Lecture 24: Security
Network-layer security

COMP 332, Spring 2018

Victoria Manfredi

WESLEYAN
UNIVERSITY

Acknowledgements: materials adapted from Computer Networking: A Top Down Approach 7th edition: ©1996-2016, J.F Kurose and K.W. Ross, All Rights Reserved as well as from slides by Abraham Matta at Boston University, and some material from Computer Networks by Tannenbaum and Wetherall.

Today

1. Announcements
   - hw9 due today at 11:59p
   - hw10 posted

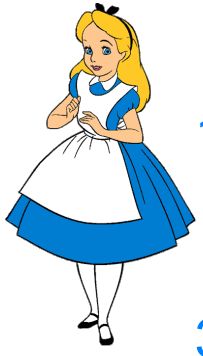
2. Transport layer security
   - real TLS/SSL

3. Network layer security
   - overview
   - Internet Protocol security (IPsec)

Transport Layer Security

REAL TLS/SSL

SSL handshake



Alice

1. Client hello ➔
   client nonce, ciphersuites

3. Verifies certificate
   generates premaster secret

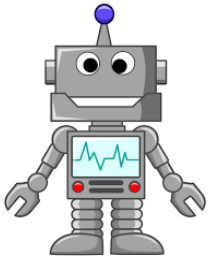
4. Premaster secret ➔
   encrypted with Bob's public key from certificate

6. Generate symmetric keys
   client nonce, server nonce, premaster, ciphersuite

8. Client hello done ➔
   MAC of all handshake msgs encrypted with client symmetric key

7. Encrypted data ➔

Bob



🡐 2. Server hello
   server nonce, chosen ciphersuite, RSA certificate

5. Generate symmetric keys
   client nonce, server nonce, premaster, ciphersuite

🡐 7. Server hello done
   MAC of all handshake msgs encrypted with server session keys

🡐 8. Encrypted data

Key derivation

Client nonce, server nonce, pre-master secret

- input into pseudo random-number generator to get master secret

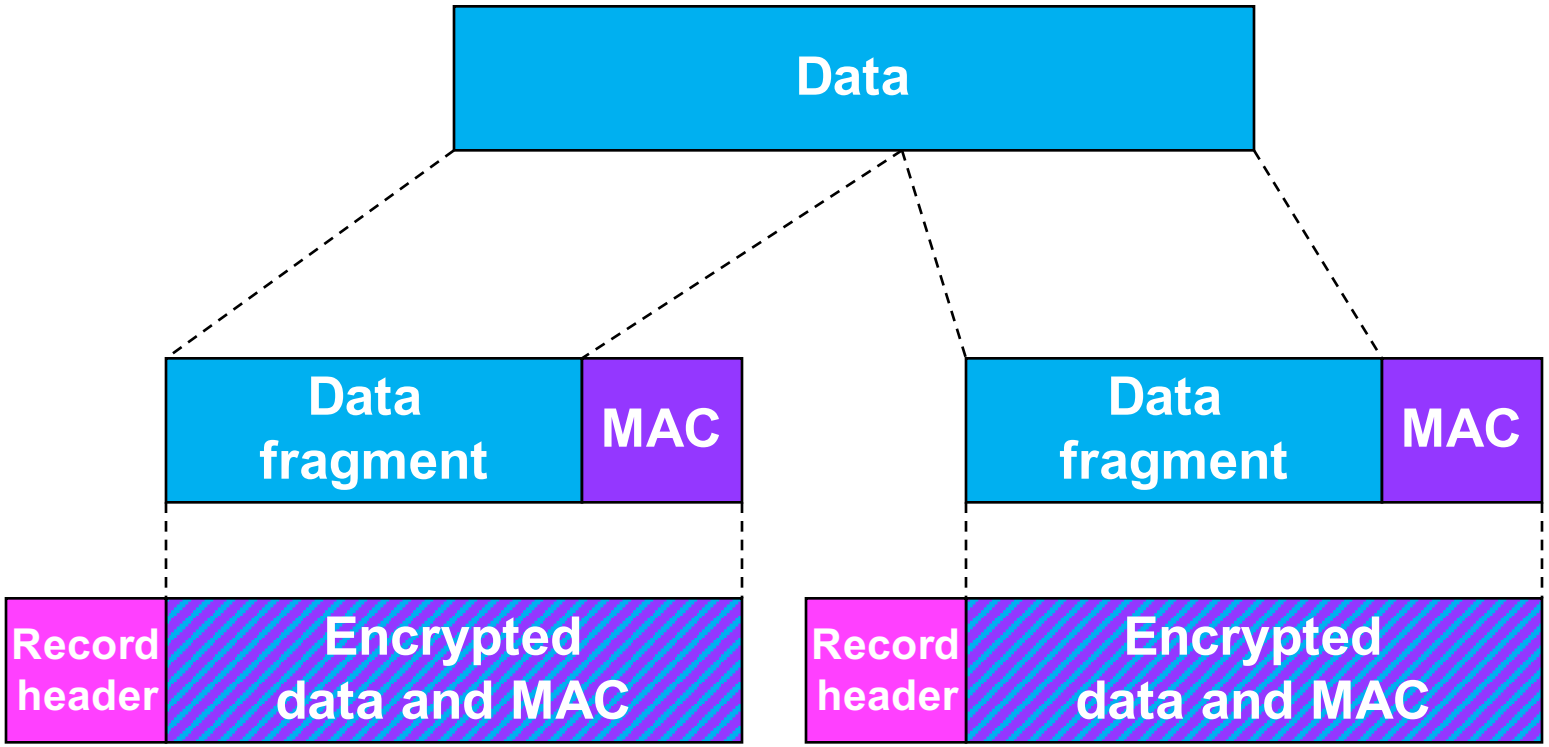
Master secret, new nonces

- input into another random-number generator to get key block

Key block sliced and diced

- client MAC key
- server MAC key
- client encryption key
- server encryption key
- client initialization vector (IV)
- server initialization vector (IV)

SSL record protocol



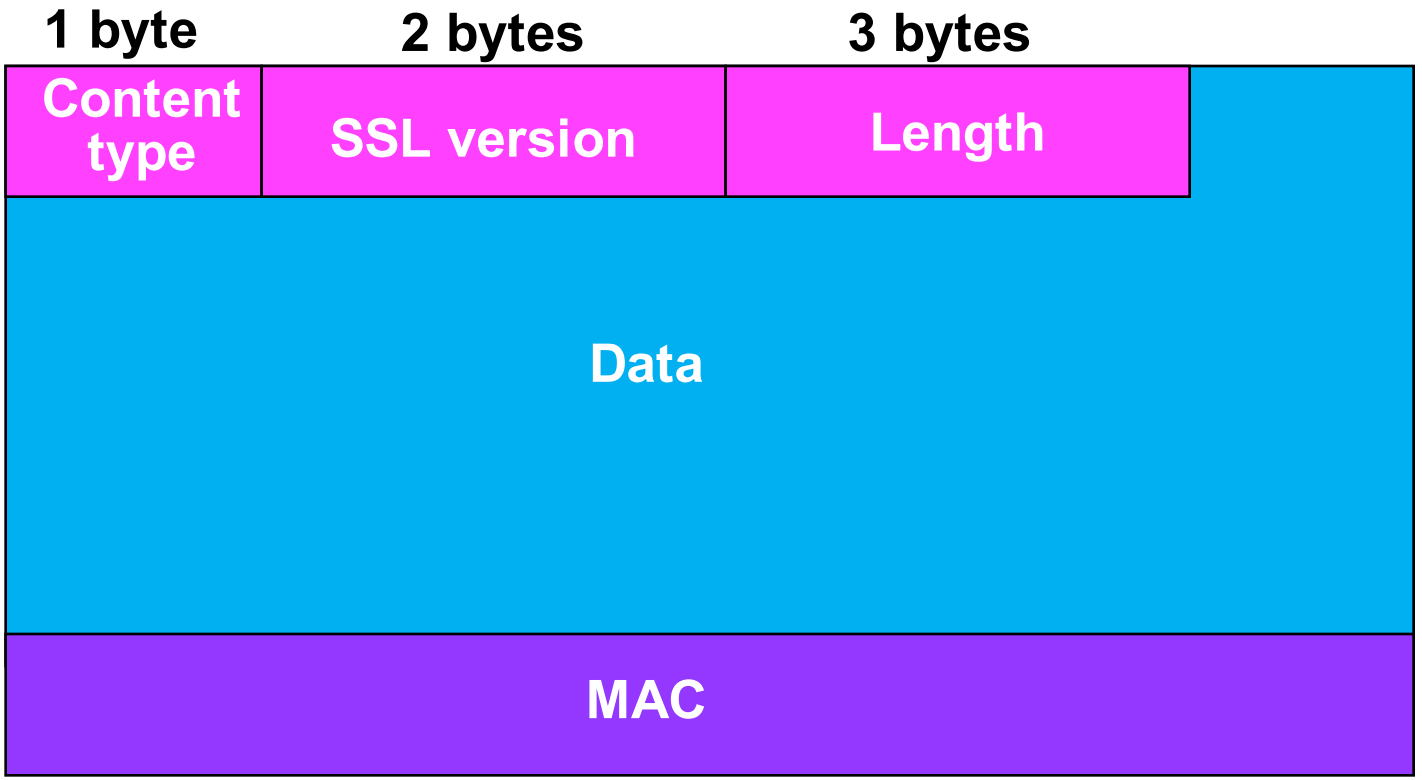
Record header: content type; version; length

MAC: includes sequence number, MAC key M_x

Fragment: each SSL fragment 2^{14} bytes (~16 Kbytes)

These records are pushed into TCP socket

SSL record format



Data and MAC encrypted (symmetric algorithm)

Wireshark

Look at TLS traffic and openssl s_client traffic

Openssl s_client

```
> echo -e "GET / HTTP/1.1\r\nHost: www.wesleyan.edu\r\n\r\n" | openssl s_client -ign_eof -connec
t www.wesleyan.edu:443
CONNECTED(00000003)
depth=3 C = SE, O = AddTrust AB, OU = AddTrust External TTP Network, CN = AddTrust External CA R
oot
verify return:1
depth=2 C = US, ST = New Jersey, L = Jersey City, O = The USERTRUST Network, CN = USERTrust RSA
Certification Authority
verify return:1
depth=1 C = US, ST = MI, L = Ann Arbor, O = Internet2, OU = InCommon, CN = InCommon RSA Server C
A
verify return:1
depth=0 C = US, postalCode = 06457, ST = CT, L = Middletown, street = 237 High Street, O = Wesle
yan University, OU = ITS, CN = www.wesleyan.edu
verify return:1
---
Certificate chain
 0 s:/C=US/postalCode=06457/ST=CT/L=Middletown/street=237 High Street/O=Wesleyan University/OU=I
TS/CN=www.wesleyan.edu
   i:/C=US/ST=MI/L=Ann Arbor/O=Internet2/OU=InCommon/CN=InCommon RSA Server CA
 1 s:/C=SE/O=AddTrust AB/OU=AddTrust External TTP Network/CN=AddTrust External CA Root
   i:/C=SE/O=AddTrust AB/OU=AddTrust External TTP Network/CN=AddTrust External CA Root
 2 s:/C=US/ST=New Jersey/L=Jersey City/O=The USERTRUST Network/CN=USERTrust RSA Certification Au
thority
   i:/C=SE/O=AddTrust AB/OU=AddTrust External TTP Network/CN=AddTrust External CA Root
 3 s:/C=US/ST=MI/L=Ann Arbor/O=Internet2/OU=InCommon/CN=InCommon RSA Server CA
   i:/C=US/ST=New Jersey/L=Jersey City/O=The USERTRUST Network/CN=USERTrust RSA Certification Au
thority
---
Server certificate
-----BEGIN CERTIFICATE-----
```

Network Layer Security

OVERVIEW

We've secured the transport layer

… but what about the network layer?

- or, what's not protected when we use TLS? What is protected?

How to protect against

- spoofing of IP addresses?
- replaying of IP packets?
- leaking of information in IP header?
- leaking of information in TCP header?
- …

Network layer security

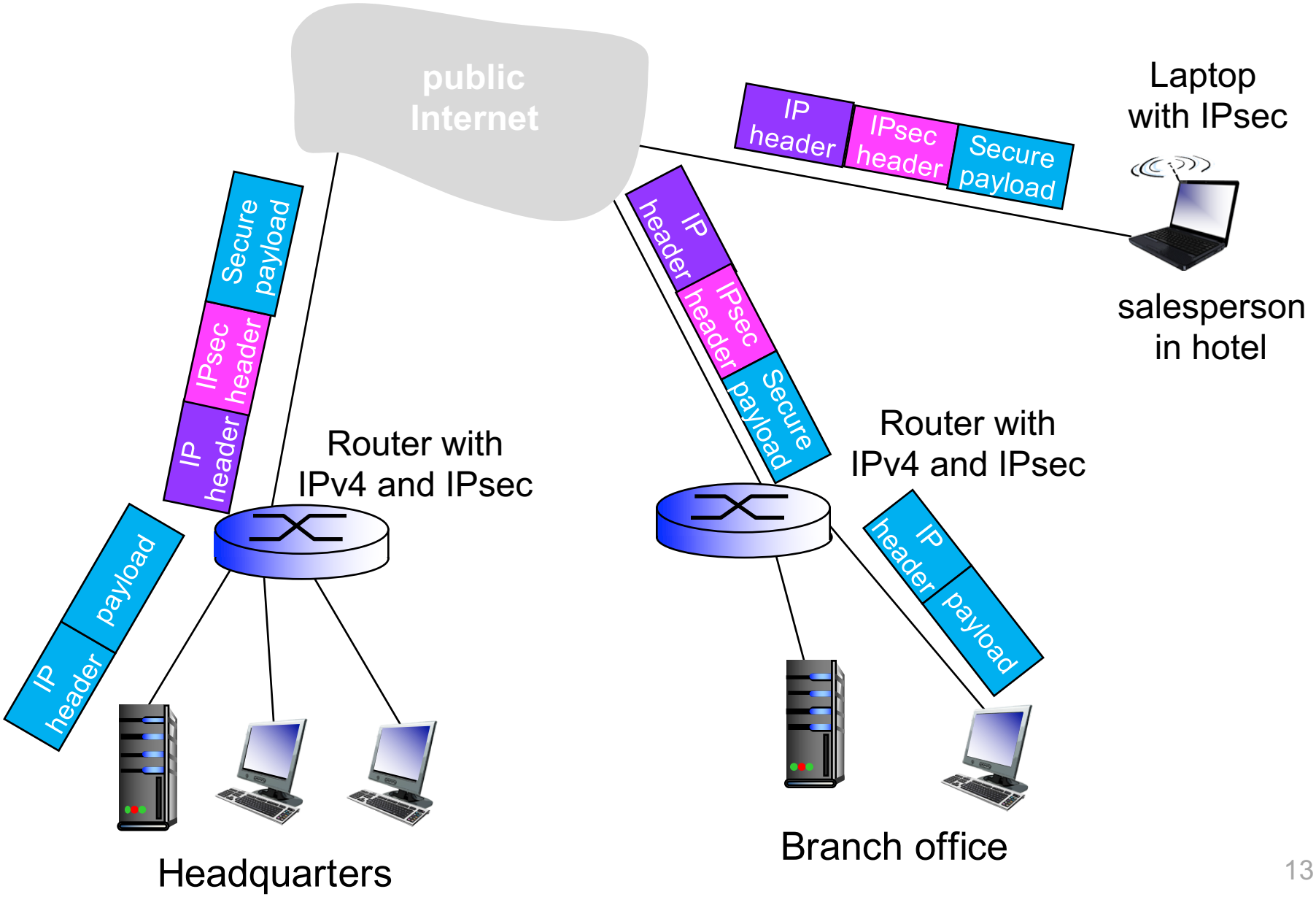
IPsec: Internet Protocol Security

- secures IP packets sent between 2 network entities
  - sending entity encrypts packet and its payload
    - TCP segment, UDP datagram, ICMP pkt, OSPF msg, ….
    - web pages, e-mail, P2P file transfers, TCP SYN packets, IP addr, …

VPNs are one big application of IPsec

- institutions want private networks for security but costly
- instead institution's inter-office traffic sent over public Internet
  - **but** encrypted before entering public Internet

Virtual Private Networks (VPNs)



Wesleyan VPN traffic

```
10733  45.964470  webvpn.wesleyan.edu                    vmanfredismbp2.wireless.wesleyan.edu
10734  45.964680  webvpn.wesleyan.edu                    vmanfredismbp2.wireless.wesleyan.edu
10735  45.964700  vmanfredismbp2.wireless.wesleyan.edu   webvpn.wesleyan.edu
10736  45.964863  webvpn.wesleyan.edu                    vmanfredismbp2.wireless.wesleyan.edu
10737  45.965052  webvpn.wesleyan.edu                    vmanfredismbp2.wireless.wesleyan.edu
10738  45.965066  vmanfredismbp2.wireless.wesleyan.edu   webvpn.wesleyan.edu

▶ Frame 10733: 1350 bytes on wire (10800 bits), 1350 bytes captured (10800 bits) on interface 0
▶ Ethernet II, Src: JuniperN_1e:18:01 (3c:8a:b0:1e:18:01), Dst: Apple_73:43:26 (78:4f:43:73:43:26)
▼ Internet Protocol Version 4, Src: webvpn.wesleyan.edu (129.133.2.4), Dst: vmanfredismbp2.wireless.wes
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▶ Differentiated Services Field: 0x20 (DSCP: CS1, ECN: Not-ECT)
    Total Length: 1336
    Identification: 0xd31b (54043)
  ▶ Flags: 0x02 (Don't Fragment)
    Fragment offset: 0
    Time to live: 62
    Protocol: Encap Security Payload (50)
    Header checksum: 0xa39b [validation disabled]
    [Header checksum status: Unverified]
    Source: webvpn.wesleyan.edu (129.133.2.4)
    Destination: vmanfredismbp2.wireless.wesleyan.edu (129.133.187.174)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
▼ Encapsulating Security Payload
    ESP SPI: 0x0f19838c (253330316)
    ESP Sequence: 241
```

Network Layer Security

IPSEC

IPsec services

2 protocols providing different service models

1. Authentication Header (AH) protocol
   - provides
     - source authentication (of data, not user)
     - data integrity (using HMAC)
     - protection against replay attacks (seq #s)
   - does **not** provide confidentiality

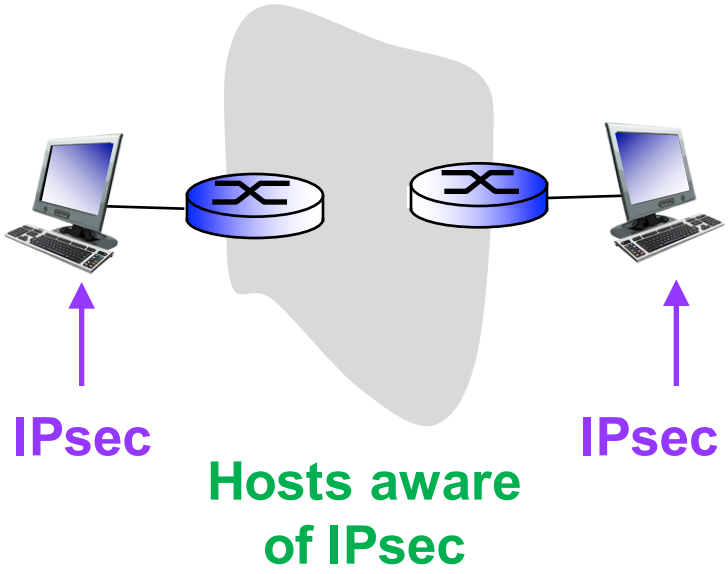
2. Encapsulation Security Protocol (ESP)
   - additionally provides confidentiality (symmetric key)
   - more widely used than AH

Choose 1 of these protocols to use

2 modes

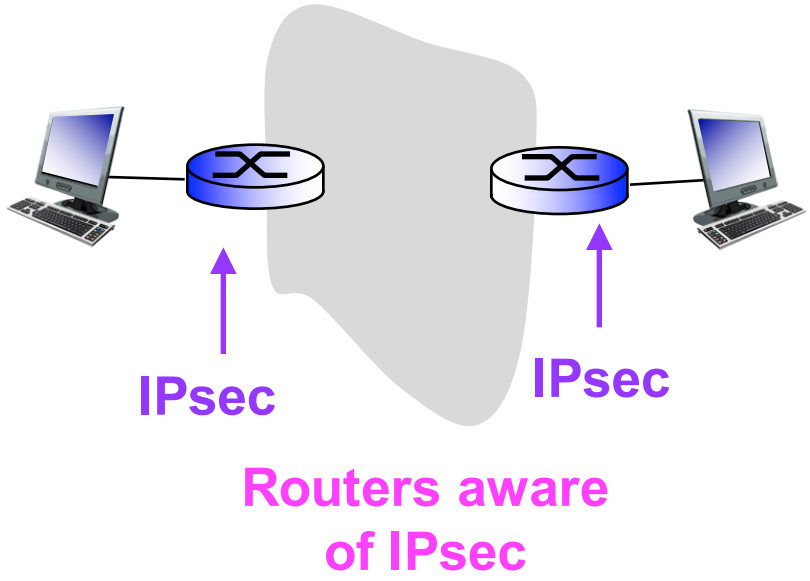
Transport mode

- primarily for communication between end hosts
- protects upper level protocols



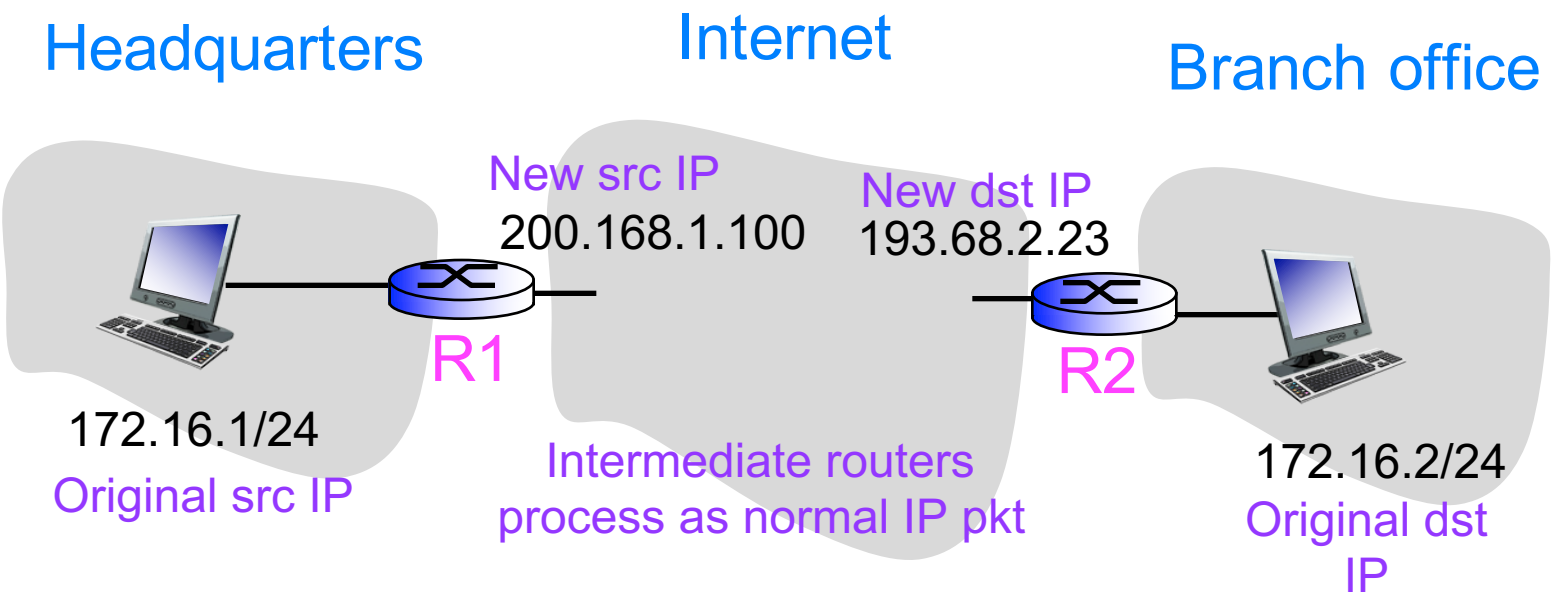
Tunnel mode

- primarily for communication between gateway routers
- e.g., as with VPNs



Choose 1 of these modes to use

IPsec example



4 combinations possible

Host mode with AH	Host mode with ESP
Tunnel mode with AH	Tunnel mode with ESP

Most common and most important

Internet Key Exchange (IKE) protocol

Can be used outside of IPsec as well as with IPsec

- exchanges and negotiates security and keys
- IKE used by IPsec to establish security associations

Security association (SA)

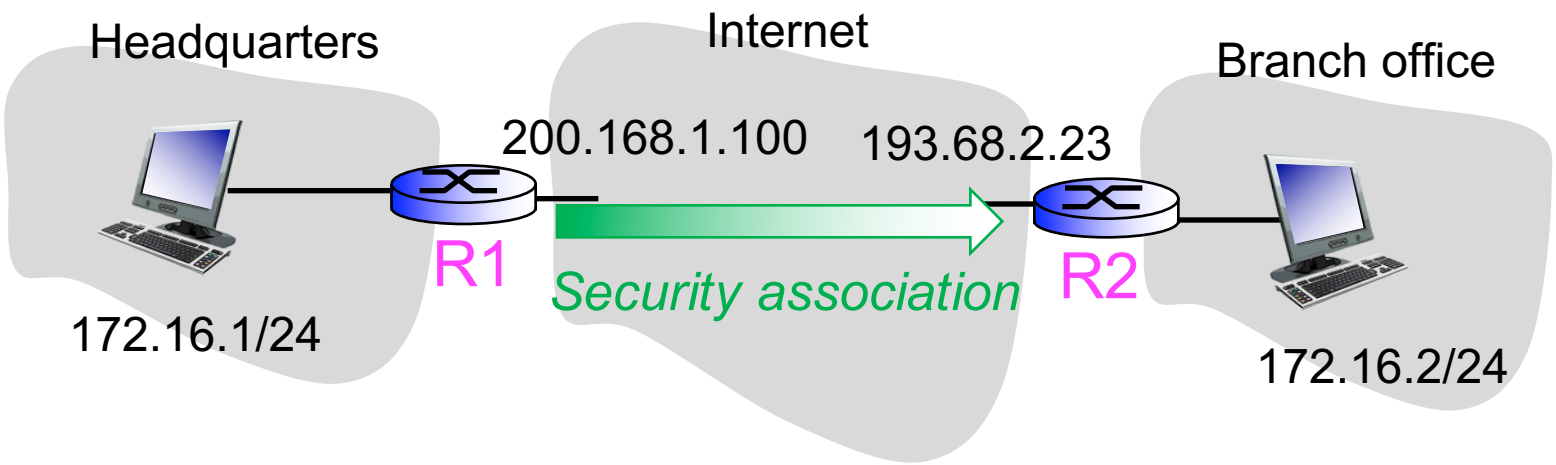
- keeps track of state associated with connection
- established before sending data, maintained by each endpoint
- exists from sending to receiving entity
  - 1-way communication; for 2-way need 2 SAs

Q: Why have a SA?

- IP is connectionless, but IPsec is connection oriented, like TCP

Example SA from R1 to R2

SA keeps track of state associated with connection



R1 stores for SA

- 32-bit SA identifier: Security Parameter Index (SPI)
- origin SA interface (200.168.1.100)
- dst SA interface (193.68.2.23)

There can be problems with IPsec and NAT, proxies

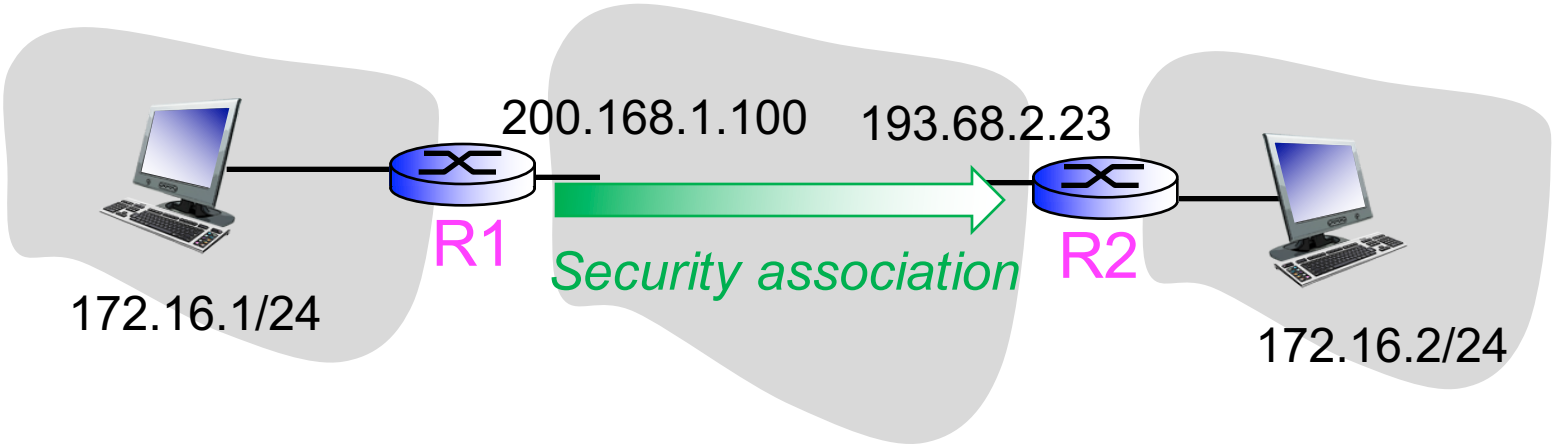
- type of encryption used (e.g., 3DES with CBC)
- encryption key
- type of integrity check used (e.g., HMAC with MD5)
- authentication key

Security Association Database (SAD)

Where endpoints store state for different SAs

When IPsec pkt sent or received

– endpoint looks in SAD to determine how to process pkt



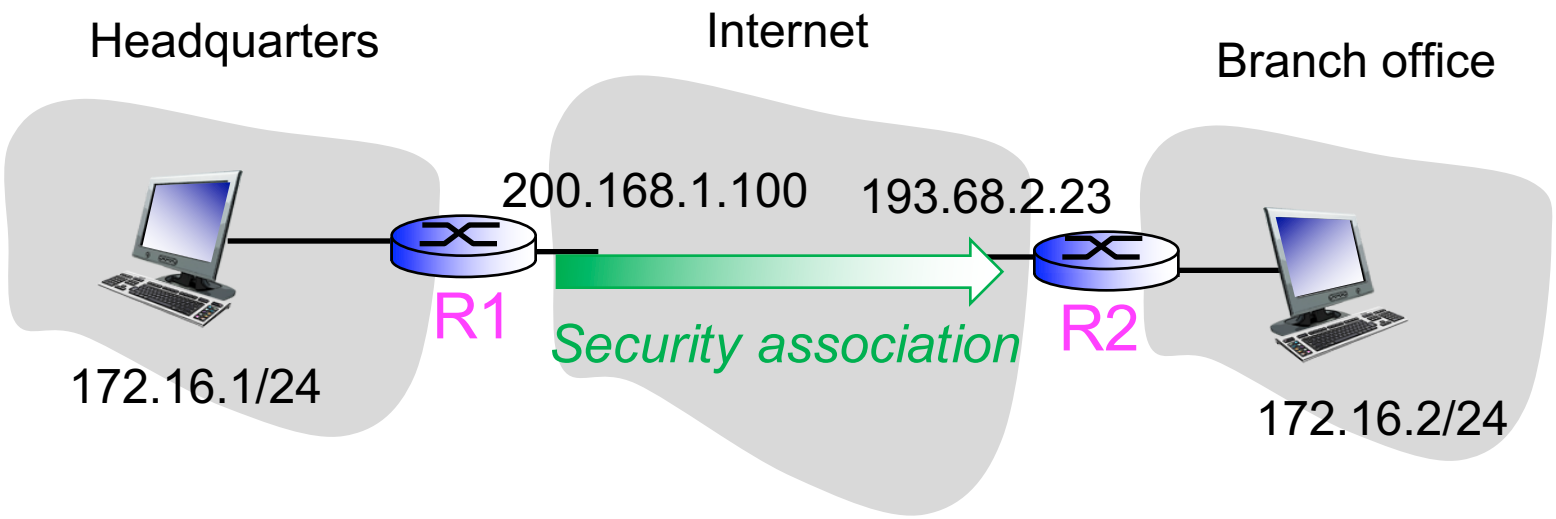
R1 sends IPsec pkt: R1 accesses SAD to determine how to process

R2 gets IPsec pkt: R2 uses SPI into index SAD, processes pkt accordingly

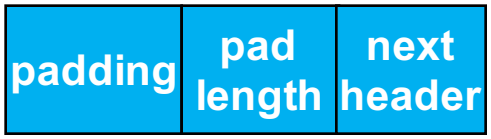
Network Layer Security

IPSEC: TUNNEL MODE + ESP

R1: converts original pkt to IPsec pket



2. Encrypts result using algorithm & key specified by SA

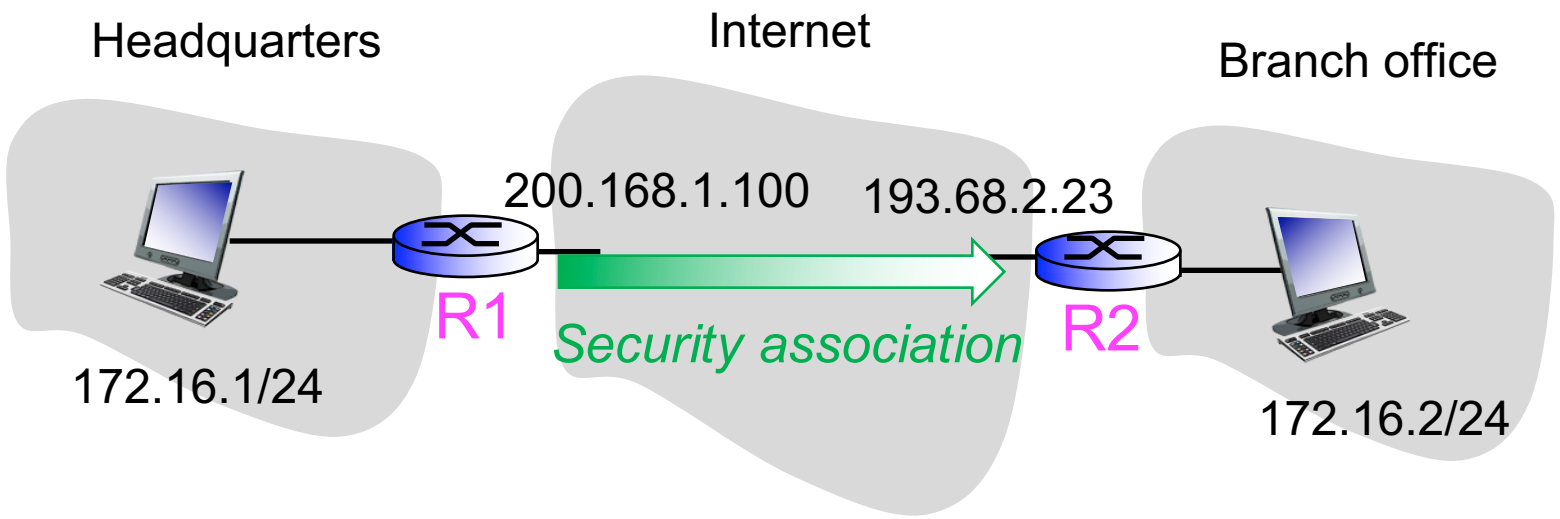


1. Appends ESP trailer field to back of original pkt

Q: What is padding used for?
Block ciphers need to fill block

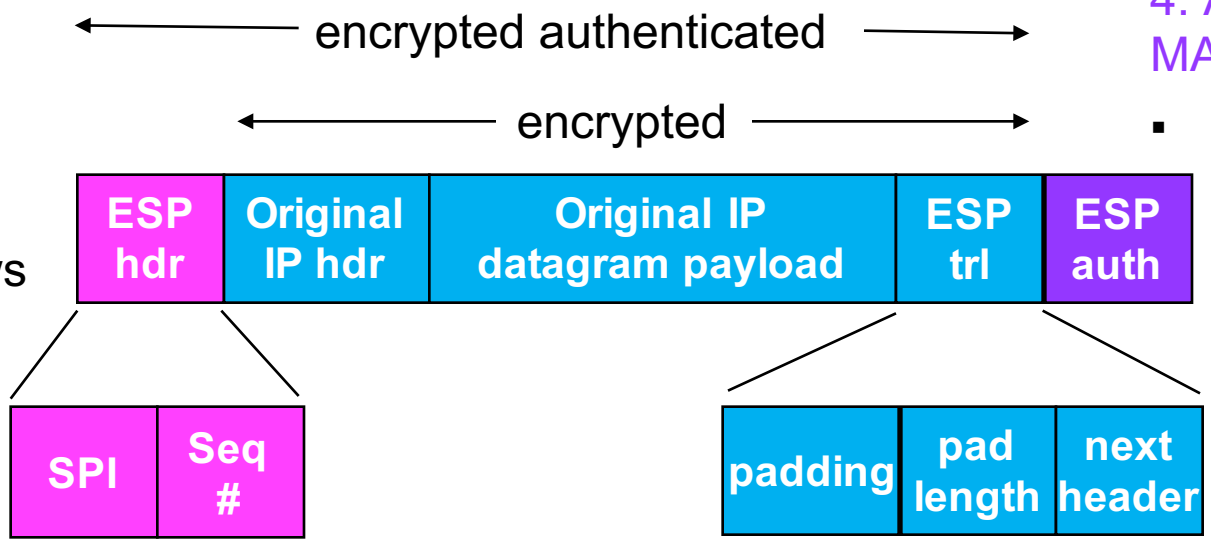
Q: What is next header?
Type of data in IP pkt payload, e.g., UDP

R1: converts original pkt to IPsec pket



3. Appends ESP header to front

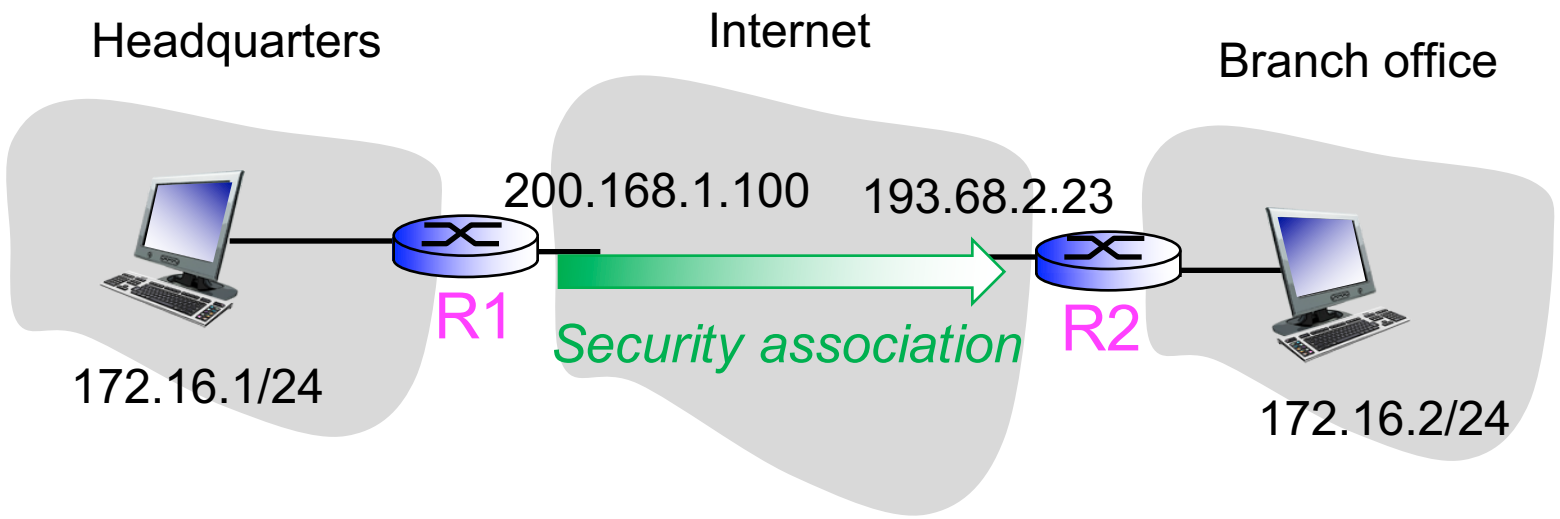
- SPI: so receiver knows which SA pkt belongs
- Seq #: to thwart replay



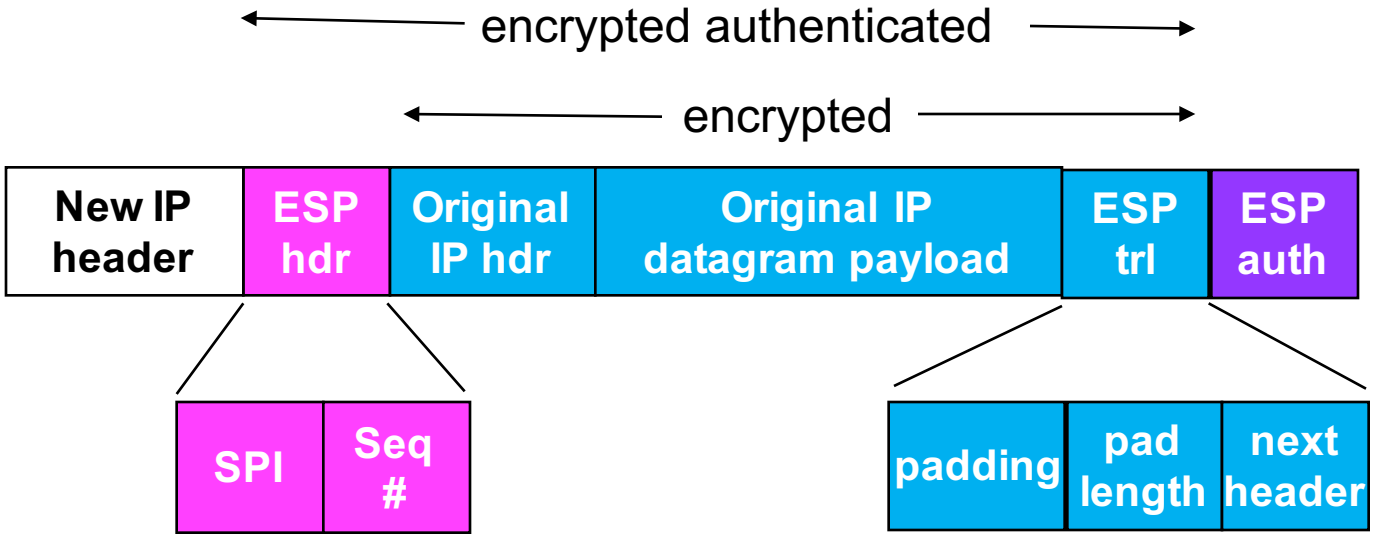
4. Appends auth MAC

- over encrypted, using algorithm, keys in SA

R1: converts original pkt to IPsec pket



5. Creates new IP header, appends before payload



Wesleyan VPN traffic

```
10733  45.964470  webvpn.wesleyan.edu                    vmanfredismbp2.wireless.wesleyan.edu
10734  45.964680  webvpn.wesleyan.edu                    vmanfredismbp2.wireless.wesleyan.edu
10735  45.964700  vmanfredismbp2.wireless.wesleyan.edu   webvpn.wesleyan.edu
10736  45.964863  webvpn.wesleyan.edu                    vmanfredismbp2.wireless.wesleyan.edu
10737  45.965052  webvpn.wesleyan.edu                    vmanfredismbp2.wireless.wesleyan.edu
10738  45.965066  vmanfredismbp2.wireless.wesleyan.edu   webvpn.wesleyan.edu

▶ Frame 10733: 1350 bytes on wire (10800 bits), 1350 bytes captured (10800 bits) on interface 0
▶ Ethernet II, Src: JuniperN_1e:18:01 (3c:8a:b0:1e:18:01), Dst: Apple_73:43:26 (78:4f:43:73:43:26)
▼ Internet Protocol Version 4, Src: webvpn.wesleyan.edu (129.133.2.4), Dst: vmanfredismbp2.wireless.wes
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▶ Differentiated Services Field: 0x20 (DSCP: CS1, ECN: Not-ECT)
    Total Length: 1336
    Identification: 0xd31b (54043)
  ▶ Flags: 0x02 (Don't Fragment)
    Fragment offset: 0
    Time to live: 62
    Protocol: Encap Security Payload (50)
    Header checksum: 0xa39b [validation disabled]
    [Header checksum status: Unverified]
    Source: webvpn.wesleyan.edu (129.133.2.4)
    Destination: vmanfredismbp2.wireless.wesleyan.edu (129.133.187.174)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
▼ Encapsulating Security Payload
    ESP SPI: 0x0f19838c (253330316)
    ESP Sequence: 241
```

Trudy between R1 and R2, doesn't know keys

Will Trudy see

- original contents of pkt?
- src, dst IP addr, transport protocol, port?

Can Trudy

- flip bits without detection?
- masquerade as R1 using R1's IP address?
- replay a packet?