

# Lecture 25: Security

## Network-layer security

COMP 411, Fall 2022

Victoria Manfredi



**Acknowledgements:** materials adapted from Computer Networking: A Top Down Approach 7<sup>th</sup> 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. Transport layer security

- real TLS/SSL

## 2. Network layer security

- overview
- Internet Protocol security (IPsec)

# Transport Layer Security

## **REAL TLS**

# Toy TLS is incomplete

How long are fields? Which encryption protocols? How do client and server negotiate encryption algorithms?

## TLS Handshake

- confidentiality
  - client and server negotiate encryption algorithms before data transfer
    - i.e., negotiate ciphersuite
  - derive keys used in data exchange
- integrity
  - check if handshake tampered with based on hash of handshake msgs
- authentication
  - using public key and server's certificate
  - optional client authentication

# TLS cipher suite

## Negotiation: client, server agree on cipher suite

- client offers choice server picks one

TLS\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA

Key exchange  
algorithm: public-  
key

Symmetric encryption  
algorithm: block cipher to  
encrypt msg stream

MAC  
algorithm

## Which supported depends on TLS version

- TLS 1.2 supports many cipher suites
- TLS 1.3 supports many fewer cipher suites

# Cipher suites

- ▼ TLSv1 Record Layer: Handshake Protocol: Client Hello
  - Content Type: Handshake (22)
  - Version: TLS 1.0 (0x0301)
  - Length: 144
- ▼ Handshake Protocol: Client Hello
  - Handshake Type: Client Hello (1)
  - Length: 140
  - Version: TLS 1.0 (0x0301)
  - ▶ Random: 5ae5dac626d5483a3ea908c593979d44170f3e628f26688d...
  - Session ID Length: 32
  - Session ID: e84d0000076240b35c57828829153be712af150acb327e17...
  - Cipher Suites Length: 32
  - ▼ Cipher Suites (16 suites)
    - Cipher Suite: TLS\_EMPTY\_RENEGOTIATION\_INFO\_SCSV (0x00ff)
    - Cipher Suite: TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CBC\_SHA384 (0xc024)
    - Cipher Suite: TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CBC\_SHA256 (0xc023)
    - Cipher Suite: TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CBC\_SHA (0xc00a)
    - Cipher Suite: TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CBC\_SHA (0xc009)
    - Cipher Suite: TLS\_ECDHE\_ECDSA\_WITH\_3DES\_EDE\_CBC\_SHA (0xc008)
    - Cipher Suite: TLS\_ECDHE\_RSA\_WITH\_AES\_256\_CBC\_SHA384 (0xc028)
    - Cipher Suite: TLS\_ECDHE\_RSA\_WITH\_AES\_128\_CBC\_SHA256 (0xc027)
    - Cipher Suite: TLS\_ECDHE\_RSA\_WITH\_AES\_256\_CBC\_SHA (0xc014)
    - Cipher Suite: TLS\_ECDHE\_RSA\_WITH\_AES\_128\_CBC\_SHA (0xc013)
    - Cipher Suite: TLS\_ECDHE\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA (0xc012)
    - Cipher Suite: TLS\_RSA\_WITH\_AES\_256\_CBC\_SHA256 (0x003d)
    - Cipher Suite: TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA256 (0x003c)
    - Cipher Suite: TLS\_RSA\_WITH\_AES\_256\_CBC\_SHA (0x0035)
    - Cipher Suite: TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA (0x002f)
    - Cipher Suite: TLS\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA (0x000a)

# TLS Client Hello

- ▶ Frame 50: 203 bytes on wire (1624 bits), 203 bytes captured (1624 bits) on interface 0
- ▶ Ethernet II, Src: Apple\_73:43:26 (78:4f:43:73:43:26), Dst: JuniperN\_1e:18:01 (3c:8a:b0:1e:18:01)
- ▶ Internet Protocol Version 4, Src: vmanfredismbp2.wireless.wesleyan.edu (129.133.187.174), Dst: 129.133.187.174
- ▶ Transmission Control Protocol, Src Port: 63173, Dst Port: 443, Seq: 41885059, Ack: 3555367379, Len: 144
- ▼ Secure Sockets Layer
  - ▼ TLSv1 Record Layer: Handshake Protocol: Client Hello
    - Content Type: Handshake (22)
    - Version: TLS 1.0 (0x0301)
    - Length: 144
  - ▼ Handshake Protocol: Client Hello
    - Handshake Type: Client Hello (1)
    - Length: 140
    - Version: TLS 1.0 (0x0301)
    - ▶ Random: 5ae5dac626d5483a3ea908c593979d44170f3e628f26688d...
    - Session ID Length: 32
    - Session ID: e84d0000076240b35c57828829153be712af150acb327e17...
    - Cipher Suites Length: 32
    - ▶ Cipher Suites (16 suites)
    - Compression Methods Length: 1
    - ▶ Compression Methods (1 method)
    - Extensions Length: 35
    - ▶ Extension: supported\_groups (len=8)
    - ▶ Extension: ec\_point\_formats (len=2)
    - ▶ Extension: status\_request (len=5)
    - ▶ Extension: signed\_certificate\_timestamp (len=0)
    - ▶ Extension: extended\_master\_secret (len=0)



# TLS 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**

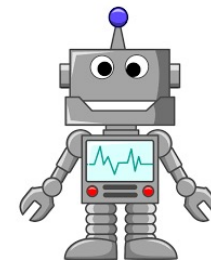
client nonce, server nonce,  
premaster, ciphersuite

**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**

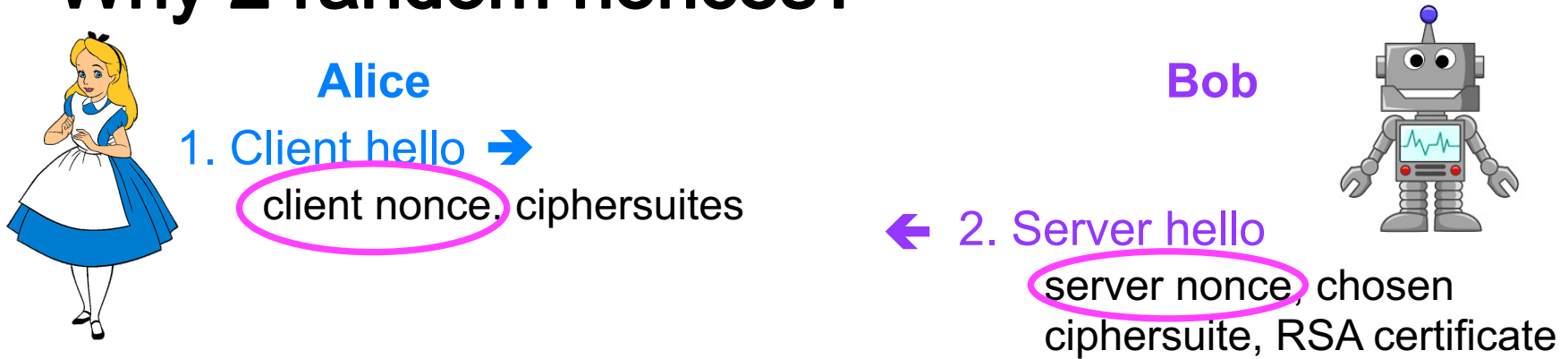
MAC of all handshake msgs  
encrypted with server session keys

**← 8. Encrypted data**

**Protect handshake from tampering**



# Why 2 random nonces?



## Suppose Trudy sniffs all messages between Alice & Bob

- next day, Trudy sets up TCP connection with Bob
  - replays sequence of records
  - Bob (Amazon) thinks Alice made two separate orders for same thing

## Solution

- Bob sends different random nonce for each connection
  - causes encryption keys to be different on the 2 days
  - Trudy's messages will fail Bob's integrity check

# Key derivation

## Client nonce, server nonce, pre-master secret

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

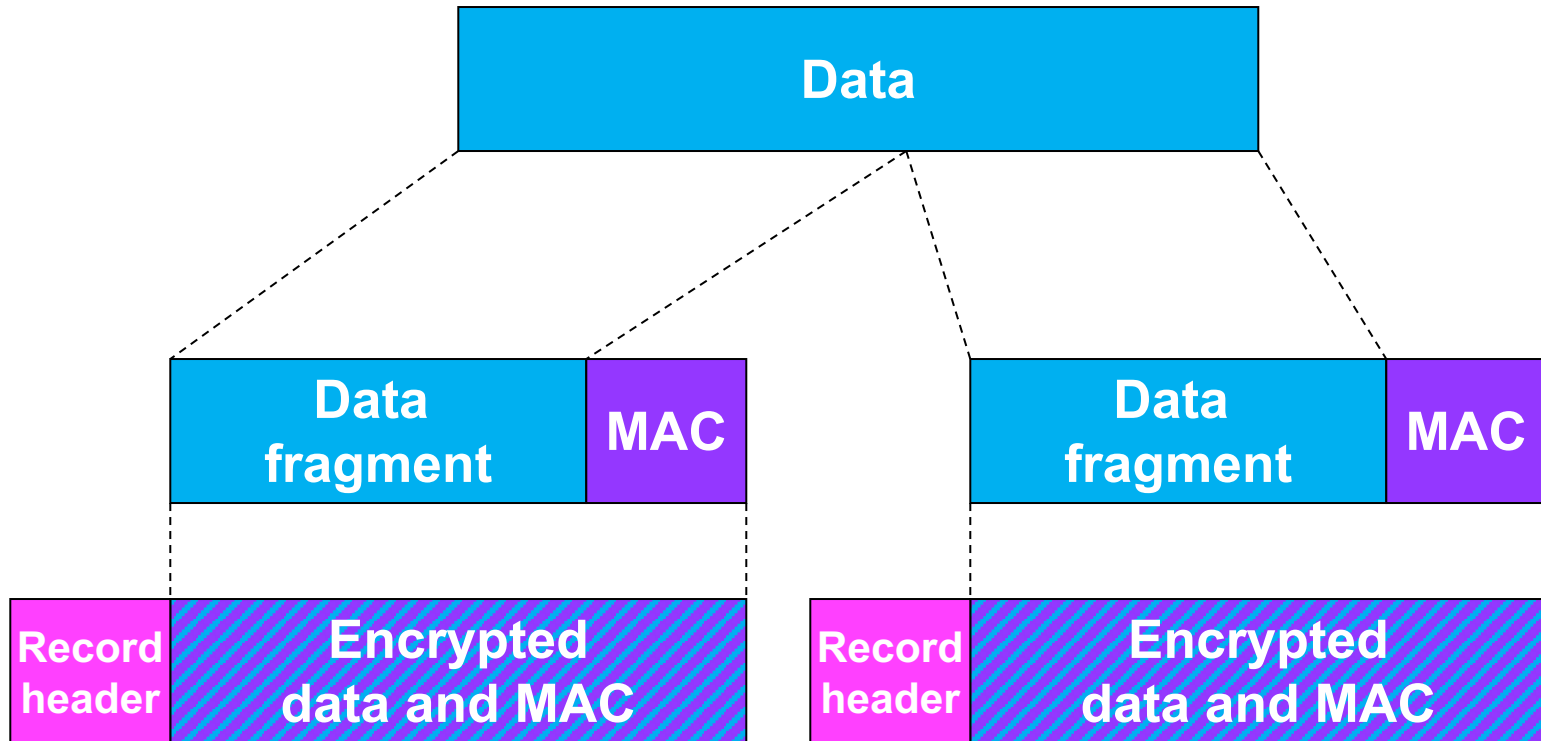
## Master secret, new nonces

- 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



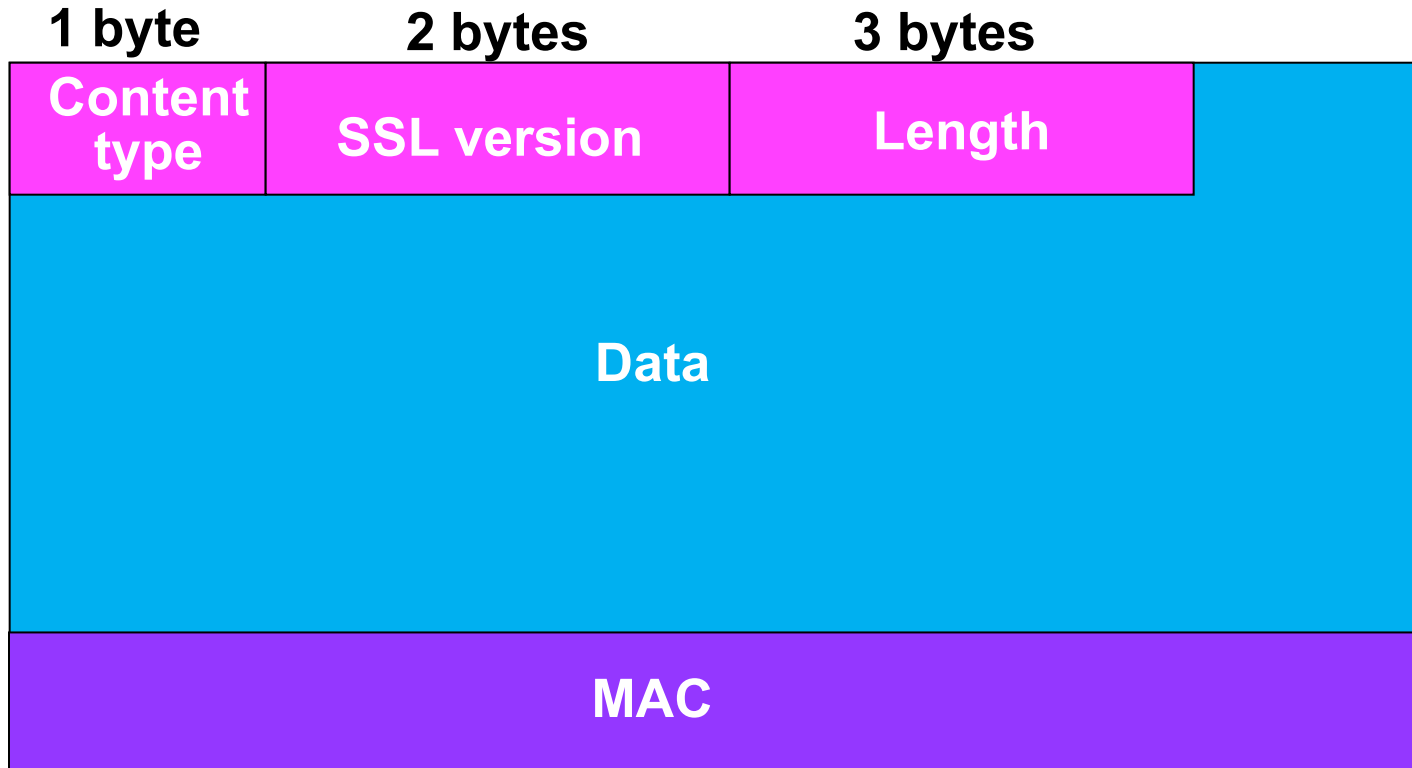
**Record header:** content type; version; length

**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 -connect www.wesleyan.edu:443
CONNECTED(00000003)
depth=3 C = SE, O = AddTrust AB, OU = AddTrust External TTP Network, CN = AddTrust External CA Root
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 CA
verify return:1
depth=0 C = US, postalCode = 06457, ST = CT, L = Middletown, street = 237 High Street, O = Wesleyan 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=ITS/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 Authority
   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 Authority
---
Server certificate
-----BEGIN CERTIFICATE-----
MIITVTCGCCD3eAwTRALC1LD7nn01zDSPTePKviuQwDOY1KeZThycNAQELR0Aw
```

# Discussion

## Email

- end-end encryption
- PGP: pretty good privacy

## Microsoft office

- cloud issues



# 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

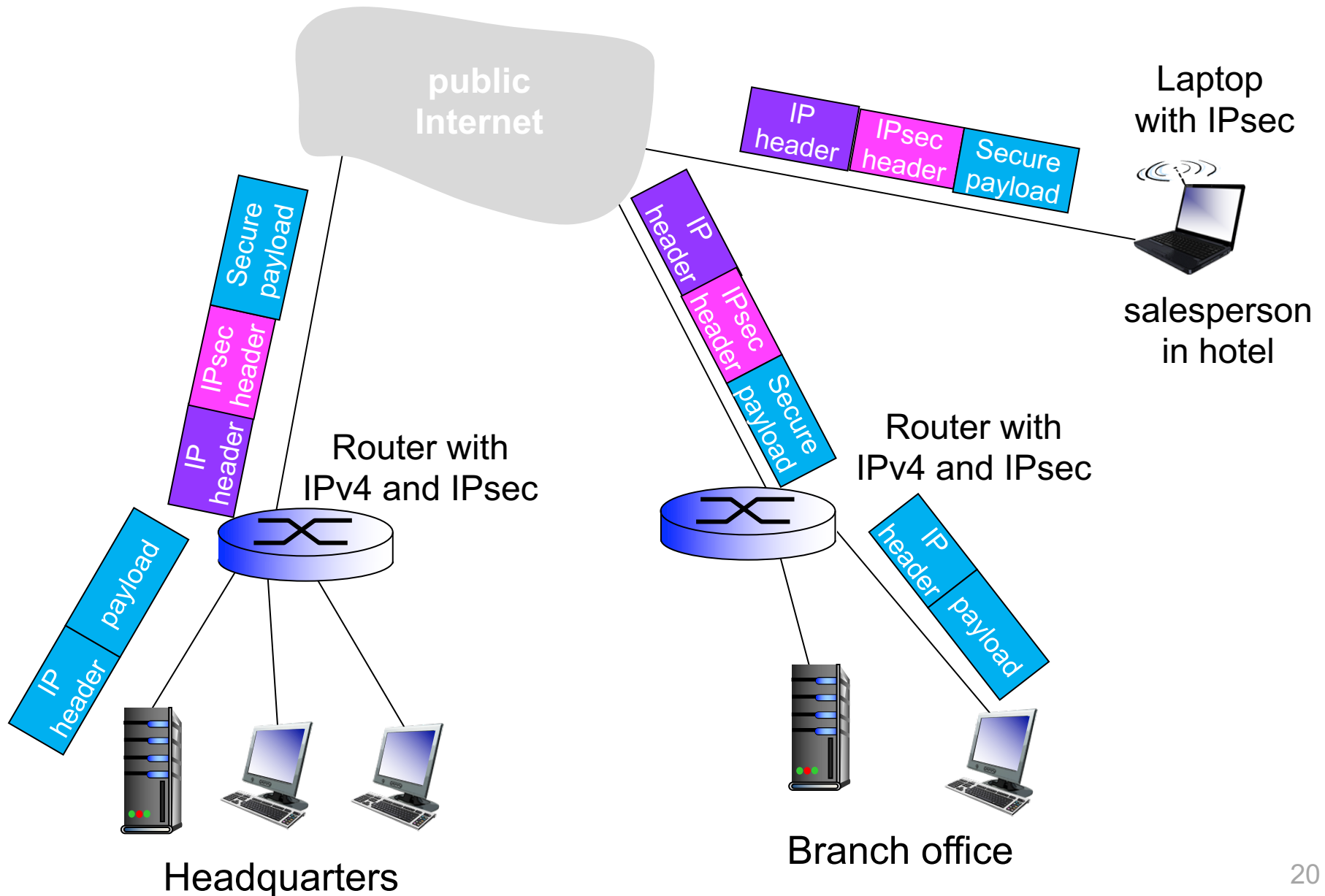
## IPsec: Internet Protocol Security

- secures IP packets sent between 2 network entities
  - sending entity encrypts packet and its payload
    - TCP segment, UDP datagram, ICMP pkt, ...
    - web pages, e-mail, P2P file transfers, TCP SYN, 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.wesleyan.edu (129.133.187.174)
  - 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**

# 2 protocols

## 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

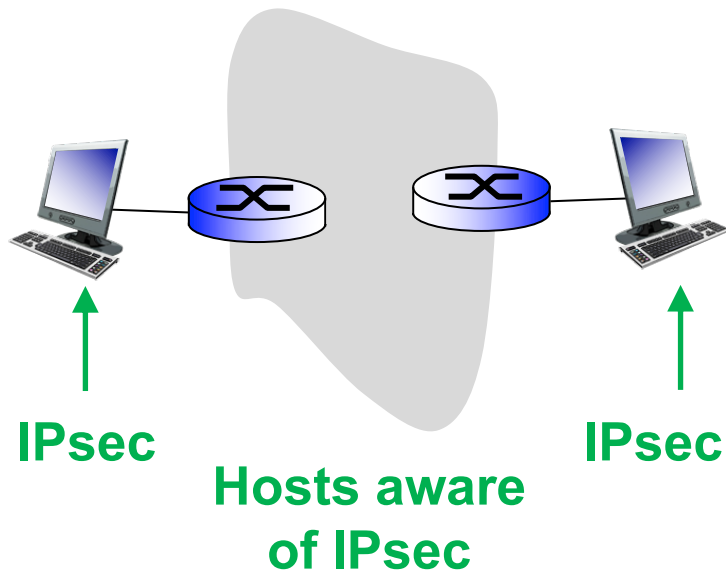
**Choose 1 of these protocols to use**



# 2 modes

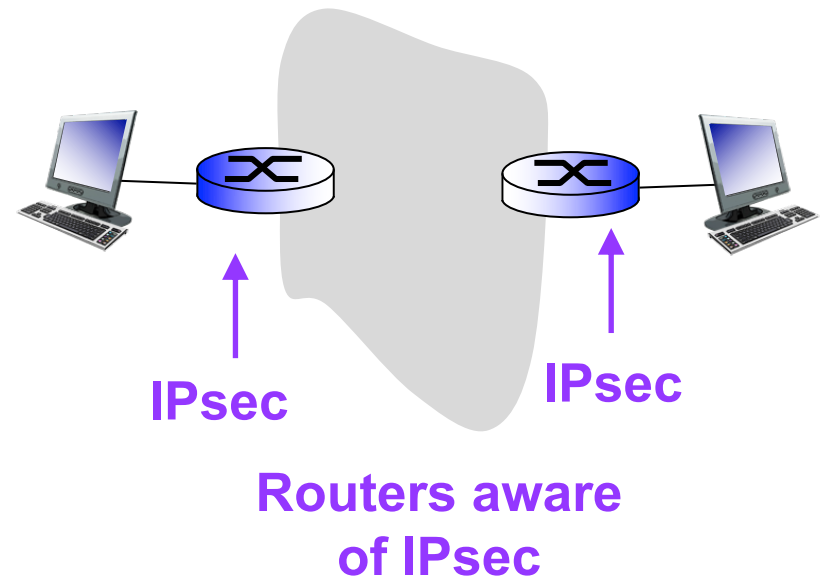
## 1. Transport mode

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



## 2. Tunnel mode

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



Choose 1 of these modes to use

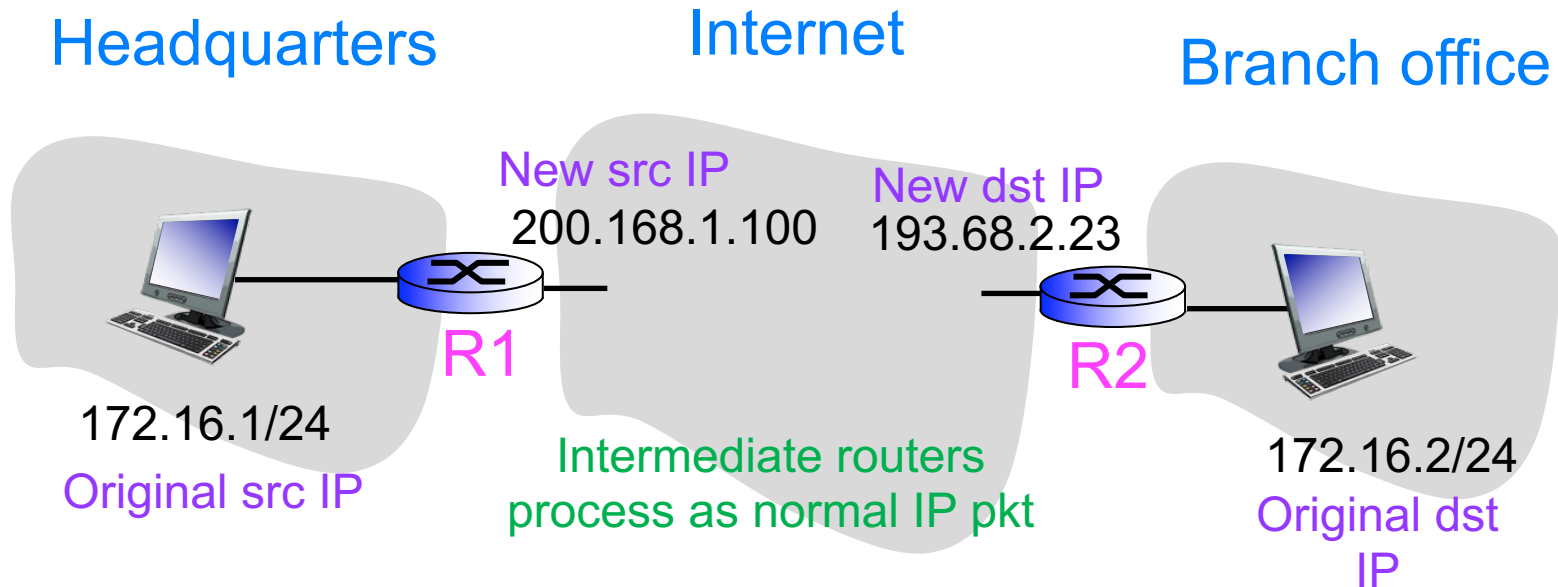
# 4 combinations possible

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

Most common and  
most important



# IPsec example



# 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)

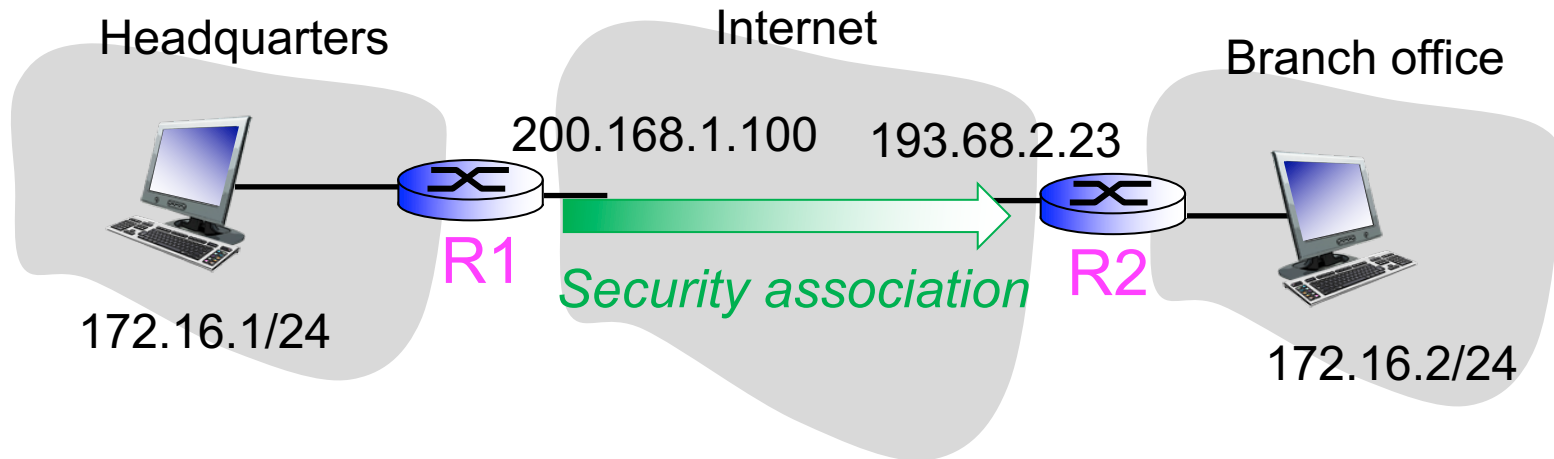
- keeps track of state associated with connection
- established before sending data, maintained by each endpoint
- exists from sending to receiving entity
  - 1-way communication; for 2-way need 2 SAs

## Q: Why have 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

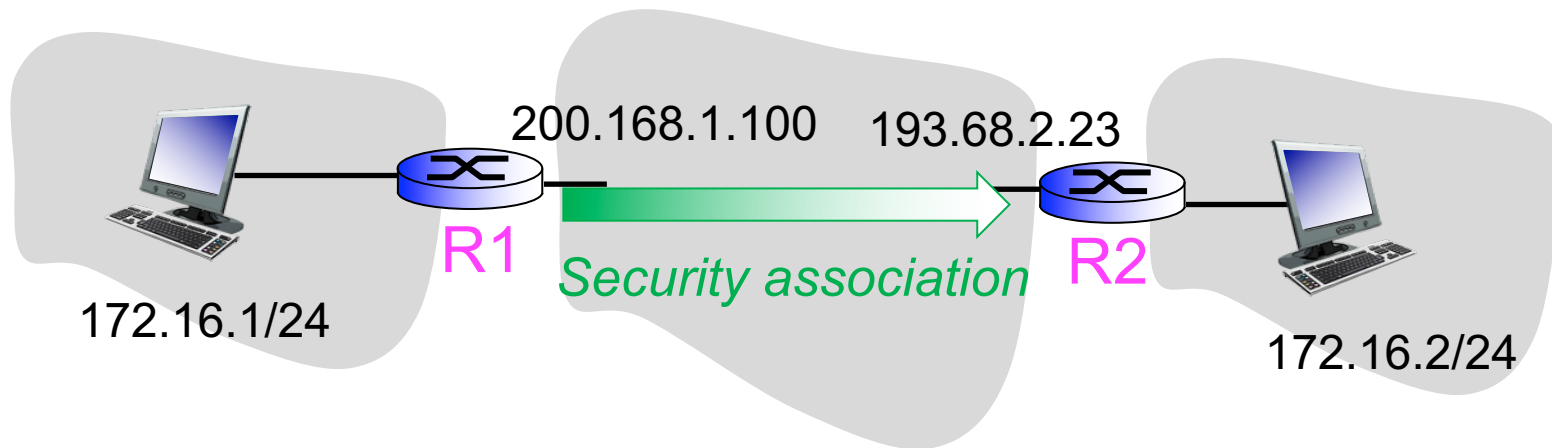
- 32-bit SA identifier: Security Parameter Index (SPI)
  - origin SA interface (200.168.1.100)
  - dst SA interface (193.68.2.23)
  - type of encryption used
  - encryption key
  - type of integrity check used
  - authentication key
- } There can be problems with IPsec and NAT, proxies

# 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

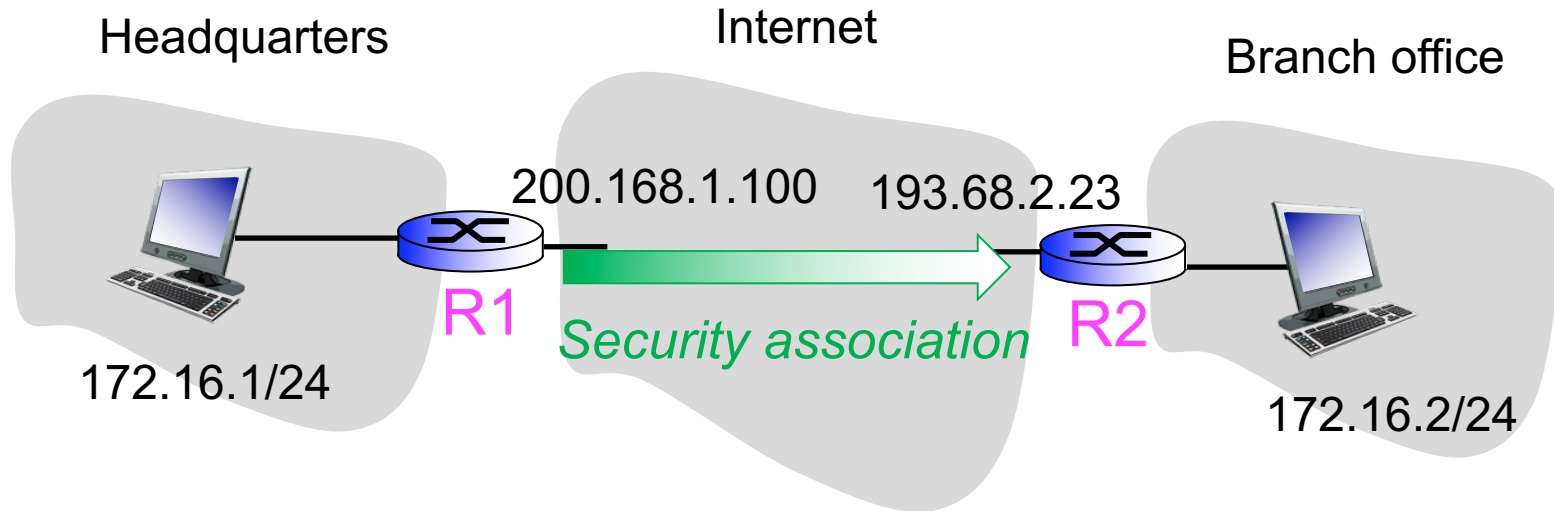
**R2 gets IPsec pkt:** R2 uses SPI into index SAD, processes pkt accordingly

# Network Layer Security

## **IPSEC: TUNNEL MODE + ESP**



# R1: converts original pkt to IPsec pkt



2. Encrypts  
result using  
algorithm & key  
specified by SA



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

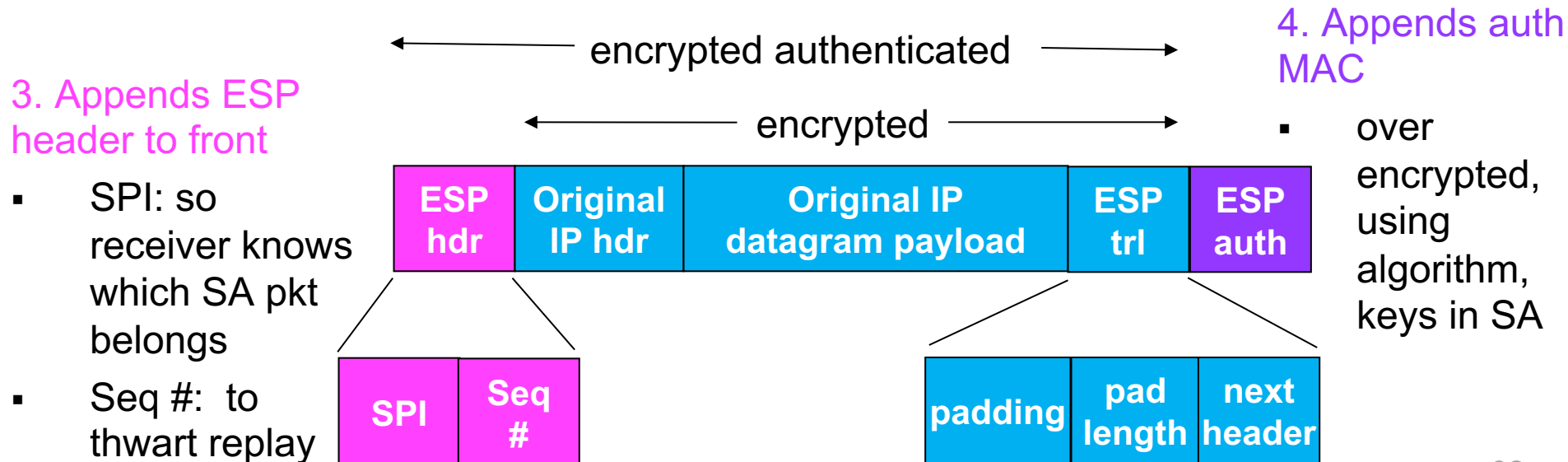
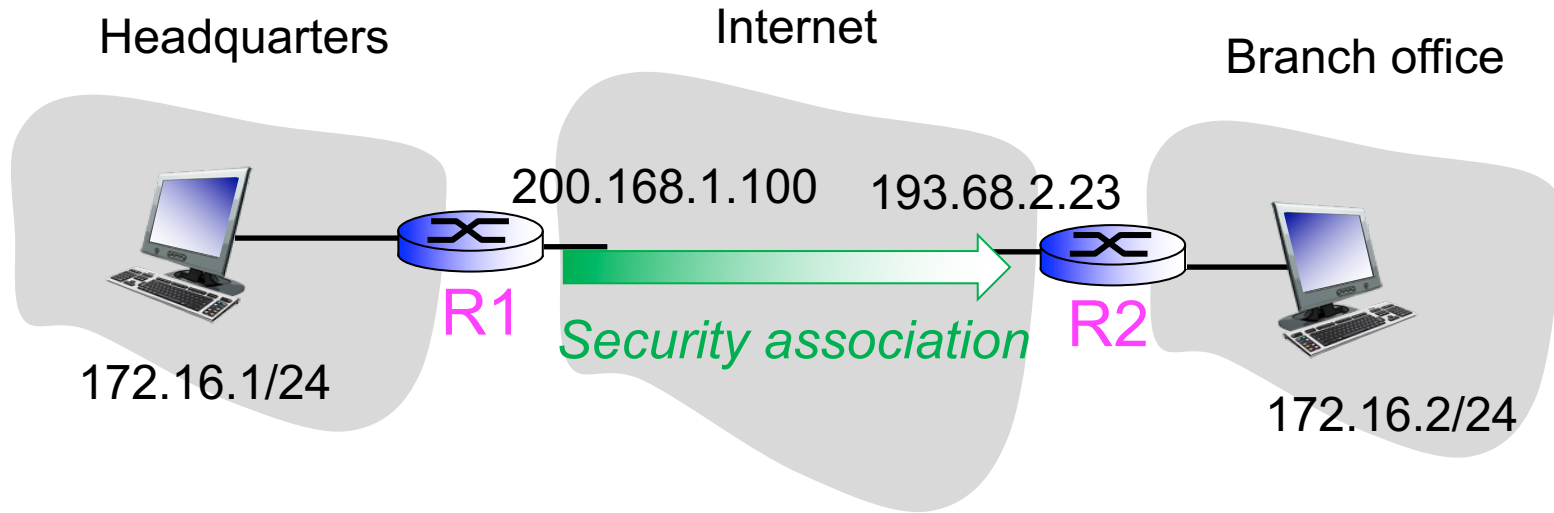
Q: What is next header?

Type of data in IP pkt payload, e.g., UDP

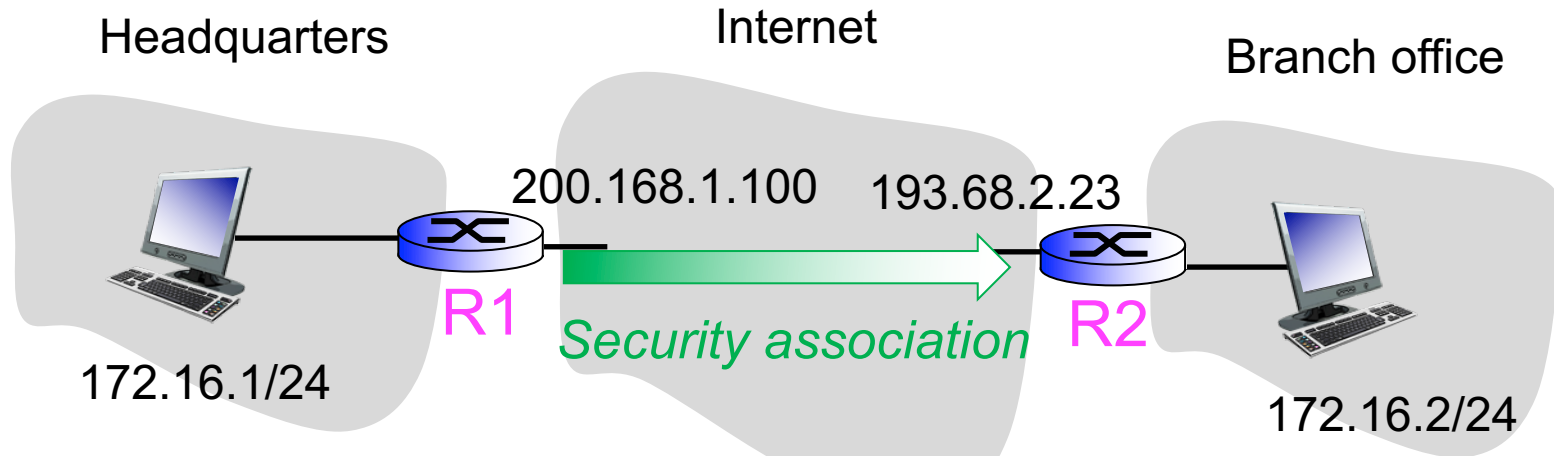


1. Appends ESP  
trailer field to back  
of original pkt

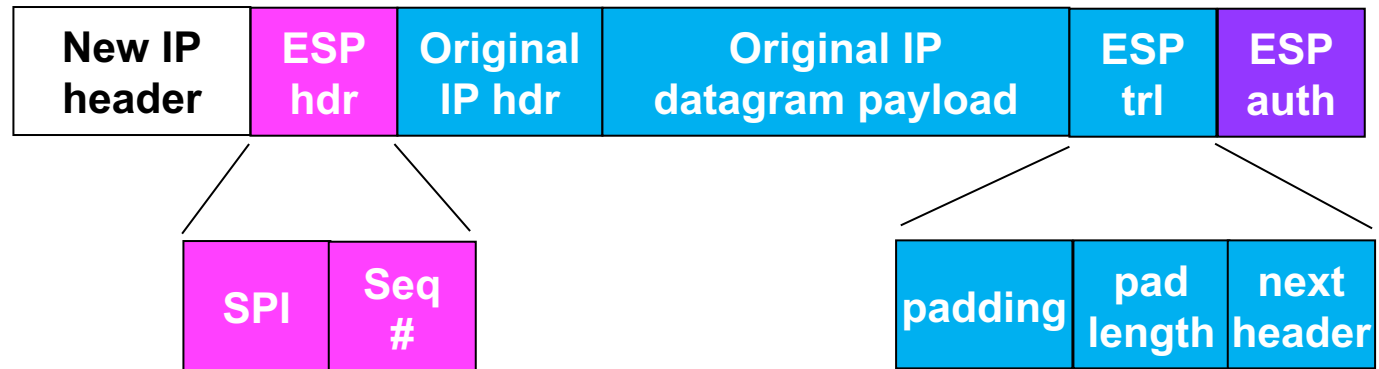
# R1: converts original pkt to IPsec pkt



# R1: converts original pkt to IPsec pkt



← encrypted authenticated →  
← encrypted →



5. Creates new IP header, appends before payload

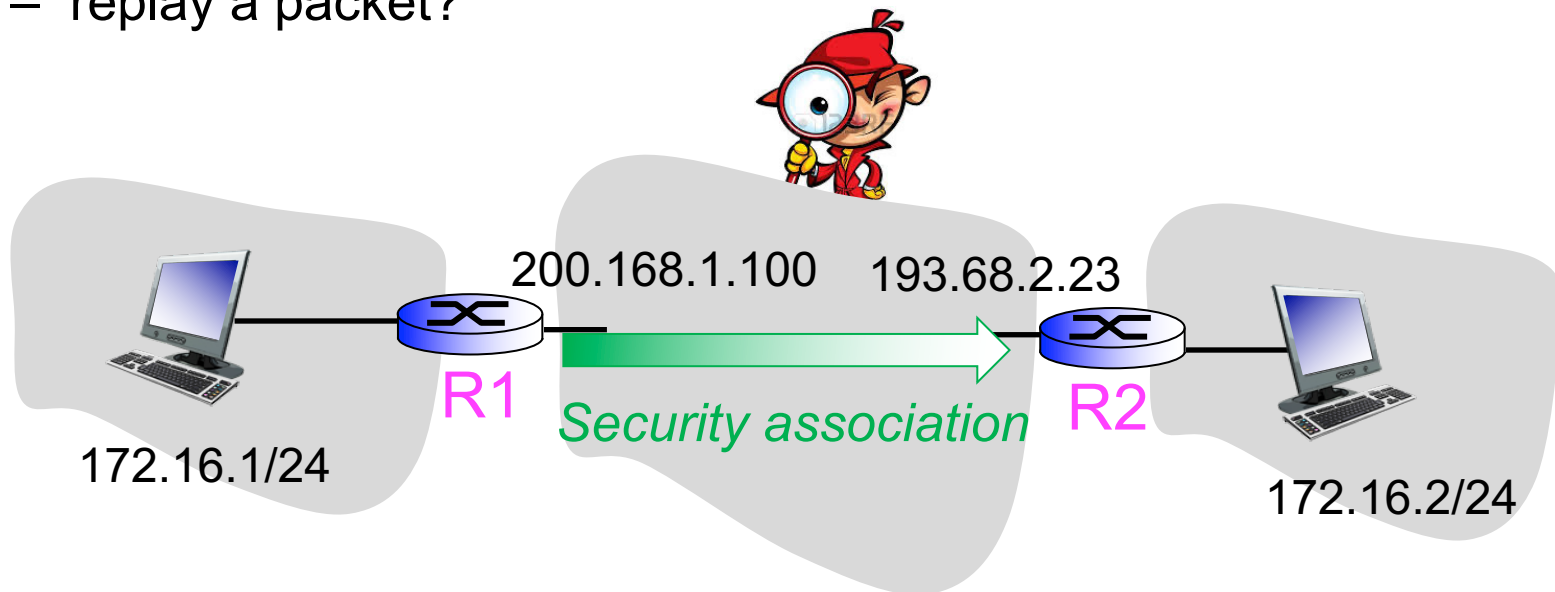
# Trudy between R1 and R2, doesn't know keys

## 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?



# 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.wesleyan.edu (129.133.187.174)
  - 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