Information System Security

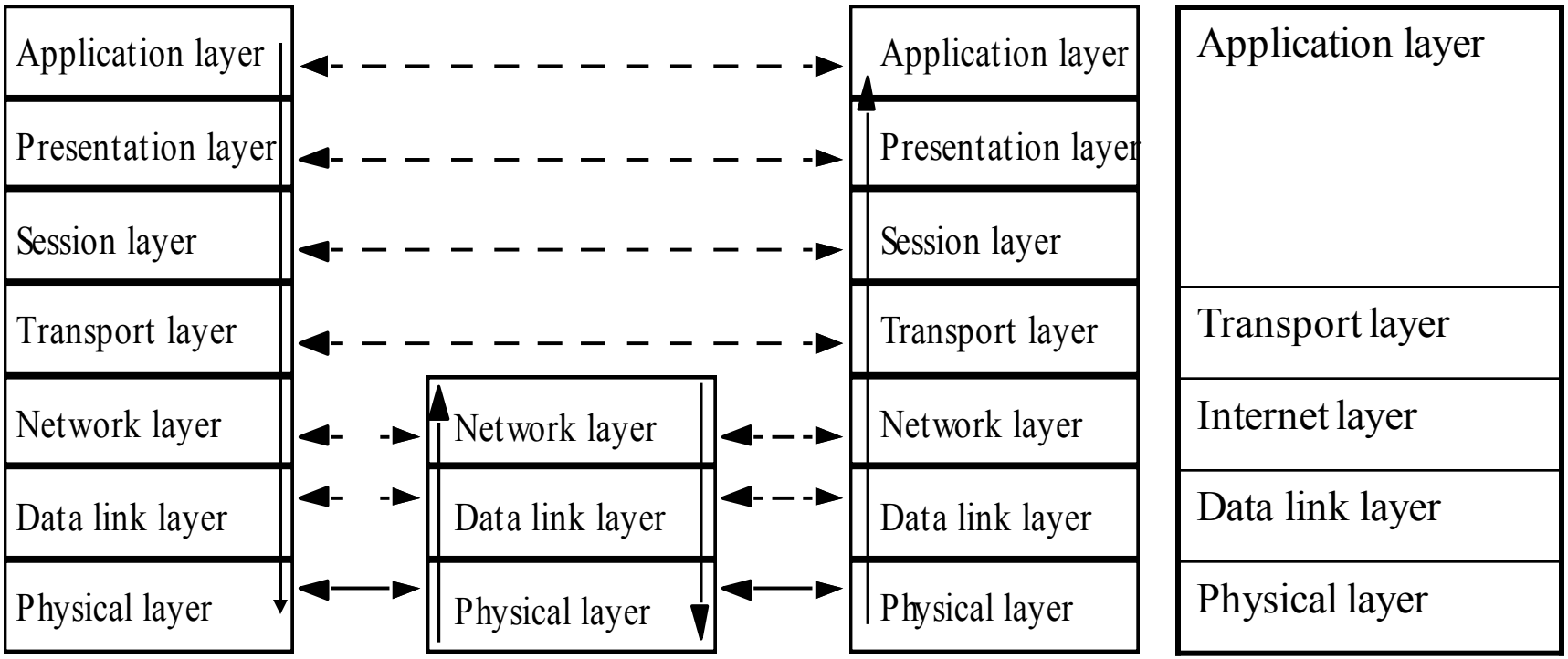
The OSI Network Model

ISO/OSI ***versus*** TCP/IP

Application layer	Application layer	HTTP, FTP, POP3, SMTP, SNMP, IMAP, IRC, SSH, Telnet, BitTorrent, … **PEM**
Presentation layer		
Session layer		
Transport layer	Transport layer	TCP, UDP, RTP… **SSL**
Network layer	Internet layer	IPv4, IPv6 … **IPSEC**
Data link layer	Data link layer	Ethernet, Wi-Fi, Token ring, FDDI,PPP…
Physical layer	Physical layer	RS-232, 10BASE-T, …

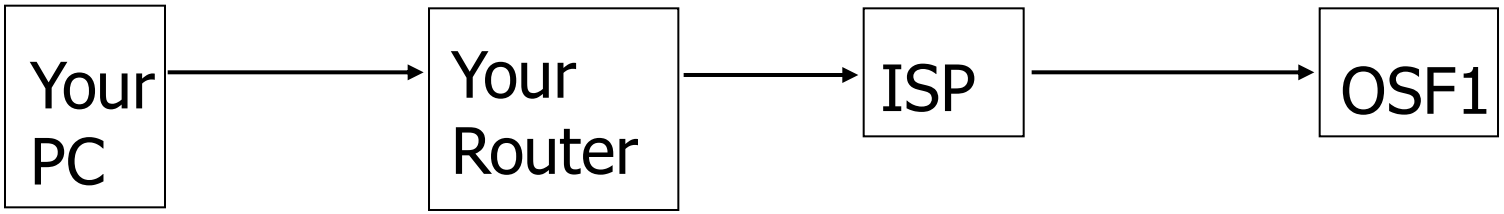
Network Model (Cont'd)

- ***Conceptually***, each host has a peer at each layer
  - Peers communicate with peers at the same layer

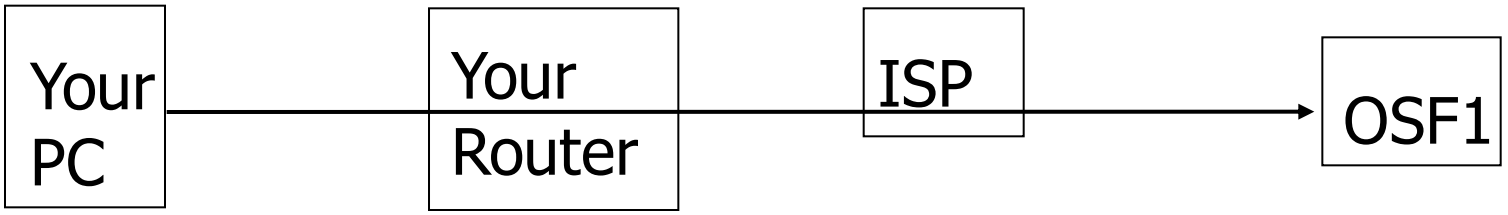


Link and End-to-End Protocols

Link Protocol (e.g., IP)



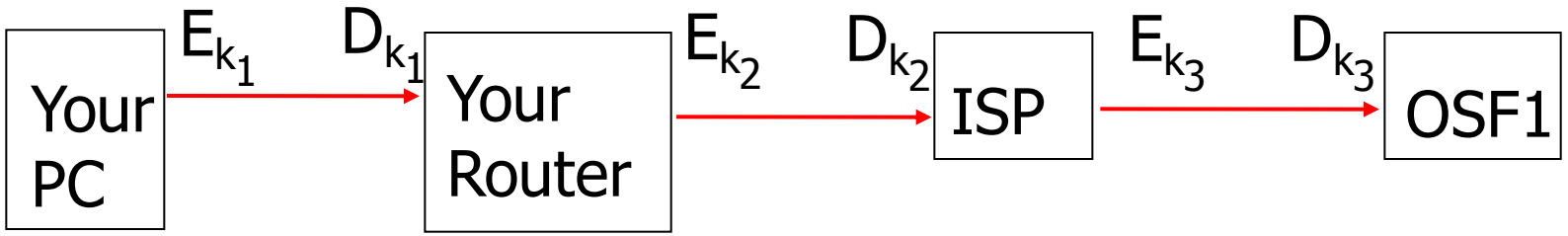
End-to-End Protocol (e.g., Telnet)



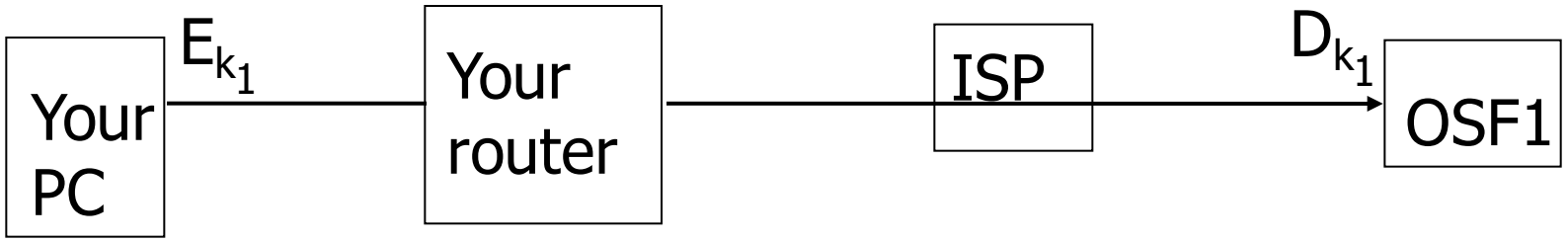
Link and End-to-End Encryption

Q: where is plaintext?

- Link encryption
  - Message is decrypted/re-encrypted at each intermediate host; e.g., PPP



- End-to-end encryption
  - Only hosts at two ends do encryption/decryption; transparent to intermediate hosts; e.g., SSL/SSH



Cryptographic Considerations

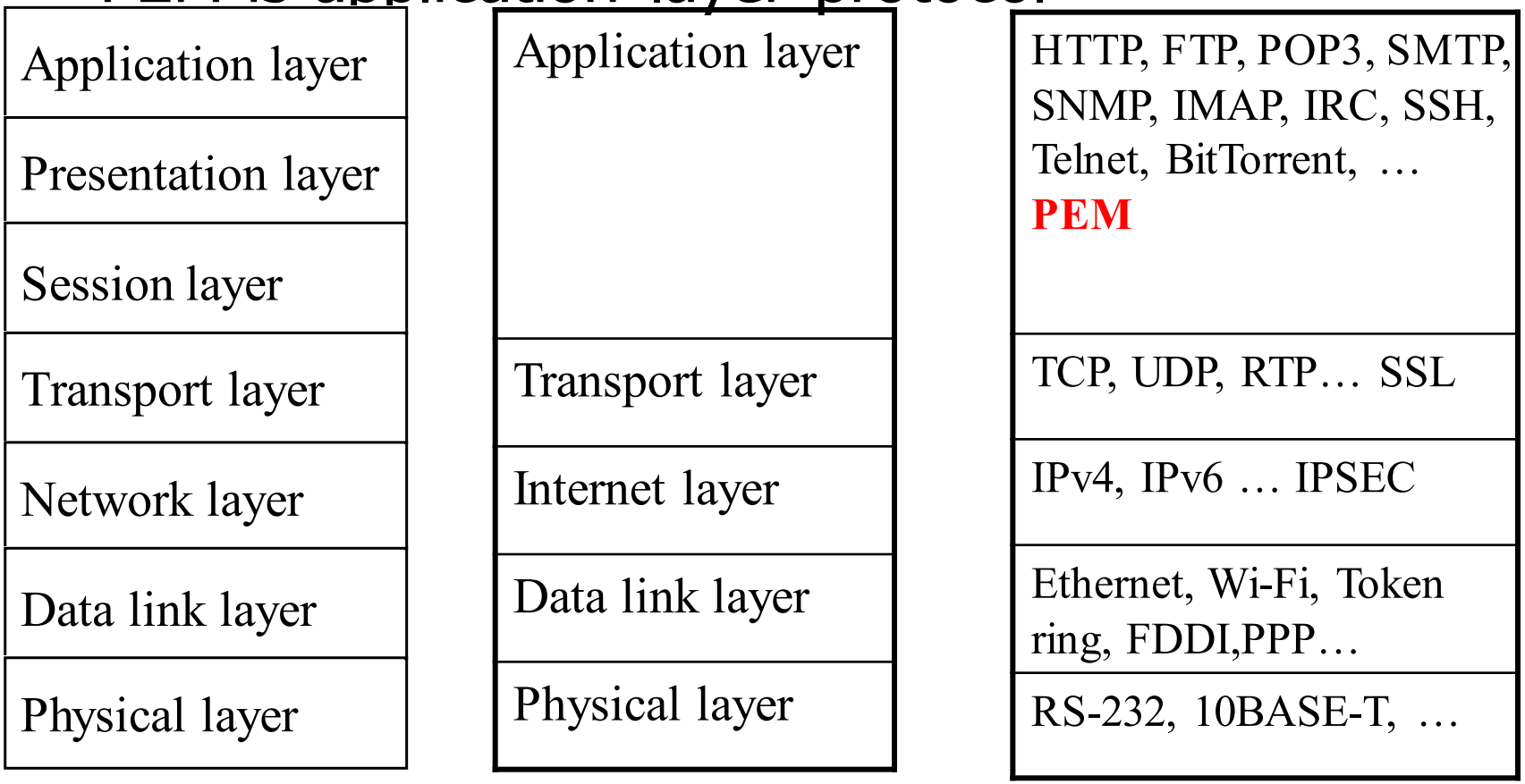
- Link encryption
  - Each host shares keys with neighbors
  - Message is read by intermediate nodes
  - Successful in military; infeasible for internet
- End-to-end
  - Only hosts at two ends need to share key
  - Message cannot be read at intermediate nodes
  - Widely used on internet (SSL/SSH)

Traffic Analysis

- The mere existence of traffic (at a certain time, between certain hosts) reveals information
- Link encryption
  - Can protect headers of packets
  - Can hide source and destination by mixing concurrent traffic
- End-to-end encryption
  - Cannot hide routing information in packet headers
    - Intermediate nodes need to route packet
  - Can easily identify source and destination

Privacy-Enhanced Electronic Mail

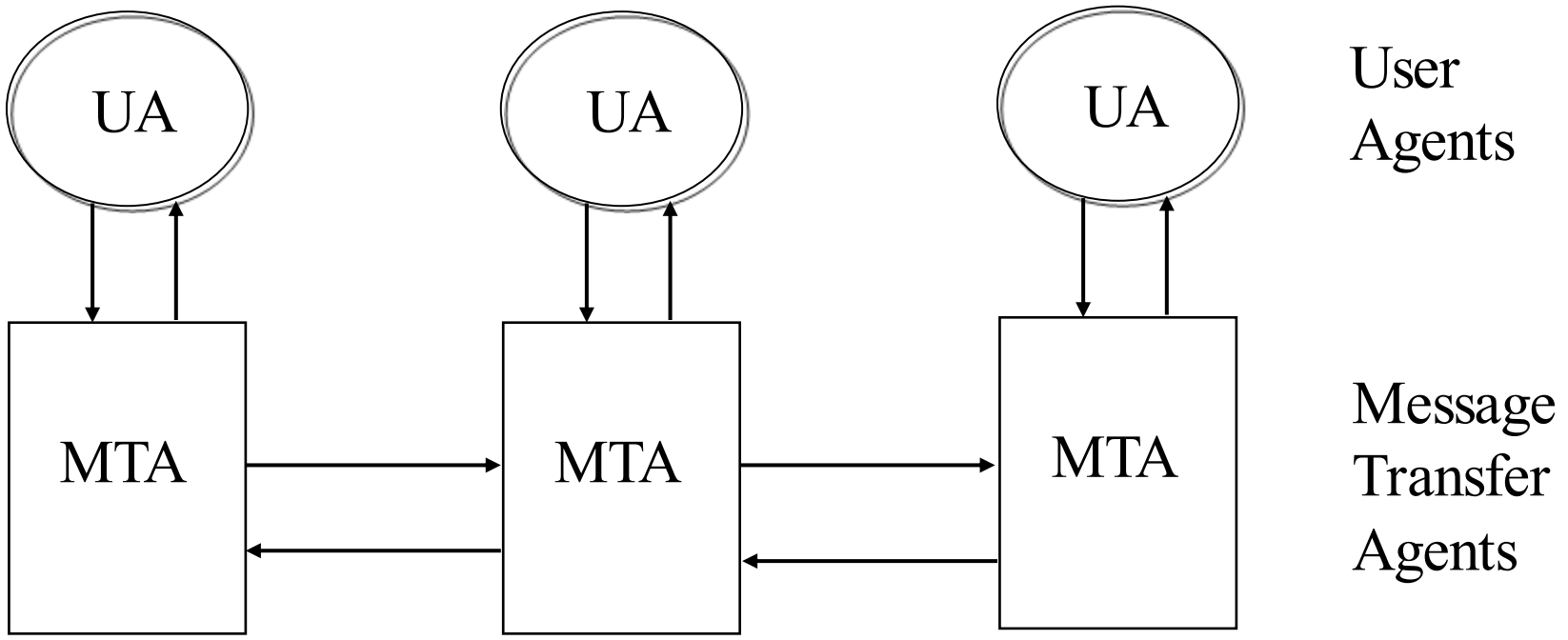
- PEM is application layer protocol



Goals

1. Confidentiality
   - Only sender and recipient(s) can read message
2. Origin authentication
   - Identify the sender precisely
3. Data integrity
   - Any changes in message are easy to detect
4. Non-repudiation of origin
   - Whenever possible …

Message Handling System



Design Principles

- Do not change related existing protocols
  - Cannot alter SMTP
- Do not change existing software
  - Need compatibility with existing software
- Make the use of PEM optional
  - Available if desired, but email still works without PEM
  - Can use part of the features (e.g., authentication only)
- Enable communication without prearrangement
  - Out-of-bands authentication, key exchange problematic

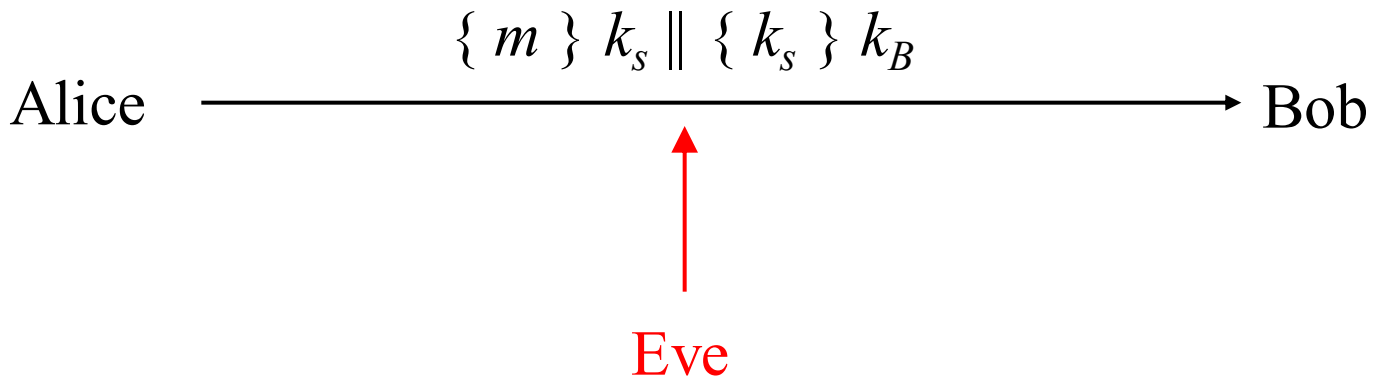
Basic Design: Keys

- Two keys
  - *Interchange keys* tied to sender, recipients and is static (for some set of messages)
    - Like a public/private key pair
    - Must be available *before* messages sent
  - *Data exchange keys* generated for each message
    - Like a session key, session being the message

Confidentiality

Confidentiality

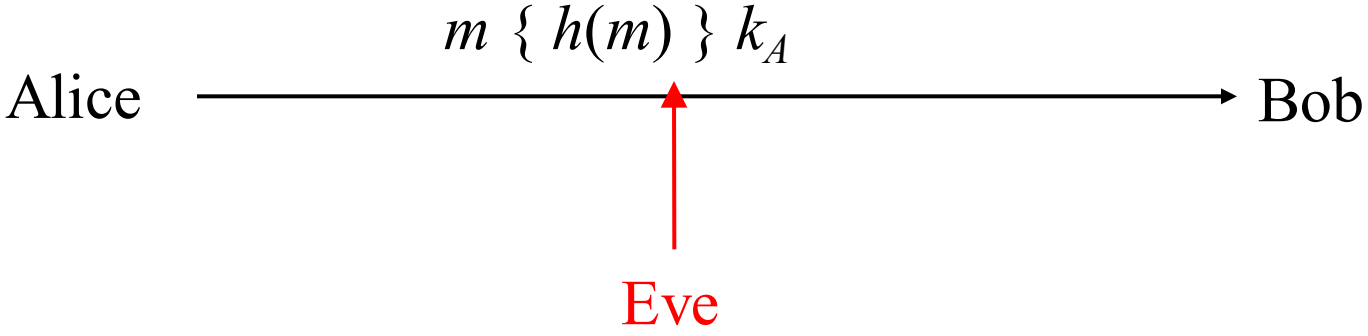
- m : message
- k_s : data exchange key
- k_B : Bob's interchange key



Integrity

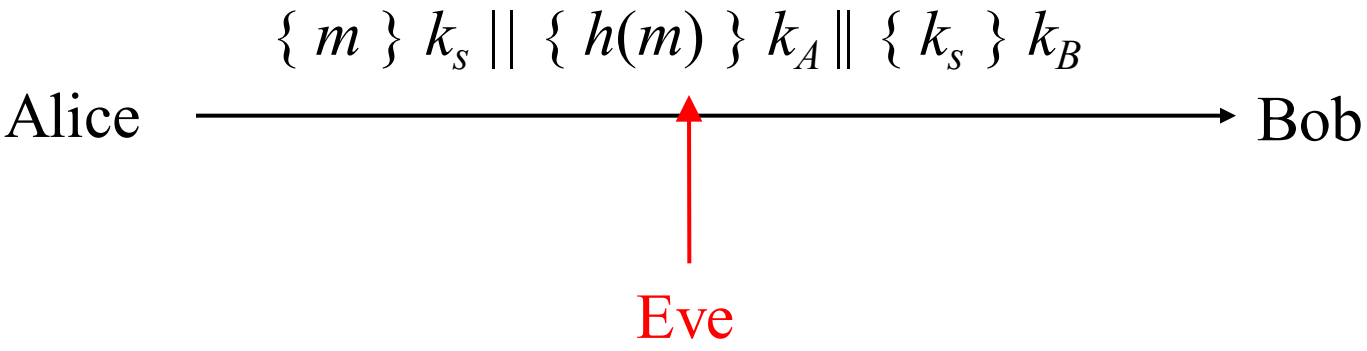
Data integrity, authentication, and non-repudiation

- m : message
- $h(m)$: hash of message m —Message Integrity Check (MIC)
- k_A : Alice's interchange key



Put It Together

Confidentiality and integrity:



Replay?

Problem

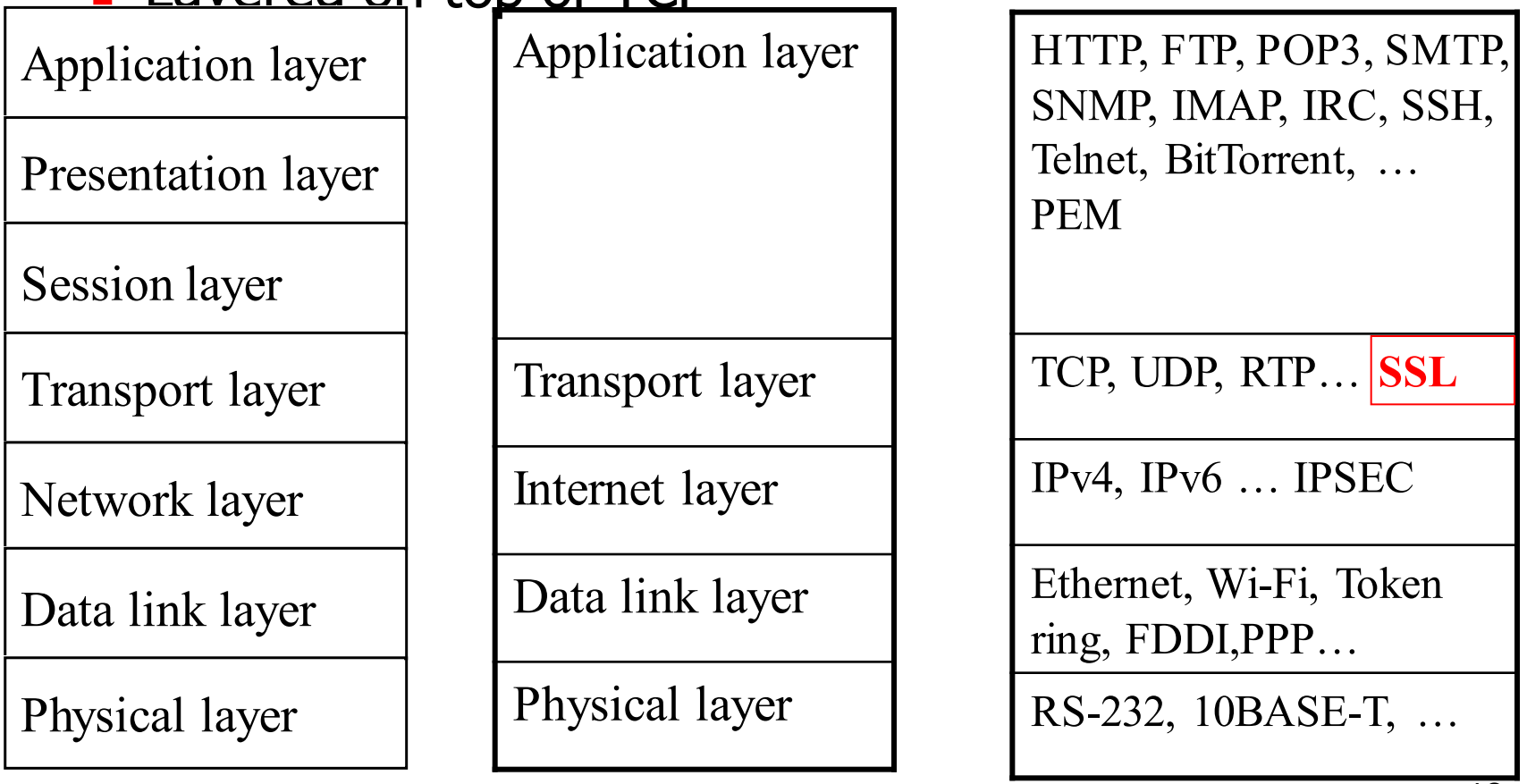
- Recipients without PEM-compliant software cannot read
  - If only the integrity part is used, they should be able to read it
  - Mode MIC-CLEAR allows this
- Hard to get certificates
  - How hard? Take hours
  - What does it promise? Email validity
  - I wait for *that* ????

Other Secure Email Protocols

- MIME Object Security Services (MOSS)
  - Supersedes PEM
- PGP/OpenPGP
  - Has most users
  - But not many
- S-MIME
  - Designed by RSA
  - Integrated in Outlook, Outlook Express, Netscape, but almost totally unused

Background

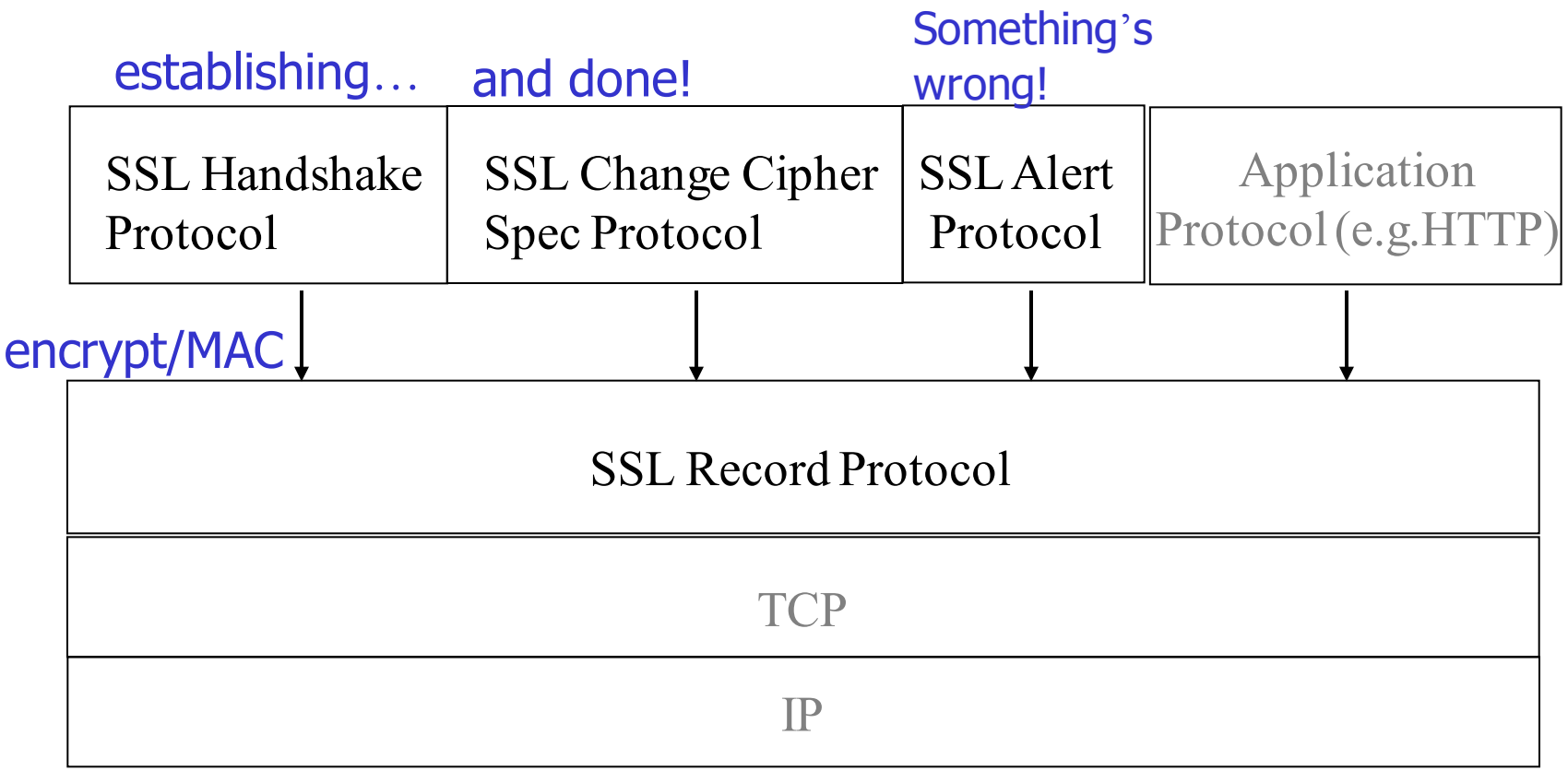
- SSL(Secure Sockets Layer) is at transport layer
  - Layered on top of TCP



Background (Cont'd)

- Developed by Netscape
  - SSL3.0 becomes IETF standard TLS (Transport layer security) http://www.ietf.org/html.charters/tls-charter.html
- Independent of application protocols
  - E.g., HTTPS, LDAP, POP3, etc.
- Provides:
  - Confidentiality and integrity of data
  - Authentication of two ends
    - Mostly for authentication of server only
    - Authentication of client: MSN Wallet, VerifyByVISA, etc.

SSL Protocol Stack



Before we zoom on each of them, we consider two things

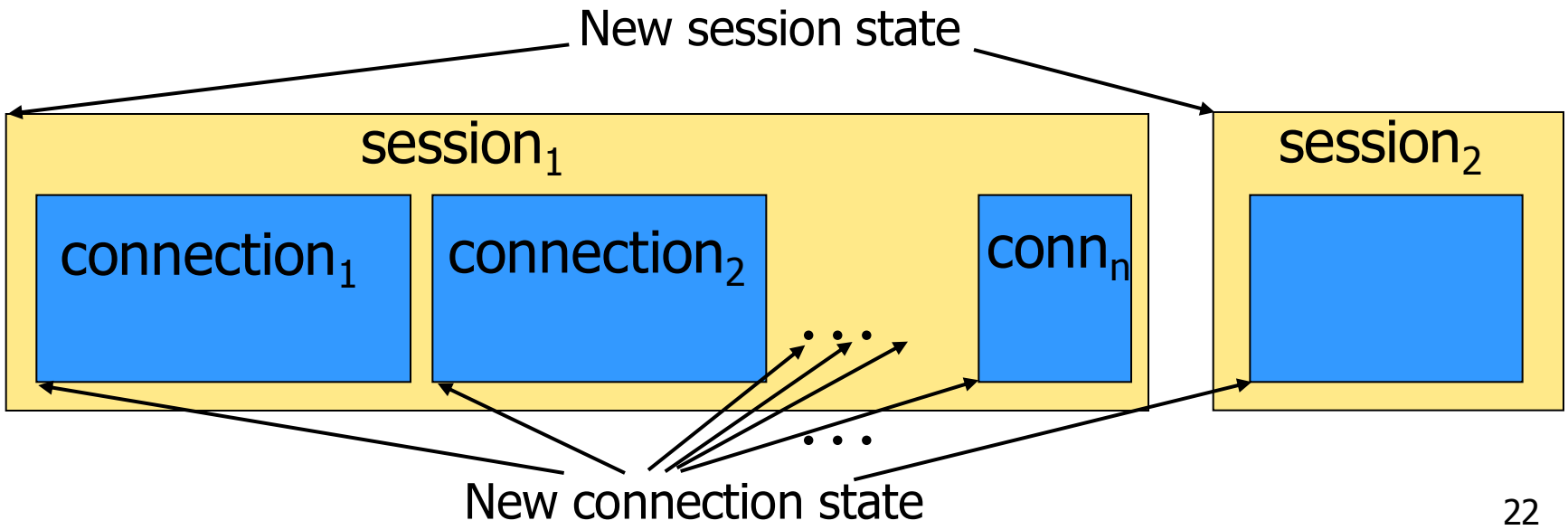
1. How to characterize an SSL connection (i.e., SSL parameters)
2. What cipher techniques can be used

SSL Parameters

- SSL parameters are divided into two sets:
  - Session states
    - Session identifier: generated by the server
    - Peer certificate: X.509 certificate of the peer
    - Compression method: compression prior to encryption
    - CipherSpec: data encryption algorithm and hash algorithm
    - Master secret: a 48 Byte shared secret used to derive keys
    - "is resumable" flag: whether ok to initiate new connections
  - Connection states
    - Server and client random: nonce generated by client and server
    - Server write MAC secret: the MAC key of server (client also uses it)
    - Client write MAC secret: the MAC key of client
    - Server write key: the encryption key of server
    - Client write key: the encryption key of client
    - Sequence number: maintained by server for identifying messages

SSL Session and Connection (Cont'd)

- Why two separate terms?
  - So the two sets of parameters can change independently
  - Session states change less frequently (for performance)
  - Connection states change more frequently (for security)
  - One session (re-used by) multiple connections

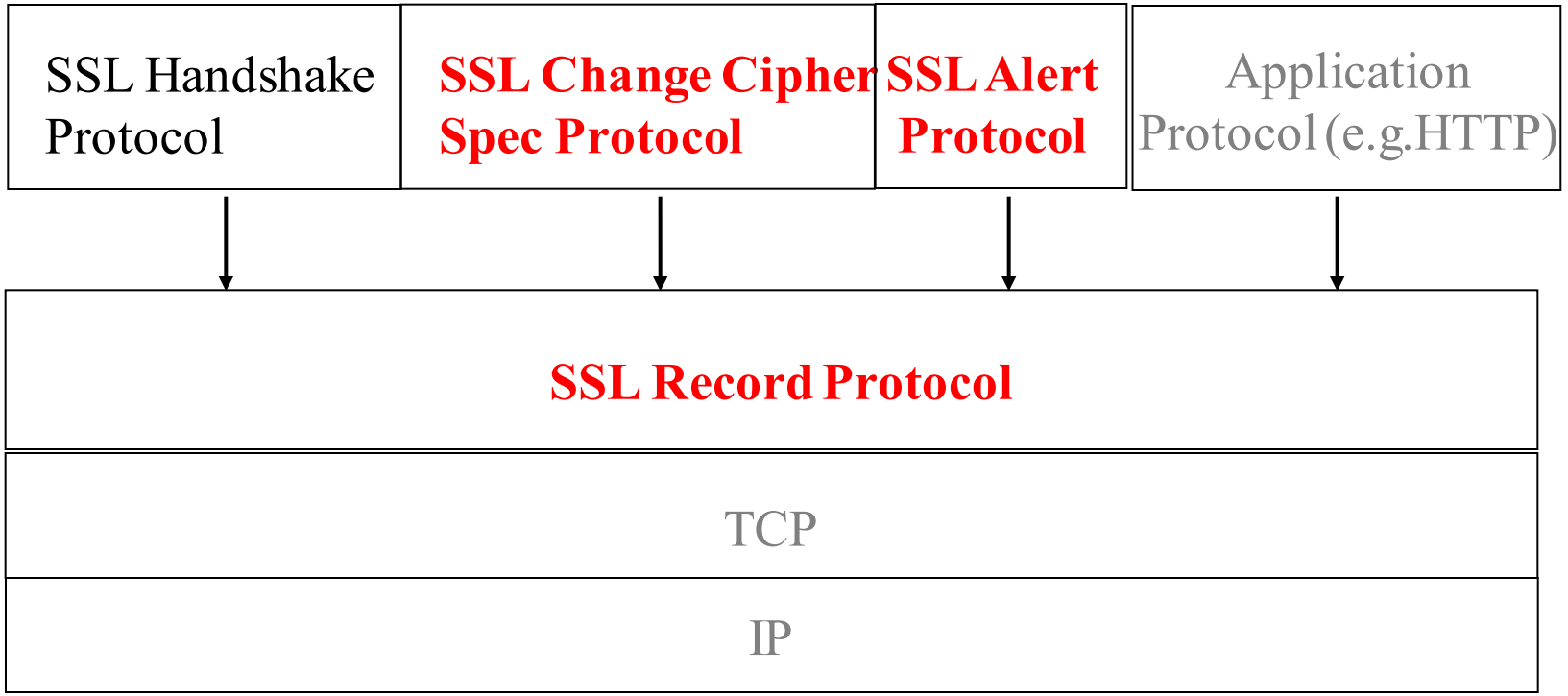


CipherSpec Overview

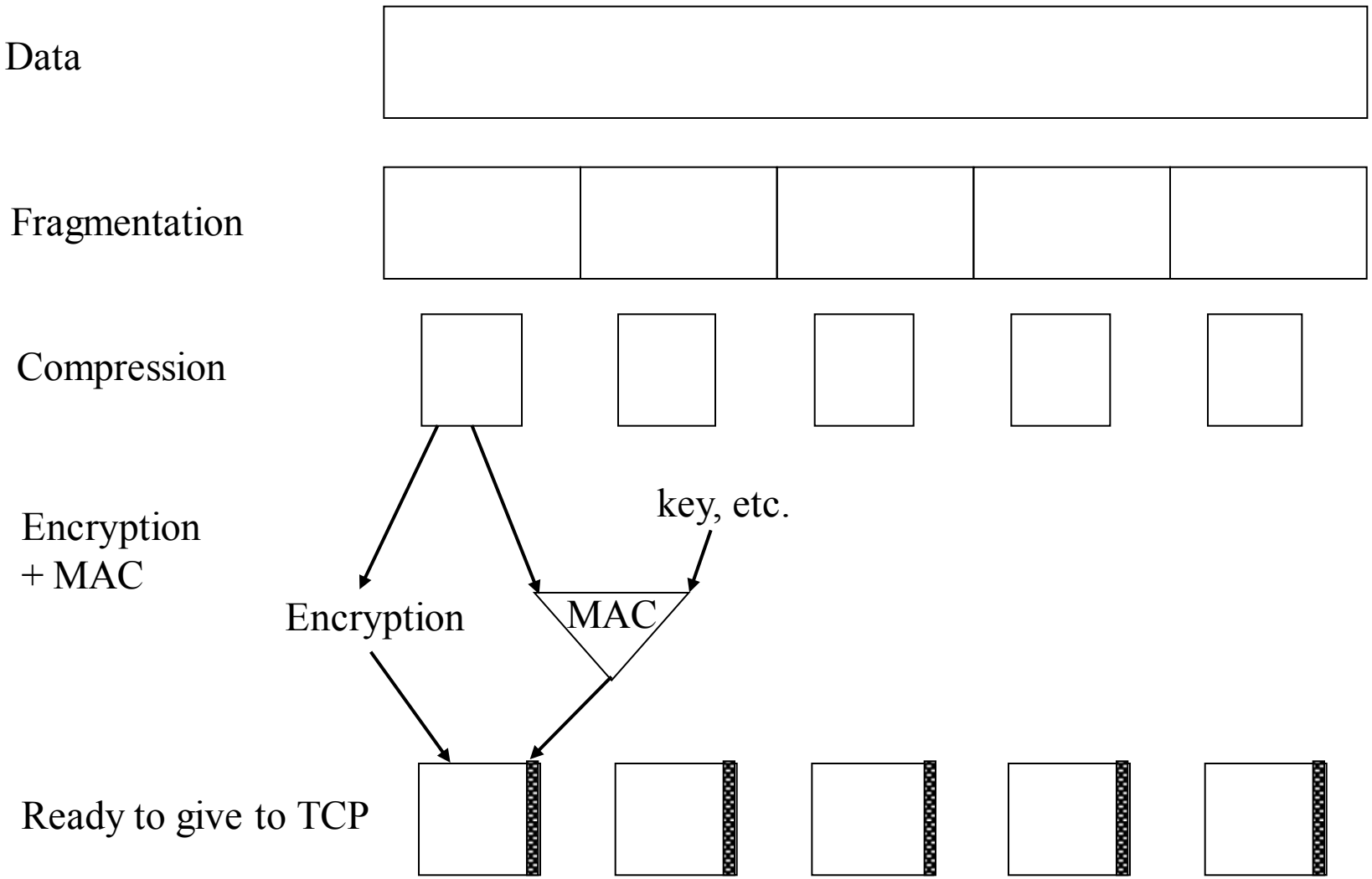
- Key exchanges
  - RSA, Diffie-Hellman, Fortezza (DoD)
- Encryption
  - RC2, RC4, IDEA, DES (CBC or 2-encryption mode)
- Hash function
  - MD5, SHA1
- Digital signalture
  - RSA
- Only certain combinations of those are allowed

Now we are ready to discuss each of the protocols

The Straightforward Ones



SSL Record Protocol



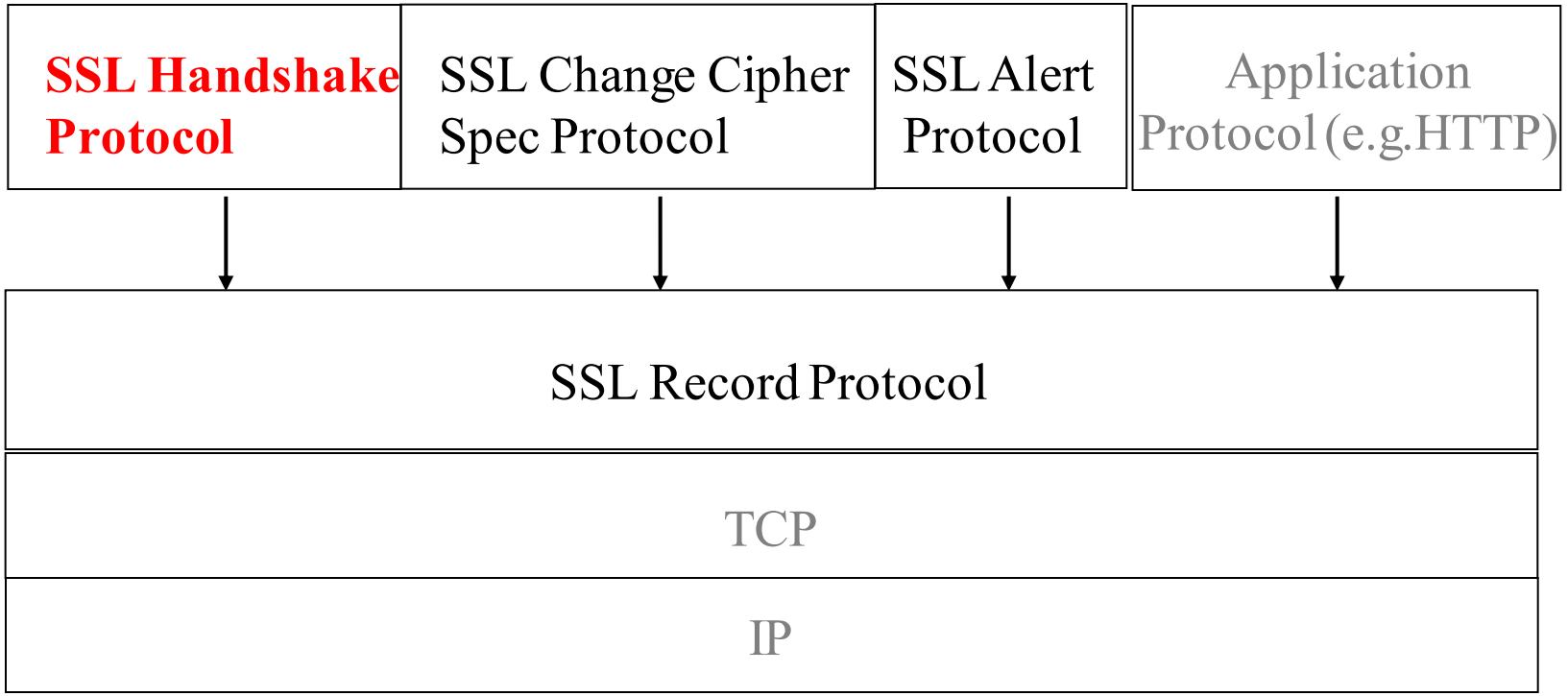
SSL Change Cipher Spec Protocol

- Following handshake protocol
- Sending single byte of message with value 1
- Signals the conclusion of handshake
  - "Let's switch to new parameters now!"

SSL Alert Protocol

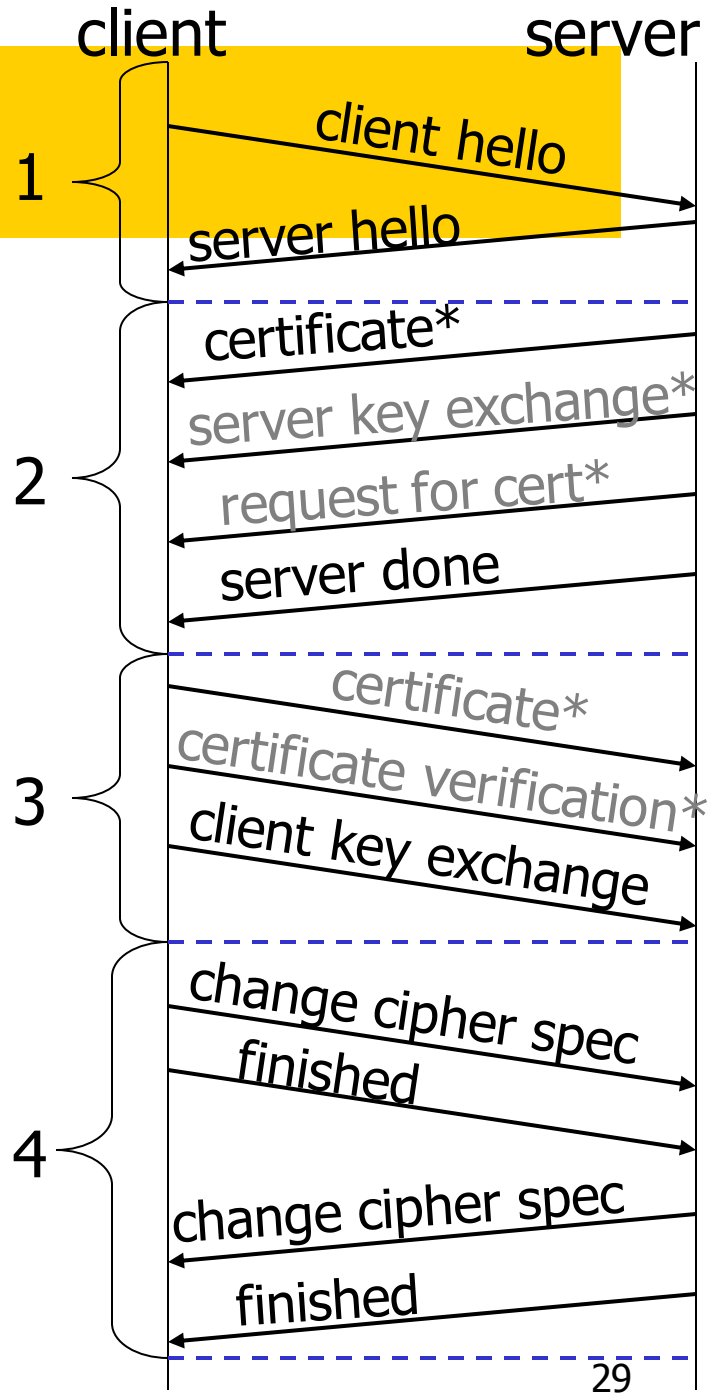
- Each message consists of two bytes
  - The first byte takes either “warning” (1) or “fatal” (2), which determines the severity of the message sent
  - The next byte of the message contains one of the defined error codes
- A ‘fatal’ message results in an immediate termination of the SSL session
  - E.g., unexpected_message, bad_record_mac, decompression_failure, handshake_failure, illegal_parameter

The Complicated One



Overview 1

1. Negotiate security capabilities between client, server
2. Server authenticates itself and key exchange
3. Client validates server and key exchange
4. Finish and acknowledgement



We shall only consider 1-way handshake with RSA (only server authenticates itself to client)

* Indicate optional or situation-dependent messages that are not always sent

Handshake Round 1

Hey, here's my chosen parameters and my capabilities

$$\{ v_C \parallel r_1 \parallel s_1 \parallel ciphers \parallel comps \}$$

Client ⟶ Server

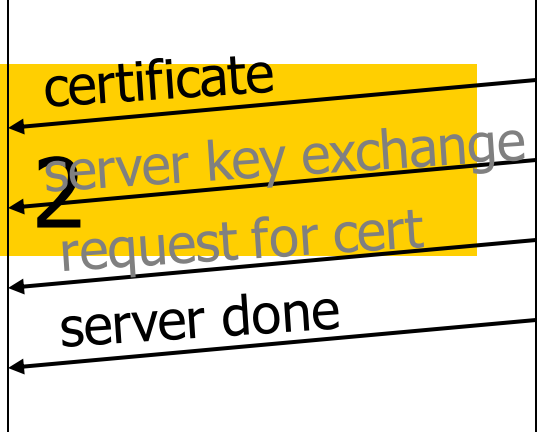
$$\{ v \parallel r_2 \parallel s_2 \parallel cipher \parallel comp \}$$

Client ⟵ Server

Alright, here's my chosen parameters, and what we should use (based on what we have in common)

v_C	Client's version of SSL
v	Highest version of SSL that Client, Server both understand
r_1, r_2	nonces (timestamp and 28 random bytes)
s_1	Current session id (0 if new session)
s_2	Current session id (if s1 = 0, new session id)
ciphers	Ciphers that client understands
comps	Compression algorithms that client understand
cipher	Cipher to be used
comp	Compression algorithm to be used

Handshake Round 2



Here's my X.509v3 certificate

Client ← $\{certificate\}$ — Server

Client ← $\{er2\}$ — Server

I'm done for this round

k_S Server's private key

$er2$ End round 2 message

Handshake Round 3

Here's a random secret I have chosen

Client $\xrightarrow{\{pre\}e_s}$ Server

pre a 48-bit random value generated by client

e_s server's public key (in its certificate)

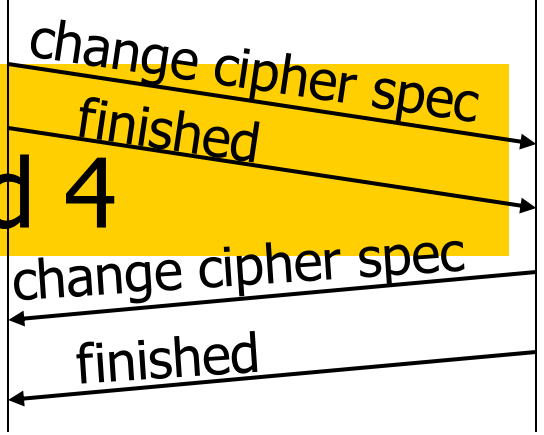
After the message, both client and server compute the *master secret*:

master = MD5(*pre* || SHA('A' || *pre* || r_1 || r_2) ||
MD5(*pre* || SHA('BB' || *pre* || r_1 || r_2) ||
MD5(*pre* || SHA('CCC' || *pre* || r_1 || r_2)

And derive four keys (MAC+encryption) from the master secret

The server can compute this only if he has the private key corresponding to e_s

Handshake Round 4



Handshake done for me. I will start using the new cipher parameters

Client —— "change cipher spec" ——→ Server

Let me prove that I have the master secret and I know all the previous rounds

$\{ h(master \,||\, opad \,||\, h(msgs \,||\, 0x434C4E54 \,||\, master \,||\, ipad\,)) \}$

Client ——→ Server

Handshake done for me. I will start using the new cipher parameters

Client ←—— "change cipher spec" —— Server

Let me prove that I have the master secret and I know all the previous rounds

$\{ h(master \,||\, opad \,||\, h(msgs \,||\, master \,|\, ipad)) \}$

Client ←—— Server

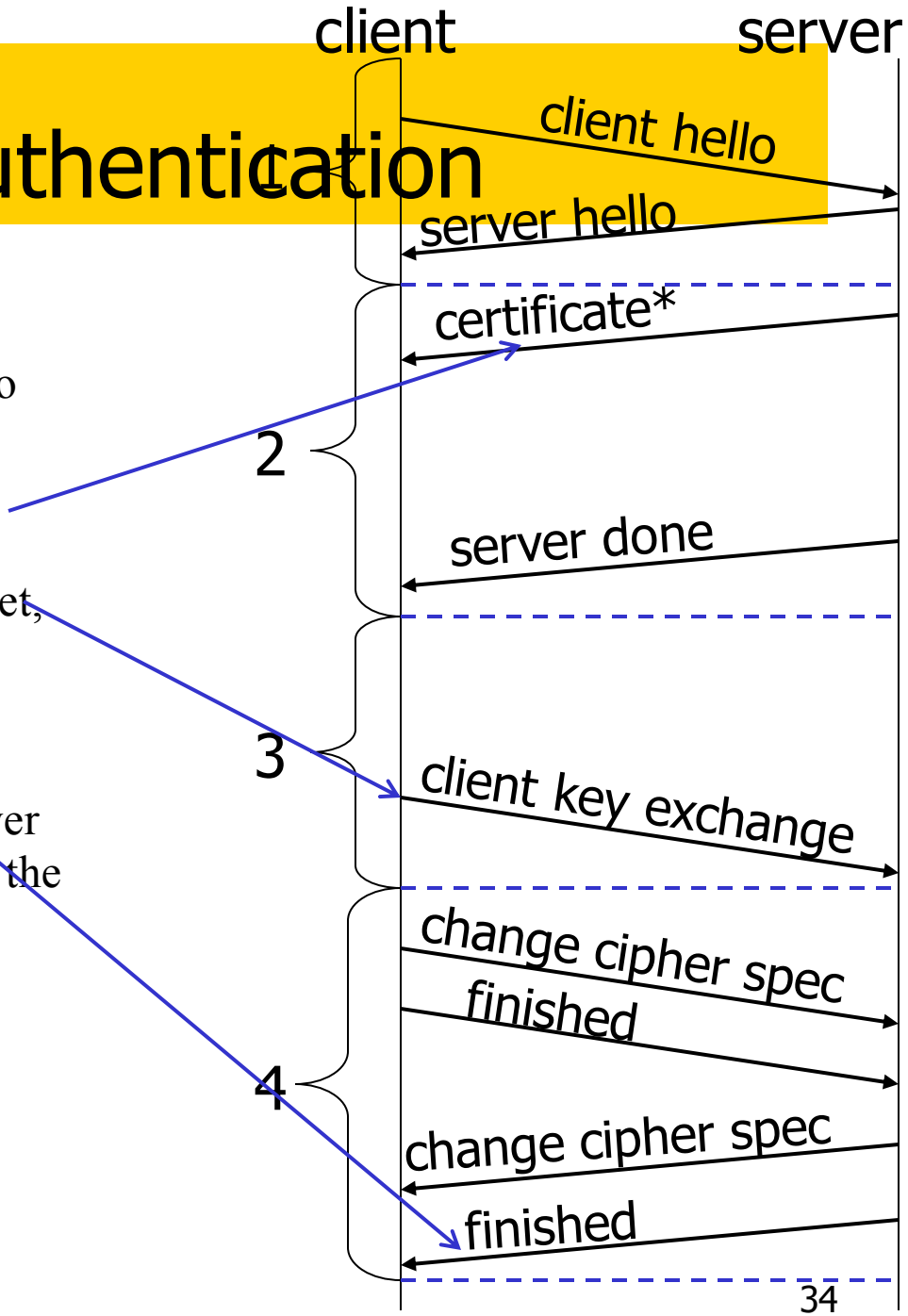
msgs	Concatenation of messages sent/received in *previous* rounds (does not include the messages in the current round)
opad, *ipad*	fixed padding from HMAC

Server Authentication

Why should the client believe he is talking to the server?

1. The server can decrypt the 'client key exchange' and compute the master secret, only if he has the private key corresponding to his certificate.

2. The 'finished' message proves that server has the master secret, and hence he has the private key.

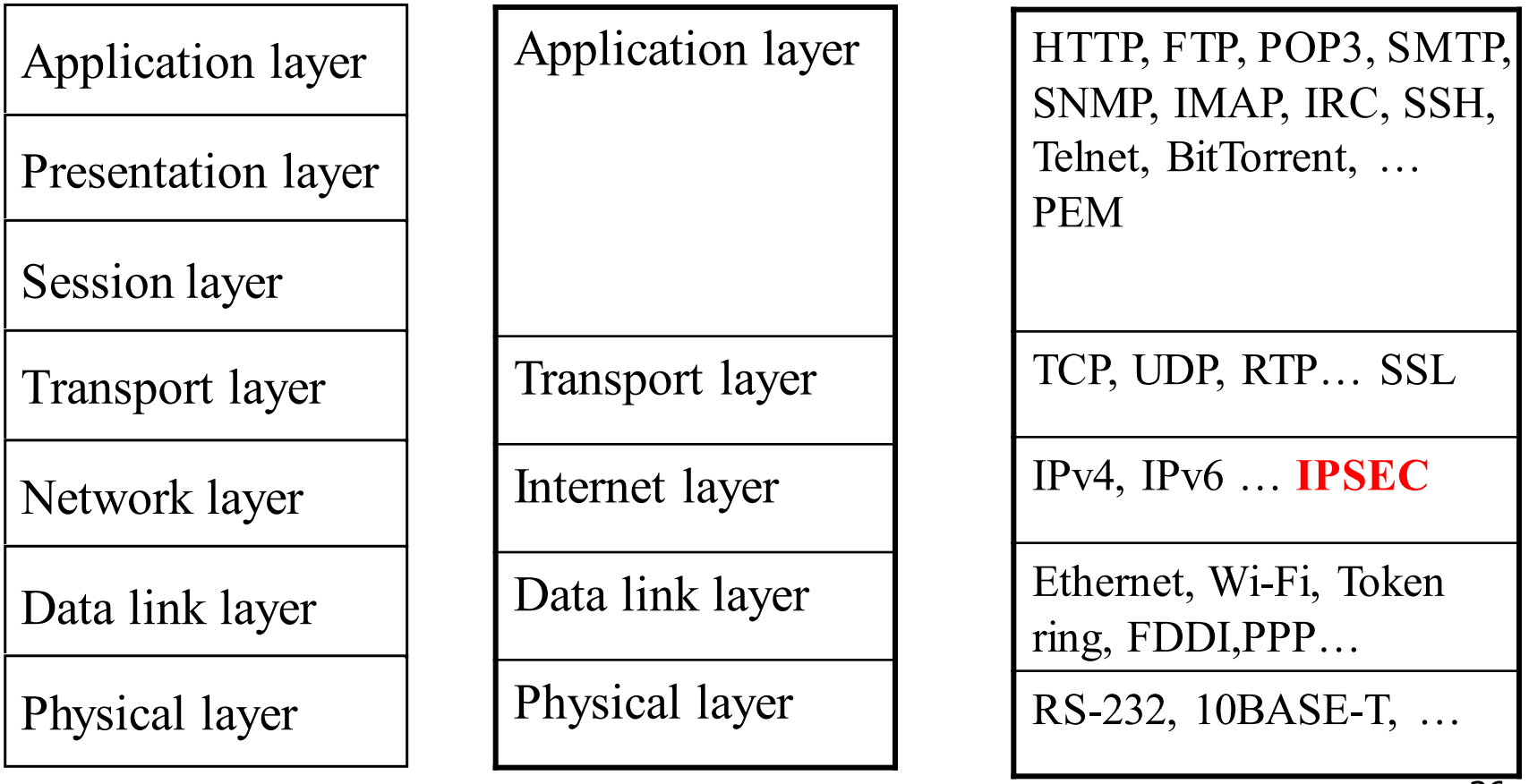


Overview

- Background
- PEM
- SSL
- IPSEC

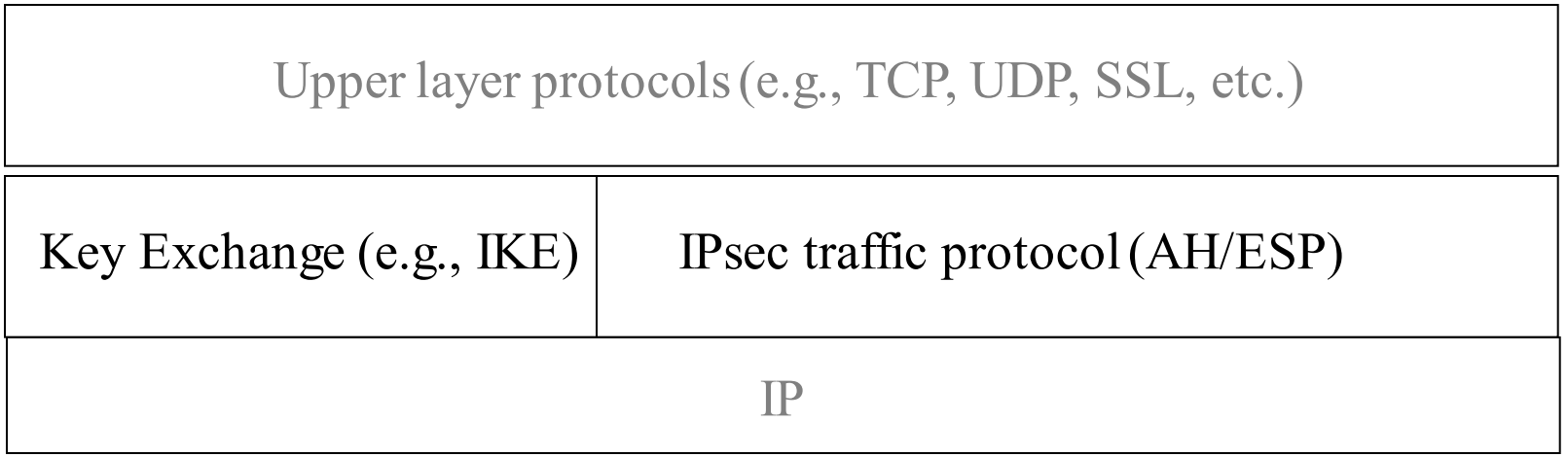
Background

- IPsec (IP Security) is at network layer



IPsec Overview

- Security Association
- Transport mode and tunnel mode
- Traffic protocols
  - IP AH (Authentication header) protocol
  - IP ESP (Encapsulating security protocol)
- Key exchange protocol
  - IKE



Security Association Overview

- Security Association (SA)
  - A logical association between peers for security services
    - Like session/connection of SSL
  - Can be established by IKE or manual keying
  - Uniquely identified by
    - A unique 32-bit security parameter index (SPI)
    - Destination address
    - Traffic protocol (AH or ESP)
  - A communication may need multiple SA
    - SA is *unidirectional*
    - Each SA can use either AH or ESP, but not both
    - Two way communication using both AH and ESP requires 4 SAs

Security Association Close-up

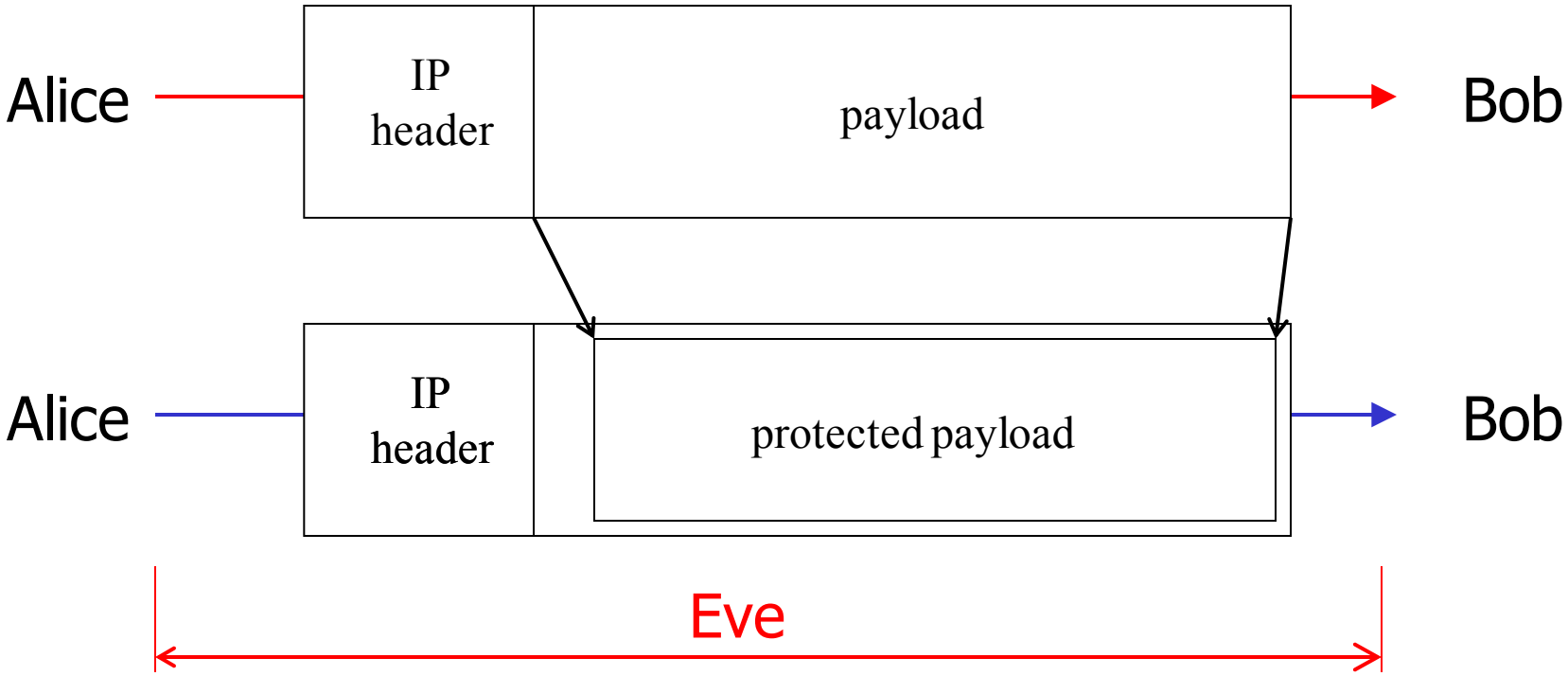
- An SA has those parameters
  - Sequence number counter
    - For outbound traffic; used to generate SPI for AH/ESP
  - Overflow flag
    - For inbound traffic; whether abort if the counter overflows
  - Anti-Replay Window (will discuss shortly)
  - AH algorithm, keys, etc. (if AH used)
  - ESP algorithm, keys, etc. (if ESP used)
    - For confidentiality or for authentication/integrity
  - SA lifetime
  - IPsec mode
    - Tunnel, transport, wildcard (mode specified by application)

IPsec Mode Overview

- Both traffic protocols (AH/ESP) can run in
  - Transport mode
  - Tunnel mode
- Four combinations
  - (AH,ESP)× (transport, tunnel)
  - For different purposes

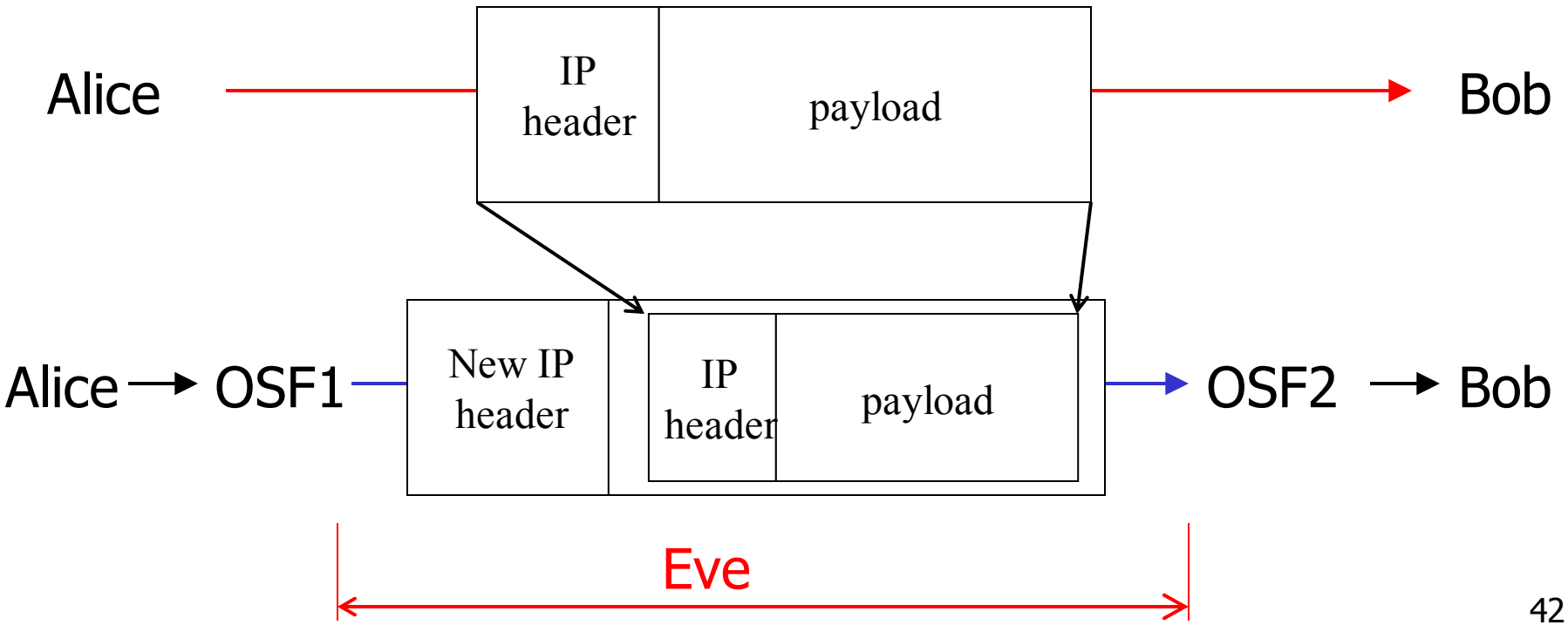
Transport Mode

- End to end (like SSL)
- The IP header is in clear (for routing)
  - The goal is to protect payload only



Tunnel Mode

- Security gateway to security gateway
- The whole packet is embedded as payload
  - The goal is to protect payload as well as traffic (the gateway usually has concurrent connections)

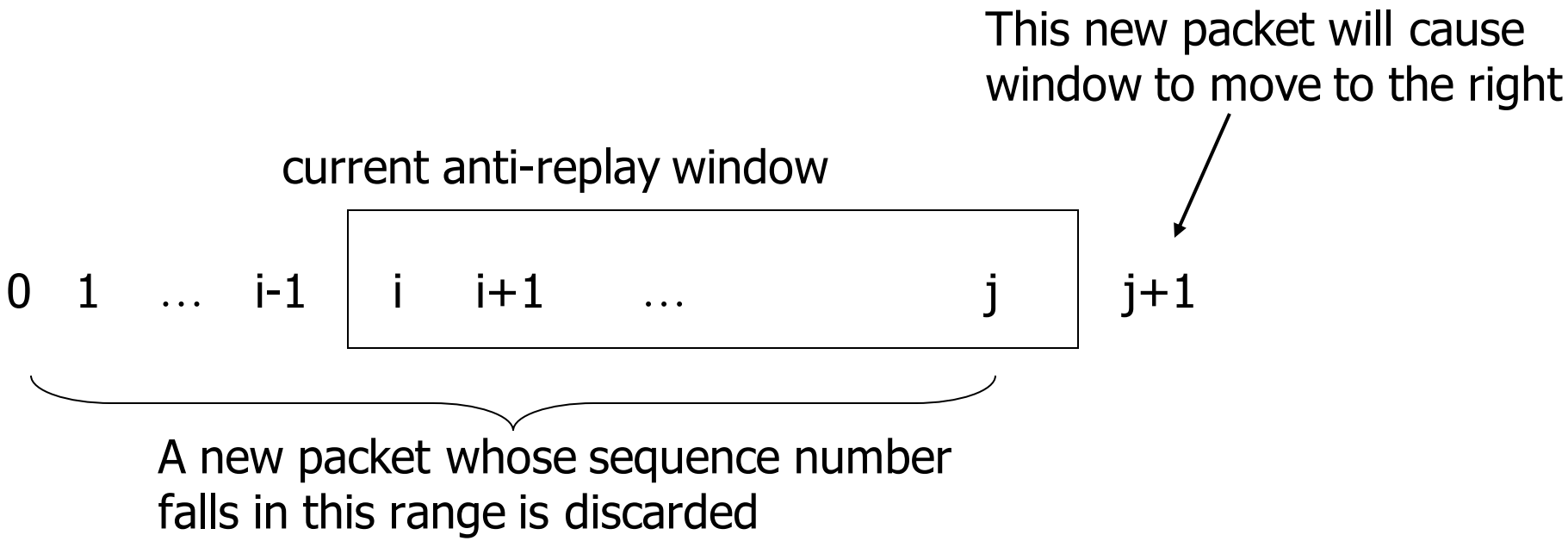


Traffic Protocols Overview

- Authentication Header (AH)
  - MAC of packet
  - Provides
    - Data integrity
    - Authentication
    - (no confidentiality)
- Encapsulating Security Payload (ESP)
  - Encryption (and optionally MAC) of packet
  - Provides
    - Data confidentiality (also for traffic in tunnel mode)
    - Data integrity (optionally)
    - Authentication (optionally)

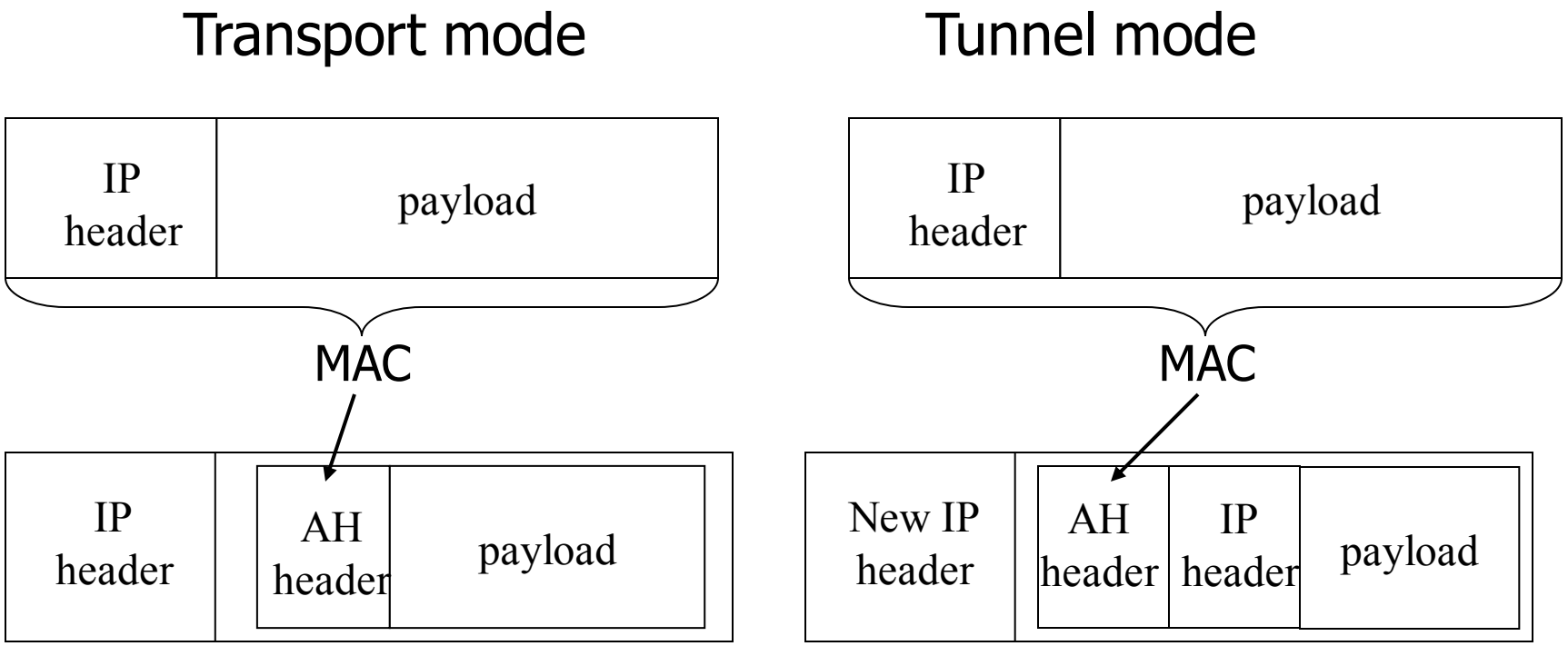
Replay Prevention

- Both AH and ESP prevents replay
  - Through incremental sequence number of packet
  - The 'anti-replay window' parameter in SA determines how many sequence numbers to keep in history
    - $<2^{32}$



AH Protocol Overview

- MAC on IP header and payload
  - Fields that change per hop are set to 0
  - The new IP header has protocol type changed to AH



AH Header Close-up

0	1	2	3
0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7
Next Header	Payload Length	RESERVED	
Security Parameters Index (SPI)			
Sequence Number			
Integrity Check Value (ICV)			

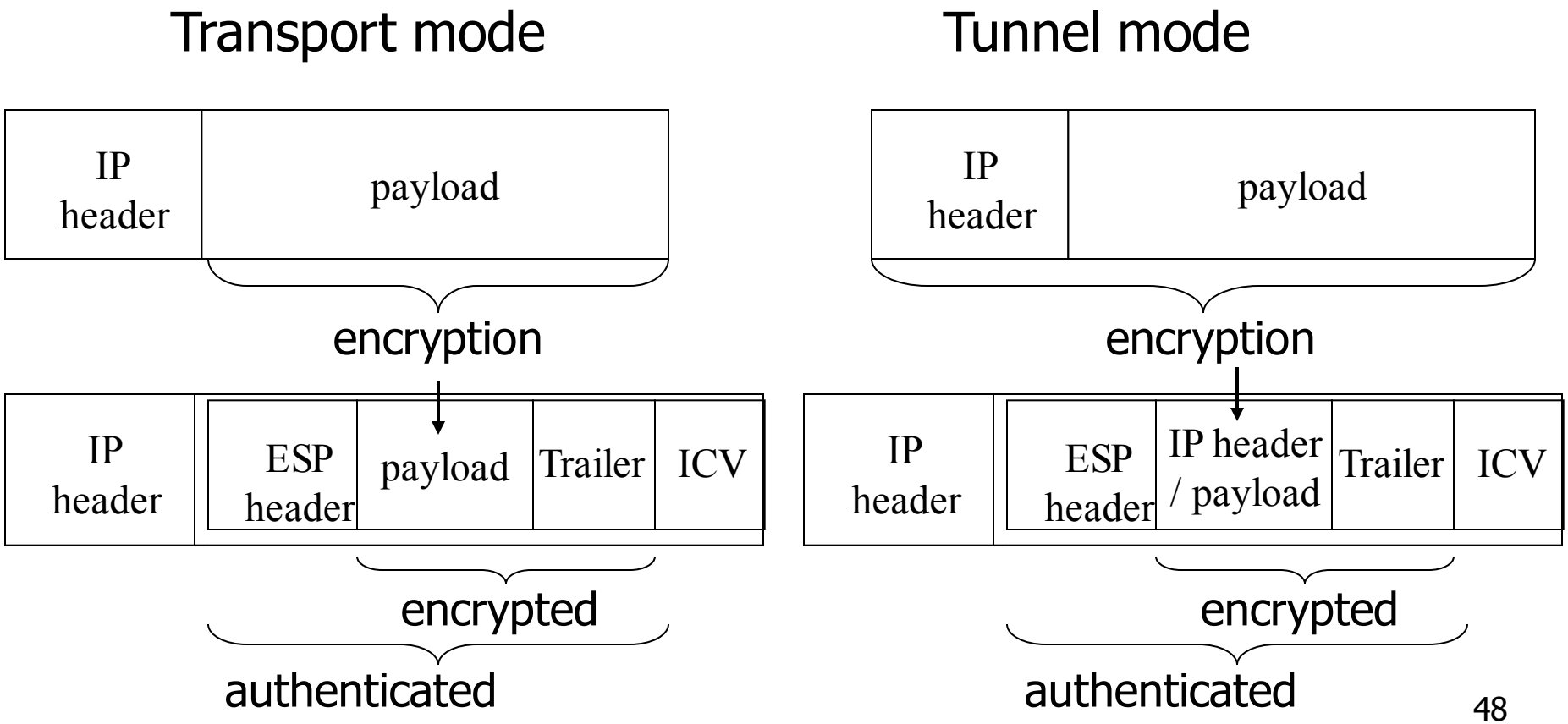
Sender needs to increment sequence number, and compute MAC of packet (ICV)

Recipient

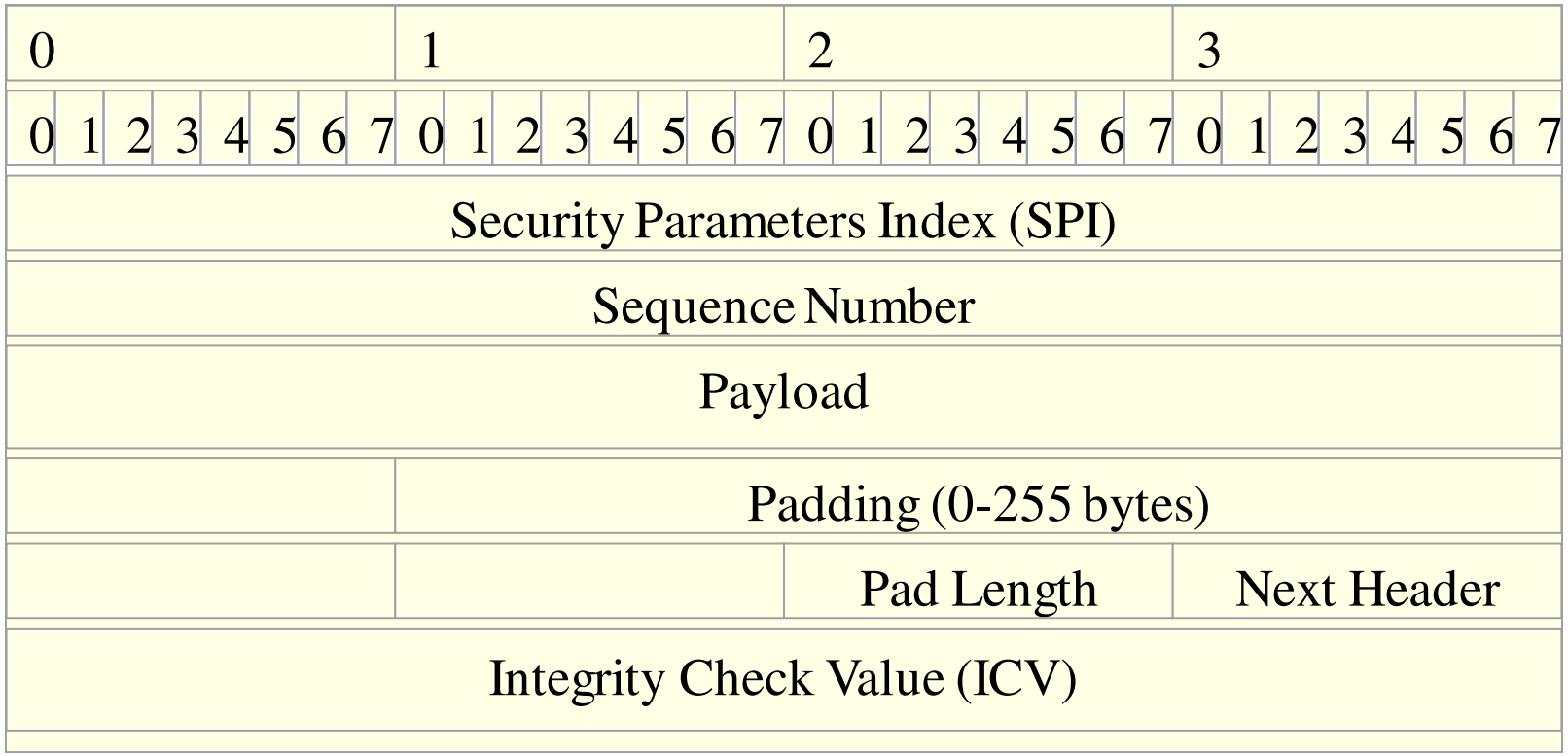
- Lookup SA based on SPI in AH header
  - If no associated SA, discard packet
- Verify IVC is correct
  - If not, discard
- Anti-replay window check (if used)
  - If repeated or out, discard
- Extract the original packet

ESP Protocol Overview

- Encrypt packet for confidentiality
- Optionally, authentication/integrity with ICV



ESP Header Close-up



Key Points

- Security protocols on different network layers
- End-to-end security vs link-security
- PEM is application-layer secure email protocol
- SSL is transport-layer security protocol
- IPsec is network-layer security protocol