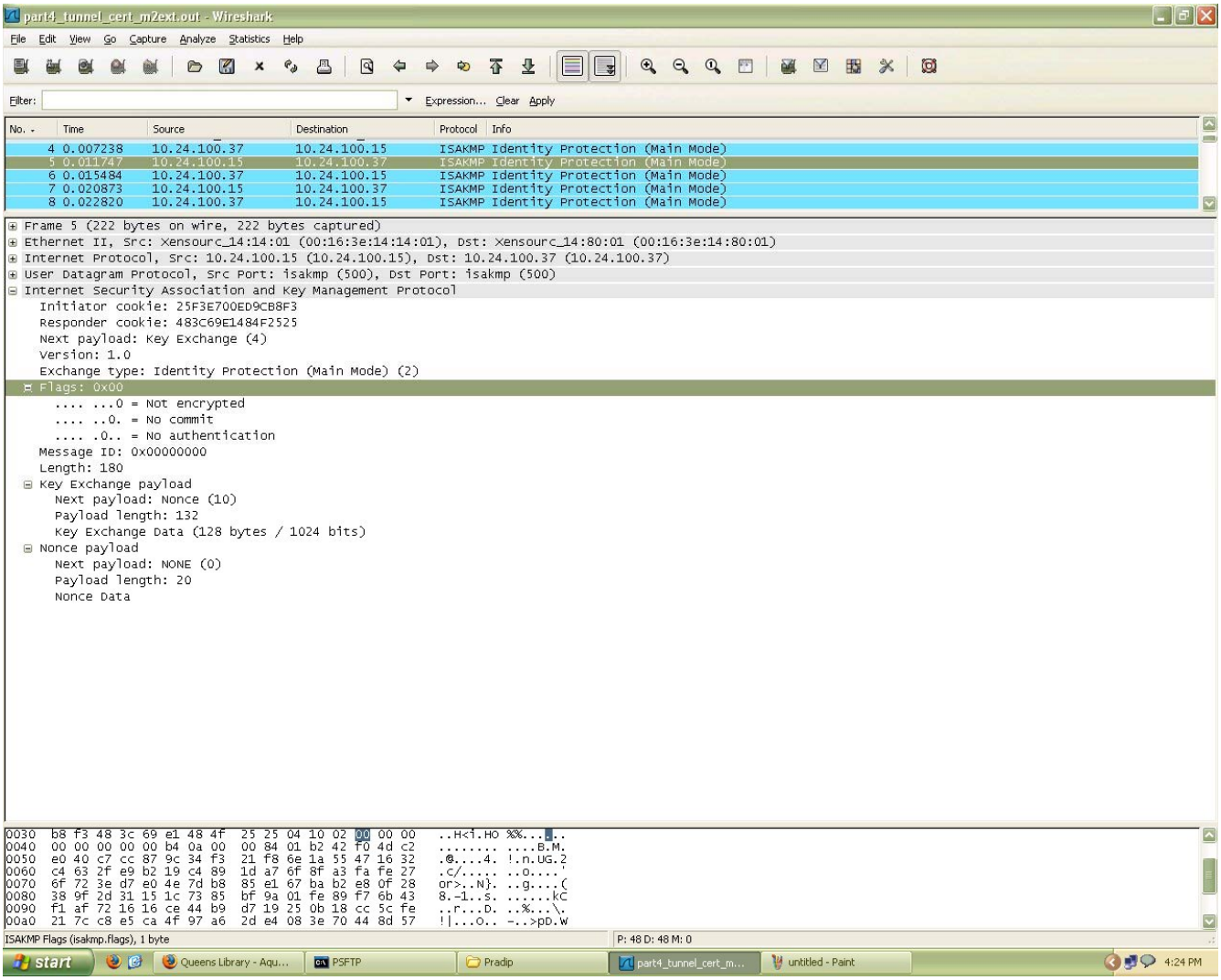


As expected, ISAKMP is running on top of UDP and is using port 500 for this process.

Messages 3 and 4:

These messages are used for the actual key exchange. The initiator (in this case 10.24.100.15) computes the first Diffie Hellman exponent and sends it over to its peer (10.24.100.37) along with a nonce in message three. Similarly, the second Diffie-Hellman exponent is computed at the other end and that along with the second nonce is sent across to the initiator.

Once both ends have each other's exponents, they compute the session key independently. The computed key is also a function of the nonces. Nonces are used in this transaction because then, by just changing the nonces, a new key can be easily computed after some period of time. This avoids the computationally expensive task of calculating new Diffie-Hellman exponents each time the key needs to be changed.



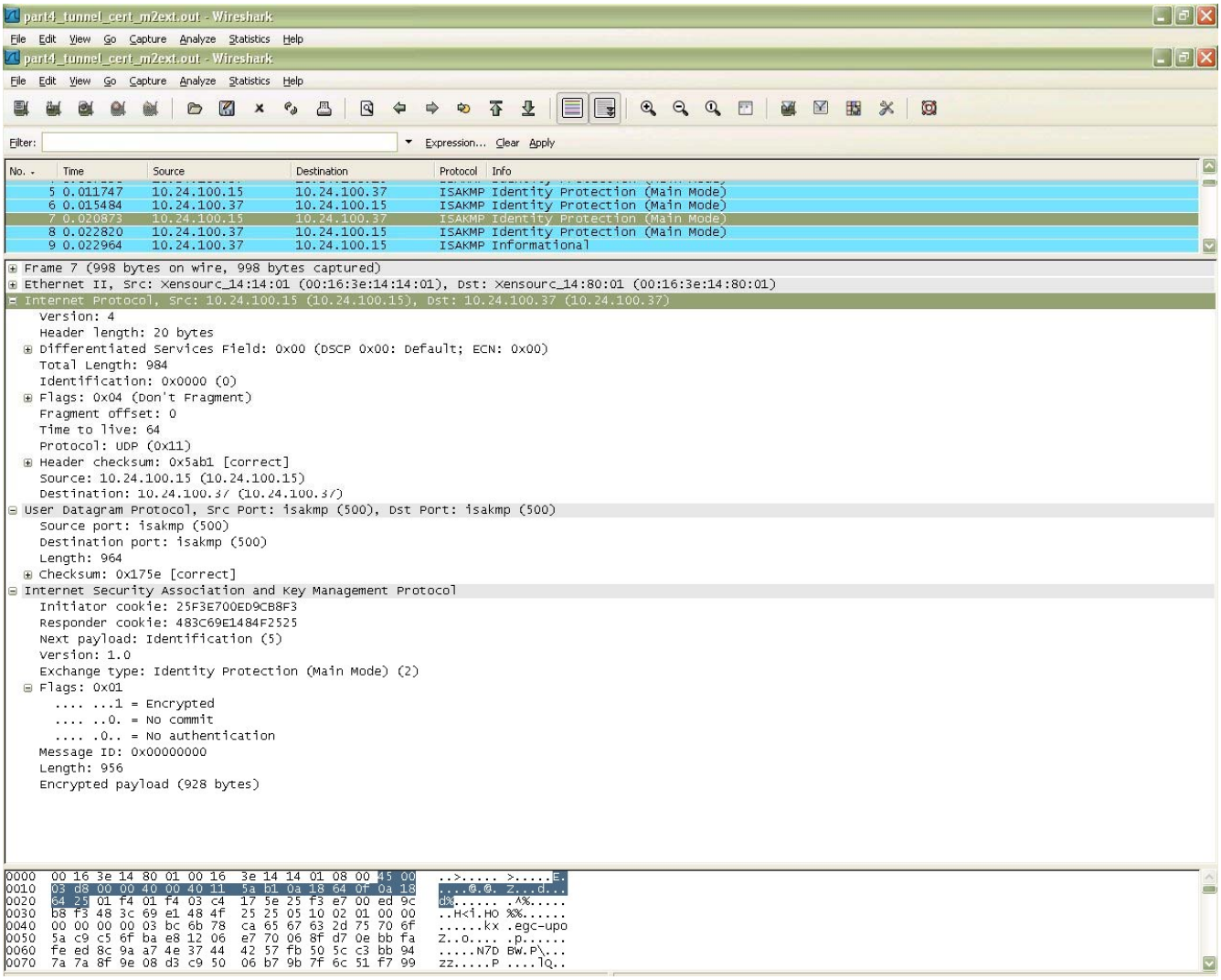
In some of the variants of signature based connections, the Session key is also a function of the initiator and responder cookies.

Messages 5 and 6:

The newly created session key is put to use in the final two steps in Phase 1 of the IKE. The 5th message is needed for proof of identity. Alice should know that she is talking to Bob and not anyone else. Similarly Bob needs to know if he really is talking to Alice.

The proof of identity in this case consists of a signature with the private key over the hash of all the critical information that was sent in the previous messages such as the Diffie Hellman values, the nonces, the cookies etc. The identity itself is suggested to be the IP address of the communicating hosts. To make this transaction more effective, the entire packet is encrypted using the session key that was derived from the previous two messages so that even if an intruder sniffs this message he would not be able to decrypt it.

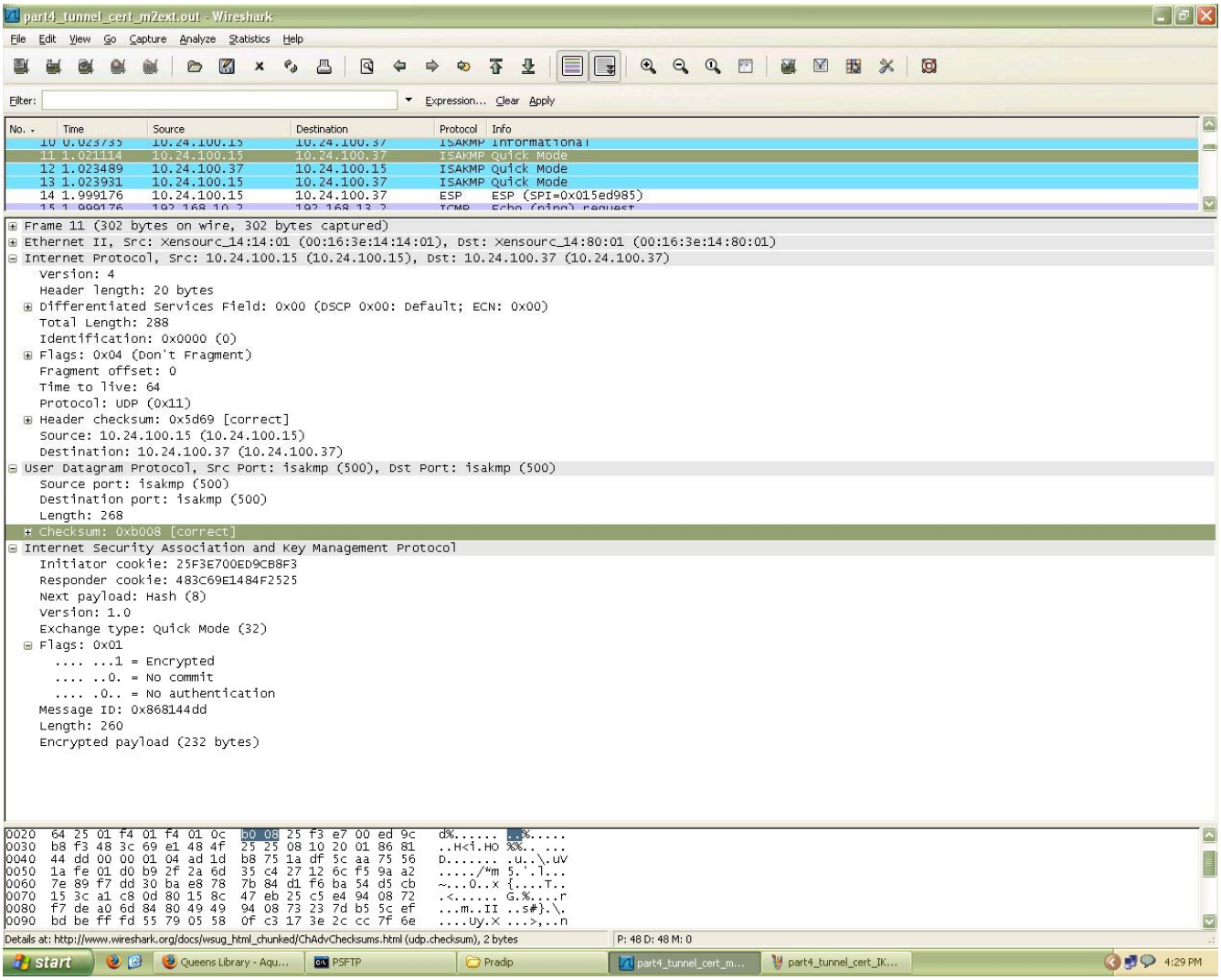
Following is a screen shot that shows this transaction.



It can be seen that the payload itself is completely encrypted this time and the encryption flag is set. Message 6 is symmetric to message 5 and is also encrypted to protect identities. This ends the first Phase of IKE and now the ISAKMP SA is setup.

Phase 2: IPSEC SA establishment (Quick Mode)

The Quick Mode is a 3- message protocol which negotiates parameters for the Phase2 SA, including cryptographic parameters and the SPI for each direction. The SPIs will then be used as identifiers for the IPSec SA. The SPIs are set automatically by racoon and we do not have to include it in the configuration files.



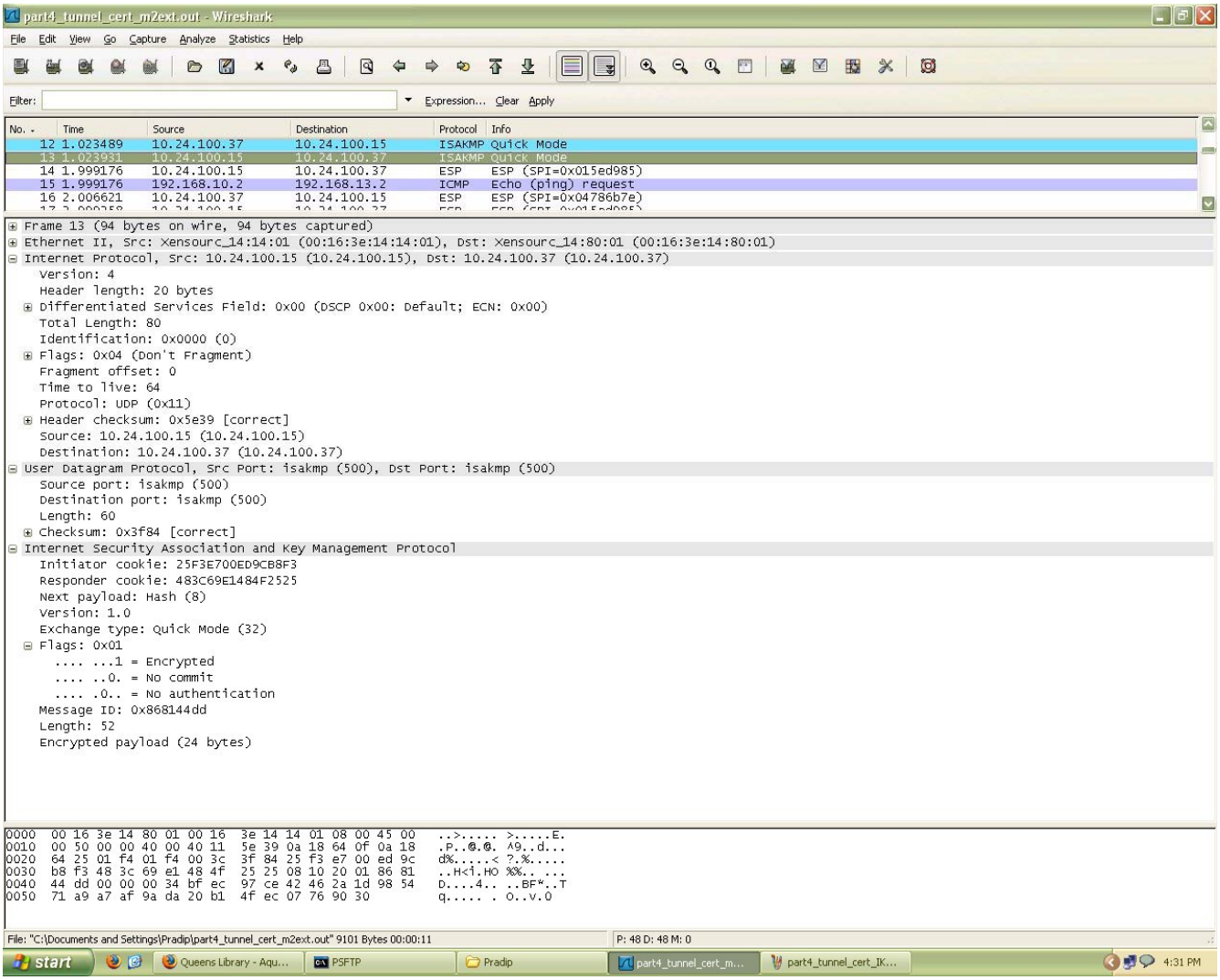
The second phase need not be initiated by the same pair of hosts that initiated the first pair. It can be started by any host to which the Secuity Policy applies to. The first message constitutes of the pair of cookies agreed upon in Phase 1, a new 32 bit number chosen by the initiator to distinguish this phase 2 setup, a nonce and the first Diffie-Hellman exponentiation, along with some traffic which consists of the new crypto proposal for the IPSec SA.

Except for the cookies which identify the ISAKMP SA, all the other information is encrypted. The cookie pair serves as the identifier for Phase 1 SA and this can be common common to multiple IPSec SAs. Similarly the new 32 bit number serves as the identifier for each of the individual IPSec SAs, allowed by the policy, as they usually share the same Phase 1 information.

The second message is very similar to the first and apart from all the usual identifiers it contains the accepted crypto proposal and the second Diffie Hellman exponentiation. The DH key exchange is to allow Perfect Forward Secrecy (PFS), but the parameters are not negotiated on the fly. Hence there is a different DH group specified in the racoon.conf for the quick mode in the line,

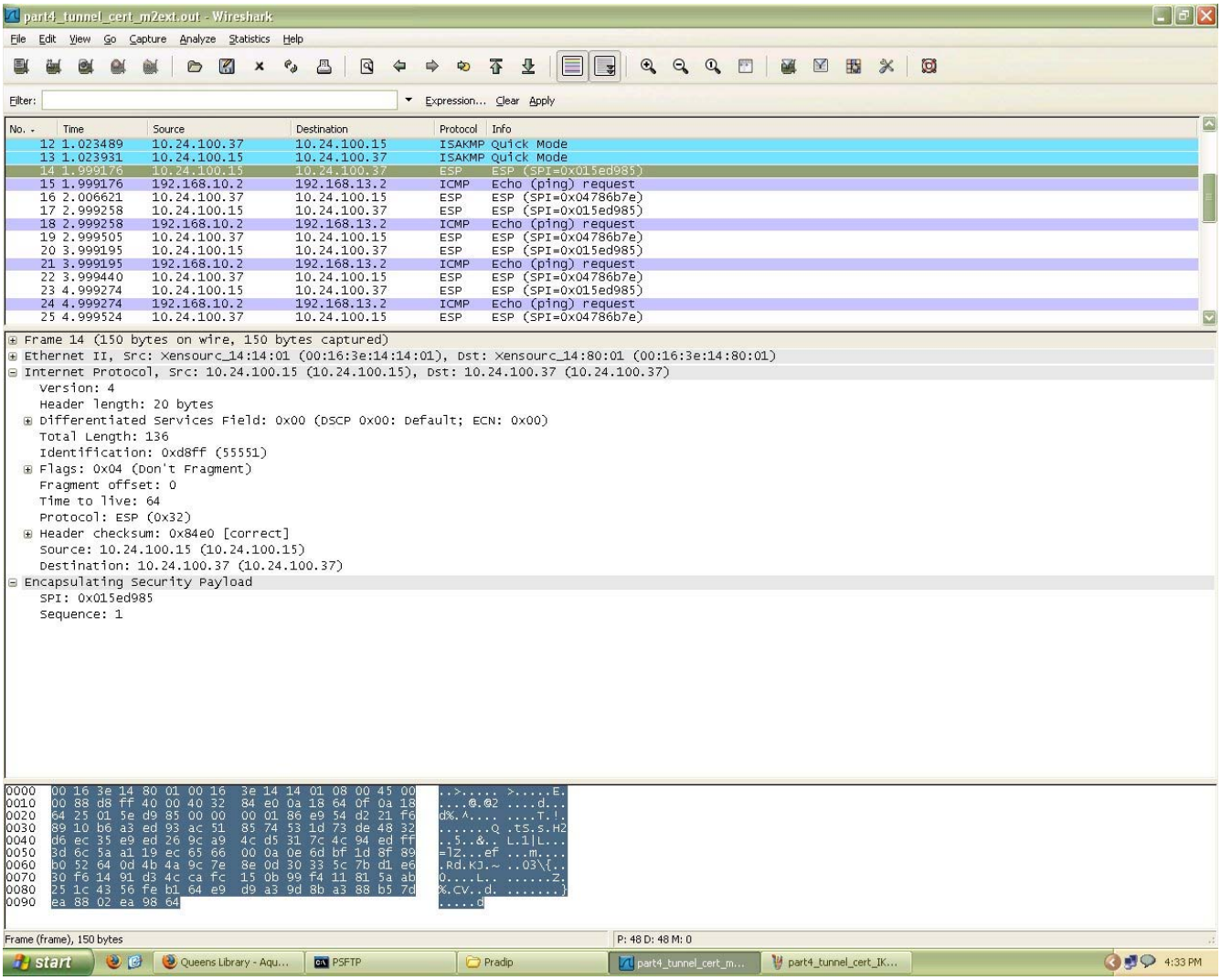
pfs_group modp768;

Message 3: The final message of Phase 2 is the acknowledgment from the initiator of the quick mode along with both the identifiers of Phase 1 and Phase 2 ie, the pair of cookies and the 32 bit identifier for Phase 2. This message marks the end of IKE and sets up the IPSec SA. The resulting keys will be used for encryption and integrity for this IPSec session.



Once this is done, the actual traffic flows between each of the hosts over the specified protocol and in our case, the ESP protocol.

The first ESP packet is shown in the following screen shot.



We can notice in the above screen shot that there are two ESP packets and one ICMP packet. The ESP packets correspond to the incoming and outgoing IPSec enabled packets . It is to be noticed that the ICMP packet is decrypted and sent to the internal nodes at the same interface. Hence, we are able to capture the incoming packet at the external interface. It is to be noted that we do not see a corresponding outgoing ICMP packet as the packet is enabled with IPSec before it reaches the external interface.

2. IPSec provides confidentiality though encryption using 3des and both integrity and authentication using HMAC and MD5 as the underlying hash function. Certificates are also used to authentication.

3. Perfect forward secrecy is a property of a protocol in which an intruder who sniffs encrypted traffic cannot later decrypt the conversation. This is achieved by using two rounds of cryptographic algorithms, one each in phase 1 and phase2. Even if the phase 1 keys are discovered, phase 2 uses different keys.

4. In preshared keys method, the keys is a function of the preshared key, sender and responder cookies, the diffie hellman key, and the nonces while it is a function of the nonces and the diffie hellman key. Certificates are more feasible in the realworld as it is not possible to have preshared keys will all hosts with which communication needs to be performed. The drawback is that the certificates need not be signed by a known Certificate Authority and as in this case can also be self signed.

5. SPI for the SA between 10.24.100.15 to 10.24.100.37 is 0x015ed985 while that for the SA between 10.24.100.24 is 0x04786b7e.

6. No the same configuration will not protect traffic flowing between the other two internal hosts. This is because, although, the other two hosts have the same respective gateways, they lie in different subnets. We can modify the configuration a little bit to allow traffic to be protected by IPSec.

Note:

The private keys were created using Openssl's *genrsa* command. Similarly the certificates were also created using other OpenSSL commands. The following screenshot shows this.