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

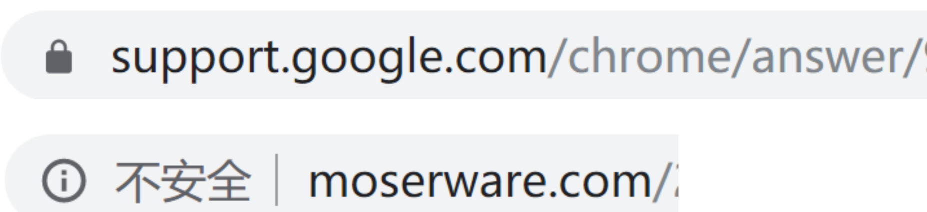
- Widely deployed security protocol
  - Supported by almost all browsers and web servers
  - https
  - Tens of billions $ spent per year over SSL
- Originally designed by Netscape in 1993
- Number of variations:
  - TLS: transport layer security, RFC 2246
- Provides
  - Confidentiality
  - Integrity
  - Authentication
- Original goals:
  - Had Web e-commerce transactions in mind
  - 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/TLS

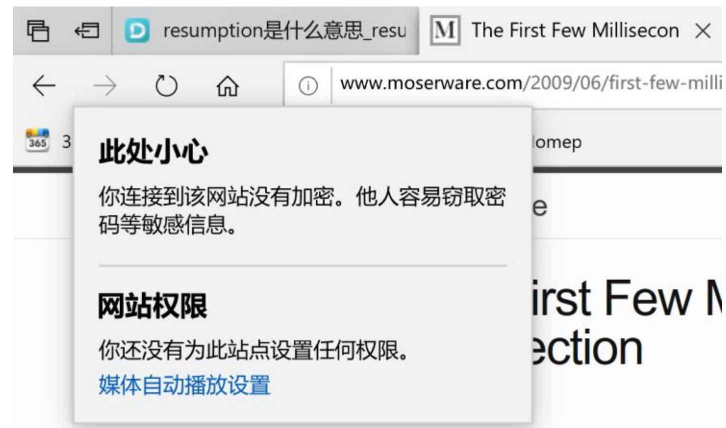
SSL/TLS

Firefox:

Chrome:



MS Edge:



SSL/TLS

1994年，NetScape公司设计了SSL协议（Secure Sockets Layer）的1.0版，但是未发布。

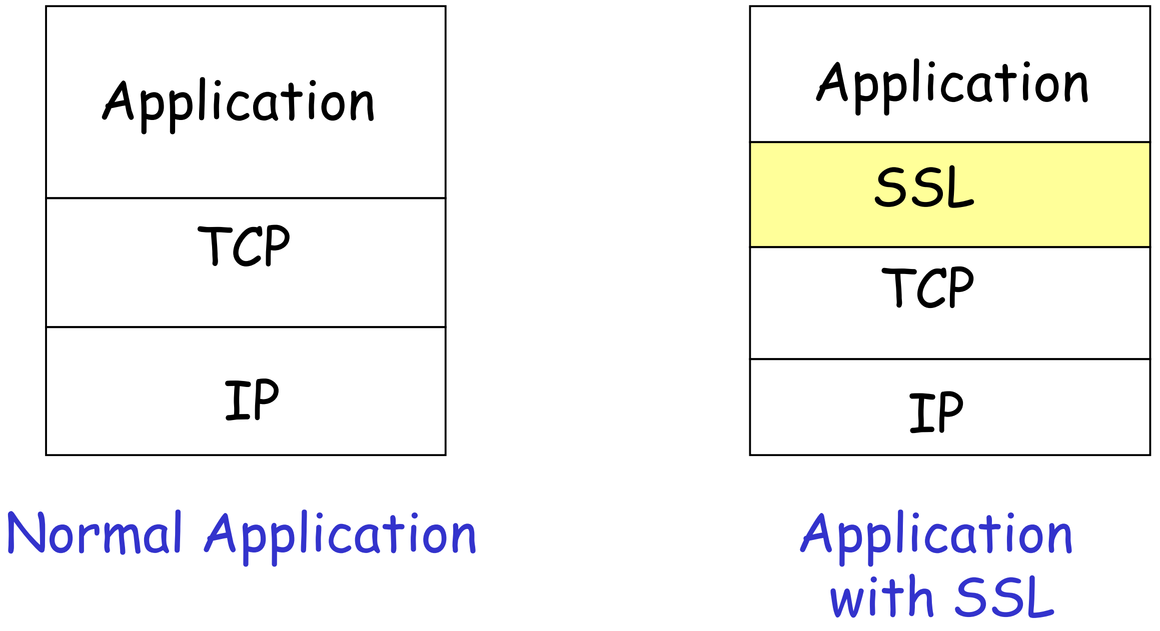
1995年，NetScape公司发布SSL 2.0版，很快发现有严重漏洞。

1996年，SSL 3.0版问世，得到大规模应用。

1999年，互联网标准化组织ISOC接替NetScape公司，发布了SSL的升级版TLS 1.0版。

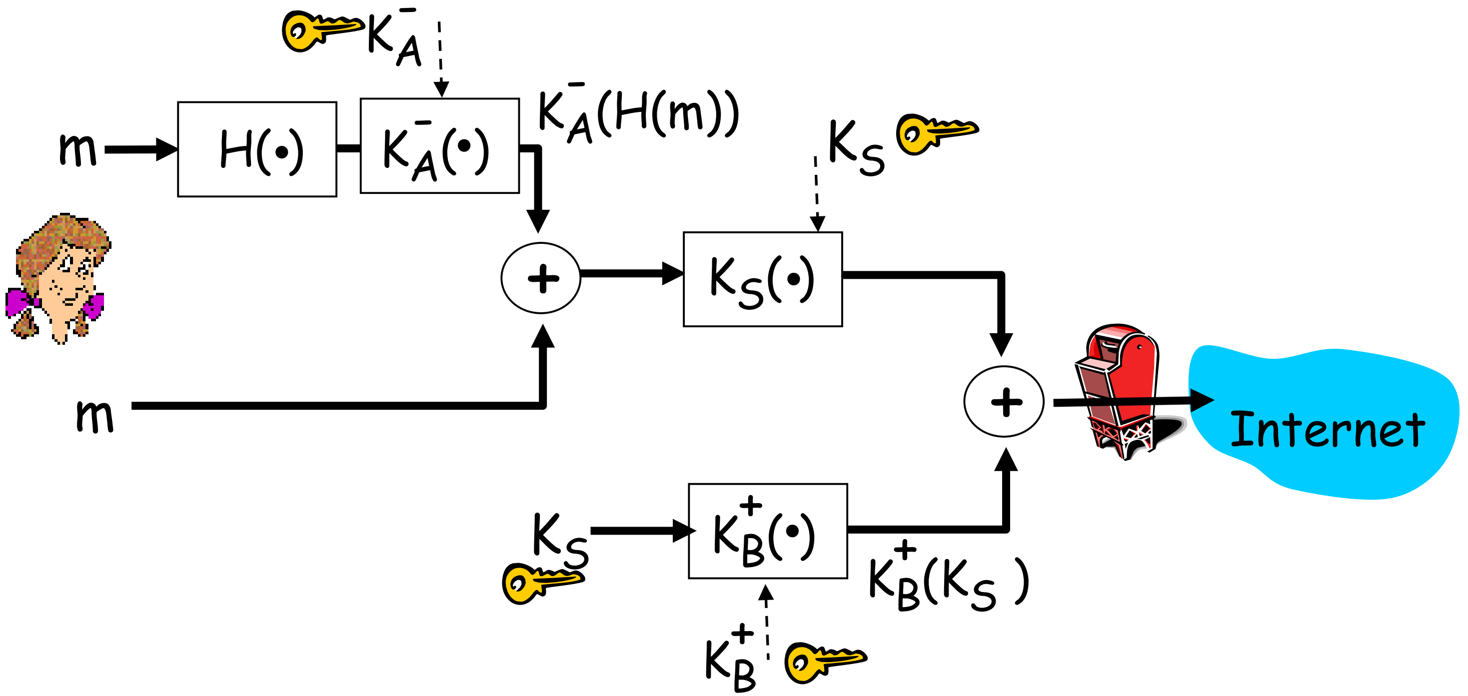
2006年和2008年，TLS进行了两次升级，分别为TLS 1.1版和TLS 1.2版。最新的变动是2011年TLS 1.2的修订版。

SSL and TCP/IP



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

Could do something like PGP:

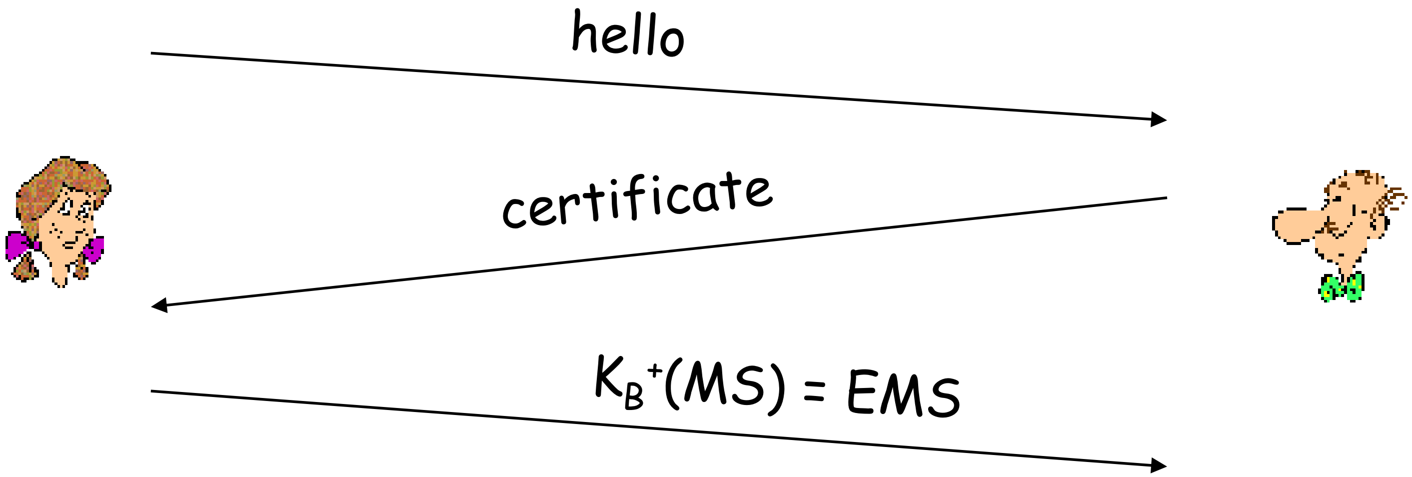


- But want to send byte streams & interactive data
- Want a set of secret keys for the entire connection
- Want certificate exchange part of protocol: handshake phase

Toy SSL: a simple secure channel

- Handshake: Alice and Bob use their certificates and 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 a 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.
  - For example, 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

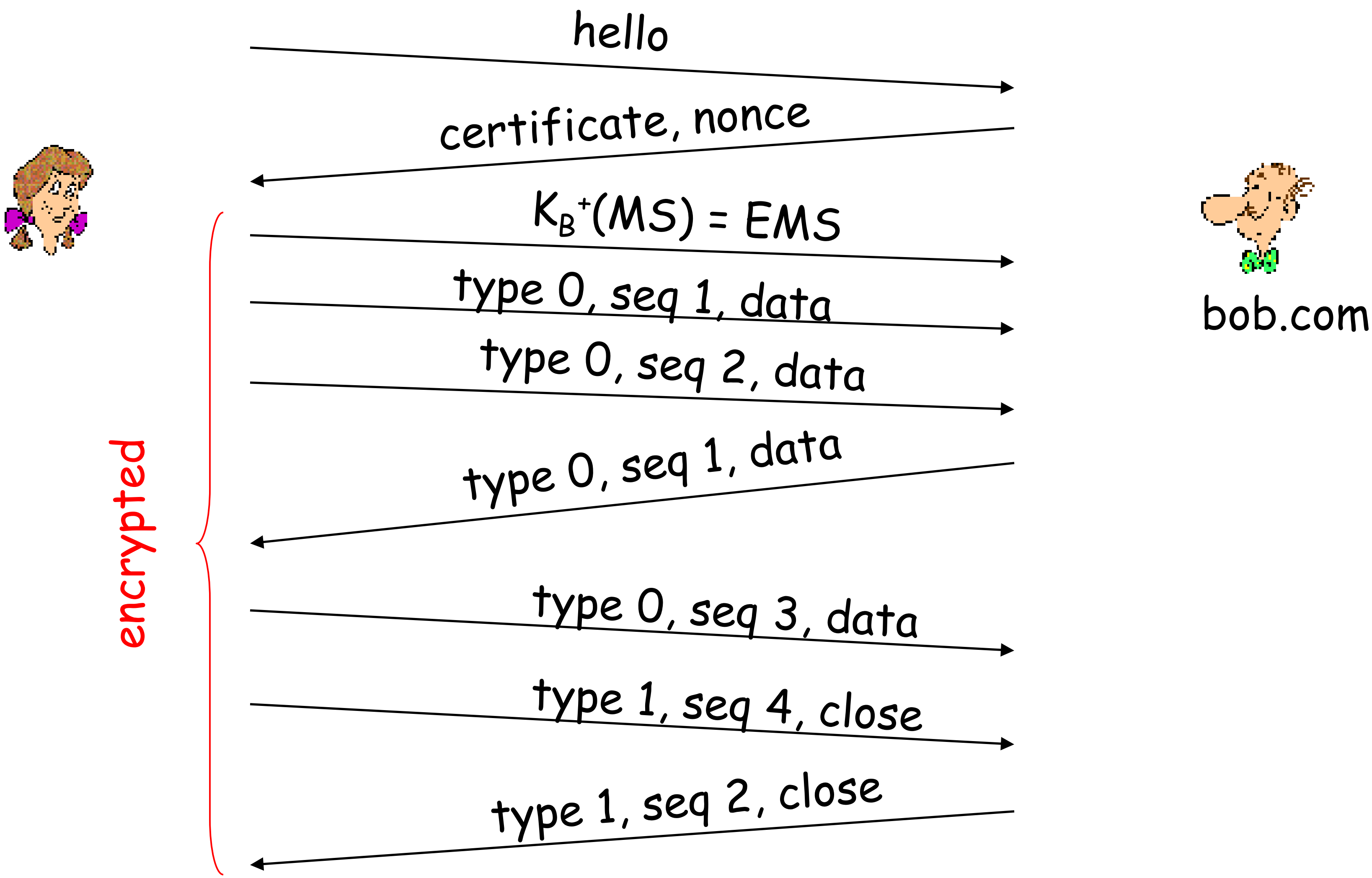
- Attacker can capture and replay record or re-order records
- Solution: put sequence number into MAC:
  - MAC = MAC(M_x, sequence||data)
  - Note: no sequence number field
- Attacker could still replay all of the records
  - Use random nonce

Toy: Control information

- 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)

length	type	data	MAC

Toy SSL: summary



Toy SSL isn't complete

- How long are the fields?
- What encryption protocols?
- No negotiation
  - Allow client and server to support different encryption algorithms
  - Allow client and server to choose together specific algorithm before data transfer

Most common symmetric ciphers in SSL

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

Public key encryption

- RSA

SSL Cipher Suite

- Cipher Suite
  - Public-key algorithm
  - Symmetric encryption algorithm
  - MAC algorithm
- SSL supports a variety of cipher suites
- Negotiation: client and server must agree on cipher suite
- Client offers choice; server picks one

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)

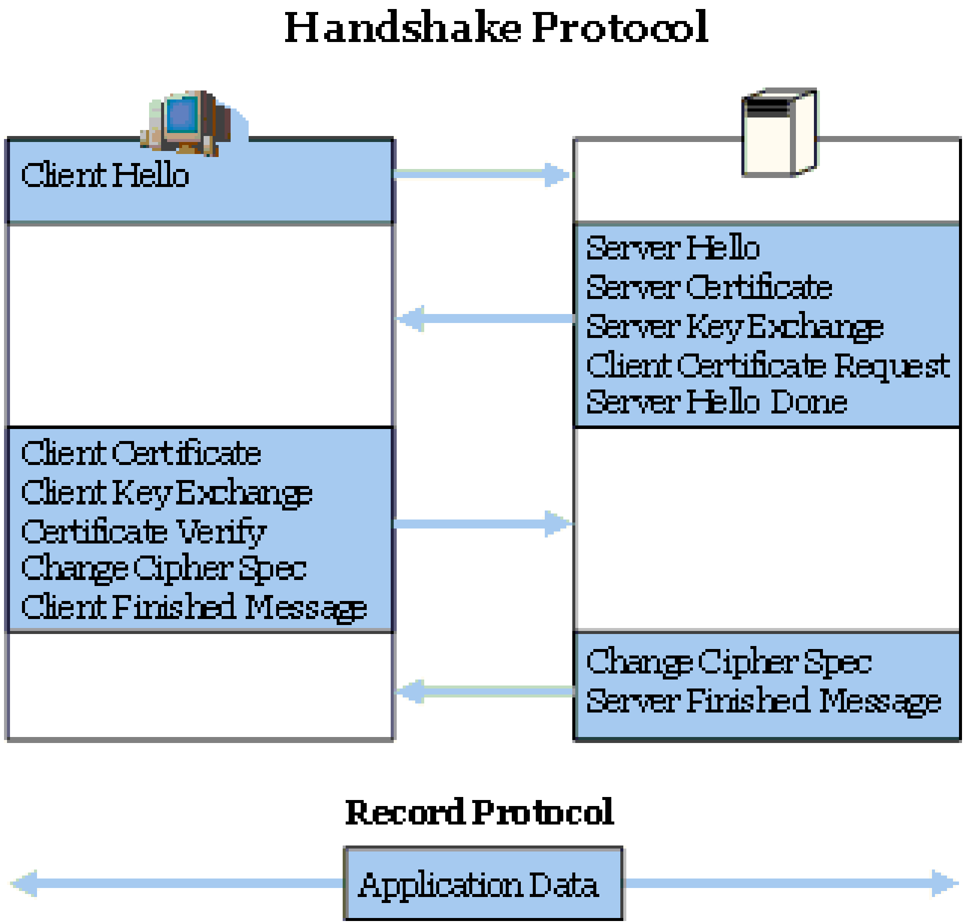
Last 2 steps protect handshake from tampering

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

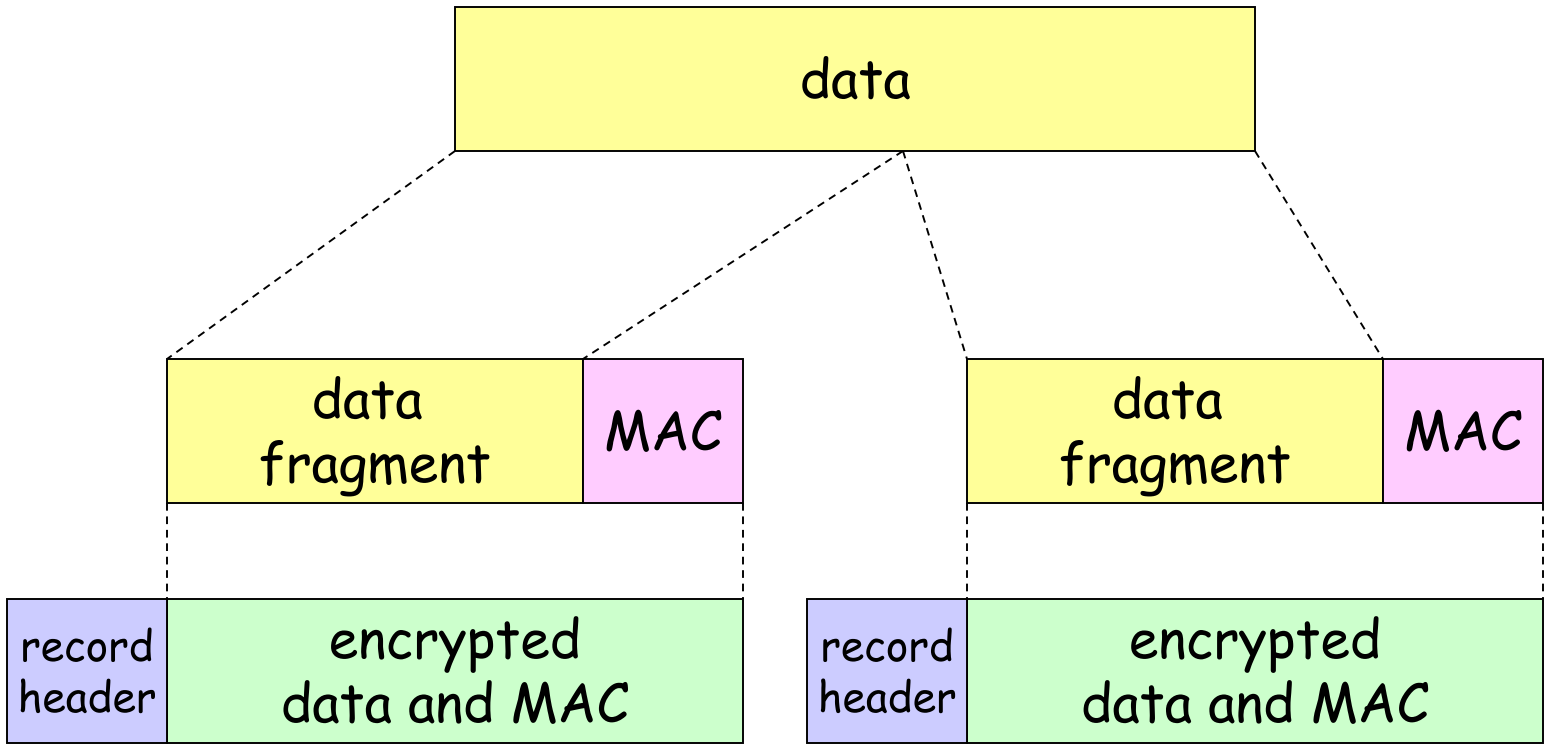
Real SSL: Handshaking (4)

- Why the two random nonces?
- Suppose Trudy sniffs all messages between Alice & Bob.
- Next day, Trudy sets up TCP connection with Bob, sends the 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.

Real SSL: Handshaking (5)



SSL Record Protocol

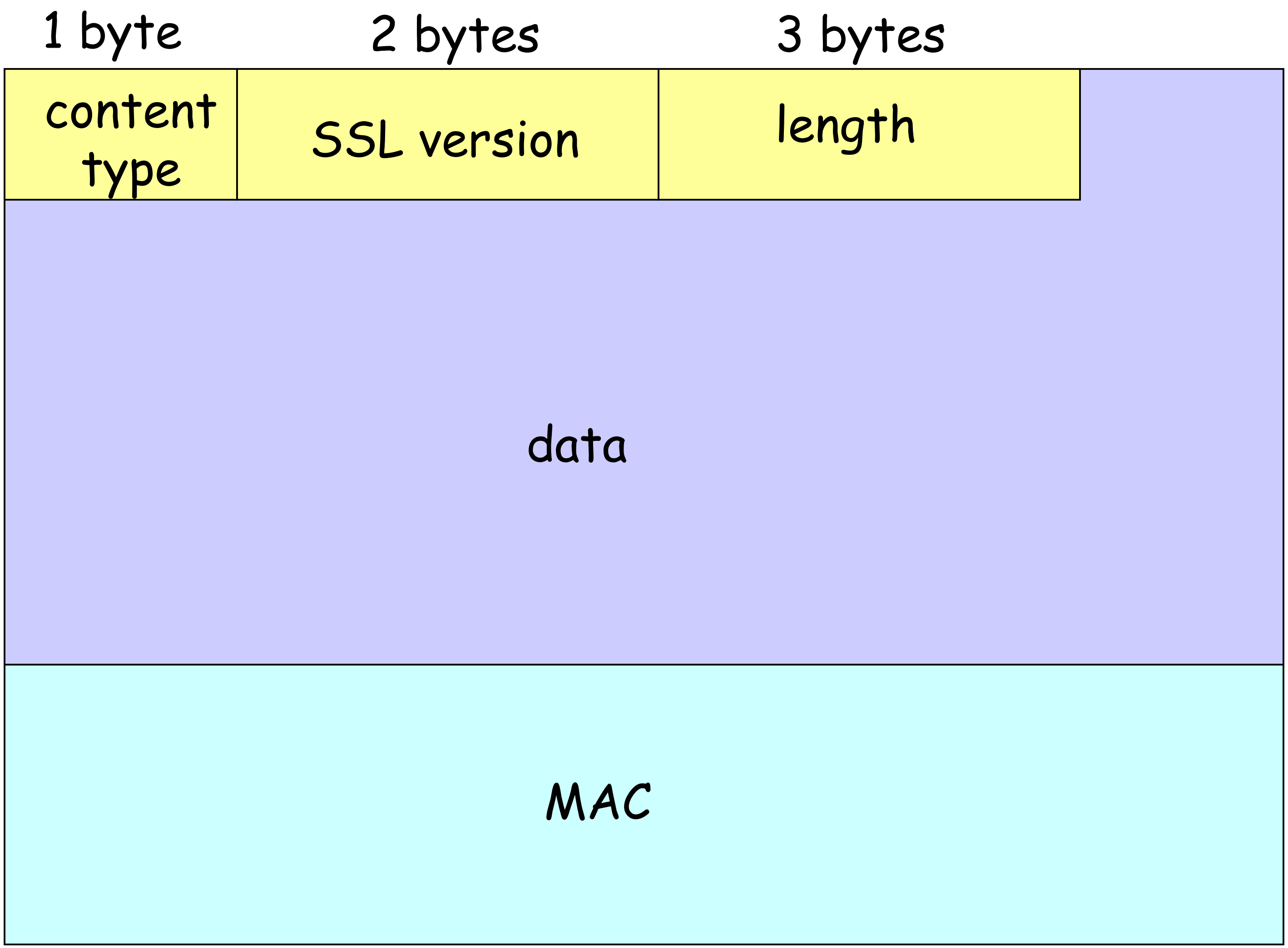


record header: content type; version; length

MAC: includes sequence number, MAC key M_x

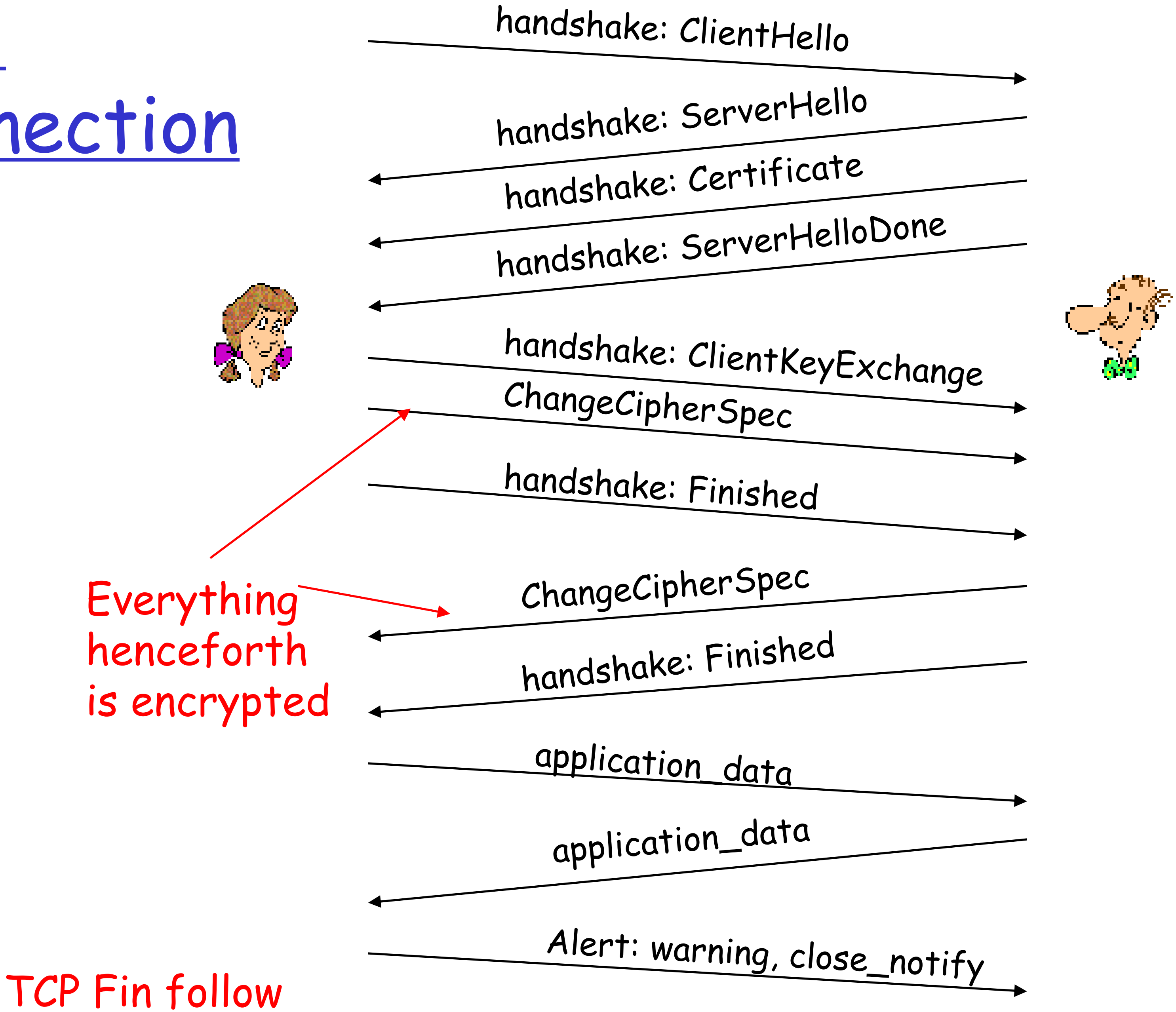
Fragment: each SSL fragment 2^{14} bytes (~16 Kbytes)

SSL Record Format



Data and MAC encrypted (symmetric algo)

Real Connection



Key derivation

- Client nonce, server nonce, and pre-master secret input into pseudo random-number generator.
  - Produces master secret
- Master secret and new nonces inputed 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)

SSL/TLS

- Recommended reading list:
  - MicroSoft TechNet, "SSL/TLS in Detail"
  - Jeff Moser, "The First Few Milliseconds of an HTTPS Connection"

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 confidentiality at the network-layer?

Between two network entities:

- Sending entity encrypts the payloads of datagrams. Payload could be:
    - TCP segment, UDP segment, ICMP message, OSPF message, and so on.
- All data sent from one entity to the other would be hidden:
    - Web pages, e-mail, P2P file transfers, TCP SYN packets, and so on.
- That is, “blanket coverage”.

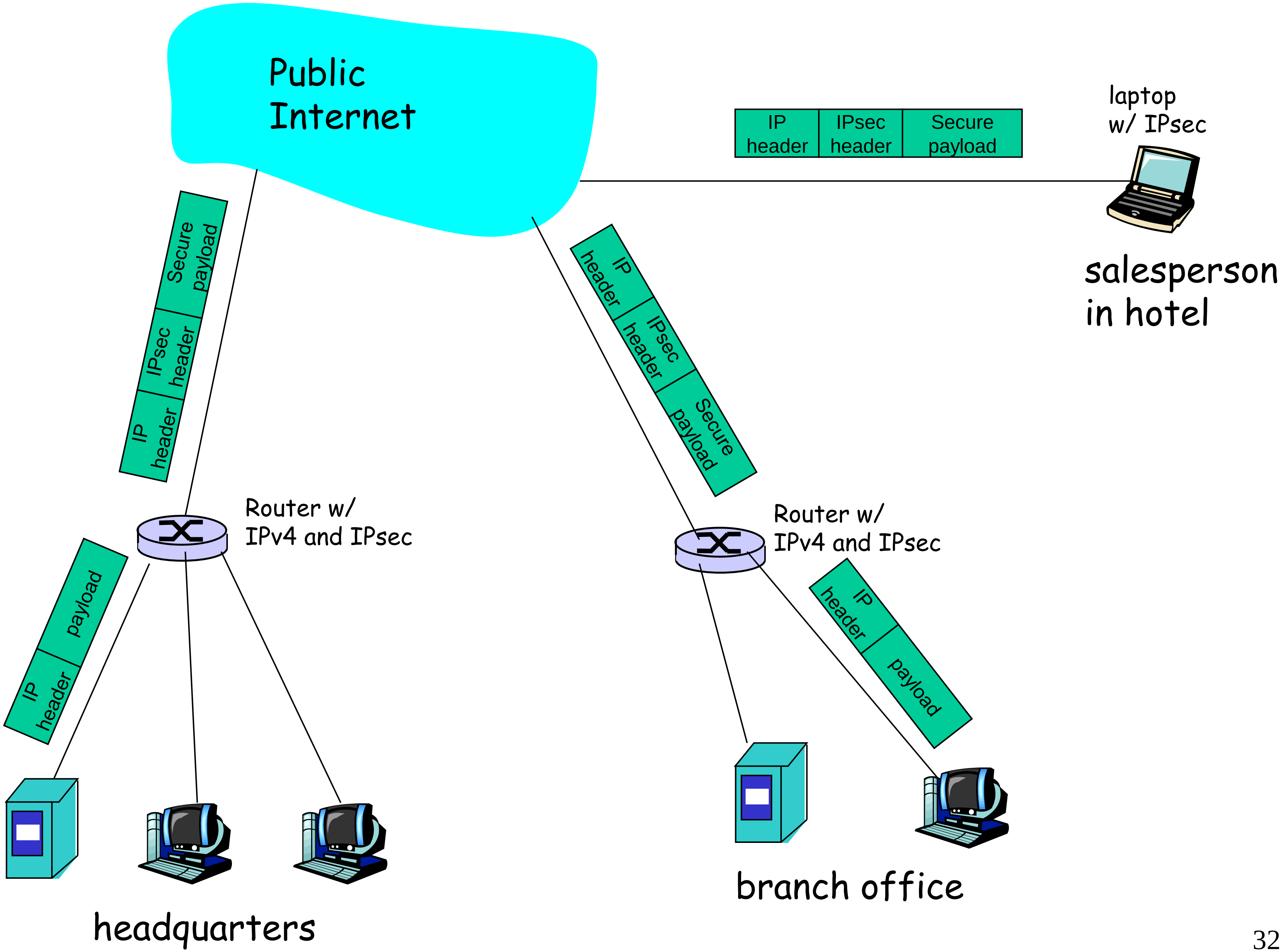
IPSec history

- IPSec(IP Security)产生于IPv6的制定之中，用于提供IP层的安全性。
- 由于所有因特网通信都要经过IP层的处理，所以提供了IP层的安全性就相当于为整个网络提供了安全通信的基础。
- 鉴于IPv4的应用仍然很广泛，所以后来在IPSec的制定中也增添了对IPv4的支持。
- 在2005年第二版标准文档发布，新的文档定义在 RFC 4301 和 RFC 4309 中。

Virtual Private Networks (VPNs)

- Institutions often want private networks for security.
  - Costly! Separate routers, links, DNS infrastructure.
- With a VPN, institution's inter-office traffic is sent over public Internet instead.
  - But inter-office traffic is encrypted before entering public Internet

Virtual Private Network (VPN)

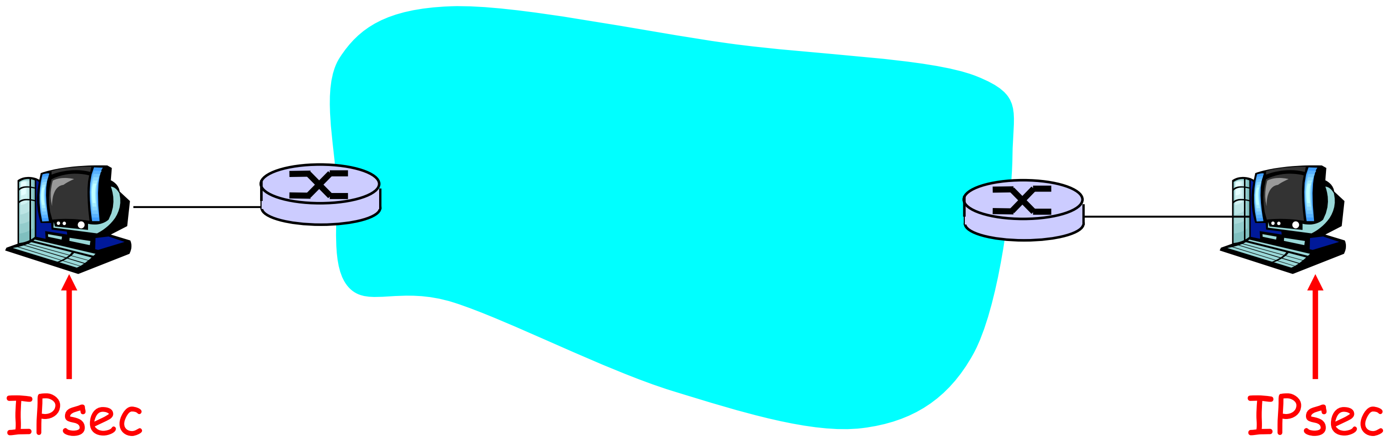


IPsec services

- Data integrity
- Origin authentication
- Replay attack prevention
- Confidentiality

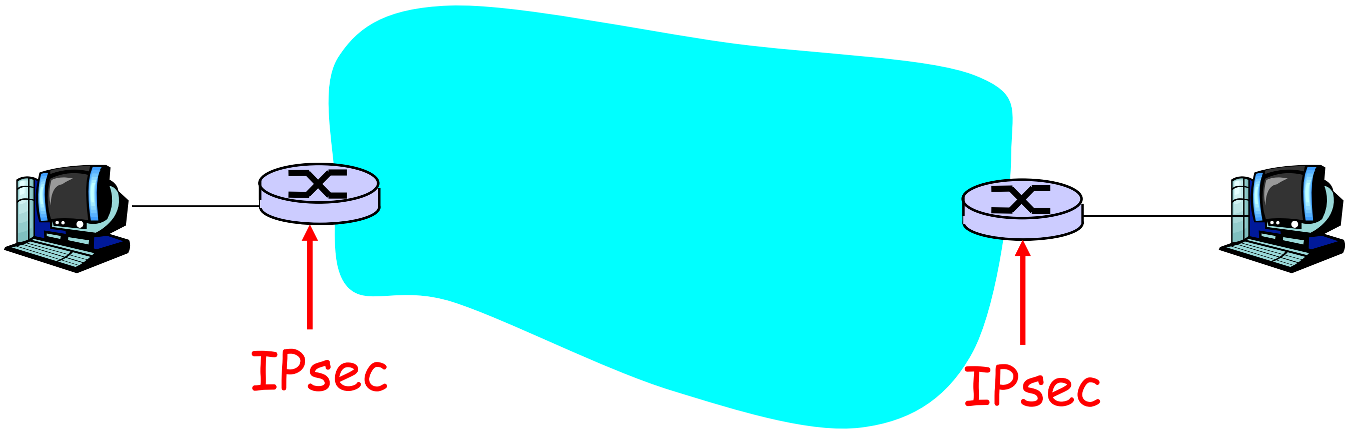
- Two protocols providing different service models:
  - AH
  - ESP

IPsec Transport Mode



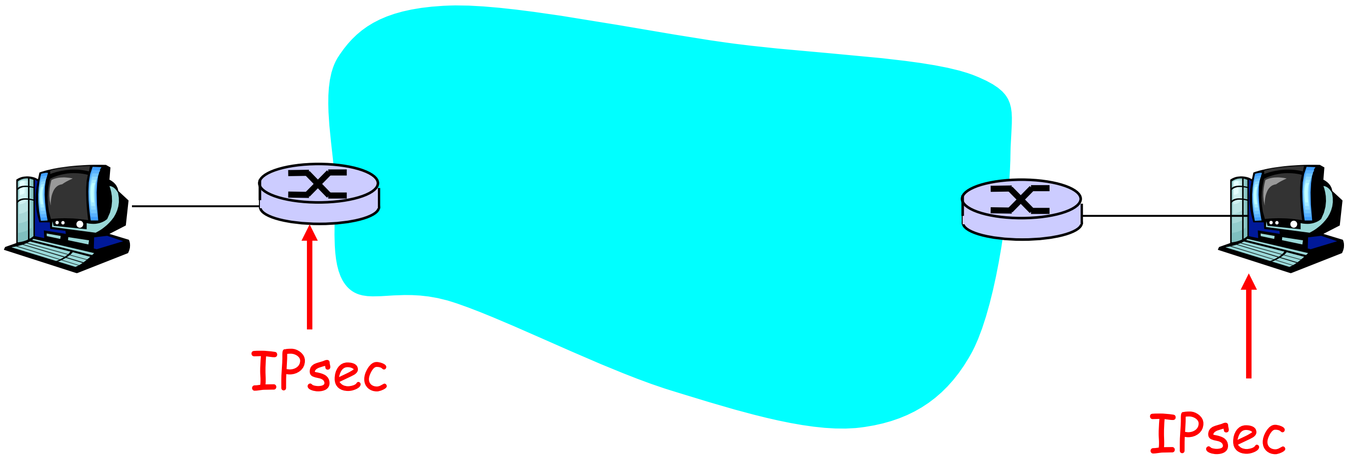
- IPsec datagram emitted and received by end-system.
- Protects upper level protocols

IPsec - tunneling mode (1)



- End routers are IPsec aware. Hosts need not be.

IPsec - tunneling mode (2)



- Also tunneling mode.

Two protocols

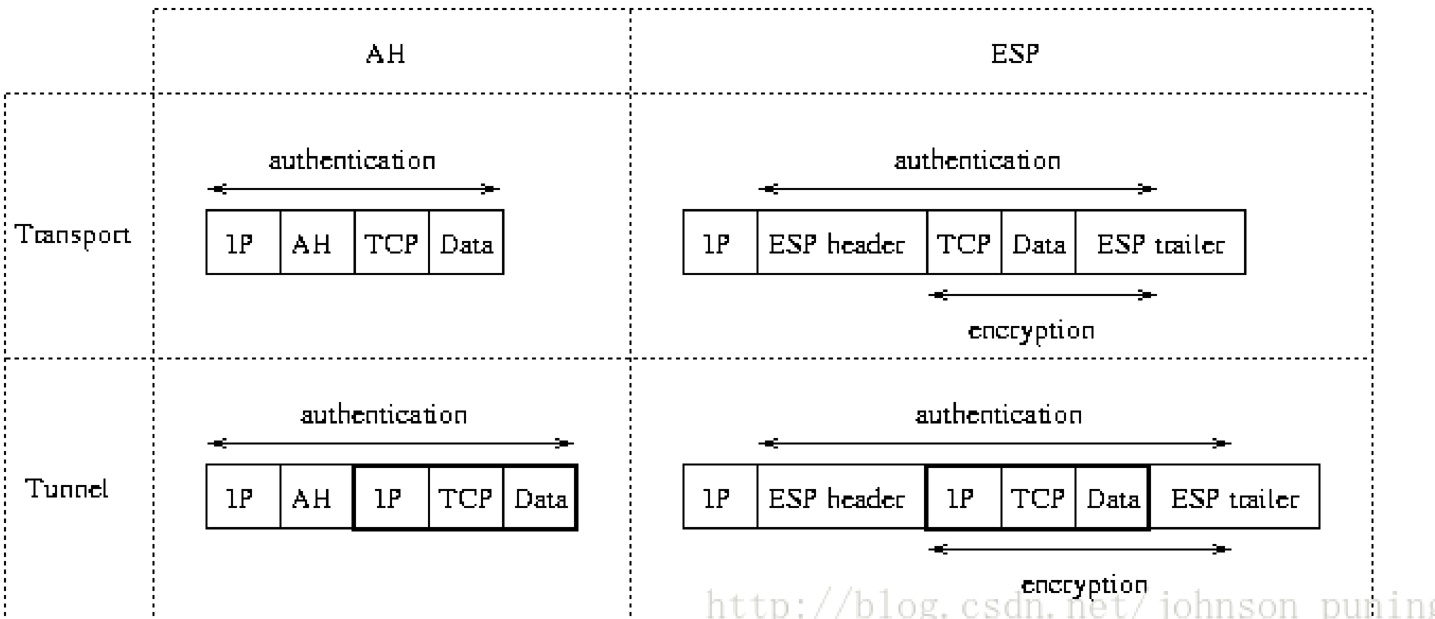
- Authentication Header (AH) protocol
  - provides source authentication & data integrity but *not* confidentiality
- Encapsulation Security Protocol (ESP)
  - provides source authentication,data integrity, *and confidentiality*
  - more widely used than *AH*

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

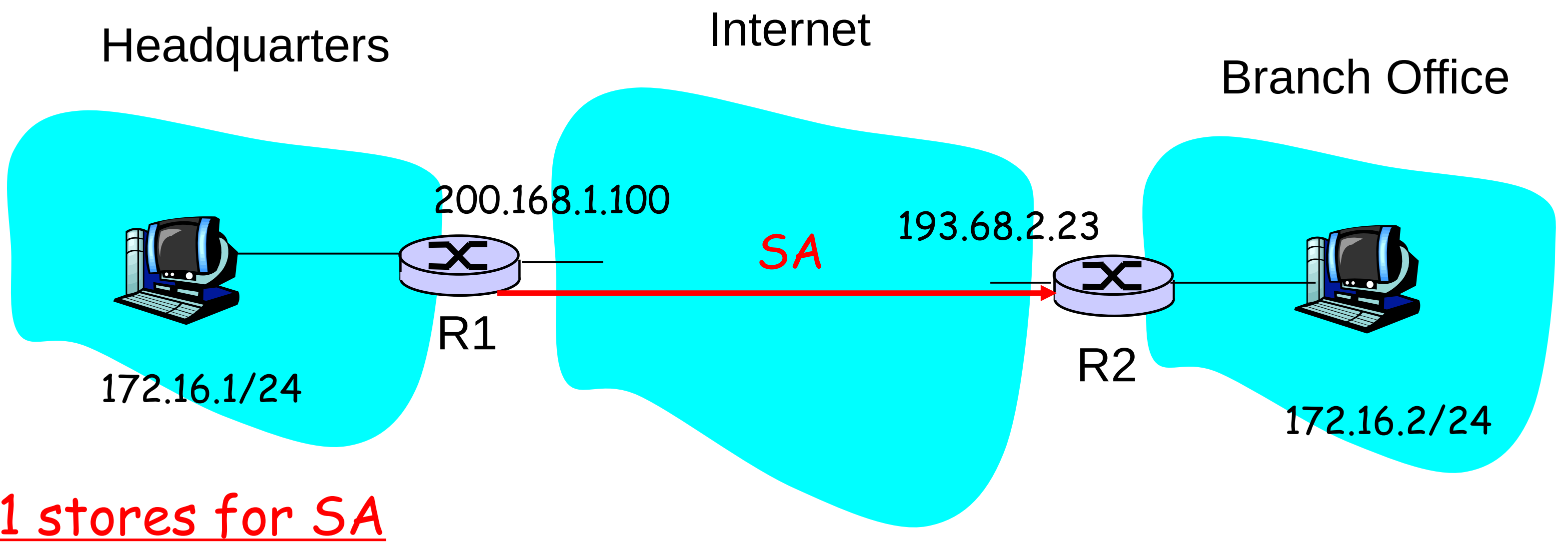
Four combinations are possible!



Security associations (SAs)

- Before sending data, a virtual connection is established from sending entity to receiving entity.
- Called "security association (SA)"
  - SAs are simplex: for only one direction
- Both sending and receiving entites maintain *state information* about the SA
  - Recall that TCP endpoints also maintain state information.
  - IP is connectionless; IPsec is connection-oriented!
- How many SAs in VPN w/ headquarters, branch office, and n traveling salesperson?

Example SA from R1 to R2



R1 stores for SA

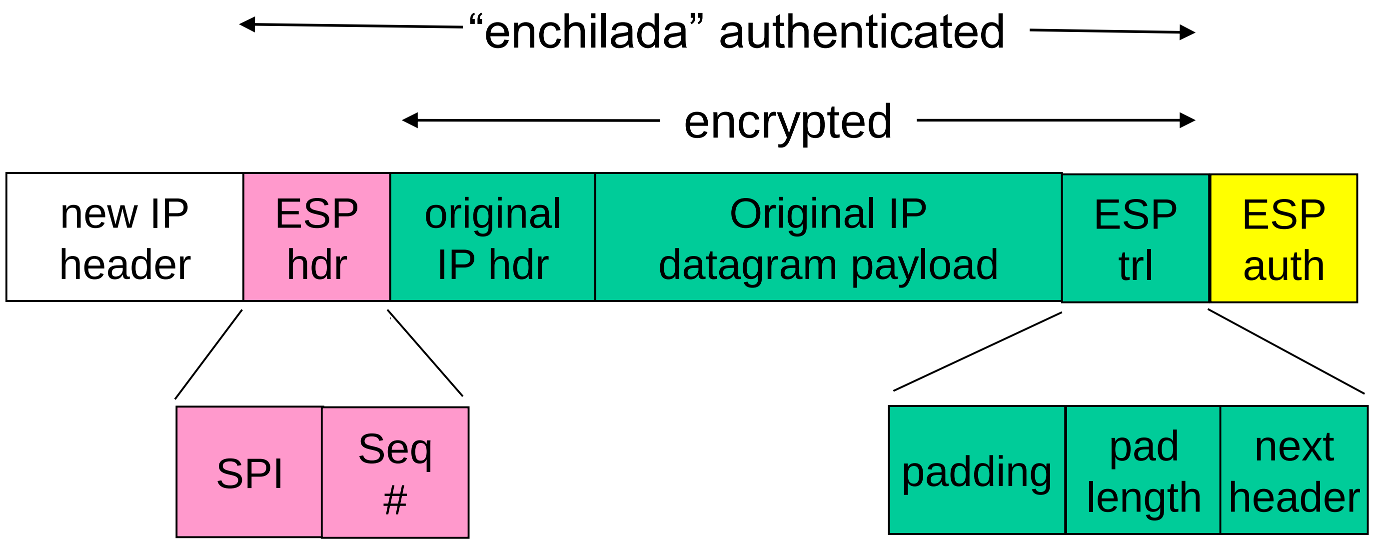
- 32-bit identifier for SA: *Security Parameter Index (SPI)*
- the origin interface of the SA (200.168.1.100)
- destination interface of the SA (193.68.2.23)
- type of encryption to be used (for example, 3DES with CBC)
- encryption key
- type of integrity check (for example, HMAC with with MD5)
- authentication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

Security Association Database (SAD)

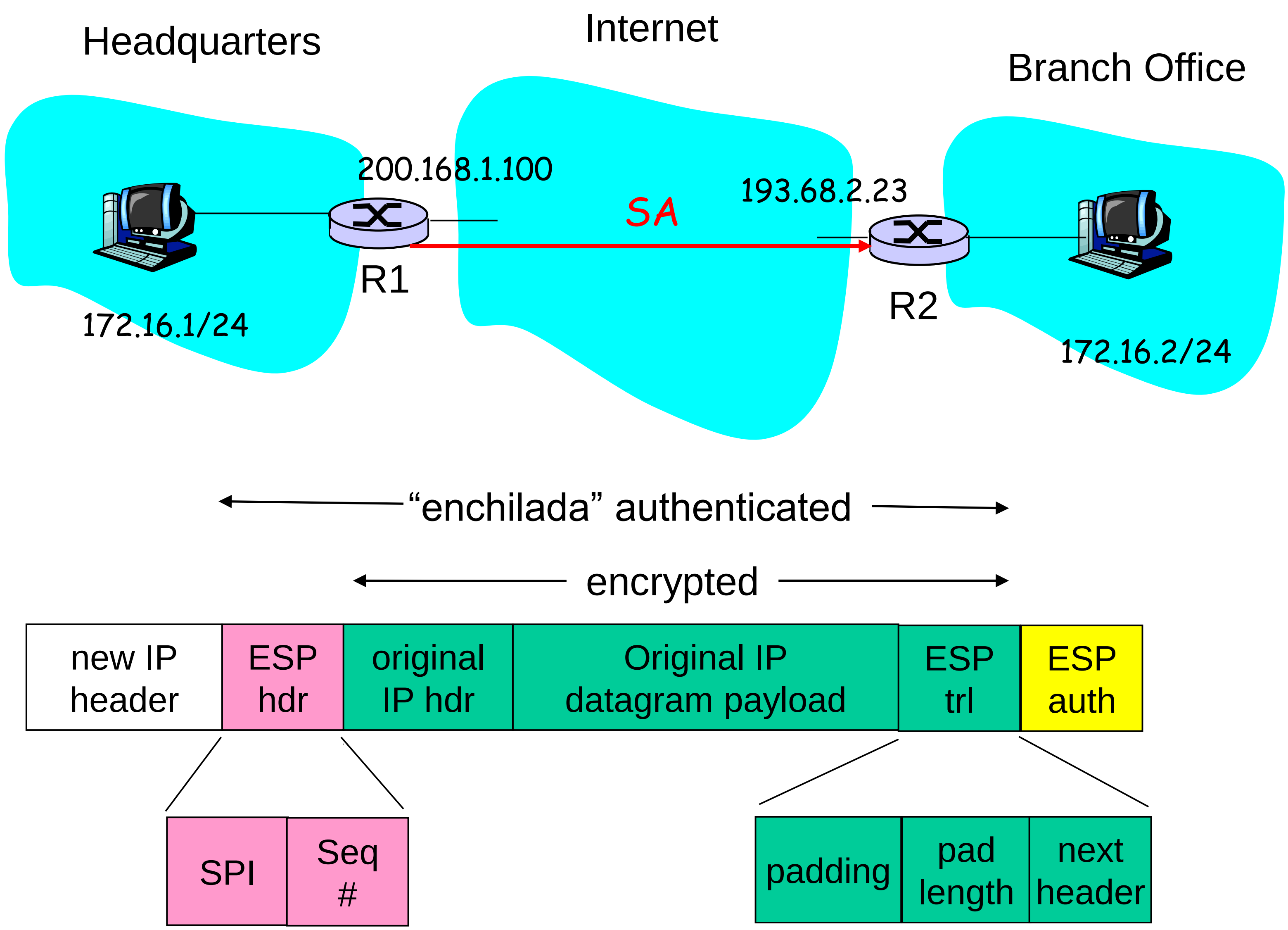
- Endpoint holds state of its SAs in a SAD, where it can locate them during processing.
- With n salespersons, 2 + 2n SAs in R1's SAD
- When sending IPsec datagram, R1 accesses SAD to determine how to process datagram.
- When IPsec datagram arrives to R2, R2 examines SPI in IPsec datagram, indexes SAD with SPI, and processes datagram accordingly.

IPsec datagram

Focus for now on tunnel mode with ESP



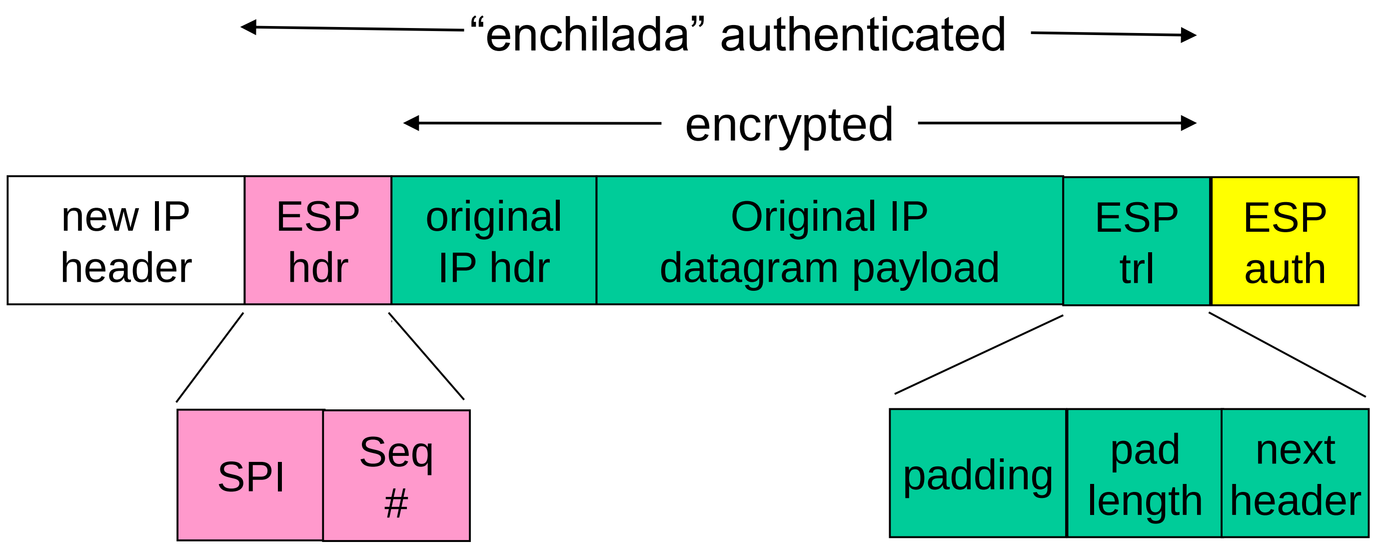
What happens?



R1 converts original datagram into 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
  - But 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 the 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 contents of original 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?

Internet Key Exchange

- In previous examples, we manually established 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…

- Such manually keying is impractical for large VPN with, say, hundreds of sales people.
- Instead use *IPsec IKE (Internet Key Exchange)*

IKE: PSK and PKI

- Authentication (proof who you are) with either
  - pre-shared secret (PSK) or
  - with PKI (pubic/private keys and certificates).
- With PSK, both sides start with secret:
  - then run IKE to authenticate each other and to generate IPsec SAs (one in each direction), including encryption and authentication keys
- With PKI, both sides start with public/private key pair and certificate.
  - run IKE to authenticate each other and 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
    - Also called ISAKMP security association
  - Phase 2: ISAKMP is used to securely negotiate the 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

Summary of IPsec

- IKE message exchange for algorithms, secret keys, SPI numbers
- Either the AH or the ESP protocol (or both)
- The AH protocol provides integrity and source authentication
- The 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

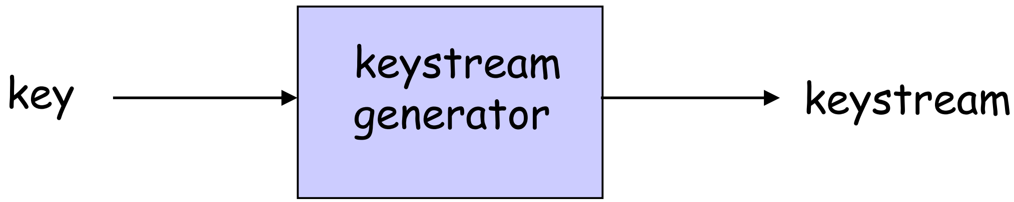
Chapter 8 roadmap

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

WEP Design Goals

- Symmetric key crypto
  - Confidentiality
  - Station authorization
  - Data integrity
- Self synchronizing: each packet separately encrypted
  - Given encrypted packet and key, can decrypt; can continue to decrypt packets when preceding packet was lost
  - Unlike Cipher Block Chaining (CBC) in block ciphers
- Efficient
  - Can be implemented in hardware or software

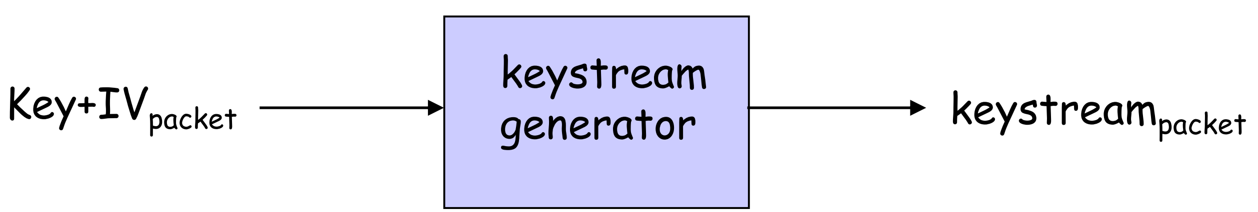
Review: Symmetric Stream Ciphers



- Combine each byte of keystream with byte of plaintext to get ciphertext
- m(i) = ith unit of message
- ks(i) = ith unit of keystream
- c(i) = ith unit of ciphertext
- $c(i) = ks(i) \oplus m(i)$ ($\oplus$ = exclusive or)
- $m(i) = ks(i) \oplus c(i)$
- WEP uses RC4

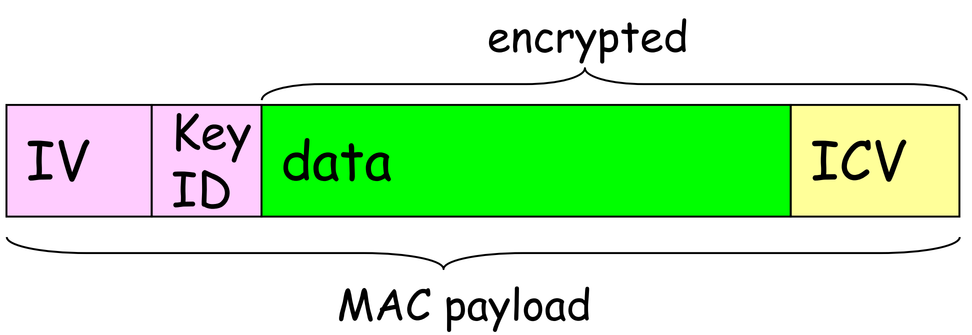
Stream cipher and packet independence

- Recall design goal: each packet separately encrypted
- If for frame n+1, use keystream from where we left off for frame n, then each frame is not separately encrypted
  - Need to know where we left off for packet n
- WEP approach: initialize keystream with key + new IV for each packet:

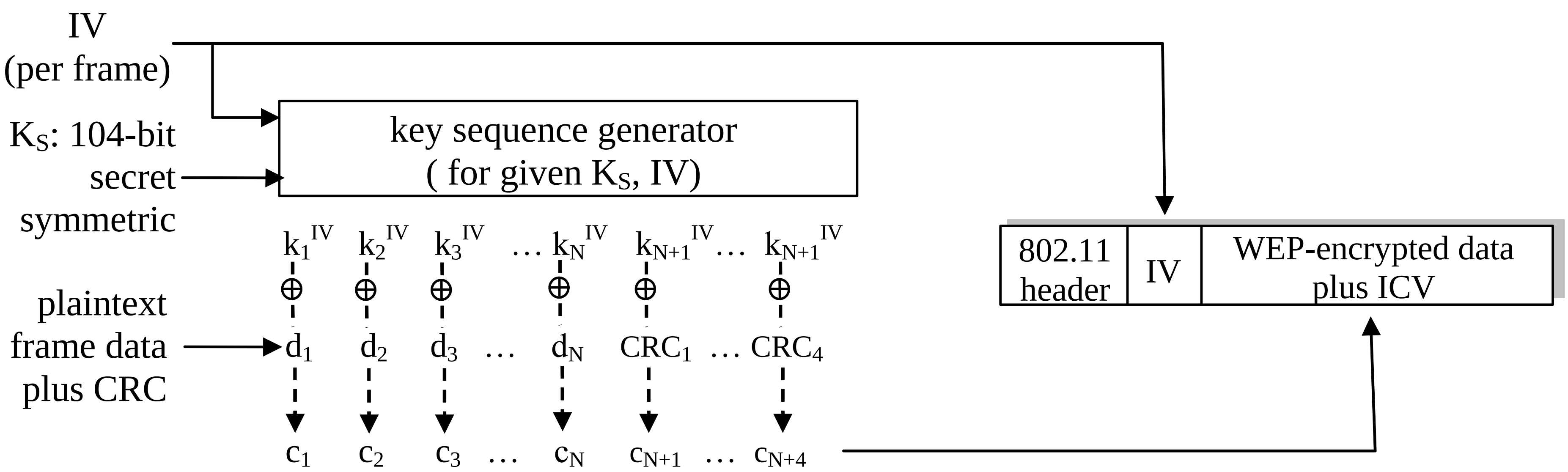


WEP encryption (1)

- Sender calculates Integrity Check Value (ICV) over data
  - four-byte hash/CRC for data integrity
- Each side has 104-bit shared key
- Sender creates 24-bit initialization vector (IV), appends to key: gives 128-bit key
- Sender also appends keyID (in 8-bit field)
- 128-bit key inputted into pseudo random number generator to get keystream
- data in frame + ICV is encrypted with RC4:
  - Bytes of keystream are XORed with bytes of data & ICV
  - IV & keyID are appended to encrypted data to create payload
  - Payload inserted into 802.11 frame

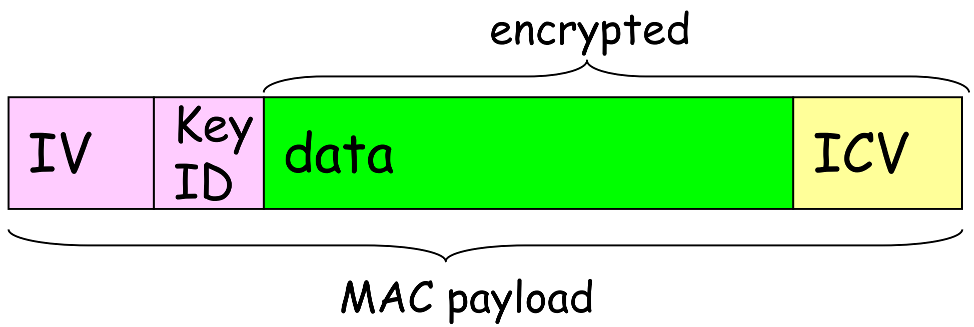


WEP encryption (2)



New IV for each frame

WEP decryption overview

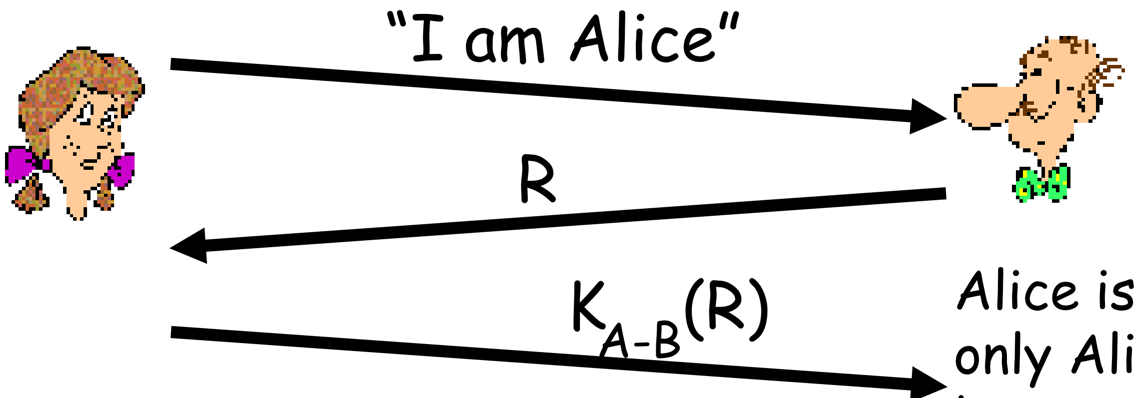


- Receiver extracts IV
- Inputs IV and shared secret key into pseudo random generator, gets keystream
- XORs keystream with encrypted data to decrypt data + ICV
- Verifies integrity of data with ICV
  - Note that message integrity approach used here is different from the MAC (message authentication code) and signatures (using PKI).

End-point authentication w/ nonce

Nonce: number (R) used only *once –in-a-lifetime*

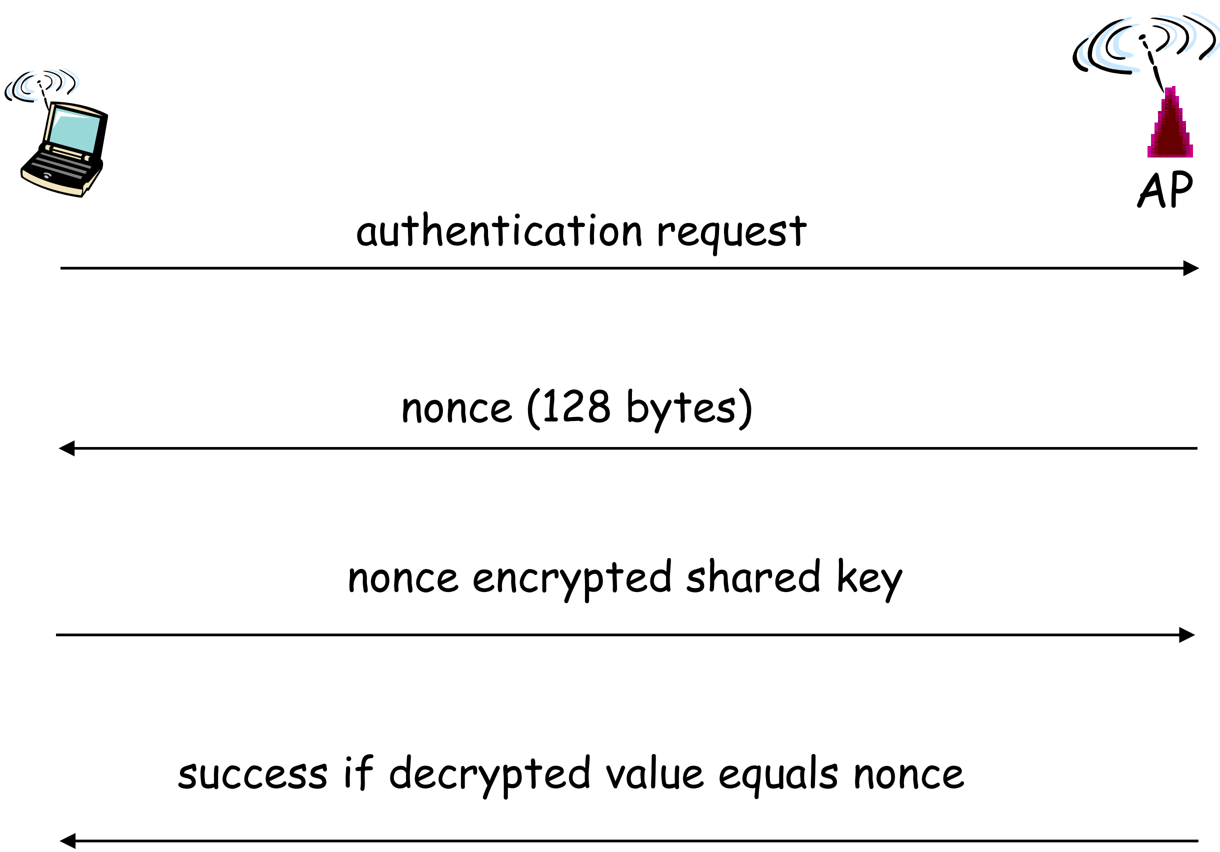
How: to prove Alice "live", Bob sends Alice nonce, R. Alice must return R, encrypted with shared secret key



Alice is live, and only Alice knows key to encrypt nonce, so it must be Alice!

WEP Authentication

Not all APs do it, even if WEP is being used. AP indicates if authentication is necessary in beacon frame. Done before association.



Breaking 802.11 WEP encryption

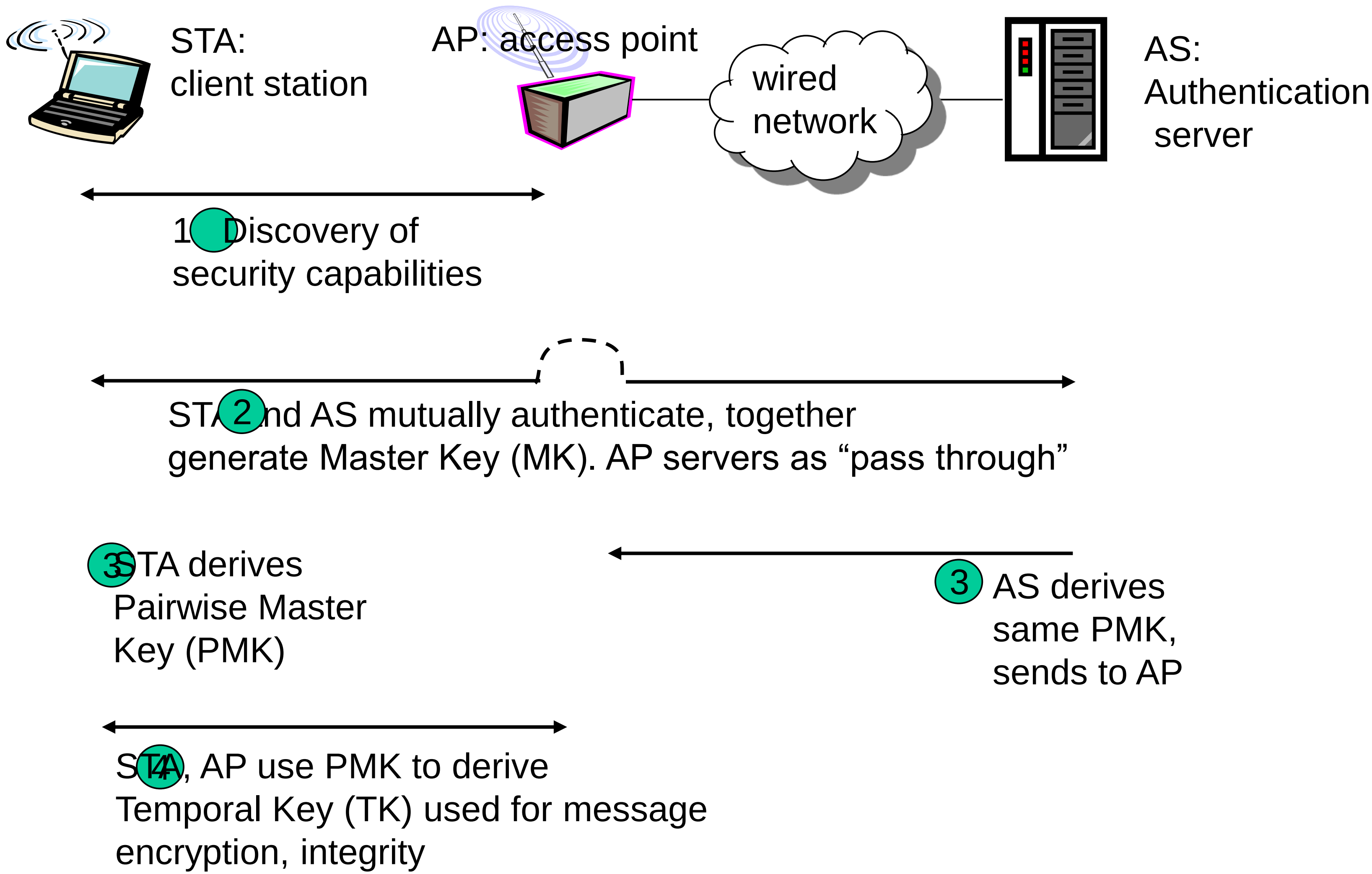
security hole:

- 24-bit IV, one IV per frame, -> IV's eventually reused
- IV transmitted in plaintext -> IV reuse detected
- attack:
  - Trudy causes Alice to encrypt known plaintext d_1 d_2 d_3 d_4 …
  - Trudy sees: $c_i = d_i$ XOR k_i^{IV}
  - Trudy knows c_i d_i, so can compute k_i^{IV}
  - Trudy knows encrypting key sequence k_1^{IV} k_2^{IV} k_3^{IV} …
  - Next time IV is used, Trudy can decrypt!

802.11i: improved security

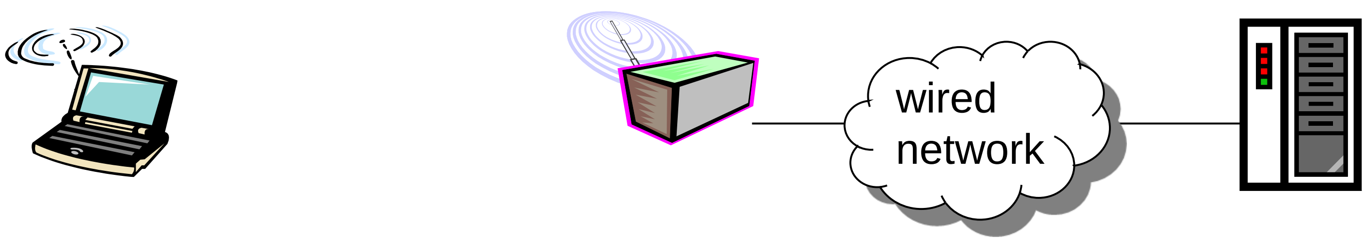
- numerous (stronger) forms of encryption possible
- provides key distribution
- uses authentication server separate from access point

802.11i: four phases of operation



EAP: extensible authentication protocol

- EAP: end-end client (mobile) to authentication server protocol
- EAP sent over separate "links"
  - mobile-to-AP (EAP over LAN)
  - AP to authentication server (RADIUS over UDP)



EAP TLS	
EAP	
EAP over LAN (EAPoL)	RADIUS
IEEE 802.11	UDP/IP

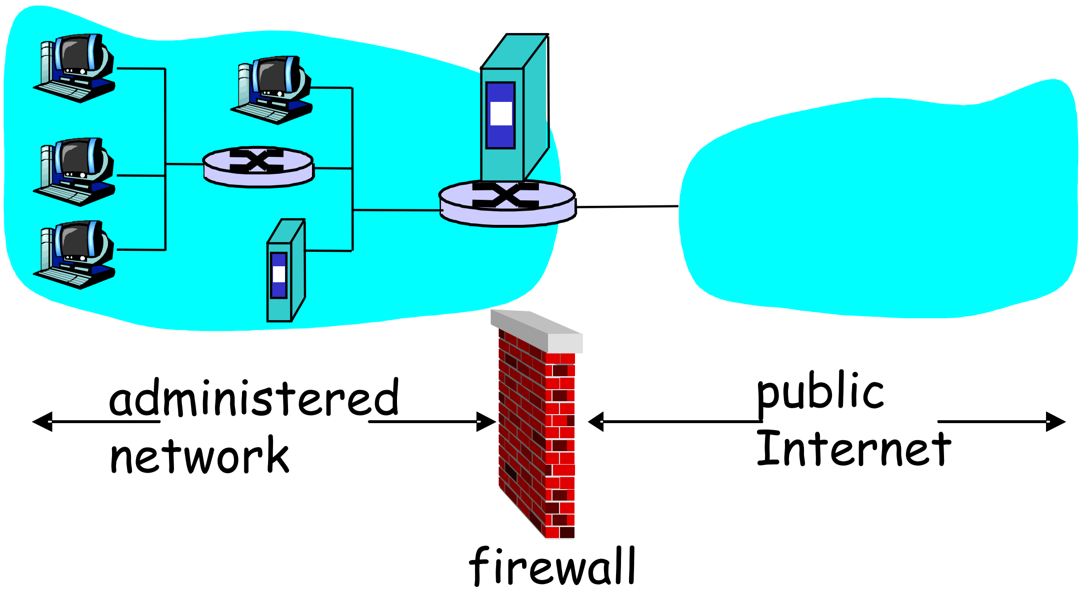
Chapter 8 roadmap

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

Firewalls

firewall

isolates organization's internal net from larger Internet, allowing some packets to pass, blocking others.



Firewalls: Why

prevent denial of service attacks:

- SYN flooding: attacker establishes many bogus TCP connections, no resources left for “real” connections

prevent illegal modification/access of internal data.

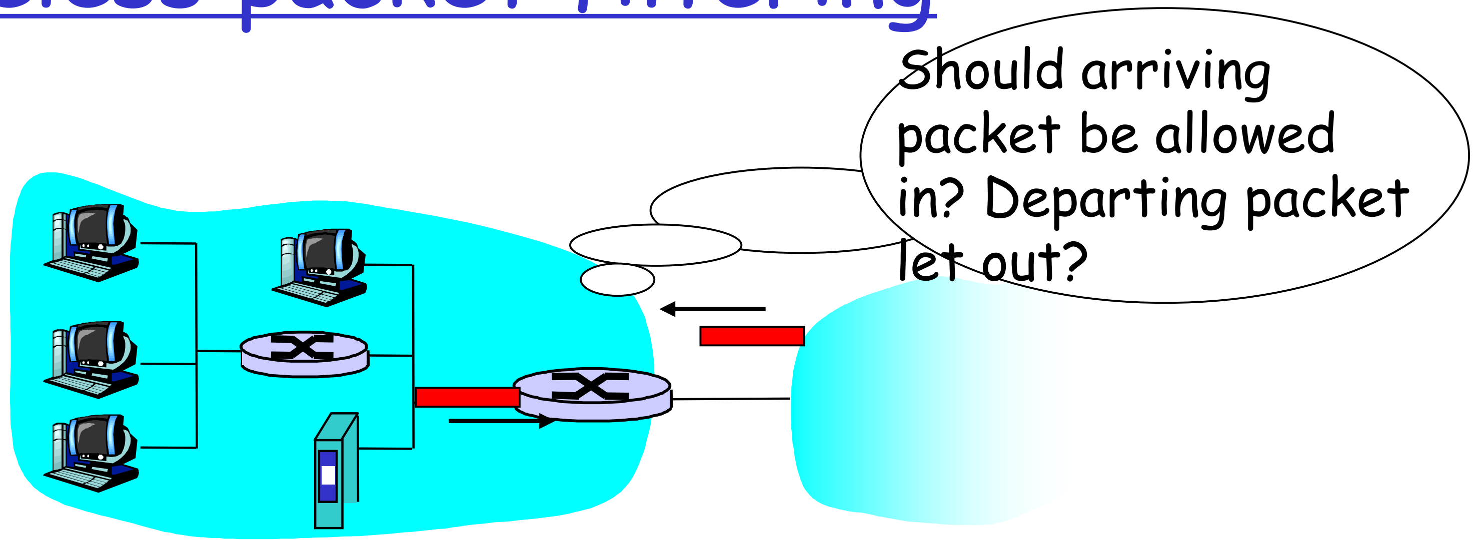
- e.g., attacker replaces CIA’s homepage with something else

allow only authorized access to inside network (set of authenticated users/hosts)

three types of firewalls:

- stateless packet filters
- stateful packet filters
- application gateways

Stateless packet filtering



- internal network connected to Internet via router firewall
- router filters packet-by-packet, decision to forward/drop packet based on:
    - source IP address, destination IP address
    - TCP/UDP source and destination port numbers
    - ICMP message type
    - TCP SYN and ACK bits

Stateless packet filtering: example

- example 1: block incoming and outgoing datagrams with IP protocol field = 17 and with either source or dest port = 23.
  - all incoming, outgoing UDP flows and telnet connections are blocked.
- example 2: Block inbound TCP segments with ACK=0.
  - prevents external clients from making TCP connections with internal clients, but allows internal clients to connect to outside.

Stateless packet filtering: more examples

Policy	Firewall Setting
No outside Web access.	Drop all outgoing packets to any IP address, port 80
No incoming TCP connections, except those for institution's public Web server only.	Drop all incoming TCP SYN packets to any IP except 130.207.244.203, port 80
Prevent Web-radios from eating up the available bandwidth.	Drop all incoming UDP packets - except DNS and router broadcasts.
Prevent your network from being used for a smurf DoS attack.	Drop all ICMP packets going to a "broadcast" address (eg 130.207.255.255).
Prevent your network from being tracerouted	Drop all outgoing ICMP TTL expired traffic

Access Control Lists

- *ACL:* table of rules, applied top to bottom to incoming packets: (action, condition) pairs

action	source address	dest address	protocol	source port	dest port	flag bit
allow	222.22/16	outside of 222.22/16	TCP	> 1023	80	any
allow	outside of 222.22/16	222.22/16	TCP	80	> 1023	ACK
allow	222.22/16	outside of 222.22/16	UDP	> 1023	53	---
allow	outside of 222.22/16	222.22/16	UDP	53	> 1023	----
deny	all	all	all	all	all	all

Stateful packet filtering

- stateless packet filter: heavy handed tool
  - admits packets that "make no sense," e.g., dest port = 80, ACK bit set, even though no TCP connection established:

action	source address	dest address	protocol	source port	dest port	flag bit
allow	outside of 222.22/16	222.22/16	TCP	80	> 1023	ACK

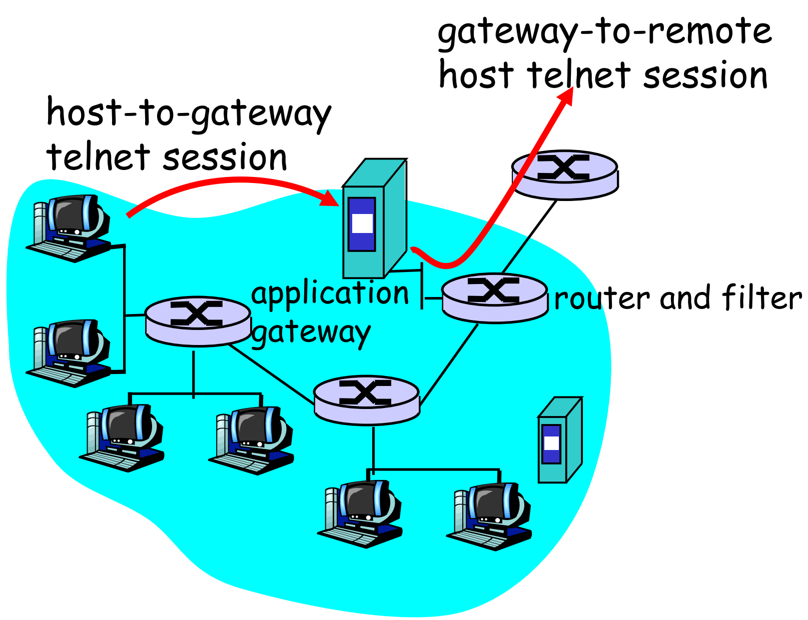
- *stateful packet filter:* track status of every TCP connection
  - track connection setup (SYN), teardown (FIN): can determine whether incoming, outgoing packets "makes sense"
  - timeout inactive connections at firewall: no longer admit packets

Stateful packet filtering

- ACL augmented to indicate need to check connection state table before admitting packet

action	source address	dest address	proto	source port	dest port	flag bit	check conxion
allow	222.22/16	outside of 222.22/16	TCP	> 1023	80	any	
allow	outside of 222.22/16	222.22/16	TCP	80	> 1023	ACK	✗
allow	222.22/16	outside of 222.22/16	UDP	> 1023	53	---	
allow	outside of 222.22/16	222.22/16	UDP	53	> 1023	----	✗
deny	all	all	all	all	all	all	

Application gateways



- filters packets on application data as well as on IP/TCP/UDP fields.
- example: allow select internal users to telnet outside.

1. require all telnet users to telnet through gateway.
2. for authorized users, gateway sets up telnet connection to dest host. Gateway relays data between 2 connections
3. router filter blocks all telnet connections not originating from gateway.

Limitations of firewalls and gateways

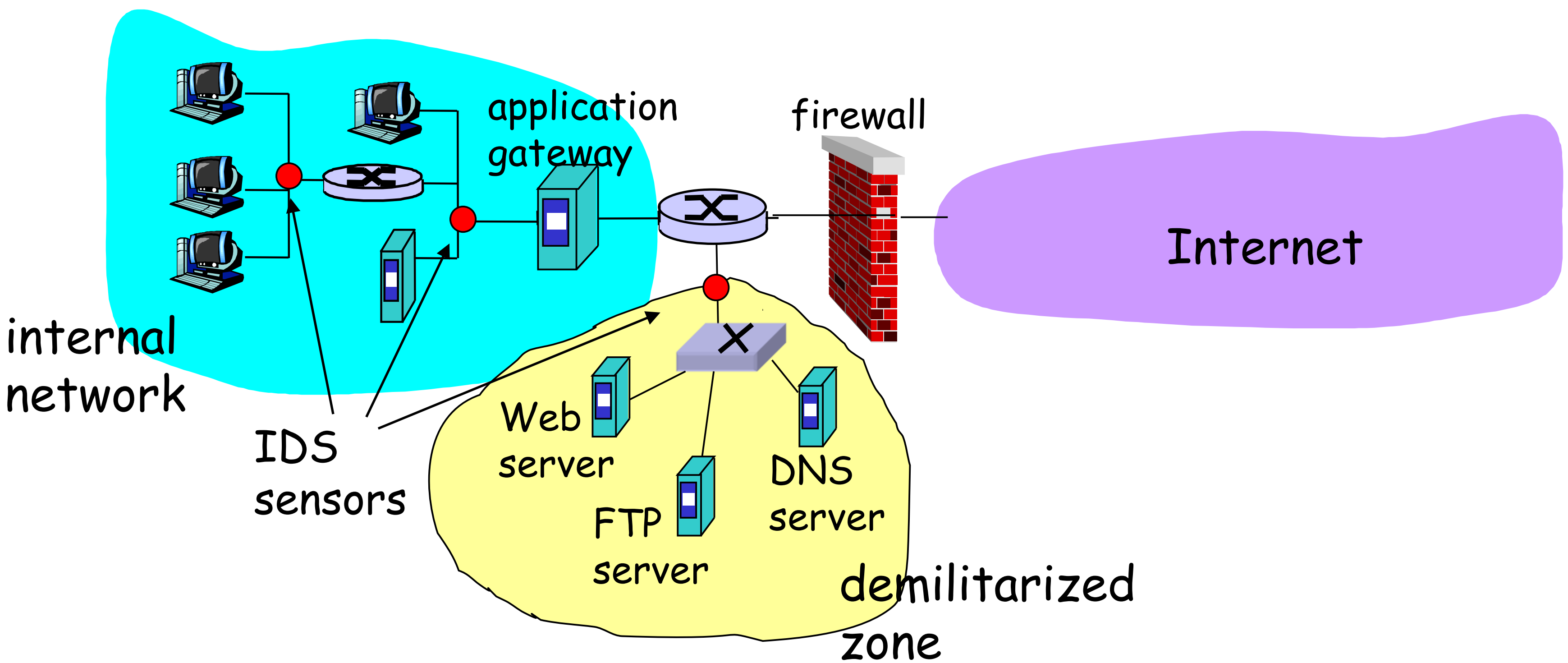
- IP spoofing: router can't know if data "really" comes from claimed source
- if multiple app's. need special treatment, each has own app. gateway.
- client software must know how to contact gateway.
  - e.g., must set IP address of proxy in Web browser
- filters often use all or nothing policy for UDP.
- tradeoff: degree of communication with outside world, level of security
- many highly protected sites still suffer from attacks.

Intrusion detection systems

- packet filtering:
  - operates on TCP/IP headers only
  - no correlation check among sessions
- *IDS: intrusion detection system*
  - *deep packet inspection:* look at packet contents (e.g., check character strings in packet against database of known virus, attack strings)
  - examine correlation among multiple packets
    - port scanning
    - network mapping
    - DoS attack

Intrusion detection systems

- multiple IDSs: different types of checking at different locations



Network Security (summary)

Basic techniques......

- cryptography (symmetric and public)
- message integrity
- end-point authentication

.... used in many different security scenarios

- secure email
- secure transport (SSL)
- IP sec
- 802.11

Operational Security: firewalls and IDS