南京大学
NANJING UNIVERSITY

Computer Networks

Wenzhong Li, Chen Tian

Nanjing University

Material with thanks to James F. Kurose, Mosharaf Chowdhury, and other colleagues.

Chapter 5. Network Security

- Network Attacks
- Cryptographic Technologies
- Authentication
- Message Integrity
- Key Distribution
- Security in Different Network Layers
- Firewalls

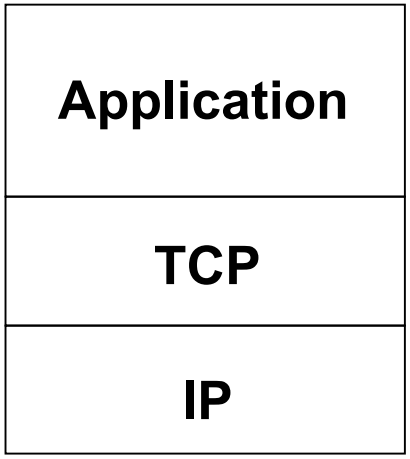
Transport Layer Security

- SSL – Secure Sockets Layer
  - Used by Netscape
  - 1996, SSL v3 was created with public review from industry
  - IETF started with this version to develop a common standard
  - Provides
    - confidentiality
    - integrity
    - authentication
- Variation: TLS – Transport Layer Security
  - 1999, RFC 2246 by IETF
  - Essentially SSL v3.1 with minor difference

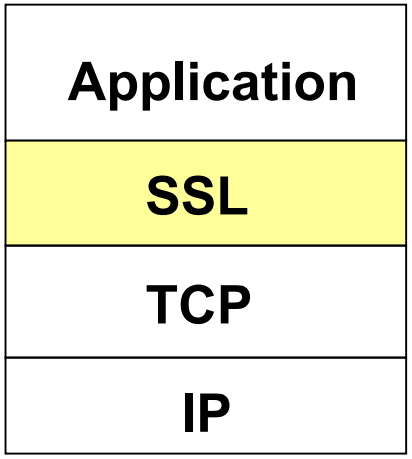
SSL/TLS Characteristics

- Protects application traffic for all applications that are SSL/TLS aware
  - Applications must be SSL enabled by design
- Typical applications
  - http (https) in web browsers
  - IMAP (Internet Message Access Protocol, for email like POP3)
  - LDAP (Lightweight Directory Access Protocol)
  - 802.1x authentication
  - Many VPN systems use SSL/TLS to send encrypted traffic
- Mandatory server authentication
  - Client checks server's certificate, also against CRLs (certificate revocation lists)
- Client authentication supported but normally not used

SSL and TCP/IP



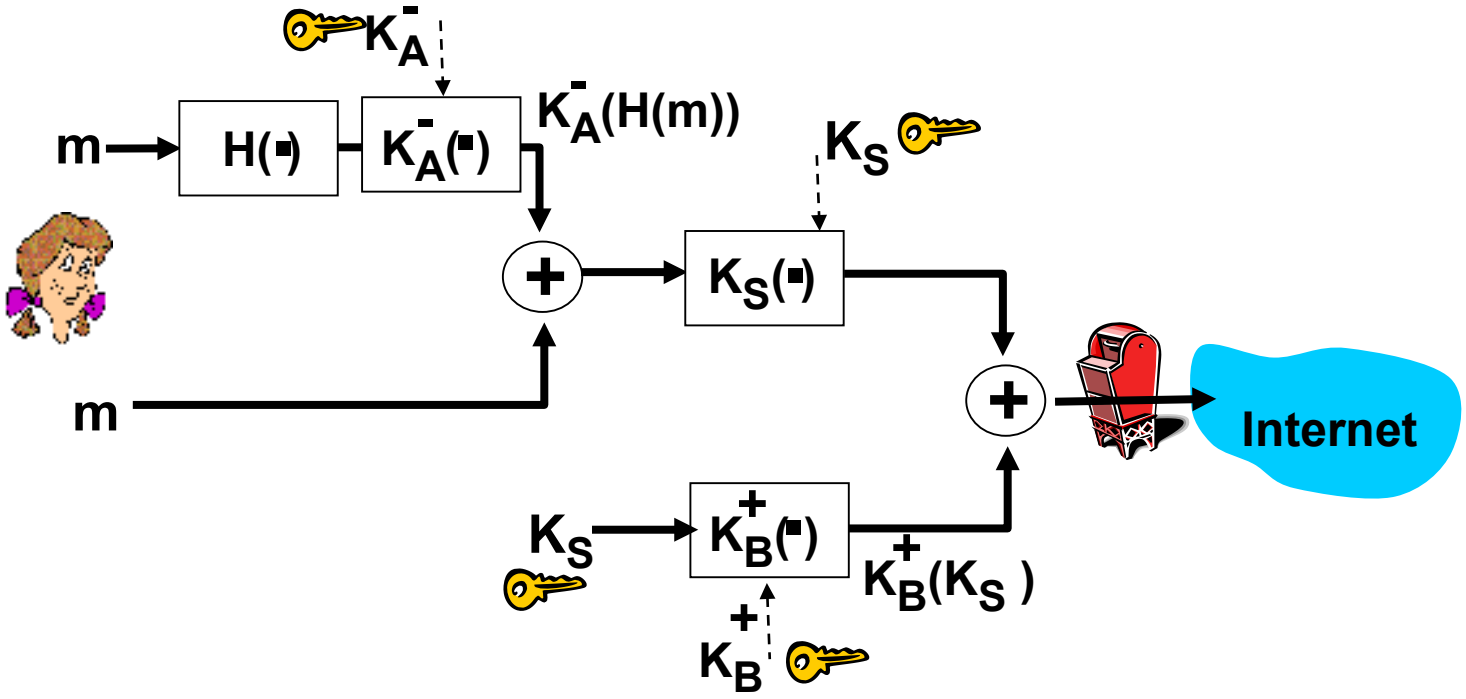
normal application



application with SSL

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

Could do something like PGP:

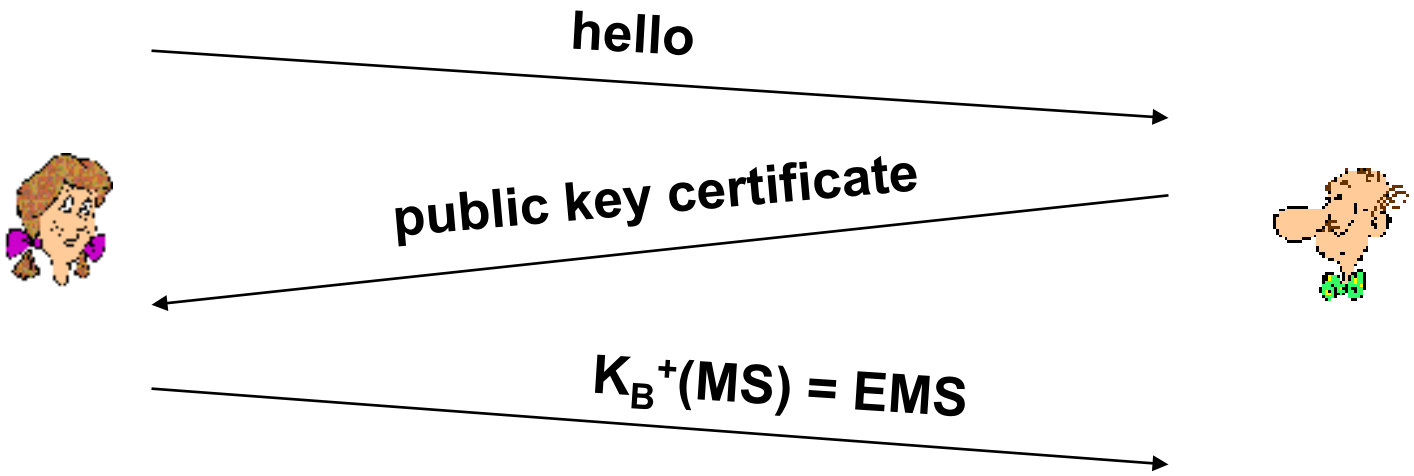


- **PGP(Pretty Good Privacy)**: securing Email by Zimmermann 1991
- But want to send byte **streams** & **interactive** data
- Want set of secret keys for entire connection
- Want certificate exchange as part of protocol: handshake phase

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

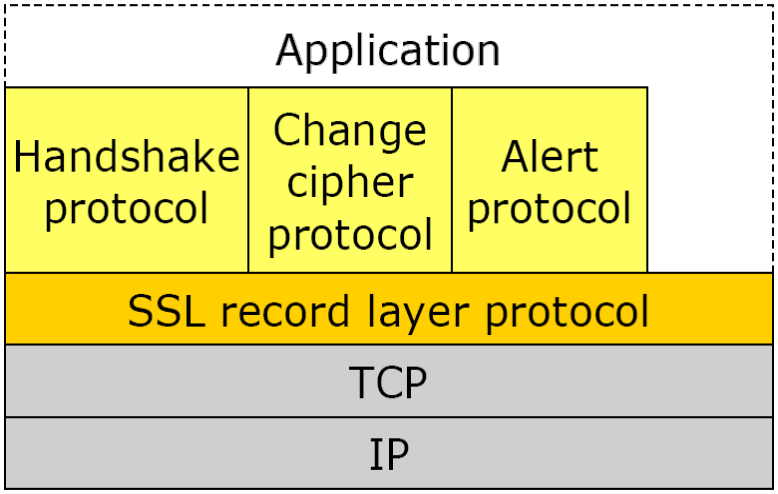
encrypted

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

Real SSL: Architecture

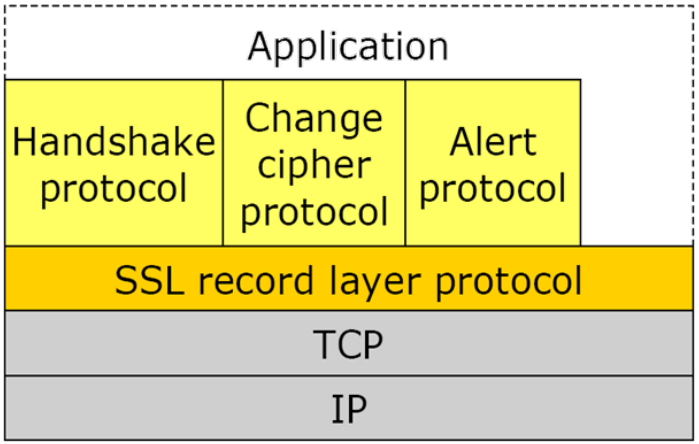
- SSL resides on TCP to provide reliable end-to-end secure service
  - 2 layers of protocols
- Record Protocol provides basic security services to various higher-layer protocols
  - Underlying protocol suite, transparent to applications
- 3 higher-layer protocols for management of SSL exchanges
  - Embedded in specific packages, within IE or Netscape

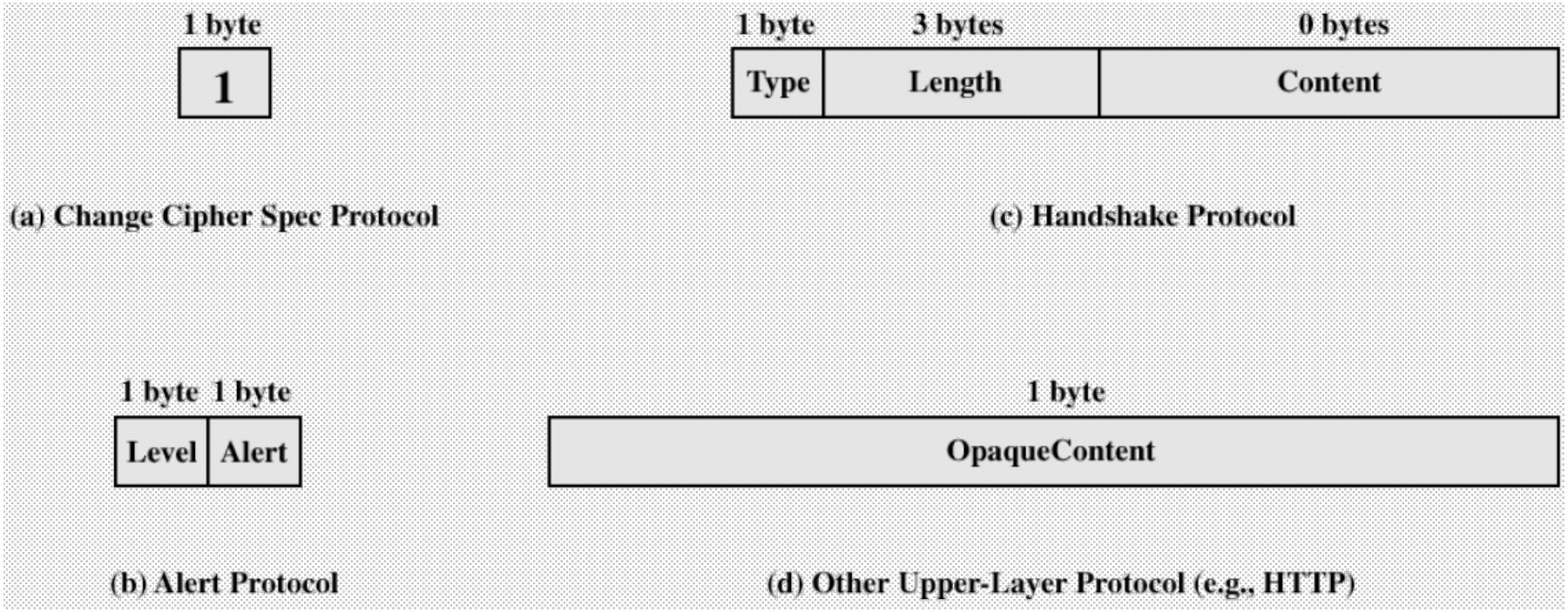


SSL Session

- Association between client and server
  - Created by Handshake Protocol
- Each with a set of cryptographic security parameters
  - Peer's (Server) certificate, for public keys
  - A master secret of 48 octets, for shared keys
  - Compression, cipher or *MAC* (hash) to use
- May have many (TCP) connections within
  - Used to avoid negotiation of new security parameters for each connection
  - Multiple sessions between same pair of apps are supported (not used)

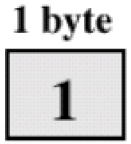
SSL Application Protocols

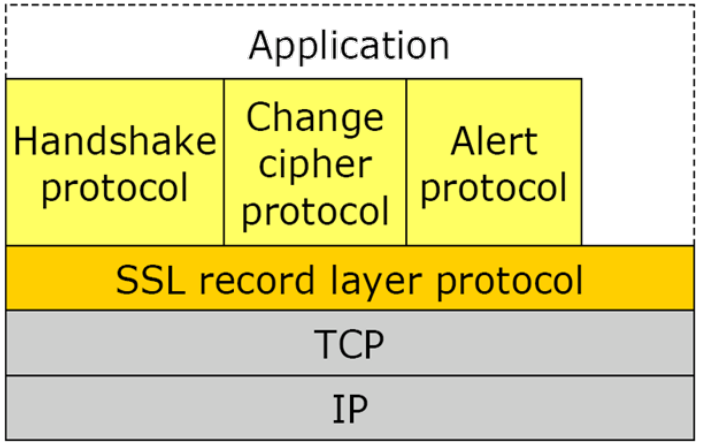




Change Cipher Spec Protocol

- Single octet message
  - Set value *1*
- Cipher suite
  - public-key algorithm
  - symmetric encryption algorithm
  - MAC algorithm
- Negotiation: client, server agree on cipher suite
  - client offers choice
  - server picks one





common SSL symmetric ciphers

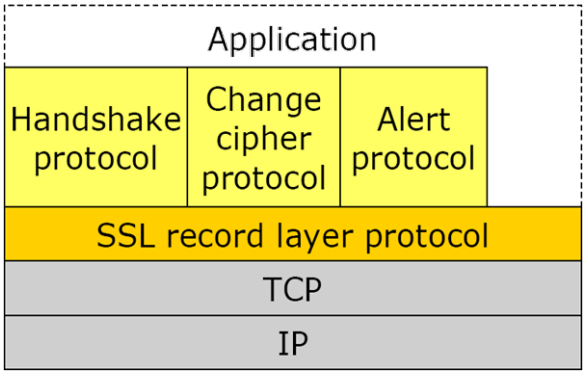
- **DES – Data Encryption Standard: block**
- **3DES – Triple strength: block**
- **RC2 – Rivest Cipher 2: block**
- **RC4 – Rivest Cipher 4: stream**

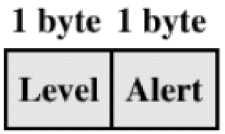
SSL Public key encryption

- **RSA**

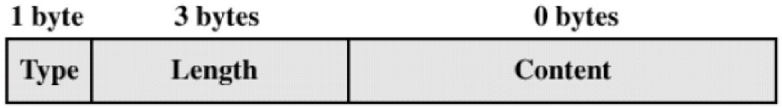
Alert Protocol

- Convey *SSL*-related error or alerts to peer entity
  - Alert messages compressed and encrypted
- Alert level
  - 1: warning, 2: fatal
  - If fatal, *SSL* immediately terminates connection
  - Other connections on session may continue but no new connections accepted on session
- Alert description, e.g.
  - Fatal: UnexpectedMessage, BadRecordMAC, HandshakeFailure
  - Warning: CloseNotify, Certificate Unsupported/Revoked, Illegal Parameter





Handshake Protocol

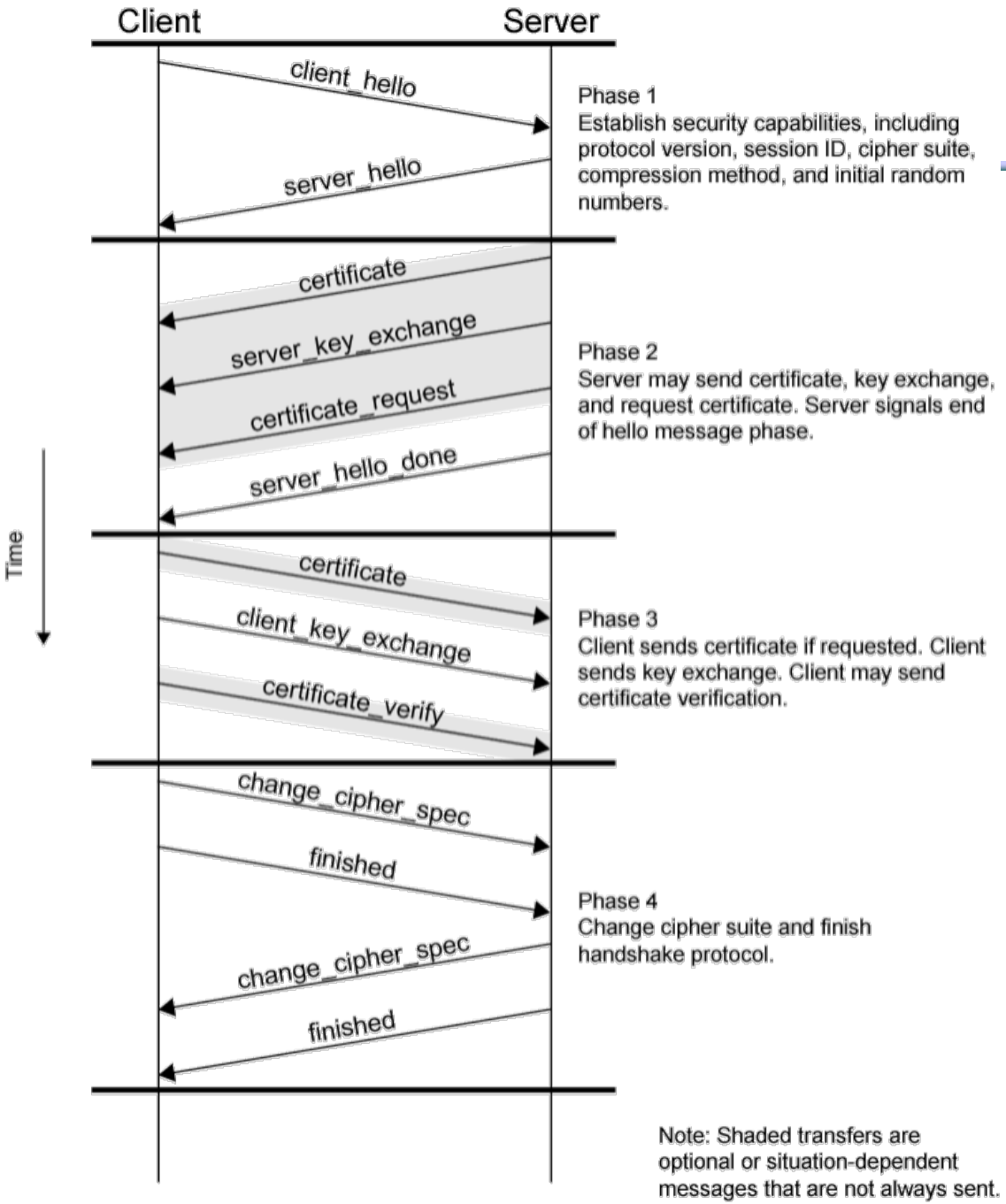


(c) Handshake Protocol

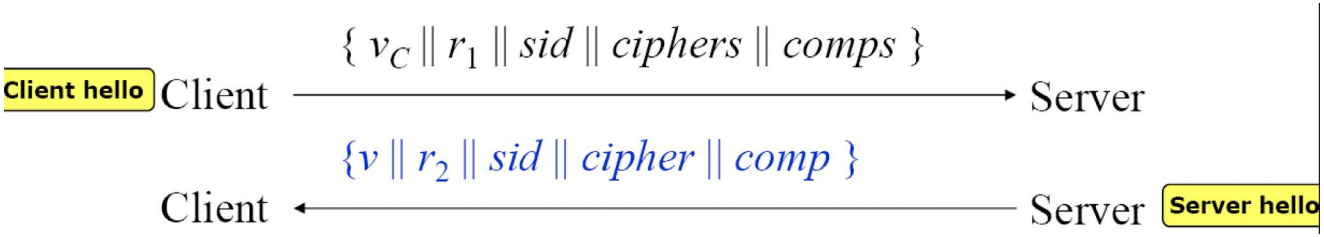
- Purpose
  - Authenticate sender/receiver
  - Negotiate encryption and *MAC* algorithm and cryptographic keys
- 4 rounds
  - Create *SSL* connection between client and server
    - Establish security capabilities
  - Server authenticates itself
    - Presents public key suitable for shared key distribution
  - Client validates server, begins key exchange
  - Acknowledgments all around
    - Change cipher according to agreement

- 建立安全能力
- 服务器鉴别与密钥交换
- 客户机鉴别与密钥交换
- 确认

Handshake Protocol Actions



Round 1



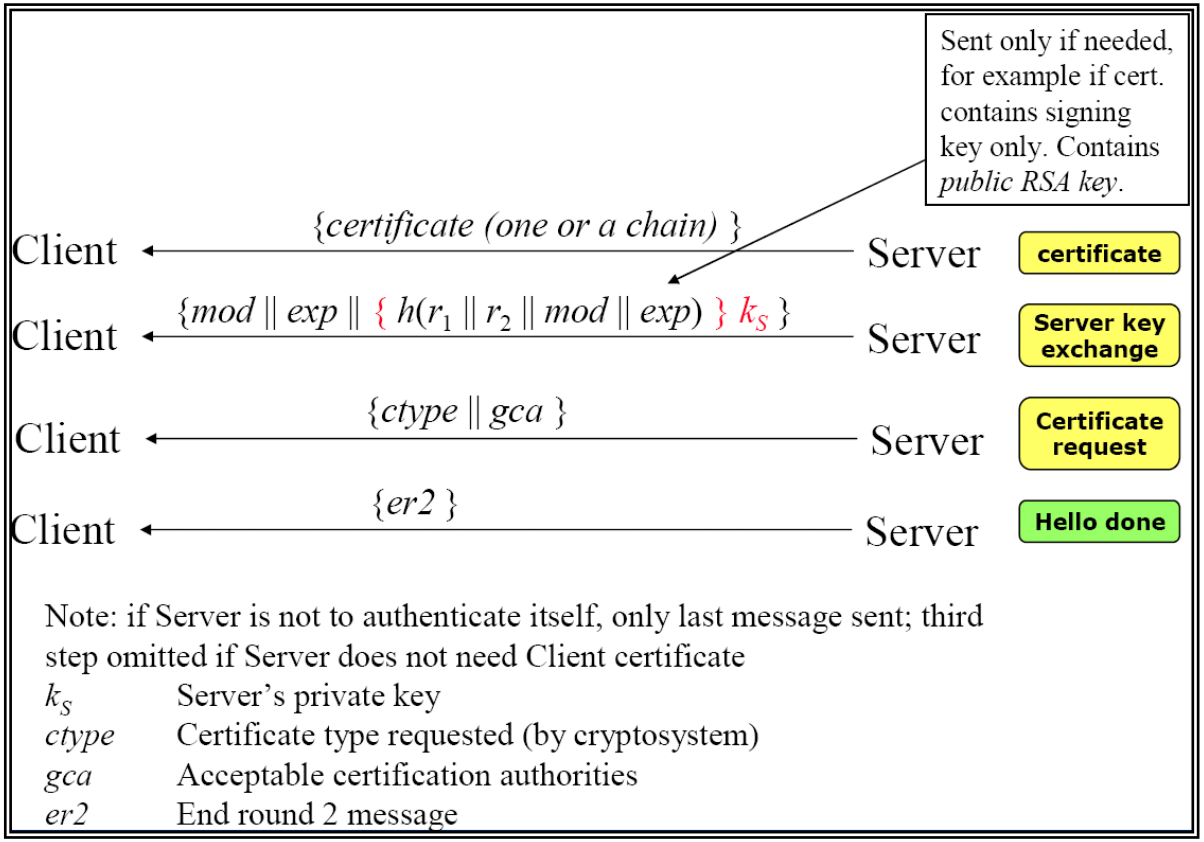
- ClientHello message:
  - v_C: the client's version of SSL
  - r_1: nonces (random number)
  - sid: current session id (0 if new session)
  - Ciphers: a list of ciphers that client supports
  - Comps: a list of compression algorithms that client supports
- ServerHello message:
  - V: highest SSL version both client and server support
  - r2: nonces (random number)
  - sid: current session id (0 if new session)
  - Cipher: the cipher to be used
  - Comp: the compression algorithm to be used

- After round 1, the client knows
  - Version of SSL
  - Cipher algorithms for key exchange, message authentication, and encryption algorithm
  - Compression algorithm
  - Two nonces for key generation

- 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

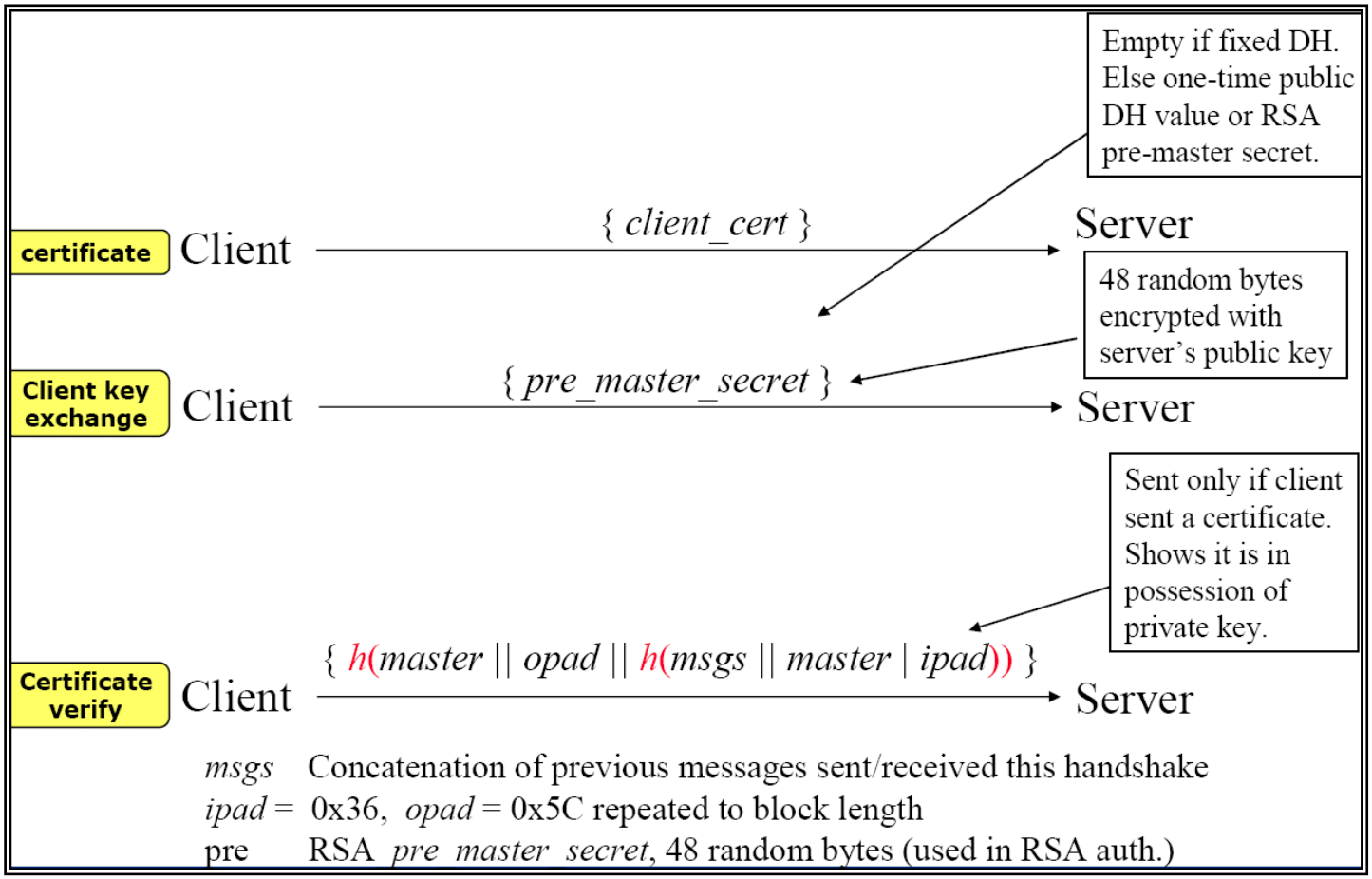
Round 2

- Round 2 depends on underlying encryption scheme
  - Server certificate is required on new session
  - exchange key (depend on algorithm)
  - may request certificates from client
  - Server_hello done



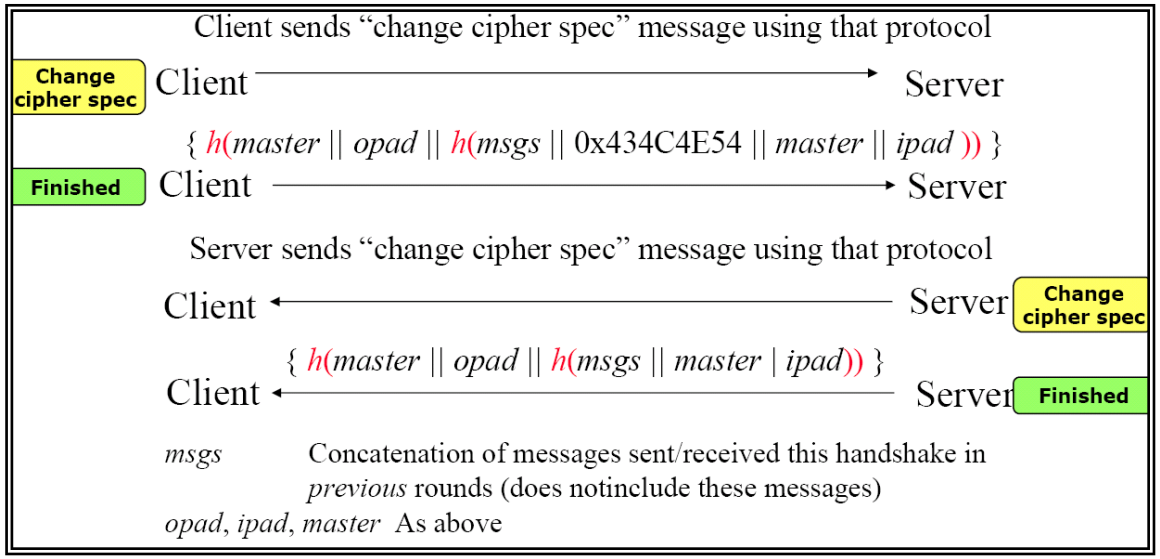
Round 3

- Round 3
  - Client verifies certificate if needed and check server_hello parameters
  - Client sends secrets to server, depending on underlying public-key scheme

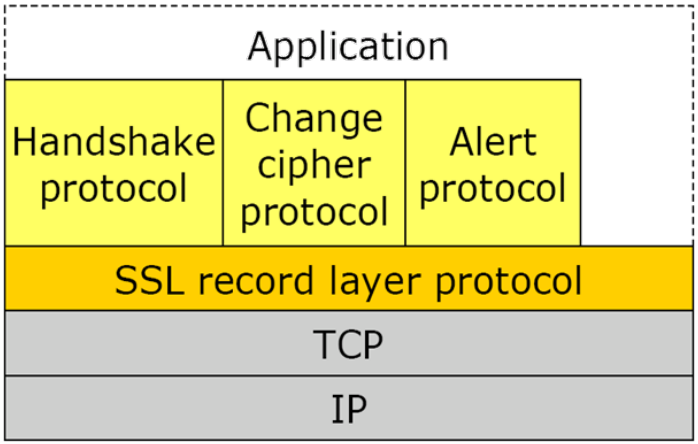


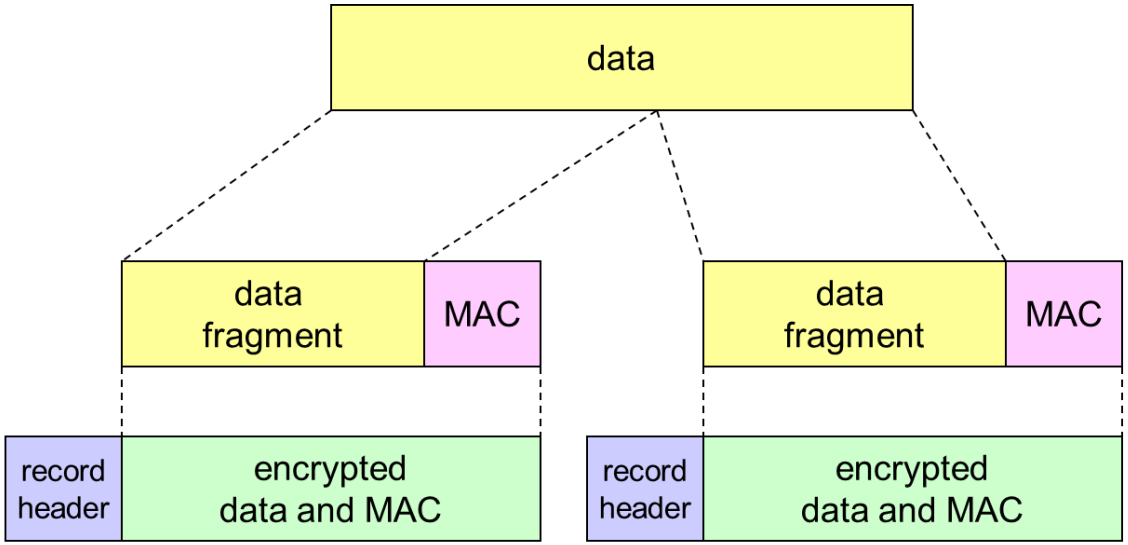
Round 4

- Client sends change_cipher_spec
  - Copies pending CipherSpec into current CipherSpec
  - Sent using Change Cipher Spec Protocol
- Client sends finished message under new algorithms, keys, and secrets
  - Finished message verifies key exchange and authentication successful
- Server sends own change_cipher_spec message
  - Transfers pending CipherSpec to current CipherSpec
  - Sends its finished message
- Handshake complete



SSL Record Protocol

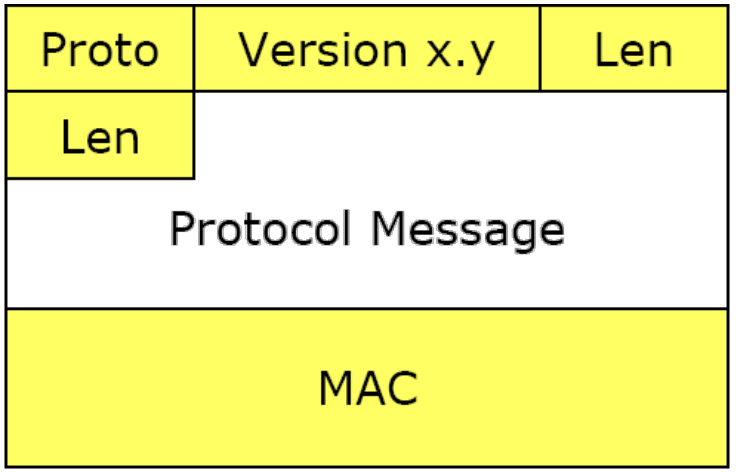




- Each upper-layer message fragmented
  - 2^{14} octets (16384 octets) or less
- Compressed message plus *MAC* encrypted using symmetric encryption
  - Compression optionally applied
- Add SSL record header, PDU transmits in TCP segment

SSL Record PDU

- Protocol Type (1 octet)
  - 20: change_cipher_spec, 21: alert, 22: handshake, 23: application
  - No distinction between applications
- Major Version (1 octet)
  - SSL v3 is 3
- Minor Version (1 octet)
  - SSL v3 is 0
- Compressed Length (2 octets)
  - Per octet, maximum 2^{14}+2048
- MAC (0, 16, or 20 octets)



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

- OpenSSL: The Open Source toolkit for SSL/TLS
  - Widely used in Linux, BSD, Apache server, etc.
- 2014年，Heartbleed 漏洞
  - Keep-alive: the Heartbeat Extension provides a new protocol for TLS/DTLS allowing the usage of keep-alive functionality.
  - A missing bounds check in the handling of the TLS heartbeat extension can be used to reveal up to 64k of memory to a connected client or server.
  - 缓冲区溢出：由于实现时忘记边界检查，如果“载荷长度”字段（payload）被发送端设置得很大，而实际的载荷长度比较短，就会把本来不属于载荷区域的内存复制到响应缓冲区，可泄漏64K内存信息
- OpenSSL 1.0.1g以上版本已修复。

IPSec

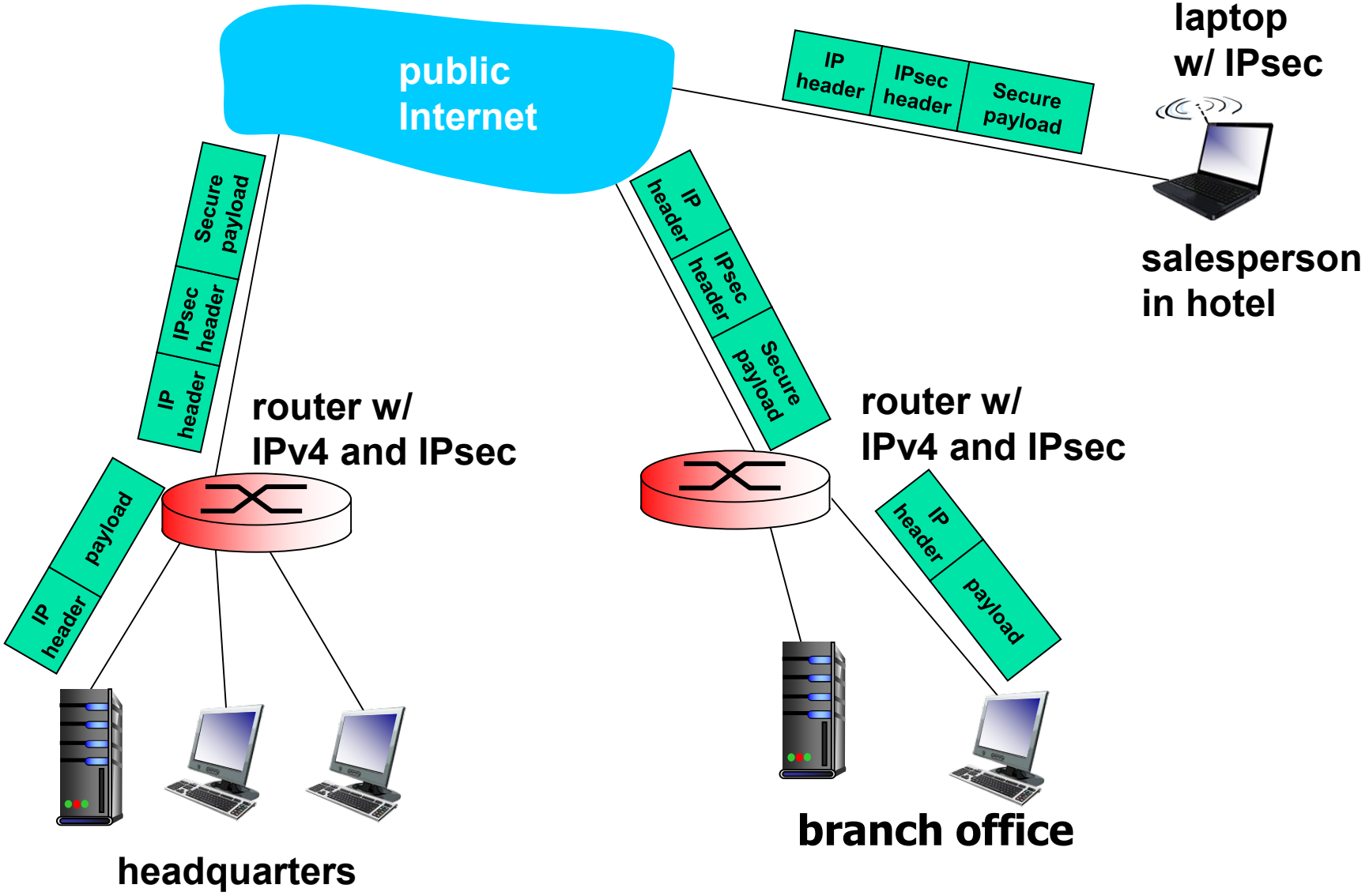
- Encryption of traffic at IP level
  - Transparent for transport layer (TCP, UDP)
  - De-facto standard for site-to-site VPNs
- Mandatory in IPv6, optional in IPv4
- IPsec services
  - data integrity
  - origin authentication
  - replay attack prevention
  - confidentiality
- Application examples
  - Branch office connectivity over the Internet
  - Secure remote access (user to site)
  - Extranet and intranet connectivity
  - Server to server traffic encryption
  - Enhanced electronic commerce security

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

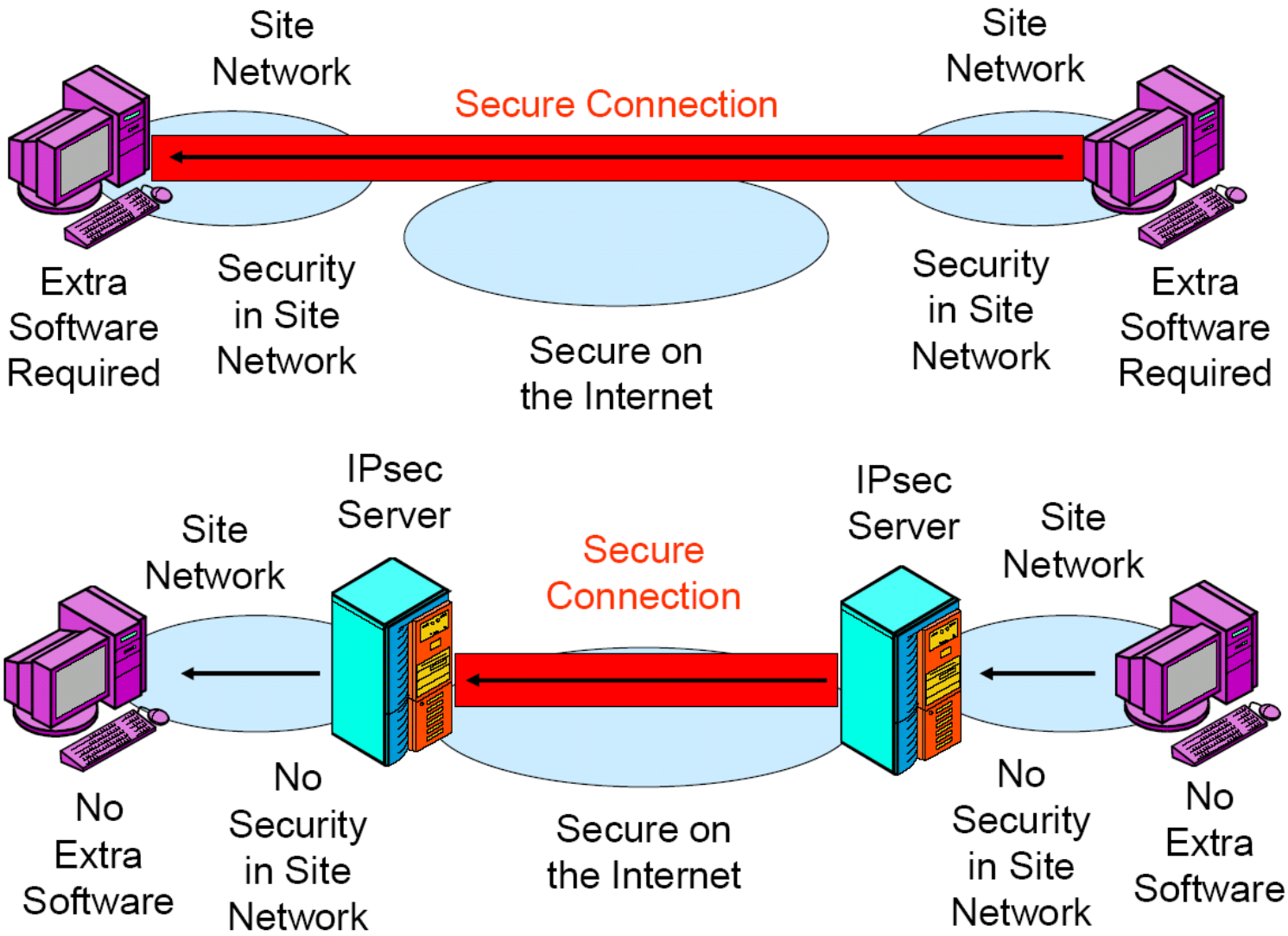
Virtual Private Networks (VPNs)



IPSec Operation

- Transport mode
  - Offers end-to-end encryption
  - Often used for remote access
  - End-devices must implement Ipsec
- Tunnel mode
  - Often used between firewalls
  - Used to build Virtual Private Networks (VPN)
  - Encrypts all traffic over insecure networks

Transport Mode vs. Tunnel Model

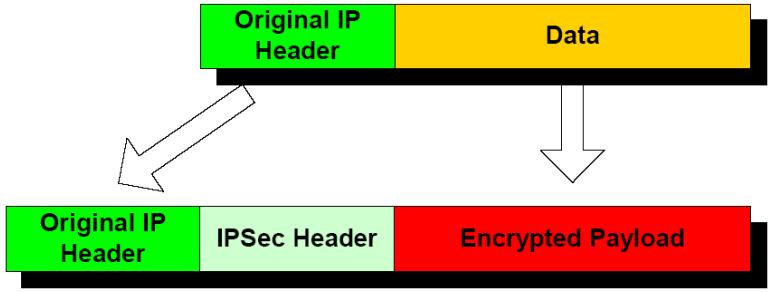


Tunnel and Transport Modes

Transport mode

- Header IP addresses are actual addresses
- Original IP header not protected

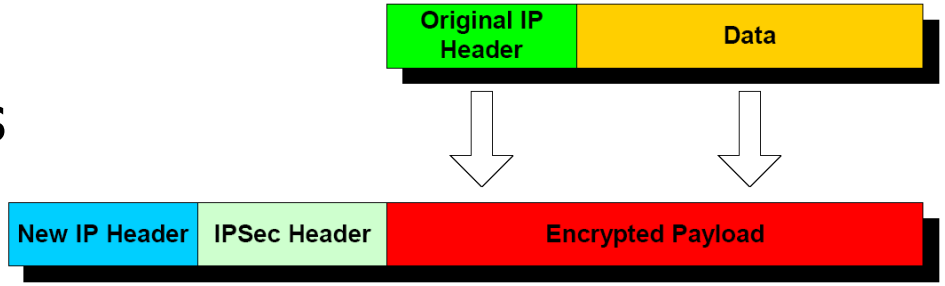
Transport mode



Tunnel mode

- Header IP Addresses are IPSec Gateway Addresses
- Host IP Address is not Revealed

Tunnel mode

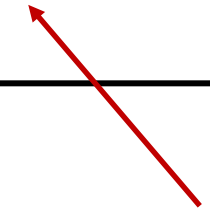


IPsec protocols

- Authentication protocol
  - Authentication Header (AH)
  - Does not encrypt messages
- Combined authentication/encryption protocol
  - Encapsulating Security Payload (ESP)
  - Provides message confidentiality (encryption) plus authentication
- Internet Key exchange protocol (IKE)
  - Negotiates security capabilities between two peers

Four combinations are possible!

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



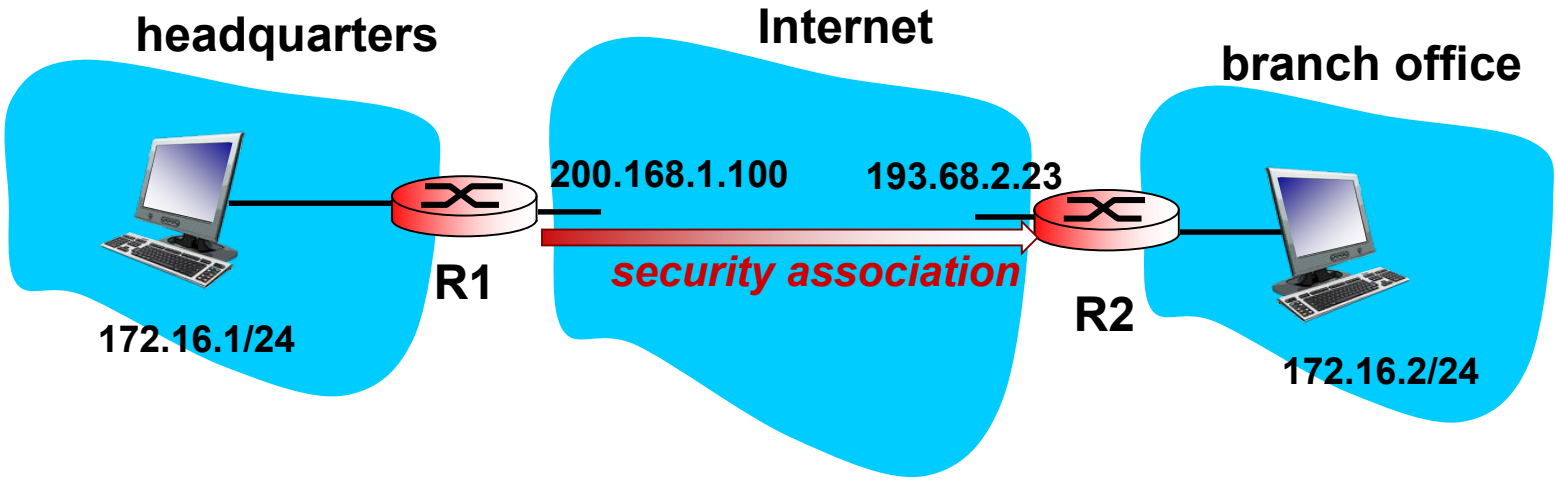
most common and most important

Security Associations (SA)

安全关联

- Before sending data, "security association (SA)" established from sending to receiving entity
  - The *SA* defines one-way relationships between sender and receiver
  - 2 *SAs* are normally required for full duplex communication
- Ending, receiving entitles maintain state information about SA
  - recall: TCP endpoints also maintain state info
  - IP is connectionless; IPsec is connection-oriented!
- The Security Parameters Index (SPI) tells under what *SA* a received packet be processed
  - Each host has a table containing the *SAs*
  - SPI is the index used to find the entry for a particular *SA*
  - The index is local for two peers (no global meanings)

Example SA from R1 to R2

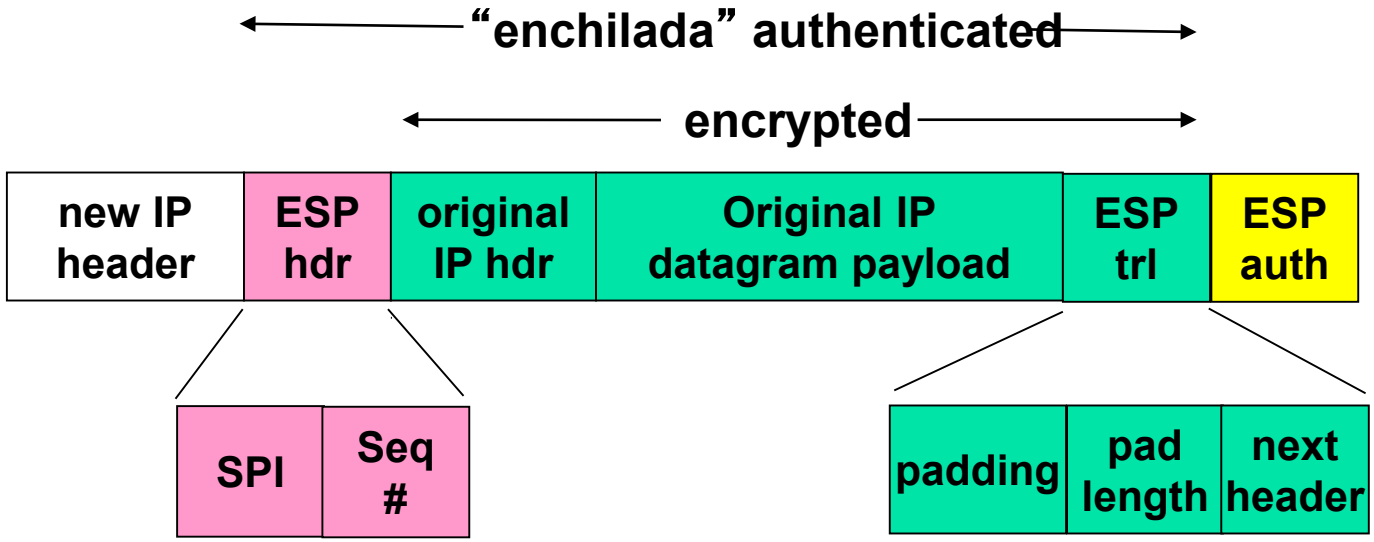


R1 stores for SA:

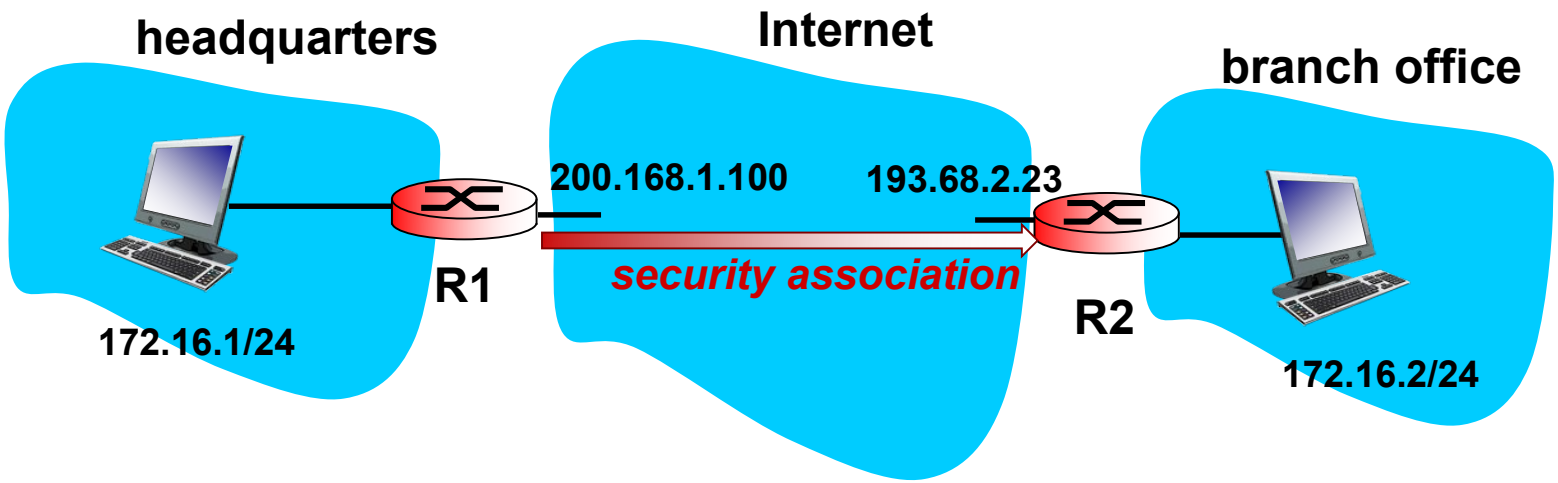
- 32-bit SA identifier: *Security Parameter Index (SPI)*
- origin SA interface (200.168.1.100)
- destination SA interface (193.68.2.23)
- type of encryption used (e.g., 3DES with CBC)
- encryption key
- type of integrity check used (e.g., HMAC with MD5)
- authentication key

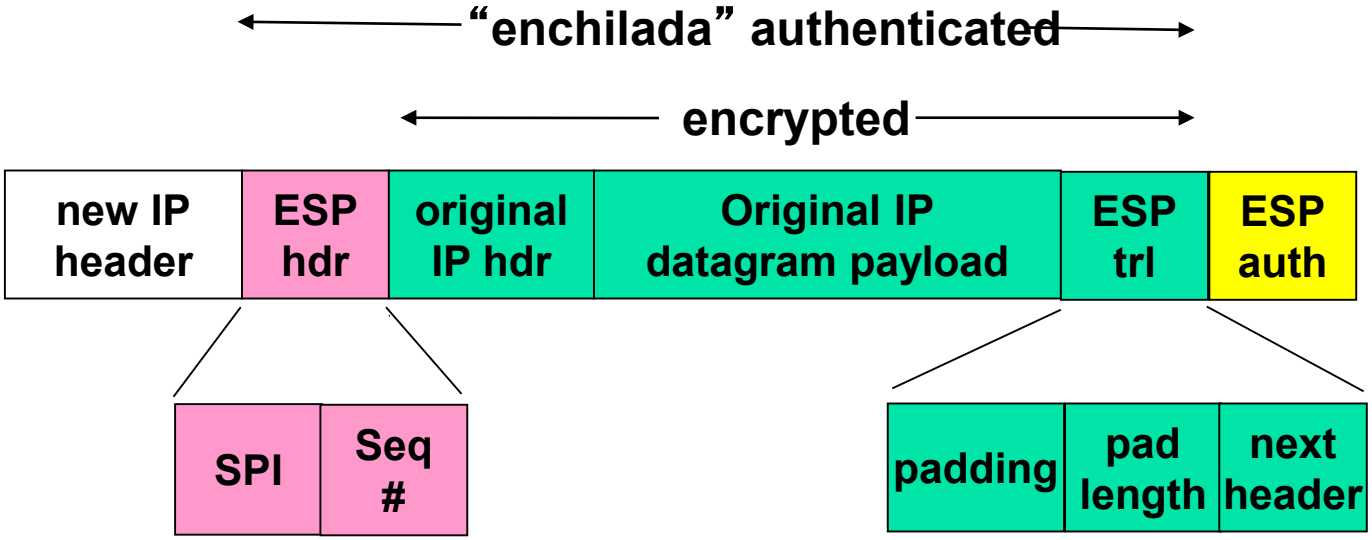
IPsec datagram

focus for now on tunnel mode with ESP



What happens?

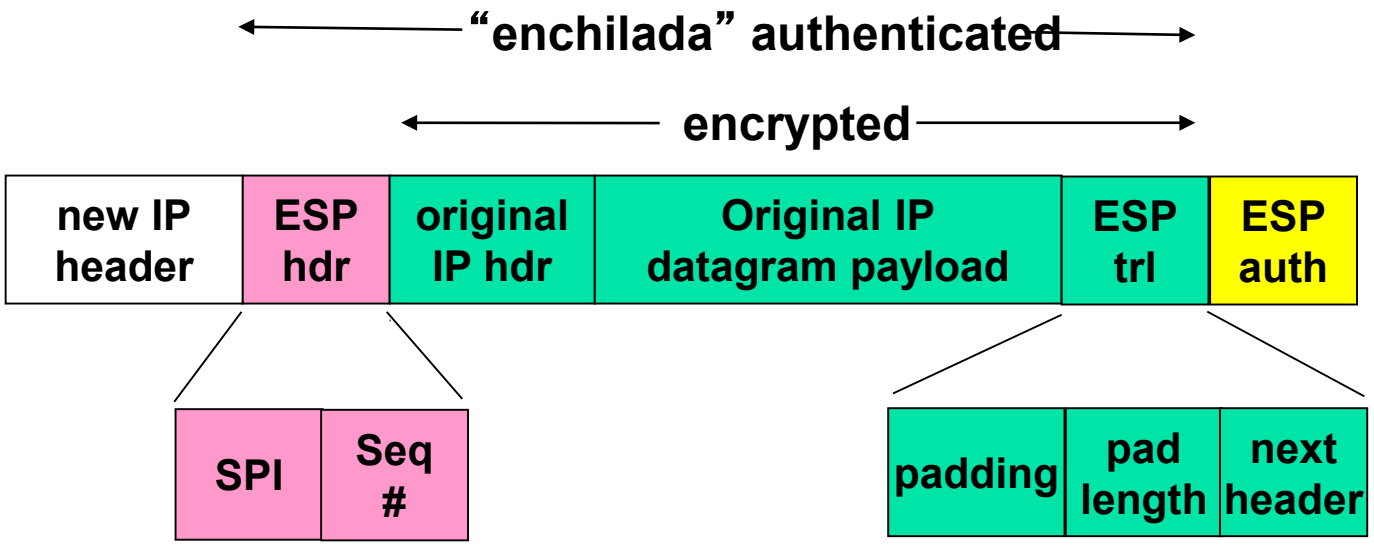




R1: convert original datagram to IPsec datagram

- 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

Attacking 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)*

Internet Key Exchange (IKE)

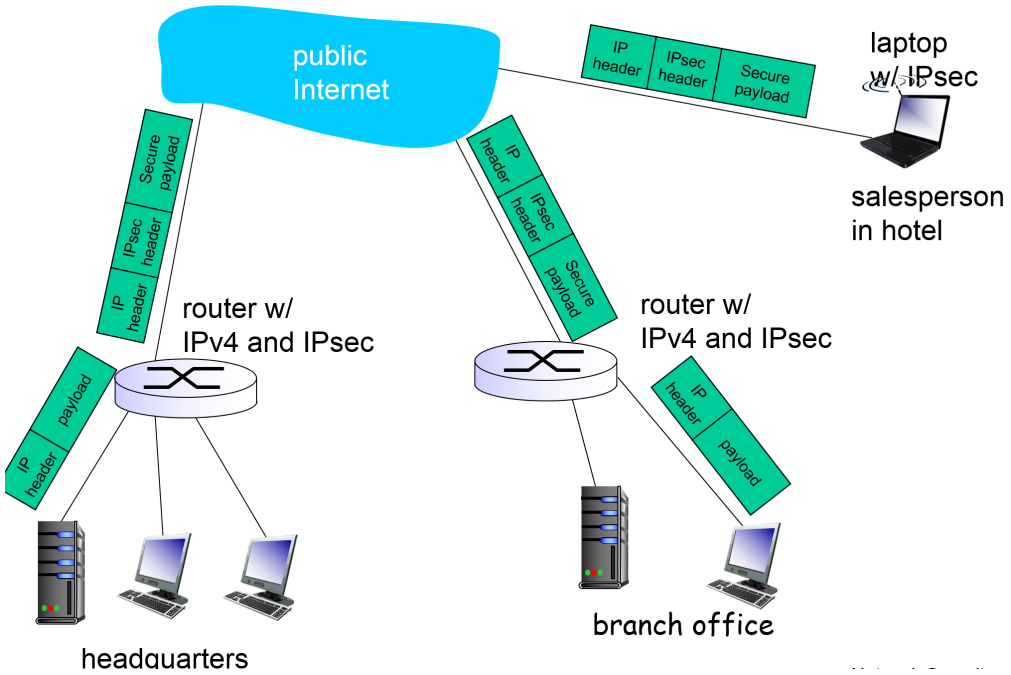
- Used to establish, modify and delete security associations (*SAs*)
- RFC 2409, based on
  - *ISAKMP* (Internet Security Association Key Management Protocol)
  - *Oakley* Key Generation Protocol
- *IKE* performs the following tasks
  - Agrees upon security algorithms
  - Authentication (Key-Hashed *MAC*)
  - Exchange of (symmetric) session crypto keys

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.

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



Securing Wireless LANs

- Securing 802.11
  - Authentication + Encryption
  - 1st attempt: Wired Equivalent Privacy (WEP), failed
  - Current attempt: 802.11i
- Wired Equivalent Privacy
  - Use shared key: 40-bit master key + 24-bit initialization vector (IV)
  - No key distribution mechanism, key set manually
  - Access point supposes only the mobile host has key

Wired Equivalent Privacy

Authentication

- Mobile host requests authentication from access point
- Access point sends back 128-bit nonce (against replay)
- Host encrypts nonce using shared master key K_S
- Access point decrypts nonce, authenticates host

Wired Equivalent Privacy

Encryption

- 40-bit K_S + 24-bit IV used to generate a stream of keys
  - Generator assures same key stream for similar 64-bit key
- Key stream XOR'ed with plaintext and checksum to produce cipher text
  - For each octet of msg data d_i: $c_i = k_i \oplus d_i$
  - For each octet of CRC crc_j: $c_{n+j} = k_{n+j} \oplus crc_j$
- IV and cipher text sent in frame

Attack WEP

- Security hole
  - 24-bit IV, one IV per frame -> IV's eventually reused
    - If assigned randomly, expected reuse once per 5000 frames
    - If assigned sequentially, reused at each startup
  - IV transmitted in plaintext -> IV reuse detected

- Attack is easy, since

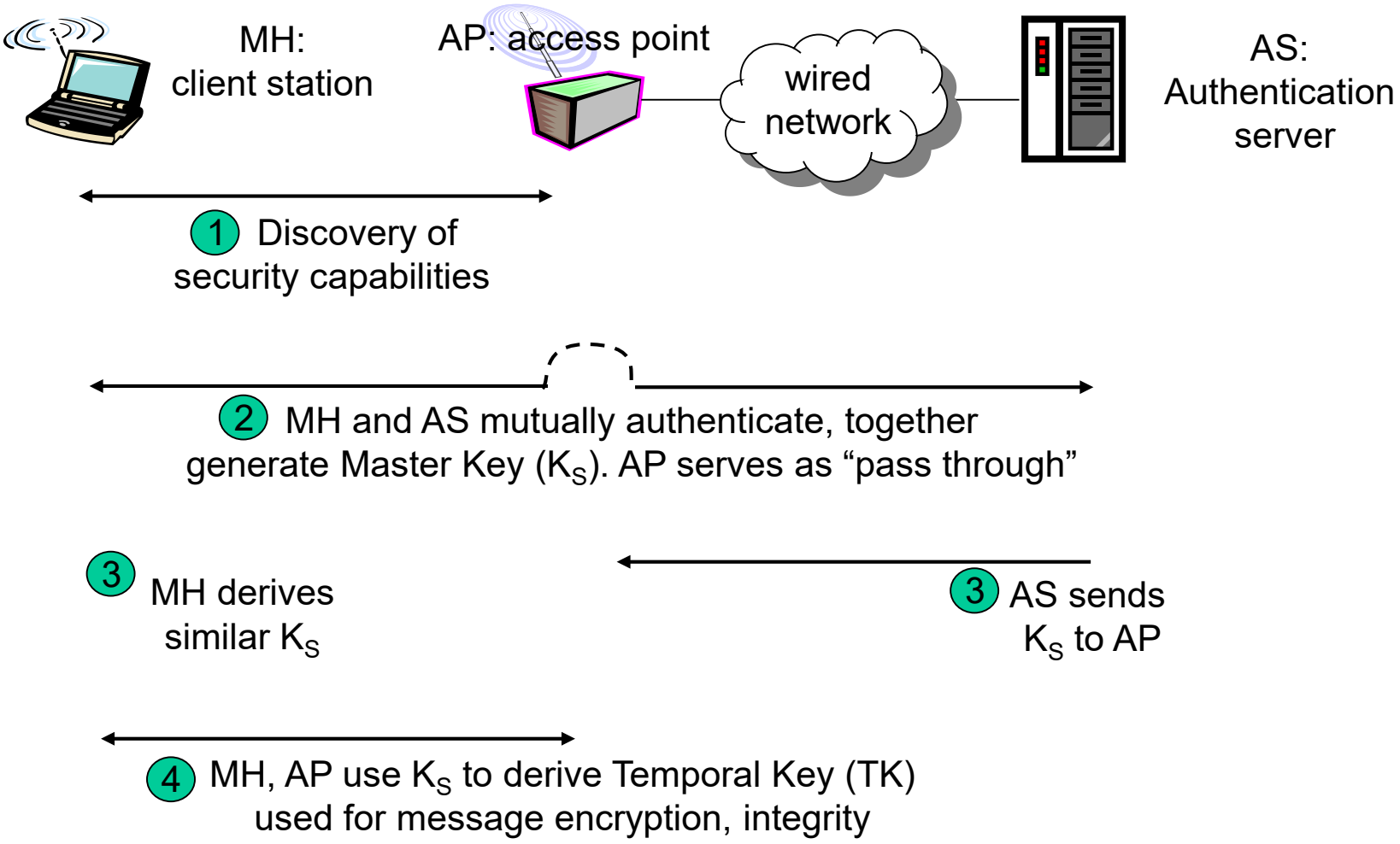
$$(P_1 \oplus C) \oplus (P_2 \oplus C) = P_1 \oplus P_2$$

  - Cipher text C is the same if IV reused
  - If Trudy causes Alice encrypt a known plain text P_1
  - P_2 will be known once the IV reappear

802.11i Improved Security

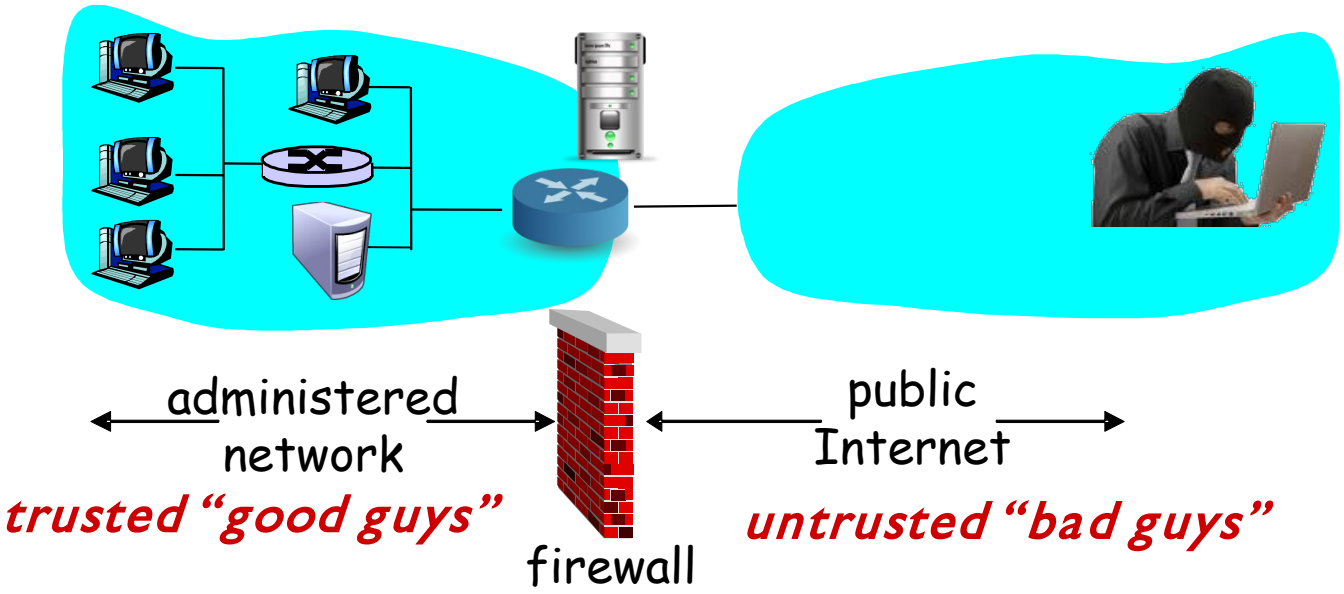
- Uses authentication server separate from access point
- Provides key distribution mechanism
- Numerous (stronger) forms of encryption possible

802.11i Procedure



Firewalls

- Isolate organization's intranet from larger Internet
  - Allowing some packets to pass, blocking others
- Ensure intranet/system security from hackers/malwares outside

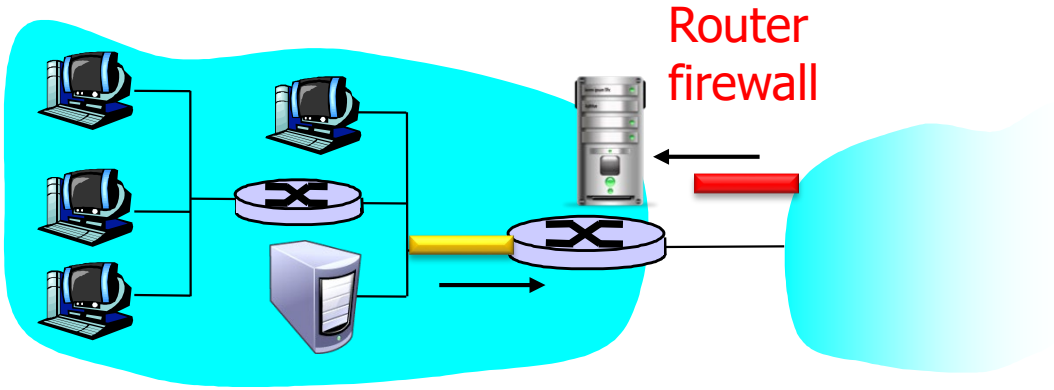


Functions of Firewalls

- Prevent denial of service attacks
  - *SYN flooding*, by preventing attackers from establishing bogus TCP connections / trying pings
- Allow only authorized access to inside network
  - Set of authenticated users/hosts
- Prevent illegal access/modification of internal data
  - Prevent access of specified servers/applications

- 3 types
  - Stateless packet filters
  - Stateful packet filters
  - Application gateways

Packet Filtering



- Check if arriving packet be allowed in, departing packet let out
- Router firewall 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

Filtering Example

- Block incoming/outgoing datagrams with IP protocol field = 17
  - All incoming, outgoing UDP flows are blocked
- Block incoming/outgoing datagrams with either source or dest port = 23
  - All telnet connections (bbs) are blocked
- Block incoming TCP segments with ACK bit=0
  - Prevents external clients from making TCP connections to internal hosts (i.e. DOS attacks)

ACK比特为0时，可能是SYN请求等建立TCP连接的控制报文

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 advertisements.
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

ACL: Access Control List

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

路由器可采用访问控制列表来实现防火墙规则

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): determine whether incoming, outgoing packets "makes sense"**
  - **timeout inactive connections at firewall: no longer admit packets**

A Stateful ACL

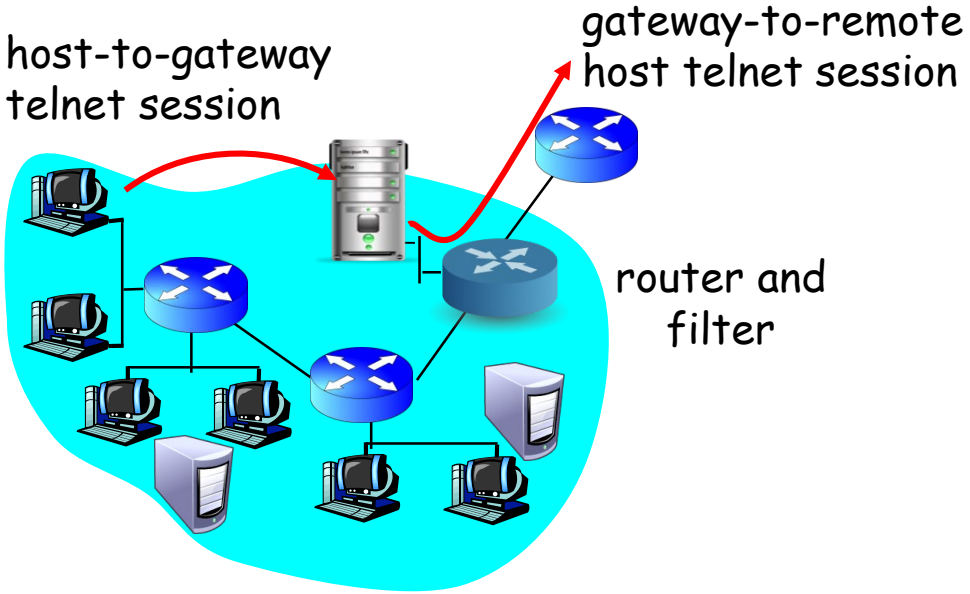
- ACL augmented to indicate need to check connection state table before admitting packet

- A stateful ACL

action	source address	dest address	proto	source port	dest port	flag bit	is closed
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	X
allow	222.22/16	outside of 222.22/16	UDP	> 1023	53	---	
allow	outside of 222.22/16	222.22/16	UDP	53	> 1023	----	X
deny	all	all	all	all	all	all	

More Advanced: Application Gateways

- Filters packets on application data as well as on IP/TCP/UDP fields
  - e.g. allow select internal users to telnet outside, but user authentication should be in application level
- Application Gateway
  - TCP connections must be relayed by gateway
    - Router filter blocks all TCP connections not originating from gateway



Firewall: In Conclusion

- 3 types
  - Stateless packet filters
  - Stateful packet filters
  - Application gateways
- Many things to do
  - *IP spoofing:* router can't know if data "really" comes from claimed source
  - Gateway is the most powerful, but not transparent (by proxy setting)
  - Limited functions for UDP communications
    - Filters often use all or nothing policy for UDP
  - Setting rules is always a step later
  - *Tradeoff:* degree of communication with outside world, level of security

Summary

- Security in different network layers
  - Transport Layer Security
  - IP Security
  - Securing Wireless LANs
- Firewalls
  - Stateless packet filters
  - Stateful packet filters
  - Application gateways

Homework

- Chapter 8：R23, P19