CHAPTER 6: Applications: IPSEC

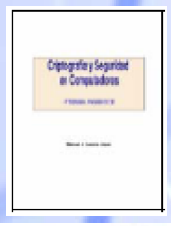
Coding Techniques

Mario Muñoz Organero

Departamento de INGENIERIA TELEMATICA
www.it.uc3m.es

Bibliography

- **Basic bibliography:**
  - **Stallings [16]**
- **Complementary bibliography:**
  - **Ramió [19]**
  - **Lucena [16]**
  - **Comer [30]**



IPsec – Basic Functionality

- **IPsec ensures:**
  - Confidentiality, integrity, and authenticity
- **Allows secure communication in the Internet**
  - Independent from the application or higher protocols
- **Network-layer security instead of application-layer security**
  - Compatible with schemes providing security at the application layer
    - ✓ Can be applied simultaneously
- **Further advantages:**
  - Can be applied to all network traffic
  - Transparent to the applications
  - Transparent to the users

Applications for IPsec

- Secure connection among different branches of the same company
  - Virtual Private Network (VPN)
- Secure remote access to an Intranet through the (insecure) Internet
  - Allows secure remote workers
- Secure communication between peers
- Adding security for electronic commerce applications
  - Mainly B2B
  - B2C y C2C are web-based and thus, use SSL

Components of IPsec

- Almost all components already mentioned:
- Symmetric encryption schemes like DES
  - For confidential communication
- Diffie-Hellman to establish session keys
  - For symmetric key exchange
- Keyed hash functions like HMAC
  - For message authentication
- Digital certificates acting as ID card
- Authentication based on shared secret or public key

Security Association (SA) I

- **Before transmitting encrypted data:**
  - Negotiate algorithms and keys to be used between the two communication peers: SA
- **Definition: Security association (SA)**
  - Negotiated cryptographic parameters:
    - ✓ The IPsec protocols
    - ✓ The keys
    - ✓ The duration of the key validity
  - Unidirectional between the sender and the receiver
- **SAs are associations identified by a SPI (Security Parameter Index)**

Security Association (SA) II

- Two types of SAs:
  - IKE (Internet key exchange) SA: “master”
    - ✓ Long-term validity
    - ✓ Used to negotiate the IPSEC SA
  - IPSEC SA: “session”
    - ✓ Used for data transmission
- For establishing a secure communication between two IP hosts:
  - Negotiate IKE SA
  - Using IKE SA to negotiate IPSEC SA
  - Using IPSEC SA to encrypt the data to transmit
- The IKE SA is of long-term duration.

Why are two types of SAs?

- We start with a non-secure channel
- We negotiate a secure channel first
  - We provide a series of security services:
    - ✓ Authentication
    - ✓ Integrity
    - ✓ Confidentiality
- Will all data need these services?
  - No, some data will require confidentiality only. Others, integrity only…
- Besides:
  - Sessions keys must be changed after some time to difficult cryptanalysis.

Data Transmissions

- **After the negotiation of an IPSEC SA:**
  - **Transmission of data possible, adding a special header to the data**
    - ✓ **Specifying the negotiated IPSEC SA the data belongs to**
    - ✓ **Containing necessary cryptographic information**
    - ✓ **The header is a necessary tradeoff.**

Data Transmissions

- **IPsec has high flexibility:**
  - **Possibility to choose the cryptographic service to use:**
    - ✓ **Integrity**
    - ✓ **Confidentiality**
    - ✓ **Or both**
    - ✓ **Authentication is always present**
    - ✓ **By means of fields at the header**

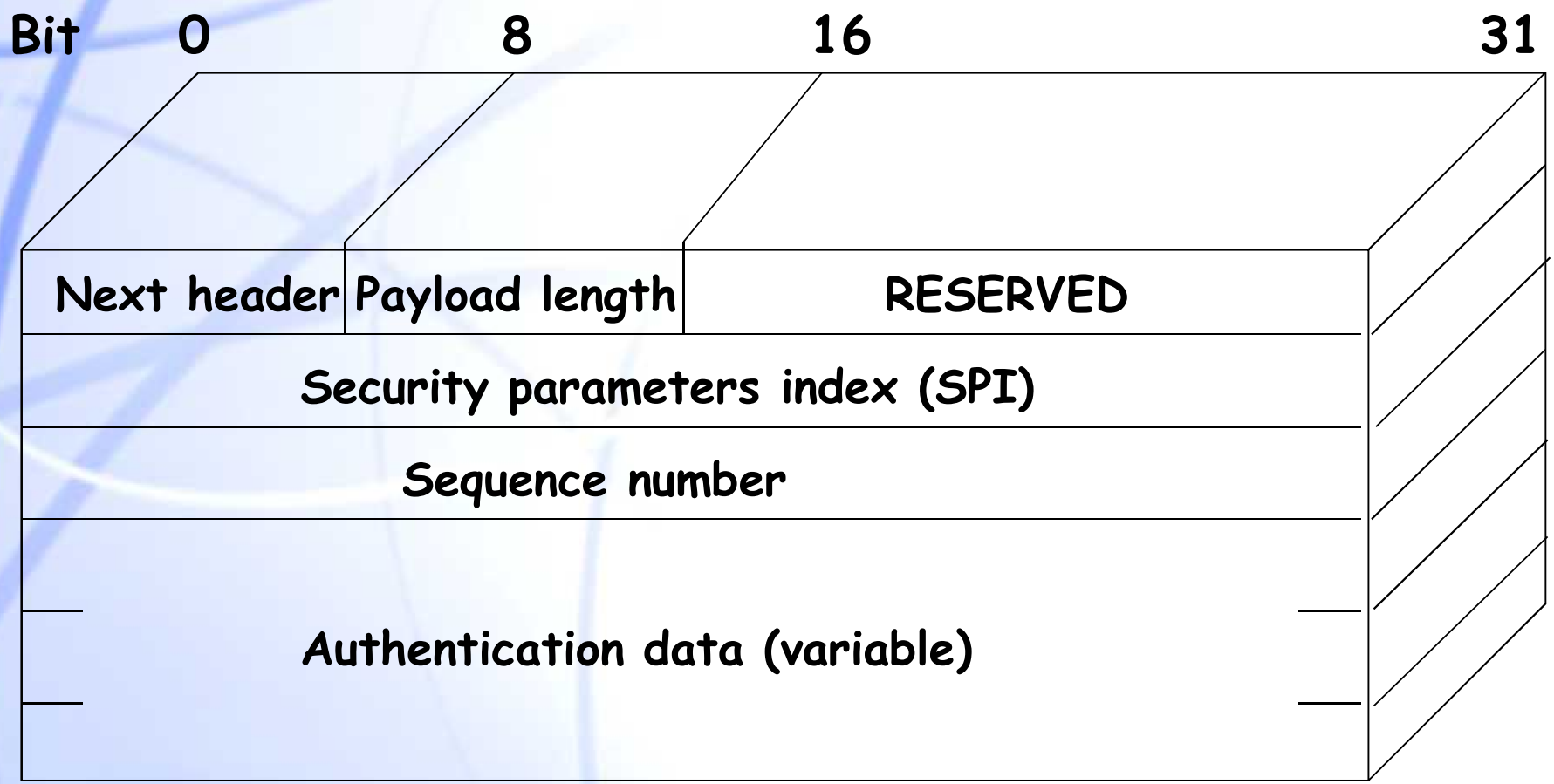
IP Header Extensions

- **Authentication Header (AH)**
  - Guarantee of integrity and authenticity of the data sent in the IP packets
  - No confidentiality (i.e., encryption)
  - Uses a keyed hash function
- **Encapsulating Security Payload (ESP)**
  - It ensures confidentiality of the data
  - It ensures authentication
  - Integrity in ESP is optional

Possibilities

- **Two flavors:**
  - AH (protocol 51 or 33 hex)
  - ESP (protocol 50 or 32 hex)
- **Two modes:**
  - Transport mode
    - ✓ Client to client
  - Tunnel mode
    - ✓ Inserts the original packet in a new IP packet
    - ✓ Useful for interconnecting gateways
    - ✓ Useful for remote access to a corporate intranet by means of Internet

Fields of an Authentication Header



Fields of AH

- **Next Header:** Identifies the protocol of the transferred data
- **Payload length:** size of AH header in 32 bits words minus 2.
- **SPI:** Identifies the security parameters, which, in combination with the IP address, identify the security association implemented with this packet.
  - Values up to 255 are reserved
- **Sequence number:** A monotonically increasing number, used to prevent replay attacks.
- **Authentication data:** Contains the *integrity check value* (ICV) necessary to authenticate the packet; it may contain padding. It is a hash.

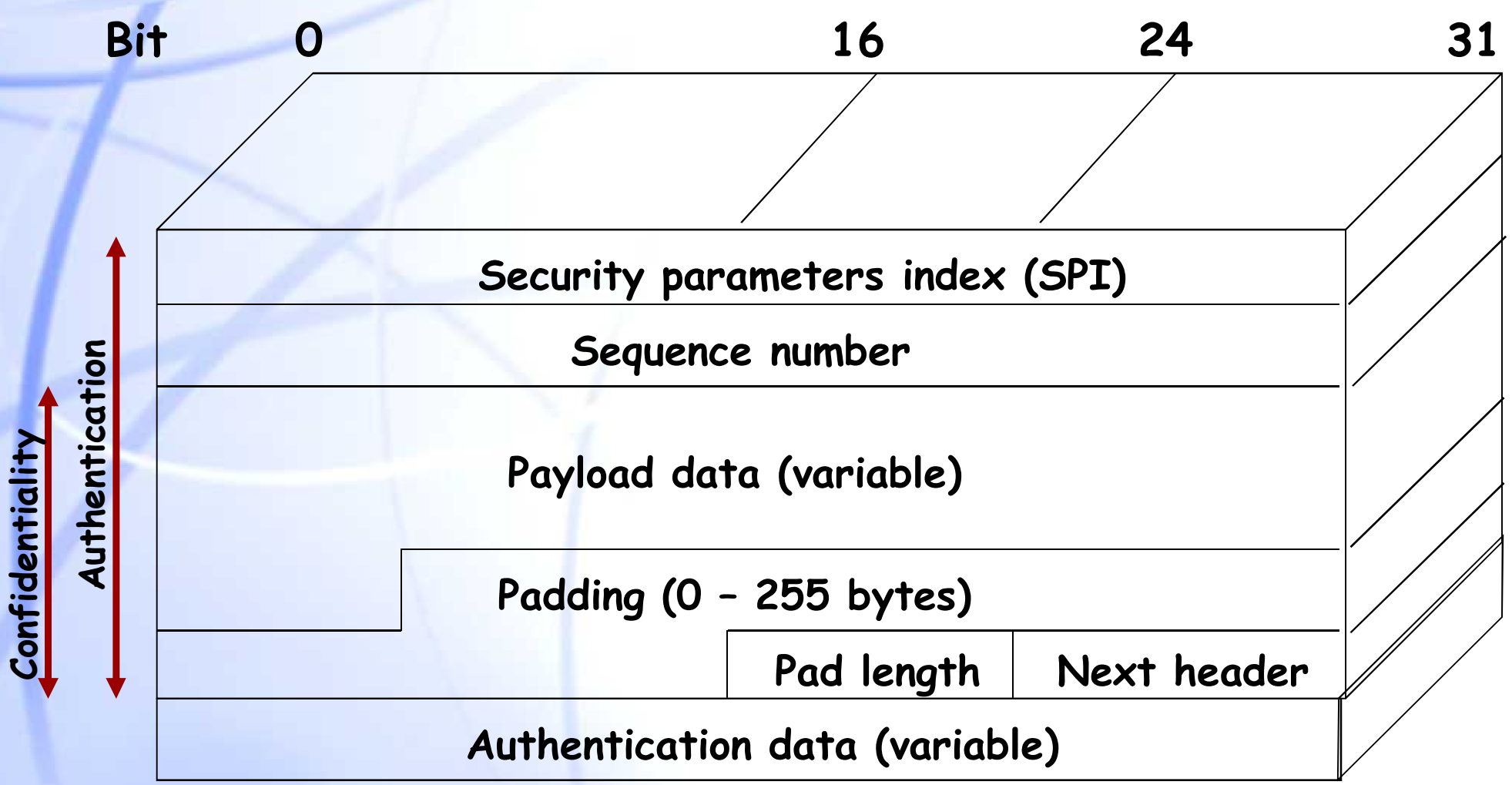
Hash computation

- Put to 0 the variable fields of the IP header and the hash value in the AH header
  - Type of Service (TOS)
  - Flags
  - Fragment Offset
  - Time to Live (TTL)
  - Header Checksum
- Compute hash of the resulting IP packet with the key
- Insert the hash value in the AH header

Compulsory options

- At least these two options must be implemented:
  - HMAC-MD5-96
  - HMAC-SHA-1-96
- In both cases, the AH header has 24 bytes and the “payload length” is 4.

ESP Format


Bit
0
16
24
31
Security parameters index (SPI)
Sequence number
Payload data (variable)
Padding (0 – 255 bytes)
Pad length
Next header
Authentication data (variable)
Confidentiality
Authentication

ESP fields

- SPI and sequence number (like in AH)
- Data: encrypted content of higher layer protocols according with “next header”
  - Exception: if the algorithm requires exchange
- Padding:
  - Must guarantee alignment (multiple of 4 bytes)
  - Must guarantee that length data + padding + padding length field + “next header” is a multple of the size of the block (if a block cipher is being used)
  - By default it is proposed a padding which starts at 1 and is increased byte by byte (by 1 increments).
    - ✓ Provides minimum integrity

IPSEC and NAT

- **AH is incompatible with NAT**
  - Because integrity is also applied to IP addresses
  - Solution: put NAT outside the VPN
- **ESP transport mode is also incompatible with NAT**
  - NAT changes IP addresses therefore TCP checksum must be recomputed.
  - Solution: put NAT outside the VPN
- **ESP tunnel mode may work together with NAT. But not with PAT (NAPT).**
  - PAT causes problems because:
    - ✓ TCP/UDP ports are encrypted and are not accessible
    - ✓ There are problems with SAs negotiation, which must be done in port 500.
  - Solution: integrate NAT management equipment and tunnel.

IPSEC

Establishing SAs

Departamento de INGENIERIA TELEMATICA
www.it.uc3m.es

What are we going to study?

- IKE (Internet key exchange) and session set up
  - IKEv1
  - IKEv2
- Security associations (IKE SA e IPSEC SA).

Our needs

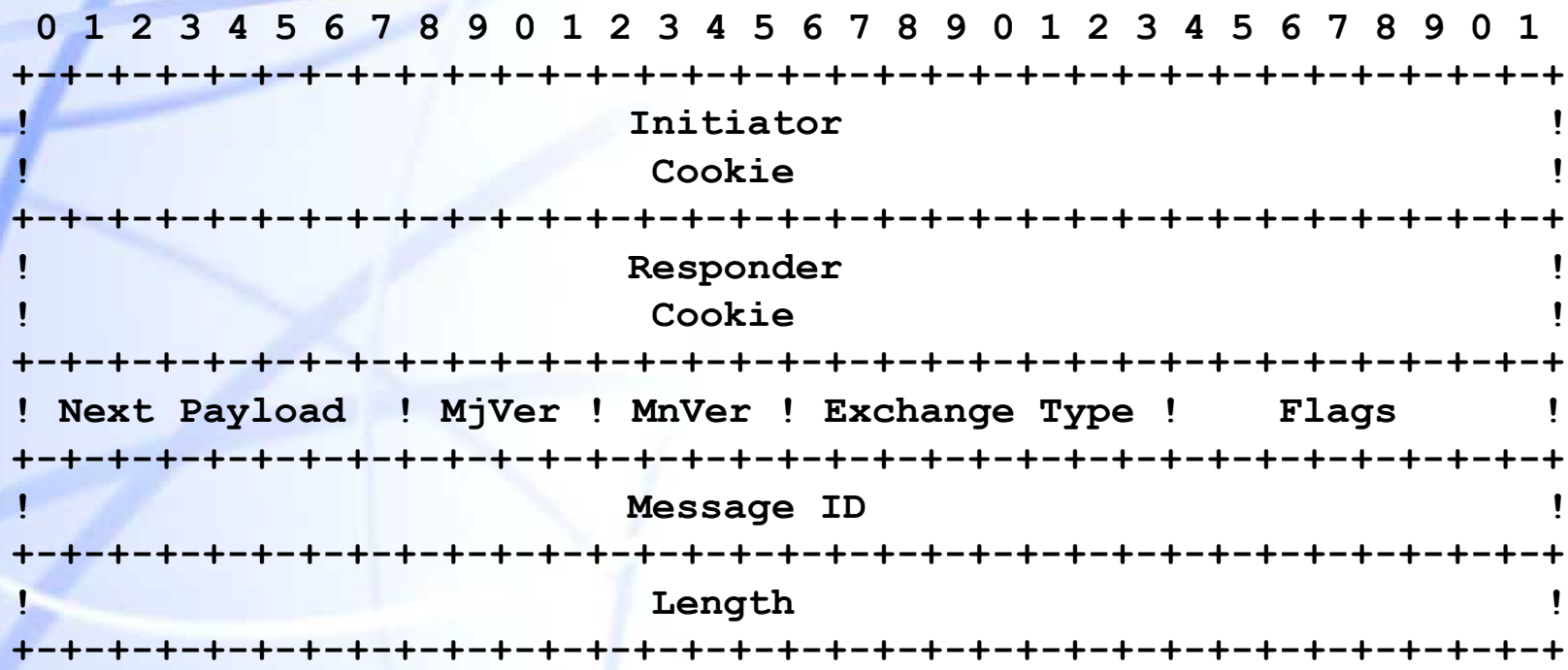
- Negotiate the cryptographic parameters of the security associations:
  - Session keys
  - Encryption algorithm
  - Hash functions
- Assure the identity of the other extreme:
  - Authentication
- Sometimes guarantee also:
  - Anonimity – nobody else knows about the identity of the participants
  - Non-repudiation – use of digital signatures
- Make easy the quick refresh of keys

Was there anything before 1998?

- **ISAKMP** (*Internet Security Association and Key Management Protocol*) protocol that defines formats and SAs set up message exchanges.
- **Oakley** describes a series of specific mechanisms (modes) to exchange keys, based on Diffie-Hellman.
- **SKEME** describes a versatile technique to exchange keys that provides anonymity repudiability, and quick refresh.

- The result for **IPSEC** → IKE: protocol that uses part of Oakley and part of SKEME together with ISAKMP to set up the SAs.

ISAKMP Header



- **Cookies depend of data from the sender and the receiver as well as their IP addresses, and they are kept during the whole session.**

ISAKMP Payload

- Values for the "*Next Payload Type*" field in the ISAKMP header:

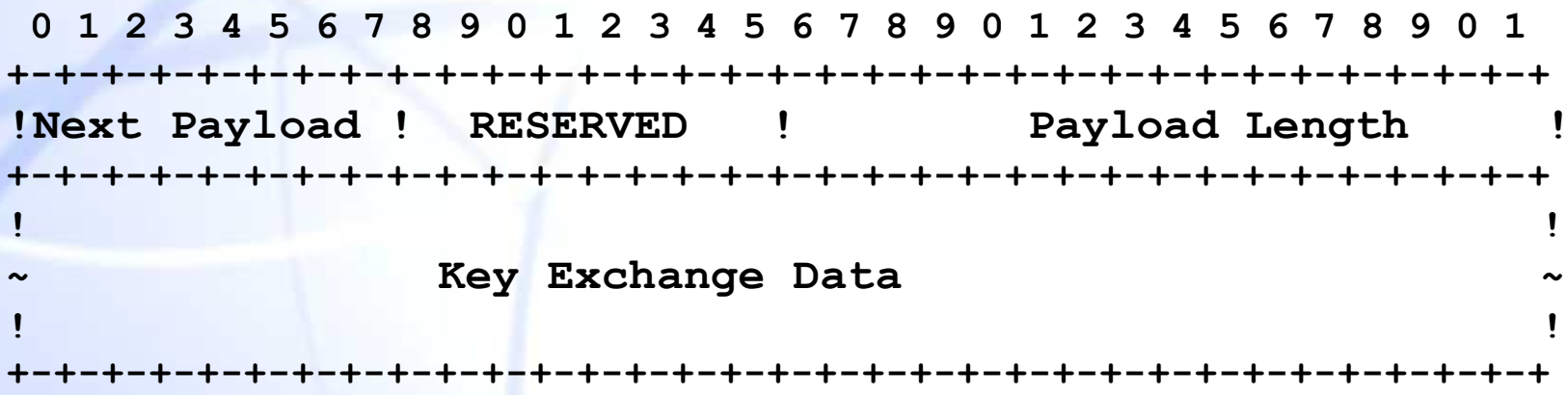
NONE	0
Security Association (SA)	1
Proposal (P)	2
Transform (T)	3
Key Exchange (KE)	4
Identification (ID)	5
Certificate (CERT)	6
Certificate Request (CR)	7
Hash (HASH)	8
Signature (SIG)	9
Nonce (NONCE)	10
Notification (N)	11
Delete (D)	12
Vendor ID (VID)	13
RESERVED	14 - 127
Private USE	128 - 255

Exchange Modes

- Values for the field “Exchange Type”:
  - NONE 0
  - Base 1
  - Identity Protection 2
  - Authentication Only 3
  - Aggressive 4
  - Informational 5
  - ISAKMP Future Use 6 - 31
  - DOI Specific Use 32 – 239
  - Private Use 240 - 255

Example: Payload for Key Exchange

- The following fields are used for a Key Exchange:



IKE versions

- **Two versions:**
  - **Version 1 (1998)**
  - **Version 2 (2005)**
    - ✓ **Optimizes the message exchange**

IKE v1

- **Two negotiations:**
  - **IKE SA set up**
    - ✓ **Start unencrypted on port 500 UDP**
  - **Use of IKE SA to set up IPSEC SAs**
    - ✓ **Use secure channel provided by the IKE SA**
    - ✓ **Continue on port 500.**
    - ✓ **We can establish as many IPSEC associations as we want**
      - Data with different security requirements
      - Session renegociation

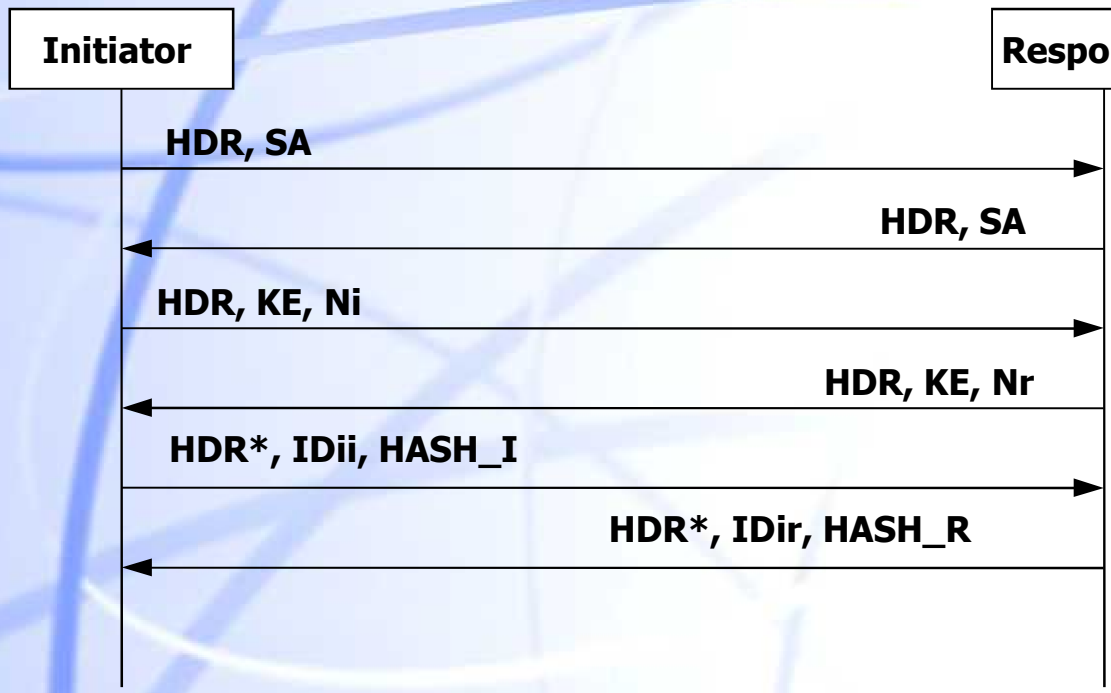
Minimum requisites to support

- **Implementations must support, at least:**
  - **DES with CBC to cipher**
  - **MD5 and SHA as hash functions**
  - **Shared secret authentication**
- **Also recommended:**
  - **3DES**
  - **RSA signatures and RSA encryption for authenticantion**

IKEv1: modes

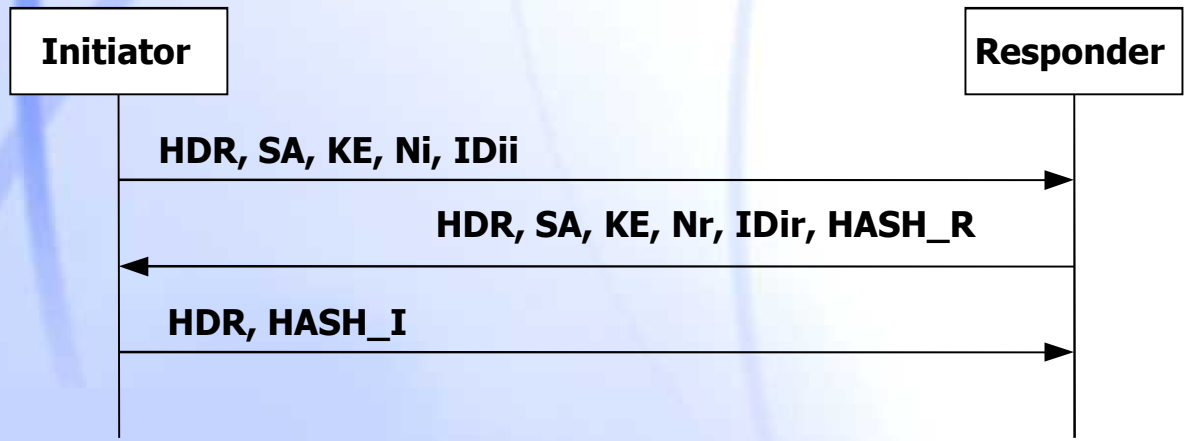
- **Phase 1: Negotiation of the IKE SAs**
  - Authentication using a PKI or a shared secret
  - Two type of negotiations:
    - ✓ "Main Mode" (6 messages) & "Agressive Mode" (3 messages)
  - "Aggressive Mode" more efficient
    - ✓ Shorter negotiation
    - ✓ But: No protection of the identity
- **Phase 2: Negotiation of the IPSEC SAs**
  - Single type: "Quick Mode" (3 messages)

Phase 1 overview



- HDR = header ISAKMP,
- HDR* = Encrypted payload

- SA = Security Association payload
- KE = Key Exchange payload (Diffie-Hellman)
- Ni, Nr = Nonce payload
- IDii, Idir = Identification payload
- HASH_I, HASH_R = Hash payload



How to perform the computations?

- We need 3 keys:
  - SKEYID_d – Master key to derive the rest
  - SKEYID_a – For authentications
  - SKEYID_e – For confidentiality
- There are 3 ways to authenticate:
  - By means of digital signatures
  - By means of asymmetric encryption
  - By means of shared secret

How to perform the computations?

- First we compute an initial SKEYID
  - For signatures:
    - ✓ SKEYID = prf(Ni_b | Nr_b, g^xy)
  - For asymmetric encryption:
    - ✓ SKEYID = prf(hash(Ni_b | Nr_b), CKY-I | CKY-R)
      - ➢ CKY are the cookies of the ISAKMP header
  - For shared secret:
    - ✓ SKEYID = prf(pre-shared-key, Ni_b | Nr_b)
- With *prf* (pseudo-random function) a MAC function (Message authentication code, a short piece of information used to authenticate a message)
  - By default HMAC (*keyed-Hash Message Authentication Code*) is used with the hash algorithm negotiated

How to perform the computations?

- From SKEYID:
  - SKEYID_d = prf(SKEYID, g^xy | CKY-I | CKY-R | 0)
  - SKEYID_a = prf(SKEYID, SKEYID_d | g^xy | CKY-I | CKY-R | 1)
  - SKEYID_e = prf(SKEYID, SKEYID_a | g^xy | CKY-I | CKY-R | 2)
- For the authentication hash:
  - HASH_I = prf(SKEYID, g^xi | g^xr | CKY-I | CKY-R | SAi_b | IDii_b)
  - HASH_R = prf(SKEYID, g^xr | g^xi | CKY-R | CKY-I | SAi_b | IDir_b)

Authentication with certificates

```
 Initiator                                  Responder
-----------                                -----------
 HDR, SA                          -->
                                  <--      HDR, SA
 HDR, KE, Ni                      -->
                                  <--      HDR, KE, Nr
 HDR*, IDii, [ CERT, ] SIG_I      -->
                                  <--      HDR*, IDir, [ CERT, ] SIG_R
```

```
 Initiator                                   Responder
-----------                                 -----------
 HDR, SA, KE, Ni, IDii            -->
                                  <--      HDR, SA, KE, Nr, IDir,
                                                [ CERT, ] SIG_R
 HDR, [ CERT, ] SIG_I             -->
```

Authentication with asymmetric encryption

```
 Initiator                                  Responder
-----------                                -----------
 HDR, SA                          -->
                                  <--       HDR, SA
 HDR, KE, [ HASH(1), ]
   <IDii_b>PubKey_r,
     <Ni_b>PubKey_r               -->
                                            HDR, KE, <IDir_b>PubKey_i,
                                  <--                  <Nr_b>PubKey_i
 HDR*, HASH_I                     -->
                                  <--       HDR*, HASH_R
```

- **HASH (1) → of the certificate of the receiver**

```
 Initiator                                   Responder
-----------                                 -----------
 HDR, SA, [ HASH(1),] KE,
   <IDii_b>Pubkey_r,
    <Ni_b>Pubkey_r                -->
                                             HDR, SA, KE, <IDir_b>PubKey_i,
                                  <--              <Nr_b>PubKey_i, HASH_R
 HDR, HASH_I                      -->
```

Authentication with shared secret

```
 Initiator                              Responder
----------                             -----------
 HDR, SA                  -->
                          <--     HDR, SA
 HDR, KE, Ni              -->
                          <--     HDR, KE, Nr
 HDR*, IDii, HASH_I       -->
                          <--     HDR*, IDir, HASH_R
```

```
 Initiator                               Responder
----------                              -----------
 HDR, SA, KE, Ni, IDii    -->
                          <--     HDR, SA, KE, Nr, IDir, HASH_R
 HDR, HASH_I              -->
```

Problems of aggresive mode

- Identities are sent as plaintext
- If we use shared secret authentication, brute-force attacks to recover the secret are possible
  - We know HASH_I and HASH_R that:
    - ✓ HASH_I = prf(SKEYID, g^xi | g^xr | CKY-I | CKY-R | SAi_b | IDii_b)
    - ✓ SKEYID = prf(pre-shared-key, Ni_b | Nr_b)

Phase 2

- Uses the IKE SA from phase 1 to ensure security
  - IKE SA provides confidentiality, authentication, integrity
- Objective: Establish an IPSEC SA
- Three messages in phase 2:
  - Message 1: Proposal of parameters for negotiation and material to generate session keys, authenticated with SKEYID_a
  - Message 2: Election of parameters and HMAC signature of the first message, authenticated with SKEYID_a
  - Message 3: Values to generate session keys, authenticated with SKEYID_a

Phase 2

```
 Initiator                                  Responder
-----------                                -----------
 HDR*, HASH(1), SA, Ni
   [, KE ] [, IDci, IDcr ]  -->
                            <--     HDR*, HASH(2), SA, Nr
                                          [, KE ] [, IDci, IDcr ]
 HDR*, HASH(3)              -->
```

- HASH(1) = prf(SKEYID_a, M-ID | SA | Ni [| KE] [| IDci | IDcr])
- HASH(2) = prf(SKEYID_a, M-ID | Ni_b | SA | Nr [| KE] [| IDci | IDcr])
- HASH(3) = prf(SKEYID_a, 0 | M-ID | Ni_b | Nr_b)

- M-ID of the header ISAKMP

Phase 2: Explanations

- **IPSEC SA key (session key) built from:**
  - The IKE SA key
  - The exchanged random data
- **Result of phase 2:**
  - Two unidirectional IPSEC SA, each with an appropriate SPI
- **PFS (Perfect Forward Secrecy):**
  - Can be achieved by forcing a renegotiation of keys
    - ✓ E.g., using Diffie-Hellman
- **On expiration of the IPSEC SA**
  - Negotiation of a new one using the IKE SA

Phase 2 – key generation

- **Without PFS:**
  - **KEYMAT = prf(SKEYID_d, protocol | SPI | Ni_b | Nr_b).**
    - ✓ **Protocol and SPI taken from the IKE association**
- **With PFS:**
  - **KEYMAT = prf(SKEYID_d, g^xy | protocol | SPI | Ni_b | Nr_b)**

IKEv2

- **Fewer messages**
  - 4 messages establish both the IKE association and IPSEC
  - The problems of aggressive mode of IKEv1 are avoided
- **New authentication mechanisms accepted**
  - Base in EAP (Extensible Authentication Protocol)

IKEv2 (without EAP)

Initiator **Listener**

HDR, SAi1, KEi, Ni -->

<-- HDR, SAr1, KEr, Nr, [CERTREQ]

HDR, SK {IDi, [CERT,] [CERTREQ,] [IDr,] AUTH, SAi2, TSi, TSr} -->

<-- HDR, SK {IDr, [CERT,] AUTH, SAr2, TSi, TSr}

IKEv2 → additional SAs

Initiator

Listener

HDR, SK {[N], SA, Ni,
[KEi], [TSi, TSr]} -->

<-- HDR, SK {SA, Nr,
[KEr], [TSi, TSr]}