# IPSec & OpenVPN

Emil Bureacă

# Contents

1. Context

2. IPSec
   - Authentication Header
   - Encapsulating Security Payload
   - Internet Key Exchange

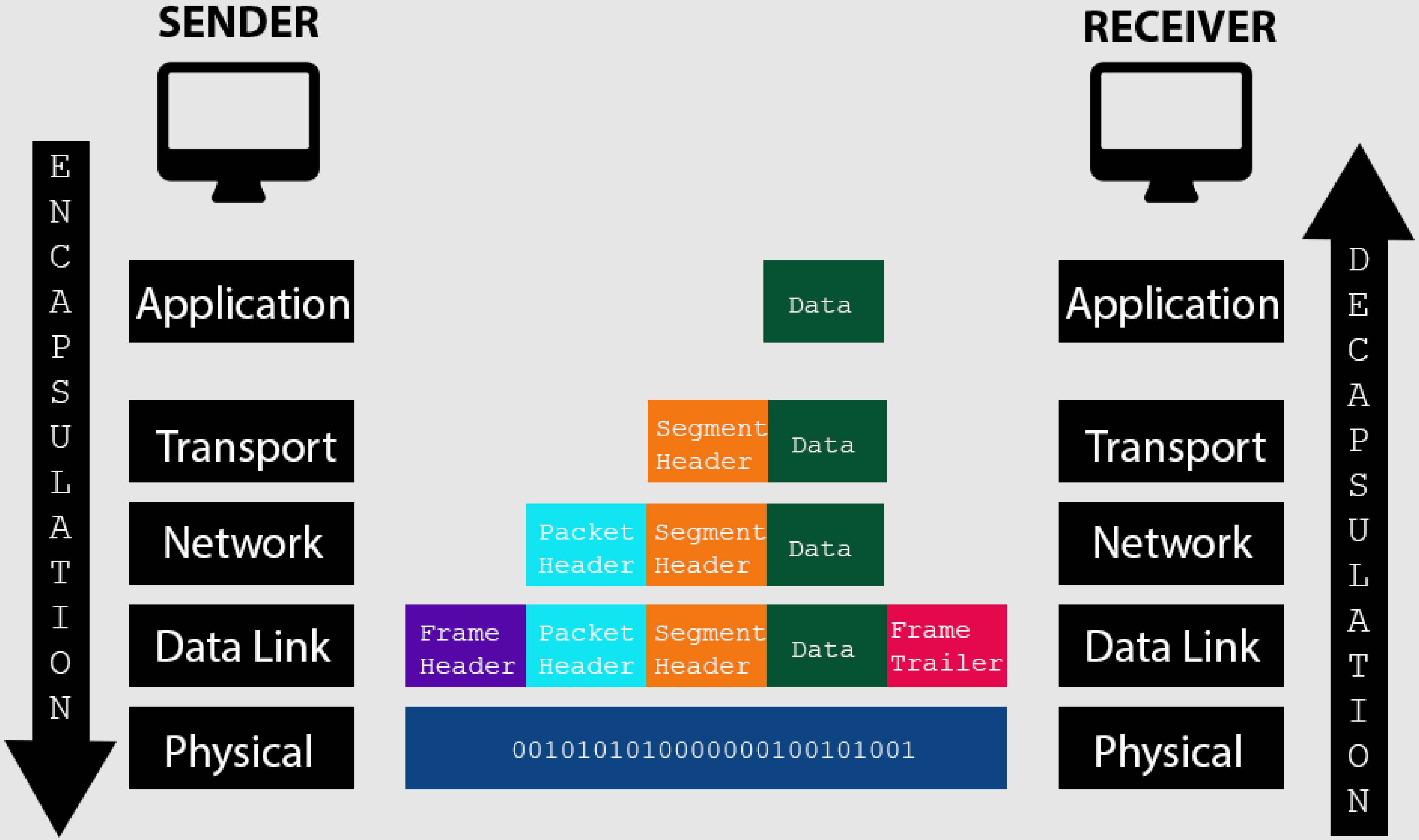
3. IPSec Modes of operation

4. Internet Key Exchange
   - Modes of operation

5. IPSec Scenarios

6. Creating own VPN using OpenVPN

# Open Systems Interconnection model
# (OSI model)



SENDER

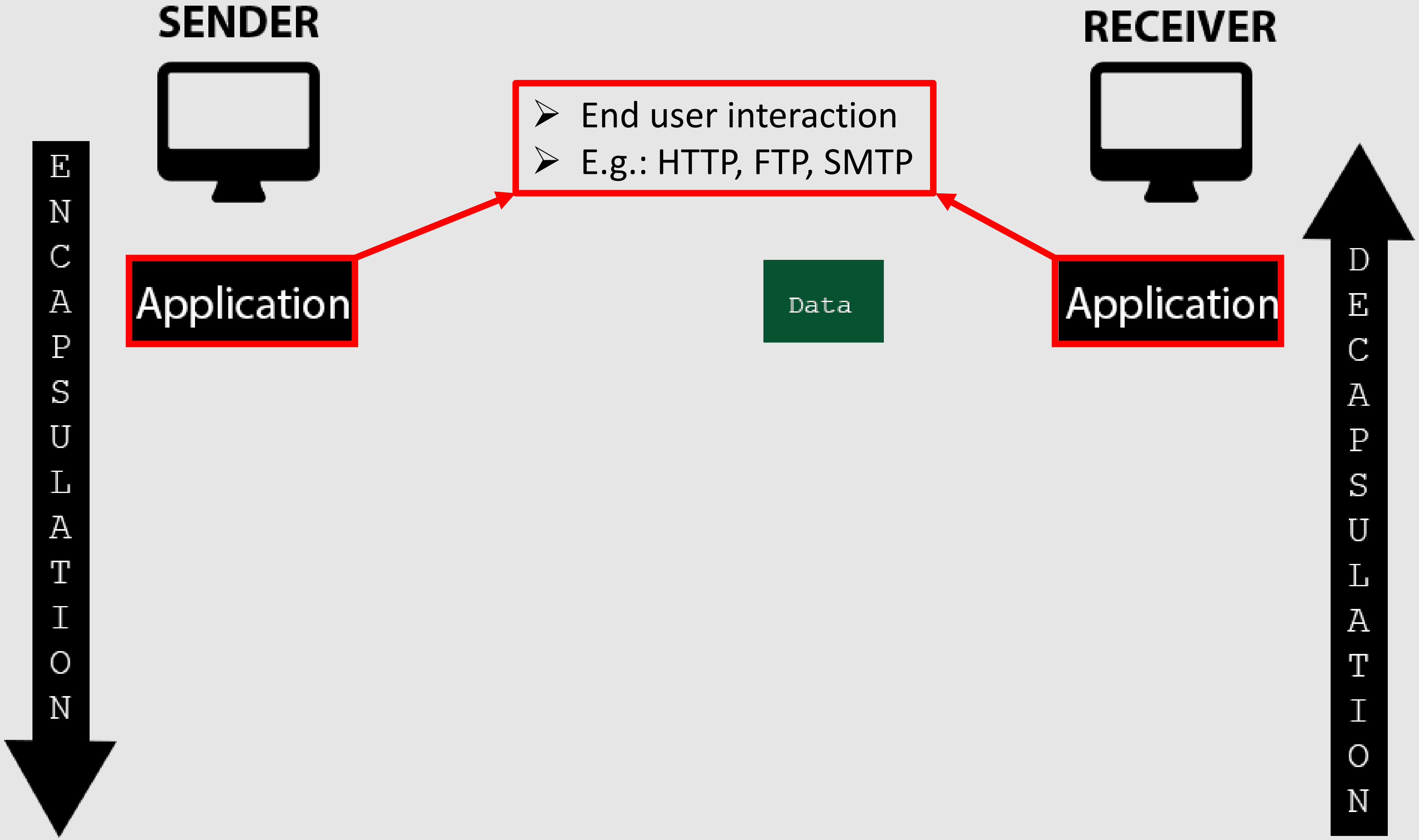
RECEIVER

ENCAPSULATION

DECAPSULATION

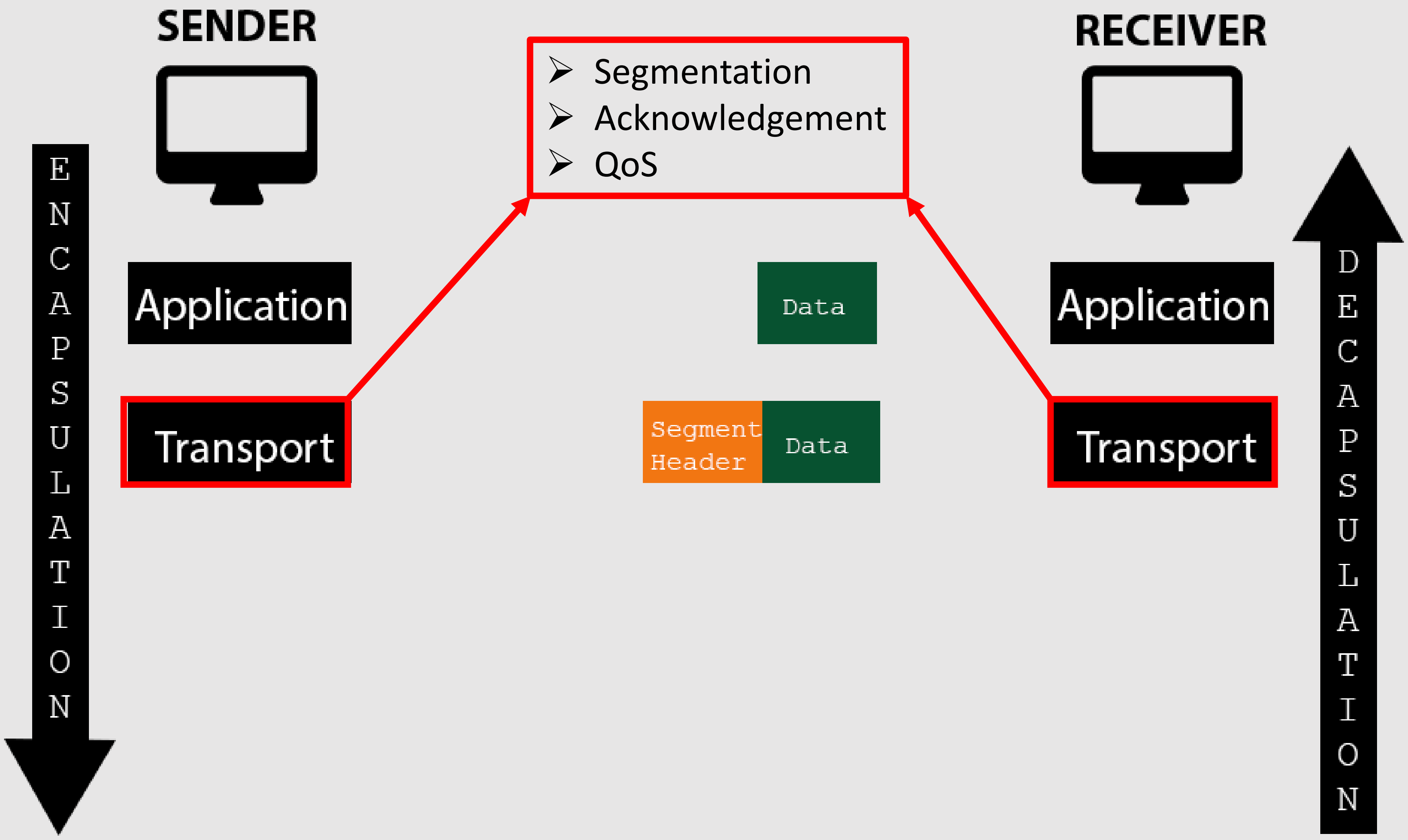
| | | | | | |
|---|---|---|---|---|---|
| Application | | | | Data | |
| Transport | | | Segment Header | Data | |
| Network | | Packet Header | Segment Header | Data | |
| Data Link | Frame Header | Packet Header | Segment Header | Data | Frame Trailer |
| Physical | 0010101010000000100101001 | | | | |

# Open Systems Interconnection model
# (OSI model)



- End user interaction
- E.g.: HTTP, FTP, SMTP

# Open Systems Interconnection model (OSI model)



SENDER

RECEIVER

- Segmentation
- Acknowledgement
- QoS

Application

Transport

Data

Segment Header | Data

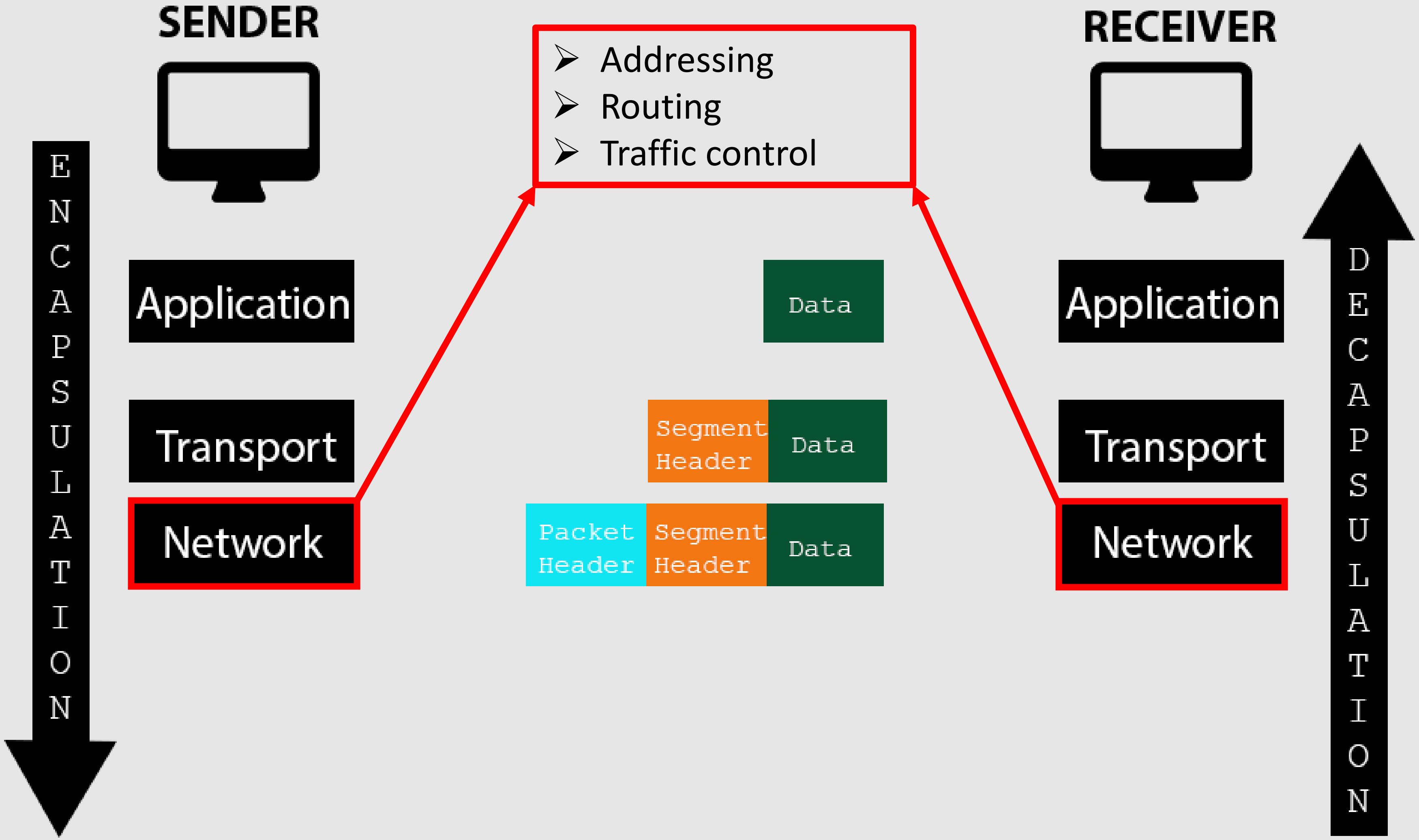
Application

Transport

ENCAPSULATION

DECAPSULATION

# Open Systems Interconnection model (OSI model)



SENDER

ENCAPSULATION

Application

Transport

Network

- Addressing
- Routing
- Traffic control

Data

Segment Header | Data

Packet Header | Segment Header | Data

RECEIVER

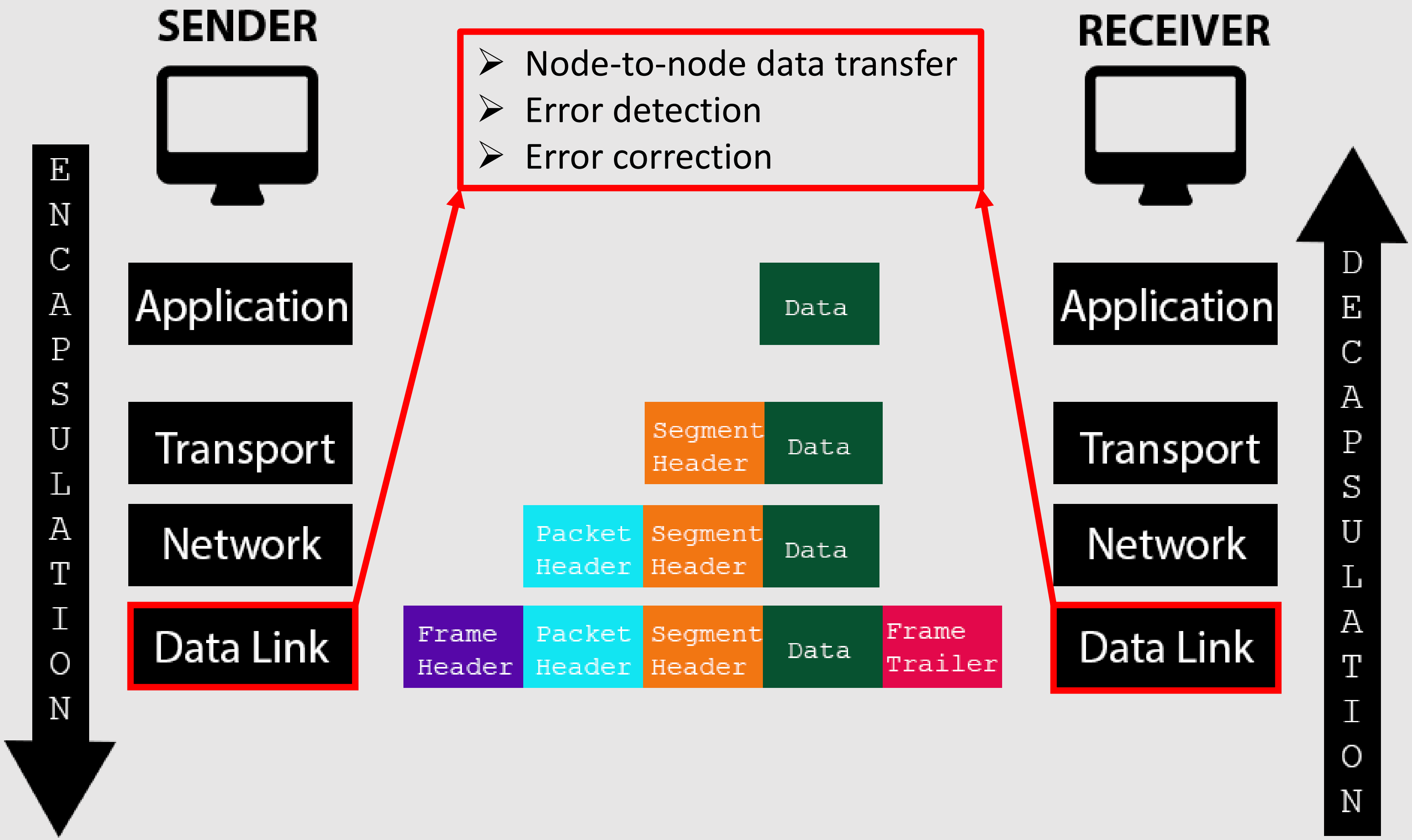
Application

Transport

Network

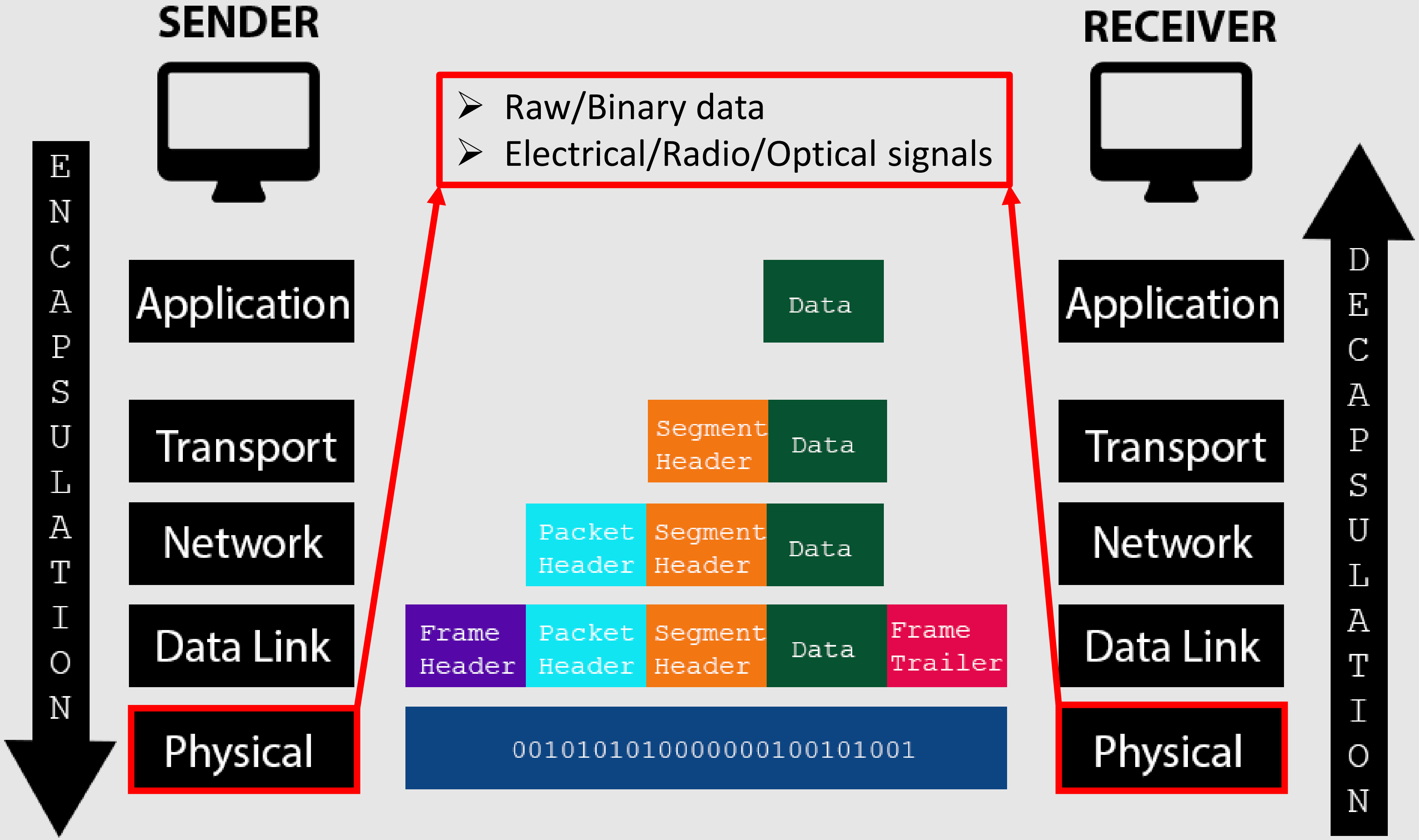
DECAPSULATION

# Open Systems Interconnection model (OSI model)



- Node-to-node data transfer
- Error detection
- Error correction

# Open Systems Interconnection model (OSI model)



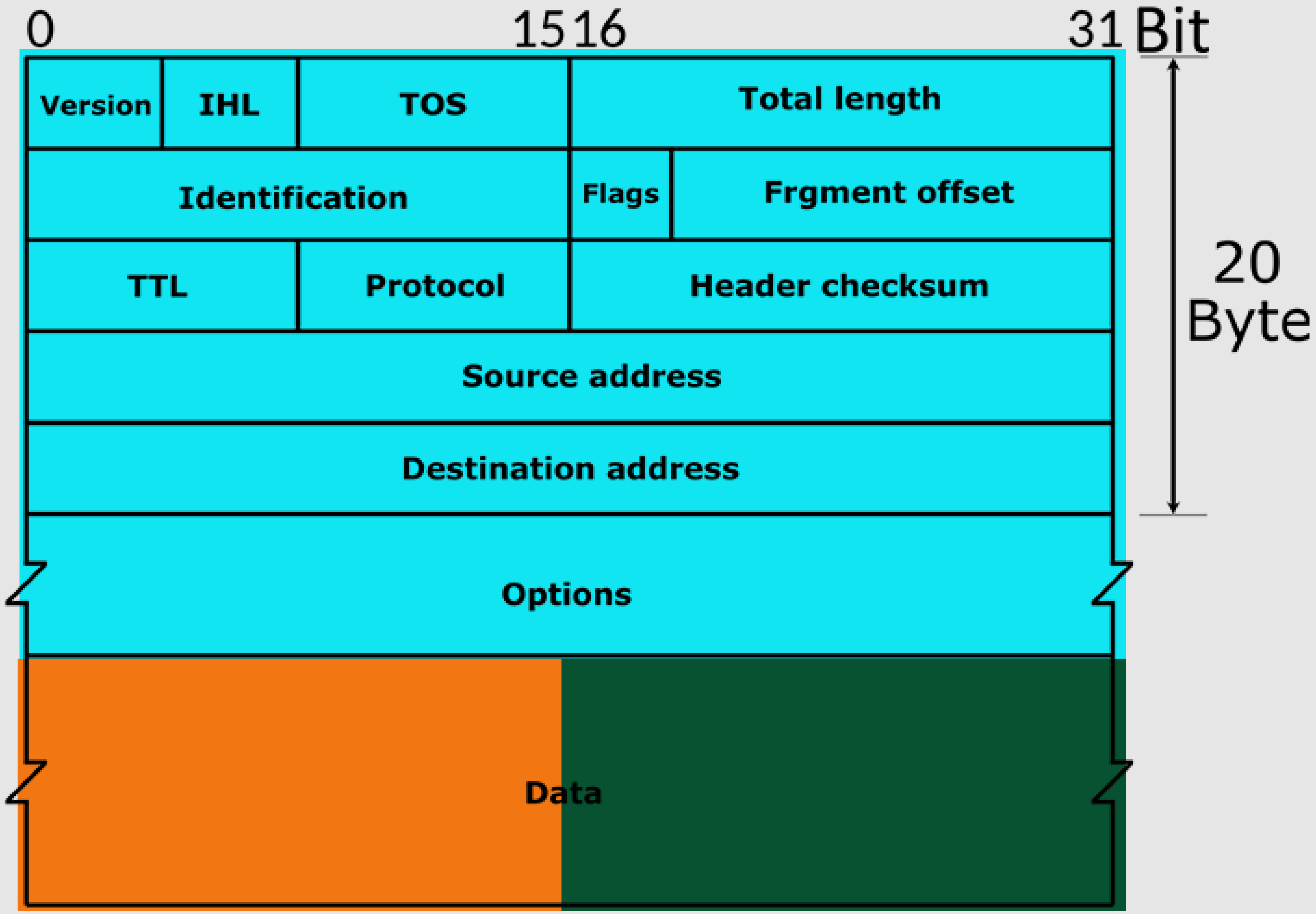
- Raw/Binary data
- Electrical/Radio/Optical signals

# Internet Protocol

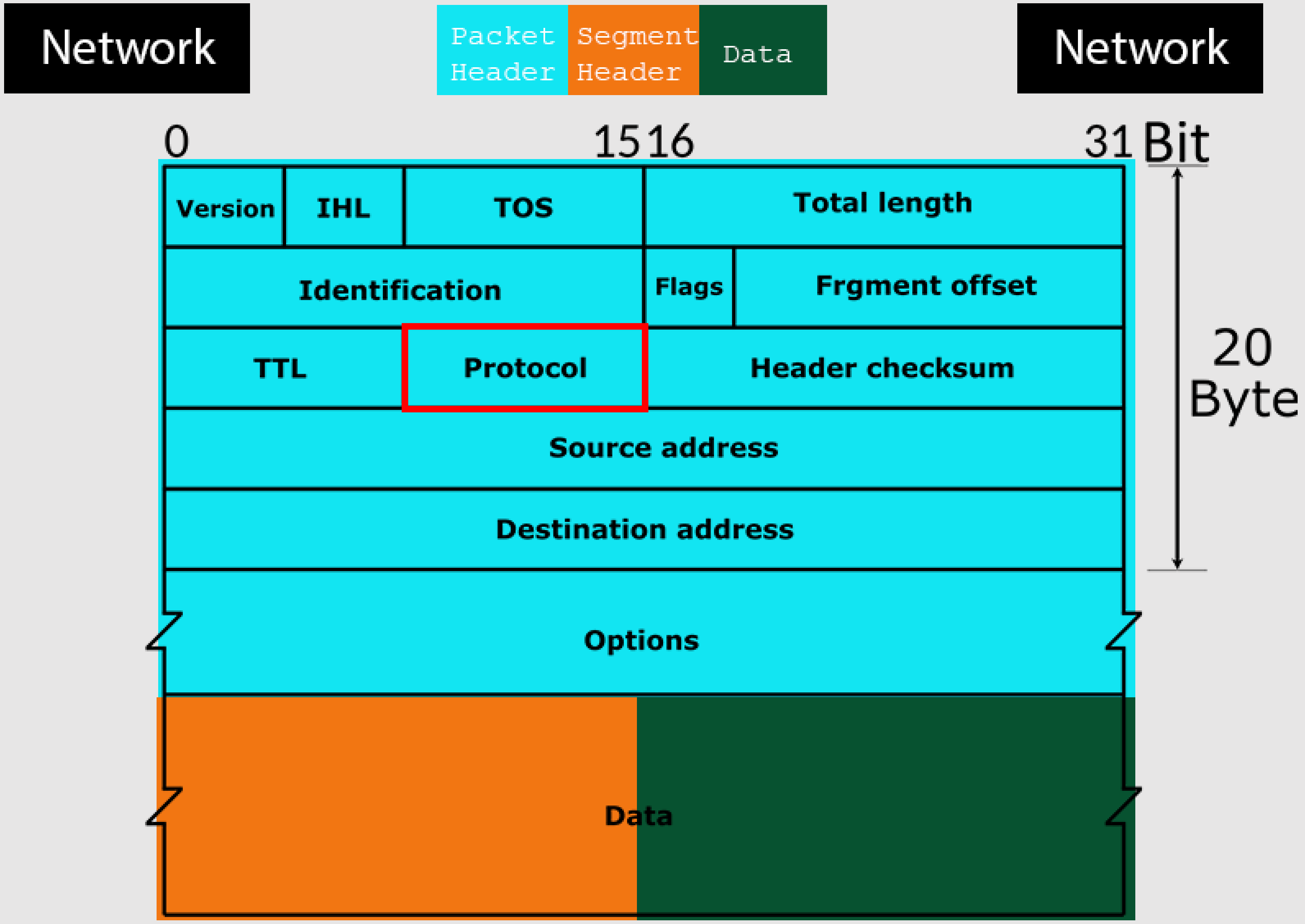
Network

Packet Header | Segment Header | Data

Network



# Internet Protocol



Network

Packet Header

Segment Header

Data

Network

0

15

16

31

Bit

Version

IHL

TOS

Total length

Identification

Flags

Frgment offset

TTL

Protocol

Header checksum

Source address

Destination address

Options

Data

20 Byte

# IP

- no security mechanism
- everyone can read/modify the packets

# IP

- no security mechanism
- everyone can read/modify the packets

## IP

- no security mechanism
- everyone can read/modify the packets

## IPSec

- adds **security** to IP
- **collection of protocols** to secure IP communications:
    - Authentication Header (**AH**);
    - Encapsulating Security Payload (**ESP**);
    - Internet Key Exchange (**IKE**).

- technical specifications:
    - RFC 4301 (Security Architecture for the Internet Protocol);
    - RFC 4302 (IP Authentication Header);
    - RFC 4303 (IP Encapsulating Security Payload);
    - RFC 4308 (Cryptographic Suites for IPSec);
    - RFC 7296 (Internet Key Exchange Protocol Version 2 (IKEv2)).

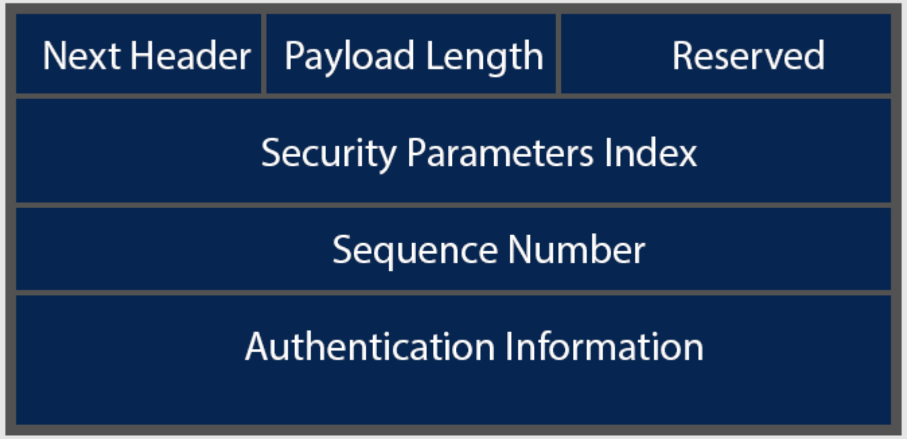
## Authentication Header (AH)

- **authenticates** the origins of packets
- ensures the **integrity** of transmitted datagram
- **anti-replay** mechanism

## Encapsulating Security Payload (ESP)

- ensures data **confidentiality**
- verifies **data integrity**
- **anti-replay** mechanism

# Authentication Header (AH)



Next Header | Payload Length | Reserved

Security Parameters Index

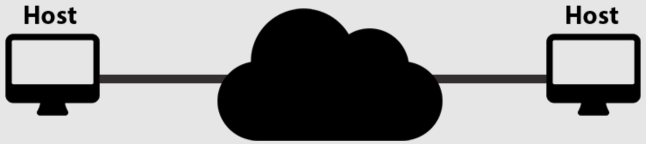
Sequence Number

Authentication Information

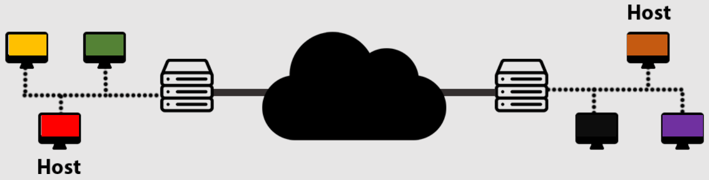
- **Next Header**: indicates the next protocol to which the payload is addressed (IP=4, TCP=6, UDP=17)
- **Payload Length**: specified in 32-bit word multiple
- **Reserved**: future developments
- **Security Parameters Index**: 32-bit number used to identify the connection
- **Sequence Number**: to avoid replay attacks
- **Authentication Information**: MAC value calculated over IP packet (header + payload)

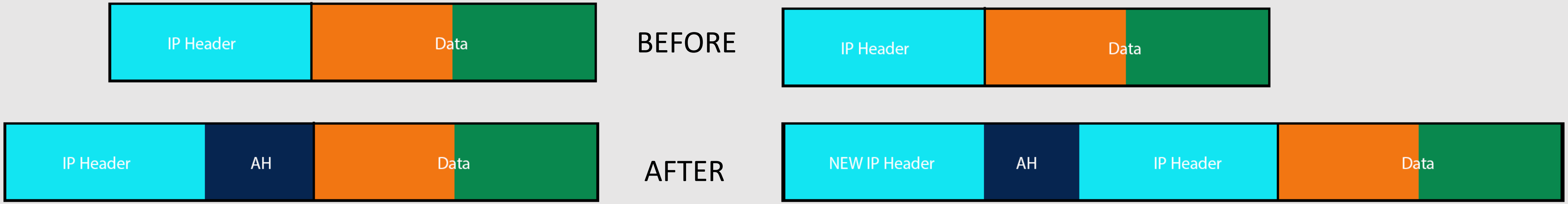
# Modes of operation - AH

## Transport



## Tunnel





Transport, BEFORE: IP Header | Data

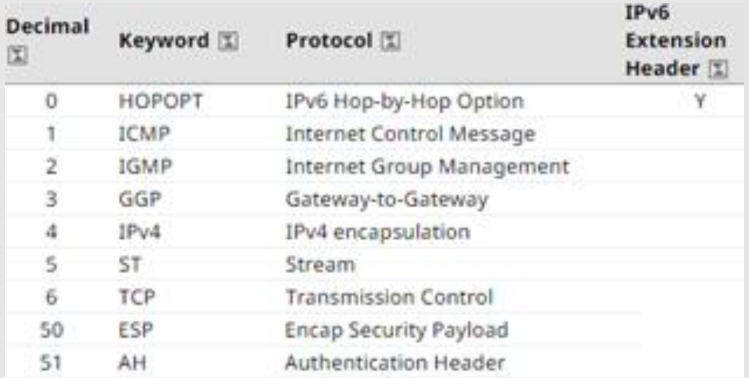
Tunnel, BEFORE: IP Header | Data

Transport, AFTER: IP Header | AH | Data

Tunnel, AFTER: NEW IP Header | AH | IP Header | Data

- AH->NextHeader = IP->Protocol
- IP->Protocol = 51

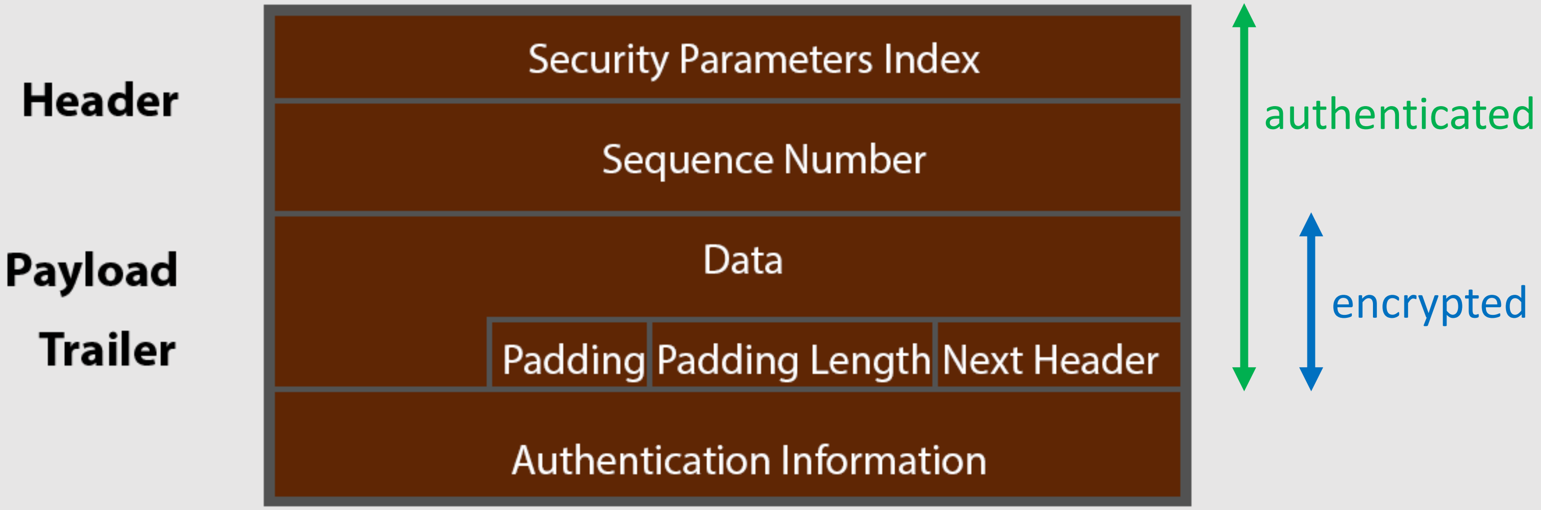
- NewIP->Protocol = 51
- AH->NextHeader = 4



| Decimal | Keyword | Protocol | IPv6 Extension Header |
|---|---|---|---|
| 0 | HOPOPT | IPv6 Hop-by-Hop Option | Y |
| 1 | ICMP | Internet Control Message | |
| 2 | IGMP | Internet Group Management | |
| 3 | GGP | Gateway-to-Gateway | |
| 4 | IPv4 | IPv4 encapsulation | |
| 5 | ST | Stream | |
| 6 | TCP | Transmission Control | |
| 50 | ESP | Encap Security Payload | |
| 51 | AH | Authentication Header | |

https://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml

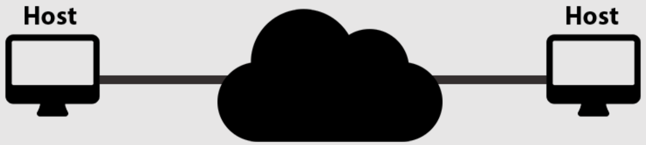
# Encapsulating Security Payload (ESP)

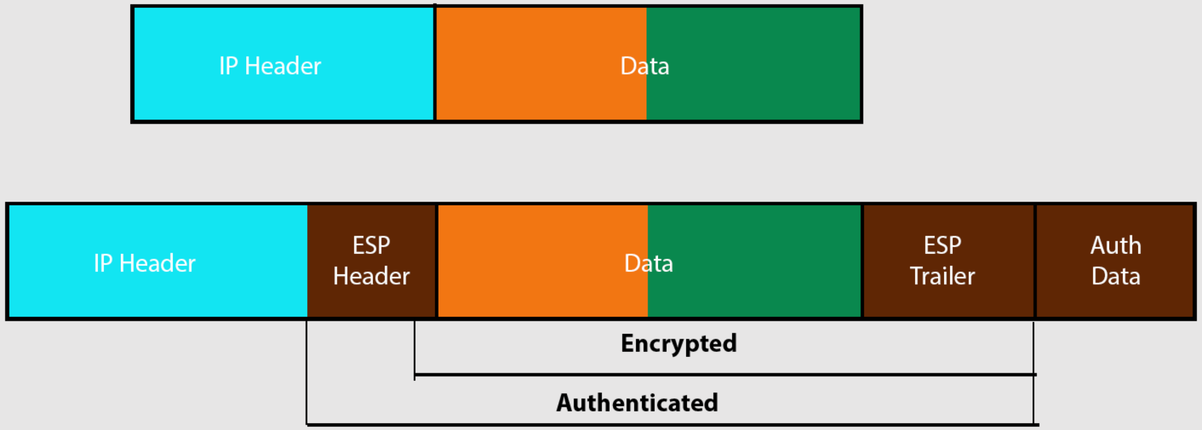


- **Payload**: encrypted IP payload (without IP header)
- **Padding + Padding Length**:
  - extends total length to a multiple of encryption block
  - extends ESP Trailer to a 32bit multiple
- **Authentication Information**: MAC value calculated over IP packet (~~header~~ + payload) => data integrity

# Modes of operation - ESP

## Transport





IP Header | Data

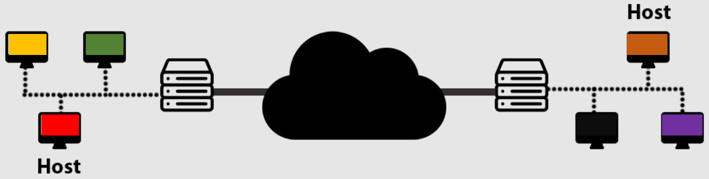
IP Header | ESP Header | Data | ESP Trailer | Auth Data

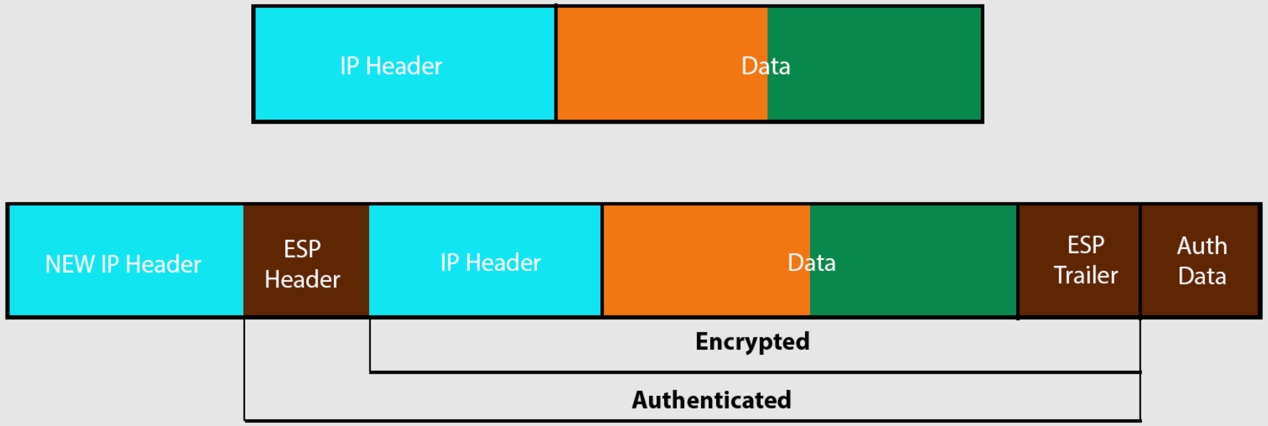
Encrypted

Authenticated

- ESP->NextHeader = IP->Protocol
- IP->Protocol = 50

## Tunnel





IP Header | Data

NEW IP Header | ESP Header | IP Header | Data | ESP Trailer | Auth Data

Encrypted

Authenticated

- NewIP->Protocol = 50
- ESP->NextHeader = 4

# Internet Key Exchange (IKE)

- Negotiation, creation and managing security associations automatically
- Security Association (SA)
  - Set of information which defines IPSec connection properties
  - Keys and algorithms used by both parties

- Two phases:
  1. Setting up **initial SA** (IKE SA)
     - Used to encrypt and authenticate next IKE protocol messages
     - Bidirectional
  2. Setting up a **general SA** (IPSec SA)
     - Used to protect data exchanged between IPSec communicating parties
     - Unidirectional

- X.509 PKI certificates for authentication
- Diffie-Hellman to set up a shared session secret key

# Modes of operation – IKE

**Exchange 1**

- Encryption algorithm (e.g. AES-256)
- MAC algorithm (e.g. SHA-256)
- Authentication method
  (preshared keys / certificates)
- Diffie-Hellman parameters
- Lifetime (e.g. 2 hours)

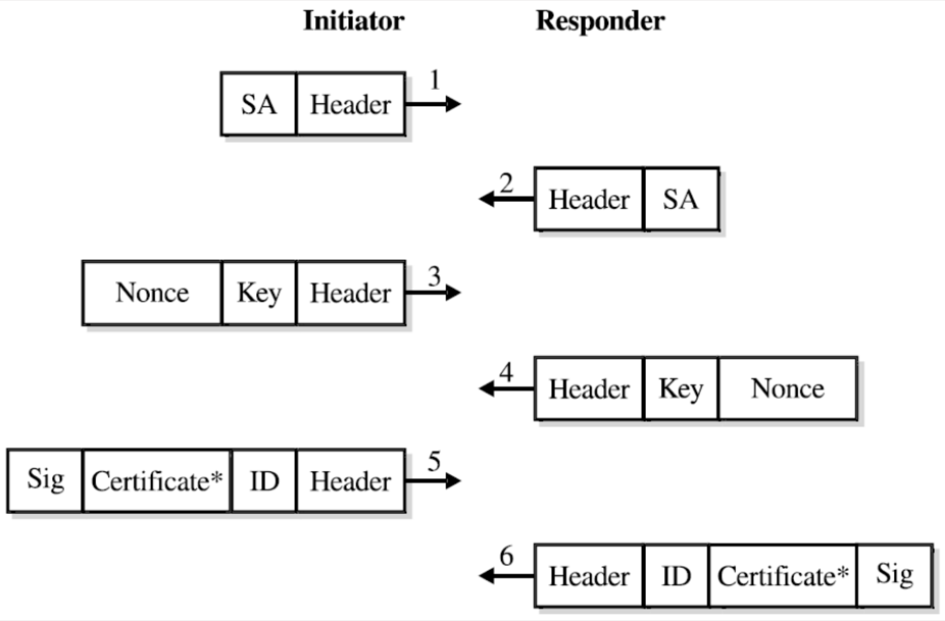
**Exchange 2**

- Establish session key (DH)
- Nonce

**Exchange 3**

- Authentication
- Identity protection (due to encryption)

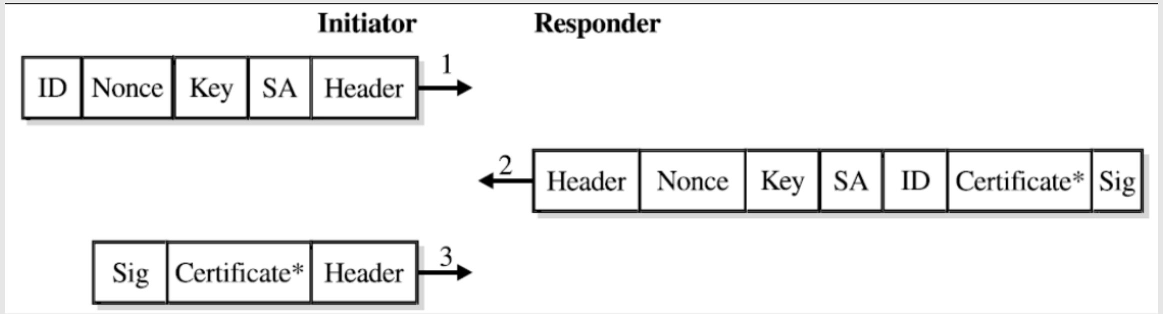
## Main mode



# Modes of operation - IKE

## Aggressive mode

- Faster than main mode (3 messages instead of 6)
- Same as main mode:
  - Establish protection suite
  - Sets up the session key
  - Authenticate entities
- ~~Identity protection~~

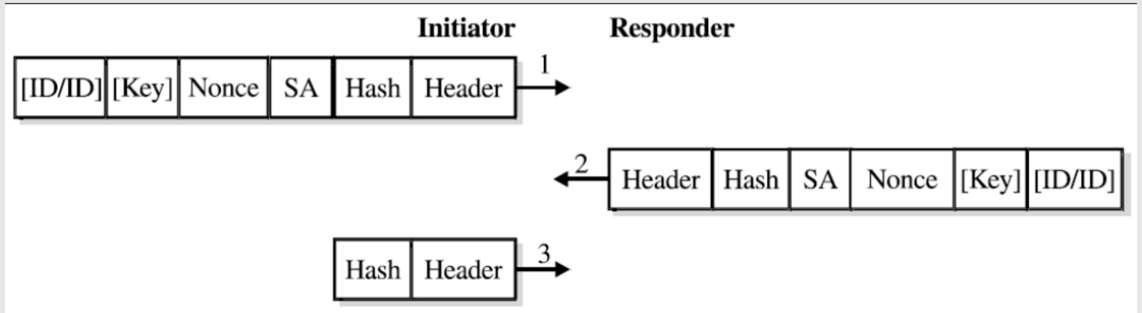


Initiator
Responder
ID | Nonce | Key | SA | Header
1
2
Header | Nonce | Key | SA | ID | Certificate* | Sig
Sig | Certificate* | Header
3

---

## Quick mode

- Used in second phase
- Before SA expires



Initiator
Responder
[ID/ID] | [Key] | Nonce | SA | Hash | Header
1
2
Header | Hash | SA | Nonce | [Key] | [ID/ID]
Hash | Header
3

# Virtual private networks (VPNs)

- Most important application of IPSec

- VPN = virtual network which allow computers communicate securely across a public network

# Virtual private networks (VPNs)

- Most important application of IPSec

- VPN = virtual network which allow computers communicate securely across a public network
  - **Secure transfer of data**

# Virtual private networks (VPNs)

- Most important application of IPSec

- VPN = virtual network which allow computers communicate securely across a public network
  - Secure transfer of data
  - **Less expensive than dedicated network lines**

# Virtual private networks (VPNs)

- Most important application of IPSec

- VPN = virtual network which allow computers communicate securely across a public network
  - Secure transfer of data
  - Less expensive than dedicated network lines
  - **Remote access**

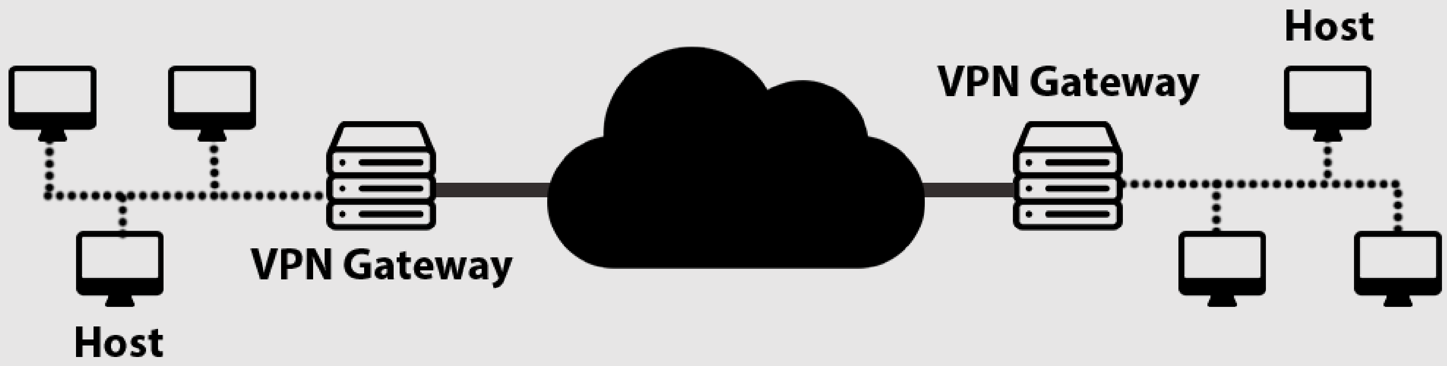
# Virtual private networks (VPNs)

- Most important application of IPSec

- VPN = virtual network which allow computers communicate securely across a public network
    - Secure transfer of data
    - Less expensive than dedicated network lines
    - Remote access

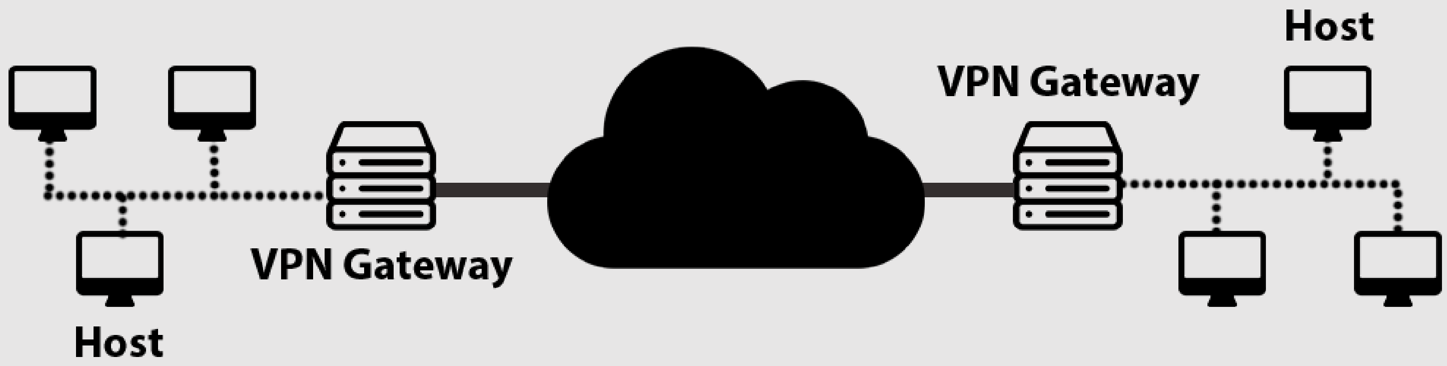
- Arhitecture
    - Gateway-to-Gateway
    - Host-to-Gateway
    - Host-to-Host

# Gateway-to-Gateway VPN



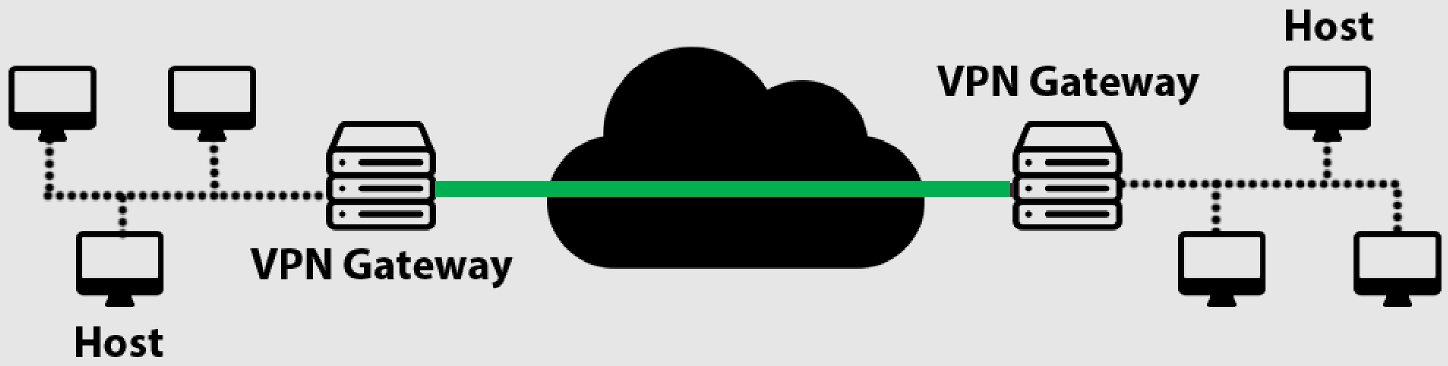
- Usually, for securing communication between two networks

# Gateway-to-Gateway VPN



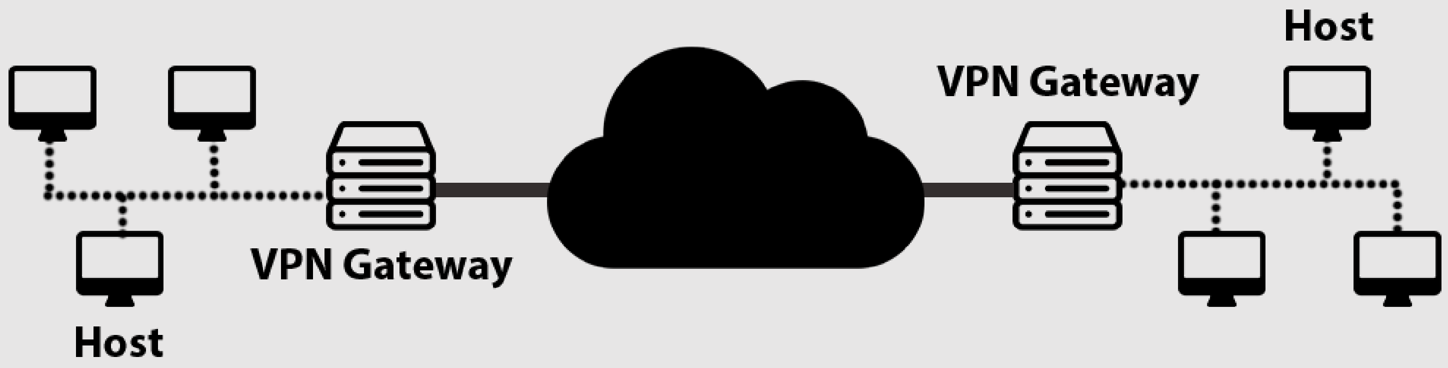
- Usually, for securing communication between two networks
- **Establish an IPSec tunnel between gateways**

# Gateway-to-Gateway VPN



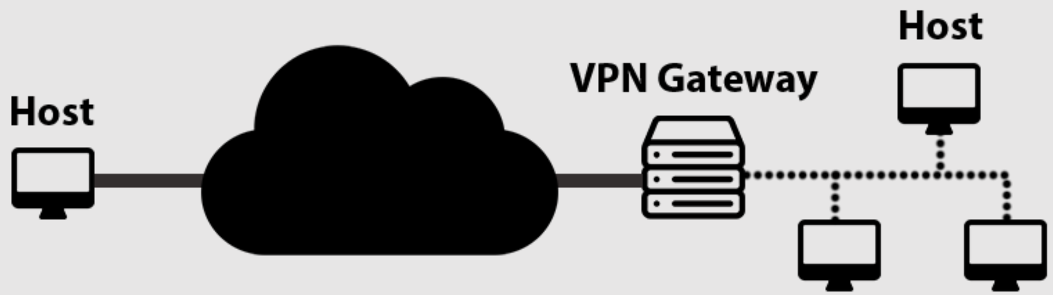
- Usually, for securing communication between two networks
- Establish an IPSec tunnel between gateways
- **Protects data only between gateways**

# Gateway-to-Gateway VPN



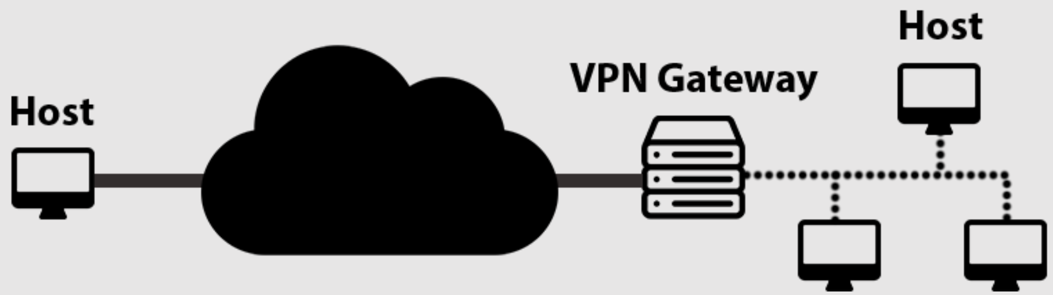
- Usually, for securing communication between two networks
- Establish an IPSec tunnel between gateways
- Protects data only between gateways
- **A.k.a. Site-to-Site VPN**
- **Easy to implement, easy to manage**
- **Transparent**

# Host-to-Gateway VPN



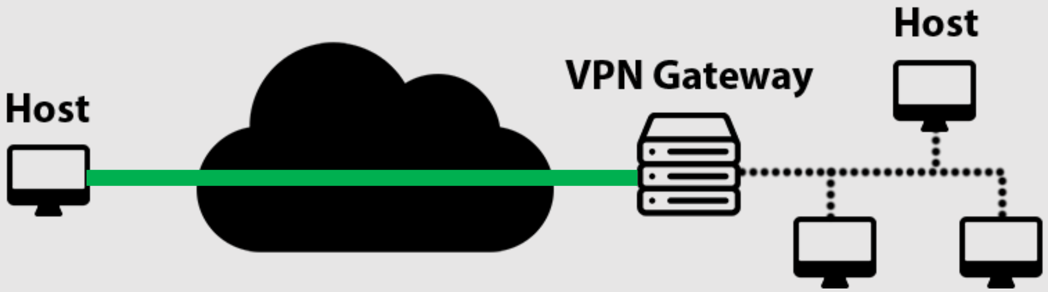
- Recently, most used (work-from-home)

# Host-to-Gateway VPN



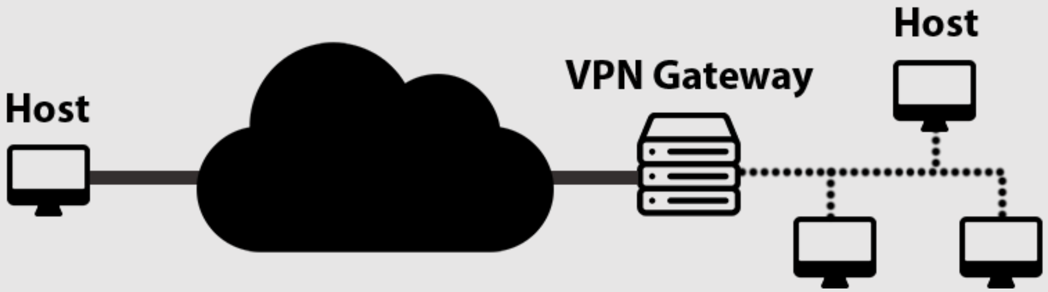
- Recently, most used (work-from-home)
- **Establish an IPSec tunnel between host and VPN gateway**

# Host-to-Gateway VPN



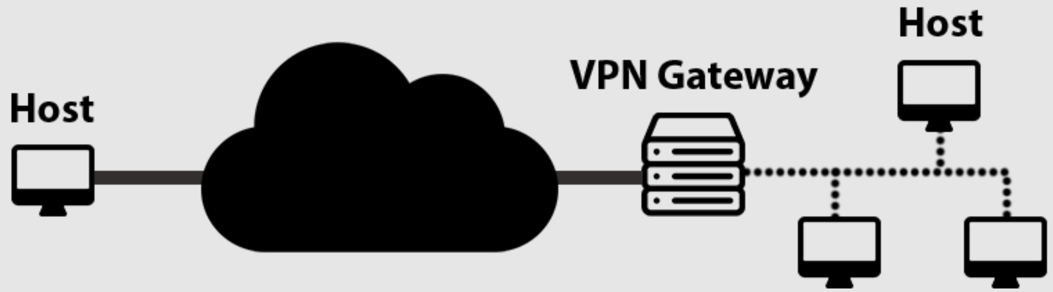
- Recently, most used (work-from-home)
- Establish an IPSec tunnel between host and VPN gateway
- **Data is protected only between host and gateway**

# Host-to-Gateway VPN



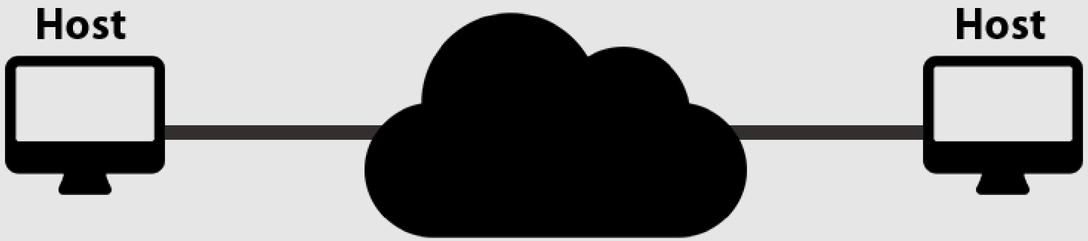
- Recently, most used (work-from-home)
- Establish an IPSec tunnel between host and VPN gateway
- Data is protected only between host and gateway
- **A.k.a. Remote Access VPN**

# Host-to-Gateway VPN



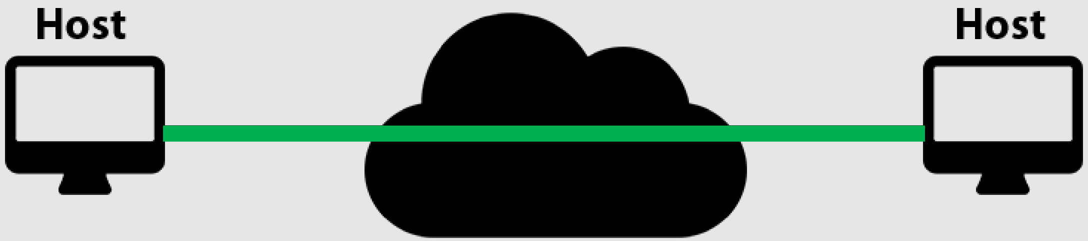
- Recently, most used (work-from-home)
- Establish an IPSec tunnel between host and VPN gateway
- Data is protected only between host and gateway
- A.k.a. Remote Access VPN
- **Client VPN software required**

# Host-to-Host VPN



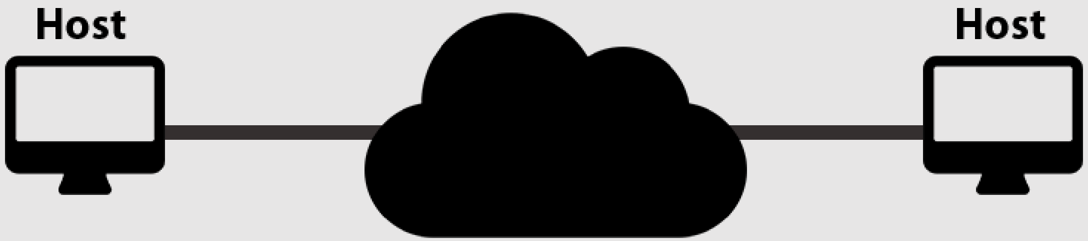
- Use-case: system administrator which wants to configure a server remotely

# Host-to-Host VPN



- Use-case: system administrator which wants to configure a server remotely
- **Data is end-to-end protected**

# Host-to-Host VPN



- Use-case: system administrator which wants to configure a server remotely
- Data is end-to-end protected
- **Firewall/IDS can’t inspect content**

# IPSec scenarios with Strong Swan

- StrongSwan = Strong Secure WAN

**Case 1**

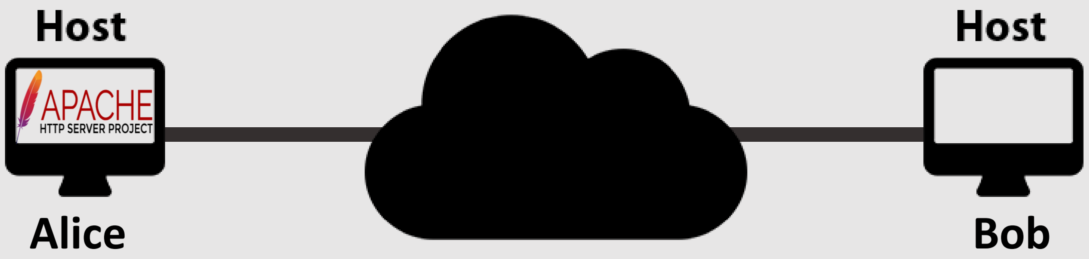
a) StrongSwan AH
b) Ping Host1 -> Host2
c) Wireshark

**Case 2**

a) StrongSwan ESP
b) Ping Host1 -> Host2
c) Wireshark

**Case 3**

a) Securing HTTP communication
b) Apache Web Server
c) IPSec
d) WireShark



```
# Start the infrastructure
docker-compose up -d
```

```
# Display the containers
docker ps
```

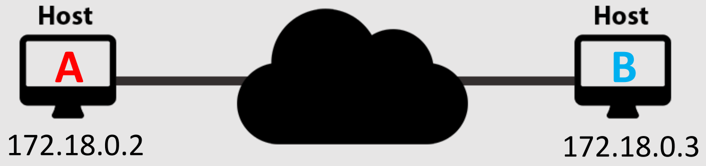
```
# Retrieve IPs
docker inspect -f '{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' alice
docker inspect -f '{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}’ bob
```

```
# Access containers
docker exec –it alice /bin/bash
docker exec –it bob /bin/bash
```

```
# Display network interfaces
ip a
```

```
# Listen traffic between Alice and BOB
sudo tcpdump -i <INTERFACE>host <IP_ALICE> and host <IP_BOB> -s0 -vv -X -c 1000
```

# IPSec scenarios with Strong Swan



**Case 1**

- StrongSwan **AH**
- Ping Host1 -> Host2
- Tcpdump + Wireshark

a) Check strongswan service status
   - › **ipsec status**

b) Edit config files: **/etc/ipsec.conf** and **/etc/ipsec.secrets**

c) (Re)start strongswan service
   - › **ipsec restart**

d) Capture network traffic between Alice and Bob

### ALICE
### /etc/ipsec.conf

```
conn alice-to-bob
        keyexchange=ikev1
        authby=secret
        left=172.18.0.2
        right=172.18.0.3
        auto=start
        type=transport
        ah=sha256-sha512-modp2048!
```

### ALICE & BOB
### /etc/ipsec.secrets

```
: PSK "This is a strong password"
```

### BOB
### /etc/ipsec.conf

```
conn alice-to-bob
        keyexchange=ikev1
        authby=secret
        left=172.18.0.3
        right=172.18.0.2
        auto=start
        type=transport
        ah=sha256-sha512-modp2048!
```

# IPSec scenarios with Strong Swan

**Case 1**

- StrongSwan **AH**
- Ping Host1 -> Host2
- Tcpdump + Wireshark

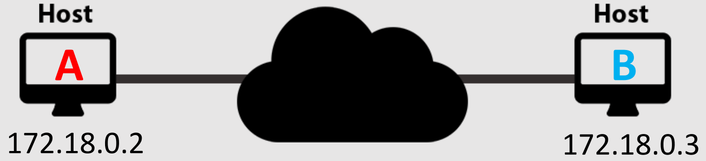
a) Check strongswan service status
   - › **ipsec status**

b) Edit config files: **/etc/ipsec.conf** and **/etc/ipsec.secrets**

c) (Re)start strongswan service
   - › **ipsec restart**

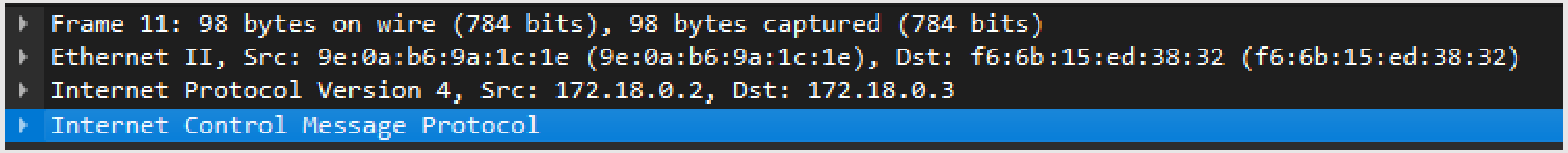
d) Capture network traffic between Alice and Bob

**sudo tcpdump -i <INTERFACE> -vv -s0 -X -c 1000 -w ping_capture.pcap**

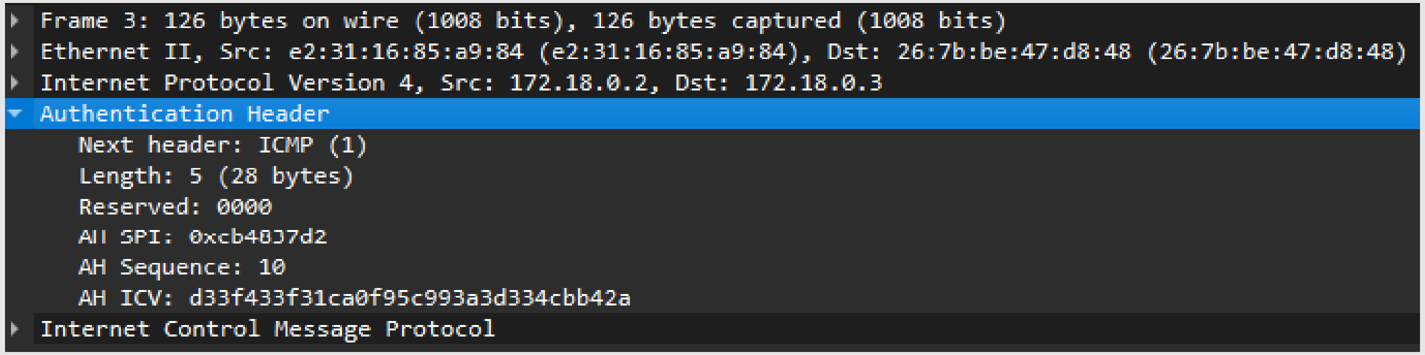


**# Copy file configs to containers**

```
docker cp alice/ipsec_ah.conf alice:/etc/ipsec.conf
docker cp bob/ipsec_ah.conf bob:/etc/ipsec.conf
docker cp ipsec.secrets alice:/etc/ipsec.secrets
docker cp ipsec.secrets bob:/etc/ipsec.secrets
```



```
Frame 11: 98 bytes on wire (784 bits), 98 bytes captured (784 bits)
Ethernet II, Src: 9e:0a:b6:9a:1c:1e (9e:0a:b6:9a:1c:1e), Dst: f6:6b:15:ed:38:32 (f6:6b:15:ed:38:32)
Internet Protocol Version 4, Src: 172.18.0.2, Dst: 172.18.0.3
Internet Control Message Protocol
```



```
Frame 3: 126 bytes on wire (1008 bits), 126 bytes captured (1008 bits)
Ethernet II, Src: e2:31:16:85:a9:84 (e2:31:16:85:a9:84), Dst: 26:7b:be:47:d8:48 (26:7b:be:47:d8:48)
Internet Protocol Version 4, Src: 172.18.0.2, Dst: 172.18.0.3
Authentication Header
    Next header: ICMP (1)
    Length: 5 (28 bytes)
    Reserved: 0000
    AH SPI: 0xcb4837d2
    AH Sequence: 10
    AH ICV: d33f433f31ca0f95c993a3d334cbb42a
Internet Control Message Protocol
```

# IPSec scenarios with Strong Swan

**Case 2**

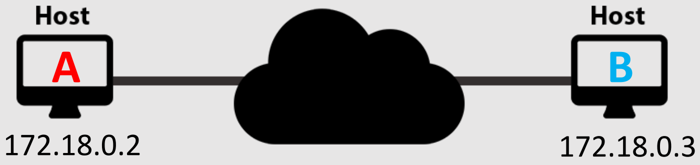
- StrongSwan **ESP**
- Ping Host1 -> Host2
- Tcpdump + Wireshark

a) Check strongswan service status
  - › **ipsec status**

b) Edit config files: **/etc/ipsec.conf** and **/etc/ipsec.secrets**

c) (Re)start strongswan service
  - › **ipsec restart**

d) Capture network traffic between Alice and Bob



**ALICE**

**/etc/ipsec.conf**

```
conn alice-to-bob
        keyexchange=ikev1
        authby=secret
        left=172.18.0.2
        right=172.18.0.3
        auto=start
        type=transport
        esp=aes128-sha256
```

**ALICE & BOB**

**/etc/ipsec.secrets**

```
: PSK "This is a strong password"
```

**BOB**

**/etc/ipsec.conf**

```
conn alice-to-bob
        keyexchange=ikev1
        authby=secret
        left=172.18.0.3
        right=172.18.0.2
        auto=start
        type=transport
        esp=aes128-sha256
```

# IPSec scenarios with Strong Swan

**Case 2**

- StrongSwan **ESP**
- Ping Host1 -> Host2
- Tcpdump + Wireshark

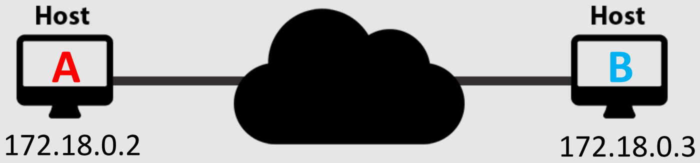
a) Check strongswan service status
   › **ipsec status**

b) Edit config files: **/etc/ipsec.conf** and **/etc/ipsec.secrets**

c) (Re)start strongswan service
   › **ipsec restart**

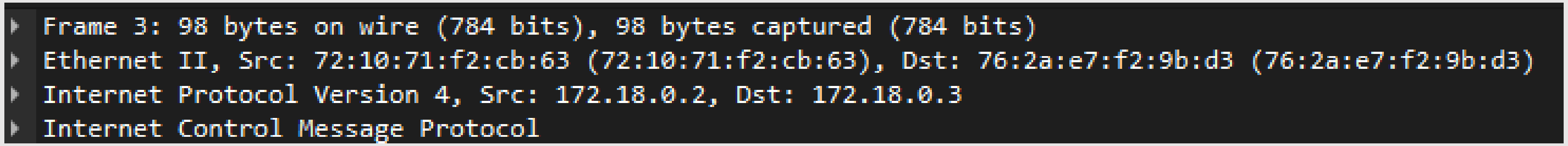
d) Capture network traffic between Alice and Bob

```
sudo tcpdump -i <INTERFACE> -vv -s0 -X -c 1000 -w ping_capture.pcap
```

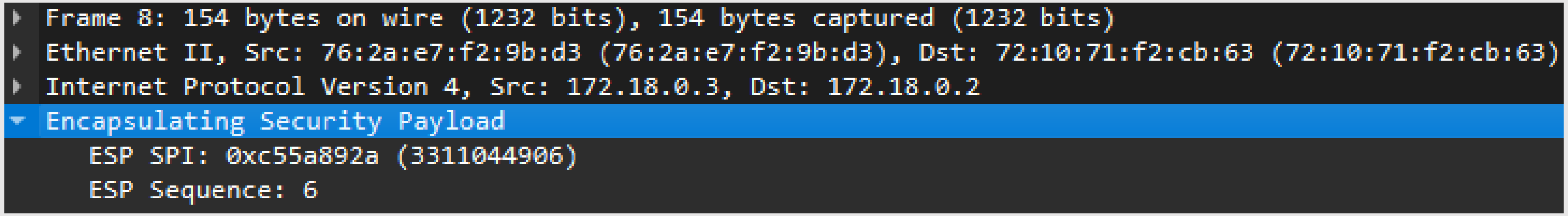


**# Copy file configs to containers**

```
docker cp alice/ipsec_esp.conf alice:/etc/ipsec.conf
docker cp bob/ipsec_esp.conf bob:/etc/ipsec.conf
docker cp ipsec.secrets alice:/etc/ipsec.secrets
docker cp ipsec.secrets bob:/etc/ipsec.secrets
```



```
Frame 3: 98 bytes on wire (784 bits), 98 bytes captured (784 bits)
Ethernet II, Src: 72:10:71:f2:cb:63 (72:10:71:f2:cb:63), Dst: 76:2a:e7:f2:9b:d3 (76:2a:e7:f2:9b:d3)
Internet Protocol Version 4, Src: 172.18.0.2, Dst: 172.18.0.3
Internet Control Message Protocol
```

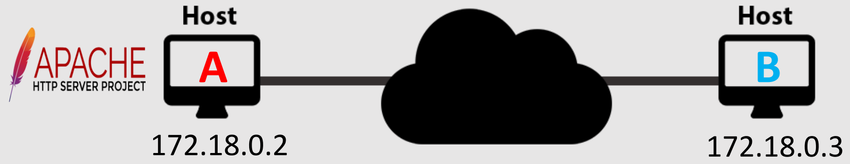


```
Frame 8: 154 bytes on wire (1232 bits), 154 bytes captured (1232 bits)
Ethernet II, Src: 76:2a:e7:f2:9b:d3 (76:2a:e7:f2:9b:d3), Dst: 72:10:71:f2:cb:63 (72:10:71:f2:cb:63)
Internet Protocol Version 4, Src: 172.18.0.3, Dst: 172.18.0.2
Encapsulating Security Payload
    ESP SPI: 0xc55a892a (3311044906)
    ESP Sequence: 6
```

# IPSec scenarios with Strong Swan

**Case 3**

- Securing HTTP communication
- Apache Web Server
- Tcpdump + WireShark



a) Check apache2 service status
   › **service apache2 status**

b) Start apache2 service
   › **service start apache2 start**

c) Capture network traffic between Alice and Bob while accessing **http://ALICE_IP** from Bob

```
curl -X POST -d'{"username": "bob", "password": "secret" }' -H "Content-Type: application/json" http://ALICE_IP
```

a) Check strongswan service status
   › **ipsec status**

b) Edit config files: **/etc/ipsec.conf** and **/etc/ipsec.secrets**

c) (Re)start strongswan service
   › **ipsec restart**

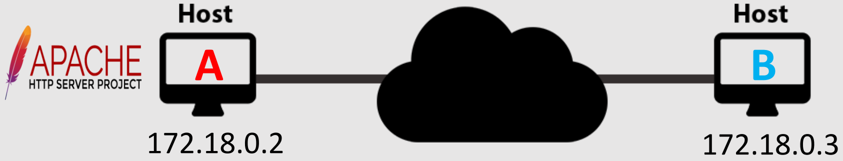
d) Capture network traffic between Alice and Bob while accessing **http://ALICE_IP** from Bob

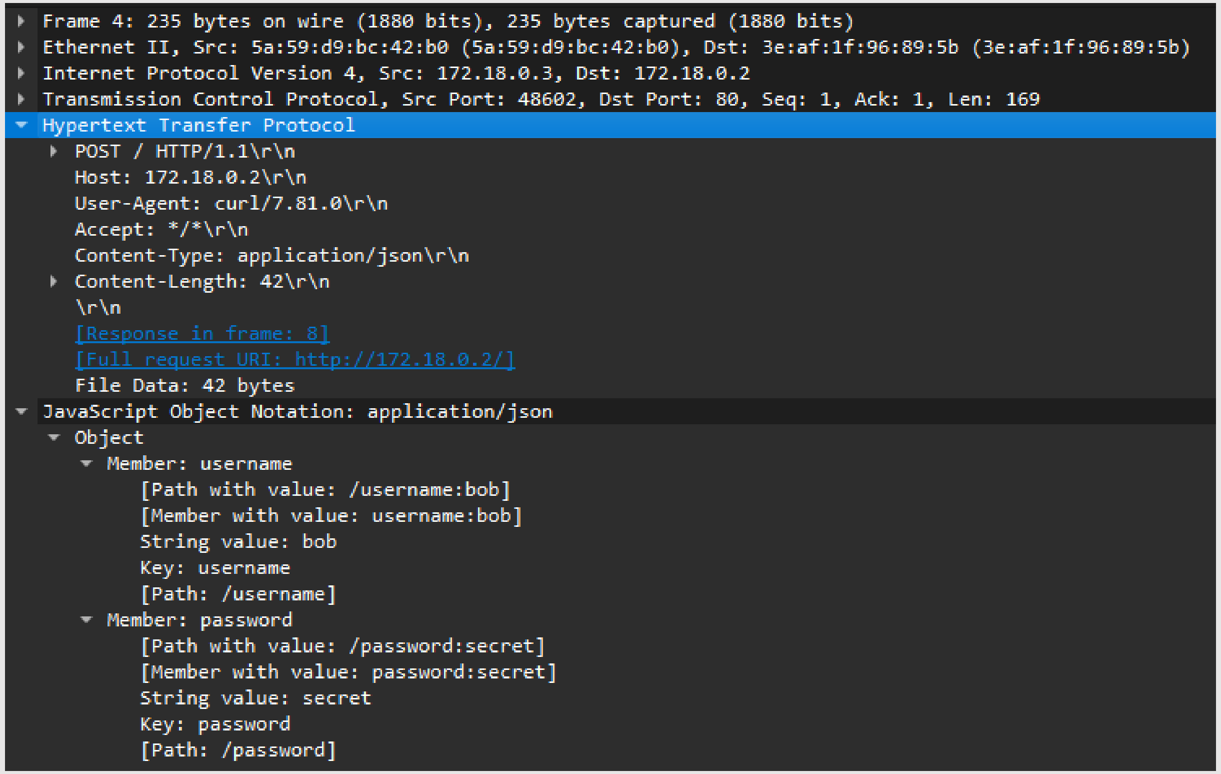
# IPSec scenarios with Strong Swan

**Case 3**

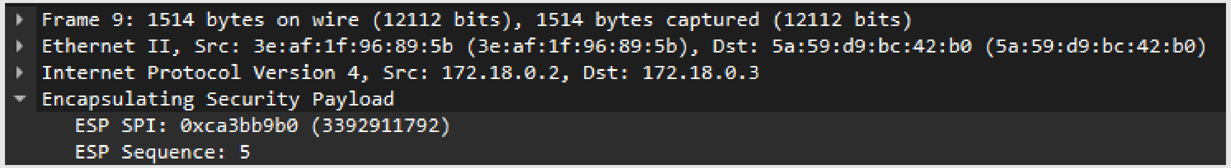
- Securing HTTP communication
- Apache Web Server
- Tcpdump + WireShark

**curl -X POST -d'{"username": "bob", "password": "secret" }' -H "Content-Type: application/json" http://172.18.0.2**





```
▸ Frame 4: 235 bytes on wire (1880 bits), 235 bytes captured (1880 bits)
▸ Ethernet II, Src: 5a:59:d9:bc:42:b0 (5a:59:d9:bc:42:b0), Dst: 3e:af:1f:96:89:5b (3e:af:1f:96:89:5b)
▸ Internet Protocol Version 4, Src: 172.18.0.3, Dst: 172.18.0.2
▸ Transmission Control Protocol, Src Port: 48602, Dst Port: 80, Seq: 1, Ack: 1, Len: 169
▾ Hypertext Transfer Protocol
  ▸ POST / HTTP/1.1\r\n
    Host: 172.18.0.2\r\n
    User-Agent: curl/7.81.0\r\n
    Accept: */*\r\n
    Content-Type: application/json\r\n
  ▸ Content-Length: 42\r\n
    \r\n
    [Response in frame: 8]
    [Full request URI: http://172.18.0.2/]
    File Data: 42 bytes
▾ JavaScript Object Notation: application/json
  ▾ Object
    ▾ Member: username
        [Path with value: /username:bob]
        [Member with value: username:bob]
        String value: bob
        Key: username
        [Path: /username]
    ▾ Member: password
        [Path with value: /password:secret]
        [Member with value: password:secret]
        String value: secret
        Key: password
        [Path: /password]
```



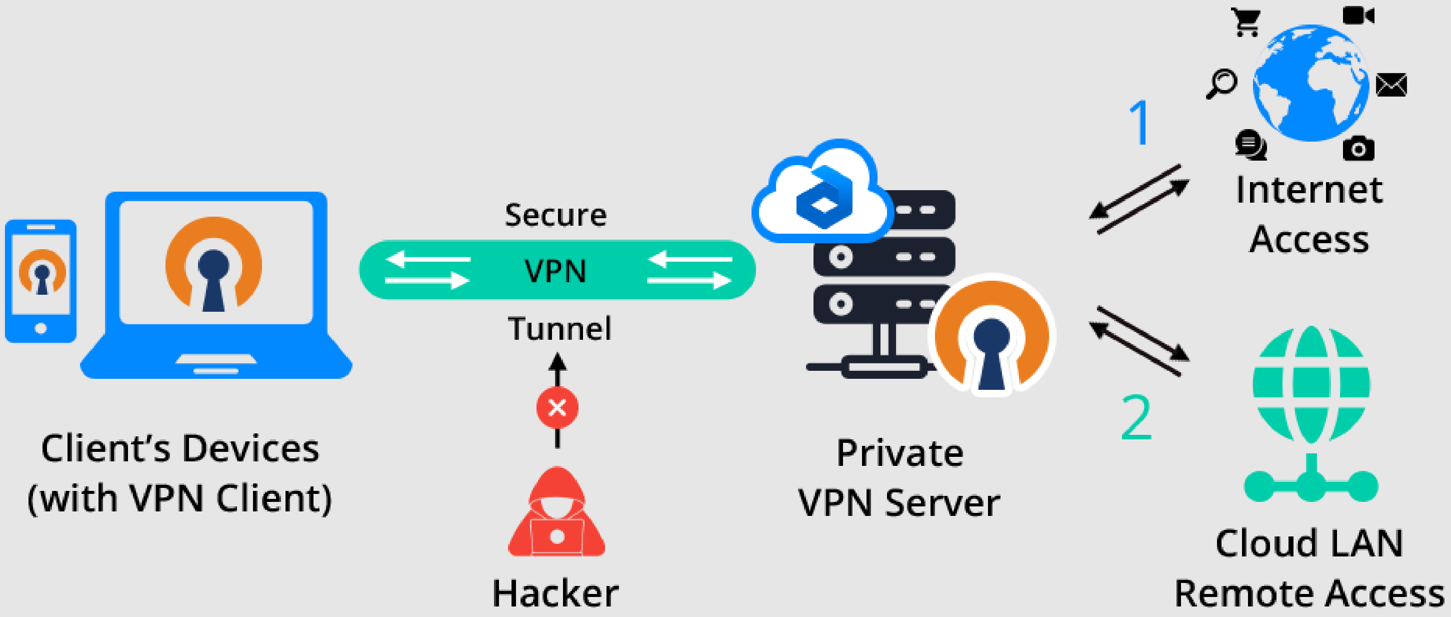
```
▸ Frame 9: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits)
▸ Ethernet II, Src: 3e:af:1f:96:89:5b (3e:af:1f:96:89:5b), Dst: 5a:59:d9:bc:42:b0 (5a:59:d9:bc:42:b0)
▸ Internet Protocol Version 4, Src: 172.18.0.2, Dst: 172.18.0.3
▾ Encapsulating Security Payload
    ESP SPI: 0xca3bb9b0 (3392911792)
    ESP Sequence: 5
```

# OpenVPN

- Virtual private network system for creating point-to-point / site-to-site connections;
- Its own security protocol with the same name;
- Uses OpenSSL library;
- TCP / UDP (recommended);

**Authentication methods**

a) Pre-shared keys;
b) Certificates;
c) Username & password.



Ref: https://www.virtuozzo.com/company/blog/private-openvpn-access-server/

# OpenVPN

**Steps**

a) Creating a Virtual Private Server (VPS)

- A brief list of VPS providers: https://www.g2.com/categories/virtual-private-servers-vps;
- We'll go with Azure: https://azure.microsoft.com/en-us/free/;
- Create an Ubuntu Server with one of the latest versions, after 18.04, preferably;
- Save its public IP address.

b) Generating SSH keys for authentication

```
ssh-keygen –t rsa –b 4096 –f PRIVATE_KEY_FILENAME
Ssh-keygen –f PRIVATE_KEY_FILENAME -y PUBLIC_KEY_FILENAME
```

c) Log in to the server

```
ssh USERNAME@VPS_PUBLIC_IP
```

d) Update the operating system

```
sudo apt-get update && apt-get upgrade
```

e) Create a non-root user and set a password

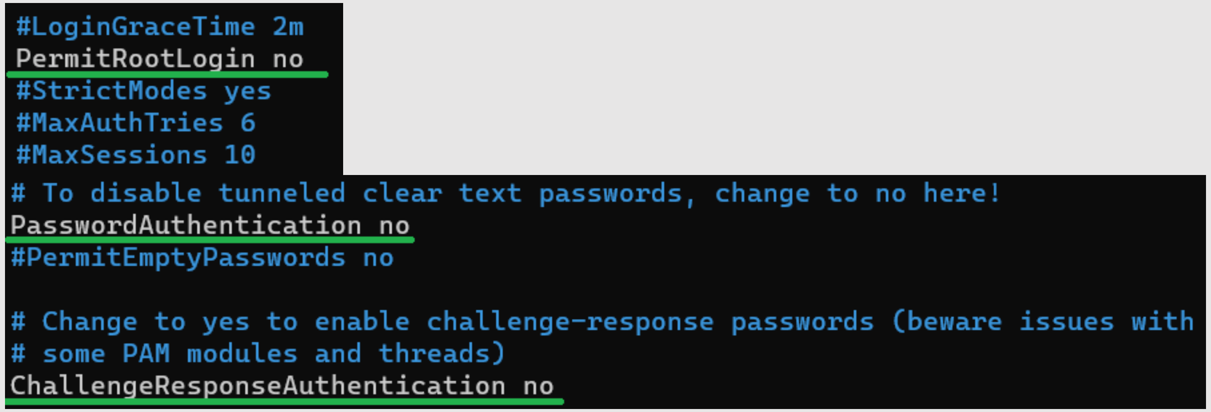
```
sudo useradd -G sudo -m USERNAME_2 -s /bin/bash
sudo passwd USERNAME_2
```

# OpenVPN

**Steps**

f) Setup SSH key authentication;

- **cat PUBLIC_KEY_PATH | ssh USERNAME_2@VPS_PUBLIC_IP "mkdir .ssh && cat >> .ssh/authorized_keys“**
- **sudo nano /etc/ssh/sshd_config**
- **sudo nano /etc/ssh/sshd_config.d/50-cloud-init.conf**



```
#LoginGraceTime 2m
PermitRootLogin no
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10
```

```
# To disable tunneled clear text passwords, change to no here!
PasswordAuthentication no
#PermitEmptyPasswords no

# Change to yes to enable challenge-response passwords (beware issues with
# some PAM modules and threads)
ChallengeResponseAuthentication no
```

- **sudo systemctl restart sshd**

g) Open a new terminal window on your local machine, and try login using the key

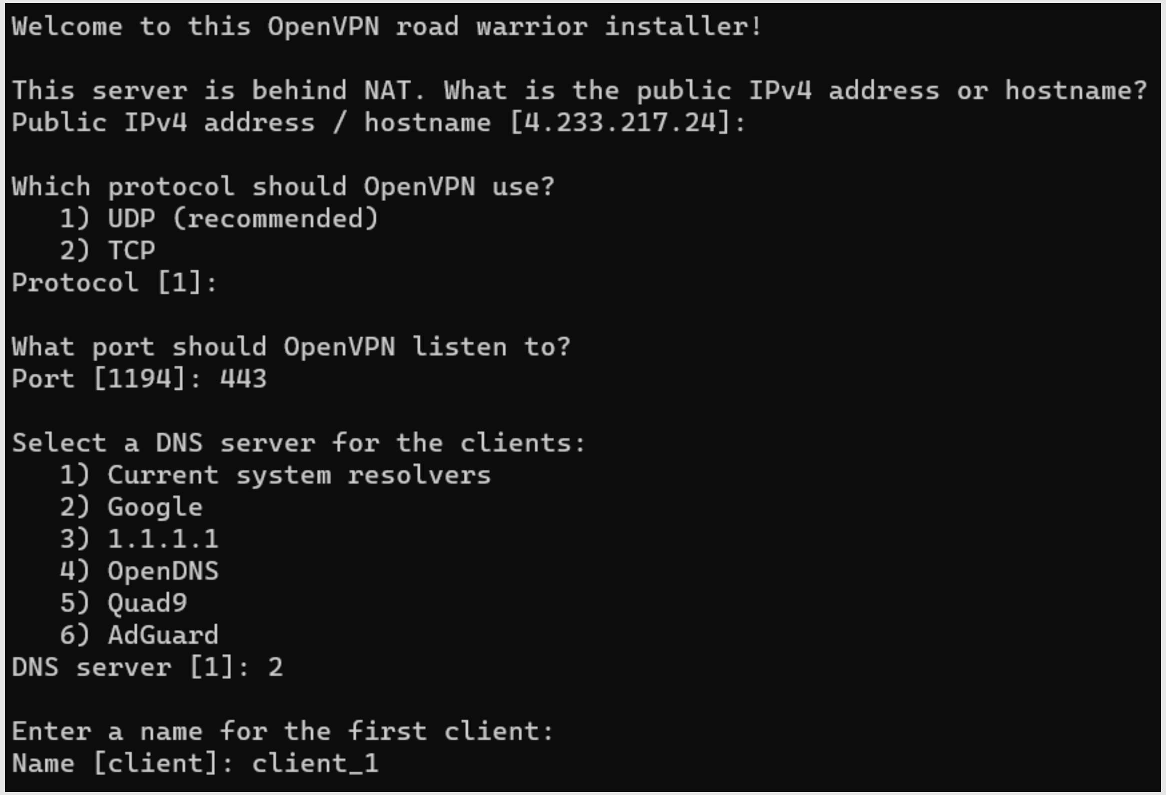
- **ssh USERNAME_2@VPS_PUBLIC_IP -i PRIVATE_KEY_PATH**

# OpenVPN

**Steps**

h) Install (and configure) OpenVPN on server-side (https://github.com/Nyr/openvpn-install)

- **sudo apt install wget**
- **wget https://git.io/vpn -O openvpn-install.sh && bash openvpn-install.sh**
- **sudo bash openvpn-install.sh**



```
Welcome to this OpenVPN road warrior installer!

This server is behind NAT. What is the public IPv4 address or hostname?
Public IPv4 address / hostname [4.233.217.24]:

Which protocol should OpenVPN use?
   1) UDP (recommended)
   2) TCP
Protocol [1]:

What port should OpenVPN listen to?
Port [1194]: 443

Select a DNS server for the clients:
   1) Current system resolvers
   2) Google
   3) 1.1.1.1
   4) OpenDNS
   5) Quad9
   6) AdGuard
DNS server [1]: 2

Enter a name for the first client:
Name [client]: client_1
```

# OpenVPN

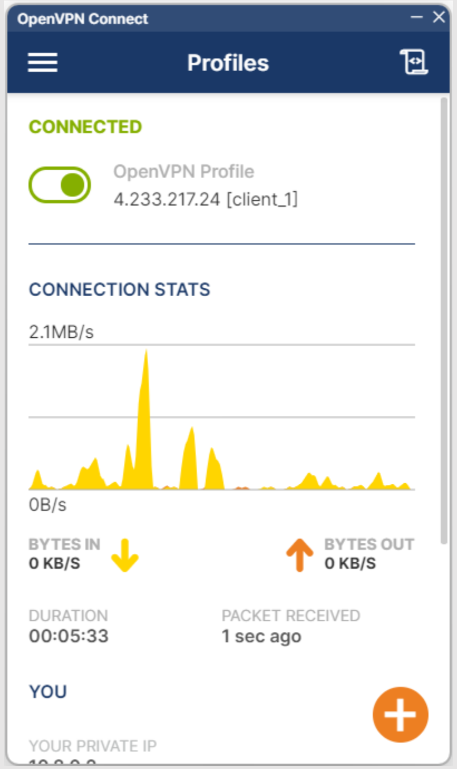
**Steps**

i) Send the newly created OpenVPN profile to the client

- **sudo mv /root/CLIENT_NAME.ovpn ~**
- **sudo chown USERNAME2 CLIENT_NAME.ovpn**
- **scp –i PRIVATE_KEY_PATH USERNAME_2@VPS_PUBLIC_IP:CLIENT_NAME.ovpn .**

j) Install OpenVPN on client-side and connect using the received profile.

- [https://openvpn.net/client-connect-vpn-for-windows/](https://openvpn.net/client-connect-vpn-for-windows/)



OpenVPN Connect

Profiles

CONNECTED

OpenVPN Profile
4.233.217.24 [client_1]

CONNECTION STATS

BYTES IN 0 KB/S

BYTES OUT 0 KB/S

DURATION 00:05:33

PACKET RECEIVED 1 sec ago

YOU

YOUR PRIVATE IP

?