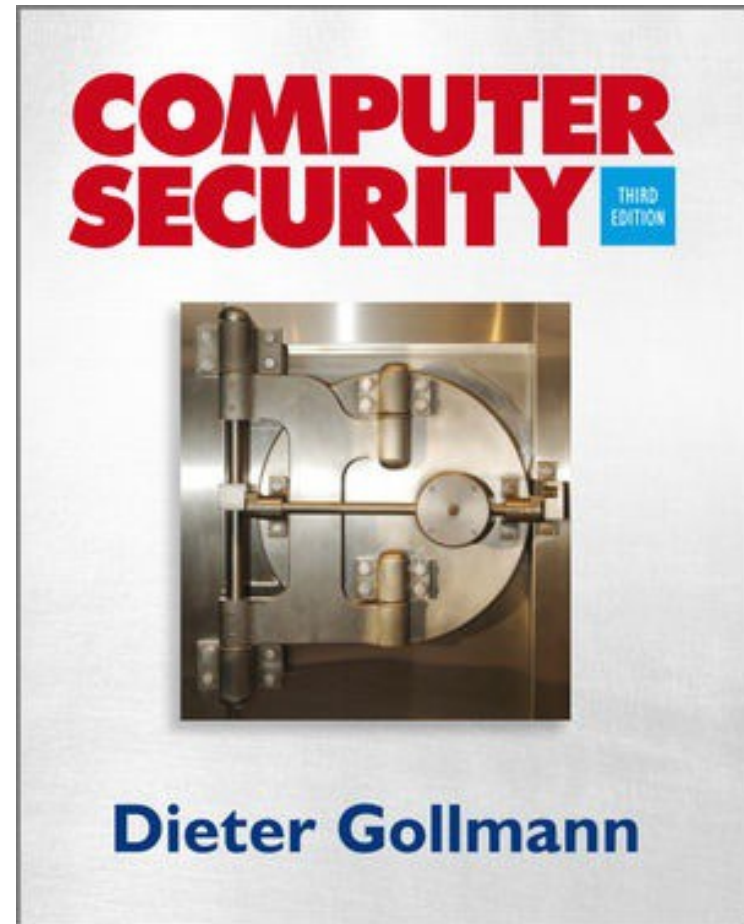


# Computer Security 3e

---

Dieter Gollmann



---

# Chapter 16: Communications Security

# Agenda

---

- Threat model
- Secure tunnels
- Protocol design principles
- IPsec
- SSL/TLS
- EAP

# Threat Model

---

- Attacker has access to communications link between two end points: can see and modify messages.
- The job of a communications security service is done once data has been delivered to an end point.
- This is the ‘old’ secret service threat model.
- A **passive attacker** just listens to traffic.
  - When the attacker is interested in the content of messages, we talk about **eavesdropping**, **wiretapping**, or **sniffing**.
  - **Traffic analysis** tries to identify communications patterns; may be possible even when attacker cannot read individual messages.
  - Attacker might also be interested in a target’s location.

# Active Attackers

---

- **Active attacker** may modify messages, insert new messages, or corrupt network management information like the mapping between DNS names and IP addresses.
- In **spoofing** attacks messages come with forged sender addresses.
- In **flooding (bombing)** attacks a large number of messages is directed at the victim.
- In **squatting** attacks, the attacker claims to be at the victim's location.
- Active attacks are not necessarily more difficult than passive attacks; e.g., in practice it is easier to send an email with a forged sender address than to intercept an email intended for someone else.

# Secure Tunnels

---

- Secure tunnel (channel): secure logical connection between two end points across an insecure network.
- Typical security guarantees are data integrity, confidentiality, and data origin authentication.
- End points might be machines named by domain names or IP addresses; end points might be specific software components hosted at a client or a server.
- Confusion about the precise nature of the end point authenticated can lead to “security services that do not provide any security at all”.
  - If the tunnel does not end where the user expects, the attacker may wait at the other side of the tunnel.
- Secure tunnels do not provide security services once data are received.

# Typical Cryptographic Primitives

---

- ‘Expensive’ asymmetric encryption and signature algorithms, **Diffie-Hellman** (still to come), only for entity authentication and key exchange.
- **Symmetric encryption** algorithms, for speed.
- ‘Cheap’ **MAC algorithms**, usually built from hash functions.
- (Keyed) **pseudo-random functions** for key derivation.
- **Sequence numbers** to prevent replay attacks.
- **Nonces** and **timestamps** for freshness in entity authentication.

# Protocol Layering

---

		Application	7
5	Application	Presentation	6
		Session	5
4	TCP	Transport	4
3	IP	Network	3
2	Network Interface	Data Link	2
1	Hardware	Physical	1

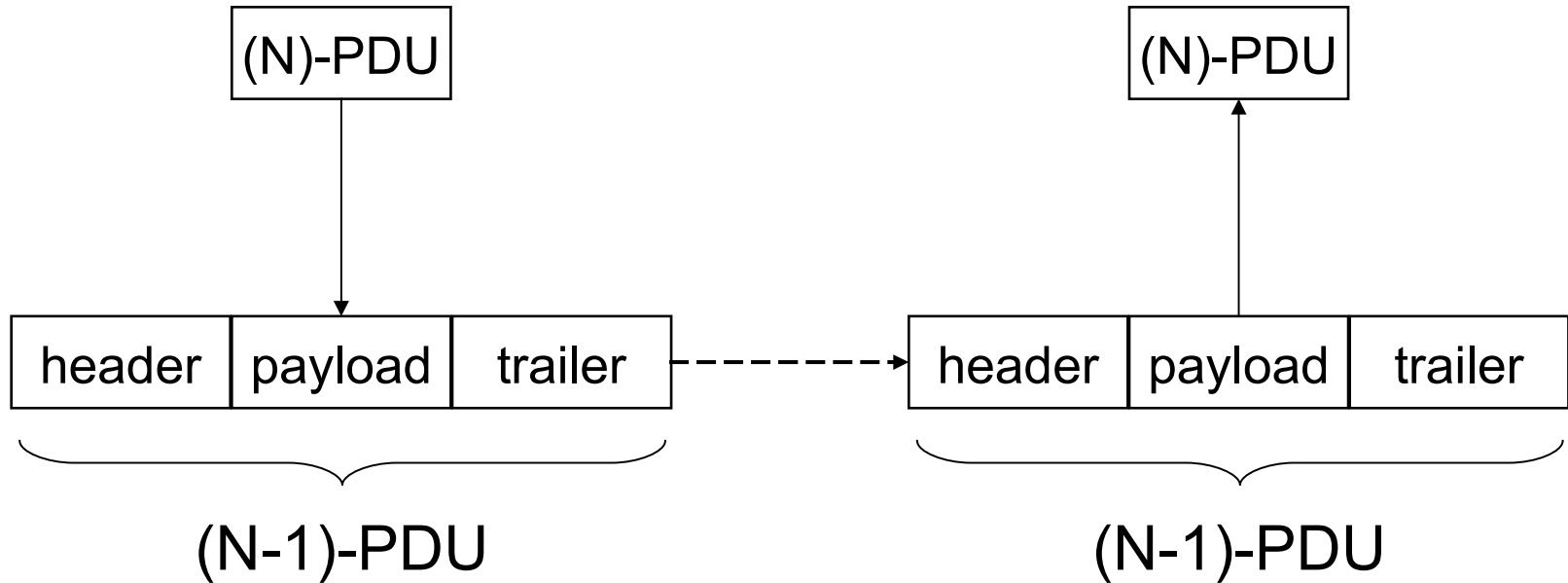
Internet

ISO/OSI 7 layer model



# Protocol Layering

---



PDU ... Protocol Data Unit

# Implementing Security Services

---

- Header in  $(N-1)$ -PDU is convenient location for storing security relevant data.
- Upper layer protocol can be **aware** of lower layer security services:
  - Upper layer protocol has to change its calls so that they refer to the security facilities provided.
- Lower layer security services can be **transparent** to upper layer protocol:
  - Upper layer protocol need not be changed at all.

# Security & Network Layers

---

- Security can be applied at any of the network layers except layer 1 (physical layer).
  - Even this is sometimes possible, e.g. spread spectrum techniques for limited privacy.
- In general, the lower the layer the more generic but the less specific the protection.
- **Endpoints** of security channels differ between layers.
- **End-to-end** or **hop-by-hop** security?
- Example: protection at Data Link (Network Interface) layer, e.g. **link level encryptor**.
  - Advantage: covers all traffic on that link, independent of protocols above.
  - Disadvantage: protection only for one 'hop'.

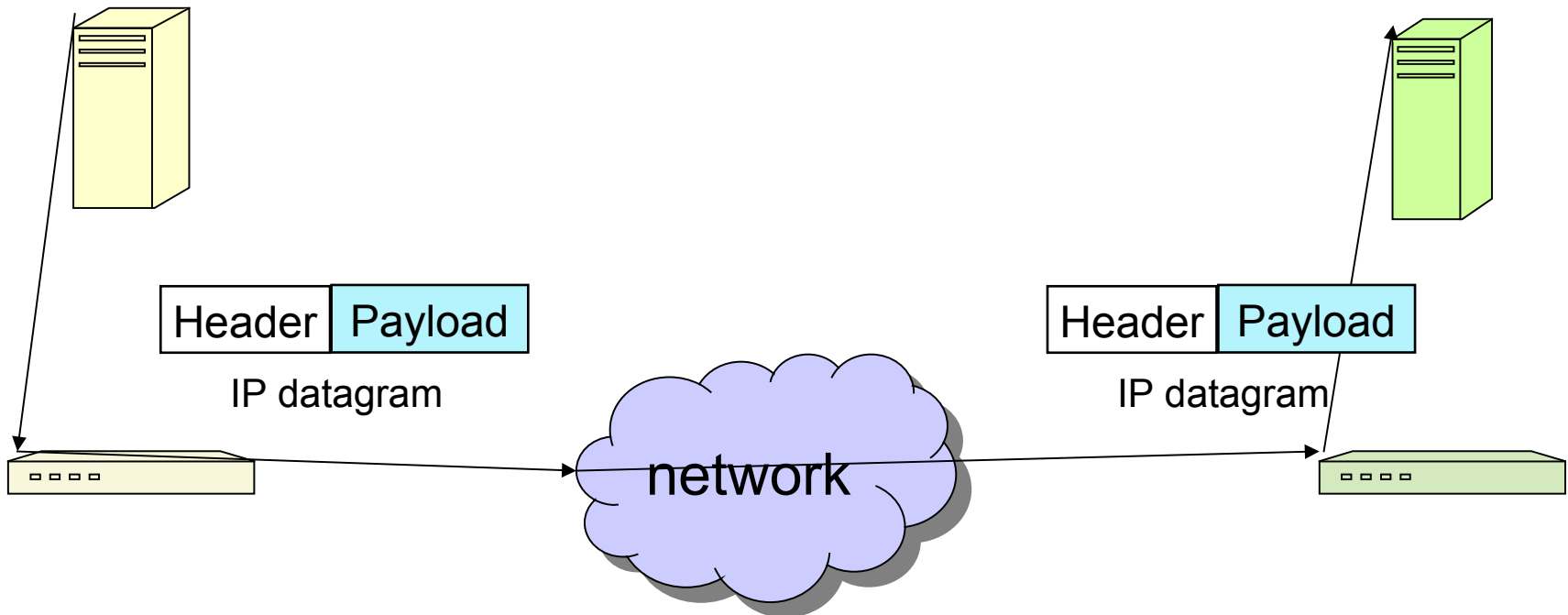
# IPsec

---

- Defined in RFCs 4301 – 4309 (obsolete 2401-2412).
- Provides security at network (Internet) layer.
  - All IP datagrams covered.
  - No re-engineering of applications.
  - **Transparent** to upper layer.
- Mandatory for next generation IPv6, optional for IPv4.
- Two basic modes of use:
  - **Transport mode**: IPsec-aware hosts as endpoints.
  - **Tunnel mode**: for IPsec-unaware hosts, tunnel established by intermediate gateways or host OS.

# IPsec Transport Mode

- Host-to-host (end-to-end) security:
  - IPsec processing performed at endpoints of secure channel.
  - Endpoint hosts must be IPsec-aware.

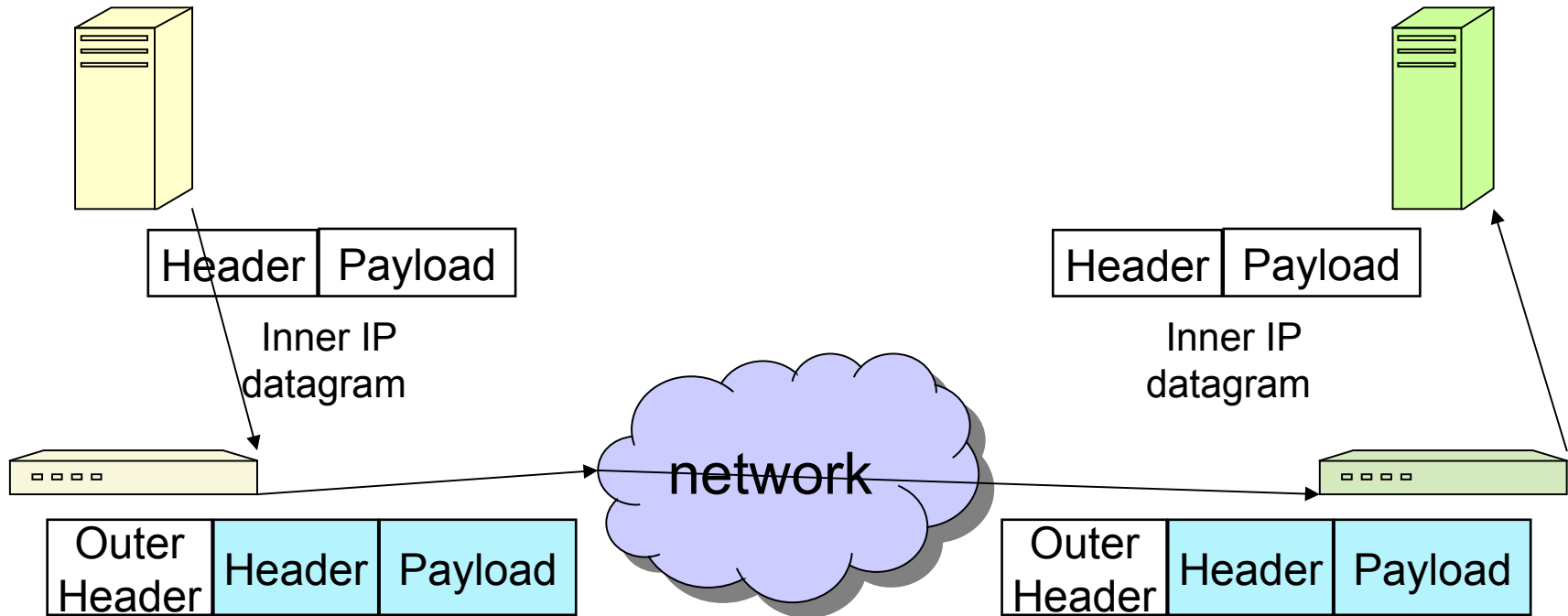


# IPsec Tunnel Mode

---

- Entire IP datagram plus security fields treated as new payload of 'outer' IP datagram.
  - Original 'inner' IP datagram **encapsulated** within 'outer' IP datagram.
- IPsec processing performed at **security gateways** on behalf of endpoint hosts.
  - Gateway could be perimeter firewall or router.
  - Gateway-to-gateway but not end-to-end security.
  - Hosts need not be IPsec-aware.
- Encrypted inner IP datagram, including original source and destination addresses, not visible to intermediate routers.

# IPsec Tunnel Mode



# IPsec

---

- Authentication and/or confidentiality services for data:
  - AH protocol [RFC 4302]
  - ESP protocol [RFC 4303 ]
- Use of AH being deprecated in favour of ESP.
  - Political reasons for introducing an authentication-only protocol in the 1990s have faded.
- (Too?) flexible set of key establishment methods (covered later in the course): IKE, IKEv2.



# AH Protocol [RFC 4302]

---

- **AH = Authentication Header**: provides connectionless data integrity and data origin authentication.
  - Authenticates whole payload and most of header.
- Prevents IP address spoofing: source IP address is authenticated.
- **Creates stateful channel using sequence numbers. Heresy!**
- Prevents replay of old datagrams: AH sequence number is authenticated.
- Uses MAC and secret key shared between endpoints.

# ESP Protocol

---

- Encapsulating Security Payload [RFC 4303].
- Provides one or both of:
  - Confidentiality for payload/inner datagram; sequence number not protected by encryption.
  - Authentication of payload/inner datagram, but not of outer IP header.
- Traffic-flow confidentiality in tunnel mode.
- Symmetric encryption and MACs based on secret keys shared between endpoints.

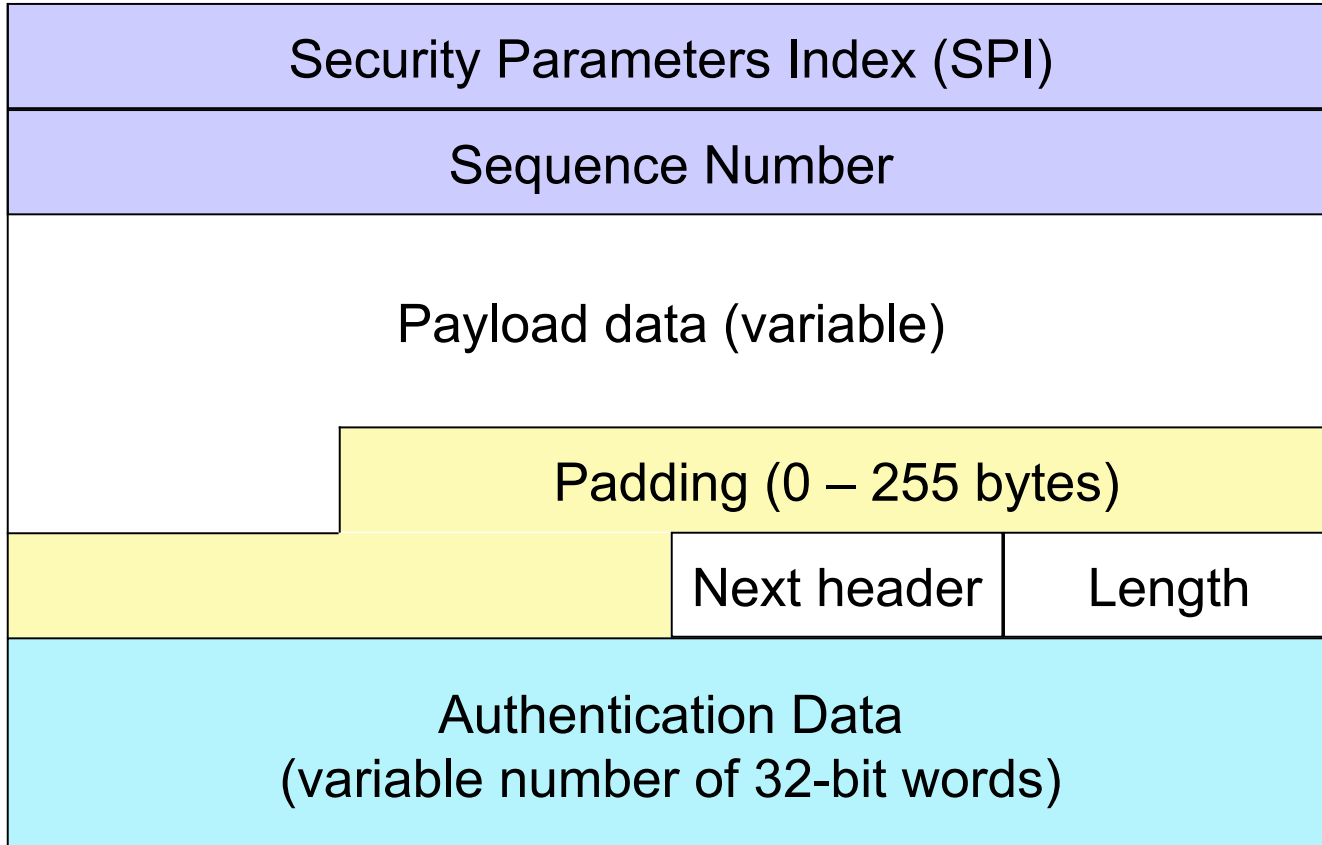
# ESP Headers

---

- ESP specifies header and trailer to be added to IP datagrams.
- Header fields include:
  - SPI (Security Parameters Index): identifies which algorithms and keys are to be used for IPsec processing (more later).
  - Sequence number.
- Trailer fields include:
  - Any padding needed for encryption algorithm (may also help disguise payload length).
  - Padding length.
  - Authentication data (if any), i.e. the MAC value.

# ESP Header (RFC 2406)

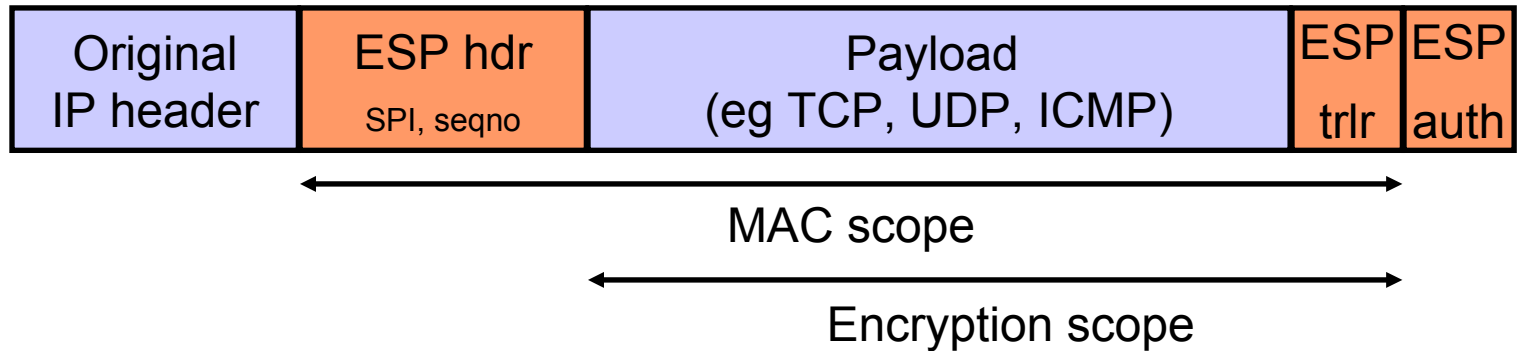
---



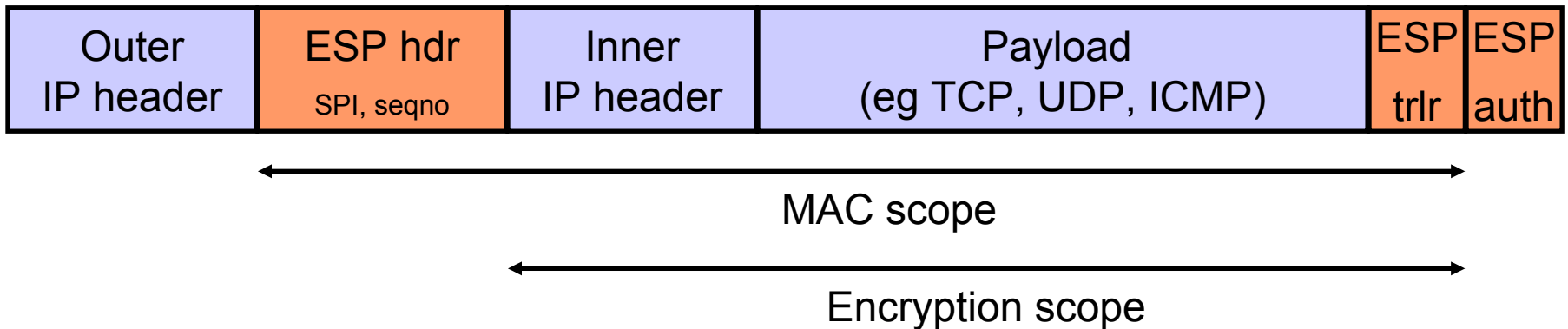
# ESP Protocol – Transport & Tunnel

---

ESP in transport mode:



ESP in tunnel mode:



# IPsec Security Association (SA)

---

- A SA is a one-way (simplex) relationship between sender and receiver.
  - Specifies processing to be applied to *this* datagram from *this* sender to *this* receiver.
- List of active SAs held in SA database (SAD).
- Each SA identified by SPI, source address, destination address; contains:
  - Sequence number counter and anti-replay window.
  - AH/ESP info: algorithms, IVs, keys, key lifetimes.
  - SA lifetime.
  - Protocol mode: tunnel or transport.
  - ...

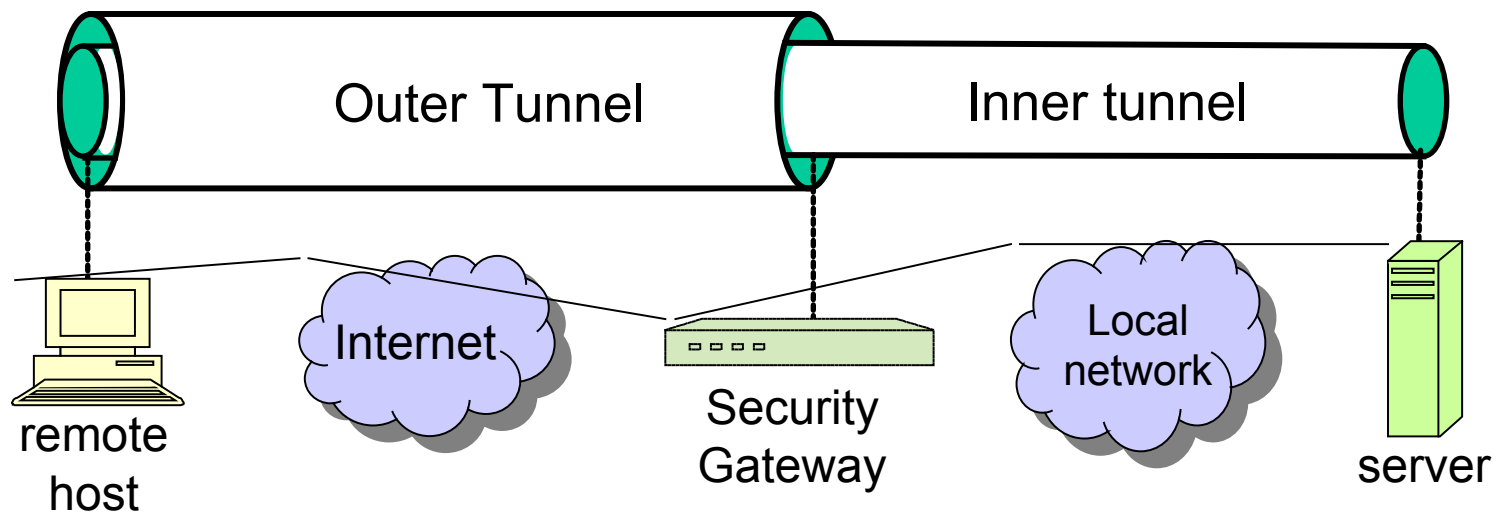
# Combining SAs

---

- IPsec security services may be provided at different points in network.
  - Host-to-host.
  - Gateway-to-gateway for Virtual Private Network (VPN).
- SAs can be combined using:
  - **Transport adjacency**: more than one SA applied to same IP datagram without tunnelling.
  - **Iterated tunnelling**: multiple levels of nesting of IPsec tunnels; each level has its own SA; each tunnel can begin/end at different IPsec site along route.

# Remote Host to Internal Server

- Remote host has Internet access to gateway, then gains access to server behind gateway.
- Traffic to server protected in inner tunnel.
- Outer tunnel protects inner traffic over Internet.





# IPsec Key Management

---

- IPsec needs a lot of symmetric keys:
  - One key for each SA.
  - Different SA for each combination of {ESP,AH} [ {tunnel,transport} [ {sender, receiver}.
- Two sources for SAs and keys:
  - **Manual keying**: works for small number of nodes but hopeless for reasonably sized networks of IPsec-aware hosts; requires manual re-keying.
  - **IKE: Internet Key Exchange** [RFC 2409, 4306]; many options and parameters.

# IKE Security Goals

---

- Entity authentication of participating parties.
- Establish fresh shared secret, to derive further keys:
  - for protecting IKE management channel,
  - for SAs for general use.
- Secure negotiation of all algorithms.
  - Authentication method, key exchange method, encryption and MAC algorithms, hash algorithms.
- Resistance to Denial-of-Service attacks.
- Options for perfect forward secrecy, deniable authentication and identity protection.

# IPsec Security Policy

---

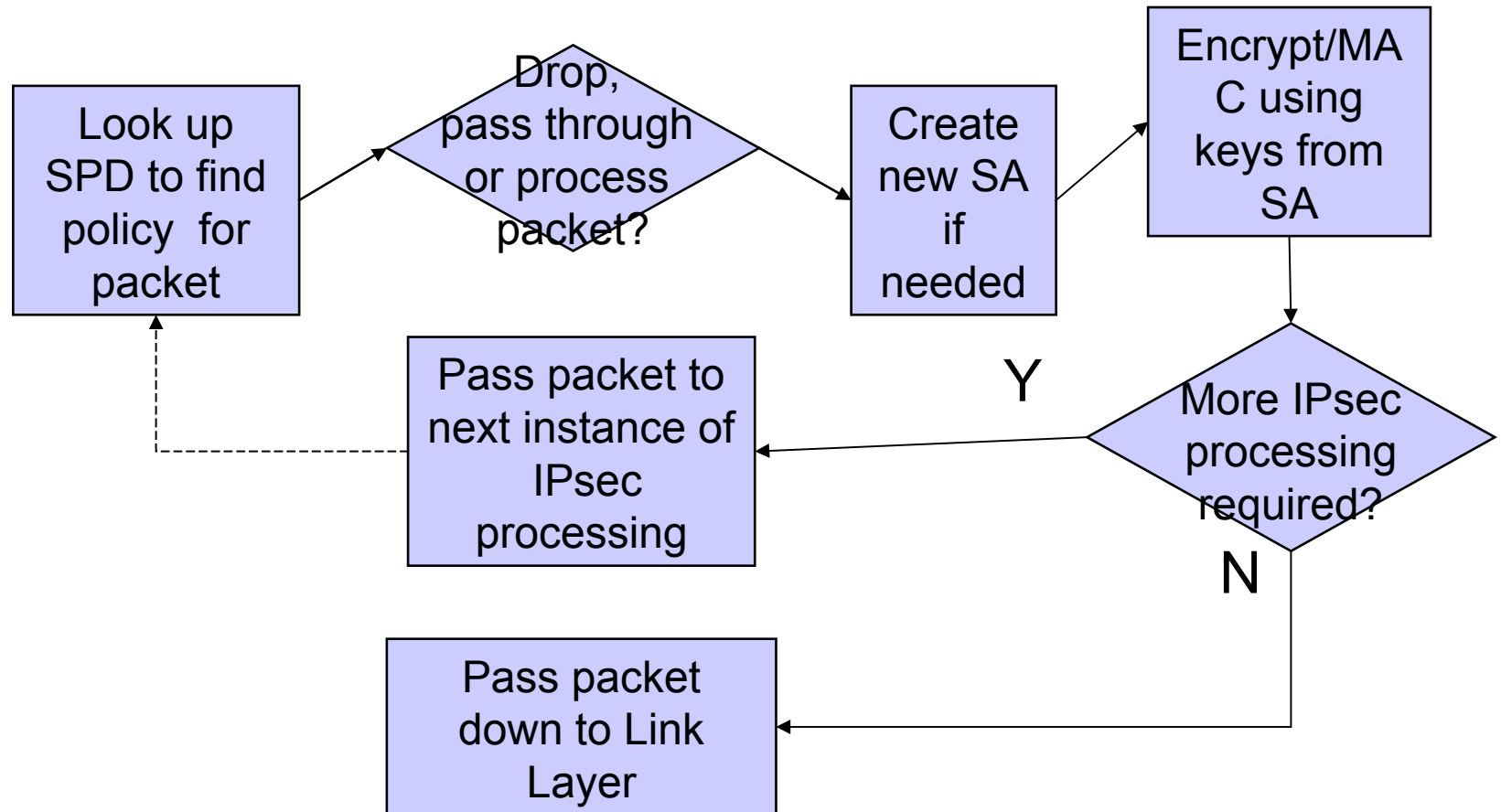
- IPsec aware host needs rules for processing packets.
  - Should packet be dropped, passed through, encrypted, MACed?
  - Which key and algorithm to apply?
- Rules stored in a **Security Policy Database (SPD)**.
- SPD consulted for each outbound & inbound packet.
- Fields in IP datagram compared to fields in SPD entries to find matches.
  - Match can be based on source and destination addresses (ranges of addresses), transport layer protocol, transport layer port numbers,...
- Match identifies a **Security Association (SA)** or group of SAs (or the need for new SA).

# Finding the ‘Right’ SA

---

- Each entry in the Security Association Database (SAD) must indicate whether the SA lookup makes use of the destination, or destination and source IP addresses in addition to the SPI.
- For each inbound IPsec-protected packet, search the SAD such that the **entry that matches the “longest” SA identifier** is found.
  1. Search the SAD for a match on **{SPI, destination address, source address}**; if an SAD entry matches, then process the inbound packet with that entry.
  2. Otherwise, search the SAD for a match on **{SPI, destination address}**; if an SAD entry matches, then process the inbound packet with that entry.
  3. Otherwise, search the SAD for a match on only **{SPI}** if the receiver has chosen to maintain a single SPI space for AH and ESP, or on **{SPI, protocol}** otherwise; if an SAD entry matches, then process the inbound packet with that entry.
  4. Otherwise, discard the packet and log an auditable event.

# IPsec Outbound Processing



---

# SSL/TLS

# SSL/TLS Overview

---

- SSL = Secure Sockets Layer.
  - Unreleased v1, flawed but useful v2, good v3.
- TLS = Transport Layer Security Protocol.
  - TLS1.0 = SSL3.0 with minor modifications
  - Version 1.2 defined in RFC 5246 (October 2008)
  - Open-source implementation at [www.openssl.org/](http://www.openssl.org/)
- Widely used in Web browsers and servers to support 'secure e-commerce' over HTTP.
  - HTTPS sessions indicated by browser lock.

# SSL/TLS Basic Features

---

- Security at session layer.
  - 'Thin layer' between TCP and, e.g., HTTP.
  - TCP provides reliable, end-to-end transport.
  - Applications must be **aware** of SSL, need some modification.
- Two layer architecture:
  - SSL Record Protocol: provides secure, reliable channel to second layer.
  - Upper layer carries **SSL Handshake Protocol**, Change Cipher Specification Protocol, Alert Protocol, HTTP, any other application protocols.



# SSL/TLS Handshake – Goals

---

- Entity authentication of participants.
  - Participants are ‘client’ and ‘server’.
  - Server nearly always authenticated, client more rarely.
  - Appropriate for most e-commerce applications.
- Establish a fresh, shared secret.
  - Shared secret used to derive further keys.
  - For confidentiality and authentication in SSL Record Protocol.
- Secure ciphersuite negotiation.
  - Encryption and hash algorithms
  - Authentication and key establishment methods.

# Sessions & Connections

---

- **Session:**

- Created by handshake protocol.
- Defines set of cryptographic parameters (encryption and hash algorithm, master secret, certificates).
- Carries multiple **connections** to avoid repeated use of expensive handshake protocol.

- **Connection:**

- State defined by nonces, secret keys for MAC and encryption, IVs, sequence numbers.
- Keys for many connections derived from single **master secret** created during handshake protocol.

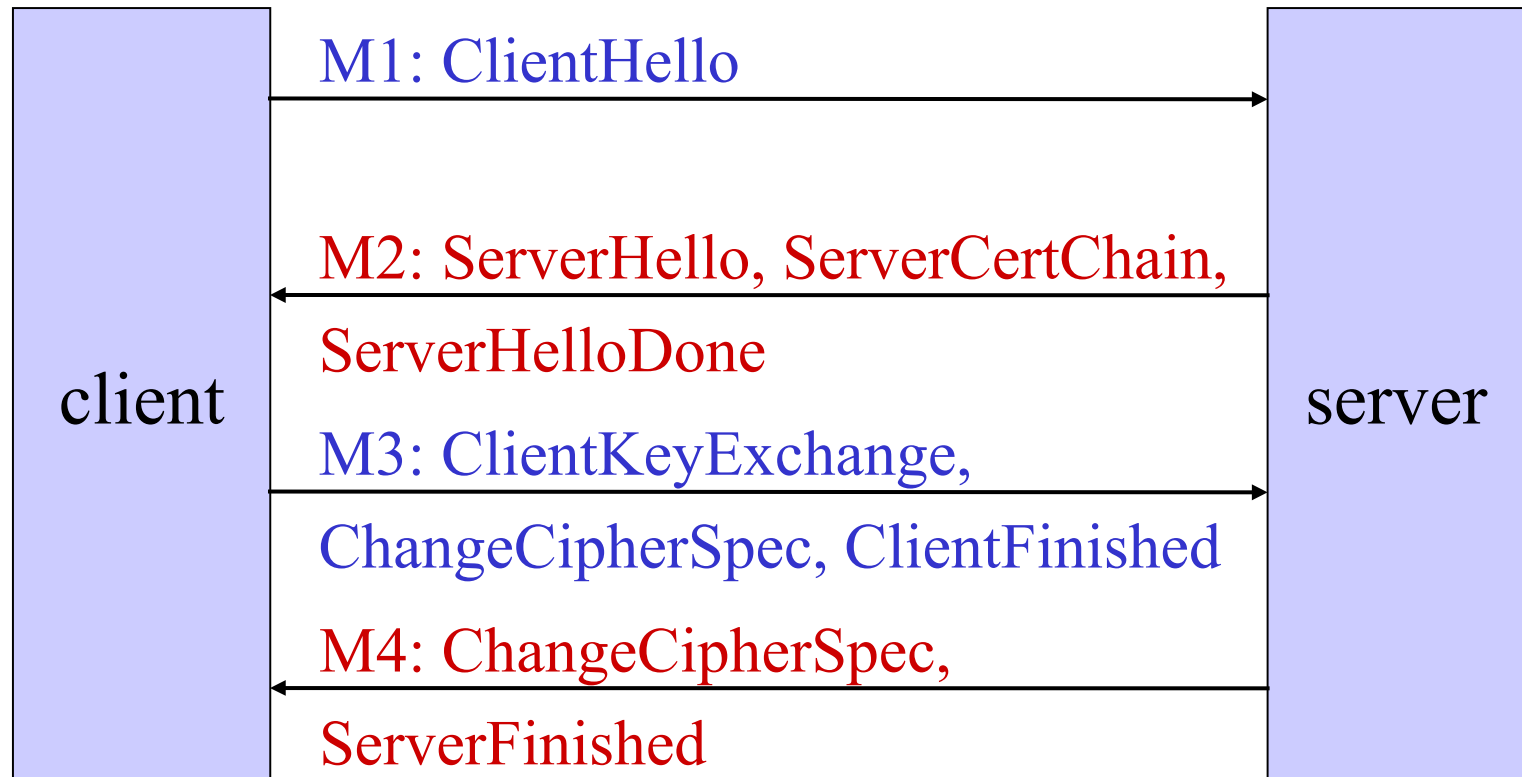
# SSL Handshake Protocol: Run

---

- We sketch the most common use of SSL:
  - No client authentication.
  - Client sends `pre_master_secret` using Server's public encryption key from Server certificate.
  - Server authenticated by ability to decrypt to obtain `pre_master_secret`, and construct correct `finished` message.
- Other protocol runs are similar.

# SSL Handshake Protocol Run

---



# M1: ClientHello

---

- Client initiates connection.
- Sends client version number.
  - 3.1 for TLS.
- Sends **ClientNonce**.
  - 28 random bytes plus 4 bytes of time.
- Offers list of ciphersuites:
  - Key exchange and authentication options, encryption algorithms, hash functions.
  - E.g. `TLS_RSA_WITH_3DES_EDE_CBC_SHA`.

# M2: `ServerHello`, ...

---

- Sends server version number.
- Sends `ServerNonce` and `SessionID`.
- Selects single ciphersuite from list offered by client.
- Sends `ServerCertChain` message.
  - Allows client to validate server's public key back to acceptable root of trust.
- (optional) `CertRequest` message.
  - Omitted in this protocol run – no client authentication.
- Finally, `ServerHelloDone`.

# M3: ClientKeyExchange, ...

---

- **ClientKeyExchange** contains encryption of **pre\_master\_secret** under server's public key.
- **ChangeCipherSpec** indicates that client is updating cipher suite to be used on this session.
  - Sent using SSL Change Cipher Spec. Protocol.
- Optional (only when client is authenticated):  
**ClientCertificate**,  
**ClientCertificateVerify** messages.
- Finally, **ClientFinished** message.
  - MAC on all messages sent so far (both sides).
  - MAC computed using **master\_secret**.

# M4: ChangeCipherSpec, ...

---

- **ChangeCipherSpec** indicates that server is updating cipher suite to be used on this session.
  - Sent using SSL Change Cipher Spec. Protocol.
- Finally, **ServerFinished** message.
  - MAC on all messages sent so far (both sides).
  - MAC computed using **master\_secret**.
  - Server can only compute MAC if it can decrypt **pre\_master\_secret** in M3.



# SSL Handshake Protocol Run

---

1. Is the client authenticated to the server in this protocol run?
  - No!
2. Can adversary learn value of **pre\_master\_secret**?
  - No! Client has validated server's public key; To learn **pre\_master\_secret** the server's private key is needed to decrypt ClientKeyExchange
3. Is the server authenticated to the client?
  - Yes! ServerFinished includes MAC on nonces computed using key derived from **pre\_master\_secret**.

# SSL/TLS Applications

---

- Secure e-commerce using SSL/TLS.
- Client authentication not needed until client decides to buy something.
- SSL provides secure channel for sending credit card information.
- Client authenticated using credit card information, merchant bears (most of) risk.
- Widely deployed (de-facto standard).

---

# EAP

# EAP

---

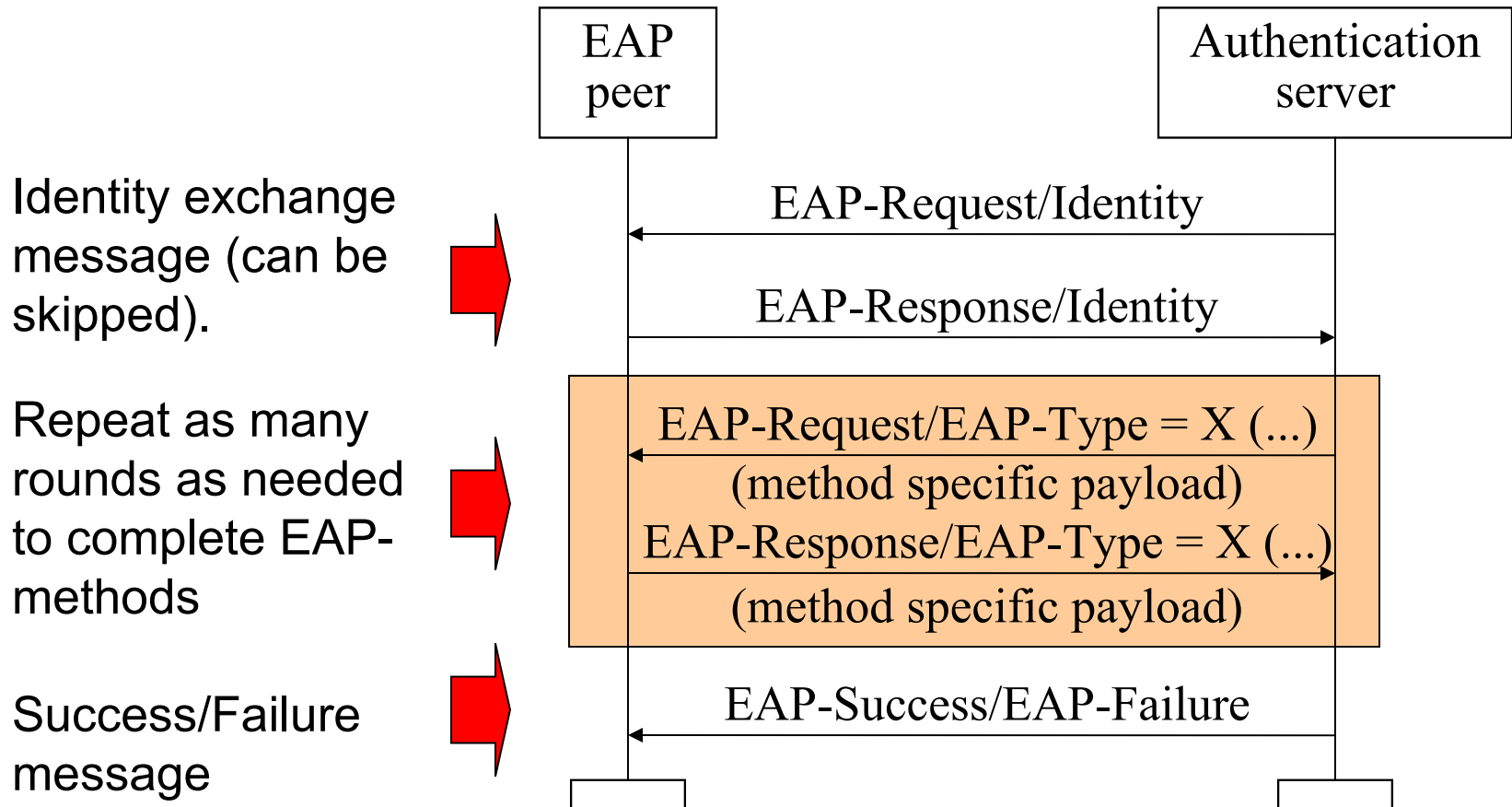
- Extensible Authentication Protocol
- Specified in IEEE 802.1X and RFC 3748 for entity authentication.
- Defines general message flow; can be implemented using a variety of underlying mechanisms.
- Application:
  - WLAN authentication (802.1X); Cisco wireless equipment implements EAP.
  - EAPOL: EAP over LAN; just a delivery mechanism; for authentication, an EAP method such as EAP-TLS or EAP-TTLS has to be defined.
  - 3GPP WLAN interworking, e.g. EAP-AKA
  - Windows supports various EAP methods for remote access.

# Design Principle

---

- Applications calling an authentication protocol just see an abstract message flow.
- Thus, authentication mechanism can be chosen independent of application.
- Authentication mechanism can be changed without having to rewrite application.
- Other examples: GSS-API, SAML profiles for web services.

# EAP Generic Message Flow



# EAP methods

---

- Generic EAP messages exchange identities, encapsulate authentication protocol messages.
- These protocols are called **methods**.
- Users identified by **Network Access Identifier (NAI)**; specified in RFC 4282.
- EAP methods provide at least one way authentication (EAP peer [ EAP server).
- Many existing or proposed EAP methods that differ in their characteristics & security levels.

# EAP-TTLS v0 (with CHAP)

