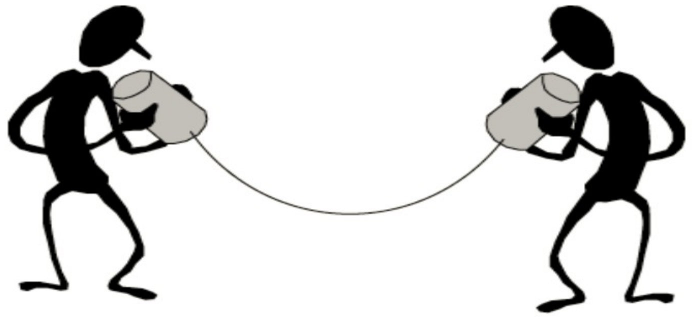
Intel 10Gbe status and other thoughts

Linux IPsec Workshop 2018

- Shannon Nelson
- Oracle Corp
- March 2018

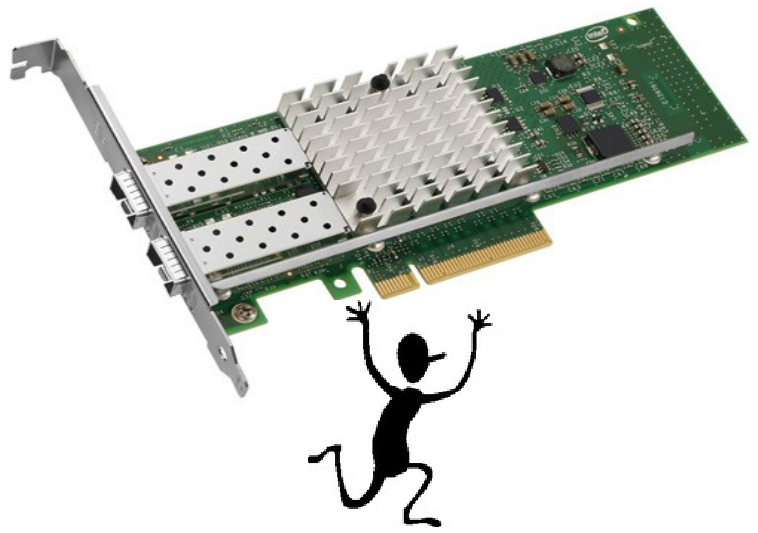


Summary

- 10Gbe Niantic and family have IPsec HW offload
- Initial driver support came out in v4.15
  - Approx 6.5 Gbps
- Recent patches released to work with TSO and Checksum offload
  - Approx 9.1 Gbps

Niantic Family

- Intel's 10Gbe
  - 82599/x520, x540, x550
  - Followup to Oplin 82598, no IPsec
- Initial release around 2009, included IPsec HW circuits
  - Windows PROset driver support included IPsec offload early on
- Oracle
  - Sold many Database platforms with Niantic built in
  - Many of Oracle Cloud servers have Niantic
  - Lots of customers with Data Security needs



Niantic IPsec Features

- IPv4 and IPv6
- aead with 128bit rfc4106(gcm(aes))
- 1024 SAs with 256 IP addresses
- Checksum and TSO offload
- RSS on decrypted contents
- Nearly line-rate performance

Niantic IPsec Missing Features

- No additional encryptions, only 128bit rfc4106(gcm(aes))
- No indication of which SA was decoded on Rx
- No ESN

Performance

- Almost line rate with TSO and Checksum offload
- … not a lot of performance testing yet

Performance – lock management

```
Sowmini's microbenchmark for the Lock Management Server

 One client and one server: the client sends a 112 byte
 request and the server sends back a 512K byte response.

Results (averaged over 6 trials):
---------------------------------
   clear traffic:          1272 messages/sec    latency 394 us
   ipsec + h/w offload     1246 messages/sec    latency 402 us
   ipsec + s/w offload      597 messages/sec    latency 839 us

So the ipsec case can now match the clear traffic case.
```

Performance – simple iperf

iperf -c 14.0.0.70 -t 60 → iperf -s

Clear traffic

```
0.0-60.0 sec  65.7 GBytes  9.41 Gbits/sec
```

IPsec hw offload

```
0.0-60.0 sec  64.2 GBytes  9.19 Gbits/sec
```

IPsec sw

```
0.0-60.1 sec  3.18 GBytes   455 Mbits/sec
```

Odd Performance thing – parallel iperf

iperf -c 14.0.0.70 -t 60 → iperf -s

Clear traffic

```
0.0-60.0 sec  65.7 GBytes  9.41 Gbits/sec
```

IPsec hw offload

```
0.0-60.0 sec  64.2 GBytes  9.19 Gbits/sec
```

IPsec sw

```
0.0-60.1 sec  3.18 GBytes  455 Mbits/sec
```

iperf -c 14.0.0.70 -t 60 -P 4 → iperf -s

Clear traffic

```
0.0-60.0 sec  65.8 GBytes  9.41 Gbits/sec
```

IPsec hw offload

```
0.0-60.0 sec  46.6 GBytes  6.67 Gbits/sec
```

IPsec sw

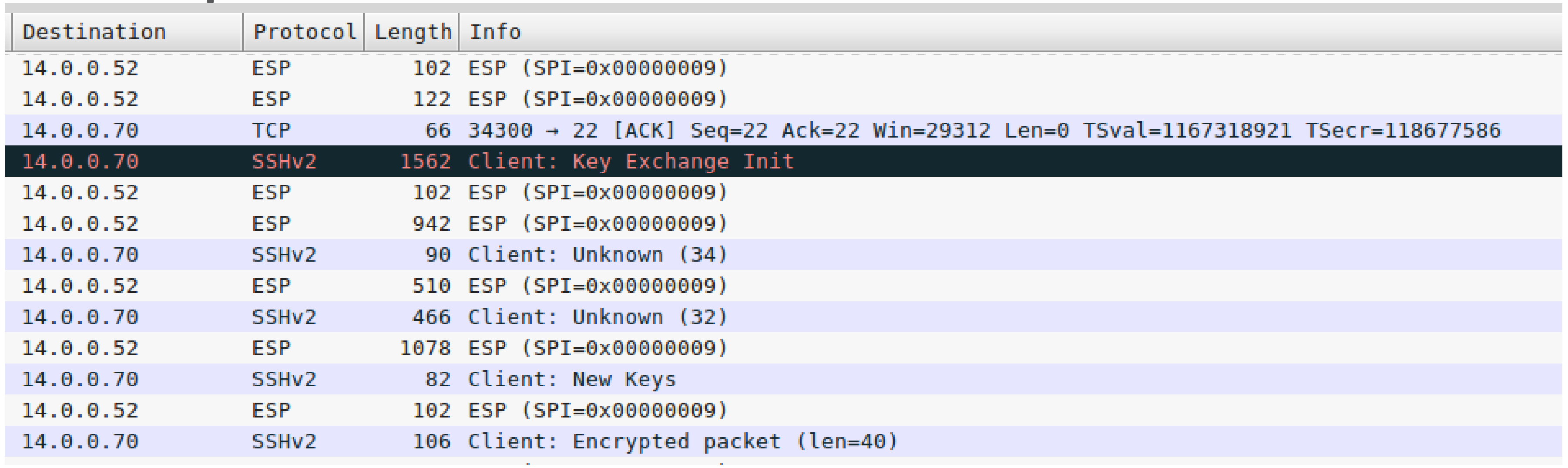
```
0.0-60.1 sec  3.13 GBytes  448 Mbits/sec
```

Out of Order GSO packets

- Seen only when NETIF_F_GSO_ESP is not used in driver
- 2nd half of GSO packet received before 1st half
- Inconsistent – doesn't always happen
- Can be seen occasionally in startup of simple ssh connection
    - Use driver with no NETIF_F_GSO_ESP on <src>
    - Set up ipsec connection between <src> and <dst>
    - Start tcpdump on <dst>
    - Run “ssh <dst>” on src (may need to try several times)
    - Watch netstat for segments retransmited
    - Tcpdump/Wireshark will point out [TCP Out-Of-Order]

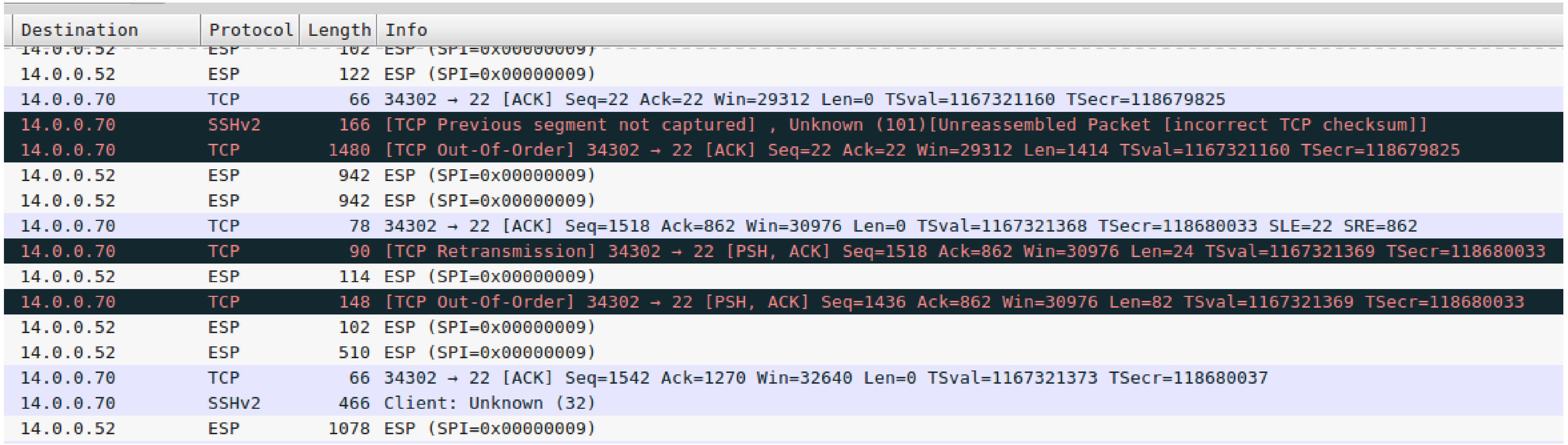
Out of Order GSO packets

Good



```
Destination         Protocol  Length  Info
14.0.0.52           ESP          102  ESP (SPI=0x00000009)
14.0.0.52           ESP          122  ESP (SPI=0x00000009)
14.0.0.70           TCP           66  34300 → 22 [ACK] Seq=22 Ack=22 Win=29312 Len=0 TSval=1167318921 TSecr=118677586
14.0.0.70           SSHv2       1562  Client: Key Exchange Init
14.0.0.52           ESP          102  ESP (SPI=0x00000009)
14.0.0.52           ESP          942  ESP (SPI=0x00000009)
14.0.0.70           SSHv2         90  Client: Unknown (34)
14.0.0.52           ESP          510  ESP (SPI=0x00000009)
14.0.0.70           SSHv2        466  Client: Unknown (32)
14.0.0.52           ESP         1078  ESP (SPI=0x00000009)
14.0.0.70           SSHv2         82  Client: New Keys
14.0.0.52           ESP          102  ESP (SPI=0x00000009)
14.0.0.70           SSHv2        106  Client: Encrypted packet (len=40)
```

Bad



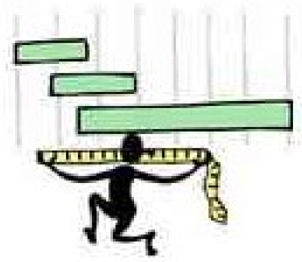
```
Destination         Protocol  Length  Info
14.0.0.52           ESP          102  ESP (SPI=0x00000009)
14.0.0.52           ESP          122  ESP (SPI=0x00000009)
14.0.0.70           TCP           66  34302 → 22 [ACK] Seq=22 Ack=22 Win=29312 Len=0 TSval=1167321160 TSecr=118679825
14.0.0.70           SSHv2        166  [TCP Previous segment not captured] , Unknown (101)[Unreassembled Packet [incorrect TCP checksum]]
14.0.0.70           TCP         1480  [TCP Out-Of-Order] 34302 → 22 [ACK] Seq=22 Ack=22 Win=29312 Len=1414 TSval=1167321160 TSecr=118679825
14.0.0.52           ESP          942  ESP (SPI=0x00000009)
14.0.0.52           ESP          942  ESP (SPI=0x00000009)
14.0.0.70           TCP           78  34302 → 22 [ACK] Seq=1518 Ack=862 Win=30976 Len=0 TSval=1167321368 TSecr=118680033 SLE=22 SRE=862
14.0.0.70           TCP           90  [TCP Retransmission] 34302 → 22 [PSH, ACK] Seq=1518 Ack=862 Win=30976 Len=24 TSval=1167321369 TSecr=118680033
14.0.0.52           ESP          114  ESP (SPI=0x00000009)
14.0.0.70           TCP          148  [TCP Out-Of-Order] 34302 → 22 [PSH, ACK] Seq=1436 Ack=862 Win=30976 Len=82 TSval=1167321369 TSecr=118680033
14.0.0.52           ESP          102  ESP (SPI=0x00000009)
14.0.0.52           ESP          510  ESP (SPI=0x00000009)
14.0.0.70           TCP           66  34302 → 22 [ACK] Seq=1542 Ack=1270 Win=32640 Len=0 TSval=1167321373 TSecr=118680037
14.0.0.70           SSHv2        466  Client: Unknown (32)
14.0.0.52           ESP         1078  ESP (SPI=0x00000009)
```

To Do

- Look into parallel performance issue
- Resolve xfrm/gso issue seen for drivers without TSO
- Look into tunnel support in ixgbe-ipsec
- Fix up kernel documentation
  - Documentation/networking/ipsec.txt is rather meager

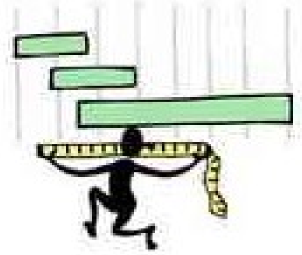
FlowDirector

- More specific conversation routing than RSS
  - "All ip4 traffic from XX to YY shall go to Rx queue Z"
  - "All tcp traffic from source port 52790 shall go to Rx queue 14"



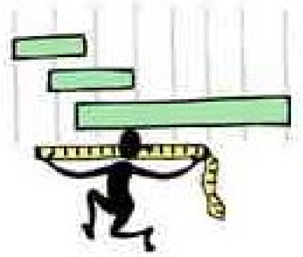
FlowDirector

- More specific conversation routing than RSS
  - “All ip4 traffic from XX to YY shall go to Rx queue Z”
  - “All tcp traffic from source port 52790 shall go to Rx queue 14”
- Basic sorting rules work on IPsec offload (decrypted) packets
  - `ethtool -U eth4 flow-type ip4 dst-ip 14.0.0.70 src-ip 14.0.0.52 action 14`
  - `ethtool -U eth4 flow-type tcp4 src-port 52790 action 14`
  - `ethtool -U eth4 flow-type tcp4 dst-ip 14.0.0.70 src-ip 14.0.0.52 src-port 52778 action 11`



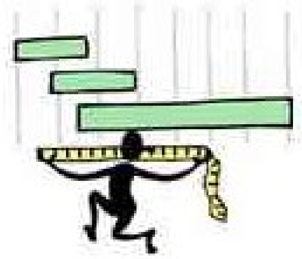
FlowDirector

- More specific conversation routing than RSS
  - "All ip4 traffic from XX to YY shall go to Rx queue Z"
  - "All tcp traffic from source port 52790 shall go to Rx queue 14"
- Basic sorting rules work on IPsec offload (decrypted) packets
  - ethtool -U eth4 flow-type ip4 dst-ip 14.0.0.70 src-ip 14.0.0.52 action 14
  - ethtool -U eth4 flow-type tcp4 src-port 52790 action 14
  - ethtool -U eth4 flow-type tcp4 dst-ip 14.0.0.70 src-ip 14.0.0.52 src-port 52778 action 11
- No support for ESP fields
  - Only IPv4/6 addrs, UDP/TCP ports, SCTP, vlan



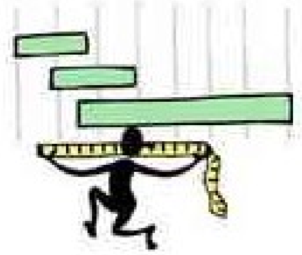
FlowDirector – FlexBytes?

- Programmable 2-byte selection anywhere in first 64 header bytes



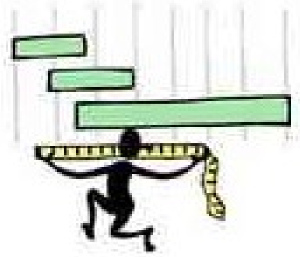
FlowDirector – FlexBytes?

- Programmable 2-byte selection anywhere in first 64 header bytes
- Currently used to implement vlan-etype in ethtool rule command
  - ethtool -U eth0 flow-type ip4 **vlan-etype 0x88a8** action -1



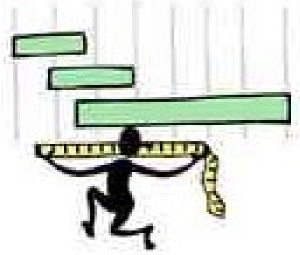
FlowDirector – FlexBytes?

- Programmable 2-byte selection anywhere in first 64 header bytes
- Currently used to implement vlan-etype in ethtool rule command
  - ethtool -U eth0 flow-type ip4 **vlan-etype 0x88a8** action -1
- Trade vlan-etype rules for SPI or other rules?
  - Experimental hacked patch works, but …
  - Only 1 flexbyte config can be set, is used by all flexbyte rules
  - ethtool's userdef tag is already used for selecting VMs
  - 2 byte filter may not be enough to be useful



FlowDirector – FlexBytes?

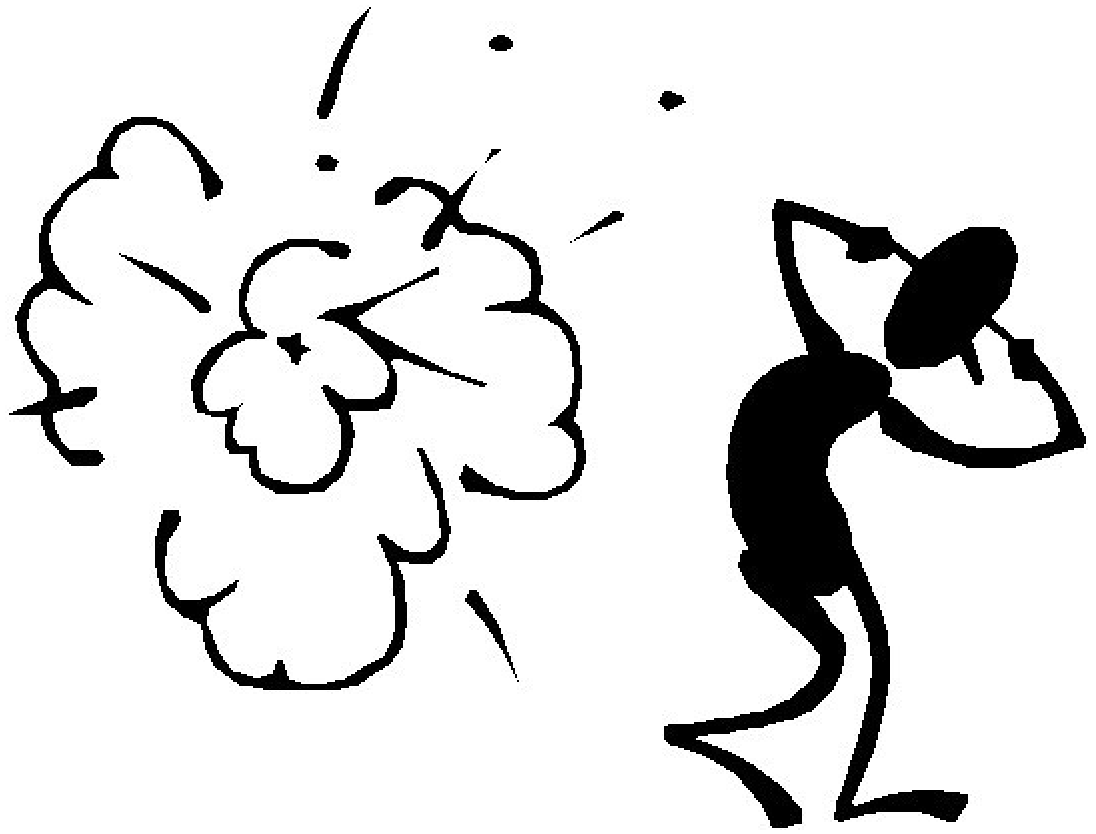
- Programmable 2-byte selection anywhere in first 64 header bytes
- Currently used to implement vlan-etype in ethtool rule command
  - ethtool -U eth0 flow-type ip4 **vlan-etype 0x88a8** action -1
- Trade vlan-etype rules for SPI or other rules?
  - Experimental hacked patch works, but …
  - Only 1 flexbyte config can be set, is used by all flexbyte rules
  - ethtool's userdef tag is already used for selecting VMs
  - 2 byte filter may not be enough to be useful
- Any interest?



My Questions

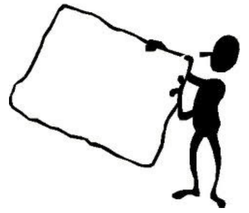
- What are the common encryptions used for IPsec?
- What encryptions should we be asking of our hardware vendors?
- What vendors have IPsec offload now, and who has future products coming?
- How to support IPsec offload for VFs?

Questions?



http://weclipart.com/screen+bean+people+clipart

Test setup example – net.all



- Left:
  - ip x p add dir out src 14.0.0.52/24 dst 14.0.0.70/24 tmpl proto esp src 14.0.0.52 dst 14.0.0.70 spi 0x07 mode transport reqid 0x07
  - ip x p add dir in src 14.0.0.70/24 dst 14.0.0.52/24 tmpl proto esp dst 14.0.0.52 src 14.0.0.70 spi 0x07 mode transport reqid 0x07
  - ip x s add proto esp src 14.0.0.52 dst 14.0.0.70 spi 0x07 mode transport reqid 0x07 replay-window 32 aead 'rfc4106(gcm(aes))' 1234567890123456dcba 128 sel src 14.0.0.52/24 dst 14.0.0.70/24 **offload dev eth4 dir out**
  - ip x s add proto esp dst 14.0.0.52 src 14.0.0.70 spi 0x07 mode transport reqid 0x07 replay-window 32 aead 'rfc4106(gcm(aes))' 1234567890123456dcba 128 sel src 14.0.0.70/24 dst 14.0.0.52/24 **offload dev eth4 dir out**
- Right:
  - ip x p add dir out src 14.0.0.70/24 dst 14.0.0.52/24 tmpl proto esp src 14.0.0.70 dst 14.0.0.52 spi 0x07 mode transport reqid 0x07
  - ip x p add dir in src 14.0.0.52/24 dst 14.0.0.70/24 tmpl proto esp dst 14.0.0.70 src 14.0.0.52 spi 0x07 mode transport reqid 0x07
  - ip x s add proto esp src 14.0.0.70 dst 14.0.0.52 spi 0x07 mode transport reqid 0x07 replay-window 32 aead 'rfc4106(gcm(aes))' 1234567890123456dcba 128 sel src 14.0.0.70/24 dst 14.0.0.52/24 **offload dev eth4 dir out**
  - ip x s add proto esp dst 14.0.0.70 src 14.0.0.52 spi 0x07 mode transport reqid 0x07 replay-window 32 aead 'rfc4106(gcm(aes))' 1234567890123456dcba 128 sel src 14.0.0.52/24 dst 14.0.0.70/24 **offload dev eth4 dir in**