Chapter 8 roadmap

8.1 What is network security?

8.2 Principles of cryptography

8.3 Message integrity

8.4 Securing e-mail

8.5 Securing TCP connections: SSL

8.6 Network layer security: IPsec

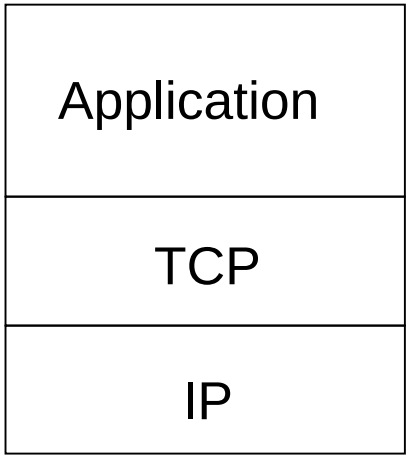
8.7 Securing wireless LANs

8.8 Operational security: firewalls and IDS

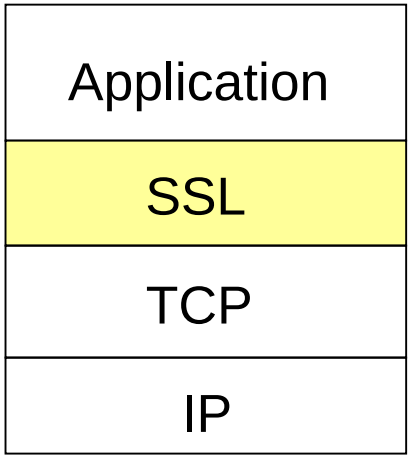
SSL: Secure Sockets Layer

- widely deployed security protocol
  - supported by almost all browsers, web servers
  - https
  - billions $/year over SSL
- mechanisms: [Woo 1994], implementation: Netscape
- variation -TLS: transport layer security, RFC 2246
- provides
  - *confidentiality*
  - *integrity*
  - *authentication*
- original goals:
  - Web e-commerce transactions
  - encryption (especially credit-card numbers)
  - Web-server authentication
  - optional client authentication
  - minimum hassle in doing business with new merchant
- available to all TCP applications
  - secure socket interface

SSL and TCP/IP



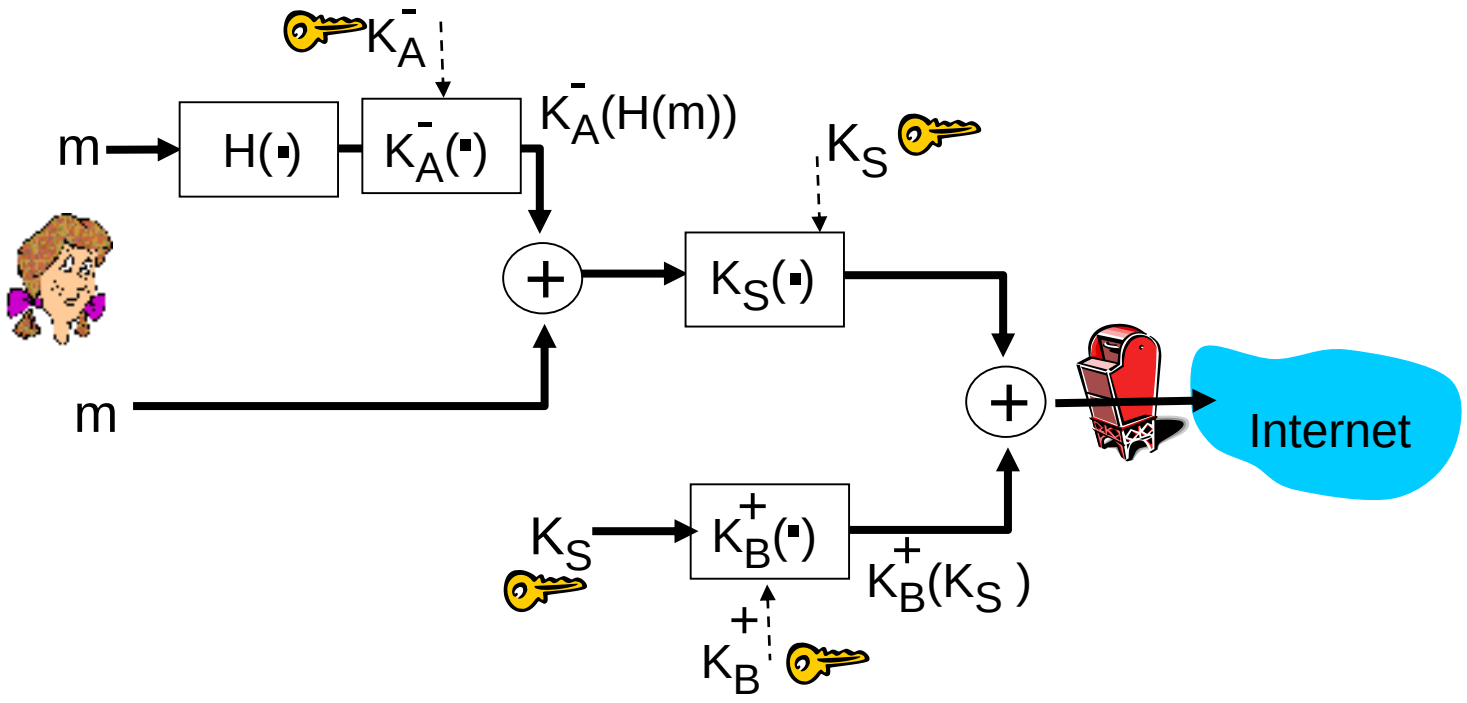
normal application



application with SSL

- SSL provides application programming interface (API) to applications
- C and Java SSL libraries/classes readily available

Could do something like PGP:

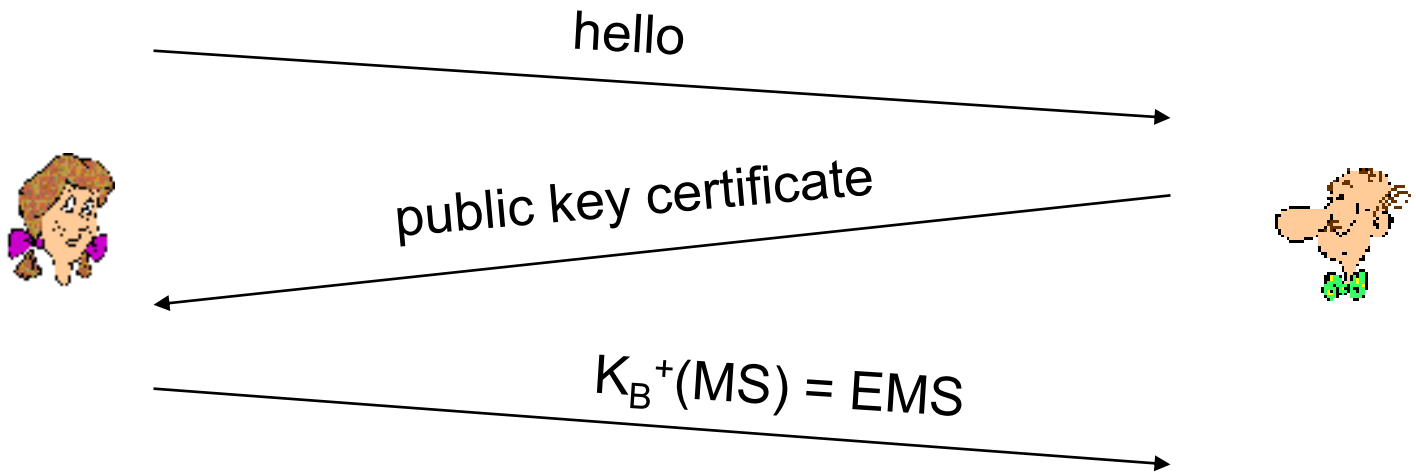


- but want to send byte streams & interactive data
- want set of secret keys for entire connection
- want certificate exchange as part of protocol: handshake phase

Toy SSL: a simple secure channel

- *handshake:* Alice and Bob use their certificates, private keys to authenticate each other and exchange shared secret
- *key derivation:* Alice and Bob use shared secret to derive set of keys
- *data transfer:* data to be transferred is broken up into series of records
- *connection closure:* special messages to securely close connection

Toy: a simple handshake



MS: master secret

EMS: encrypted master secret

Toy: key derivation

- considered bad to use same key for more than one cryptographic operation
  - use different keys for message authentication code (MAC) and encryption
- four keys:
  - K_c = encryption key for data sent from client to server
  - M_c = MAC key for data sent from client to server
  - K_s = encryption key for data sent from server to client
  - M_s = MAC key for data sent from server to client
- keys derived from key derivation function (KDF)
  - takes master secret and (possibly) some additional random data and creates the keys

Toy: data records

- why not encrypt data in constant stream as we write it to TCP?
  - where would we put the MAC? If at end, no message integrity until all data processed.
  - e.g., with instant messaging, how can we do integrity check over all bytes sent before displaying?
- instead, break stream in series of records
  - each record carries a MAC
  - receiver can act on each record as it arrives
- issue: in record, receiver needs to distinguish MAC from data
  - want to use variable-length records

Toy: sequence numbers

- *problem:* attacker can capture and replay record or re-order records
- *solution:* put sequence number into MAC:
  - MAC = MAC(M_x, sequence||data)
  - note: no sequence number field

- *problem:* attacker could replay all records
- *solution:* use nonce

Toy: control information

- *problem:* truncation attack:
  - attacker forges TCP connection close segment
  - one or both sides thinks there is less data than there actually is.
- *solution:* record types, with one type for closure
  - type 0 for data; type 1 for closure
- MAC = MAC(M_x, sequence||type||data)

Toy SSL: summary

bob.com

hello

certificate, nonce

$K_B^+(MS) = EMS$

type 0, seq 1, data

type 0, seq 2, data

type 0, seq 1, data

type 0, seq 3, data

type 1, seq 4, close

type 1, seq 2, close

encrypted

Toy SSL isn't complete

- how long are fields?
- which encryption protocols?
- want negotiation?
  - allow client and server to support different encryption algorithms
  - allow client and server to choose together specific algorithm before data transfer

SSL cipher suite

- cipher suite
  - public-key algorithm
  - symmetric encryption algorithm
  - MAC algorithm
- SSL supports several cipher suites
- negotiation: client, server agree on cipher suite
  - client offers choice
  - server picks one

common SSL symmetric ciphers

- DES – Data Encryption Standard: block
- 3DES – Triple strength: block
- RC2 – Rivest Cipher 2: block
- RC4 – Rivest Cipher 4: stream

SSL Public key encryption

- RSA

Real SSL: handshake (1)

Purpose

1. server authentication
2. negotiation: agree on crypto algorithms
3. establish keys
4. client authentication (optional)

Real SSL: handshake (2)

1. client sends list of algorithms it supports, along with client nonce
2. server chooses algorithms from list; sends back: choice + certificate + server nonce
3. client verifies certificate, extracts server's public key, generates pre_master_secret, encrypts with server's public key, sends to server
4. client and server independently compute encryption and MAC keys from pre_master_secret and nonces
5. client sends a MAC of all the handshake messages
6. server sends a MAC of all the handshake messages

Real SSL: handshaking (3)

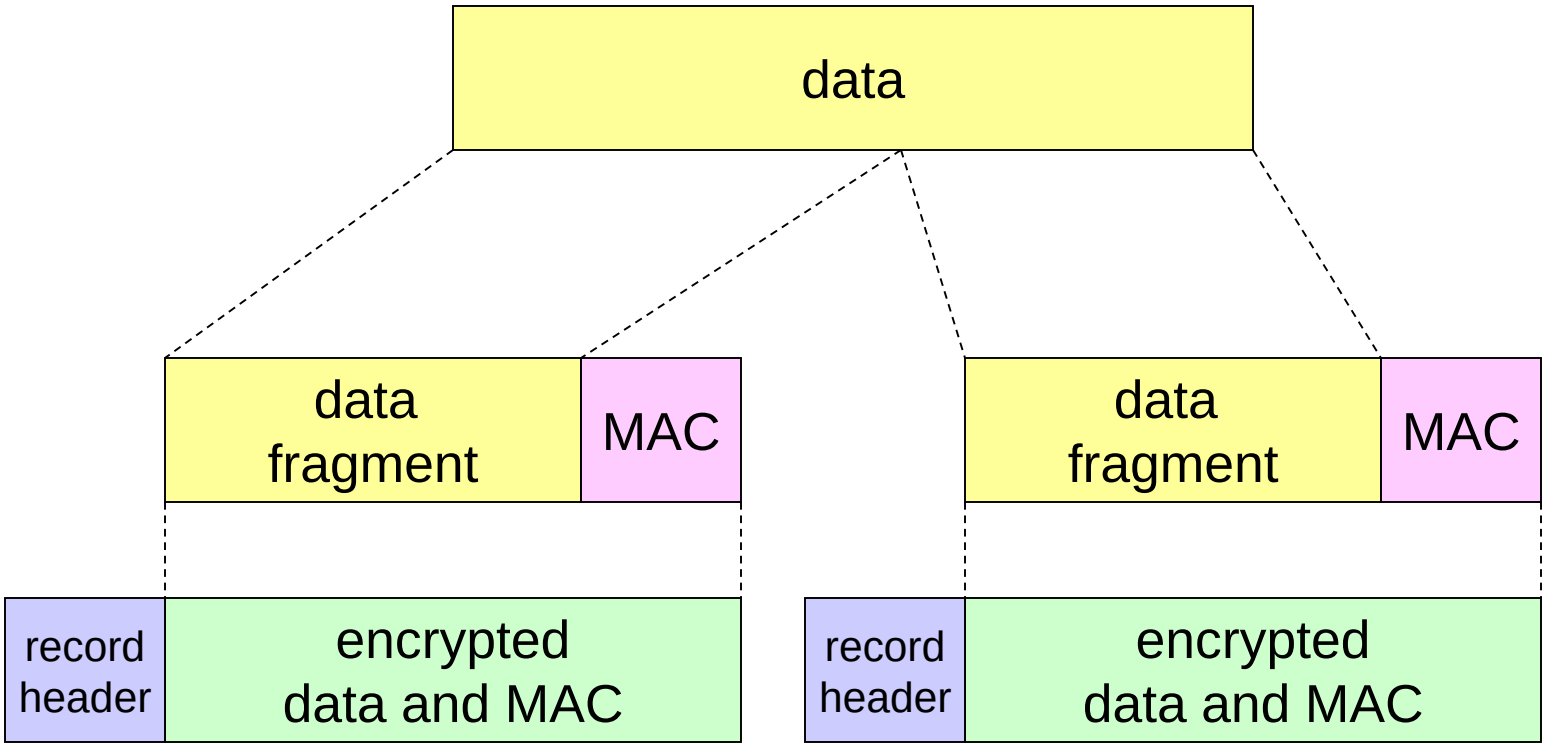
last 2 steps protect handshake from tampering

- client typically offers range of algorithms, some strong, some weak
- man-in-the middle could delete stronger algorithms from list
- last 2 steps prevent this
  - last two messages are encrypted

Real SSL: handshaking (4)

- why two random nonces?
- suppose Trudy sniffs all messages between Alice & Bob
- next day, Trudy sets up TCP connection with Bob, sends exact same sequence of records
  - Bob (Amazon) thinks Alice made two separate orders for the same thing
  - solution: Bob sends different random nonce for each connection. This causes encryption keys to be different on the two days
  - Trudy's messages will fail Bob's integrity check

SSL record protocol

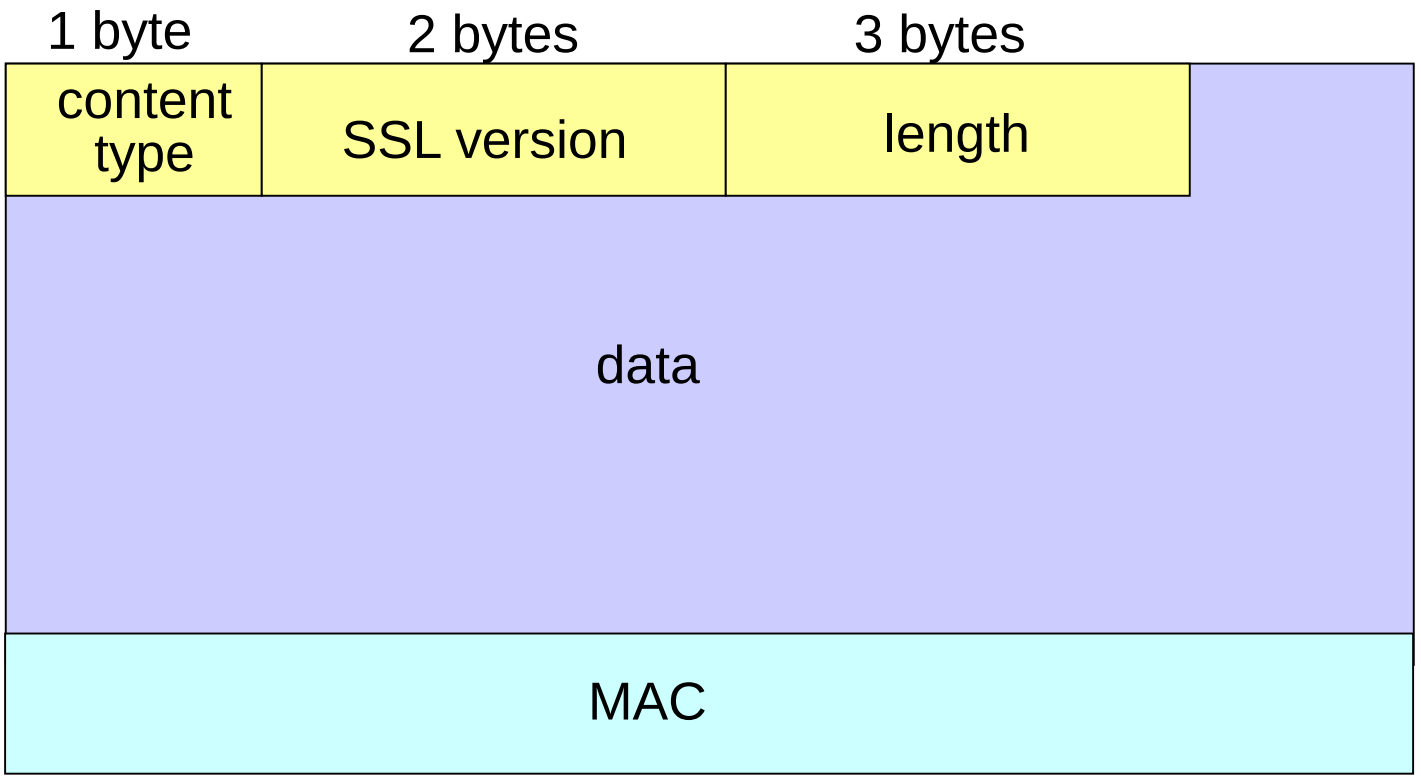


record header: content type; version; length

MAC: includes sequence number, MAC key M_x

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

SSL record format



data and MAC encrypted (symmetric algorithm)

Real SSL connection

handshake: ClientHello

handshake: ServerHello

handshake: Certificate

handshake: ServerHelloDone

handshake: ClientKeyExchange

ChangeCipherSpec

handshake: Finished

everything henceforth is encrypted

ChangeCipherSpec

handshake: Finished

application_data

application_data

Alert: warning, close_notify

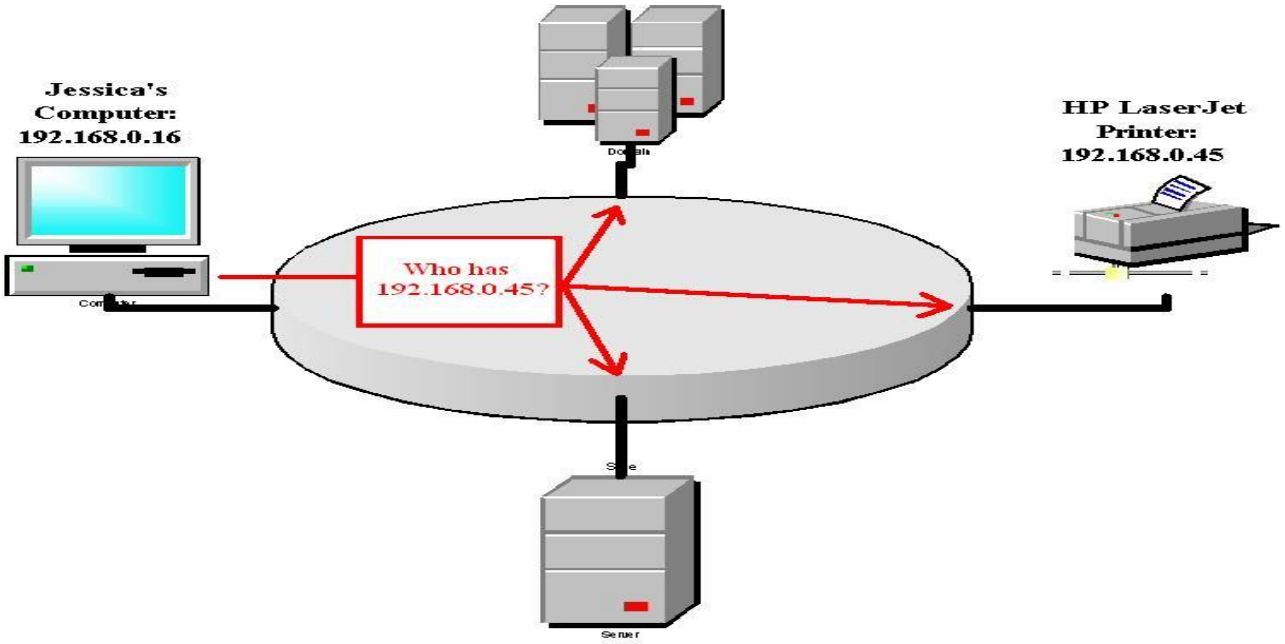
TCP FIN follows

Key derivation

- client nonce, server nonce, and pre-master secret input into pseudo random-number generator.
  - produces master secret
- master secret and new nonces input into another random-number generator: "key block"
  - because of resumption: TBD
- 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)

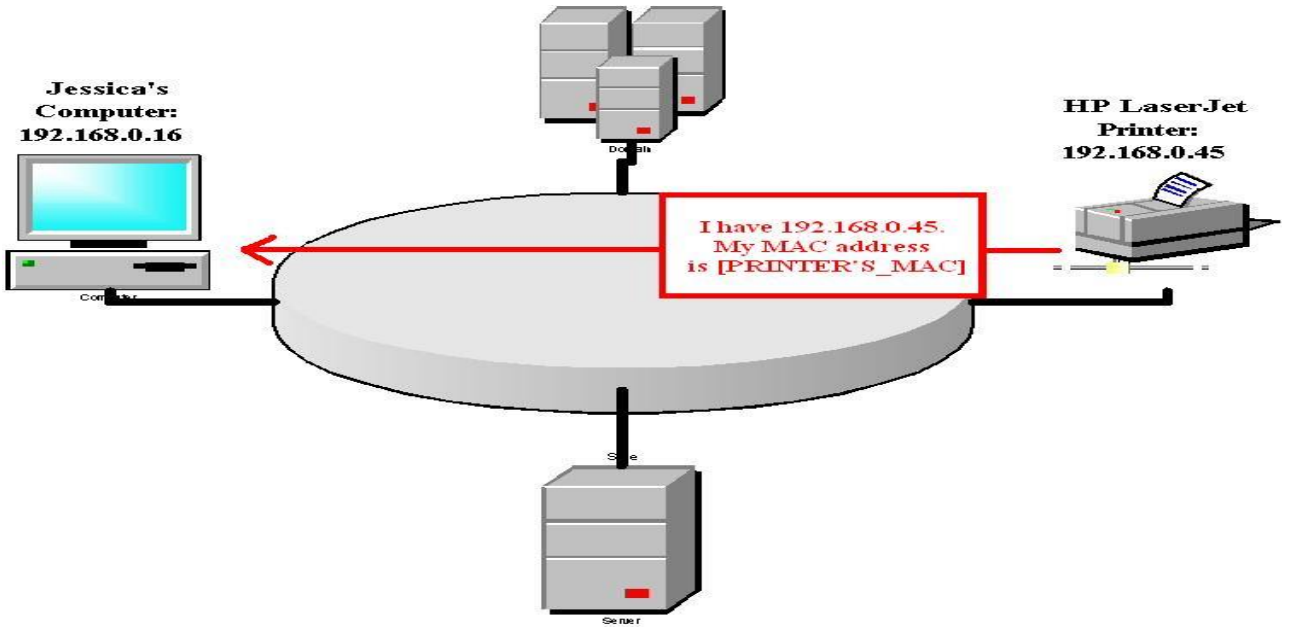
ARP

- ARP request
  - Computer A asks the network, "Who has this IP address?“



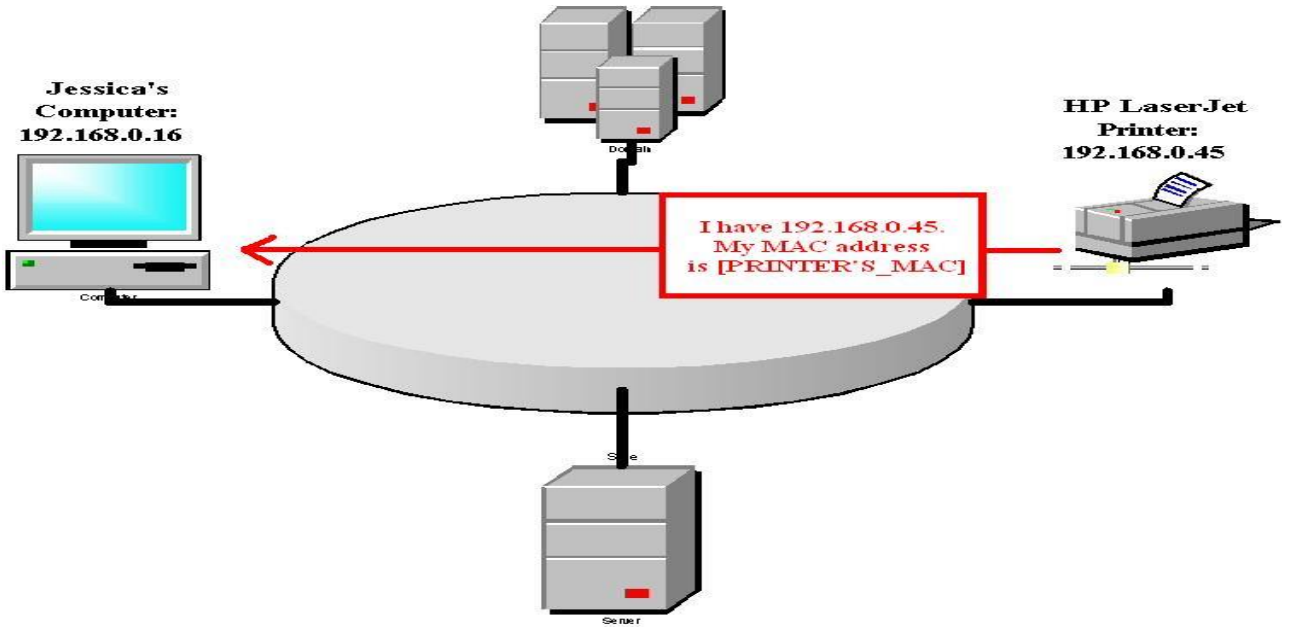
ARP

- ARP reply
  - Computer B tells Computer A, "I have that IP. My Physical Address is [whatever it is].“



ARP

- ARP reply
  - Computer B tells Computer A, "I have that IP. My Physical Address is [whatever it is]."



Cache table

- A short-term memory of all the IP addresses and Physical addresses
- Ensures that the device doesn't have to repeat ARP Requests for devices it has already communicated with
- Implemented as an array of entries
- Entries are updated

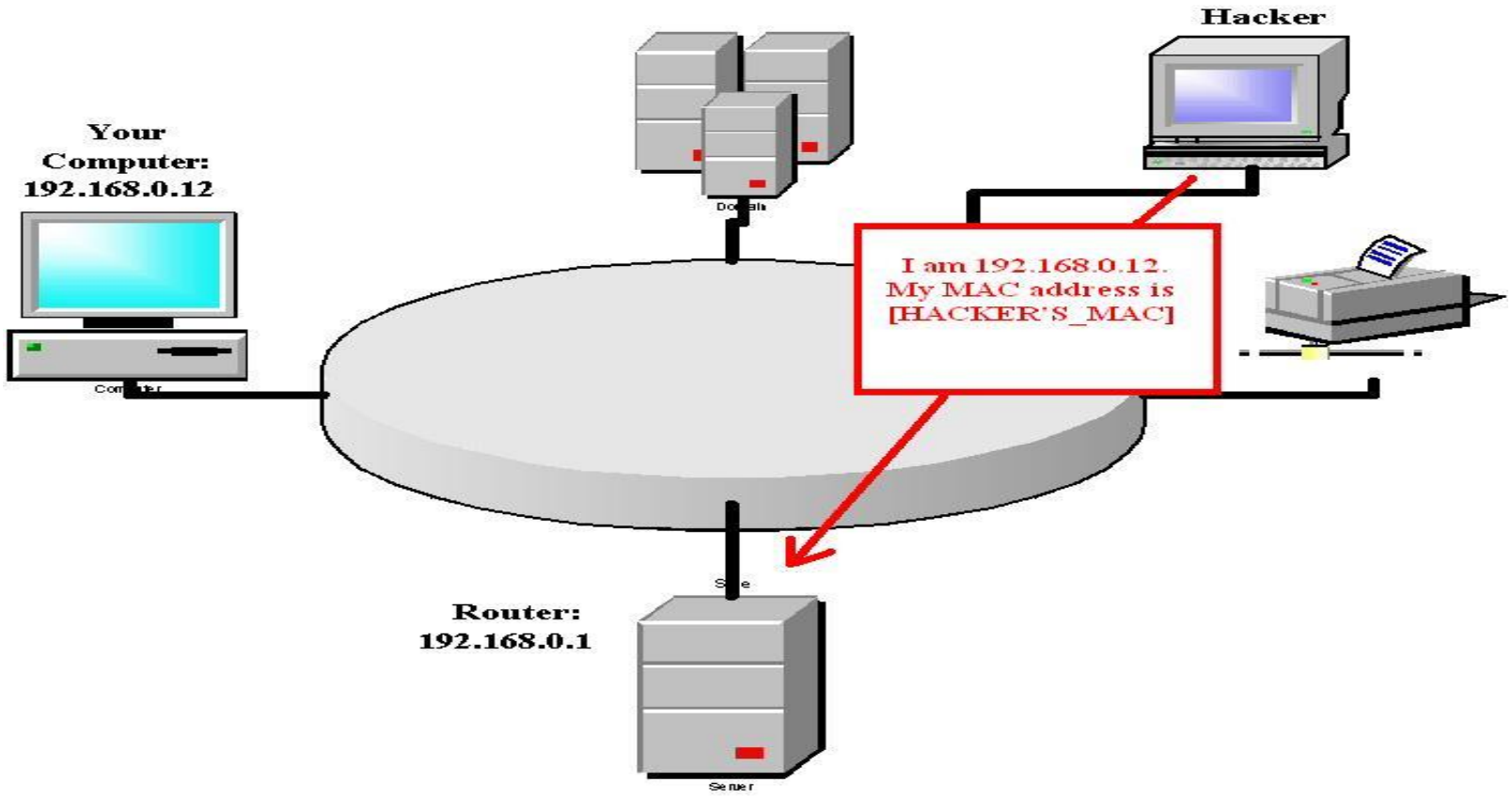
Cache table

State	Queue	Attempt	Time-out	IP Address	Physical Address
R	5		900	180.3.6.1	ACAE32457342
P	2	2		129.34.4.8	
P	14	5		201.11.56.7	
R	8		450	114.5.7.89	457342ACAE32
P	12	1		220.55.5.7	
F					
R	9		60	19.1.7.82	4573E3242ACA
P	18	3		188.11.8.71	

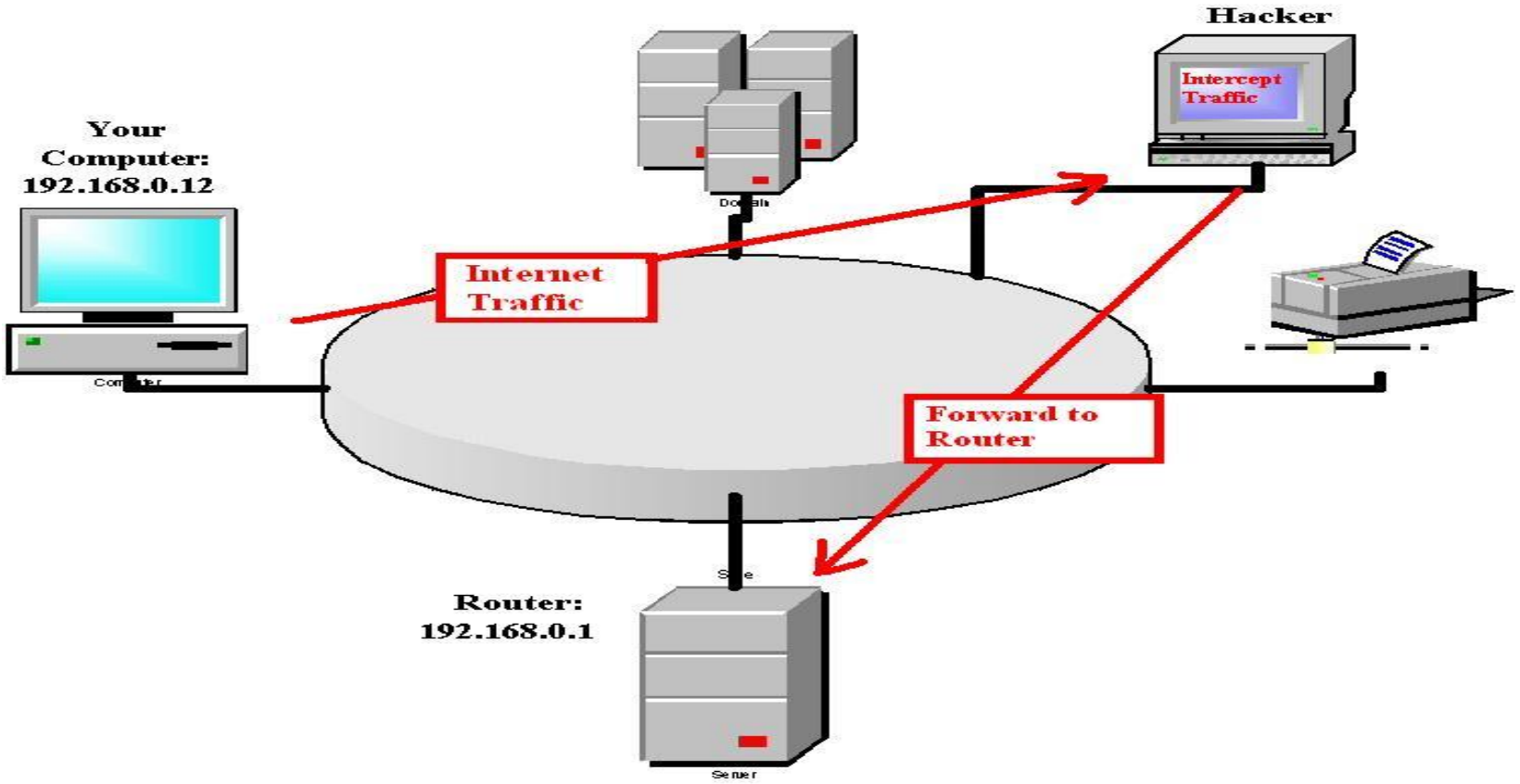
ARP spoofing

- Simplicity also leads to major insecurity
  - No Authentication
    - ARP provides no way to verify that the responding device is really who it says it is
    - Stateless protocol
      - Updating ARP Cache table
- Attacks
  - DOS
    - Hacker can easily associate an operationally significant IP address to a false MAC address
  - Man-in-the-Middle
    - Intercept network traffic between two devices in your network

ARP spoofing (MITM)



ARP spoofing (MITM)



Prevent ARP spoofing

- For Small Network
  - Static Arp Cache table
- For Large Network
  - Arpwatch
- As an administrator, check for multiple Physical addresses responding to a given IP address

Chapter 8 roadmap

8.1 What is network security?

8.2 Principles of cryptography

8.3 Message integrity

8.4 Securing e-mail

8.5 Securing TCP connections: SSL

8.6 Network layer security: IPsec

8.7 Securing wireless LANs

8.8 Operational security: firewalls and IDS

What is network-layer confidentiality ?

between two network entities:

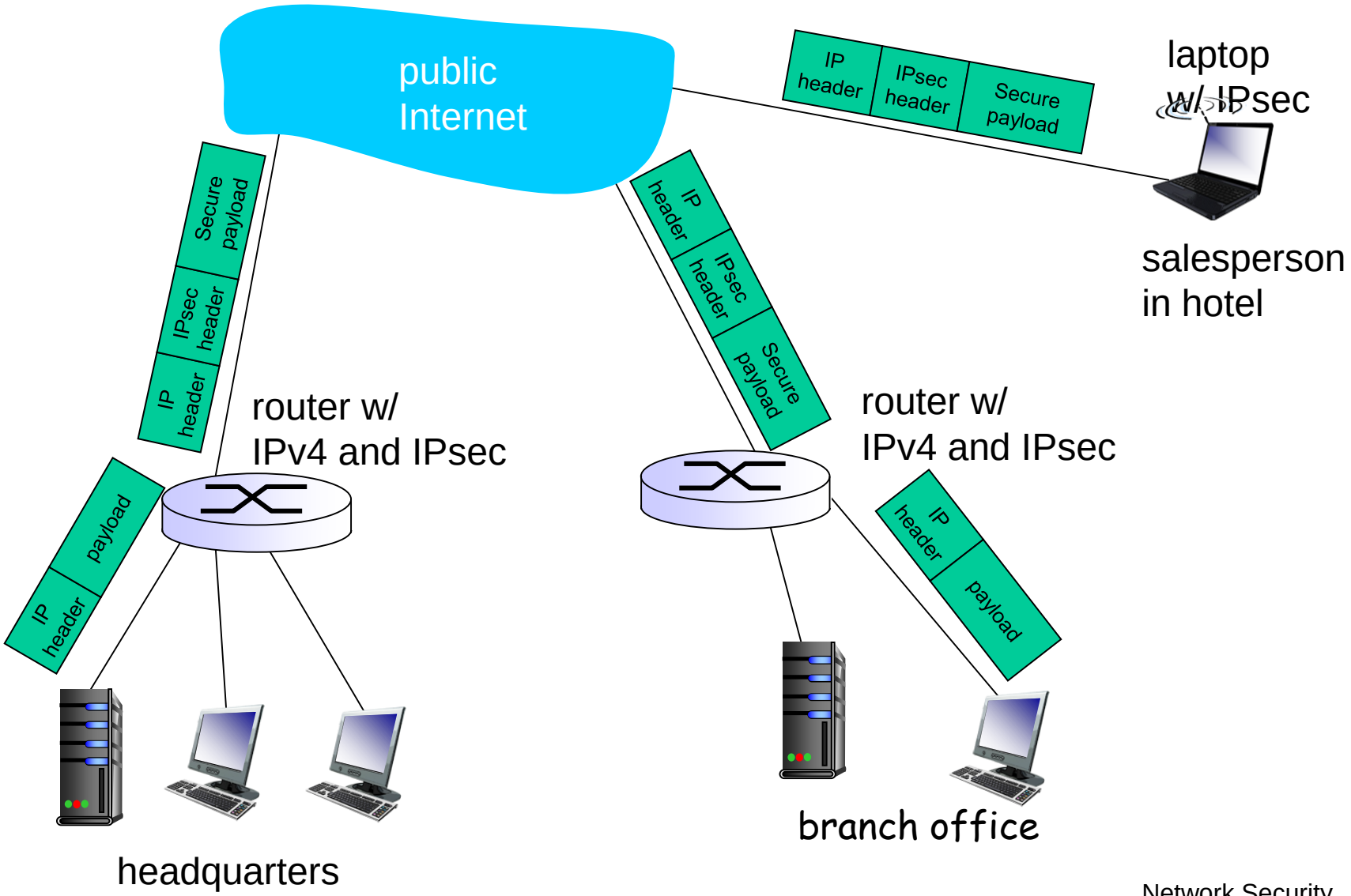
- sending entity encrypts datagram payload, payload could be:
  - TCP or UDP segment, ICMP message, OSPF message ….
- all data sent from one entity to other would be hidden:
  - web pages, e-mail, P2P file transfers, TCP SYN packets …
- "blanket coverage"

Virtual Private Networks (VPNs)

motivation:

- institutions often want private networks for security.
  - costly: separate routers, links, DNS infrastructure.
- VPN: institution's inter-office traffic is sent over public Internet instead
  - encrypted before entering public Internet
  - logically separate from other traffic

Virtual Private Networks (VPNs)

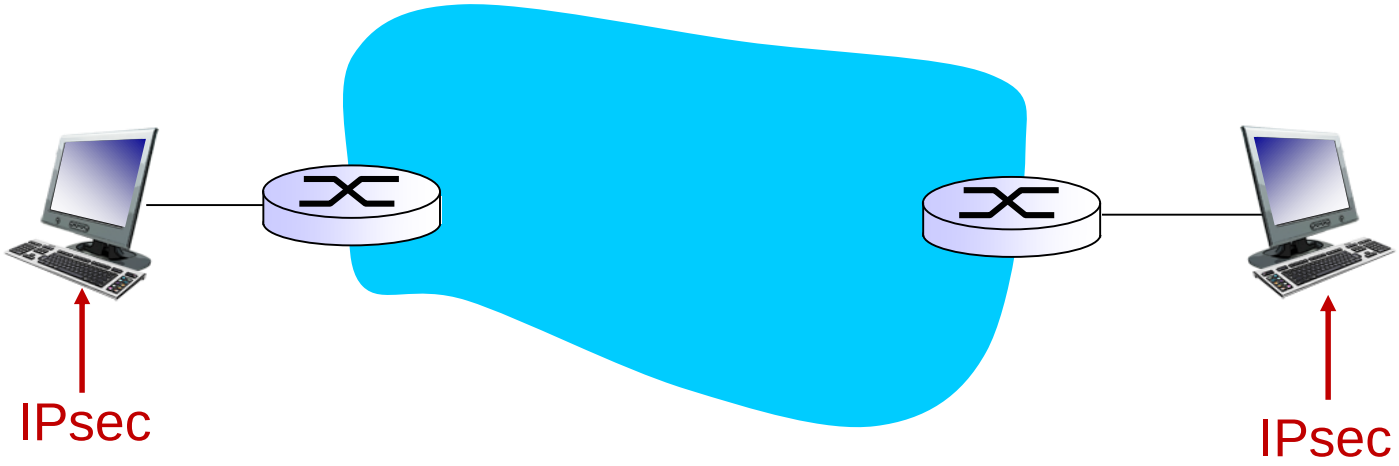


IPsec services

- data integrity
- origin authentication
- replay attack prevention
- confidentiality

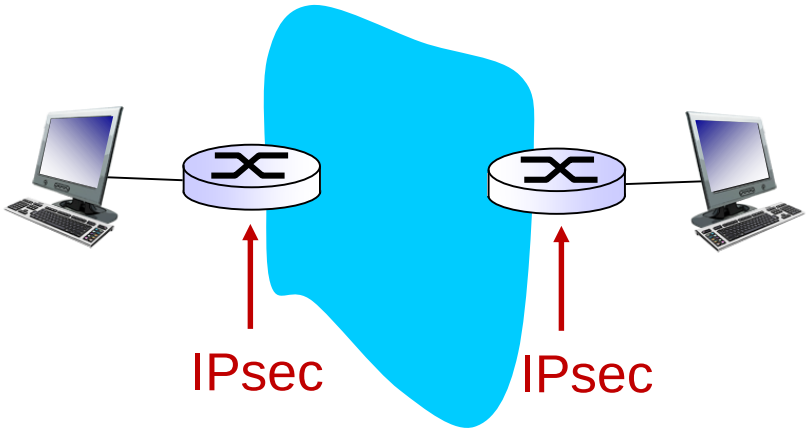
- two protocols providing different service models:
  - AH
  - ESP

IPsec transport mode

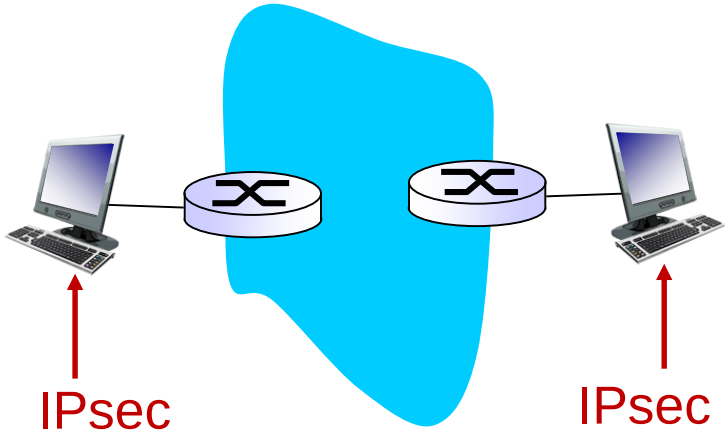


- IPsec datagram emitted and received by end-system
- protects upper level protocols

IPsec – tunneling mode



- edge routers IPsec-aware



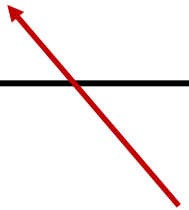
- hosts IPsec-aware

Two IPsec protocols

- Authentication Header (AH) protocol
  - provides source authentication & data integrity but *not* confidentiality
- Encapsulation Security Protocol (ESP)
  - provides source authentication, data integrity, *and confidentiality*
  - more widely used than AH

Four combinations are possible!

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

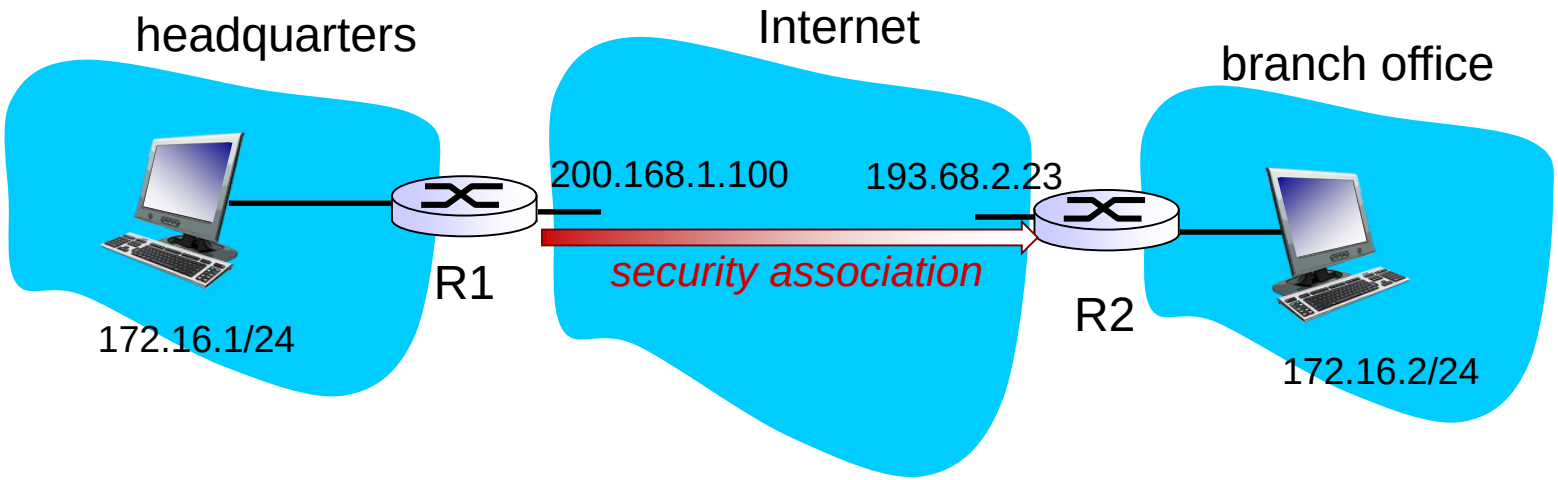


most common and most important

Security associations (SAs)

- before sending data, "security association (SA)" established from sending to receiving entity
  - SAs are simplex: for only one direction
- ending, receiving entitles maintain *state information* about SA
  - recall: TCP endpoints also maintain state info
  - IP is connectionless; IPsec is connection-oriented!
- how many SAs in VPN w/ headquarters, branch office, and n traveling salespeople?

Example SA from R1 to R2



R1 stores for SA:

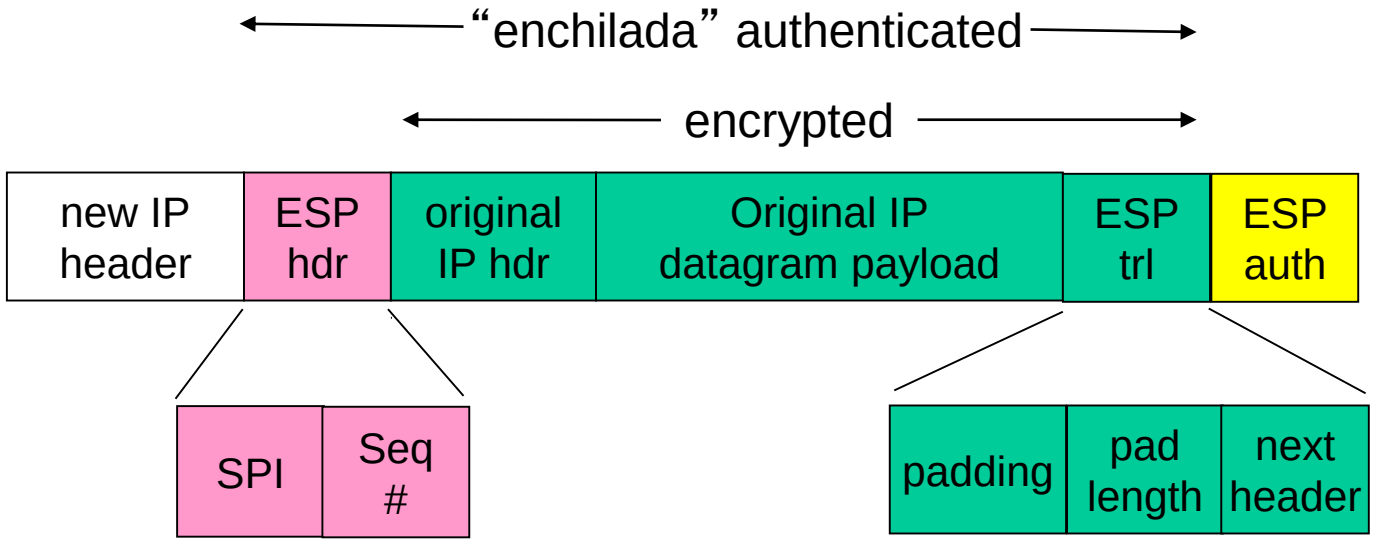
- 32-bit SA identifier: *Security Parameter Index (SPI)*
- origin SA interface (200.168.1.100)
- destination SA interface (193.68.2.23)
- type of encryption used (e.g., 3DES with CBC)
- encryption key
- type of integrity check used (e.g., HMAC with MD5)
- authentication key

Security Association Database (SAD)

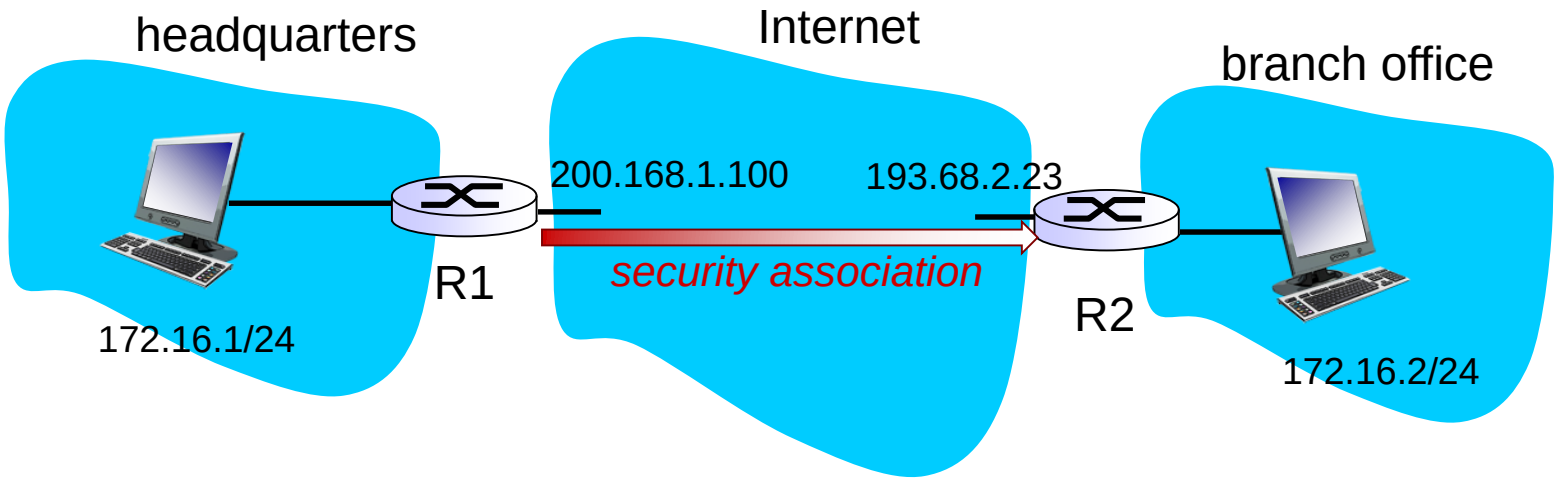
- endpoint holds SA state in *security association database (SAD)*, where it can locate them during processing.
- with n salespersons, 2 + 2n SAs in R1's SAD
- when sending IPsec datagram, R1 accesses SAD to determine how to process datagram.
- when IPsec datagram arrives to R2, R2 examines SPI in IPsec datagram, indexes SAD with SPI, and processes datagram accordingly.

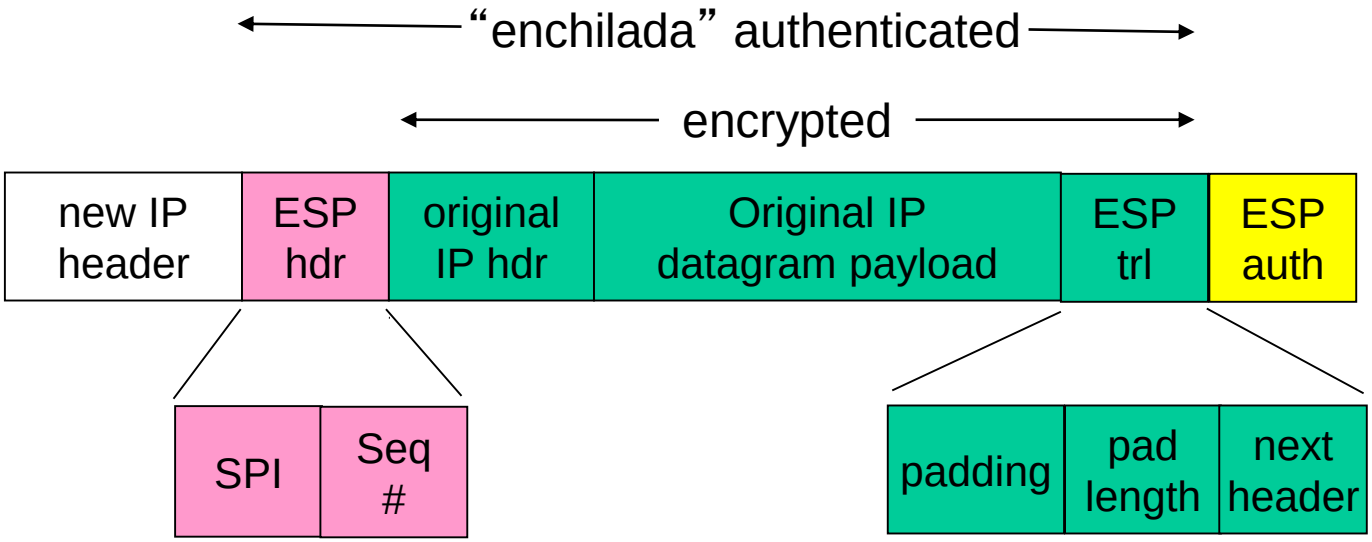
IPsec datagram

focus for now on tunnel mode with ESP



What happens?

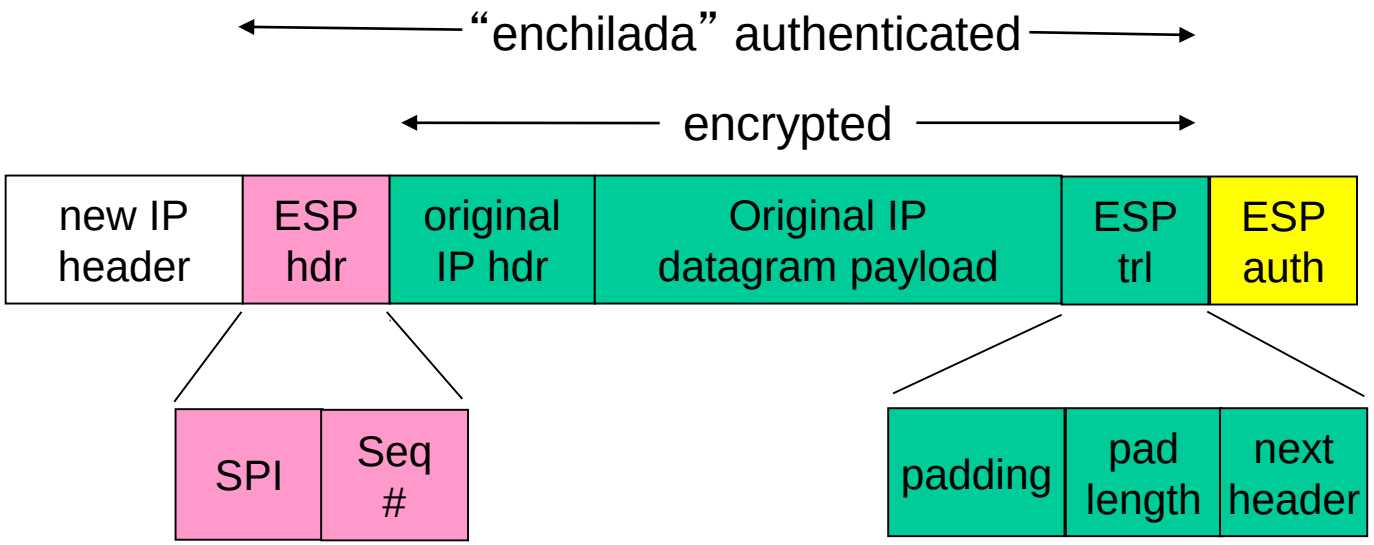




R1: convert original datagram to IPsec datagram

- appends to back of original datagram (which includes original header fields!) an "ESP trailer" field.
- encrypts result using algorithm & key specified by SA.
- appends to front of this encrypted quantity the "ESP header, creating "enchilada".
- creates authentication MAC over the *whole enchilada*, using algorithm and key specified in SA;
- appends MAC to back of enchilada, forming *payload*;
- creates brand new IP header, with all the classic IPv4 header fields, which it appends before payload.

Inside the enchilada:



- ESP trailer: Padding for block ciphers
- ESP header:
  - SPI, so receiving entity knows what to do
  - Sequence number, to thwart replay attacks
- MAC in ESP auth field is created with shared secret key

IPsec sequence numbers

- for new SA, sender initializes seq. # to 0
- each time datagram is sent on SA:
  - sender increments seq # counter
  - places value in seq # field
- goal:
  - prevent attacker from sniffing and replaying a packet
  - receipt of duplicate, authenticated IP packets may disrupt service
- method:
  - destination checks for duplicates
  - doesn't keep track of *all* received packets; instead uses a window

Security Policy Database (SPD)

- policy: For a given datagram, sending entity needs to know if it should use IPsec
- needs also to know which SA to use
  - may use: source and destination IP address; protocol number
- info in SPD indicates "what" to do with arriving datagram
- info in SAD indicates "how" to do it

Summary: IPsec services

- suppose Trudy sits somewhere between R1 and R2. she doesn't know the keys.
  - will Trudy be able to see original contents of datagram? How about source, dest IP address, transport protocol, application port?
  - flip bits without detection?
  - masquerade as R1 using R1's IP address?
  - replay a datagram?

IKE: Internet Key Exchange

- *previous examples:* manual establishment of IPsec SAs in IPsec endpoints:

  Example SA

  SPI: 12345
  Source IP: 200.168.1.100
  Dest IP: 193.68.2.23
  Protocol: ESP
  Encryption algorithm: 3DES-cbc
  HMAC algorithm: MD5
  Encryption key: 0x7aeaca…
  HMAC key:0xc0291f…

- manual keying is impractical for VPN with 100s of endpoints
- instead use *IPsec IKE (Internet Key Exchange)*

IKE: PSK and PKI

- authentication (prove who you are) with either
  - pre-shared secret (PSK) or
  - with PKI (pubic/private keys and certificates).
- PSK: both sides start with secret
  - run IKE to authenticate each other and to generate IPsec SAs (one in each direction), including encryption, authentication keys
- PKI: both sides start with public/private key pair, certificate
  - run IKE to authenticate each other, obtain IPsec SAs (one in each direction).
  - similar with handshake in SSL.

IKE phases

- IKE has two phases
  - *phase 1:* establish bi-directional IKE SA
    - note: IKE SA different from IPsec SA
    - aka ISAKMP security association
  - *phase 2:* ISAKMP is used to securely negotiate IPsec pair of SAs
- phase 1 has two modes: aggressive mode and main mode
  - aggressive mode uses fewer messages
  - main mode provides identity protection and is more flexible

IPsec summary

- IKE message exchange for algorithms, secret keys, SPI numbers
- either AH or ESP protocol (or both)
  - AH provides integrity, source authentication
  - ESP protocol (with AH) additionally provides encryption
- IPsec peers can be two end systems, two routers/firewalls, or a router/firewall and an end system