# Internet Security

## Alessandro Armando

Computer Security Laboratory (CSec)
DIBRIS, University of Genova

Computer Security
Academic year 2018-2019

# Computer networks

- Physically: a collection of "segments" that transmit bit streams.
  **Examples:** wire between two nodes or multi-access links like a LAN
  (e.g., Ethernet, token rings, packet radio networks).

- Logically: a communication medium between principals.
  **Example**: client communicates to server.

- A secure channel is yet another abstraction.
  Other abstractions may concern availability, privacy of communication partners, etc.
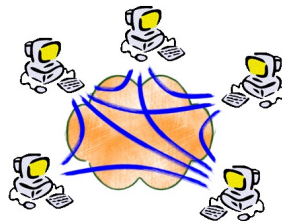
# Layered communication



- Logical functionality built in layered way.
  Application communication                            builds upon
  reliable transport between nodes                     builds upon
  unreliable transport across links and switches       builds upon
  packet transportation across single links

- TCP/IP protocol reference model

  | | |
  |---|---|
  | Application | Telnet, FTP, HTTP, RPC, SMTP, SET, ... |
  | Transport/session | TCP, UDP |
  | Network (Internet) | IP, ICMP, ... |
  | Link (Interface) | Network interface & device drivers (IEEE 802.x, PPP, SLIP) |

- Alternative model (OSI) with three additional layers: presentation, physical, and transport/session distinction.
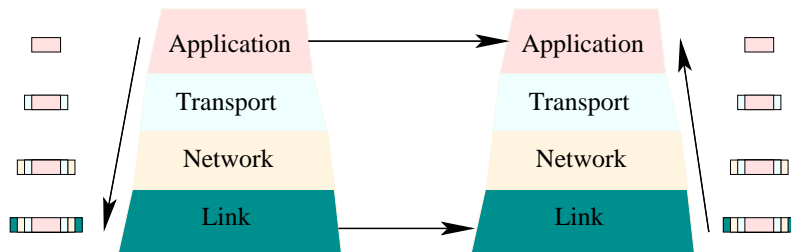
- *i*-th layer of one node communicates with *i*-th layer of different node, each using services provided by their lower layers.
- Headers/trailers added to (or stripped from) packets as they traverse the protocol stack.
- Layers are an abstraction. Reality is usually rather different. TCP/IP model developed by practioners in parallel to ISO/OSI.

# Internet security

- Internet: Confederation of networks using TCP/IP protocols.

- No global domain of trust.
  - Different subnetworks may (or may not) be trustworthy.
  - 15+ hops for a packet from source to destination is common.

- Problem: how do we secure communication/applications?

- One possibility: secure applications over insecure channels.
  **Example:** Kerberos is typically implemented and support by
  different applications. Requires "kerberized" applications.
  **Example:** Use of PGP to encrypt/sign mail.

- Securing other layers is also possible.

- Internet Protocol (IP): deliver data across a network.
  - Packet headers specify source and destination addresses.
  - Protocol computes path and forwards packets over multiple links from source to destination.
  - Current version is IPv4 (transition to IPv6 under way).

- Transmission Control Protocol (TCP): establishes reliable communication between systems across a network.
  Reliable: either all data delivered without loss, duplication, or reordering, or the connection is terminated.

- Neither provide security: no authentication or confidentiality. Addresses can be faked. Payload can be read and modified.

- For most implementations of IP stacks
  - Transport layer and below implemented in operating system.
  - Above transport layer implemented in user process.

- Two representative examples:
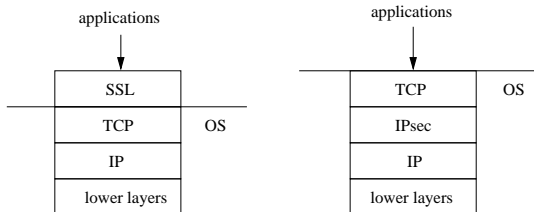


**SSL (or TLS/SSH):** OS doesn't change, applications do. SSL API is a superset of "sockets" API to TCP.

**IPsec:** OS changes. Applications and (TCP) API unchanged.

**Application (or end-to-end):**

:-) No assumptions needed about security of protocols used, routers, etc.

:-) Security decision can be based on user-ID, data, etc.

:-( Applications must be designed "security aware".

**Between application layer and transport layer:** e.g., SSL

:-) No modification to OS. Minimal changes to applications.

:-( Problems interacting with TCP. SSL may reject data that TCP accepts. SSL must then drop connection $\Rightarrow$ easy DOS.

**IPsec:**

:-) Transport layer security without modifying applications.

:-( Only authenticates IP addresses, no user authentication.

:-| More is possible, but it requires changing API and applications.

# Outline

- Provides a secure channel for all applications.
  Encryption and/or authentication of traffic.

- Ability to do filtering, based on a policy database.
  Just as if there were a firewall between the two ends.

- Installed in:
  **Operating systems:** for end-to-end security;

  **Security gateways:** firewalls or routers.
  Latter used for implementing Virtual Private Networks (VPNs).

# Applications of IPsec

- Secure branch office connectivity over the Internet

- Secure remote access over the Internet

- Establishing extranet and intranet connectivity with partners

- Echancing e-commerce security
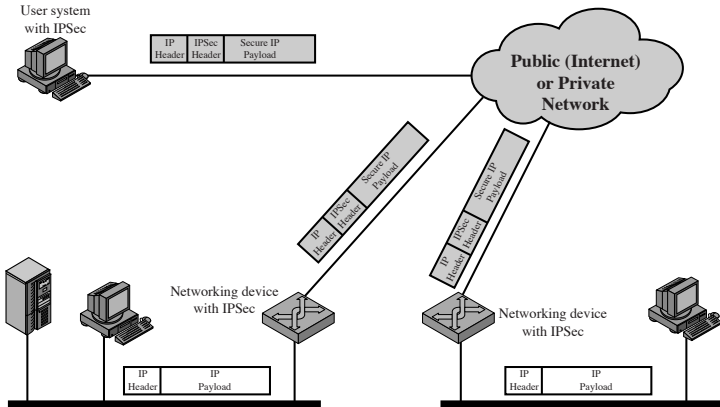
**Figure 16.1 An IP Security Scenario**

IPsec is an IETF Standard.

Complex specification covering protocols for a variety of purposes:

**Authentication Header (AH):** protects the integrity and the authenticity of IP datagrams (but not their confidentiality).

**Encapsulating Security Payload (ESP):** protects confidentiality and optionally also integrity.

**Key Management (IKE):** Internet Key Exchange Protocol.

# IPsec Services

**Table 16.1  IPSec Services**

|  | AH | ESP (encryption only) | ESP (encryption plus authentication) |
|---|:---:|:---:|:---:|
| Access control | ✔ | ✔ | ✔ |
| Connectionless integrity | ✔ |  | ✔ |
| Data origin authentication | ✔ |  | ✔ |
| Rejection of replayed packets | ✔ | ✔ | ✔ |
| Confidentiality |  | ✔ | ✔ |
| Limited traffic flow confidentiality |  | ✔ | ✔ |

# IPsec: Security Associations (SA)

- A security association is a one-way relationship between sender and receiver defining security services.



I encrypt using 3−DES
and use MD5 for authentication
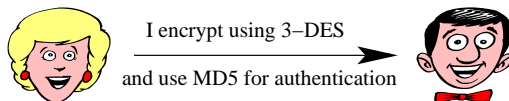
- Specifies things like: authentication algorithm (AH), encryption algorithm (ESP), keys, key lifetimes, lifetime of security association, protocol mode (tunnel or transport), ...

- Identified by fields in AH/ESP headers including the Security Parameters Index and destination address.

- SA is established using IKE, or possibly some other protocol. Implementation stores these in a security association database.

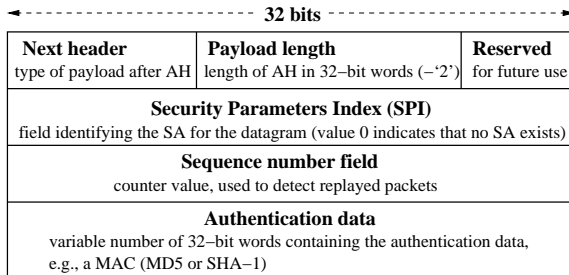# Outline

# IPsec: Authentication Header (AH)

<---------------------- **32 bits** ---------------------->

| **Next header** | **Payload length** | **Reserved** |
|---|---|---|
| type of payload after AH | length of AH in 32−bit words (−'2') | for future use |
| **Security Parameters Index (SPI)** | | |
| field identifying the SA for the datagram (value 0 indicates that no SA exists) | | |
| **Sequence number field** | | |
| counter value, used to detect replayed packets | | |
| **Authentication data** | | |
| variable number of 32−bit words containing the authentication data, e.g., a MAC (MD5 or SHA−1) | | |

Extra header between layers 3 and 4 (IP and TCP) providing destination enough information to identify SA. AH guarantees integrity only, but also protects part of IP header.

| TCP |
|---|
| IPsec |
| IP |
| lower layers |
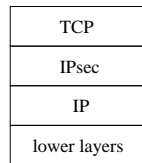
Sequence number is initialized to zero and incremented by the sender for each packet. Receiver stores incoming packets in a sliding window (default size 64) to order and sort out duplicates. (IP does not guarantee delivery or order.)

**Original Datagram (here for TCP):**

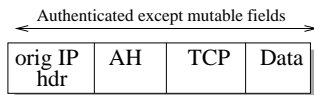| orig IP hdr | TCP | Data |
|---|---|---|

**Transport mode:**

Authenticated except mutable fields

| orig IP hdr | AH | TCP | Data |
|---|---|---|---|

- AH inserted after IP header, before IP payload.
- MAC taken of entire packet (except for mutable fields).
- Provides end-to-end protection between IPsec-enabled systems.

**Tunnel mode:**

Authenticated except mutable fields in new IP header

| New IP hdr | AH | orig IP hdr | TCP | Data |
|---|---|---|---|---|

- Entire original packet authenticated; new outer IP header.
- Inner header carries ultimate source/destination address.
- New outer header also protected (except mutable fields) and may contain different IP addresses, e.g. firewalls or security gateways.
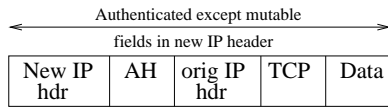
AH used to provide authenticated channels either end-to-end (typically transport mode) or in tunnel model to a security gateway.

# IPsec: Encapsulating Security Payload (ESP)

Header specifies encryption and optional authentication
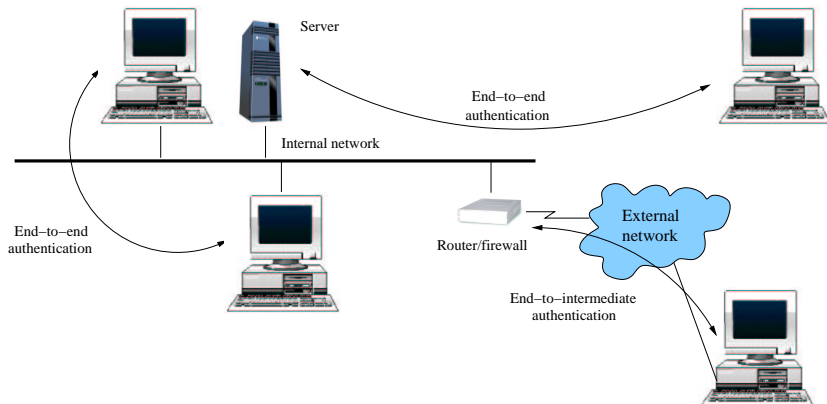
<------------------------ **32 bits** ------------------------>

| **Security Parameters Index (SPI)** |
|---|
| field identifying the SA for the datagram (value 0 indicates that no SA exists) |
| **Opaque transform data** |
| protected field containing further parameters relevant for the processing of the cryptographic algorithm |

**Transport mode:**

<------- Authenticated ------->
<----- Encrypted ----->

| orig IP hdr | ESP hdr | TCP | Data | ESP trlr | ESP auth |
|---|---|---|---|---|---|
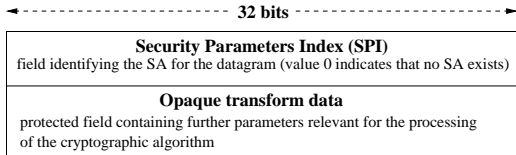
encrypts only the data portion (payload) of each packet, but leaves the header untouched.

**Tunnel mode:**

<-------------- Authenticated -------------->
<----------- Encrypted ----------->

| New IP hdr | ESP hdr | orig IP hdr | TCP | Data | ESP trlr | ESP auth |
|---|---|---|---|---|---|---|
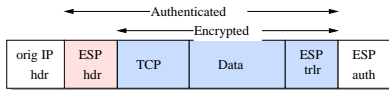
Entire IP datagram encapsulated within the ESP. Therefore both header and payload encrypted (and optionally authenticated).
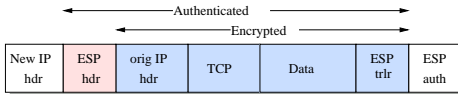
**Transport mode** provides
end-to-end encryption
between hosts supporting
IPSec.
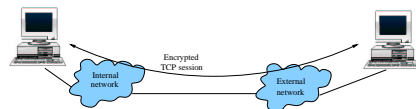
**Tunnel mode** can be used to
set up a VPN.
Here hosts on different
networks use Internet over
tunnels between security
gateways. Hosts needn't
implement security
capabilities.
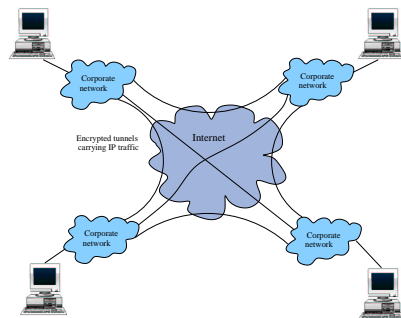
# Outline
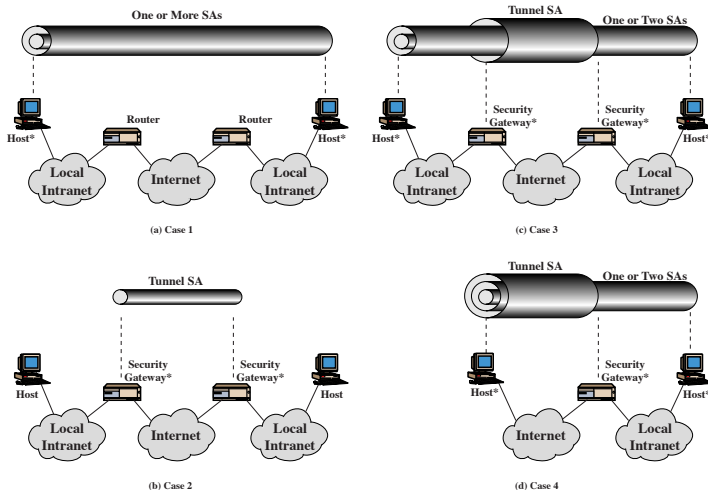
**Figure 16.10 Basic Combinations of Security Associations**

# The Internet Key Exchange (IKE) protocol

- IKE establishes not just keys, but Security Associations:
  - the protocol format used (many),

  - the cryptographic and hashing algorithm used,

  - and, of course, keys...

- IKE is very flexible. E.g., supports authentication based on a variety of pre-shared secrets (master keys).

- But also very complex. Many options, alternative subprotocols, ...

- IKE evolved from a number of different protocols, including:
  - ISAKMP (Internet Security Association and Key Management Protocol): provides a framework and a generic negotiation protocol for establishing SAs and cryptographic keys, but does not prescribe any particular authentication mechanism.
  - Oakley: a suite of key agreement protocols in which two parties generate a key jointly.

- Roughly speaking, IKE combines packet formats of ISAKMP and exchanges of OAKLEY, which are based on Diffie-Hellman.

- We start by reviewing Diffie-Hellman and considering extensions.

# The Diffie-Hellman protocol



**I**

Choose g, p
Generate x
Compute $X = g^x \bmod p$

(1) $X [,g,p]$

**R**

Generate y
Compute $Y = g^y \bmod p$

(2) $Y$

Compute $k = Y^x \bmod p$

Compute $k = X^y \bmod p$
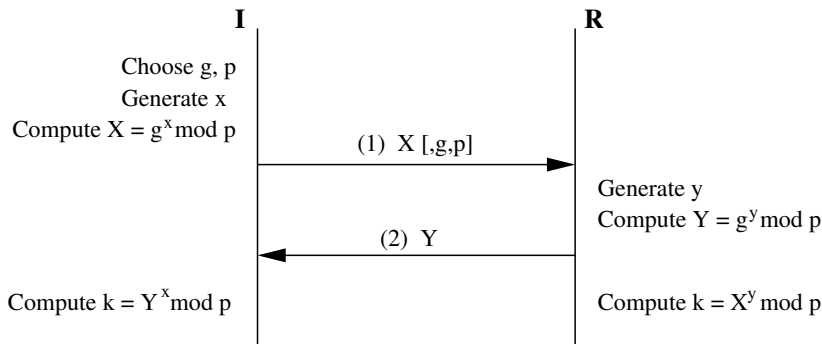
**Basic Diffie-Hellman key-exchange**: initiator I and responder R exchange "half-keys" to arrive at mutual session key $k$.

- Denial of Service (DoS) attack on R via flooding:

Attacker sends series of
request packets, each with
different spoofed source IP
address, so that R must
process each request.
Expensive exponentiation
and storage (of $y$s).

```
                              I                          R
                        Choose g, p
                      Generate series
                     of random numbers
                     X₁, X₂, ..., Xₙ
                                  (1)  X₁[,g,p]
                                  ───────────────────▶
                                                        Generate y₁
                                                        Compute Y₁ = g^{y₁} mod p
                                  (2)  X₂[,g,p]
                                  ───────────────────▶
                                                        Generate y₂
                                                        Compute Y₂ = g^{y₂} mod p
                                  (n)  Xₙ[,g,p]
                                  ───────────────────▶
                                                        Generate yₙ
                                                        Compute Yₙ = g^{yₙ} mod p
```
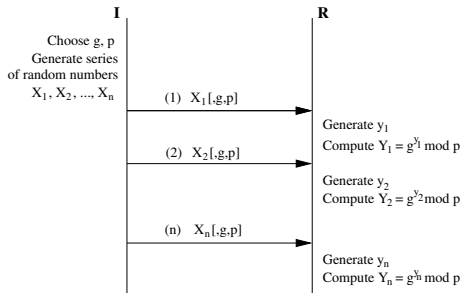
- Weak forms of protection available, e.g.,
    - Demand a response from a claimed address.
    - Make initiator perform some computation.

## Cookies against DOS

**I**                                                                 **R**

Generate $C_I$

       (1) $C_I$ →

     ← (2) $C_I$, $C_R$         Generate $C_R$

Choose g, p
Generate x
Compute $X = g^x \bmod p$

    (3) $C_I$, $C_R$, X [,g,p] →

                       Validate cookie
                       Generate y
                       Compute $Y = g^y \bmod p$

     ← (4) $C_I$, $C_R$, Y

Validate cookie                   Compute $k = X^y \bmod p$
Compute $k = Y^x \bmod p$
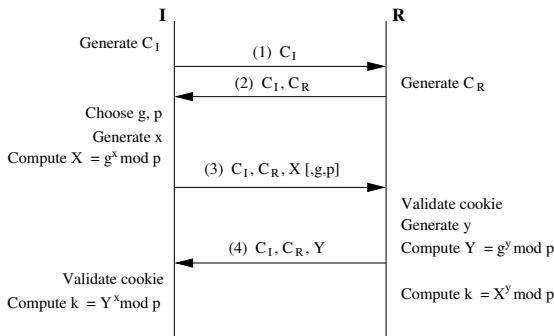
- I and R send "cookies" $C_I$ and $C_R$ to partner.
  Cookies are either randomly generated numbers, or even better,
  stateless, e.g., $C = \text{hash(IP address, secret)}$.
- Attacker must be at address and complete a cookie exchange for
  each address it spoofs.

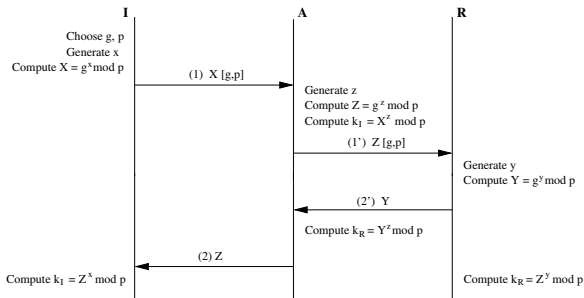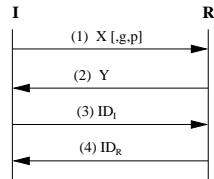# Diffie Hellman – man-in-the-middle

- Unauthenticated keys $\implies$ open to man-in-the-middle attacks



```
        I                        A                        R
   Choose g, p
   Generate x
 Compute X = g^x mod p
                    (1)  X [g,p]
                   ───────────────▶  Generate z
                                     Compute Z = g^z mod p
                                     Compute k_I = X^z mod p
                                            (1')  Z [g,p]
                                           ───────────────▶  Generate y
                                                             Compute Y = g^y mod p
                                            (2')  Y
                                           ◀───────────────
                                     Compute k_R = Y^z mod p
                    (2)  Z
                   ◀───────────────
 Compute k_I = Z^x mod p                                     Compute k_R = Z^y mod p
```

- Defend by adding two authentication steps.
    - $ID_I$ and $ID_R$ are digitally signed messages binding half keys to sender.
    - May also be encrypted with computed DH key.



```
    I                    R
         (1)  X [,g,p]
        ───────────────▶
         (2)  Y
        ◀───────────────
         (3)  ID_I
        ───────────────▶
         (4)  ID_R
        ◀───────────────
```

## But if you already share keys ...

- Why bother with Diffie-Hellman?

- Answer: perfect forward secrecy
  If someone records the entire conversation and later discovers
  Alice's and Bob's private keys, you don't want them to be able to
  decrypt everything.

- Example without PFS: SSL
  Alice chooses a secret, encrypts it with Bob's PK and rest of the
  session is protected based on that secret.

- Moreover, periodic generation of new keys and keying material
  complicates cryptanalysis.