Chair of Network Architectures and Services
Department of Informatics
Technical University of Munich

Network Security (NetSec)
IN2101 – WS 16/17

Prof. Dr.-Ing. Georg Carle

Dr. Heiko Niedermayer, Cornelius Diekmann

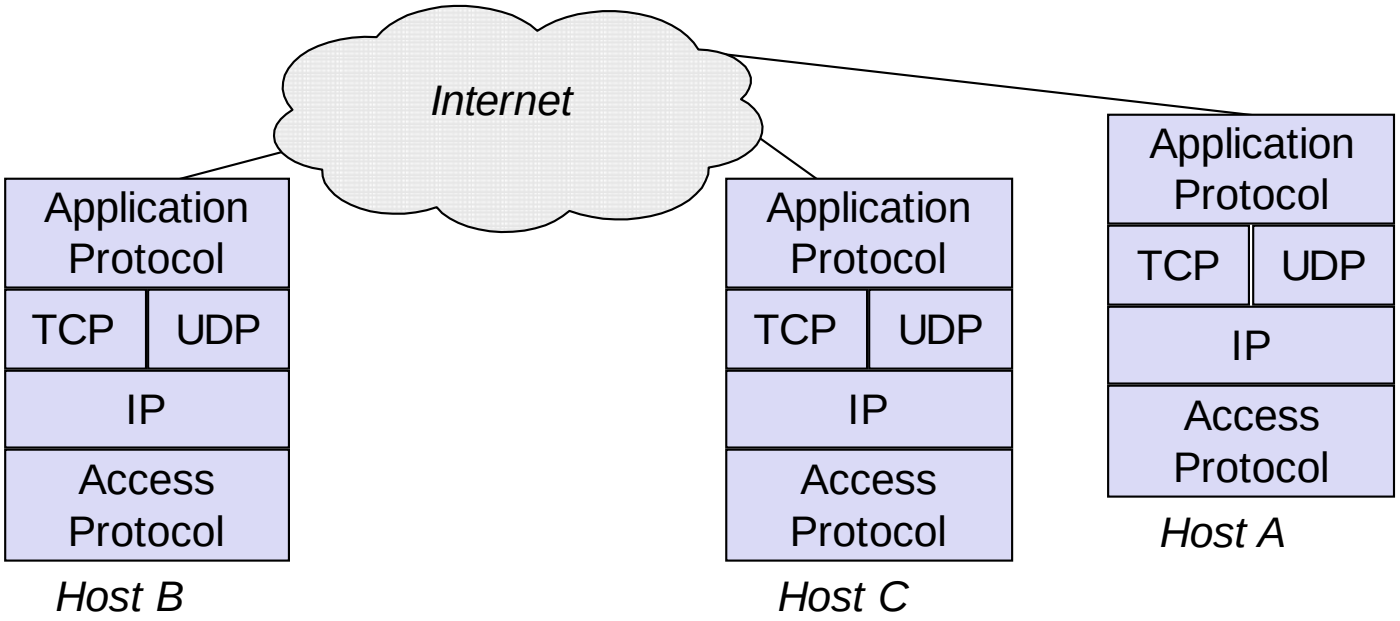
Version: 15.12. 2016

Technical University of Munich (TUM)
Department of Informatics
Chair of Network Architectures and Services

Chapter 12: The IPSec Security Architecture

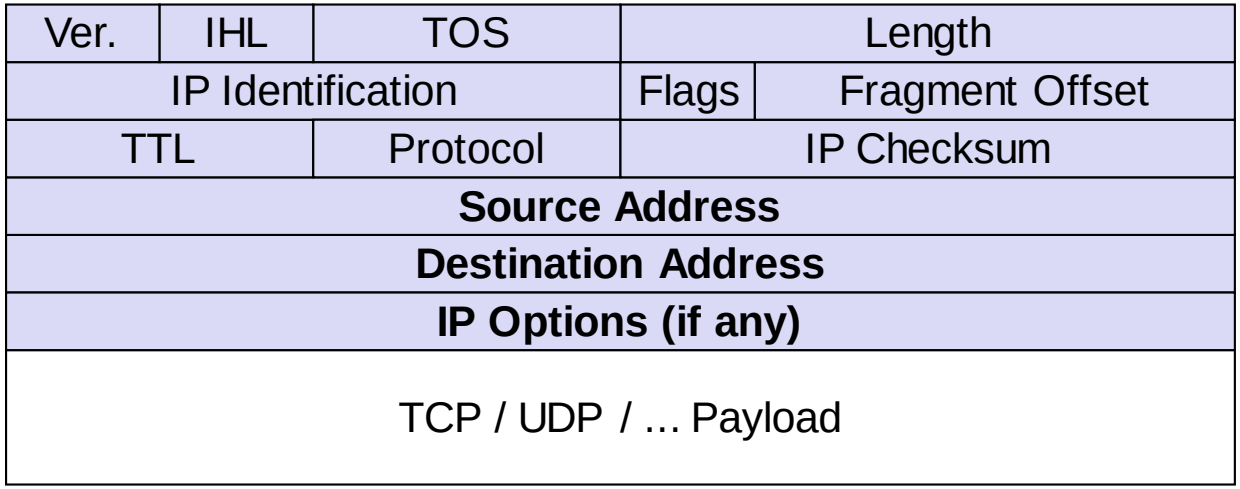
- Introduction
  - Brief introduction to the Internet Protocol (IP) suite
  - Security problems of IP and objectives of IPSec
- The IPSec architecture:
  - Overview
  - IP Replay Protection
  - IPSec security protocol modes:
    - Transport mode
    - Tunnel mode
  - IP Security Policies and the Security Policy Database (SPD)
  - Security associations (SA) and the SA Database (SAD)
  - Implementation alternatives
- IPSec security protocols:
  - Encapsulating Security Payload (ESP)
  - Authentication Header (AH)
- Entity Authentication and Key Establishment with the Internet Key Exchange (IKE)

The TCP/IP Protocol Suite



- *IP (Internet Protocol)*
  - unreliable, connectionless network protocol
- *TCP (Transmission Control Protocol)*
  - reliable, connection-oriented transport protocol
- *UDP (User Datagram Protocol)*
  - unreliable, connectionless transport protocol
- *application protocols*:
  - Examples: HTTP (Hypertext Transfer Protocol), SMTP (Simple Mail Transfer Protocol), …

The IPv4 Packet Format (1)



Version (Ver.): 4 bit

- Currently, version 4 is widely deployed
- Currently, version 6 is gradually taking over

Internet header length (IHL): 4 bit

- Length of the IP header in 32-bit words (i.e. no options ⇨ IHL=5)

Type of service (TOS): 8 bit

- Originally defined to indicate service requirements
- Redefined for DiffServ Code Points and Explicit Congestion Notification

The IP Packet Format (2)

Length: 16 bit

- The length of the packet including the header in octets
- This field is, like all other fields in the IP suite, in “big endian” representation (i.e., network byte order)

Identification: 16 bit

- Used to “uniquely” identify an IP datagram
- Important for reassembling of fragmented IP packets

Flags: 3 bit

- Bit 1: do not fragment
- Bit 2: datagram fragmented
- Bit 3: reserved for future use

Fragmentation offset: 13 bit

- The position of this packet in the corresponding IP datagram

Time to live (TTL): 8 bit

- At every processing network node, this field is decremented by one
- When TTL reaches 0, the packet is discarded to avoid packet looping

The IP Packet Format (3)

Protocol: 8 bit

- Indicates the (transport) protocol of the payload
- For example: TCP, UDP, ...

Checksum: 16 bit

- Protection of header against transmission errors
- Note well: it is not a cryptographic checksum

Source address: 32 bit

- The IP address of sender

Destination address: 32 bit

- The IP address of the intended receiver

IP Options: variable length

- IP headers can optionally carry additional information

Security Problems of the Internet Protocol

When an entity receives an IP packet, it has no assurance of:

- *Data origin authentication / data integrity:*
  - The packet has actually been sent by the entity which is referenced by the source address of the packet
  - The packet contains the original content the sender placed into it, so that it has not been modified during transport
  - The receiving entity is in fact the entity to which the sender wanted to send the packet
- *Confidentiality:*
  - The original data was not inspected by a third party while the packet was sent from the sender to the receiver

Security Objectives of IPSec

IPSec aims to ensure the following security objectives:

- *Data origin authentication / connectionless data integrity:*
  - It is not possible to send an IP datagram with neither a masqueraded IP source nor destination address without the receiver being able to detect this
  - It is not possible to modify an IP datagram in transit, without the receiver being able to detect the modification
  - *Replay protection:* it is not possible to later replay a recorded IP packet without the receiver being able to detect this
- *Confidentiality:*
  - It is not possible to eavesdrop on the content of IP datagrams
  - Limited traffic flow confidentiality

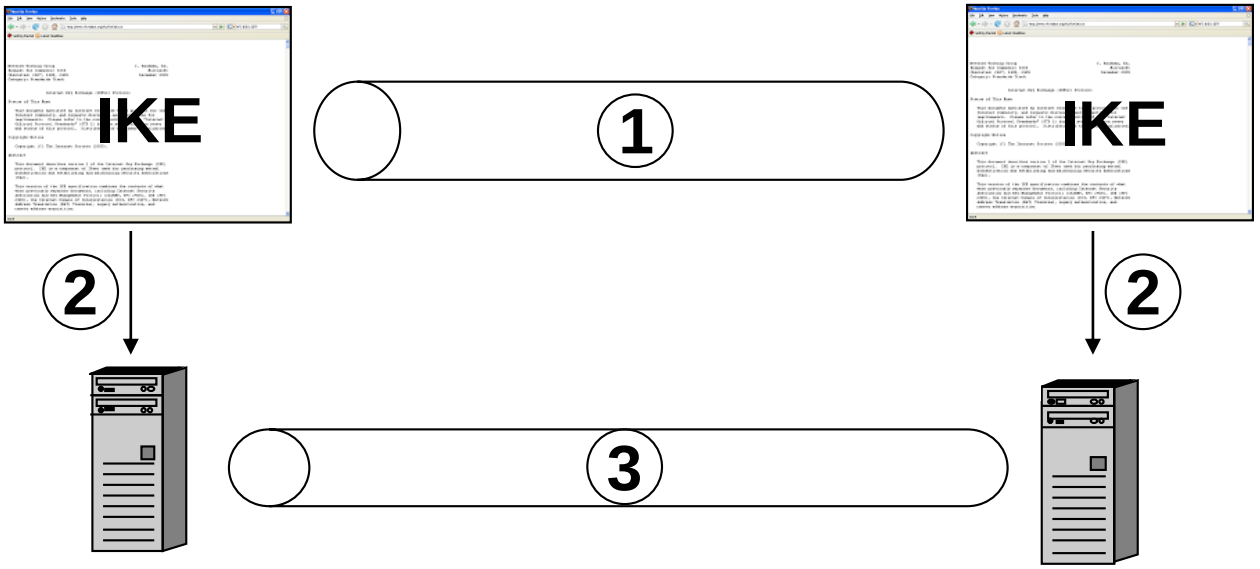
Security policy:

- Sender, receiver and intermediate nodes can determine the required protection for an IP packet according to a local security policy
- Intermediate nodes and the receiver will drop IP packets that do not meet these requirements

Overview

- Introduction
  - Brief introduction to the Internet Protocol (IP) suite
  - Security problems of IP and objectives of IPSec
- The IPSec architecture:
  - Overview
  - IP Replay Protection
  - IPSec security protocol modes:
    - Transport mode
    - Tunnel mode
  - IP Security Policies and the Security Policy Database (SPD)
  - Security associations (SA) and the SA Database (SAD)
  - Implementation alternatives
- IPSec security protocols:
  - Encapsulating Security Payload (ESP)
  - Authentication Header (AH)
- Entity Authentication and Key Establishment with the Internet Key Exchange (IKE)

Overview of the IPSec Architecture (1)



(1) Authentication, key establishment and negotiation of cryptographic algorithms

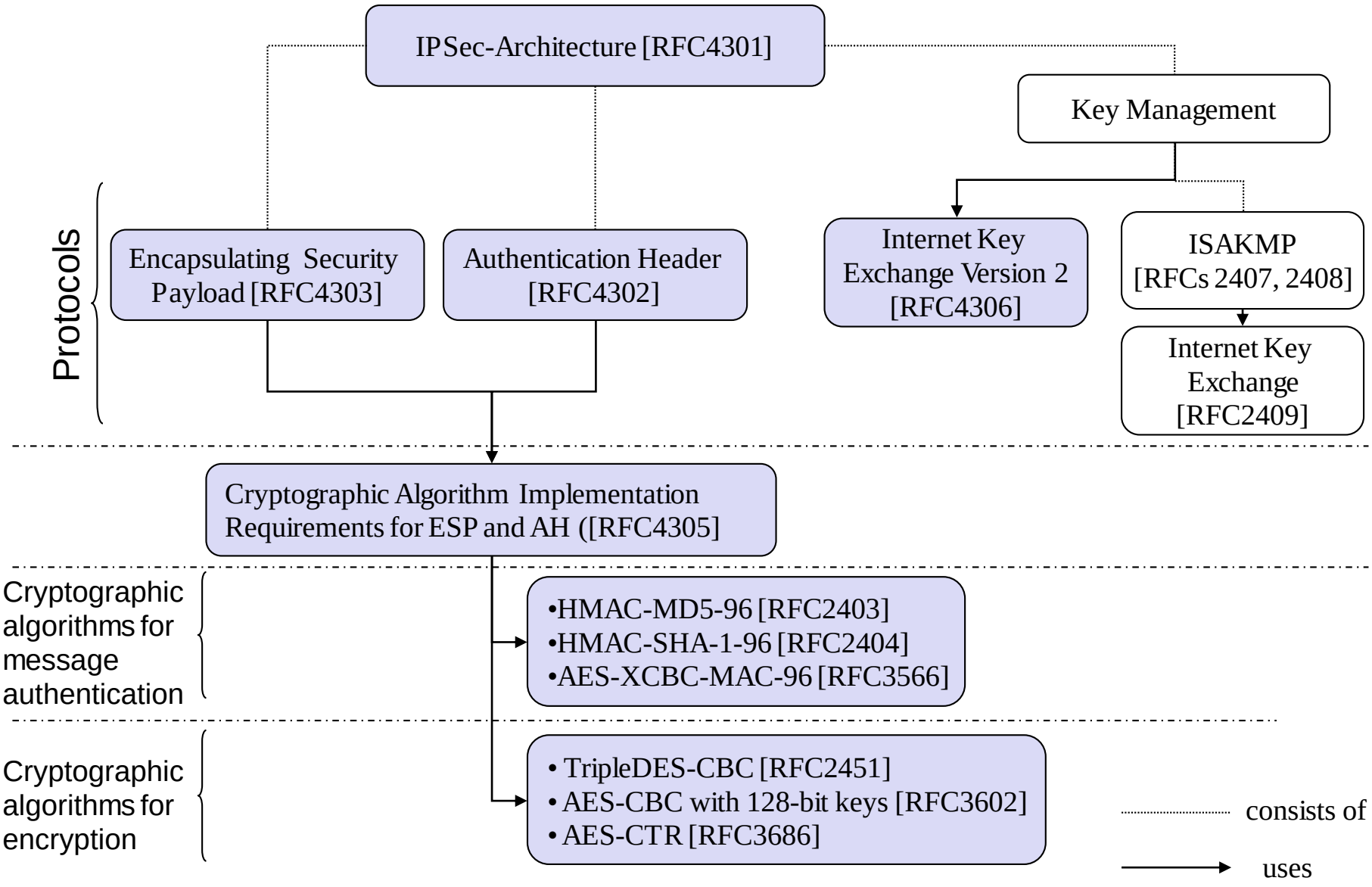
- ❑ Protocols: ISAKMP, Internet Key Exchange (IKE), IKEv2

(2) Set keys and cryptographic algorithms

(3) Secure channel, which provides

- Data integrity: using the Authentication Header (AH) protocol or the Encapsulating Security Payload (ESP)
- Confidentiality using ESP
- Note: ESP can provide both data integrity and encryption while AH provides only data integrity

Overview of the IPSec Standardization Documents



Overview of the IPSec Architecture (2)

RFC 4301 defines the basic architecture of IPSec:

- Concepts:
  - Security association (SA), security association database (SAD)
  - Security policy, security policy database (SPD)
- Fundamental IPSec Protocols:
  - Authentication Header (AH)
  - Encapsulating Security Payload (ESP)
- Protocol Modes:
  - Transport Mode
  - Tunnel Mode
- Key management protocols:
  - ISAKMP, IKE, IKEv2

A list of most RFCs related to IPSec can be found here

- http://www.ietf.org/html.charters/OLD/ipsec-charter.html

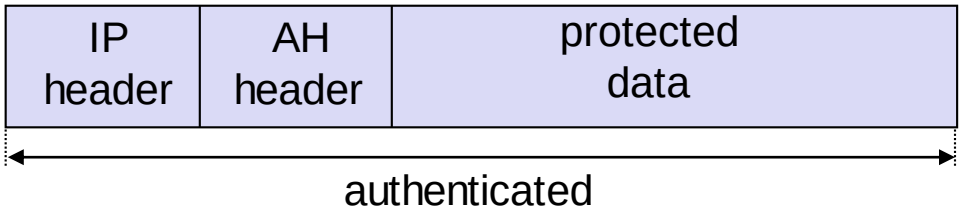
Most of the RFCs have been updated in 2005 (after several years of revision)

- Support of integration of new cryptographic primitives for encryption and data integrity easily
- Reduced complexity by better protocol design and by omitting some useless features

Overview of the IPSec Architecture (3)

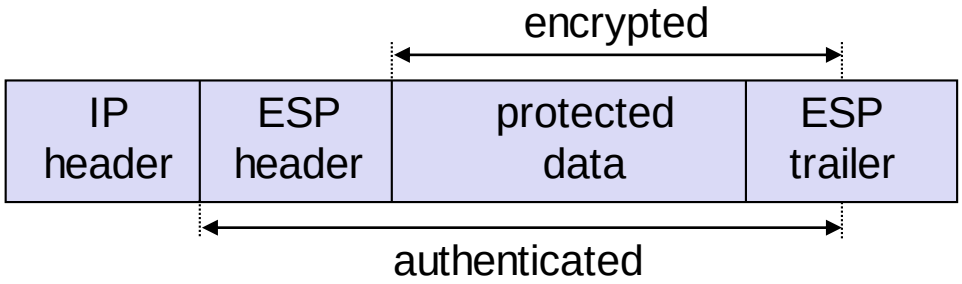
Authentication Header (AH):

- Data origin authentication and replay protection
- Inserted between the IP header and the data to be protected



Encapsulating Security Payload (ESP):

- Data origin authentication, confidentiality, and replay protection
- A header and a trailer encapsulating the data to be protected



Overview of the IPSec Architecture (4)

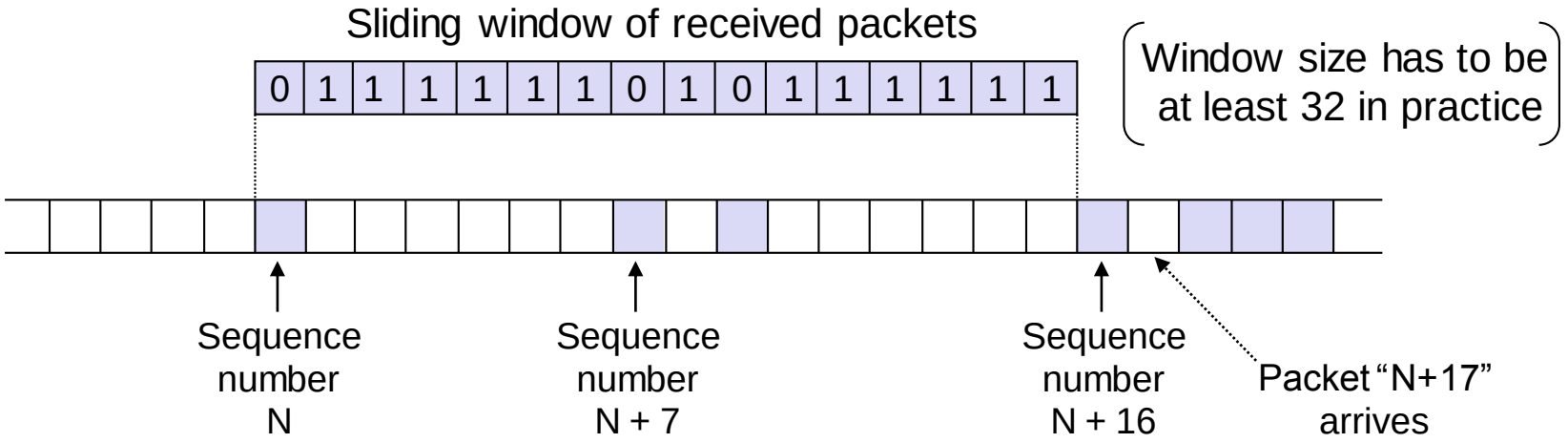
Key management and setup of security associations

- Internet Security Association Key Management Protocol (ISAKMP):
  - Defines generic framework for key authentication, key exchange and negotiation of security association parameters [RFC2408]
  - Does not define a specific authentication protocol, but specifies:
    - Packet formats
    - Retransmission timers
    - Message construction requirements
  - Use of ISAKMP for IPSec is further detailed in [RFC2407]
- Internet Key Exchange Version 2 [RFC4306]
  - Defines an authentication and key exchange protocol
  - Reduced complexity by better protocol design and by omitting some useless features

IPSec Replay Protection (1)

AH- and ESP-protected IP packets carry a sequence number for replay protection

- When setting up a security association (SA) this sequence number is initialized to zero
- The sequence number is increased with every IP packet sent
- The sequence number is 32 bit long, and a new session key is needed before a wrap-around occurs
- The receiver of an IP packet checks if the sequence number is contained in a window of acceptable numbers



Packet with sequence number N can still be accepted

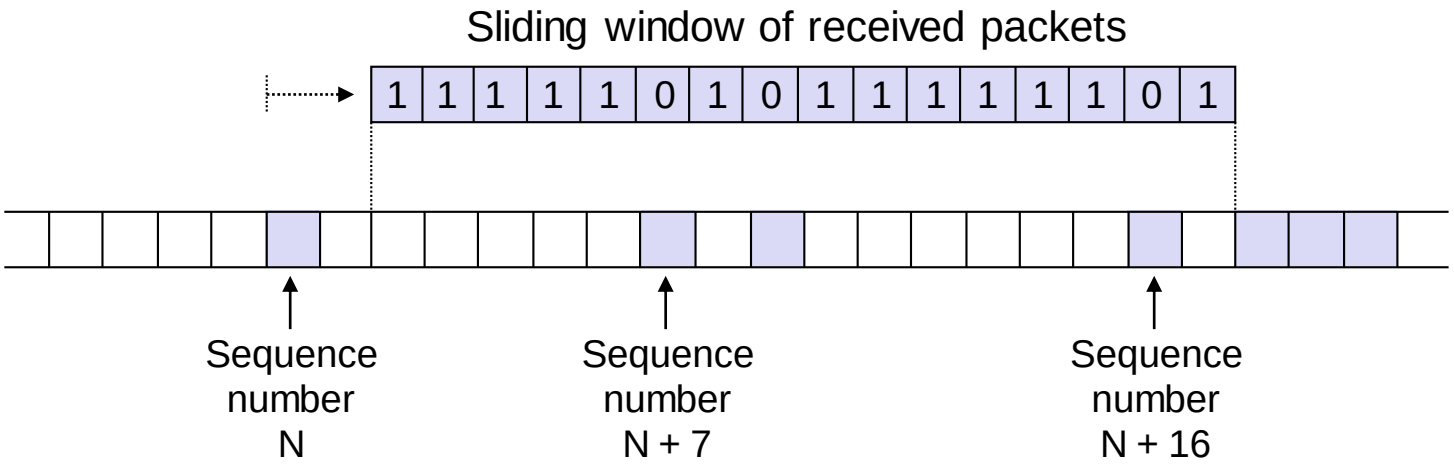
IPSec Replay Protection (2)

If a received packet has a sequence number which:

- is left of the current window ⇒ the receiver rejects the packet
- is inside the current window ⇒ the receiver accepts the packet
- is right of the current window ⇒ the receiver accepts the packet and advances the window

Of course IP packets are only accepted if they pass the authentication verification and the window is never advanced before this verification

The minimum window size is 32 packets (64 packets is recommended)



Packet with sequence number N can no longer be accepted

IPSec Modes

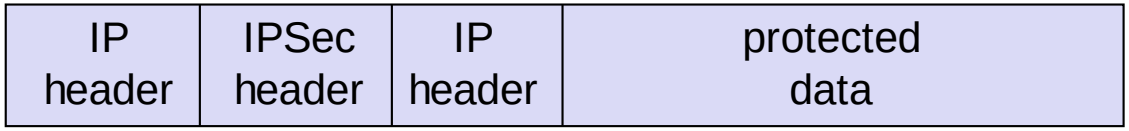
IPSec works in two modes:

- *Transport mode* can only be used between end-points of a communication:
  - host ↔ host, or
  - host ↔ gateway, if the gateway is a communication end-point (e.g. for network management)
- *Tunnel mode* can be used with arbitrary peers

The difference between the two modes is, that:

- Transport mode adds a security specific header (+ trailer for ESP):

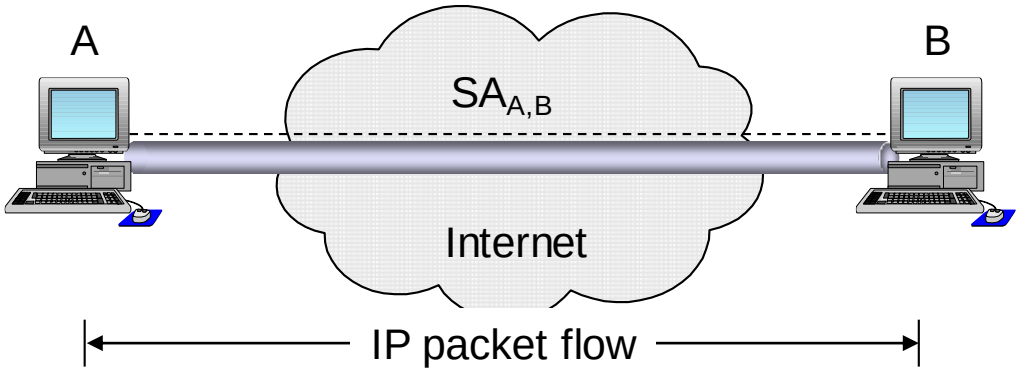
- Tunnel mode encapsulates IP packets:



Encapsulation of IP packets allows for a gateway protecting traffic on behalf of other entities (e.g. hosts of a subnetwork, etc.)

IPSec Transport Mode

- Transport mode is used when the “cryptographic endpoints” are also the “communication endpoints” of the secured IP packets
  - Cryptographic endpoints: the entities that generate / process an IPSec header (AH or ESP)
  - Communication endpoints: source and destination of an IP packet

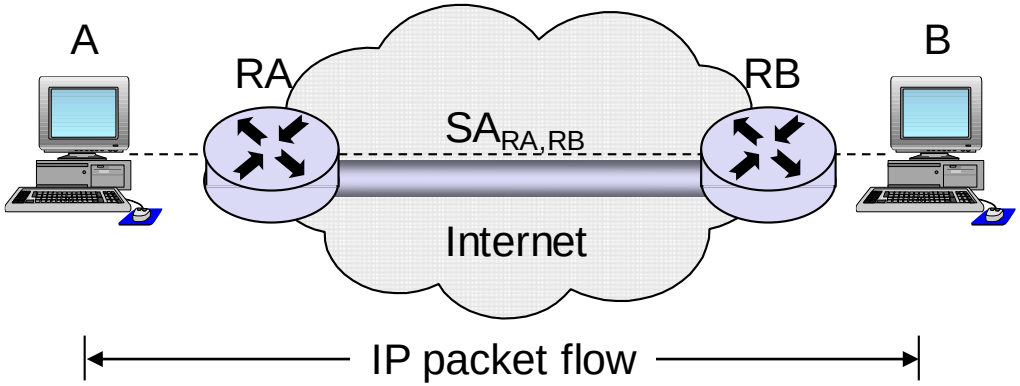


In most cases, communication endpoints are hosts (workstations, servers), but this is not necessarily the case:

- Example: a gateway being managed via SNMP by a workstation

IPSec Tunnel Mode (1)

- Tunnel mode is used when at least one “cryptographic endpoint” is not a “communication endpoint” of the secured IP packets
  - This allows for gateways securing IP traffic on behalf of other entities

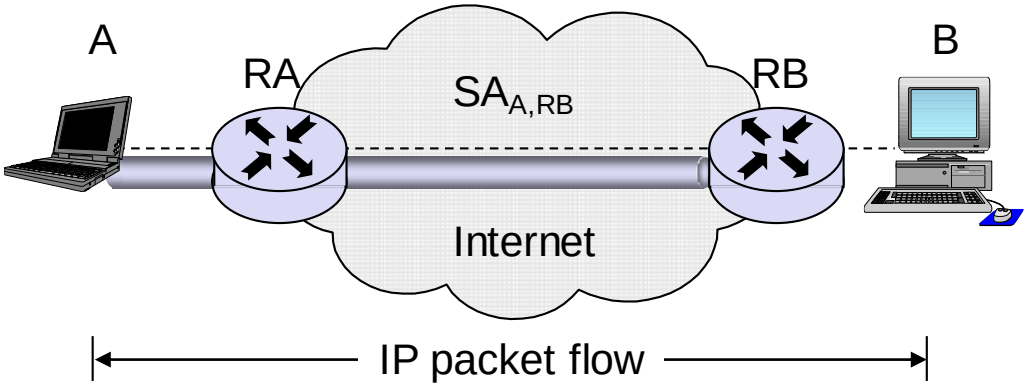


packet structure

IP header	IPSec header	IP header	protected data
Src = RA Dst = RB		Src = A Dst = B	

IPSec Tunnel Mode (2)

- The above description of application scenarios for tunnel mode includes the case in which only one cryptographic endpoint is not a communication endpoint:
  - Example: a security gateway ensuring authentication and / or confidentiality of IP traffic between a local subnetwork and a host connected via the Internet (“road warrior scenario”)



packet structure

IP header	IPSec header	IP header	protected data
Src = A Dst = RB		Src = A Dst = B	

Traffic Selectors

A Traffic Selector (TS) is a set of properties used to characterize IP packets

Each TS may contain the following information

- *IP source address:*
  - Specific host, network prefix, address range, or wildcard
- *IP destination address:*
  - Specific host, network prefix, address range, or wildcard
  - In case of incoming tunneled packets the inner header is evaluated
- *Name:*
  - DNS name, X.500 name or other name types
- *Protocol:*
  - The protocol identifier of the transport protocol for this packet (e.g. TCP or UDP)
  - This may not be accessible when a packet is secured with ESP

IPSec Security Policy Definition

Traffic Selectors are used to define security policies in the SPD

A policy definition specifies:

- Which and how security services should be provided to IP packets:
  - *Selectors* that identify specific IP flows
  - Required *Security attributes* for each flow:
    - *Security protocol:* AH or ESP
    - *Protocol mode:* transport or tunnel mode
    - *Other parameters:* e.g. policy lifetime
  - *Action:* discard, secure, bypass

The security policies are stored in the security policy database (SPD)

Note

- since it is possible to include port numbers (i.e., layer 4 addresses) in the traffic selector of a security policy, IPSec protection can be specified for specific applications running on a host

IPSec Security Associations (1)

A *security association (SA)* is a simplex "connection" that describes the way how outgoing/incoming packets need to be processed, such as encryption/authentication algorithms and encryption/authentication keys

An SA is associated with either AH or ESP, but not both

For bi-directional communication two security associations are needed

An SA can be set up between the following peers:

- Host ↔ Host
- Host ↔ Gateway (or vice versa)
- Gateway ↔ Gateway

Security associations are stored in the security association database (SAD)

IPSec Security Associations (2)

An SA is uniquely identified by a *security parameter index (SPI)*

The SPI value is specified by the receiving side during SA negotiation.

An SAD entry for an outbound SA specifies the SPI used when constructing the AH or ESP header.

For an SAD entry of an inbound SA, the SPI is used to map traffic to the appropriate SA.

IPSec Security Associations (3)

An SA entry in the SAD includes the following information

- Security Parameter Index (SPI)
- IP source address
- IP destination address
- A security protocol identifier (AH / ESP)
- Current sequence number counter (for replay protection)
- AH authentication algorithm and key (if it is an AH SA)
- ESP encryption and integrity algorithms, keys, mode, IV (if it is an ESP SA)
- SA lifetime
- IPSec protocol mode: tunnel mode or transport mode
- Additional information
  - (See RFC 4301, Section 4.4.2.1. “Data Items in the SAD”)

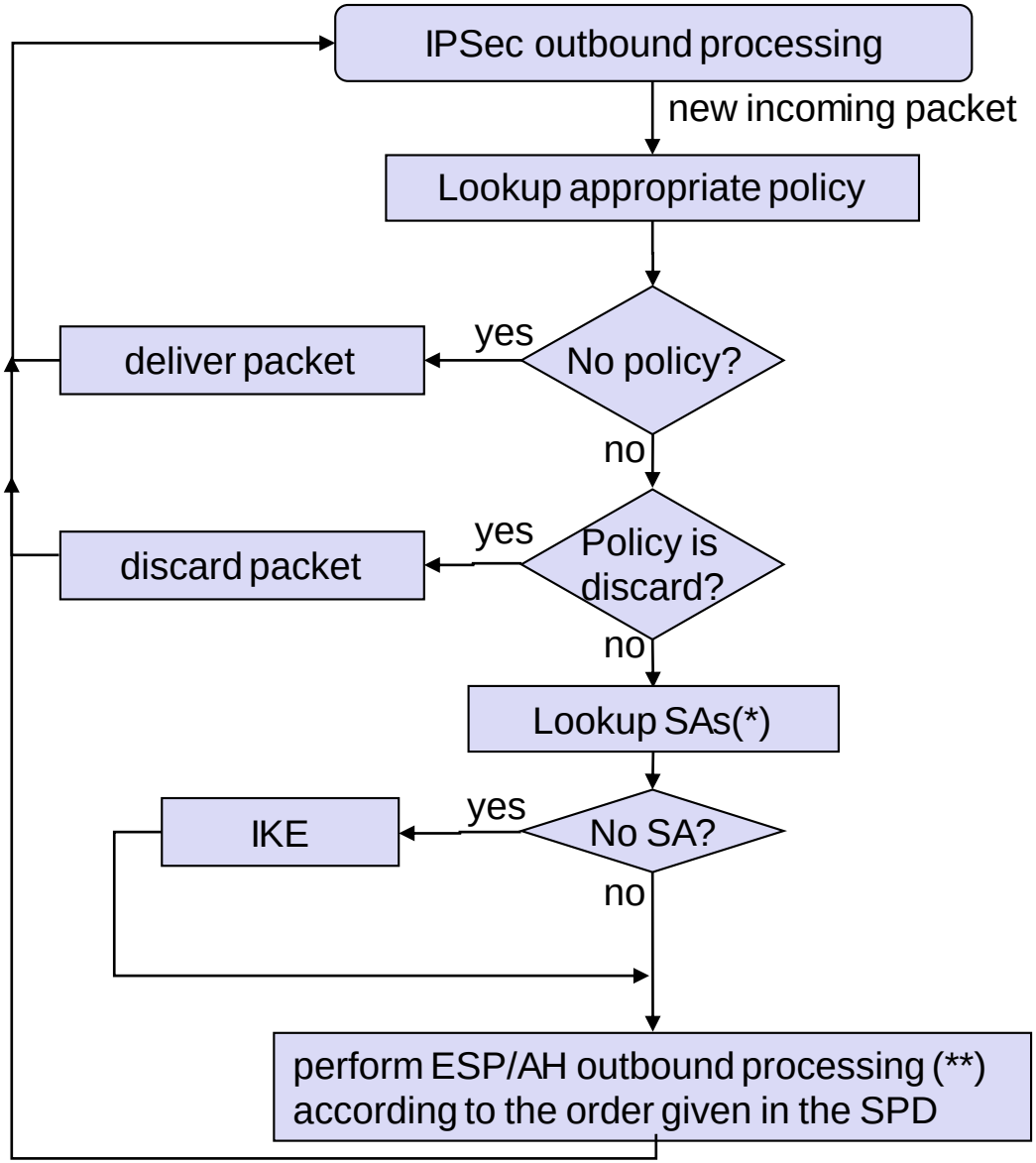
Basic Scheme of IPSec Processing: Outgoing Packets (1)

Consider the IP layer of a node (host / gateway) is told to send an IP packet to another node (host / gateway)

In order to support IPSec it has to perform the following steps:

- Determine if and how the outgoing packet needs to be secured:
  - This is realized by performing a lookup in the SPD based on the traffic selectors
  - If the policy specifies “discard” then drop the packet ⇒ done
  - If the packet does not need to be secured, then send it ⇒ done
- Determine which SA should be applied to the packet (based on the IP addresses, possibly ports)
  - If there is not yet an appropriate SA established with the corresponding node, then ask the key management process to perform IKE
- Look up the determined (or freshly created) SA in the SAD
- Perform the “security transforms” determined by the SA by using the algorithm, its parameters and the key as specified in the SA
  - This results in the construction of an AH or an ESP header
  - Possibly a new (outer) IP header will be created (tunnel mode)
- Send the resulting IP packet ⇒ done

Basic Scheme of IPSec Processing: Outgoing Packets (2)



(*) There may be one or more SAs matching the IP packet, e.g. one SA for AH and one SA for ESP

(**) AH/ESP outbound processing is illustrated later

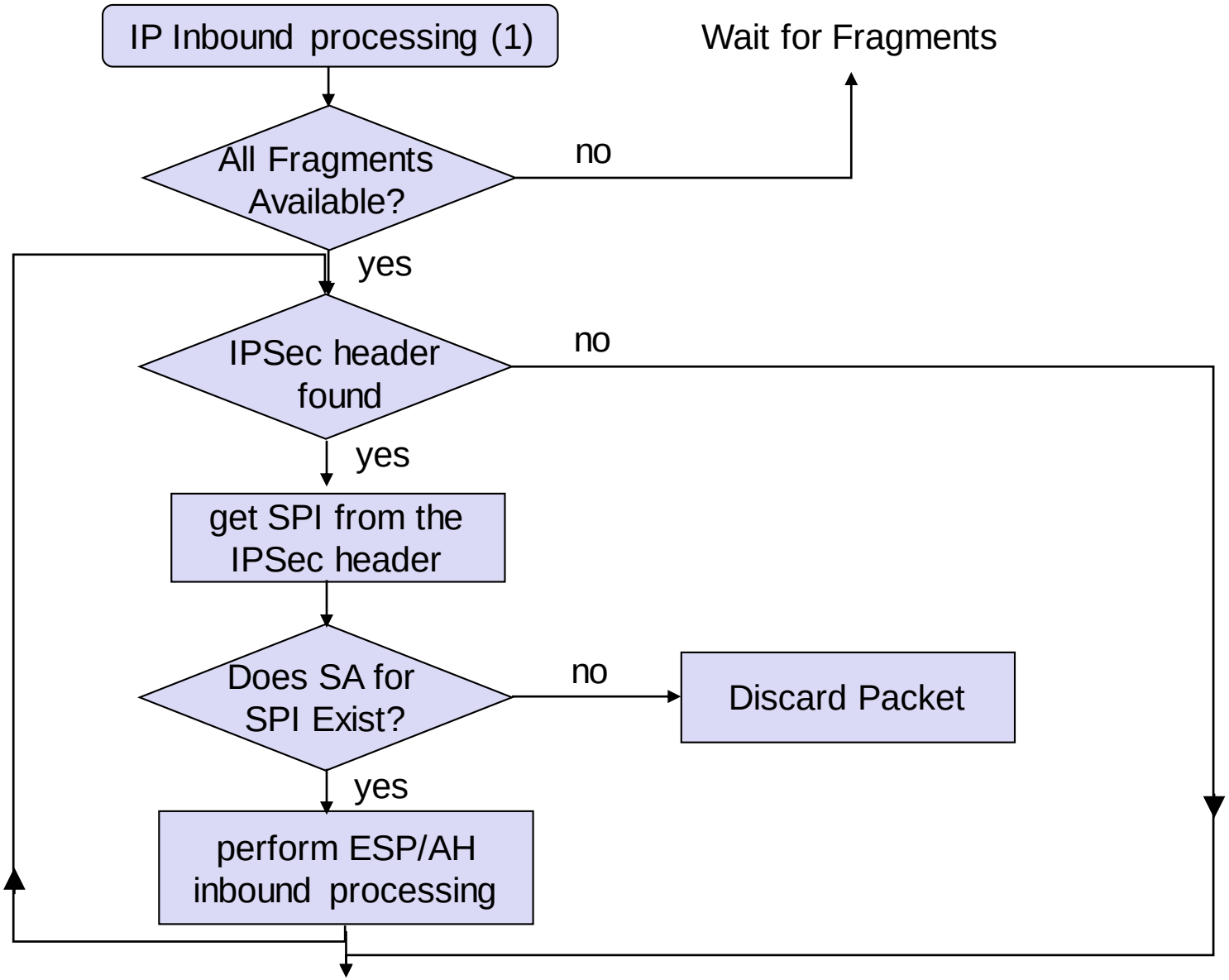
Basic Scheme of IPSec Processing: Incoming Packets

Consider the IP layer of a node (host / gateway) receives an IP packet from another node (host / gateway)

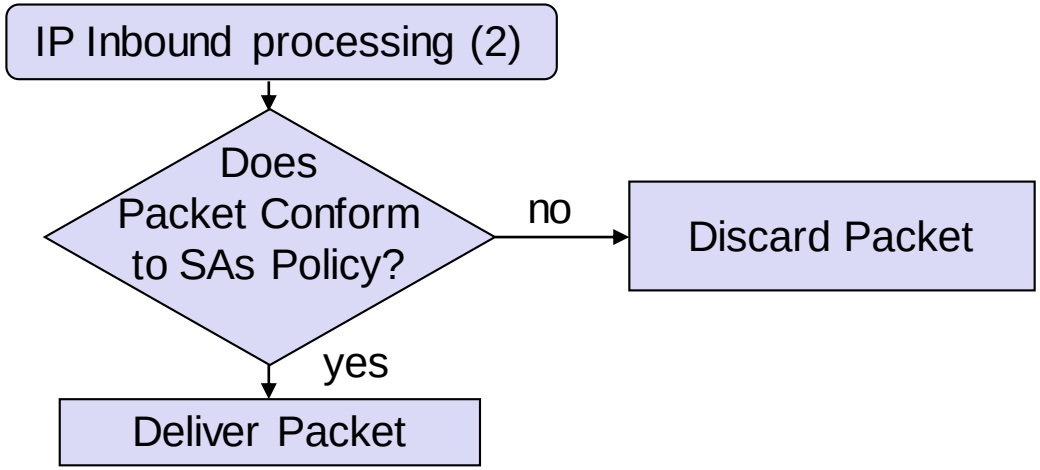
In order to support IPSec it has to perform the following steps:

- For packets containing an IPSec header this entity is supposed to process:
  - Extract the SPI from the IPSec header, look up the SA in the SAD and perform the appropriate AH/ESP processing
  - If there is no SA referenced by the SPI, drop the packet
- After AH and/or ESP processing, determine if and how the packet should have been protected:
  - This is again realized by performing a lookup in the SAD, with the lookup being performed by evaluating the inner IP header in case of tunneled packets
  - If the policy specifies “discard” then drop the packet
  - If the protection of the packet did not match the policy, drop the packet
    - This can be the case, e.g. if the policy enforces both AH and ESP protection, while the packet includes a AH header
  - If the packet had been properly secured, then deliver it to the appropriate protocol entity (network / transport layer)

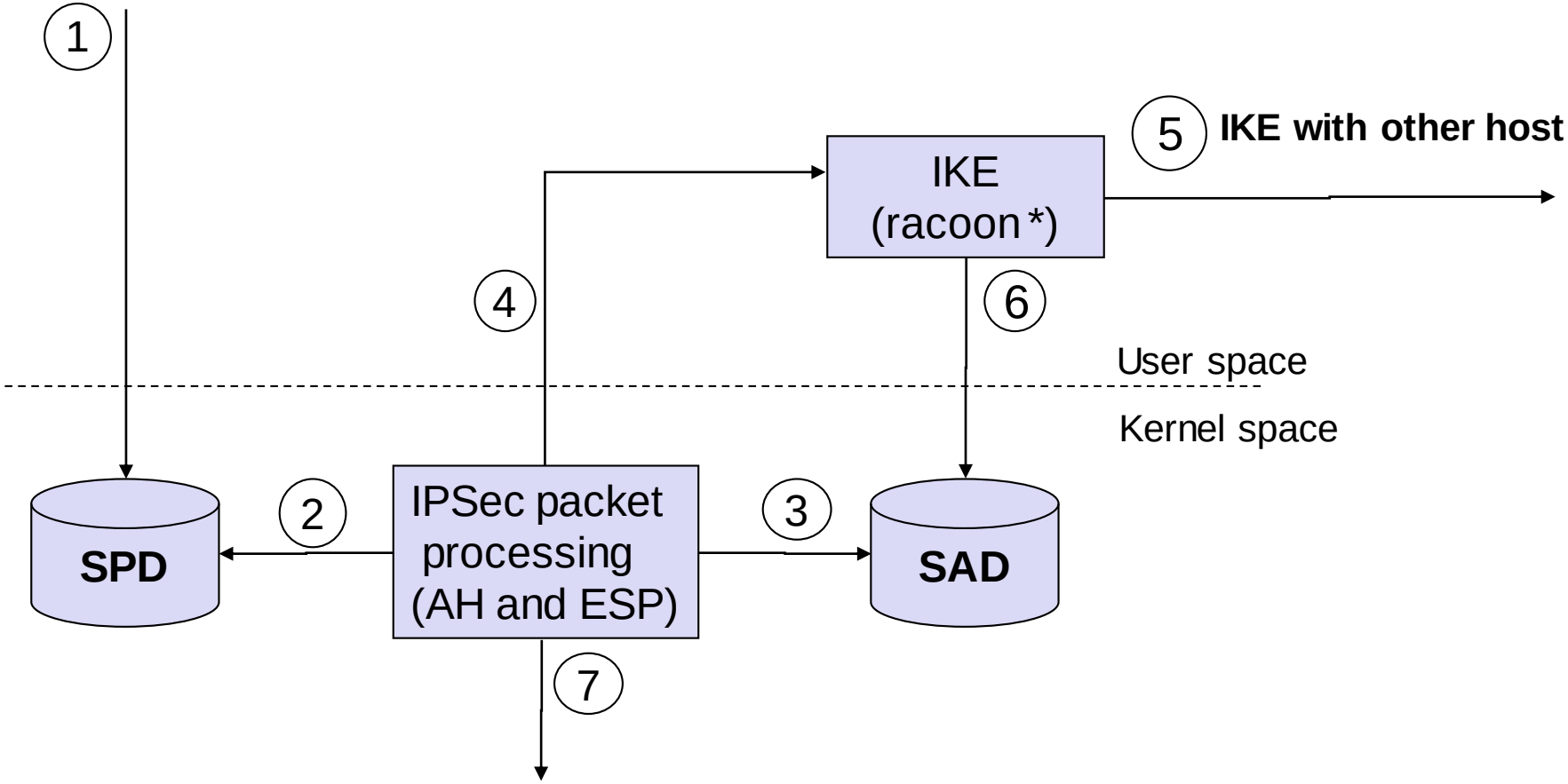
Basic Scheme of IPSec Processing: Incoming Packets (2)



Basic Scheme of IPSec Processing: Incoming Packets (3)



Example: IPSec Processing (1)



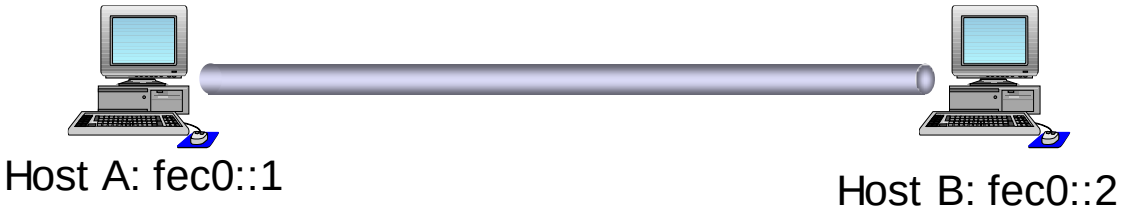
(*) *racoon* is a widespread IKE implementation in Linux

Example: IPSec Processing (2)

1) The administrator sets a policy in SPD
2) The IPSec processing module in the kernel refers to SPD in order to make a decision on applying IPsec to a packet.
3) If IPsec is required, then the IPSec processing module looks for the IPsec SA in SAD.
4) If there is no SA yet, then the IPSec processing module sends a request to IKE process (*racoon)* to get an SA
5) IKE process (*racoon)* performs a key exchange and negotiation of the cryptographic algorithms with the peer host using the IKE/IKEv2 protocol
6) IKE process (*racoon)* put the Key and all the required parameters into SAD
7) The IPSec processing module can send a packet applied IPsec

Example of IPSec Security Policies (1)

Example: IPv6. Security protocol is ***ESP***. Encapsulation mode is ***Transport***.



Configuration at Host A:

- spdadd **fec0::1 fec0::2** any -P **out** ipsec esp/transport//require ;
- spdadd **fec0::2 fec0::1** any -P **in** ipsec esp/transport//require ;

Note

- First IP address means source in the IP header
- Next IP address means destination in the IP header
- **out** means this policy holds for outgoing packet
- **in** means this policy holds for an incoming packet

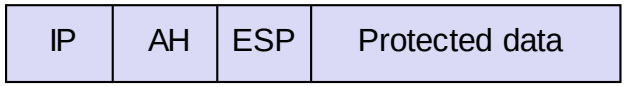
Configuration at Host B:

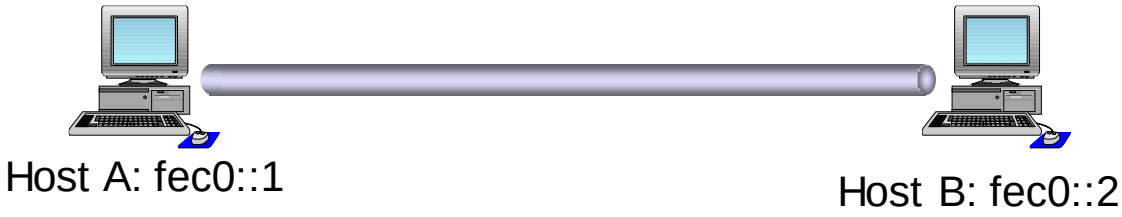
- spdadd **fec0::2 fec0::1** any -P **out** ipsec esp/transport//require ;
- spdadd **fec0::1 fec0::2** any -P **in** ipsec esp/transport//require ;

Example of IPSec Security Policies (2)

Example: ***ESP Transport*** mode applied first and ***AH Transport*** mode next

It means that the resulting packets looks as follow:





Configuration at Host A:

- spdadd fec0::1 fec0::2 any -P **out** ipsec
  esp/transport//require **ah**/transport//require ;
- spdadd fec0::2 fec0::1 any -P **in** ipsec
  esp/transport//require **ah**/transport//require ;

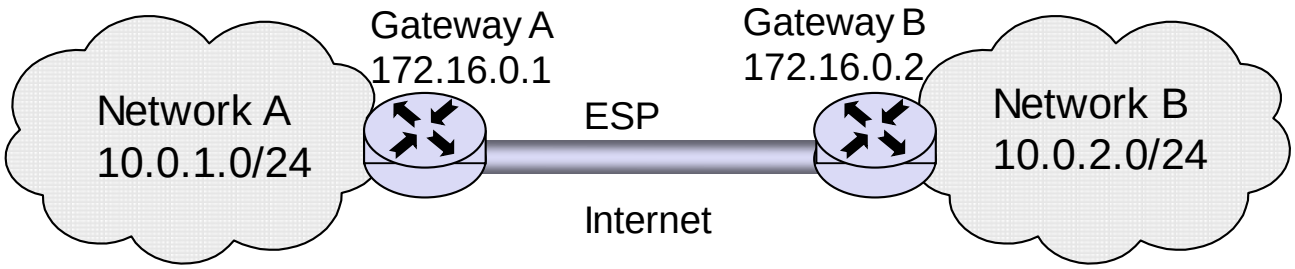
Note the ordering of the security protocol

Configuration at Host B:

- spdadd fec0::2 fec0::1 any -P out ipsec
  esp/transport//require ah/transport//require ;
- spdadd fec0::1 fec0::2 any -P in ipsec
  esp/transport//require ah/transport//require ;

Example of IPSec Security Policies (3)

ESP Tunnel for VPN



Configuration at Gateway A:

- spdadd 10.0.1.0/24 10.0.2.0/24 any -P out ipsec
  esp/tunnel/**172.16.0.1-172.16.0.2**/require ;
- spdadd 10.0.2.0/24 10.0.1.0/24 any -P in ipsec
  esp/tunnel/**172.16.0.2-172.16.0.1**/require ;

Configuration at Gateway B:

- spdadd 10.0.2.0/24 10.0.1.0/24 any -P out ipsec
  esp/tunnel/**172.16.0.2-172.16.0.1**/require ;
- spdadd 10.0.1.0/24 10.0.2.0/24 any -P in ipsec
  esp/tunnel/**172.16.0.1-172.16.0.2**/require ;

Example: Setup of Security Associations

Note: it is also possible to set a SA manually

e.g. for manually setting up an AH SA

```
# add src dst proto spi -A authalgo key;
add 3.0.0.1 5.0.0.1 ah 700 -A hmac-md5 0xbf9a081e7ebdd4fa824c822ed94f5226;
add 5.0.0.1 3.0.0.1 ah 800 -A hmac-md5 0xbf9a081e7ebdd4fa824c822ed94f5226;
```

e.g. for manually setting up an ESP SA

```
# add src dst proto spi -E encalgo key;
add 3.0.0.1 5.0.0.1 esp 701 -E 3des-cbc
0x3f0b868ad03e68acc6e4e4644ac8bb80ecea3426d3d30ada;
add 5.0.0.1 3.0.0.1 esp 801 -E 3des-cbc
0x3f0b868ad03e68acc6e4e4644ac8bb80ecea3426d3d30ada;
```

Note however that setting up SA manually is error-prone

- The administrator might choose insecure keys
- The set of SAs might be inconsistent

It is better to rely on an IKE daemon for setting up SAs

Overview

- Introduction
  - Brief introduction to the Internet Protocol (IP) suite
  - Security problems of IP and objectives of IPSec
- The IPSec architecture:
  - Overview
  - IP Replay Protection
  - IPSec security protocol modes:
    - Transport mode
    - Tunnel mode
  - IP Security Policies and the Security Policy Database (SPD)
  - Security associations (SA) and the SA Database (SAD)
  - Implementation alternatives
- IPSec security protocols:
  - Encapsulating Security Payload (ESP)
  - Authentication Header (AH)
- Entity Authentication and Key Establishment with the Internet Key Exchange (IKE)

The Encapsulating Security Payload (1)

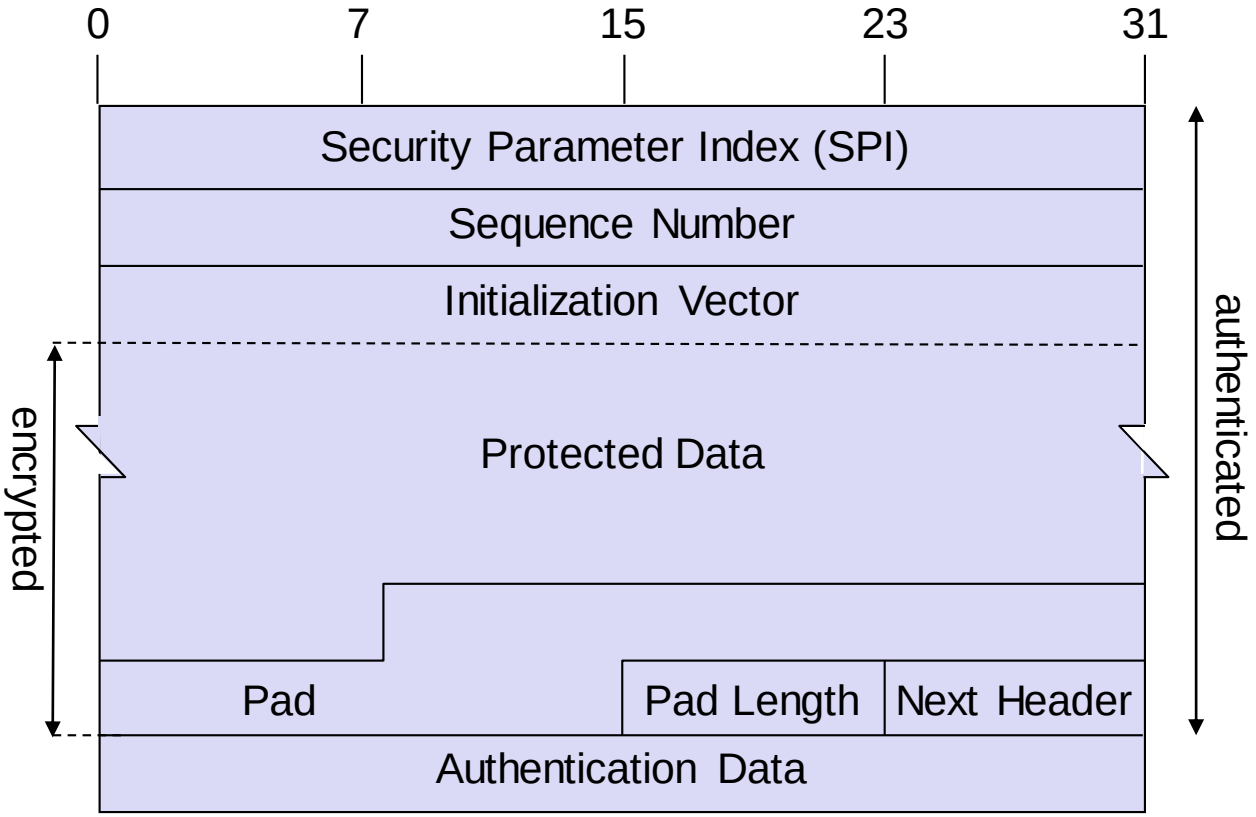
ESP is a generic security protocol that provides to IP packets replay protection and one or both of the following security services:

- Confidentiality, by encrypting encapsulated packets or just their payload
- Data origin authentication, by creating and adding MACs to packets
- Replay protection

The ESP definition is divided into two parts:

- The definition of the base protocol [RFC4303]:
  - Definition of the header and trailer format
  - Basic protocol processing
  - Tunnel and transport mode operation

The Encapsulating Security Payload (2)



The ESP header immediately follows an IP header or an AH header

The next-header field of the preceding header indicates “50” for ESP

The Encapsulating Security Payload (3)

The SPI (Security Parameter Index) field indicates the SA to be used for this packet:

- The SPI value is always determined by the receiving side during SA negotiation as the receiver has to process the packet

The sequence number provides replay protection as explained before

If the cryptographic algorithm in use requires an initialization vector, it is transmitted in the clear in every packet at the beginning of the payload

The pad field serves to ensure:

- padding of the payload up to the required block length of the cipher in use
- padding of the payload to right-justify the pad-length and next-header fields into the high-order 16 bit of a 32-bit word

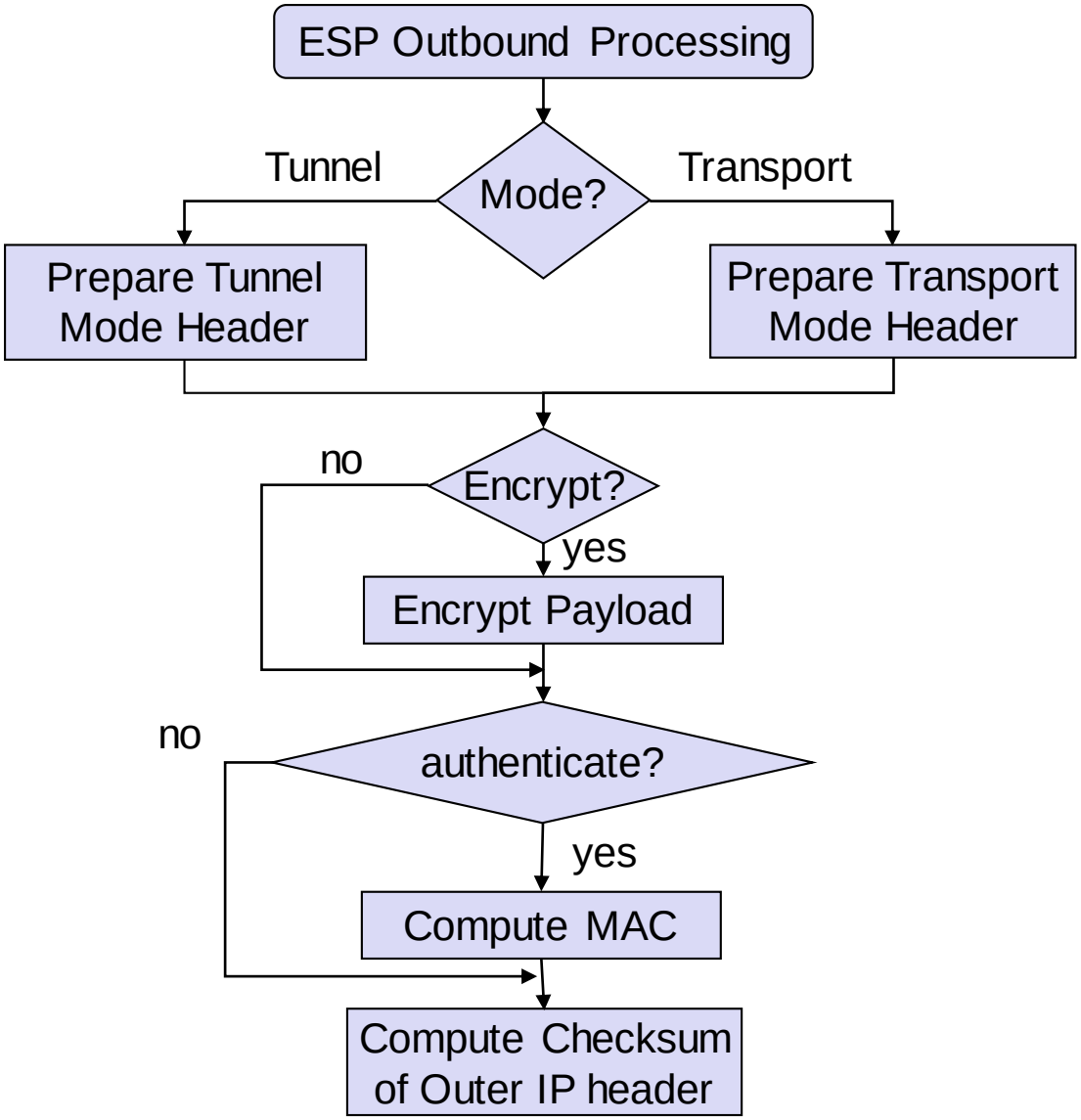
The pad length indicates the amount of padding bytes added

The next-header field of the ESP header indicates the encapsulated payload:

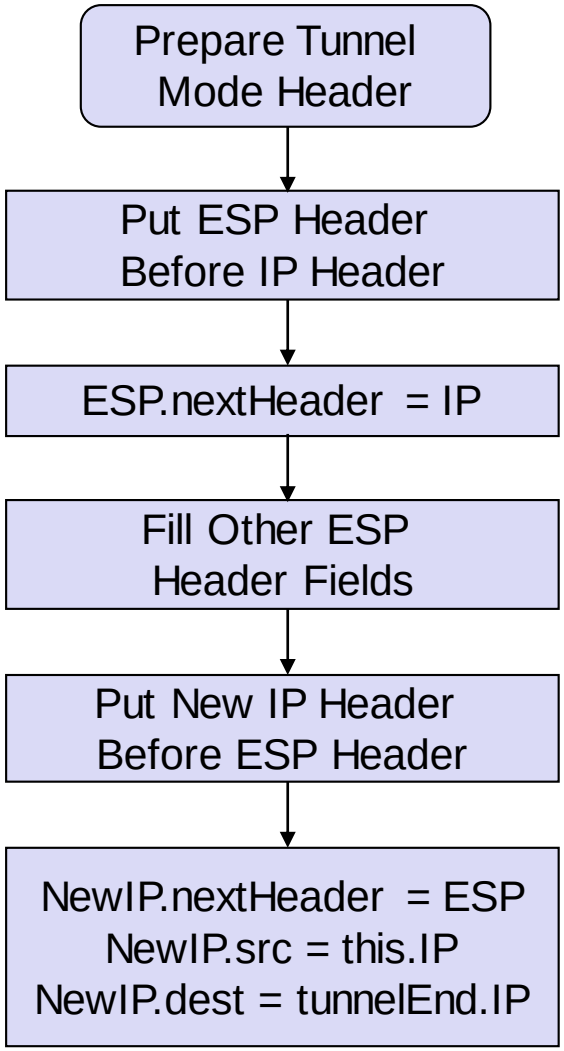
- In case of tunnel mode: IP
- In case of transport mode: any higher-layer protocol as TCP, UDP, ...

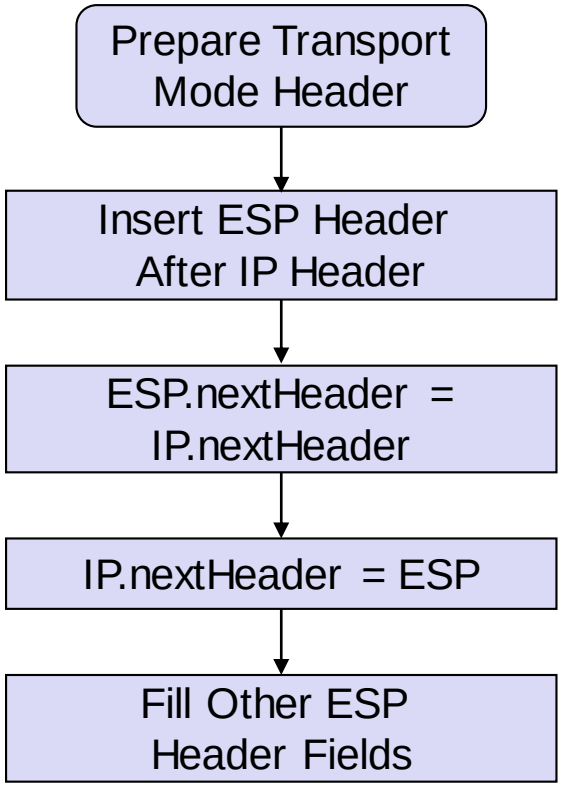
The optional authentication data field contains a MAC, if present

The Encapsulating Security Payload (4)

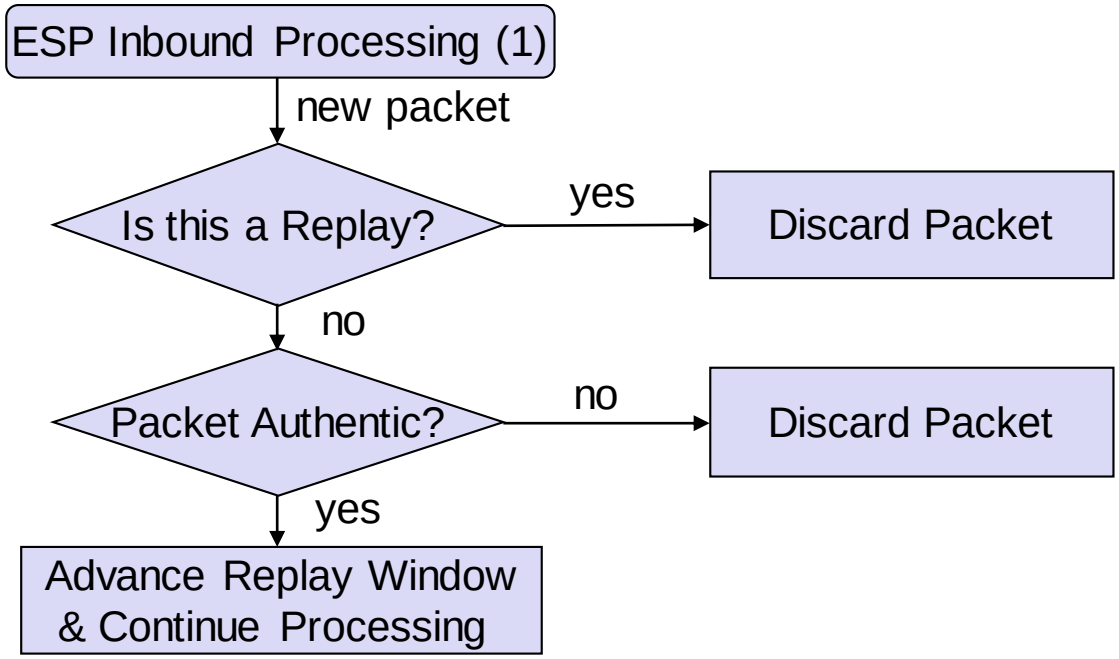


The Encapsulating Security Payload (5)

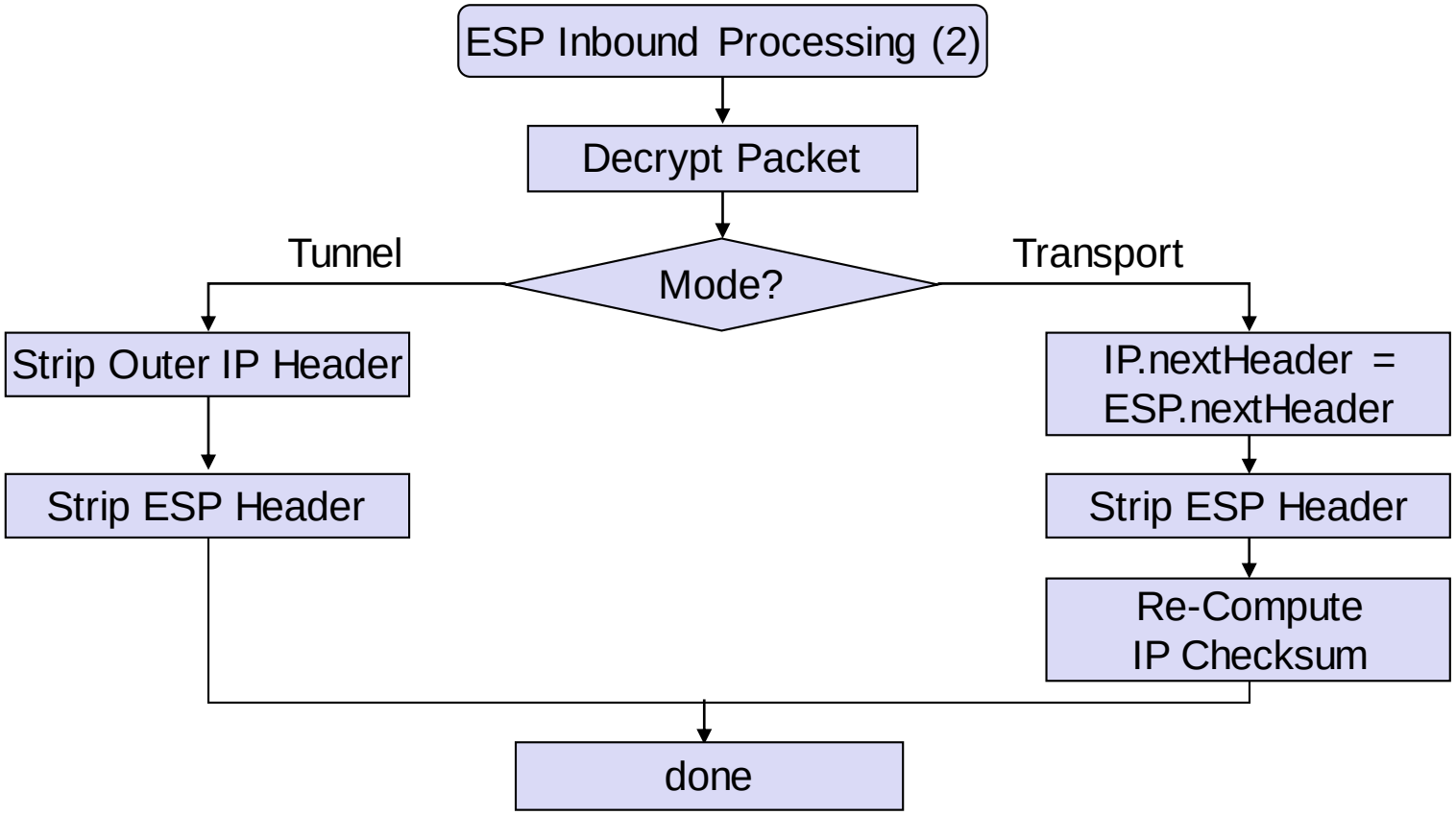




The Encapsulating Security Payload (6)



The Encapsulating Security Payload (7)



The Authentication Header (1)

AH is a generic security protocol that provides to IP packets:

- Replay protection
- Data origin authentication, by creating and adding MACs to packets that refer to the IP header

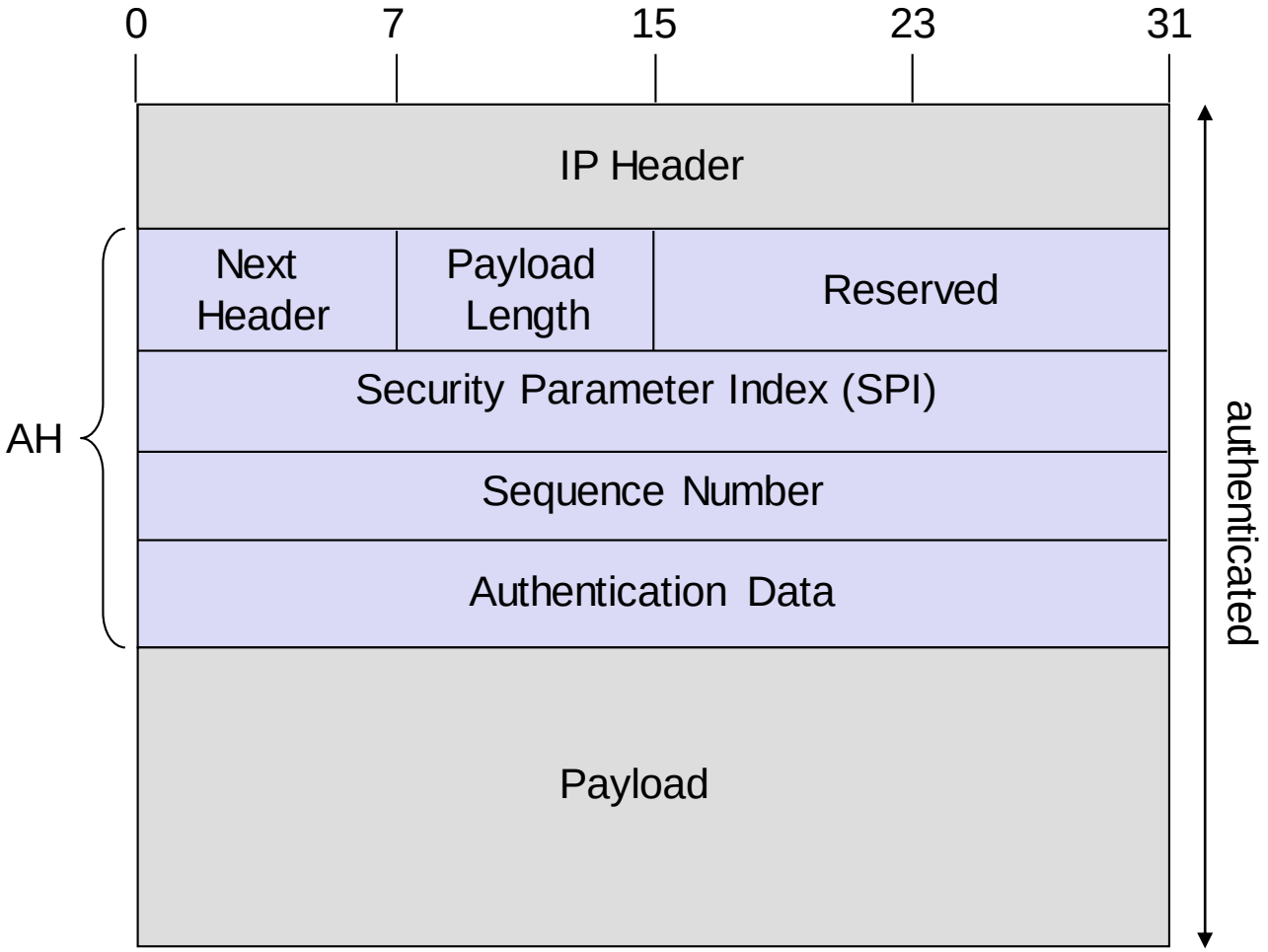
Like ESP, the AH definition is divided into two parts:

- The definition of the base protocol [RFC4302]:
  - Definition of the header format
  - Basic protocol processing
  - Tunnel and transport mode operation
- The use of specific cryptographic algorithms with AH:
  - Authentication: HMAC-MD5-96 [RFC2403], HMAC-SHA-96 [RFC2404]

If both ESP and AH are to be applied by one entity, then ESP can be applied first (but it can the other way around):

- This results in AH being the outer header
- Advantage: the ESP header can also be protected by AH
- Remark: two SAs (one for each AH, ESP) are needed for each direction

The Authentication Header (2)

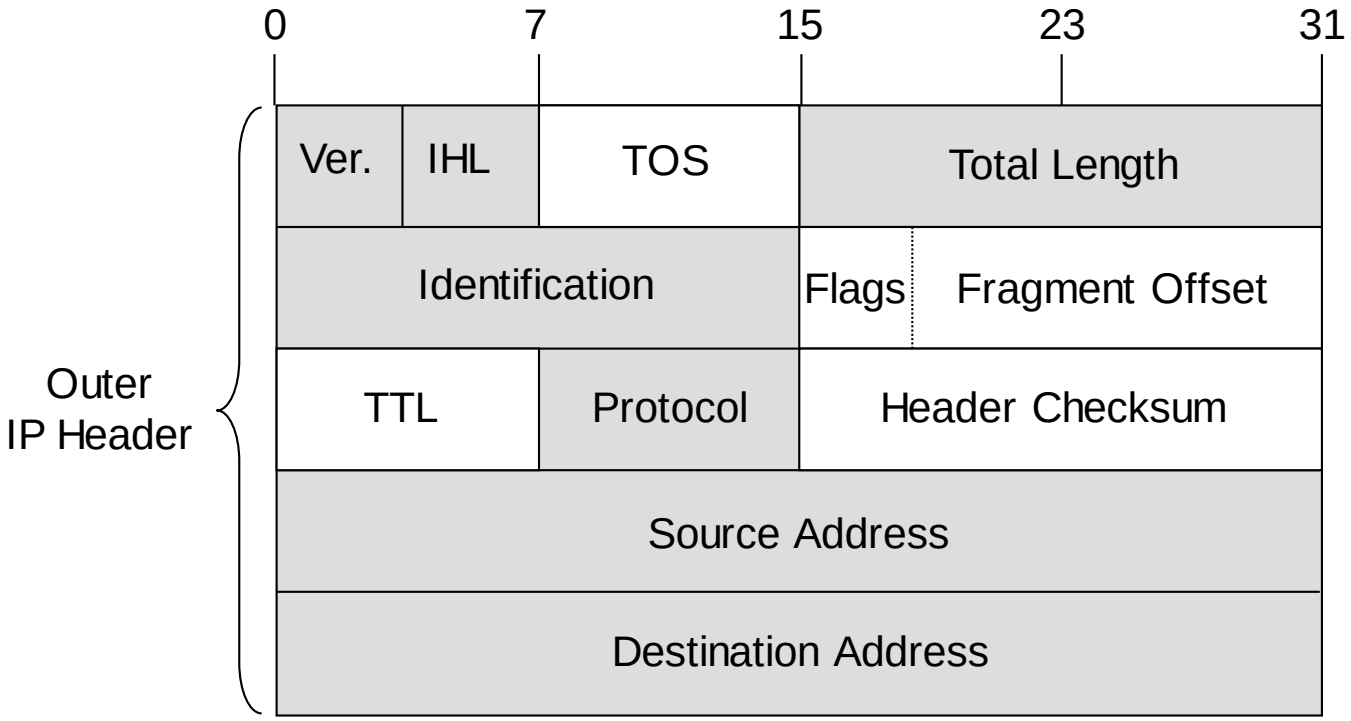


The AH header immediately follows an IP header
The next-header field of the preceding header indicates “51” for AH
In tunnel mode the payload is a complete IP packet

The Authentication Header (3)

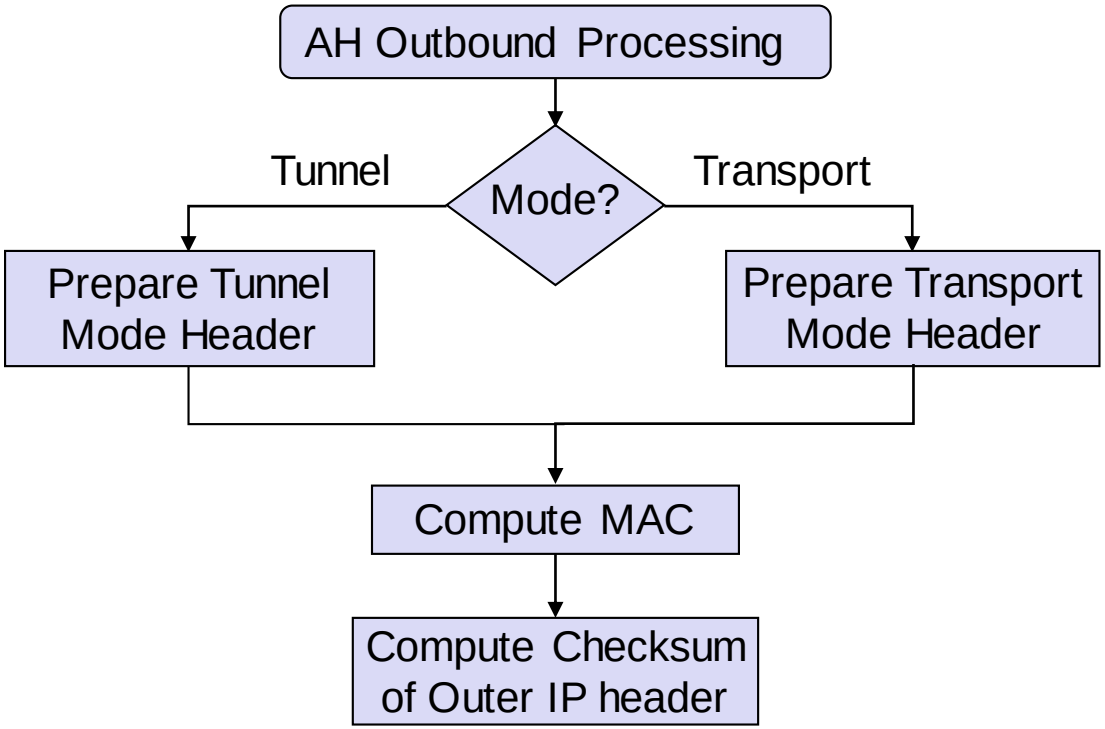
Although AH also protects the outer IP header, some of its fields *must not* be protected as they are subject to change during transit:

- This also applies to mutable IPv4 options or IPv6 extensions
- Such fields are assumed being zero when computing the MAC

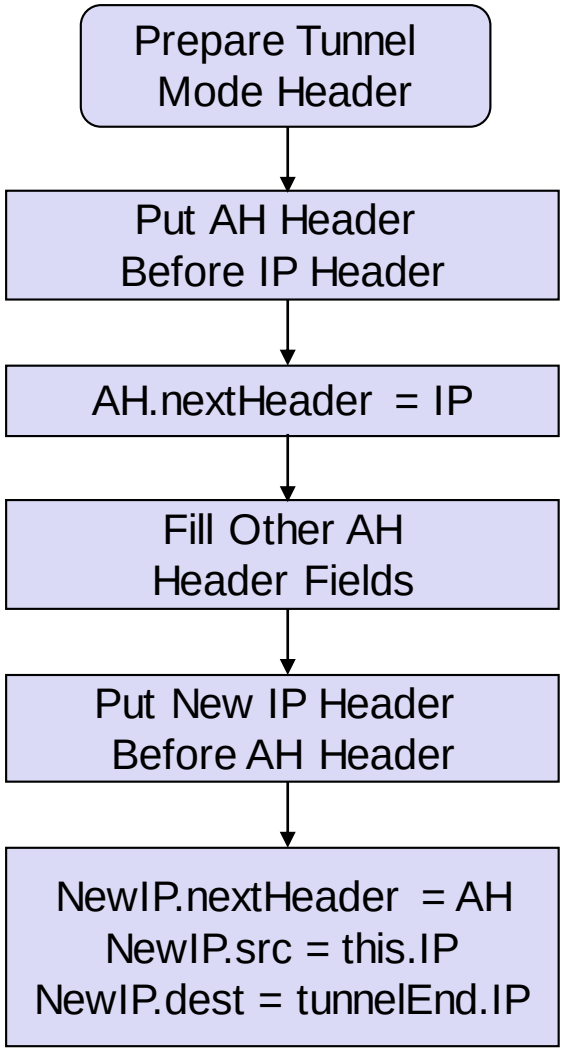


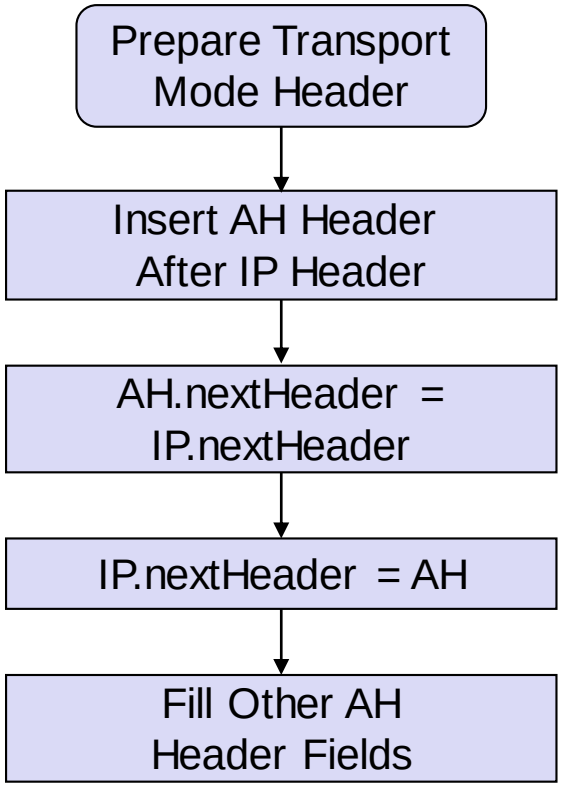
All immutable fields, options and extensions (gray) are protected

The Authentication Header (4)

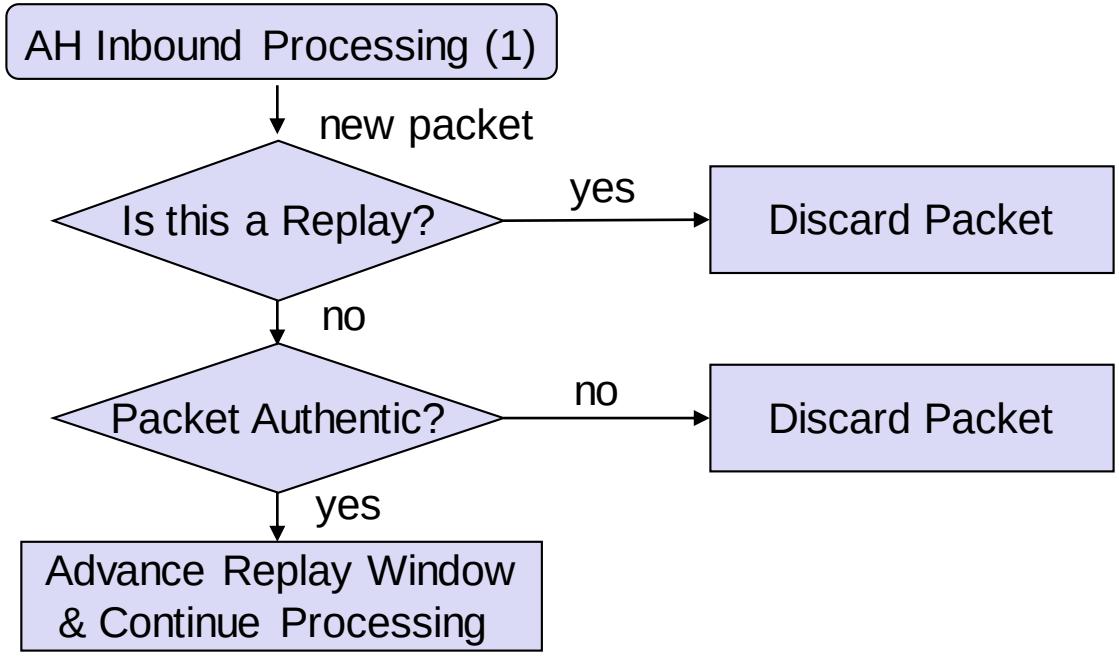


The Authentication Header (5)

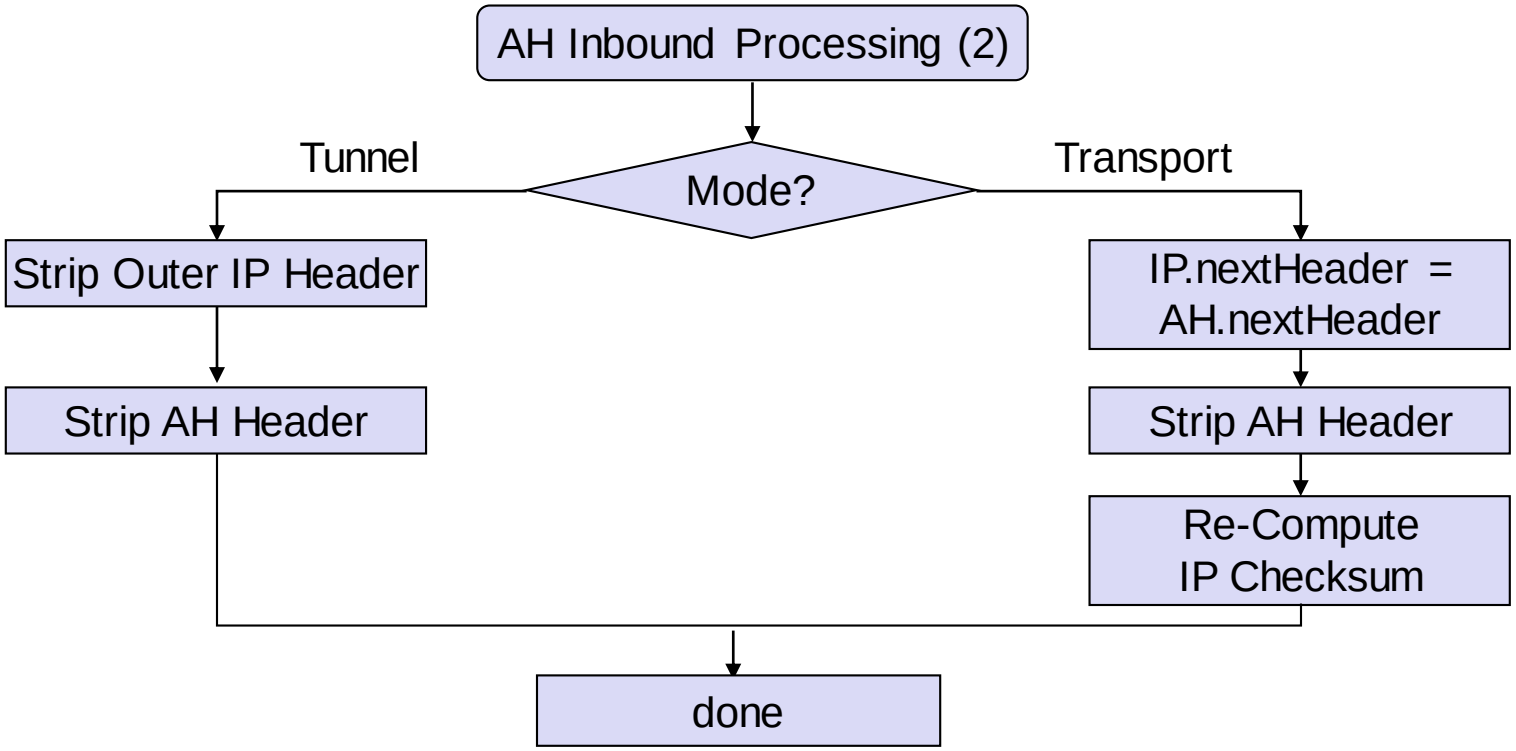




The Authentication Header (6)



The Authentication Header (7)



IPSec's Use of Cryptographic Algorithms (1)

Requirements on an IPSec implementation are classified into different classes

- "MUST", "MUST NOT", "REQUIRED", "SHOULD", "SHOULD NOT" and "MAY"
- "SHOULD+", "SHOULD-", "MUST- "

Requirements on an IPSec implementation of the ESP protocol:

Requirement	***Encryption Algorithm***	***Reference***	***Notes***
MUST	NULL		(1)
MUST-	3DES-CBC	[RFC2451]	
SHOULD+	AES-CBC with 128-bit keys	[RFC3602]	
SHOULD	AES-CTR	[RFC3686]	
SHOULD NOT	DES-CBC	[RFC2405]	(2)

Requirement	***Authentication Algorithm***	***Reference***	***Notes***
MUST	NULL		(1)
MUST	HMAC-SHA1-96	[RFC2404]	(3)
SHOULD+	AES-XCBC-MAC-96	[RFC3566]	(3)
MAY	HMAC-MD5-96	[RFC2403]	(3, 4)

IPSec's Use of Cryptographic Algorithms (2)

Notes

1) Since ESP encryption and authentication are optional, support for the two "NULL" algorithms is required to maintain consistency with the way these services are negotiated. While authentication and encryption can each be "NULL", they MUST NOT be both "NULL".
2) DES, with its small key size and publicly demonstrated and open-design special-purpose cracking hardware, is of questionable security for general use.
3) **The "-96" in the algorithms mentioned above means that the output of the hash function is truncated to the 96 leftmost bits**
4) Weaknesses have become apparent in MD5; however these should not affect the use of MD5 with HMAC

Algorithms for AH are the same as the authentication algorithms for ESP except that "NULL" is not allowed

Overview

- Introduction
  - Brief introduction to the Internet Protocol (IP) suite
  - Security problems of IP and objectives of IPSec
- The IPSec architecture:
  - Overview
  - IP Replay Protection
  - IPSec security protocol modes:
    - Transport mode
    - Tunnel mode
  - IP Security Policies and the Security Policy Database (SPD)
  - Security associations (SA) and the SA Database (SAD)
  - Implementation alternatives
- IPSec security protocols:
  - Encapsulating Security Payload (ESP)
  - Authentication Header (AH)
- Entity Authentication and Key Establishment with the Internet Key Exchange (IKE)

Overview

- Entity Authentication and Key Establishment with the Internet Key Exchange Version 2 (IKE2)
  - Introduction
  - Protocol Exchanges
  - Generation of Keying Material
  - Negotiation of Security Associations
  - Advanced IKEv2 Features
  - Message and Payload Format

IKEv2 – Introduction (1) - Refresher

Prior to any packet being protected by IPSec, a *Security Association* (SA) has to be established between the two "cryptographic endpoints" providing the protection

SA establishment can be realized:

- *Manually*

  ➔ regarding key management, this approach does not scale well in large networks, and keys have to be distributed confidentially

  ➔ Manual establishment can be used only in very restricted configurations (e.g. between two VPN gateways) and should not be a permanent solution

- *Dynamically*, using a standardized authentication & key management protocol

A standardized method for SA establishment is the *Internet Key Exchange* protocol (IKE)

IKEv2 – Introduction (2)

IKEv2 was standardized in [RFC4306] (Dec. 2005)

- Parts of IKE (version 1) were poorly specified and the specification was spread over multiple, overlapping documents (RFCs 2407, 2408, 2409)
- IKEv2 provides a unified authentication and key establishment protocol, which tries to achieve a reasonable tradeoff between features to choose from, overall protocol complexity and reasonable security under a realistic threat model

➔In the following, we will restrict our discussion to IKEv2

[RFC5996] obsoletes [RFC4306] (Sep. 2010)

IKEv2 – Introduction (3)

IKE runs on

- UDP, ports 500 and 4500 are the corresponding “well-known” ports

IKEv2 provides

- *Mutual authentication* of the “Initiator” and the “Responder”
- Negotiation of *cryptographic suites* (a complete set of algorithms used for security associations)
- Support for *DoS mitigation* by the use of *cookies*
- Integrated support for requesting an IP address (remote address acquisition), which is useful for VPNs

IKEv2’s latency in the common case is 2 round trips (i.e. 4 messages)

Basic Protocol Design – Protocol Exchanges (1)

All IKEv2 communications consist of pairs of messages:

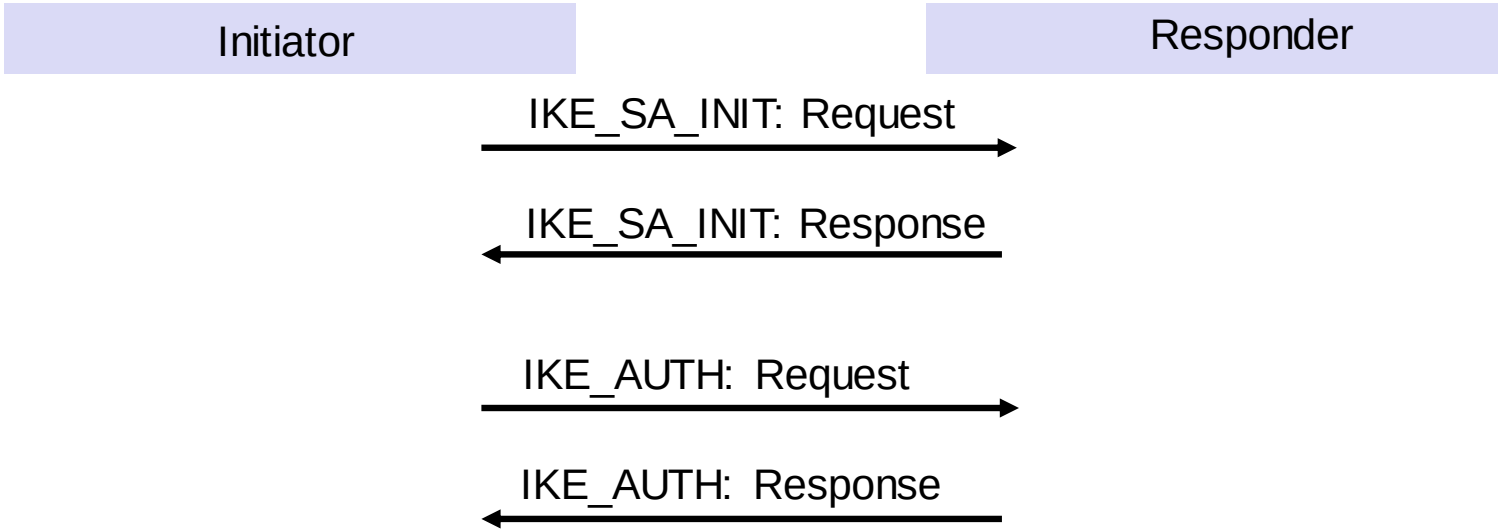
- a request and a response.

Every request requires a response

The pair (request, response) is called an *exchange*

An IKEv2 protocol run starts with two exchanges:

- *IKE_SA_INIT*
- *IKE_AUTH*



Basic Protocol Design – Protocol Exchanges (2)

IKE_SA_INIT

- negotiates security parameters for an IKE security association (*IKE_SA*), sends nonces and Diffie-Hellman values
- *IKE_SA* is a set of security associations used to encrypt and integrity-protect all the remaining IKE exchanges

IKE_AUTH

- authenticates the previous messages, transmits identities, proves knowledge of the secrets corresponding to the identities and creates a first CHILD_SA
- A CHILD_SA is a set of security associations used to encrypt and/or integrity- protect data with AH/ESP
- "CHILD_SA" is synonymous to the common definition of an SA for IPSec AH and ESP in RFC's 4301, 4302 & 4303

Basic Protocol Design – Protocol Exchanges (3)

IKE_SA_INIT and *IKE_AUTH* have to be completed in this strict order before any other exchanges

Additional IKEv2 exchanges might be

- *CREATE_CHILD_SA*
  - Used to create another CHILD_SA
  - Can also be used for *re-keying*
- *INFORMATIONAL*
  - Is used for “keep-alive”, deleting an SA, reporting error conditions and other housekeeping

Some remarks

- The IKE_AUTH includes the negotiation of an CHILD_SA simply in order to reduce the latency of IKEv2. The negotiation is CHILD_SA is “piggibacked” by the *IKE_AUTH* exchange
- IKEv2 uses UDP as a transport protocol, which does not provide reliable transport. For all exchanges, it is in the responsibility of the requester to ensure reliability, i.e., to retransmit a request after a timeout has occurred

IKEv2 Message Payloads

A IKEv2 message consists of a message header *HDR* and one or more other fields called „payloads“

Payload types

- AUTH: Authentication
- CERT: Certificate
- CERTREQ: Certificate Request
- CP: Configuration
- E: Encrypted
- ID_i / ID_r: Identification
- KE: Key Exchange
- N_i, N_r: Nonce
- SA: Security Association
- TS_i / TS_r: Traffic Selector
- N: Notify (The Notify payload is used for different purposes)

Other payloads will not be addressed in this Chapter and can be found in [RFC4306]

IKEv2 Message Header

HDR contains

- Security Parameter Indexes (SPIs) for the *IKE_SA*,
- IKE version number (in this case IKEv2 obviously 2),
- various flags

Notation

- *HDR(A,B)* indicates the IKEv2 message header where the initiator's and responder's SPI are respectively *A* and *B*

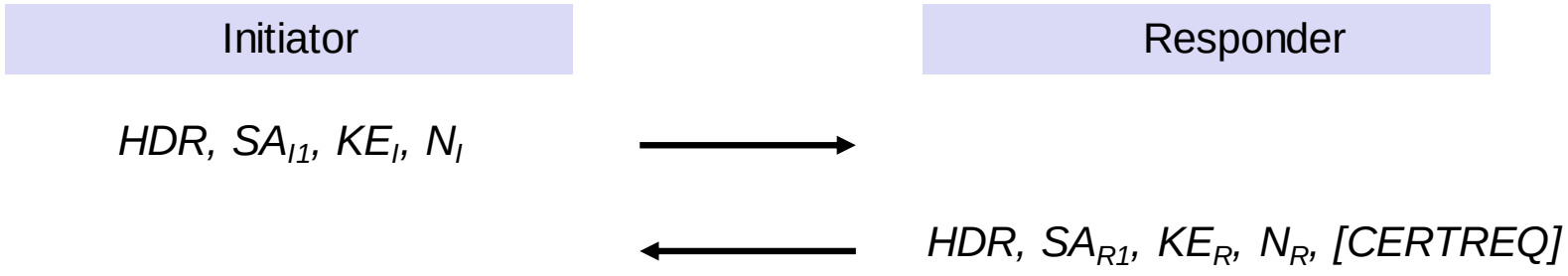
Note that unlike ESP and AH where only the recipient's SPI appears in the header of a message, in IKEv2 the sender's SPI is also sent in every message.

In the first message of an initial IKEv2 exchange, the initiator will not know the responder's SPI value and will therefore set that field to zero

Notation in the following protocol exchanges

- the protocol participants are called *Initiator* (I) and *Responder (R)*
- "[…]" marks optional components

IKEv2 – IKE_SA_INIT Exchange (1)



SA_{I1}: the cryptographic suites which the Initiator supports

SA_{R1}: the Responder chooses a cryptographic suite from SA_{I1}

KE_I, KE_R: Initiator and Responder's public Diffie-Hellman values, respectively

N_I, N_R: Initiator and Responder's (random) nonces, respectively

CERTREQ: (optional) certificate request payload which is used for authentication via certificates

Generating Keying Material Using Pseudo-Random Functions

In the context of a IKE_SA, 4 cryptographic algorithms are negotiated

- Encryption algorithm
- Integrity protection algorithm
- Diffie-Hellman group (i.e. DH parameters *p* and *g*)
- pseudo-random function (prf),

Notation

- In the remaining of this chapter, prf is a pseudo-random function with two input parameters
- E.g. prf (K,S) = $HMAC_K$ (S) = H(K $\oplus$ opad | H(K $\oplus$ ipad | S))
  where H is a cryptographic hash function, such as MD5 or SHA-1

The prf negotiated during the *IKE_SA_INIT* exchange is used for the construction of all subsequent keying material for IKE_SA and CHILD_SAs

Since the amount of keying material needed may be greater than the size of the output of the prf algorithm, prf is used iteratively

- prf+ (K,S) = T1 | T2 | T3 | T4 | ...
- T1 = prf (K, S | 0x01)
- T2 = prf (K, T1 | S | 0x02)
- T3 = prf (K, T2 | S | 0x03)
- T4 = prf (K, T3 | S | 0x04)

Generating Keying Material after IKE_SA_INIT

The DH handshake in the IKE_SA_INIT exchange enables both parties to generate a shared secret key called *SKEYSEED*,

- $SKEYSEED = \text{prf}(N_i \mid N_r, g^{ir})$

Furthermore, each peer generates keying material consisting of 7 different keys

$$SK_d \mid SK_{ai} \mid SK_{ar} \mid SK_{ei} \mid SK_{er} \mid SK_{pi} \mid SK_{pr}$$
$$= \text{prf+} (SKEYSEED, N_i \mid N_r \mid SPI_i \mid SPI_r)$$

4 keys are used for integrity protection and encryption for all the remaining IKE messages in each direction

- SK_{ai} and SK_{ar} are used as keys for integrity-protecting subsequent IKEv2 exchanges
- SK_{ei} and SK_{er} are used for encrypting (and decrypting) messages in subsequent IKEv2 exchanges

SK_d used for deriving new keys for the CHILD_SAs established with this IKE_SA

SK_{pi} and SK_{pr} will be used when generating the AUTH payloads in the *IKE_AUTH* exchange

IKEv2 – IKE_SA_INIT Exchange (2)

After the first exchange,

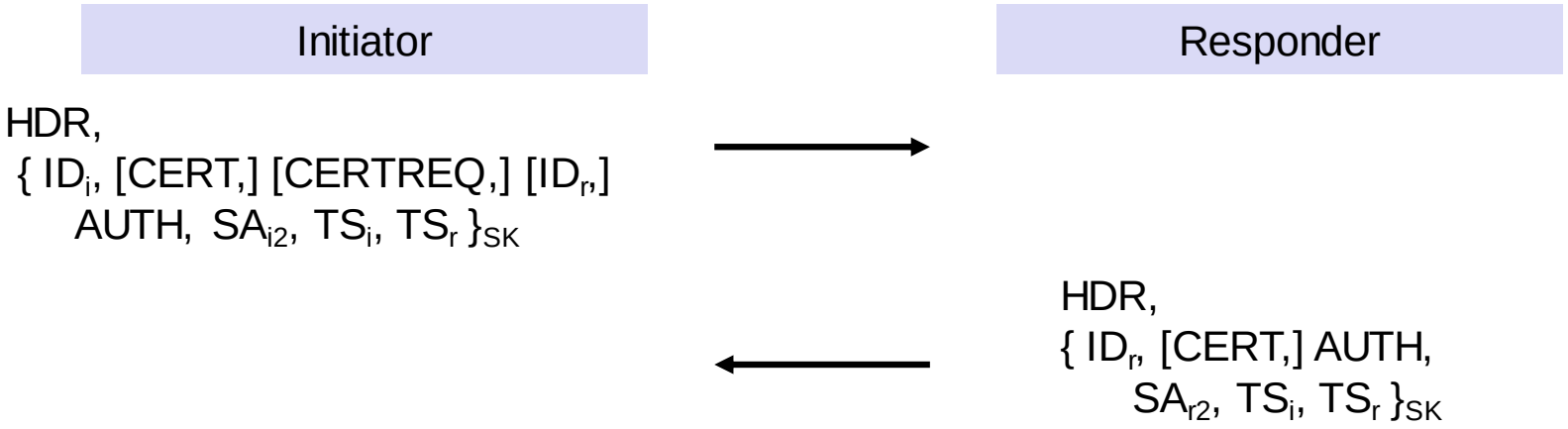
- the DH handshake is complete,
- keying material for the IKE_SA is generated
- however, the shared secret SKEYSEED (and consequently all keys derived from SKEYSEED) is still *unauthenticated*

Both parties still have to verify the identity of the other side in the upcoming IKE_AUTH exchange

All but the headers of all the IKEv2 messages that follow are encrypted and integrity protected

IKEv2 – IKE_AUTH Exchange

In the following, $\{\ldots\}_{SK}$ indicates that these payloads are encrypted and integrity protected using that direction's SK_e and SK_a:



- The Initiator (Responder) asserts his identity in the ID_I (ID_R) payload
- The Initiator (Responder) proves knowledge of the secret corresponding to ID_I (ID_R) and integrity protects the contents of the first (second) message in the IKE_SA_INIT exchange, using the AUTH payload
- The initiator may include a certificate or a certificate chain and a certificate request, if authentication should be using digital signatures
- The remaining payloads (SA_{i2} ,SA_{r2} , TS_i and TS_r) are used for establishment of the first CHILD_SA

IKEv2 – Authentication (1)

IKEv2 supports the authentication of peers using either public key signatures or a long-term pre-shared secret

"Initiator" and "Responder" are authenticated by having each sign (or MAC using a shared secret as the key) a block of data

The resulting value (digital signature or MAC) will be respectively the AUTH payload in the IKE_AUTH exchange

➔ the Initiator/Responder can authenticate the other peer by verifying the validity of the received AUTH payload

The initiator signs the concatenated string of:

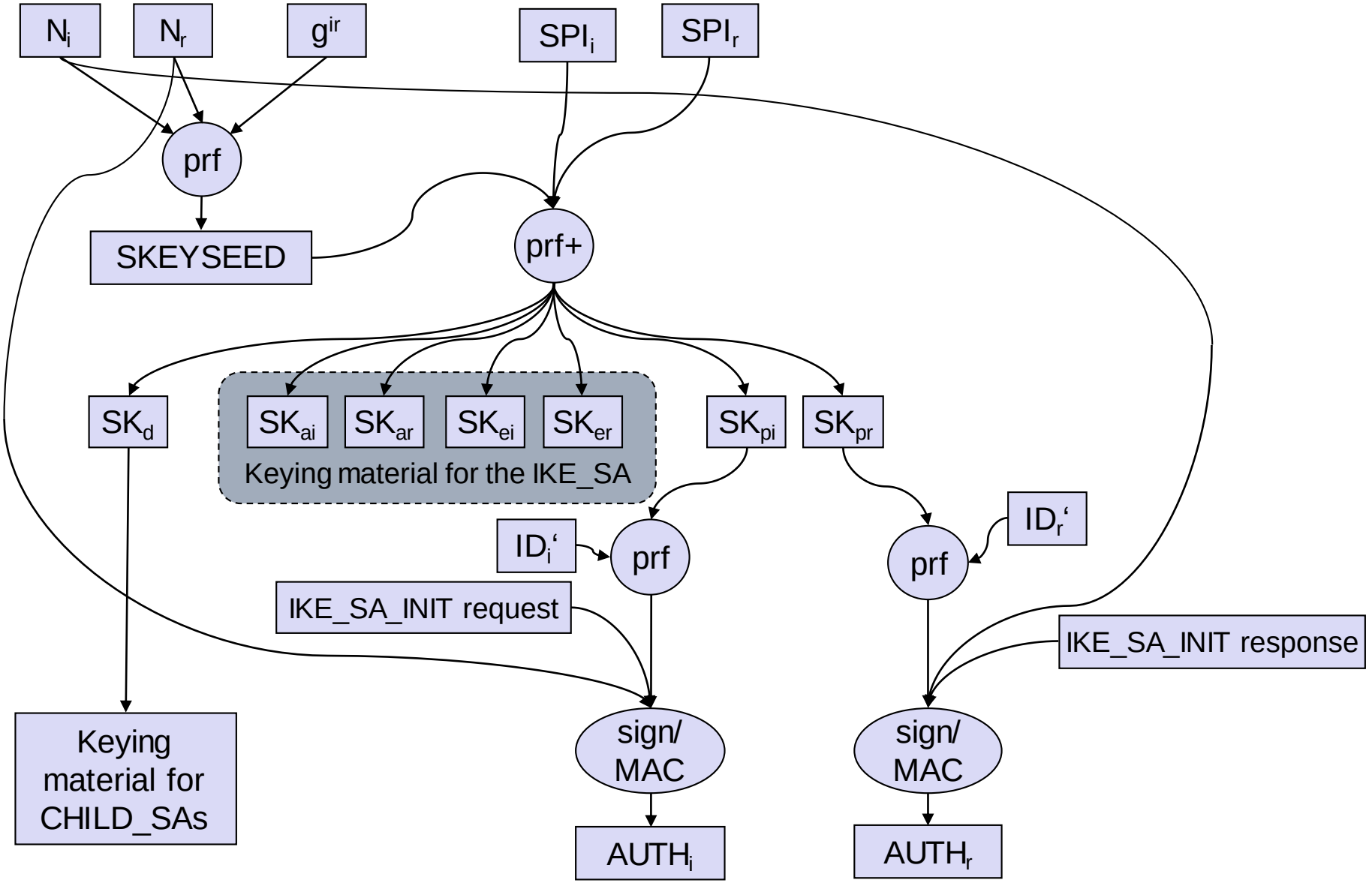
- The whole payload of his IKE_SA_INIT message
- Responder's nonce N_r
- prf (SK_{pi}, ID_i') (the prf algorithm negotiated in the IKE_SA_INIT exchange is used)

The responder signs the concatenated string of:

- The whole payload of his IKE_SA_INIT message
- Initiator's nonce N_i
- prf (SK_{pr}, ID_r')

ID_i', ID_r' are the entire ID payloads excluding the fixed headers

Dependencies between Different IKEv2 Payloads and Generated Keys



IKEv2 – Authentication (2)

The Initiator (Responder) assert his/her identity by including the ID_i' (ID_r') in the generation of the AUTH payload

It is important that each peer includes the other side's (freshly generated) nonce (N_i and N_r) in the generation of the AUTH payload in order to guarantee to the other peer that he/she is alive and that it is not a replayed protocol run

The entire IKE_SA_INIT request (response) is included in the generation of the AUTH payload. This guarantees the integrity of the IKE_SA_INIT exchange

In particular, this guarantees that the proposed algorithms in the SA negotiation (SA_{I1} ,SA_{R1}) and the DH value (KE_I and KE_R) have not been modified by an attacker

SK_{pi} and SK_{pr} are derived from the SKEYSEED and are involved in the generation of the AUTH payload respectively. Therefore, each peer can verify that the freshly generated SKEYSEED is correct

IKEv2 – Authentication (3)

In some cases, it is possible that the Initiator uses a MAC to generate the AUTH payload while the Responder generates a digital signature to generate the AUTH payload

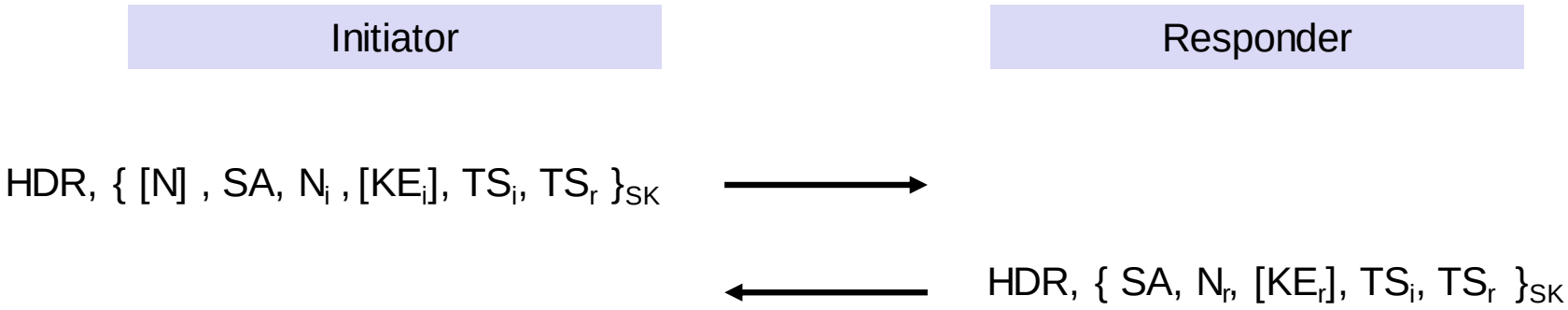
This could be the case, e.g., if the initiator is a user's host connecting to an IPSec-based VPN server, where

- the user's host is authenticated based on the user's password (or a key generated from the user's password)
- the VPN gateway is authenticated using its certificate

In case authentication is based on a certificate, IKEv2 does not enforce or suggest any algorithm for the digital signature. The certificate determines the algorithm used for the signature, e.g. RSA, DSA (El-Gamal), or ECC

IKEv2 – CREATE_CHILD_SA exchange (1)

This exchange may be initiated by either end of the IKE_SA after the initial exchanges are completed



- N: a notify payload; is only used in case in case of re-keying
- SA: SA offer(s)
- N_I, N_R : nonces
- KE_I, KE_R : optional DH values
- TS_I, TS_R : proposed traffic selector payloads

IKEv2 – CREATE_CHILD_SA exchange (2)

A single (CREATE_)CHILD_SA negotiation may result in multiple security associations

Requesting an internal address (and other configuration information, i.e., addresses of DNS servers) on a remote network can be done by including a $CP_{request}$ field in the request

The IPsec Remote Access Server (e.g. a IPSec-based VPN gateway) may manages these addresses, e.g., via DHCP

Don't confuse the IKE SPIs in the HDR with the CHILD_SA SPIs in the SA fields!

Generating Keying Material for a CHILD_SA

Exercise

- Read the Section 2.17 “Generating Keying Material for CHILD_SAs” of [RFC5996] and write down how the necessary keying material for a CHILD_SA is generated

IKEv2 - Protection against flooding attacks

Flooding the responder with IKE_SA_INIT requests from forged IP addresses may result in state and CPU exhaustion attacks

The responder may switch to an alternative protocol (which involves 3 exchanges) in case there's a large number of "half-open" IKE_SAs

In the alternative protocol run, a *Cookie* is exchanged before any state is preserved and any CPU-intensive operations are done at the responder:

1. I → R: HDR(A,0), SA_{I1}, KE_I, N_I
2. R → I: HDR(A,0), N(Cookie);

   where "N(Cookie)" is Notify payload with the Cookie

1. I → R: HDR(A,0), N(Cookie), SA_{I1}, KE_I, N_I
2. R → I: HDR(A,B), SA_{R1}, KE_R, N_R, [CERTREQ]
3. I → R: HDR(A,B), { ID_I, [CERT,] [CERTREQ,] [ID_R,] AUTH, SA_{I2}, TS_I, TS_R }$_{SK_I}$
4. R → I: HDR(A, B), { ID_R, [CERT,] AUTH, SA_{R2}, TS_I, TS_R }$_{SK_R}$

➔ 3. and 4. are the usual IKE_AUTH

➔ This DoS mitigation measure solely assures that the initiator is able to receive and send packets from its IP address from the IP packet header

➔ The Cookie should be generated in a way to not require any state:

i.e., Cookie = <VersionIDofSecret> | Hash(N_I | IP_I | SPI_I | <secret>)

Rekeying

Secret keys should only be used for a limited amount of time and to protect a limited amount of data

2 possible solutions in IKEv2:

1. Establish new security associations
2. Re-establishment of security associations by *rekeying* (optional feature which makes use of the CREATE_CHILD_SA exchange)

➔ Each endpoint is responsible for enforcing its own lifetime policy on SAs

Traffic Selector Negotiation

Traffic selector (TS) payloads allow endpoints to communicate some of their Security Policy Database (SPD) information to their peers

- New feature in IKEv2
- May be dynamically updated
- May serve as a consistency check in some scenarios to assure that the SPDs are consistent

Each TS contains the following information

- Address range (IPv4 or IPv6)
- Port range
- IP protocol ID

IKEv2 - INFORMATIONAL Exchange

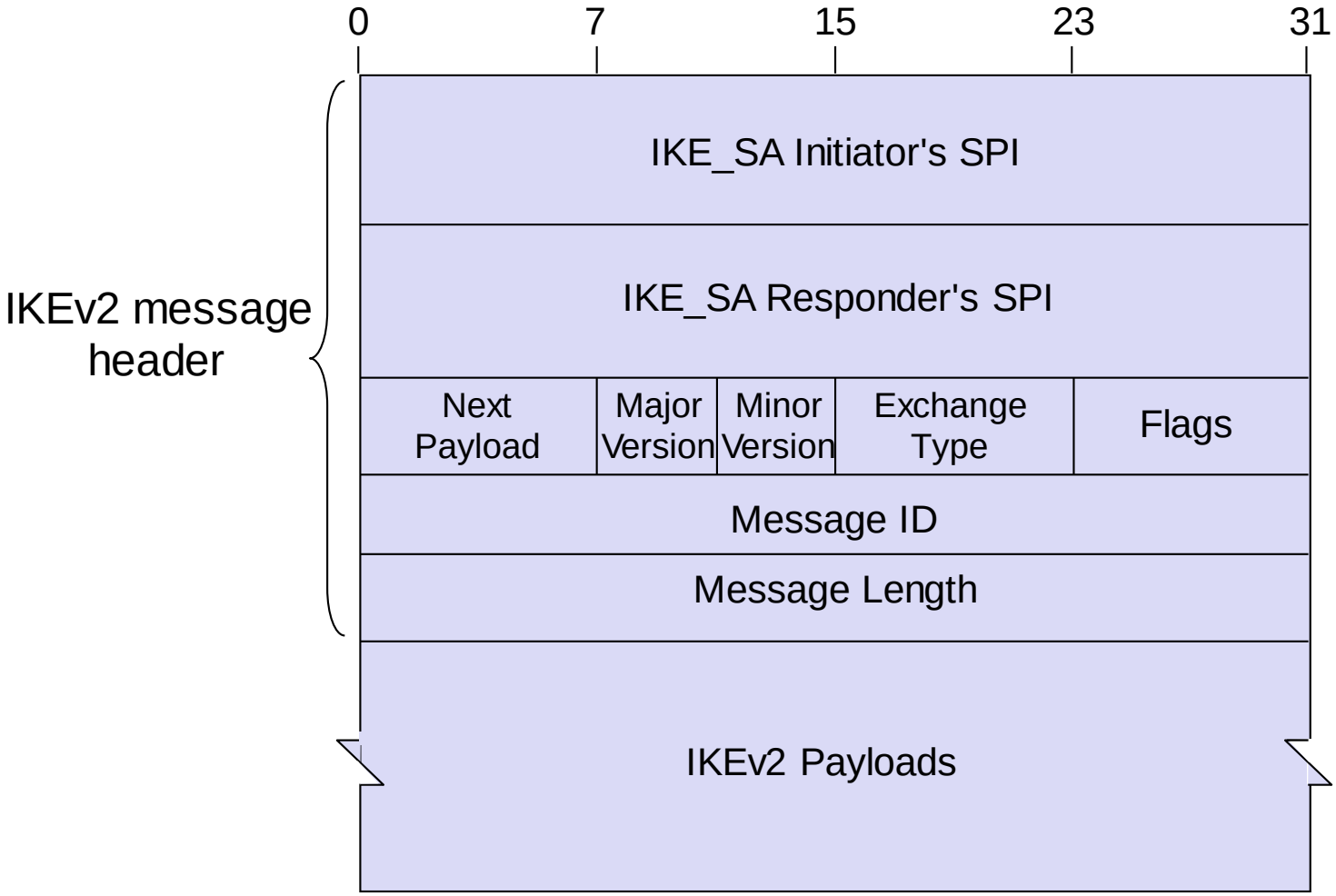
INFORMATIONAL exchanges are used for

- Control messages (delete SA, configuration, …)
- Error messages
- Notifications of certain events
- Keep-alive messages (empty payload)

INFORMATIONAL exchanges must only occur after the initial exchanges and are protected with the IKE_SA

IKEv2 – Basic Message Format (1)

As mentioned above, an IKEv2 message consist of an IKEv2 header and one or more IKEv2 payloads



IKEv2 – Basic Message Format (2)

IKE_SA Initiator's SPI: (8 bytes)

- Identifies the IKE security association at the initiator side.
- Can not be zero

IKE_SA Responder's SPI: (8 bytes)

- Identifies the IKE security association at the responder side.
- Must be zero in the first message of the IKE_SA_INIT exchange (since at this time the Initiator does not know the SPI value at the Responder's side)

Next payload: (1 byte)

- Indicates the type of payload that immediately follows the header

Major & minor version: (4 bits)

- identify the supported versions of the IKE protocol.

Exchange type:

- Indicates the type of exchange being used (IKE_SA_INIT, IKE_AUTH, etc.)

IKEv2 – Basic Message Format (3)

Message ID: (4 bytes)

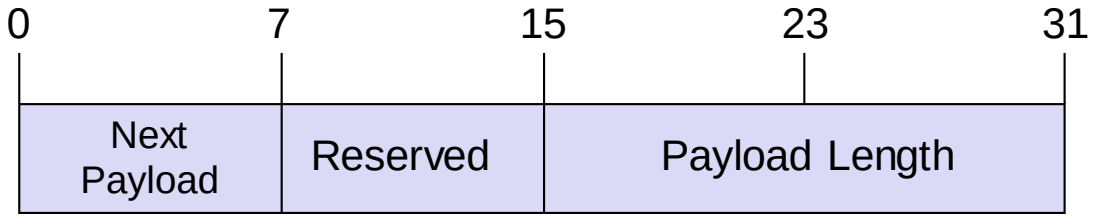
- Used to identify messages to control retransmission of lost packets
- Also used to match responses to the corresponding requests:
  each response message must carry the same message ID as the corresponding request

Message Length: (4 bytes)

- Total length of the message (header + payload) in bytes

Payload:

- All IKEv2 payloads start with a common payload header



- *Next Payload:* the payload type of the next payload in the message
  - e.g. 0 for no next payload, 33 for SA, 34 for KE, 35 for IDi, …
- *Payload Length:* total length of current payload (including this header)

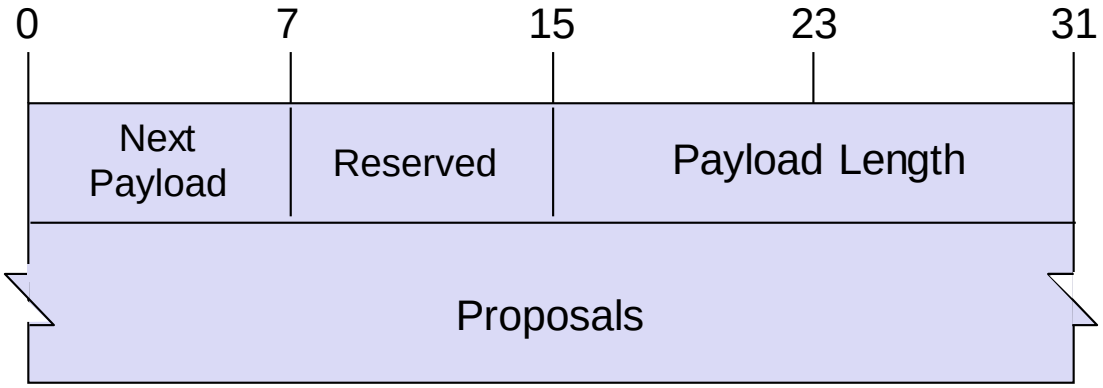
IKEv2 - SA Negotiation (1)

An SA payload may contain multiple “proposals”

Each “proposal” contains a set of security protocols (IKEv2, AH, ESP) with the corresponding cryptographic algorithms to be used

Allowed transform types for:

- AH: integrity check algorithms
- ESP: encryption algorithm and integrity check algorithm
- IKEv2: Diffie-Hellman group, prf algorithm, encryption algorithm and integrity check algorithm



SA payload

IKEv2 - SA Negotiation (2)

Proposals are ordered from most preferred to least preferred

Each proposal is identified with a proposal number: „Proposal #“

- Each proposal substructure (see next slide) has the same Proposal # as the previous one or be one (1) greater
- The first Proposal MUST have a “Proposal #” of one (1)
- If two successive proposal substructures have the same “Proposal #”, it means that the proposal consists of the first structure AND the second
- e.g. a proposal of AH AND ESP would have two proposal substructures, one for AH and one for ESP (but with a different “protocol ID”)

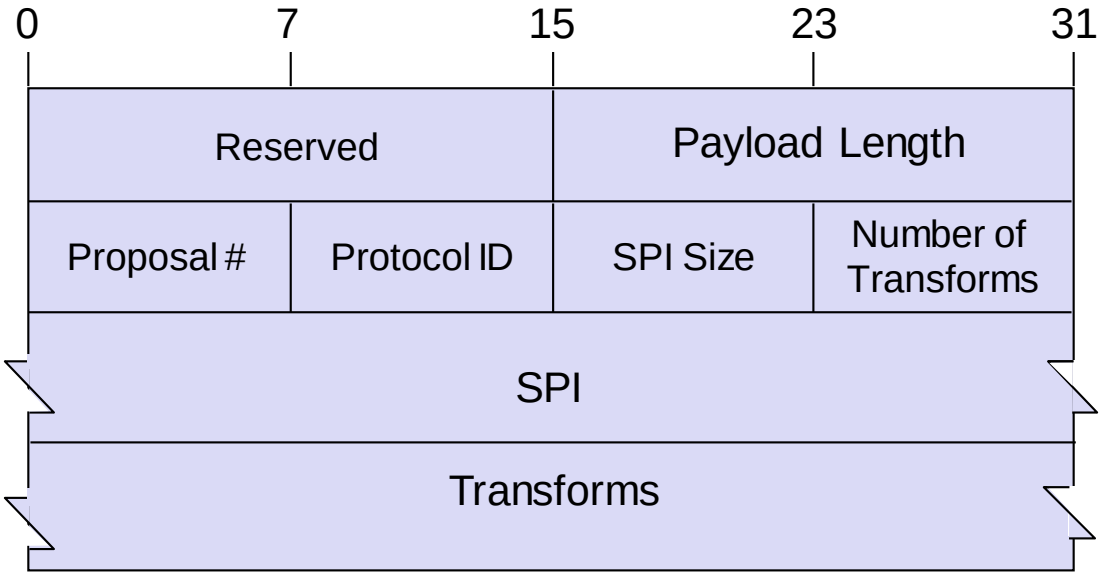
Each proposal substructure contains one or more “transforms”

Each transform specifies a cryptographic algorithm and may contain additional “attributes” that are required for this cryptographic algorithm

Therefore, security associations are encoded in a hierarchical manner:

- SA payload → proposals → protocols → transforms → attributes

IKEv2 – The Proposal Substructure



Proposal # (see last slide)

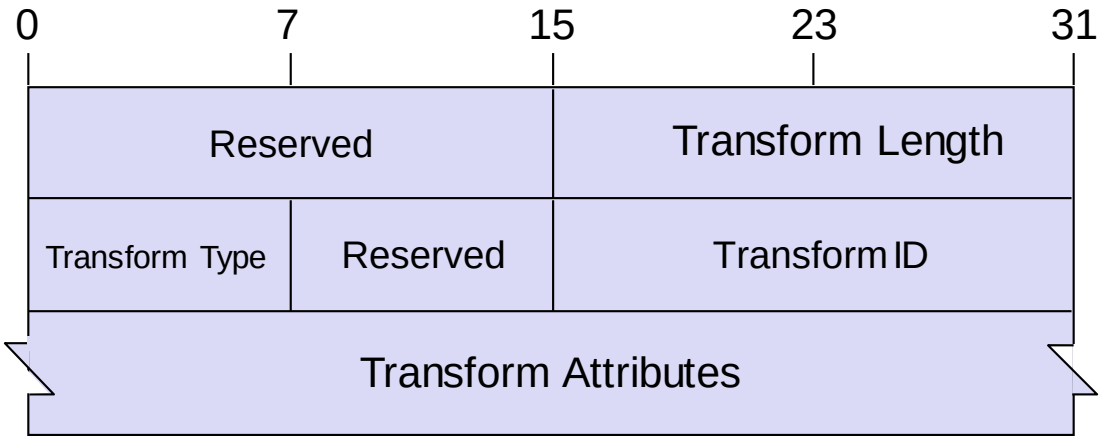
Protocol ID specifies the protocol identifier, i.e. AH or ESP, IKE(v2)

SPI Size specifies the length of the contained SPI value

Number of Transforms specifies how many transforms belong to this proposal

- e.g. if "protocol ID" is „ESP", the proposal may contain several transforms for data integrity and encryption, e.g. ENCR_3DES, ENCR_AES_CTR, AUTH_HMAC_SHA1_96 and AUTH_AES_XCBC_96
- Note however, that in the response, the Responder must choose a unique algorithm for a each purpose, e.g. ENCR_3DES and AUTH_HMAC_SHA1_96

IKEv2 – The Transform Substructure



A transform determines a cryptographic algorithm to be used to secure the communications channel

Transform types:

- Integrity check algorithms (INTEG)
- Encryption algorithm (ENCR)
- Diffie-Hellman group (D-H),
- prf algorithm (PRF)

Each transform is uniquely identified by a *Transform ID*,

- e.g. ENCR_AES_CTR, ENCR_AES_CBC, PRF_HAMC_MD5, etc.
- Identifiers of the different transforms are listed in [RFC4306] Section 3.3.2

Attributes: only one attribute is defined in [RFC4306]

- Key length: in case an algorithm with variable key length is used, e.g. AES

IKEv2 - SA Negotiation (3)

Responders must select a single proposal from the ones proposed in the initiator's request (or reject all offers if none are acceptable)

Each of the proposal substructures in the response must contain a single transform of each transform type

The responder should retain the *Proposal #* field in the proposal payload

- Retention of proposal numbers should speed the initiator's protocol processing by avoiding the need to compare the responder's selection with every offered option
- This value enable the initiator to perform the comparison directly and quickly

The initiator must verify that the SA payload received from the responder matches one of the proposals sent initially

SA Negotiation – Example (1)

This example shows an ESP and AH protection suite:

Two transforms are suggested for ESP

- ENCR: ENCR_3DES
- ENCR: ENCR_AES_CTR (with 256 bit key length)

The responder must select from the two transforms proposed for ESP

a single transform is suggested for AH

- INTEG: AUTH_HMAC_SHA1_96

The resulting protection suite will be:

- either ENCR_3DES and AUTH_HMAC_SHA1_96,
- or ENCR_AES_CTR (with 256 bit key length) and AUTH_HMAC_SHA1_96,

depending on which ESP transform was selected by the responder

SA Negotiation – Example (2)

In this case, the SA payload in the initiator's request will look as follow:

- Proposal 1: ESP:
  - ENCR: ENCR_3DES, ENCR : ENCR_AES_CTR (with 256 bit key length)
- Proposal 1: AH
  - INTEG: AUTH_HMAC_SHA1_96

The SA payload in the responder's response will look as follow:

- Either
  - Proposal 1: ESP:
    - Transform 1: ENCR_3DES
  - Proposal 1: AH
    - Transform 1: AUTH_HMAC_SHA1_96
- Or
  - Proposal 1: ESP:
    - Transform 2 : ENCR_AES_CTR (with 256 bit key length)
  - Proposal 1: AH
    - Transform 1: AUTH_HMAC_SHA1_96

Remainder:

- Two or more proposal substructures with the same proposal # (as shown in the example above) refer to the same proposal
- The SA negotiation in this example will result in 4 SAs: 2 SAs per direction

Summary (1)

IPSec is IETF's security architecture for the Internet Protocol

It provides the following security services to IP packets:

- Data origin authentication
- Replay protection
- Confidentiality

It can be realized in end systems or intermediate systems (gateways)

Two fundamental security protocols have been defined:

- Authentication header (AH)
- Encapsulating security payload (ESP)

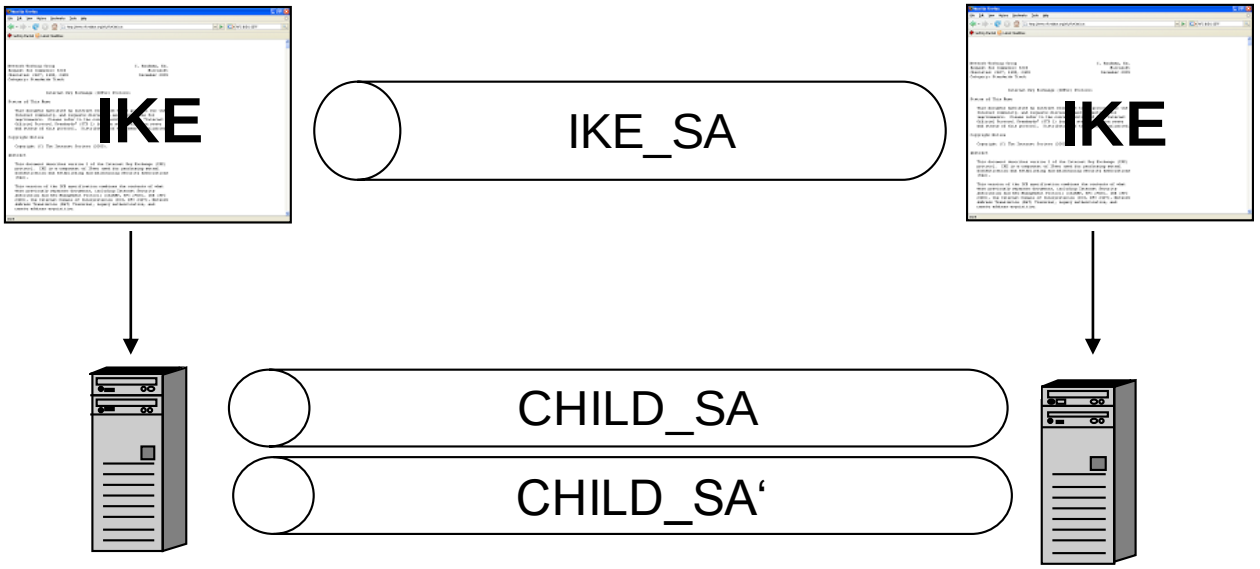
Both protocols can be run in „transport mode" or „tunnel mode"

Authentication, SA negotiation and key management can be realized with the Internet Key Exchange Protocol IKEv2

Authentication can be performed based on certificates or a long-term pre-shared key

IKEv2 provides DoS protection using cookies

Summary (2)



IKE_SA is a set of security associations established after the initial IKEv2 exchange (IKE_SA_INIT) and used to encrypt and integrity-protect all the remaining IKE exchanges

CHILD_SA is a set of security associations used to protect IP traffic with the AH/ESP protocol

AH provides data integrity, replay protection

ESP provides data integrity, replay protection and encryption

Additional References (1)

[Fer98a] N. Ferguson, B. Schneier. *A Cryptographic Evaluation of Ipsec*. http://www.schneier.com/paper-ipsec.html, 1998.

[RFC2403] C. Madson, R. Glenn. *The Use of HMAC-MD5-96 within ESP and AH.* RFC 2403, IETF, 1998.

[RFC2404] C. Madson, R. Glenn. *The Use of HMAC-SHA-1-96 within ESP and AH.* RFC 2404, IETF, 1998.

[RFC2405] C. Madson, N. Doraswami. "*The ESP DES-CBC Cipher Algorithm With Explicit IV"* RFC 2405, IETF, 1998.

[RFC2407] D. Piper. *The Internet IP Security Domain of Interpretation for ISAKMP.* RFC 2407, IETF, 1998.

[RFC2408] D. Maughan, M. Schertler, M. Schneider, J. Turner. *Internet Security Association and Key Management Protocol (ISAKMP).* RFC 2408, IETF, 1998.

[RFC2409] D. Harkins, D. Carrel. *The Internet Key Exchange (IKE).* RFC 2409, IETF, 1998.

[RFC2451] R. Pereira, "The ESP CBC-Mode Cipher Algorithms", RFC 2451, IETF, November 1998

[RFC3566] S. Frankel, „The AES-XCBC-MAC-96 Algorithm and Its Use With IPsec ", RFC 3566, IETF, 2003

[RFC3602] S. Frankel, "The AES-CBC Cipher Algorithm and Its Use with IPsec", RFC 3602 IETF, September 2003

[RFC3686] R. Housley, "Using Advanced Encryption Standard (AES) Counter Mode With IPsec Encapsulating Security Payload (ESP)", RFC 3686 IETF, January 2004

Additional References (2)

[RFC4301] R. Atkinson, S. Kent. *Security Architecture for the Internet Protocol.* RFC 2401, Internet Engineering Taskforce (IETF), 2005

[RFC4302] R. Atkinson, S. Kent. *IP Authentication Header (AH).* RFC 4302, IETF, 2005

[RFC4303] R. Atkinson, S. Kent. *IP Encapsulating Security Payload (ESP).* RFC 4303, IETF, December 2005

[RFC4305] D. Eastlake, „Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH)", RFC 4305, IETF, 2005

[RFC4306] C. Kaufman. *Internet Key Exchange (IKEv2) Protocol*. RFC 4306, IETF, 2005.

[RFC5996] Kaufman, et al. *Internet Key Exchange Protocol Version 2 (IKEv2).* RFC 5996, IETF, 2010

Questions?