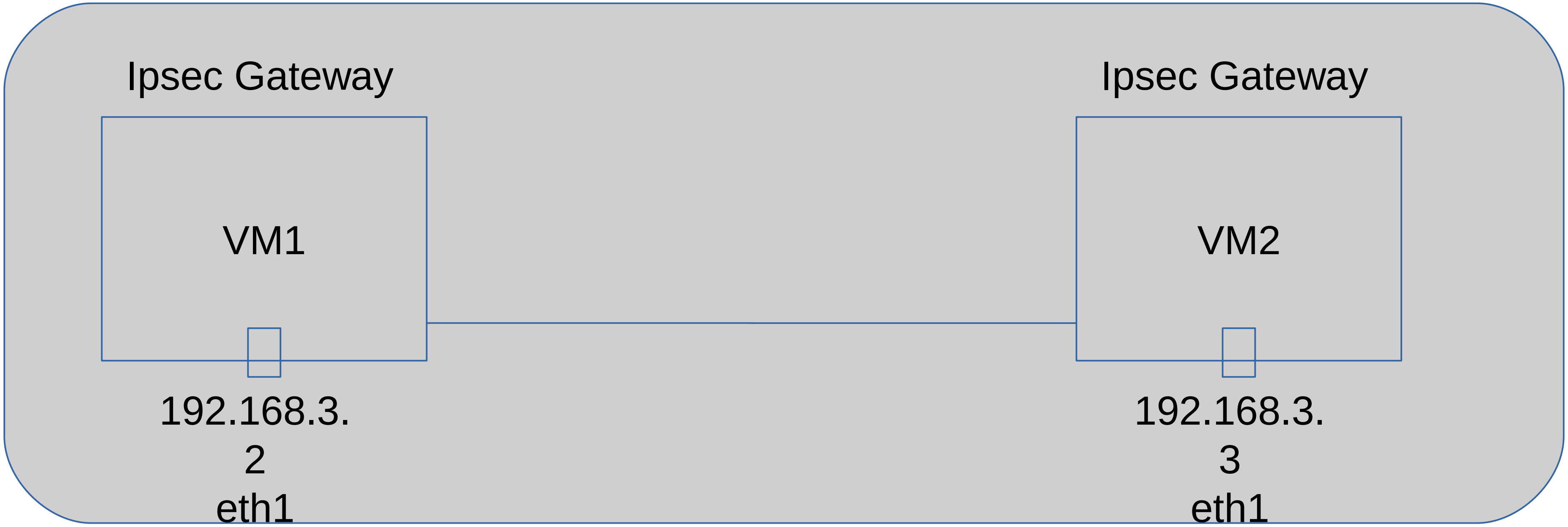
intro

- Since the UTD VPN gateway is not IPSec, so I decided to setup a vpn tunnel between to virtual machines.
- Then use the tcpdump to capture the traffic between the two virtual machines.
- Ping will be used to generate the payload

Network Topology

- Two virtual machines are instantiated for the purpose of simulate host to host ipsec traffic



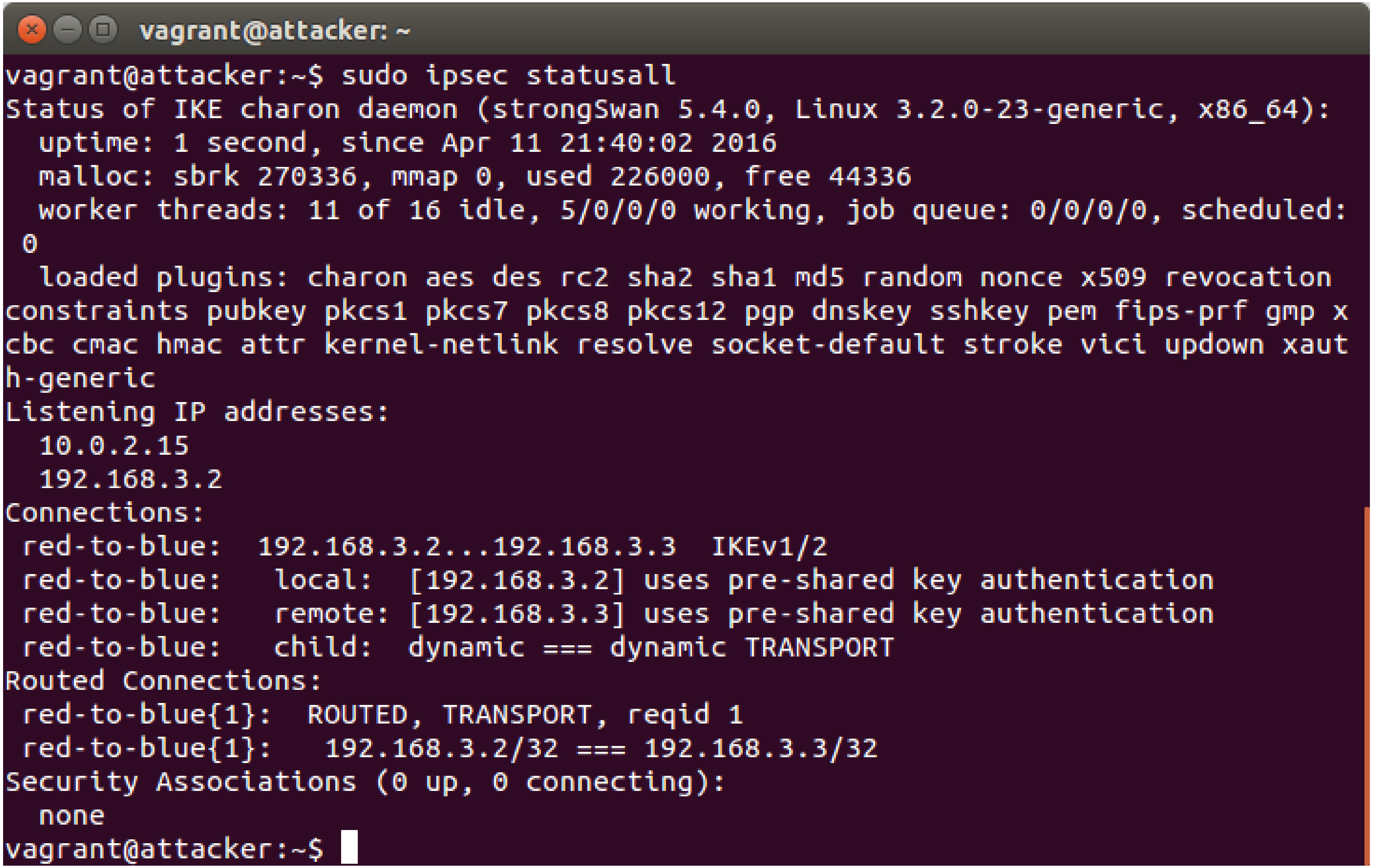
Intro to StrongSwan

- StrongSwan is a complete OpenSource IPsec-based VPN Solution providing encryption and authentication to servers and clients.
- It runs on Linux 2.6, 3.x and 4.x kernels, Android, FreeBSD, OS X and Windows

Step to verify the IPSEC traffic

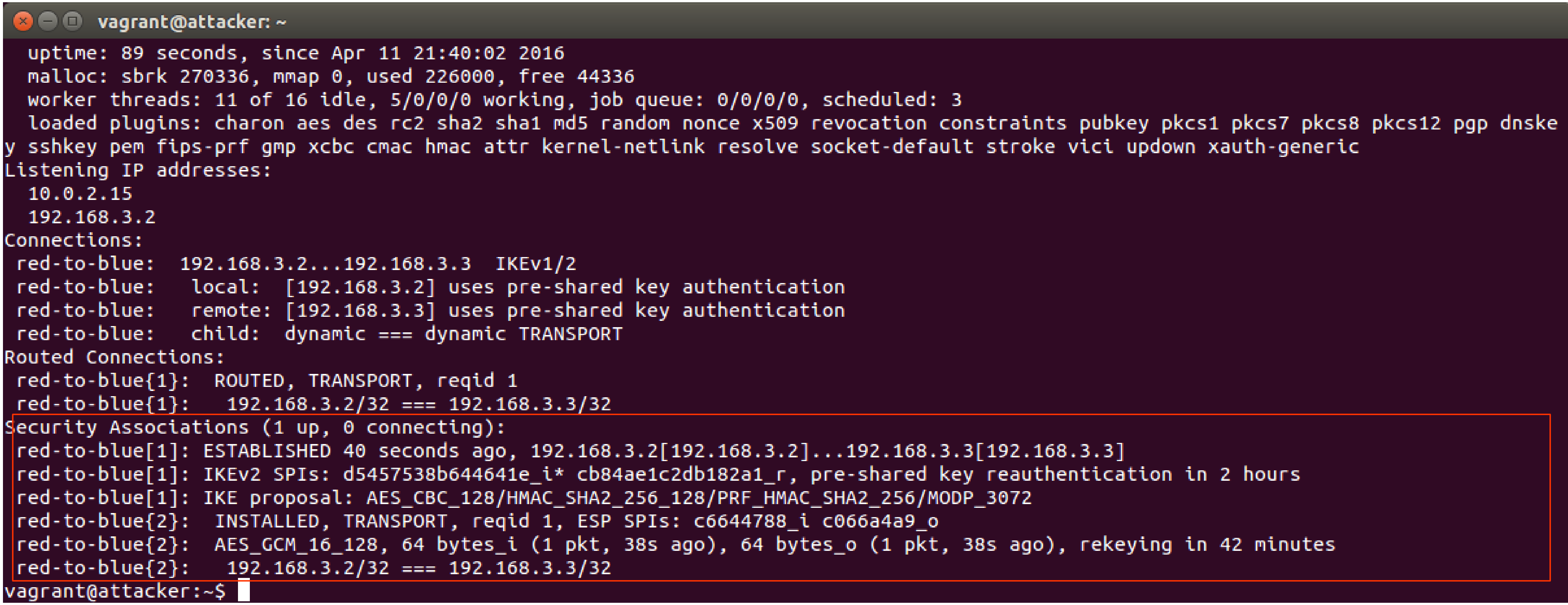
- Setup security association on VM1
- Setup security association on VM1
- Trace the I/f of vm with wireshark
- Ping VM2 from VM1
- Capture the traffic between VM1 and VM2
- End the security association

SA is down



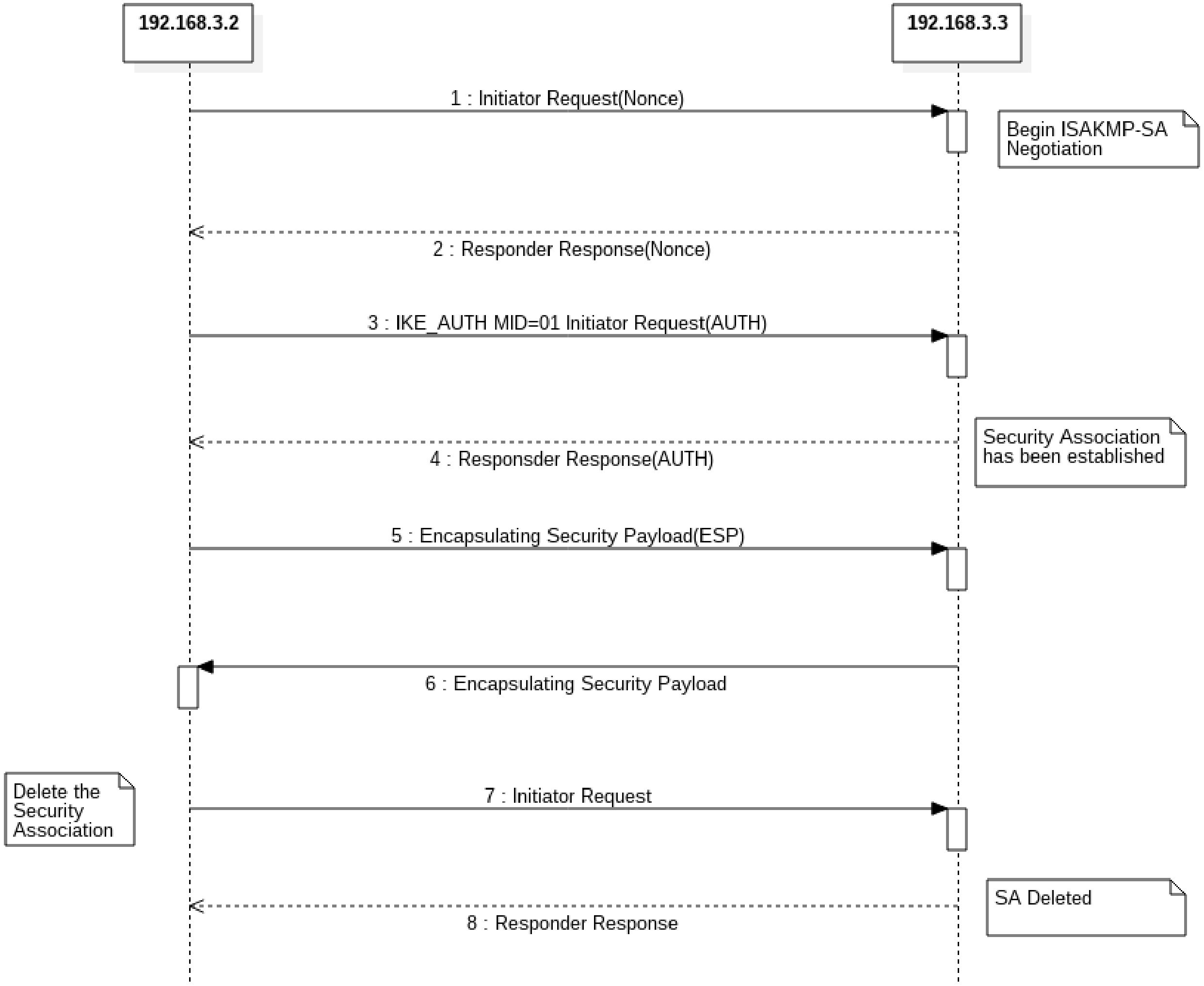
```
vagrant@attacker:~$ sudo ipsec statusall
Status of IKE charon daemon (strongSwan 5.4.0, Linux 3.2.0-23-generic, x86_64):
  uptime: 1 second, since Apr 11 21:40:02 2016
  malloc: sbrk 270336, mmap 0, used 226000, free 44336
  worker threads: 11 of 16 idle, 5/0/0/0 working, job queue: 0/0/0/0, scheduled:
 0
  loaded plugins: charon aes des rc2 sha2 sha1 md5 random nonce x509 revocation
constraints pubkey pkcs1 pkcs7 pkcs8 pkcs12 pgp dnskey sshkey pem fips-prf gmp x
cbc cmac hmac attr kernel-netlink resolve socket-default stroke vici updown xaut
h-generic
Listening IP addresses:
  10.0.2.15
  192.168.3.2
Connections:
 red-to-blue:  192.168.3.2...192.168.3.3  IKEv1/2
 red-to-blue:   local:  [192.168.3.2] uses pre-shared key authentication
 red-to-blue:   remote: [192.168.3.3] uses pre-shared key authentication
 red-to-blue:   child:  dynamic === dynamic TRANSPORT
Routed Connections:
 red-to-blue{1}:  ROUTED, TRANSPORT, reqid 1
 red-to-blue{1}:   192.168.3.2/32 === 192.168.3.3/32
Security Associations (0 up, 0 connecting):
  none
vagrant@attacker:~$
```

Check the established SA on VM

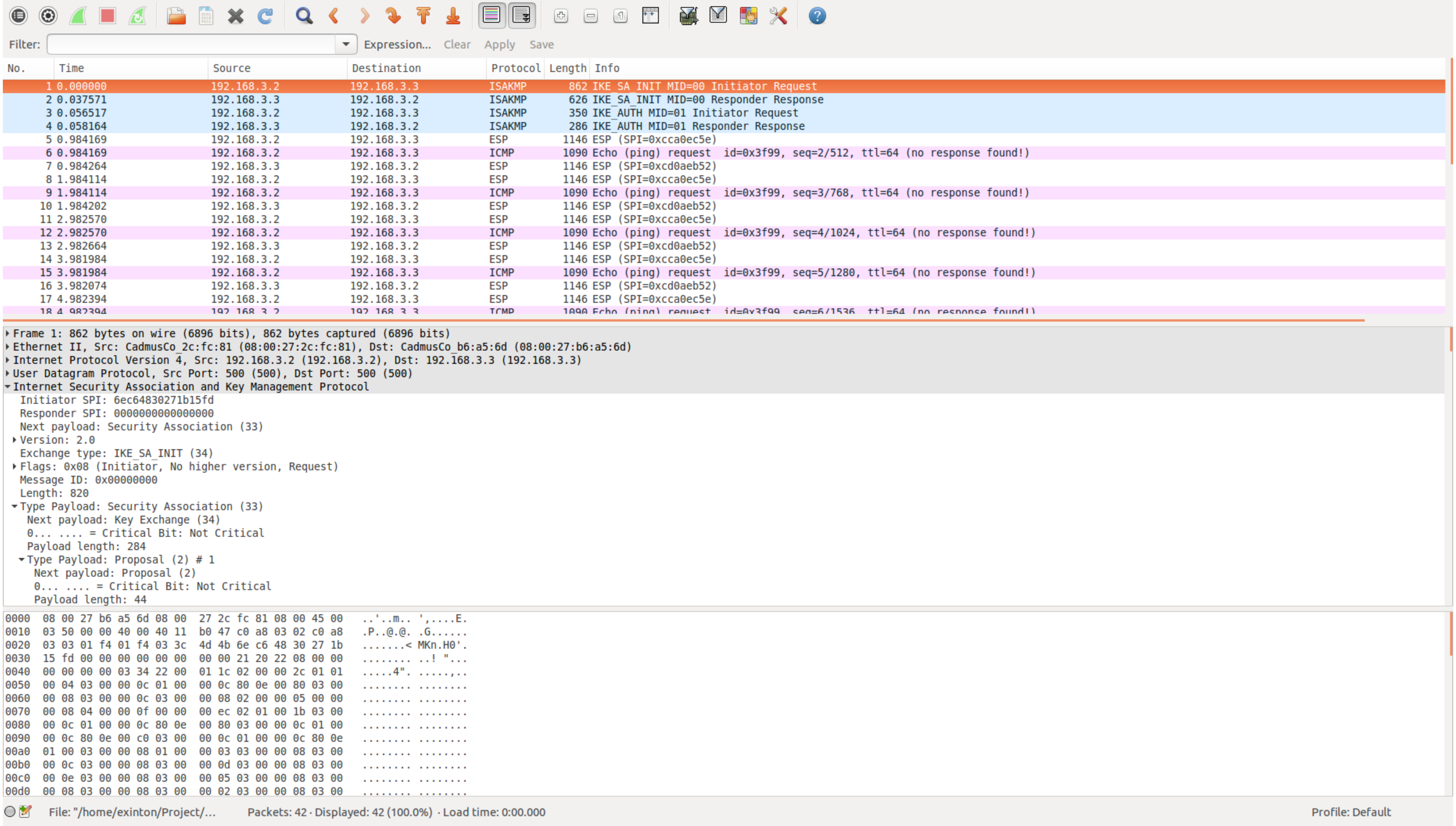


```
  uptime: 89 seconds, since Apr 11 21:40:02 2016
  malloc: sbrk 270336, mmap 0, used 226000, free 44336
  worker threads: 11 of 16 idle, 5/0/0/0 working, job queue: 0/0/0/0, scheduled: 3
  loaded plugins: charon aes des rc2 sha2 sha1 md5 random nonce x509 revocation constraints pubkey pkcs1 pkcs7 pkcs8 pkcs12 pgp dnske
y sshkey pem fips-prf gmp xcbc cmac hmac attr kernel-netlink resolve socket-default stroke vici updown xauth-generic
Listening IP addresses:
  10.0.2.15
  192.168.3.2
Connections:
 red-to-blue:  192.168.3.2...192.168.3.3  IKEv1/2
 red-to-blue:   local:  [192.168.3.2] uses pre-shared key authentication
 red-to-blue:   remote: [192.168.3.3] uses pre-shared key authentication
 red-to-blue:   child:  dynamic === dynamic TRANSPORT
Routed Connections:
 red-to-blue{1}:  ROUTED, TRANSPORT, reqid 1
 red-to-blue{1}:   192.168.3.2/32 === 192.168.3.3/32
Security Associations (1 up, 0 connecting):
 red-to-blue[1]: ESTABLISHED 40 seconds ago, 192.168.3.2[192.168.3.2]...192.168.3.3[192.168.3.3]
 red-to-blue[1]: IKEv2 SPIs: d5457538b644641e_i* cb84ae1c2db182a1_r, pre-shared key reauthentication in 2 hours
 red-to-blue[1]: IKE proposal: AES_CBC_128/HMAC_SHA2_256_128/PRF_HMAC_SHA2_256/MODP_3072
 red-to-blue{2}:  INSTALLED, TRANSPORT, reqid 1, ESP SPIs: c6644788_i c066a4a9_o
 red-to-blue{2}:  AES_GCM_16_128, 64 bytes_i (1 pkt, 38s ago), 64 bytes_o (1 pkt, 38s ago), rekeying in 42 minutes
 red-to-blue{2}:   192.168.3.2/32 === 192.168.3.3/32
vagrant@attacker:~$
```

CALL FLOW between VM


192.168.3.2
192.168.3.3
1 : Initiator Request(Nonce)
Begin ISAKMP-SA Negotiation
2 : Responder Response(Nonce)
3 : IKE_AUTH MID=01 Initiator Request(AUTH)
Security Association has been established
4 : Responsder Response(AUTH)
5 : Encapsulating Security Payload(ESP)
6 : Encapsulating Security Payload
Delete the Security Association
7 : Initiator Request
SA Deleted
8 : Responder Response

Wireshark screen capture



```
No.  Time        Source         Destination    Protocol Length Info
   1 0.000000    192.168.3.2    192.168.3.3    ISAKMP    862 IKE_SA_INIT MID=00 Initiator Request
   2 0.037571    192.168.3.3    192.168.3.2    ISAKMP    626 IKE_SA_INIT MID=00 Responder Response
   3 0.056517    192.168.3.2    192.168.3.3    ISAKMP    350 IKE_AUTH MID=01 Initiator Request
   4 0.058164    192.168.3.3    192.168.3.2    ISAKMP    286 IKE_AUTH MID=01 Responder Response
   5 0.984169    192.168.3.2    192.168.3.3    ESP      1146 ESP (SPI=0xcca0ec5e)
   6 0.984169    192.168.3.2    192.168.3.3    ICMP     1090 Echo (ping) request  id=0x3f99, seq=2/512, ttl=64 (no response found!)
   7 0.984264    192.168.3.3    192.168.3.2    ESP      1146 ESP (SPI=0xcd0aeb52)
   8 1.984114    192.168.3.2    192.168.3.3    ESP      1146 ESP (SPI=0xcca0ec5e)
   9 1.984114    192.168.3.2    192.168.3.3    ICMP     1090 Echo (ping) request  id=0x3f99, seq=3/768, ttl=64 (no response found!)
  10 1.984202    192.168.3.3    192.168.3.2    ESP      1146 ESP (SPI=0xcd0aeb52)
  11 2.982570    192.168.3.2    192.168.3.3    ESP      1146 ESP (SPI=0xcca0ec5e)
  12 2.982570    192.168.3.2    192.168.3.3    ICMP     1090 Echo (ping) request  id=0x3f99, seq=4/1024, ttl=64 (no response found!)
  13 2.982664    192.168.3.3    192.168.3.2    ESP      1146 ESP (SPI=0xcd0aeb52)
  14 3.981984    192.168.3.2    192.168.3.3    ESP      1146 ESP (SPI=0xcca0ec5e)
  15 3.981984    192.168.3.2    192.168.3.3    ICMP     1090 Echo (ping) request  id=0x3f99, seq=5/1280, ttl=64 (no response found!)
  16 3.982074    192.168.3.3    192.168.3.2    ESP      1146 ESP (SPI=0xcd0aeb52)
  17 4.982394    192.168.3.2    192.168.3.3    ESP      1146 ESP (SPI=0xcca0ec5e)
  18 4.982394    192.168.3.2    192.168.3.3    ICMP     1090 Echo (ping) request  id=0x3f99, seq=6/1536, ttl=64 (no response found!)
```

```
▸ Frame 1: 862 bytes on wire (6896 bits), 862 bytes captured (6896 bits)
▸ Ethernet II, Src: CadmusCo_2c:fc:81 (08:00:27:2c:fc:81), Dst: CadmusCo_b6:a5:6d (08:00:27:b6:a5:6d)
▸ Internet Protocol Version 4, Src: 192.168.3.2 (192.168.3.2), Dst: 192.168.3.3 (192.168.3.3)
▸ User Datagram Protocol, Src Port: 500 (500), Dst Port: 500 (500)
▾ Internet Security Association and Key Management Protocol
    Initiator SPI: 6ec64830271b15fd
    Responder SPI: 0000000000000000
    Next payload: Security Association (33)
  ▸ Version: 2.0
    Exchange type: IKE_SA_INIT (34)
  ▸ Flags: 0x08 (Initiator, No higher version, Request)
    Message ID: 0x00000000
    Length: 820
  ▾ Type Payload: Security Association (33)
      Next payload: Key Exchange (34)
      0... .... = Critical Bit: Not Critical
      Payload length: 284
    ▾ Type Payload: Proposal (2) # 1
        Next payload: Proposal (2)
        0... .... = Critical Bit: Not Critical
        Payload length: 44
```

```
0000  08 00 27 b6 a5 6d 08 00  27 2c fc 81 08 00 45 00   ..'..m.. ',....E.
0010  03 50 00 00 40 00 40 11  b0 47 c0 a8 03 02 c0 a8   .P..@.@. .G......
0020  03 03 01 f4 01 f4 03 3c  4d 4b 6e c6 48 30 27 1b   .......< MKn.H0'.
0030  15 fd 00 00 00 00 00 00  00 00 21 20 22 08 00 00   ........ ..! "...
0040  00 00 00 00 03 34 22 00  01 1c 02 00 00 2c 01 01   .....4". .....,..
0050  00 04 03 00 00 0c 01 00  00 0c 80 0e 00 80 03 00   ........ ........
0060  00 08 03 00 00 0c 03 00  00 08 02 00 00 05 00 00   ........ ........
0070  00 08 04 00 00 0f 00 00  00 ec 02 01 00 1b 03 00   ........ ........
0080  00 0c 01 00 00 0c 80 0e  00 80 03 00 00 0c 01 00   ........ ........
0090  00 0c 80 0e 00 c0 03 00  00 0c 01 00 00 0c 80 0e   ........ ........
00a0  01 00 03 00 00 08 01 00  00 03 03 00 00 08 03 00   ........ ........
00b0  00 0c 03 00 00 08 03 00  00 0d 03 00 00 08 03 00   ........ ........
00c0  00 0e 03 00 00 08 03 00  00 05 03 00 00 08 03 00   ........ ........
00d0  00 08 03 00 00 08 03 00  00 02 03 00 00 08 03 00   ........ ........
```

File: "/home/exinton/Project/...   Packets: 42 · Displayed: 42 (100.0%) · Load time: 0:00.000   Profile: Default

when the secure channel to be used by the host

- When the client wants to visit a remote network securely, then VPN connection between client and remote network is a good choice.
- The host should be able to forward his traffic to the remote network to the VPN gateway instead.
- The Gateway will authenticate the client
- All the messages between client and VPN are encrypted by either AH or ESP.

when the secure channel to be used by the VPN gateway

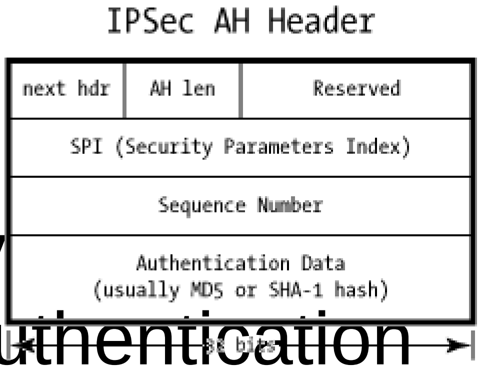
- The VPN receive the encrypted msg from the remote client, and then decrypted it.
- As long as VPN gateway decrypted the IP message, it could get the orignial information.
- Normally the VPN gateway works at the tunnel mode, which means it will forward the package inside the remote network.
- Two host could initial transparent mode inbetween.

How is the info protected by the IPSec between two VM?

- The VPN gateway will authenticate the client at the IPSec negotiation phase
- Both end send a nonce to opposite, then use their shared key to calculate a hash value as AUTH.
- After both side receive AUTH from opposite, the compare the received one with their own value to authenticate the opposite end.
-

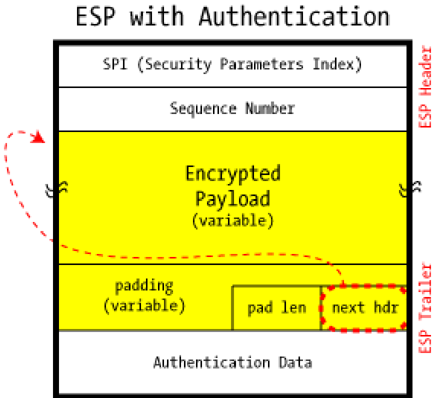
Protect the integrity and credentiality

- AH:
  - -encrypt the payload for confidentiality
  - Authentication data for Integrity and authentication
- ESP
  - Encapsulating the payload to provide confidentiality
  - In addition to encryption, ESP can also optionally provide authentication, with the same HMAC as found in AH


IPSec AH Header

next hdr	AH len	Reserved
SPI (Security Parameters Index)		
Sequence Number		
Authentication Data (usually MD5 or SHA-1 hash)		

32 bits


ESP with Authentication

SPI (Security Parameters Index)

Sequence Number

Encrypted Payload (variable)

padding (variable)

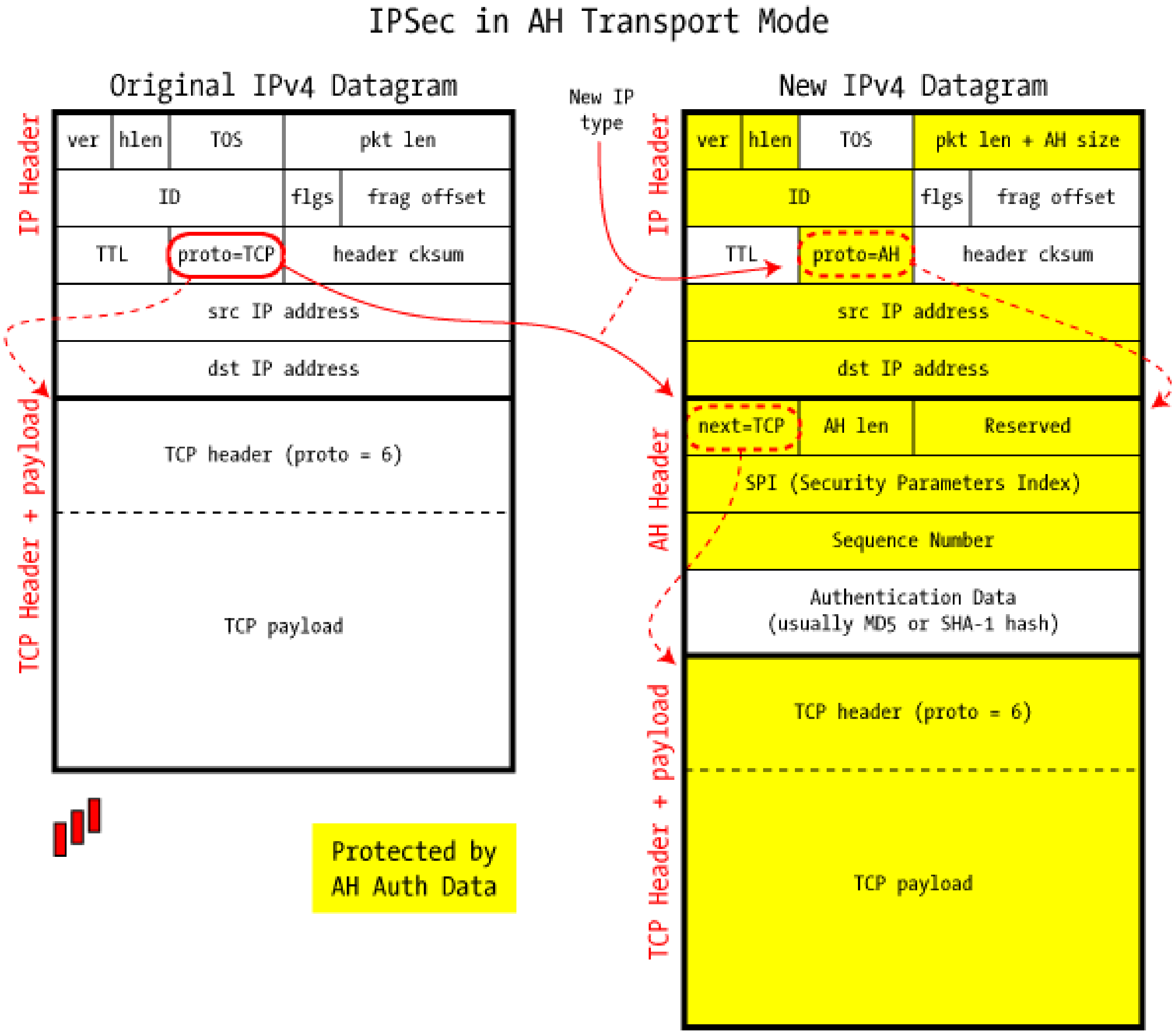
pad len

next hdr

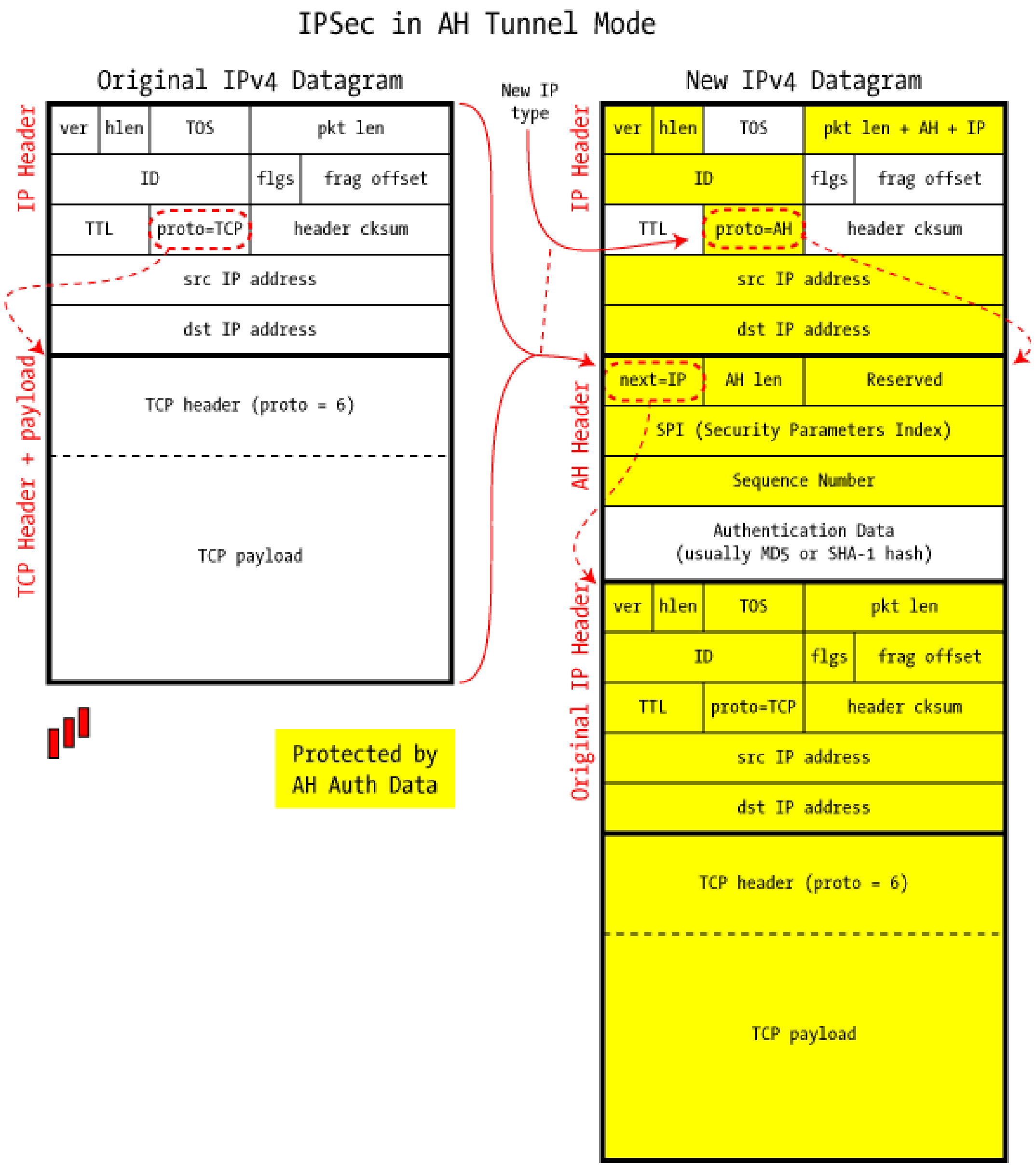
Authentication Data

In AH transport Mode,most headers are protected by authentication data

- The authentication data is hashed with secret shared by client and vpn gateway.



- The IP header between client and VPN gateway are encrypted



IPSec in AH Tunnel Mode

Original IPv4 Datagram

IP Header

ver | hlen | TOS | pkt len

ID | flgs | frag offset

TTL | proto=TCP | header cksum

src IP address

dst IP address

TCP Header + payload

TCP header (proto = 6)

TCP payload

New IP type

New IPv4 Datagram

IP Header

ver | hlen | TOS | pkt len + AH + IP

ID | flgs | frag offset

TTL | proto=AH | header cksum

src IP address

dst IP address

AH Header

next=IP | AH len | Reserved

SPI (Security Parameters Index)

Sequence Number

Authentication Data
(usually MD5 or SHA-1 hash)

Original IP Header

ver | hlen | TOS | pkt len

ID | flgs | frag offset

TTL | proto=TCP | header cksum

src IP address

dst IP address

TCP header (proto = 6)

TCP payload

Protected by
AH Auth Data