

# The IpSec architecture

Security in Networked Computing Systems

## Roadmap



- Basic architecture
- Tunnel and transport mode
- Encapsulating Security Payload (ESP)  
Authentication Header (AH)
- Internet Key Exchange (IKE)



# IpSec in a nutshell

- IpSec is an IETF proposal for security at IP level
  - RFC 2041, 2042, 2046, 2048
- IpSec is based on IP (raw socket) and is compliant with
  - IPv4 (optional): protocol field;
  - IPV6 (mandatory): next header
- IpSec makes it possible to
  - Establish [secure end-to-end channels](#)
  - Create [VPN](#) over public networks



# Security services

- Integrity of datagrams
- Origin authentication
- Anti-replay mechanism
- Confidentiality
  - Partial confidentiality of traffic flow



# Protocolli in IPsec

- IPsec is composed of three protocols
- Authentication Header (AH)
  - Packet authentication
- Encapsulating Security Payload (ESP)
  - Packet confidentiality and authentication
- Internet Key Exchange (IKE)
  - Negotiation of security parameters
  - Authentication and key exchange

# Services & Protocols

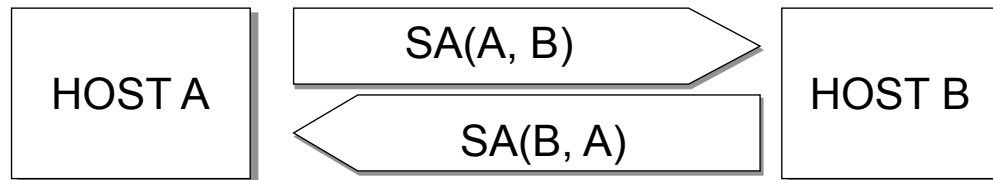


	AH	ESP (cifr.)	ESP (cifr. & aut.)
<b>Datagram integrity</b>	X		X
<b>Data origin authentication</b>	X		X
<b>Anti-replay</b>	X	X	X
<b>Confidentiality</b>		X	X
<b>Partial confidentiality</b>		X	X

# Security Association (SA)



- Unidirectional association between two hosts  
(you need two associations for bidirectional security)



- In an IP pkt, an SA is identified by three parameters
  - Security Parameter Index (SPI)
  - IP Destination Address
  - Security Protocol Identifier (AH or ESP)

## Parameters of an SA



- Sequence number counter (32 bit counter, mandatory)
- Sequence counter overflow (flag, mandatory)
- Anti-replay window
- AH information
- ESP information
- Lifetime
- IPsec protocol mode (tunnel, transport, wildcard)
- Path MTU

# Traffic ↔ Security Associations

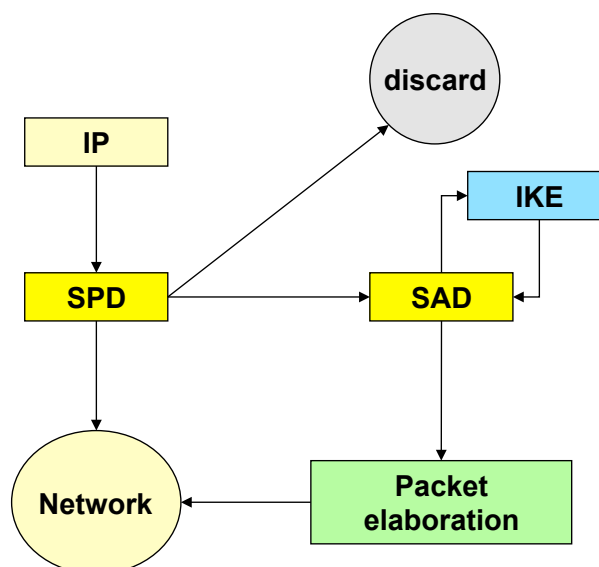


- How does IPSec associate traffic flows to security associations?
  - Security Policies Database
  - Security Associations Database

## Database di sicurezza



- **Security Association Database**
  - SAD specifies parameters associated to each SA
- **Security Policy Database**
  - SPD relates *outgoing* traffic portions to SAs
  - A traffic portion may be associated to none, one or more SAs



# Security policies: examples



- All traffic towards 192.168.2.3 must be protected by ESP in transport mode using DES-CBC
- All FTP traffic (TCP/20) towards 192.168.2.3 must be protected by ESP in tunnel mode using 3DES-CBC
- All traffic towards 192.168.2.3 must be not protected
- All traffic towards 192.168.2.3 must be discarded

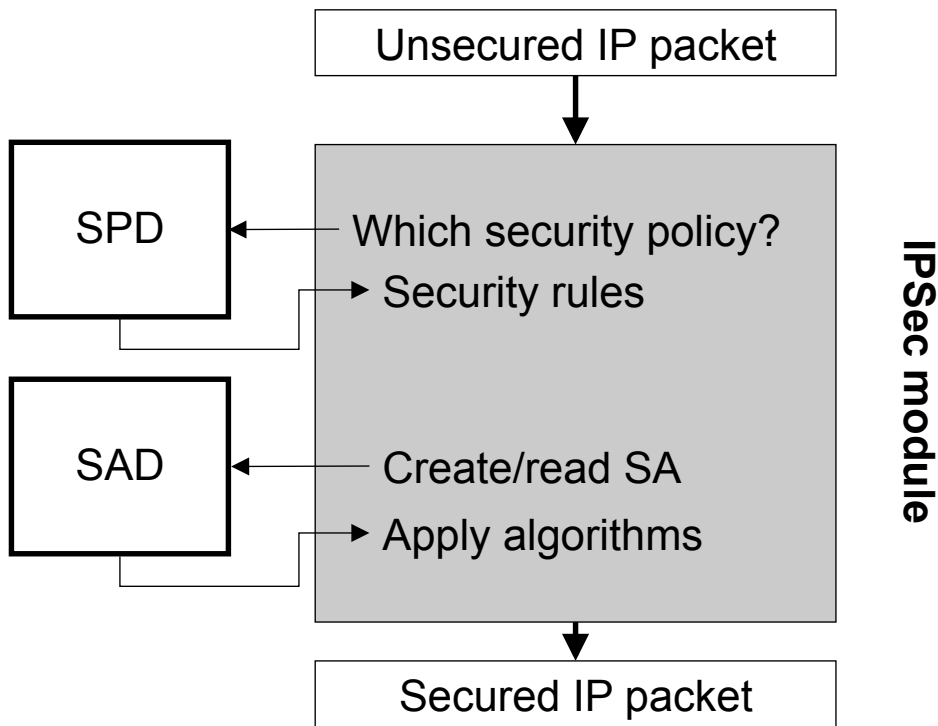
# SPD: selectors



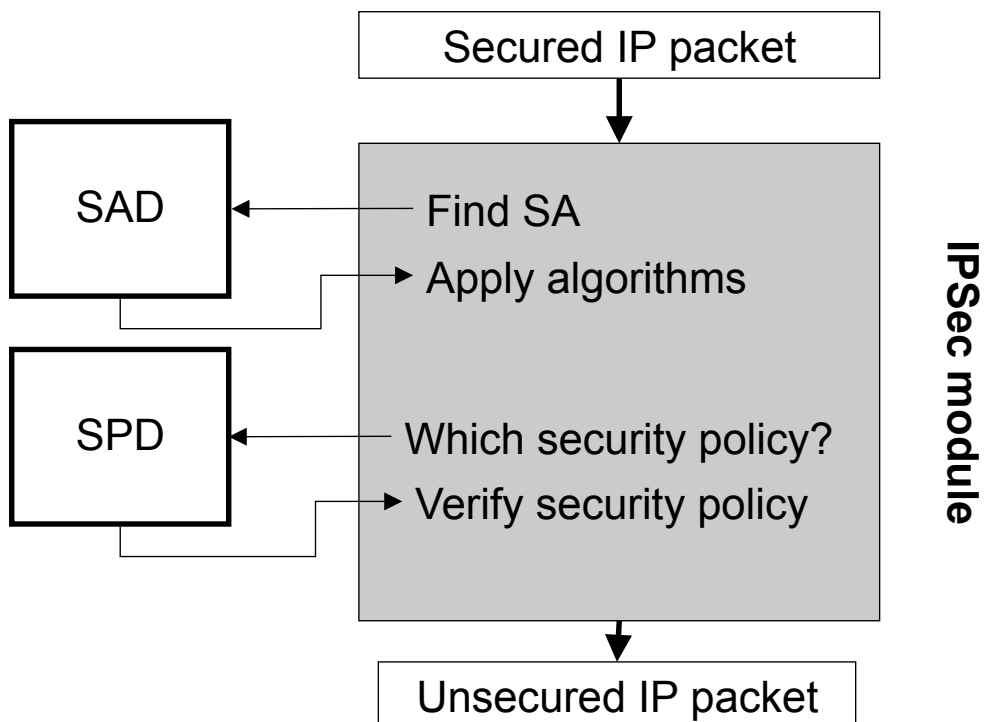
- SPD contains **policy entries**, each of which specifies an IP traffic portion and the SA for that portion
- An IP traffic portion is specified by **selectors**
  - Destination IP Address
  - Source IP Address
  - Userid
  - Data Sensitivity Level (Classified,...)
  - Transport Layer Protocol
  - Ipv6 Class
  - Ipv6 Flow Label
  - TOS, Ipv4 Type Of Service



# Sending a packet



# Packet reception

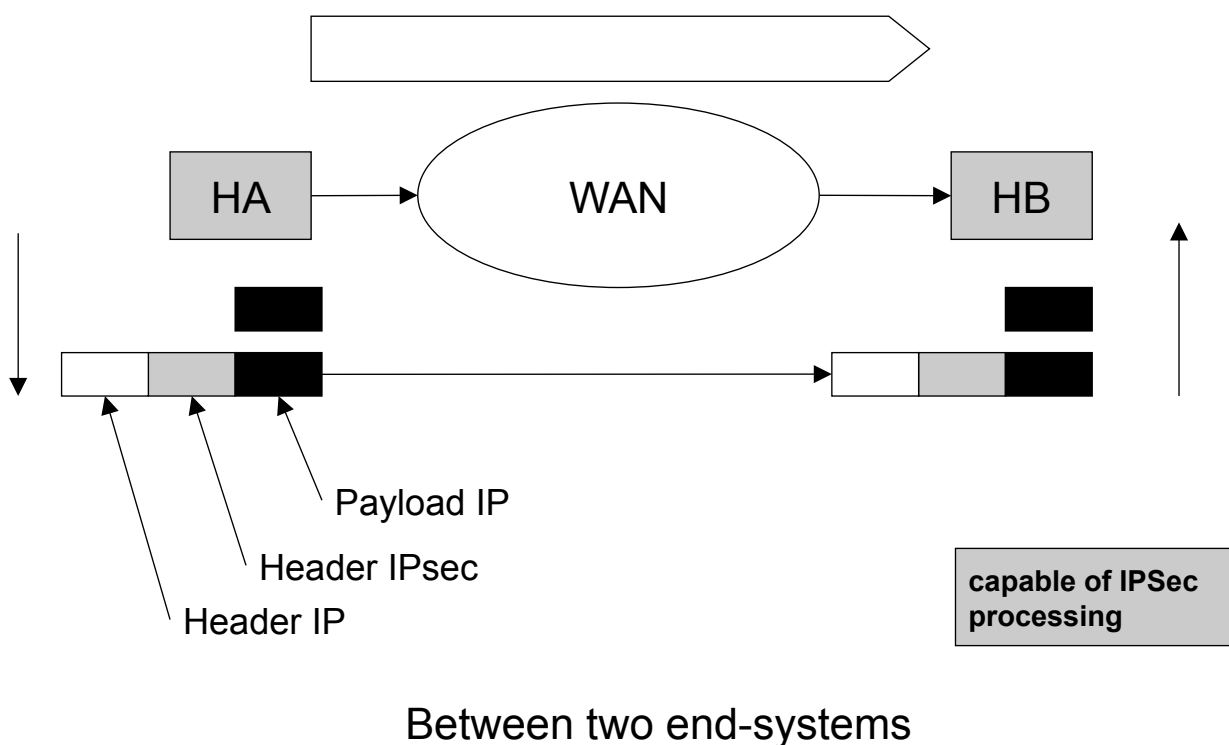




# IPSec modes

- Transport mode
  - Protect the packet payload
  - Typically, e2e secure communication between two hosts (client, server)
- Tunnel mode
  - Protect the entire packet
  - Original packet is encapsulated within an outer packet
  - Typically, a secure tunnel between two security GWs (firewalls, ipsec-enabled routers,...)

## IPsec Transport mode

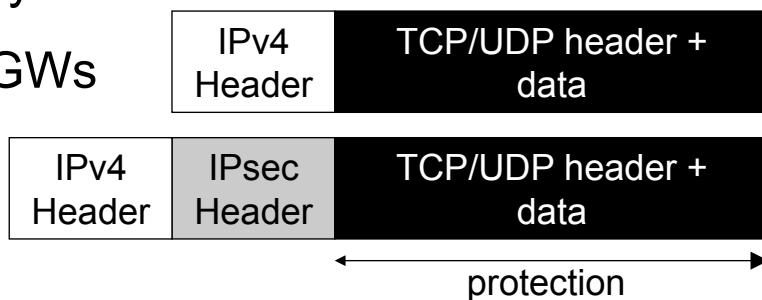




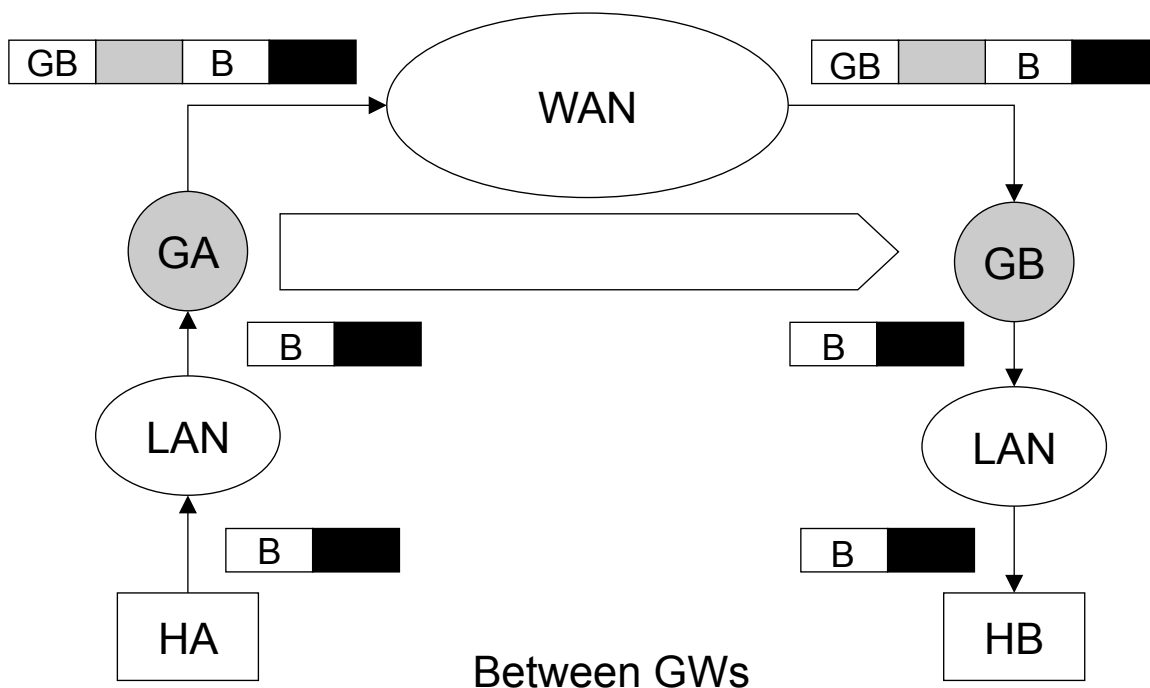


# IPsec Transport mode

- Give protection to higher level protocols (e.g., TCP, UDP, SNMP, ICMP)
  - No protection to variable fields of IP header
  - Network addresses in the IP header are not modified (they must be routable)
- End-to-end security
- It doesn't involve GWs (besides traffic to GWs)

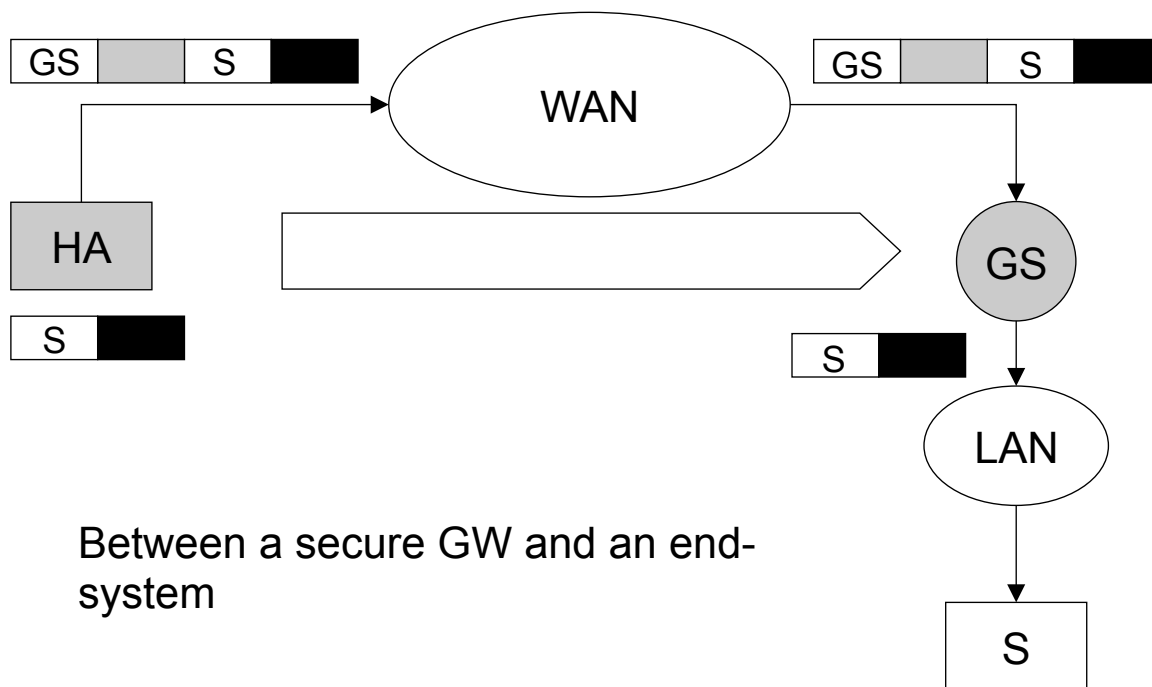


# IPsec Tunnel mode





# IPsec Tunnel mode

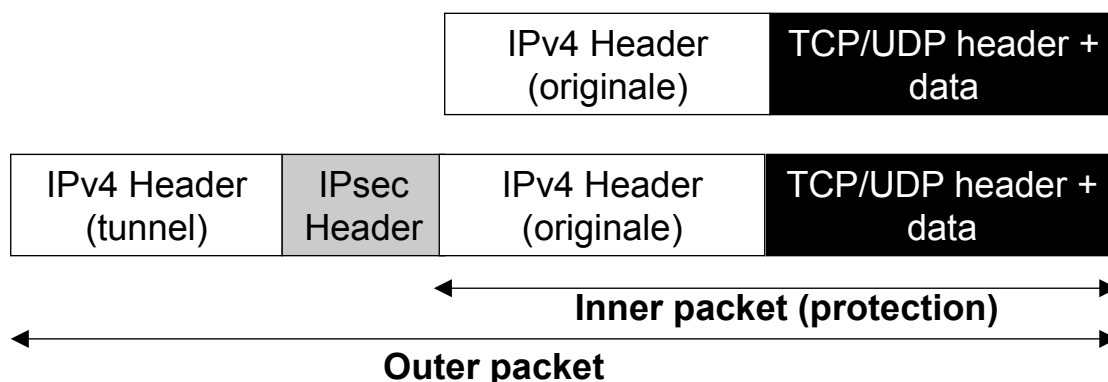


Between a secure GW and an end-system

# Ipsec Tunnel mode



- Always used when one host is a gateway
- Protect the whole original packet including the variable fields
- Addresses in outer pkt's header may be different from those in the inner pkt's header



# Functionalities



	Transport mode	Tunnel mode
<b>AH</b>	Authenticate payload and selected portions of IP header and extension headers (IPv6)	Authenticate the entire inner pkt and selected portions of the outer pkt
<b>ESP</b>	Encrypt the payload and extension headers (IPv6)	Encrypt the entire inner pkt
<b>ESP + auth</b>	Encrypt the payload and extension headers (IPv6) Authenticate payload (but not header)	Encrypt inner pkt Authenticate inner pkt

# Header IPv4



0	4	8	16	31
versione	IHL	TOS	Total Length	
Identification			Flags	Header checksum
TTL	Protocol		Header Checksum	
Source IP Address				
Destination IP Address				
Options				Padding



# Header IPv4

- Src and dst IP addresses [32 bit]
- Internet Header Length (IHL) [32 bit]
- Type Of Service (TOS)
- Length (sizeof (pkt) in bytes)
- Identification: pkt ID (for fragmets)
- Flags: may/don' t fragment; last/more fragments
- Time To Live (TTL): number of hops
- Protocol: protocol used by payload
- ...*other fields*...



IPSec

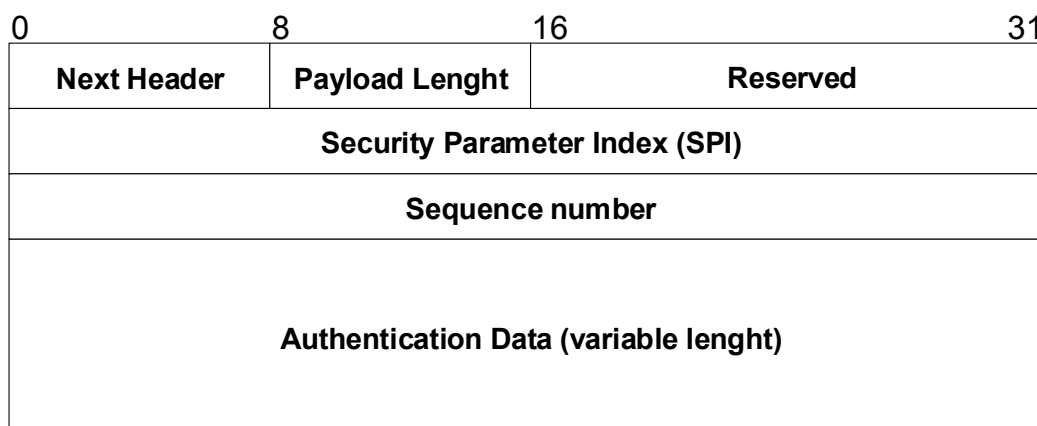
## AUTHENTICATION HEADER



# Authentication Header (AH)

- IPv4/IPv6 extension for pkt authentication (IP protocol 51, RFC-2402)
- Offered services
  - Pkt authentication
  - Data integrity
  - Data origin authentication
  - anti-replay
- Algoritmi utilizzati: HMAC
  - HMAC-MD5-96
  - HMAC-SHA-1-96

## Formato di AH

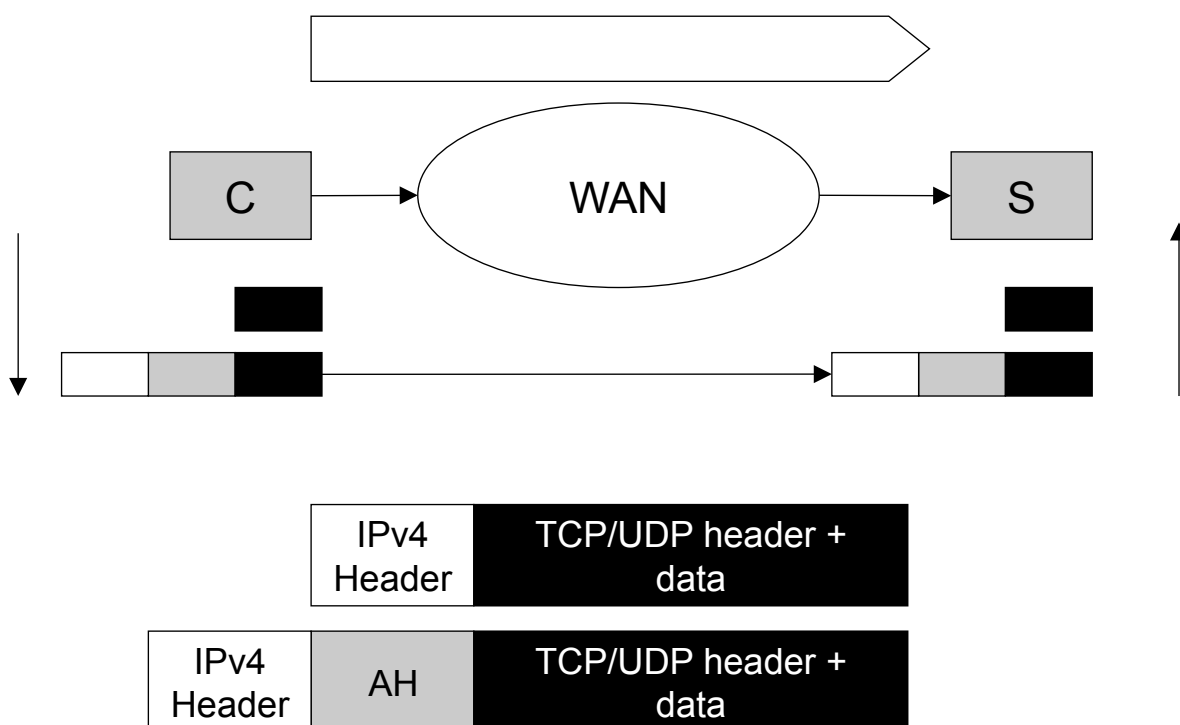




# AH format

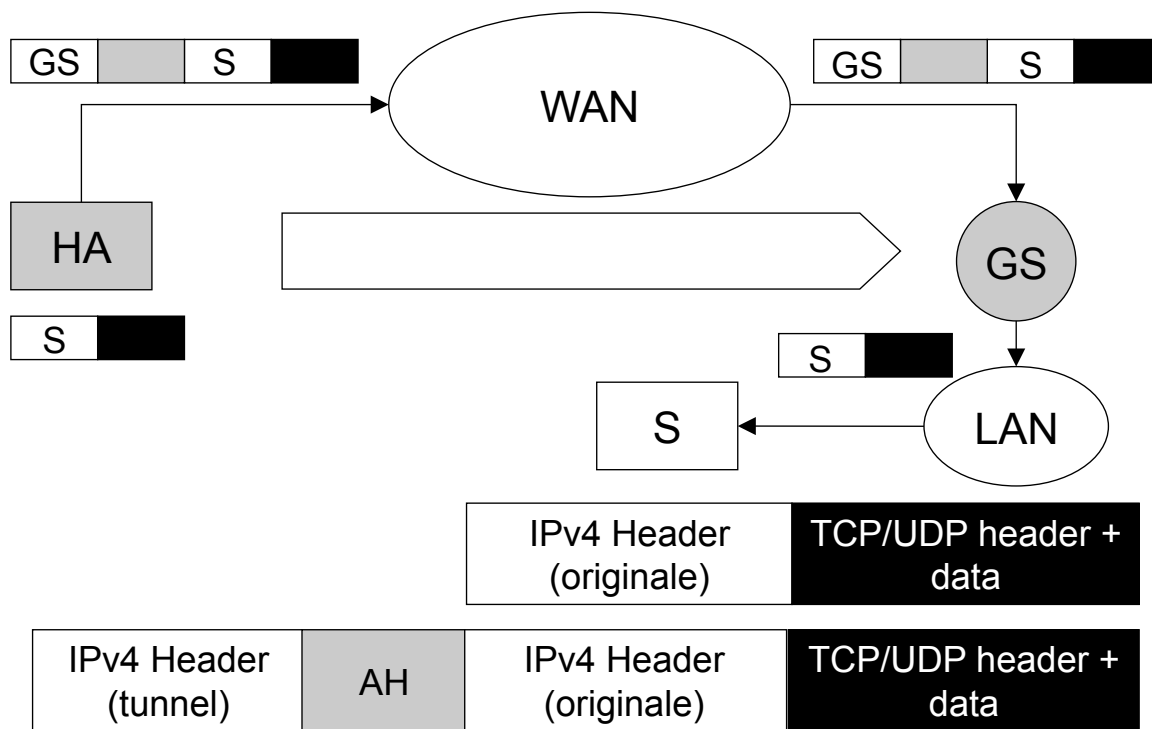
- **Next Header** (8 bit) specify the next header
- **Payload Length** (8 bit) length of AH in 32-bit word minus two
- **Reserved** (16 bit) future uses
- **SPI** (32 bit)
- **Sequence number** (32 bit)
- **Authentication data**: contains Integrity Check Value
- **Variable length**
  - multiple of 32 bits (default 96)
  - MAC or a truncated version (96 bit)

## AH: Transport mode

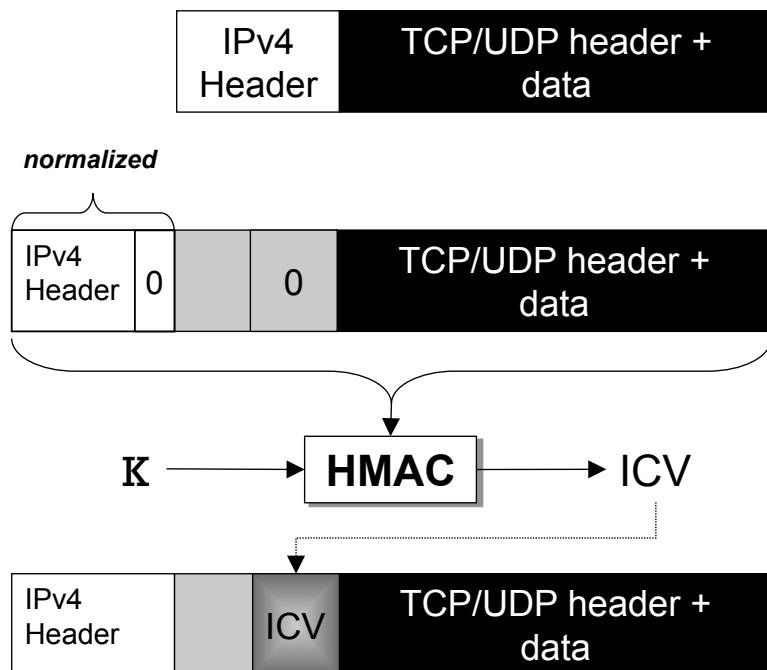




# AH: tunnel mode



# How AH is built

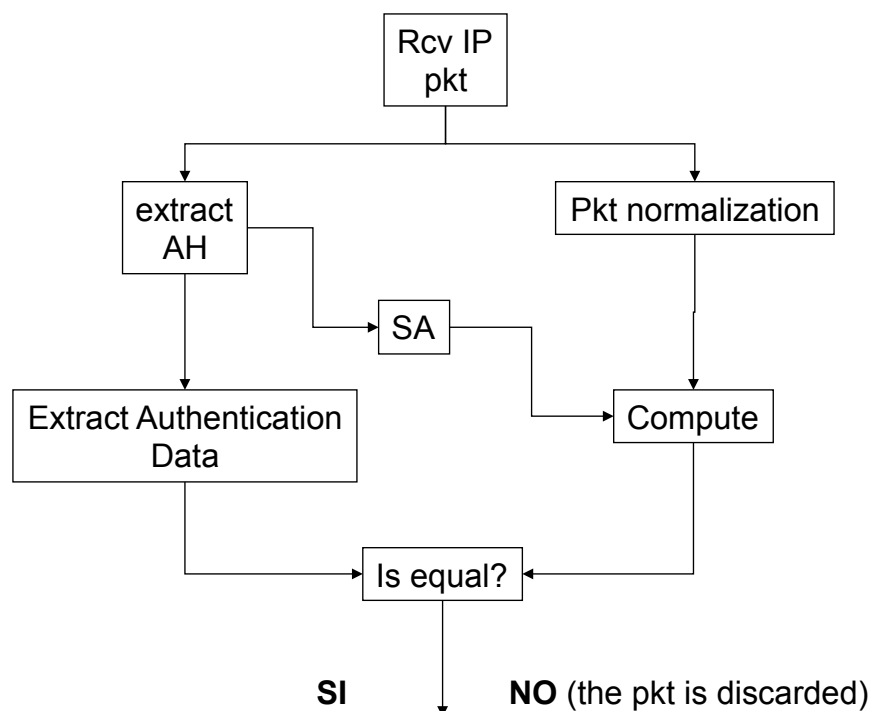




# How AH is built

1. Authentication Data is initially zeroed
2. Append payload
  - Transport mode: IP pkt' s payload
  - Tunnel mode: the whole IP pkt
3. Normalize IP header (mutable fields are zeroed)
  - Mutable fields (es. TTL, hop count) are restored after MAC computation
4. Compute ICV of normalized IP header, AH and payload
5. Copy ICV into the Authentication Data field

# Data origin authentication





# Protection level of AH



- In transport mode, authentication covers
  - The entire original IP pkt but mutable fields in the original IP pkt's header
- In tunnel mode, authentication covers
  - The entire inner IP pkt and
  - Header of the outer pkt but mutable fields

# Anti-replay

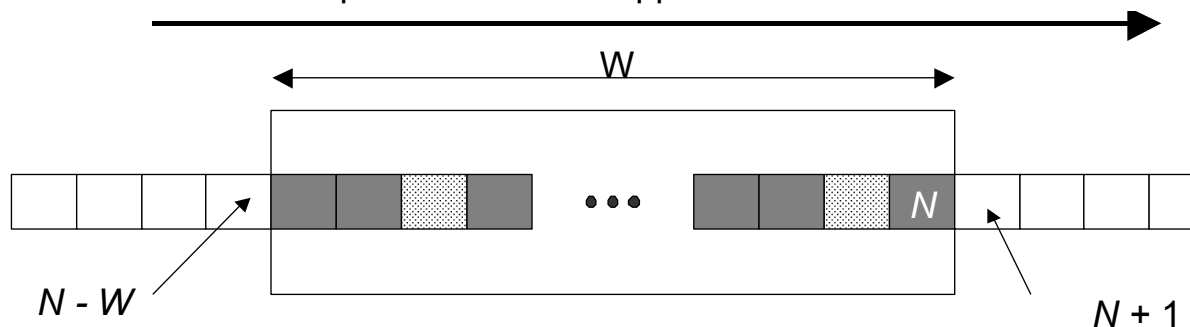




- Upon SA creation, src initializes counter to zero
- Before sending a pkt, src increments counter by one and copies value in the pkt's Sequence Number field
  - When the counter wraps around, src terminates and renegotiate the SA
- As IP does not guarantee delivery and ordering, rcv implements a window mechanism
  - Default window length is 64



# Anti-replay Window

If a pkt falls **outside of the window, on the right**, the window is shifted forward and the pkt becomes the upper bound



-  Valid pkt already received
-  Valid pkt not received yet

A non-valid pkt or a pkt that falls **outside the window, on the left**, is discarded



IPSec

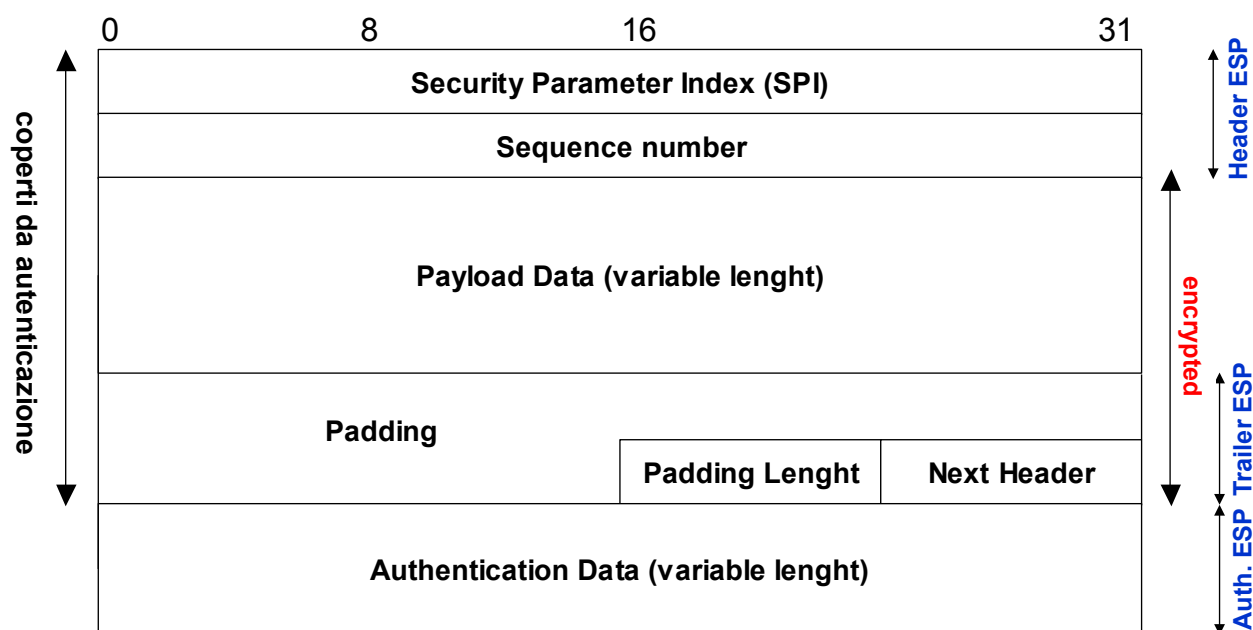
## ENCAPSULATING SECURITY PAYLOAD

# Encapsulating Security Payload (ESP)



- IPv4/IPv6 extension for pkt encryption (IP protocol 50, RFC-2406)
- Services
  - Confidentiality of payload
  - Partial confidentiality of traffic
  - Optional authentication limited to payload
- Algorithms
  - DES in CBC mode (required)
  - 3DES, RC5, IDEA, 3IDEA, CAST, Blowfish (optional)
  - HMAC-MD5-96, HMAC-SHA-1-96

## ESP format





## ESP format (→)

- *Security Parameters Index* (SPI)
- *Sequence number* (32 bit): anti replay
- *Payload data* encrypted data
  - A possible initialization vector is placed at the beginning of the field
- *Padding* (0 – 255 byte)
- *Padding length*
- *Next Header* (8 bit): type of data in the payload
- *Authentication Data*: ICV

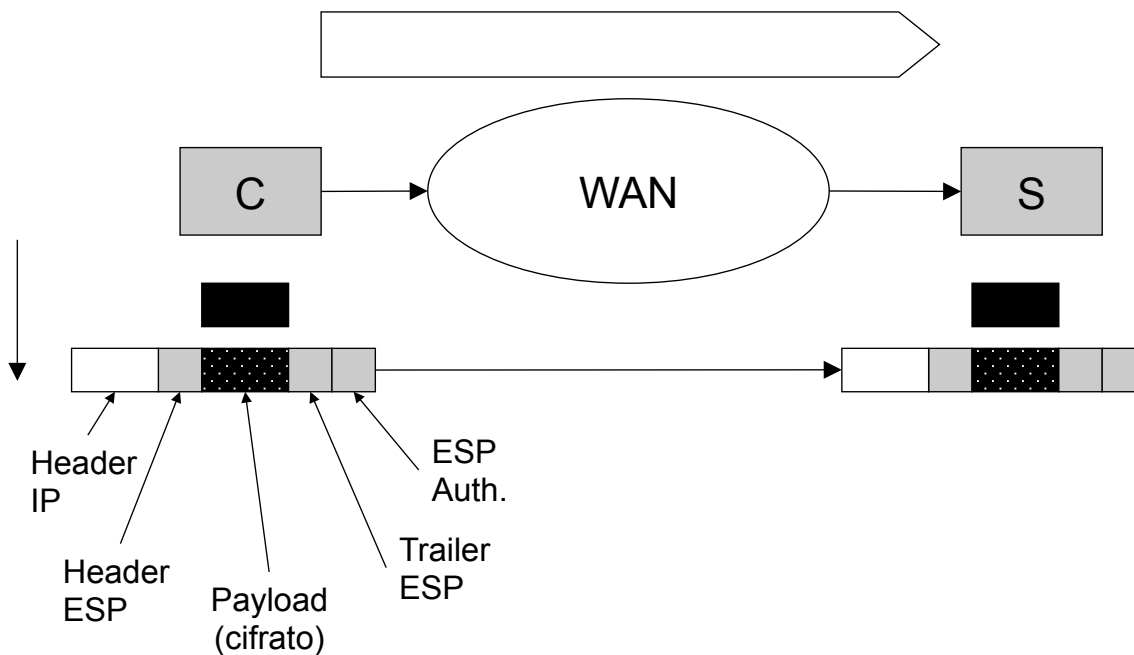


## ESP format (↓)

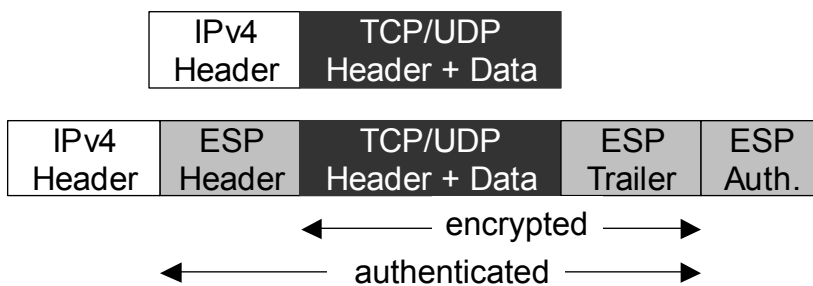
- *Header ESP* is composed of
  - SPI
  - Sequence Number
- *Trailer ESP* is composed of
  - Padding,
  - Padding Length
  - Next Header
- *Authentication ESP* contains Authentication Data



# ESP Transport mode



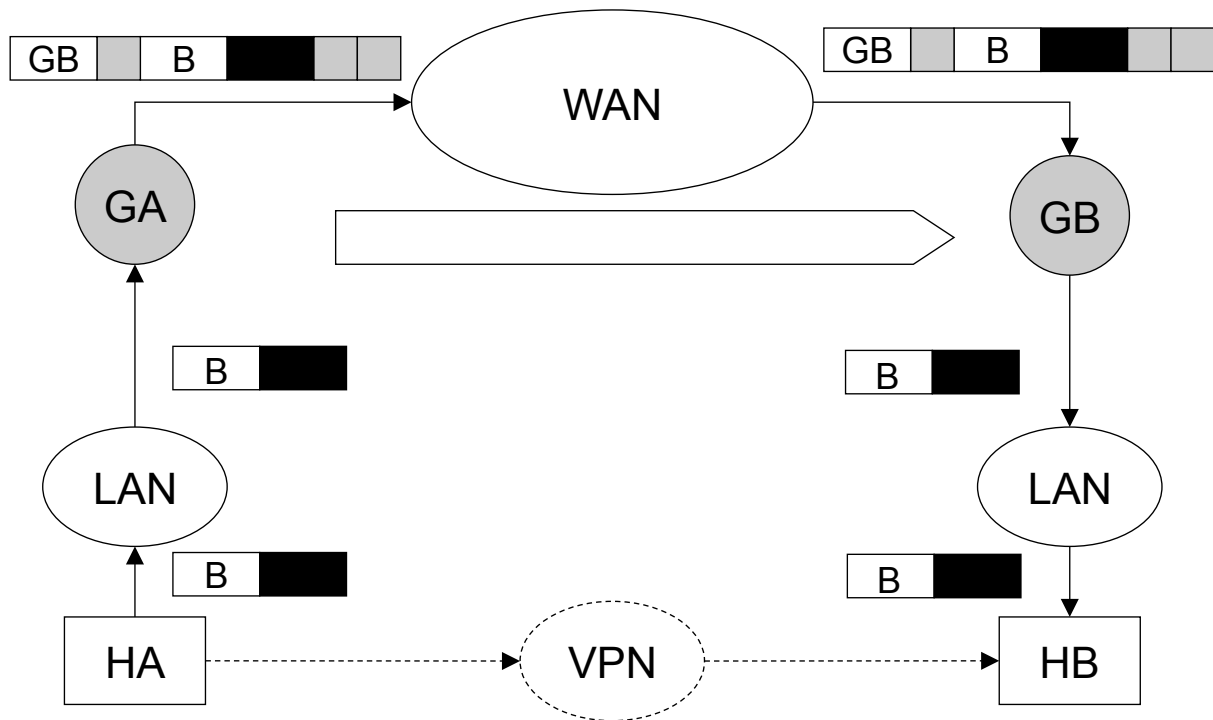
# ESP in modalità Trasporto



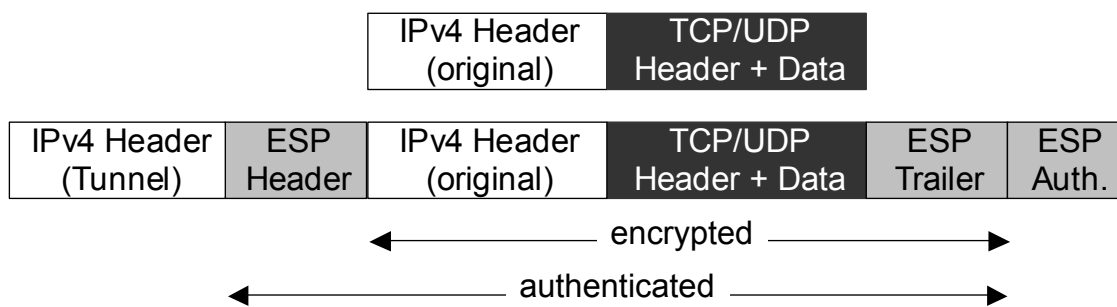
- Authentication does not cover IP header
- ESP provides a smaller authentication coverage than AH



# ESP Tunnel mode



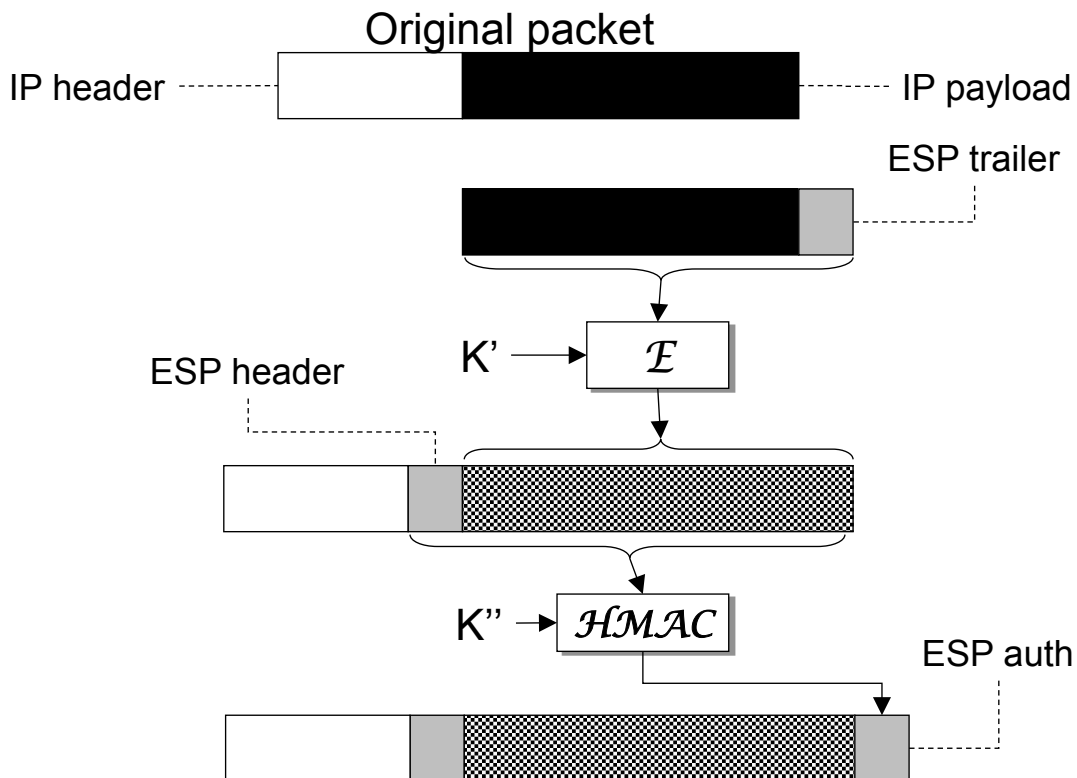
# ESP tunnel mode



- Authentication does not protect the outer ip pkt header
- ESP provides a smaller coverage than AH



# How ESP header is built



# How ESP header is built

1. Build header ESP
2. Append the payload
  - Transport mode: the original IP pkt' s payload
  - Tunnel mode: the entire original IP pkt
3. Build ESP trailer ESP and append it to the payload
4. Encrypt payload and trailer ESP
5. If required, compute HMAC of ESP header and cyphertext, and append resulting ICV into the ESP Authentication

# On padding



- Encryption algorithm requires that clear-text length is a multiple of the block size
- ESP requires that padding length and next header are aligned at the MSB of a 32-bit word
- Padding may be inserted in order to not reveal the actual payload length



## COMBINATION OF SECURITY ASSOCIATIONS



# Bundles



- Security Policy.
  - A military application requires: (i) payload is encrypted; (ii) field *Options* of the IP header specifies *sensitivity labels*; (iii) it is not possible to modify the *Options* field. Notice that neither AH nor ESP, separately, is sufficient to implement this policy.
    - AH does not encrypt the payload
    - ESP does not authenticate the IP header
- An SA **bundle** is a combination of two or more SAs
  - Algorithms of SAs are sequentially applied to the same packet

# Combinations of SAs



- SAs can be combined in a bundle as follows
  - **transport adjacency** – apply several security protocols without any tunnelling
    - Only one level of combination; further levels of combinations are useless
  - **iterated tunnelling** – apply several security protocols through tunnelling as each tunnel may have src or dst at a different IPsec host
  - A combination of both approaches is possible
  - Four combinations are mandatory

# Example: authentication and confidentiality

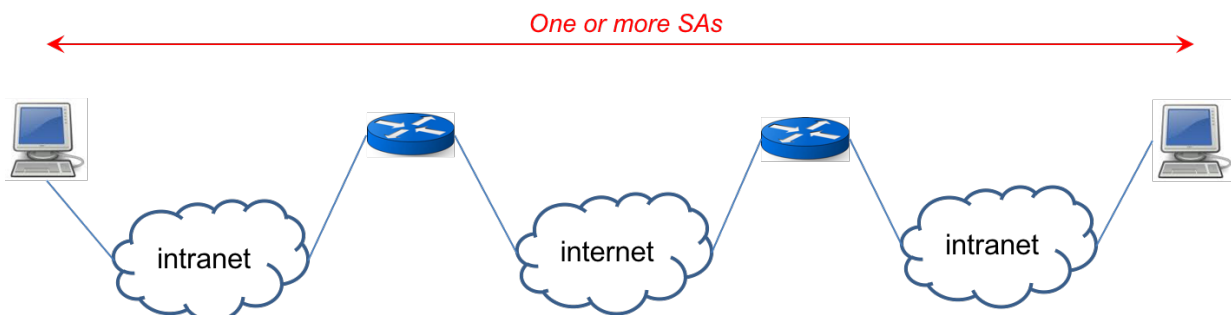


- **ESP with authentication option**
  - Transport mode
  - Tunnel mode
  - *Authentication applied to cipher-text*
- **Bundle**
  - **Bundle transport adjacency**
    - *Inner*: ESP (without authentication); *outer*: AH
    - *Authentication protects ESP and original IP header (but mutable fields)*
  - **Bundle transport-tunnel**
    - *Inner*: AH transport; *outer*: ESP tunnel
    - *Authentication is applied to clear-text*

## Basic combinations



### Case 1: end-to-end security between hosts



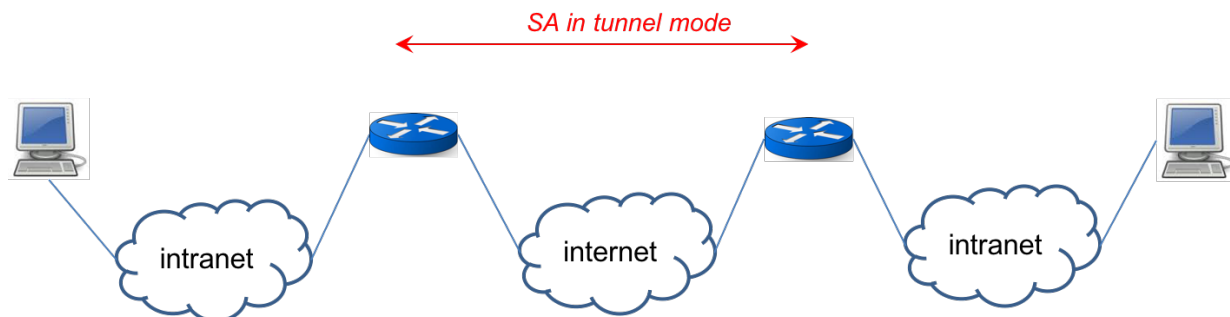
### Combinations

1. AH in transport mode
2. ESP in transport mode
3. Transport adjacency: *inner*: ESP; *outer*: AH
4. Combination 1|2|3 within AH|ESP tunnel



# Basic combinations

## Case 2: security between gateways (simple VPN)



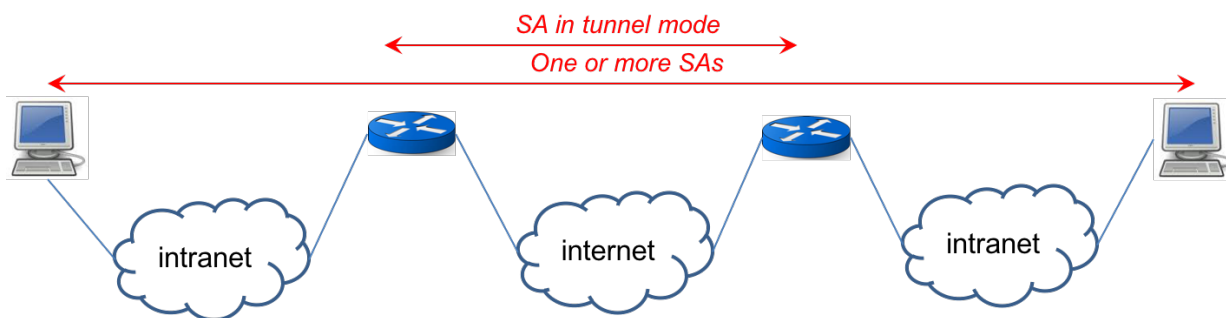
### Combination

- A single SA in tunnel mode



# Basic combinations

## Case 3: end-to-end security + security between gateways



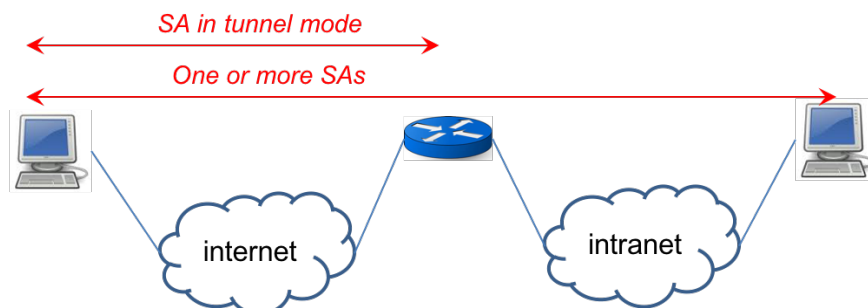
### Combinations

- The same combinations as Case 1 and 2



# Basic combinations

## Case 4: remote host



### Combinations

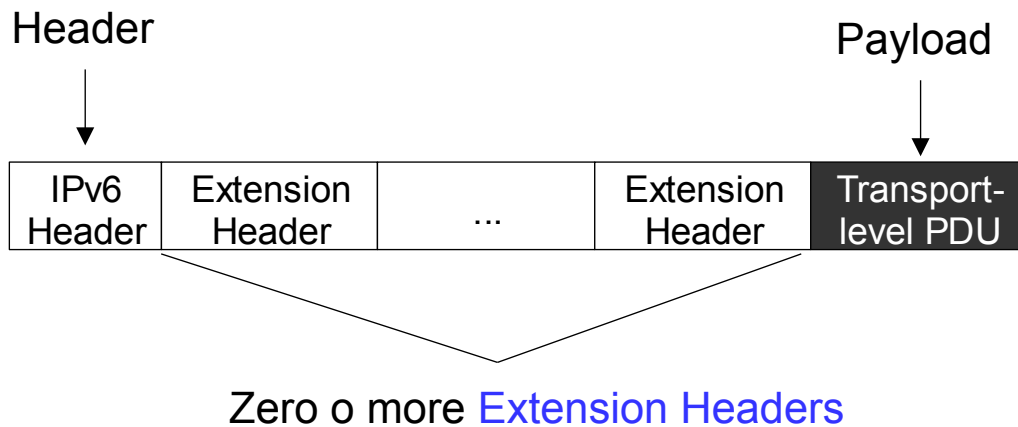
- A tunnel between the host and the dst gw
- Combinations 1a, 1b, 1c between hosts



# IPv6



## Structure of an IPv6 packet



# IPv6 ed IPsec



## Types of Extension Header:

1. Hop-by-hop options header
2. Routing header
3. Fragment header
4. Authentication Header
5. Encapsulating Payload Header
6. Destination options header





IPSec

# INTERNET KEY EXCHANGE (IKE)

IpSec

16/05/16

63

## Internet Key Exchange (IKE) – RFC 2409



IKE implements the following functions

- Negotiation of security parameters
- Authentication
- Key establishment
- Key management (after establishment)
- UDP/500

IpSec

16/05/16

64



# IKE protocol suite

IKE is composed of

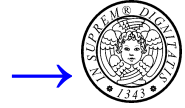
- **SKEME**
  - PK-based authentication protocol
- **OAKLEY**
  - DH-based key establishment protocol
- **ISAKMP**
  - Internet Security Association and Key Management Protocol
  - Framework for key management



## IKE: phases

- Initial event
  - A peer generates or receives a portion of data traffic that has to be IPsec protected
- Peers activate IKE
  - IKE – PHASE 1: peers negotiate and establish a **secure channel**
  - IKE – PHASE 2: peers use the **secure channel** to negotiate and establish 2 SAs
- Now, peers can generate data traffic

# IKE Phase 1



- Goals
  - Negotiation
  - Authentication
  - Key distribution
- Modes
  - main mode (6 messages)
  - aggressive mode (3 messages)

# IKE Phase I



- Main implementations of main and aggressive modes
  - main mode using preshared key authentication
  - main mode using digital signature authentication
  - aggressive mode using preshared key authentication
- But also
  - main mode using encrypted nonces authentication
  - aggressive using digital signature authentication



# IKE Phase 2



- Obiettivi
  - Negotiation
- Modes
  - Quick mode (3 messages)

# Diffie-Hellman protocol



- Pros
  - Secret key is generated only when needed
  - Key establishment does not require any pre-existing infrastructure but the knowledge of global parameters ( $p$  and  $g$ )
- Cons
  - No guarantee on peer identity
  - Subject to MiM
  - Computationally demanding



# Oakley protocol

- The protocol has been conceived to maintain DH pros while overpassing DH cons
- Protocol features
  - Use cookie to contrast clogging attacks (DoS)
  - Allow group negotiation (p, g)
  - Support anti-replay
  - Authenticate the exchange of DH pubK so avoiding MiM

# Clogging attack (SDos)



## Diffie-Hellman protocol

$$\begin{array}{l}
 A \rightarrow B: A, g^a \text{ mod } p \\
 B \rightarrow A: B, g^b \text{ mod } p
 \end{array}$$

## The clogging attack

- Adversary M masquerades as Alice and makes Bob to repeatedly perform mod exp so exhausting his computational resources

$$\begin{array}{l}
 A[M] \rightarrow B: A, X \\
 B \rightarrow A: B, g^b \text{ mod } p
 \end{array}$$



# Cookie: anti-clogging mechanisms

A cookie is a nonce

$$A \rightarrow B: \text{cookie}_A$$

$$B \rightarrow A: \text{cookie}_B$$

Bob computes mod exp iff he sees cookieB in M3

$$A \rightarrow B: A, \text{cookie}_A, \text{cookie}_B, (g^a \text{ mod } p)$$

$$B \rightarrow A: B, \text{cookie}_A, \text{cookie}_B, (g^b \text{ mod } p)$$

An adversary can only make B to communicate his cookie

$$A[M] \rightarrow B: \text{cookie}_A$$

$$B \rightarrow A: \text{cookie}_B$$

$$A[M] \rightarrow B: A, \text{cookie}_A, \text{cookie}_B, (g^a \text{ mod } p)$$

$$B \rightarrow A: B, \text{cookie}_A, \text{cookie}_B, (g^b \text{ mod } p)$$

ISAKMP RFC recommends that a cookie (8 byte) is implemented as follows

$$\text{cookie} = H(IP_{\text{dest}}, IP_{\text{source}}, Port_{\text{dest}}, Port_{\text{source}}, \text{random number}, \text{time stamp})$$

## Phase I: main mode (preshared key authentication)



### DH-based protocol Pre-Shared Key: $PSK_{ab}$

1.  $A \rightarrow B: c_a$
2.  $B \rightarrow A: c_b$
3.  $A \rightarrow B: c_a, c_b, X_a, N_a$
4.  $B \rightarrow A: c_a, c_b, X_b, N_b$
5.  $A \rightarrow B: c_a, c_b, \{A, h(SKEYID\_a, c_a, c_b, PSK_{ab}, \text{prev. msg.}, A)\}_{SKEYID\_e}$
6.  $B \rightarrow A: c_a, c_b, \{B, h(SKEYID\_a, c_a, c_b, PSK_{ab}, \text{prev. msg.}, B)\}_{SKEYID\_e}$

cookie  $c_a, c_b$ ; pre-shared key:  $PSK_{ab}$

$$SKEYID = PRF(PSK_{ab}, N_a, N_b)$$

$$SKEYID\_d = PRF(SKEYID, g^{ab}, c_a, c_b, 0)$$

$$SKEYID\_a = PRF(SKEYID, SKEYID\_d, g^{ab}, c_a, c_b, 1)$$

$$SKEYID\_e = PRF(SKEYID, SKEYID\_d, g^{ab}, c_a, c_b, 2)$$

# Quick mode



- DH-based key establishment
- Use keys defined in Phase I

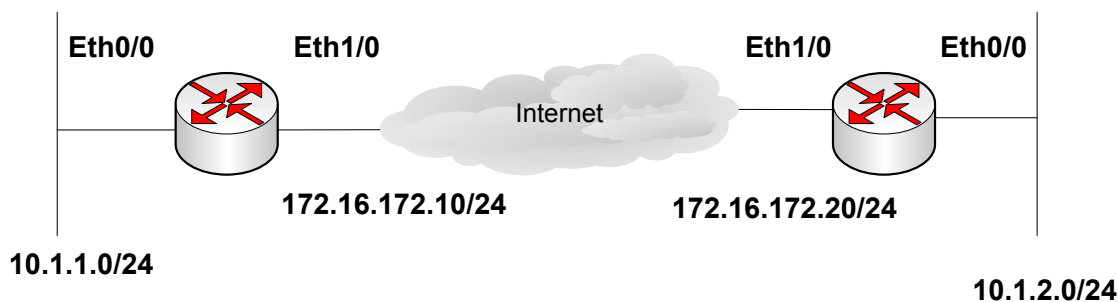
1.  $A \rightarrow B: c_a, c_b, \left\{ \begin{array}{l} h(SKEYID\_a, 1, N'_a, X'_a, \text{rest of msg}), \\ N'_a, X'_a, A, B, \text{rest of msg} \end{array} \right\}_{SKEYID\_e}$
2.  $B \rightarrow A: c_a, c_b, \left\{ \begin{array}{l} h(SKEYID\_a, 2, N'_a, N'_b, X'_b, \text{rest of msg}), \\ N'_b, X'_b, A, B, \text{rest of msg} \end{array} \right\}_{SKEYID\_e}$
3.  $A \rightarrow B: c_a, c_b, \left\{ h(SKEYID\_a, 3, N'_a, N'_b) \right\}_{SKEYID\_e}$



## A SIMPLE EXAMPLE

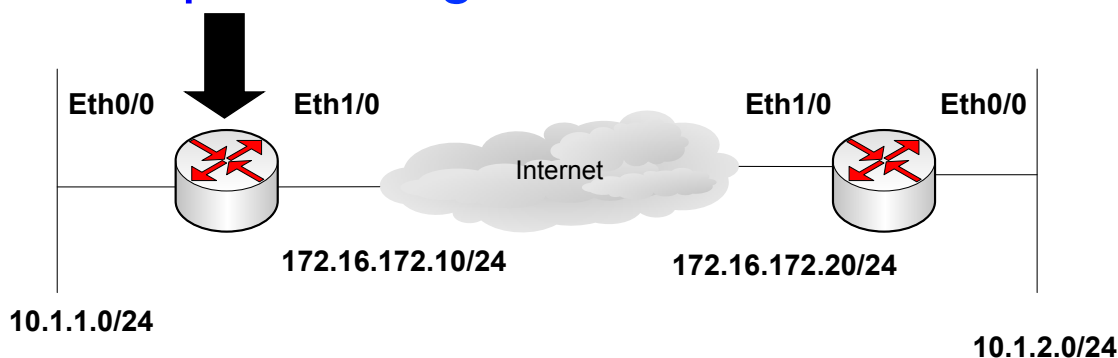


## Example: LAN-to-LAN VPN



- Authentication method: ***preshared keys***

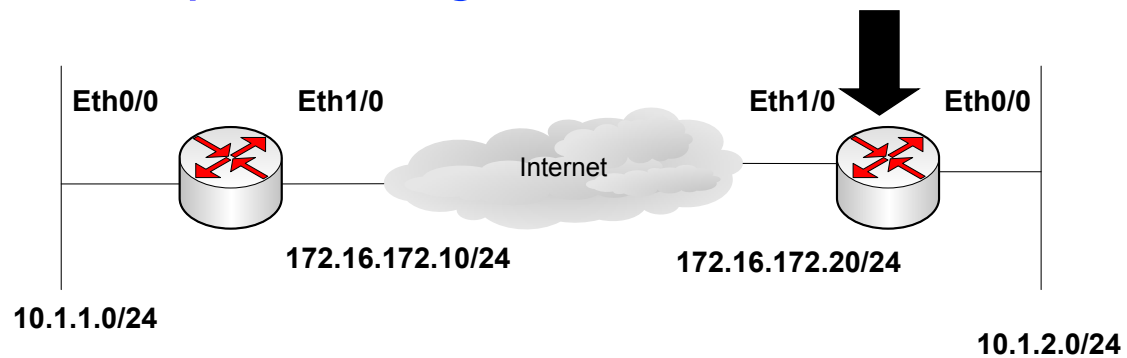
## Example: *configuration of router 1*



1. ISAKMP policy: (3des, sha, pre-shared key)
2. Security association: (tunnel mode, esp-3des esp-md5-hmac)
3. IpSec peer: **address = 172.16.172.20**
4. Pre-shared key: **key = jw4ep9846804ijl; address = 172.16.172.20**
5. Security policy: **permit ip 10.1.1.0/24 10.1.2.0/24**



## Example: *configuration of router 2*



1. ISAKMP policy: (3des, sha, pre-shared key)
2. Security Association: (tunnel mode, esp-3des esp-md5-hmac)
3. IpSec peer: address = 172.16.172.10
4. Pre-shared key: key = jw4ep9846804ijl; address = 172.16.172.10
5. Security policy: permit ip 10.1.2.0/24 10.1.1.0/24