# Lecture 5.1:

# IPsec Basics

**Recommended reading: Stallings, Chapter 16**
**(RFCs are perhaps a bit too complex and extensive for our class – use as extra reading material)**

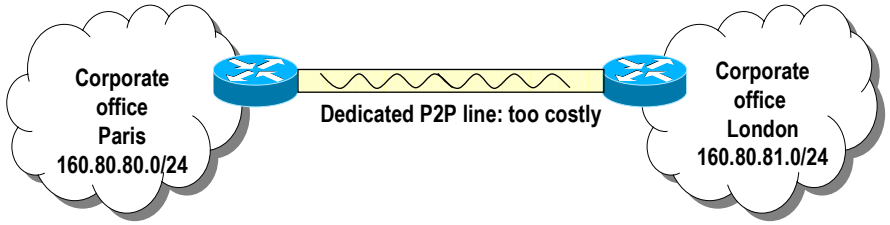
Giuseppe Bianchi

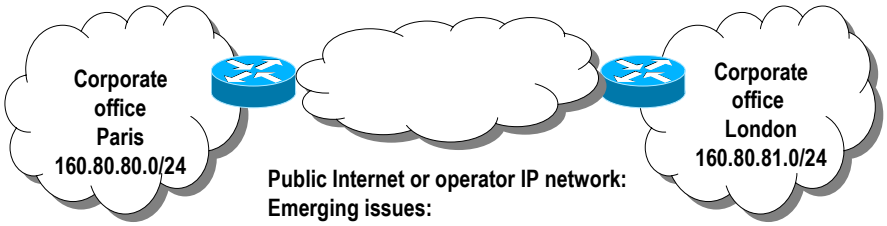
# A parenthesis
# VPNs: what they are

**Perhaps out of scope, here, as VPN and IPsec are NOT the same – more later**

Giuseppe Bianchi

# Virtual Private Networks: why?

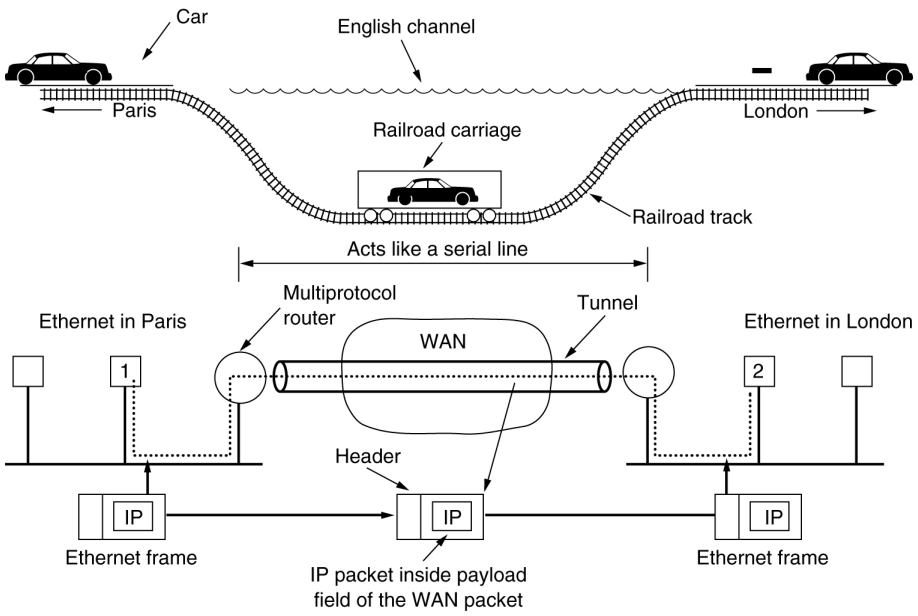


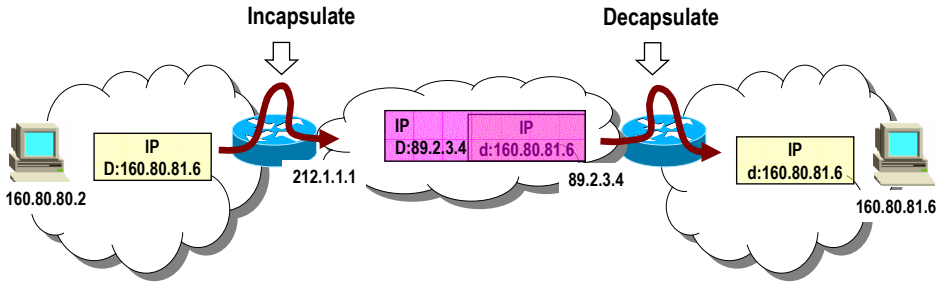

Public Internet or operator IP network:
Emerging issues:
- How to manage private address space across distributed sites?
- How to protect data in transit (especially if public Internet)?

Giuseppe Bianchi

# Virtual Networks → tunnels



# Virtual Networks over IP




➔ **IP in IP tunnels**

⇨ Not the most effective approach!

➔ **MPLS tunnels by far more performance effective**

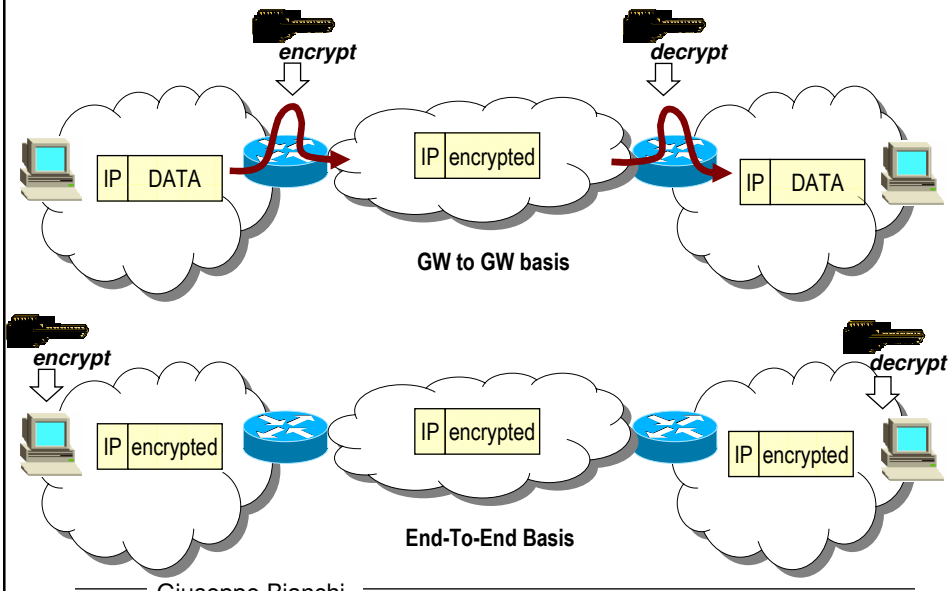
⇨ Typical VPN offer from today operators

» MPLS tunnels alone = VPN without the "P" ☺

» However customer may trust operator (the only one with "hands on" the net)

Giuseppe Bianchi

# Private Networks → encryption




Giuseppe Bianchi

# Virtual + Private Networks

➔ **VPN =**

⇨ Virtual Networks (tunnels)

+

⇨ Private Networks (authentication,encryption)

➔ **IPsec: a POSSIBLE tool for building VPN**

⇨ But IPsec and VPNs are NOT synonymous

➔ as some beginner might think

➔ IPsec VPNs not viable when non-IP traffic must be transported!

⇨ IPsec: not only tunnels; also e2e encrypted/authenticated transport

➔ **VPN alternatives:**

➔ Layer 2: GRE/PPTP, L2TP

➔ Layer 3 (actually 3-): MPLS

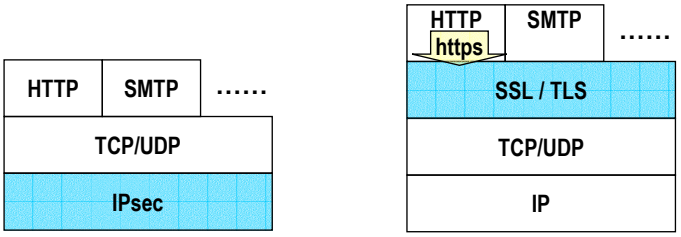
➔ Layer 4 (actually between 4 and 7): SSL tunnels

➔ Layer 7: SSH tunnels



# IPsec Components

# IPsec: layered view




Network layer security

Transport layer security

➔ **IPsec operates with & within IP, at layer 3**
- ⇨ IPsec & unprotected IP packets do coexist (of course)

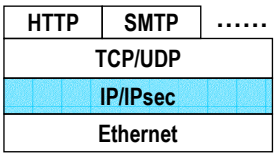
➔ **Applications/terminals unaware of IPsec**
- ⇨ IPsec protects all protocols that rely on IP

➔ **Protection on a per-host (IP) basis**
- ⇨ cannot protect SPECIFIC users/applications

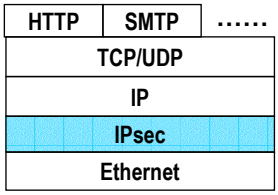


# IPsec implementation approaches

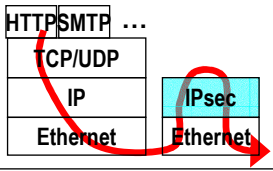
➔ **Inside the native IP code**
- ⇨ Best approach
- ⇨ But hard to deploy as requires to access and modify IP source code




➔ **Bump in the Stack (BITS)**
- ⇨ Between native IP code and device driver
- ⇨ Deployed in legacy systems




➔ **Bump in the Wire (BITW)**
- ⇨ Implemented in dedicated hardware
- ⇨ External security processor, acts as gw



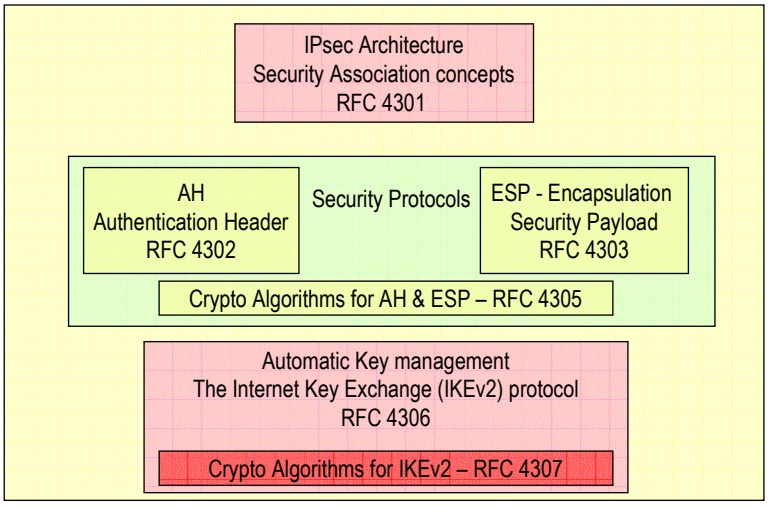
# IPsec standardization History

➔ **Three major "series" of RFCs**

- ⇨ Serie 1: RFC 1825-1827 (august 1995)
  - ➔ IPsec concepts first drafted
- ⇨ Serie 2: RFCs 2401-2412 (november 1998)
  - ➔ Significant revision of ALL the IPsec architecture
  - ➔ Describes IPsec as we know it today
- ⇨ Serie 3: RFC 4301-4307 (december 2005)
  - ➔ Born after long discussion in WG (almost 5 years)
  - ➔ basically touches/extends all the IPsec architecture
  - ➔ Most important: major revision of IKE (Internet Key Exchange protocol)
    - » IKEv2 now simplifies and glues several protocols (ISAKMP, IKE, Oakley) into one

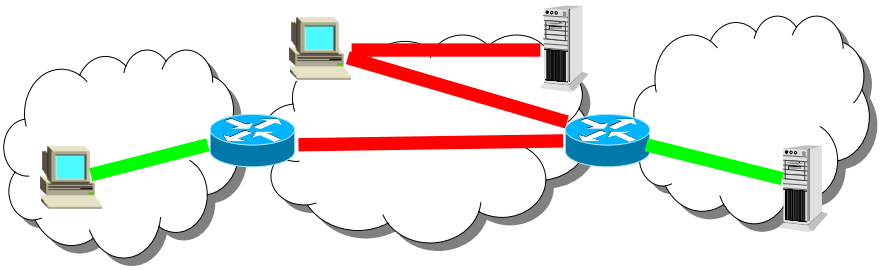


# IPsec RFCs



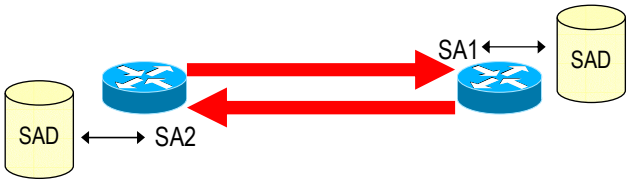
# Security Association

- **Fundamental concept in IPsec**
- **May involve:**
  - Host to host
  - Host to intermediate router (security gateways)
  - Security gateway to security gateway
- **Defines the boundaries for IP packets authentication/encryption**
  - A "connection" with security services active





# SA: monodirectional!




- **SPI = Security Parameters Index**
  - The (somewhat) unique "name" of an SA
- **Destination Address based**
  - Security Association managed at the receiver side
- **SAD = Security Associations Database**
  - SPI = search key (at least)
  - Stores set of security services per each SA, and related parameters
    - E.g. which encryption algorithm; shared key for encryption, SA lifetime, Sequence number counter, etc

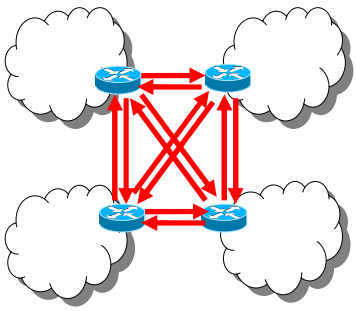
# SPI and SAD lookup

- **Security Parameters Index = 32 bit index**
- **Identifies a Security Association**
  - Uniquely, but in conjunction with destination address and possibly source address
    - Note: in multicast destination address differs from receiver address!!
  - Used to lookup into Security Association Database and retrieve security parameters
    - E.g. AH: Authentication algorithm, shared key,
  - Lookup rule: "longest match"–like
    - First look up SPI, Dest Addr, Src Addr
    - Then look up SPI, Dest Addr
    - Then Look up SPI
      - » Implementations may also specify AH/ESP during lookup



# Security Association and Key management

- **Manual**
  - Manually configure each SA and related crypto keys
    - static, symmetric
  - Typical in small-scale VPNs
    - Few security gateways, e.g. one per site
    - Meshed SA connections
- **Automatic**
  - SA management through IKEv2
    - In the past, through the combined operation of several protocols
      - » IKE+ISAKMP+others
  - On-demand SA creation
  - Session-oriented keying/rekeying

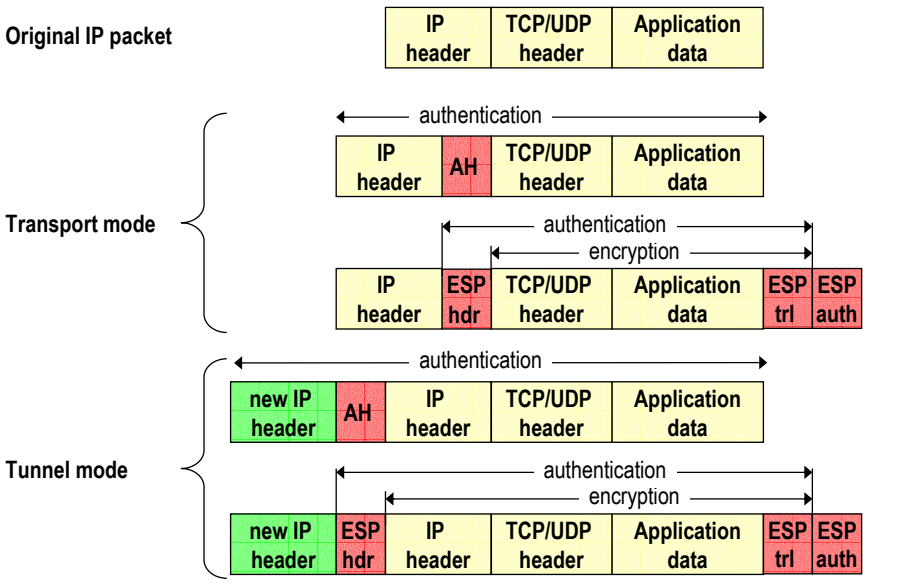


# IPsec Security protocols

- **AH (Authentication Header)**
  - Authentication (whole packet) only
- **ESP (Encapsulated Security Payload)**
  - Encryption, Authentication, both
    - Unlike AH, payload only authentication, no header
    - Not an issue, when tunnel mode used
- **ESP in most cases is the only one needed**
  - Mandatorily supported in any IPsec implementation, unlike AH
    - RFC 4301: AH downgraded: from MUST to MAY (be supported)
      - » Quoting from RFC 4301: *"Experience has shown that there are very few contexts in which ESP cannot provide the requisite security services. Note that ESP can be used to provide only integrity, without confidentiality, making it comparable to AH in most contexts."*
  - AH and ESP may be combined together, if needed (rarely)
- **Transport mode and tunnel mode for both**

Giuseppe Bianchi

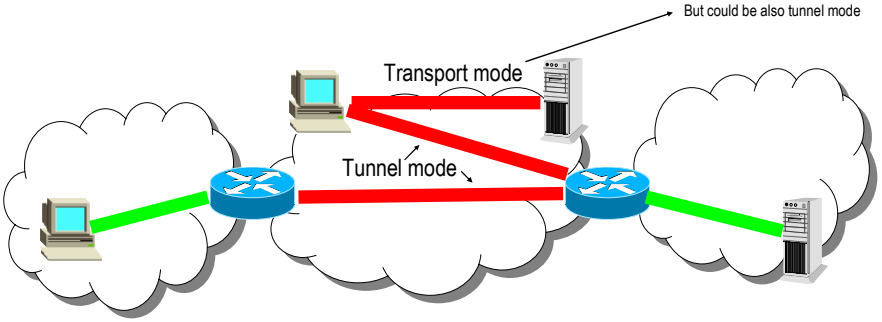
# Transport vs Tunnel – AH and ESP




Giuseppe Bianchi

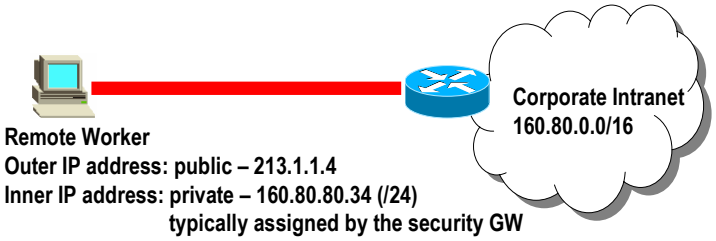
# When transport? When Tunnel?

- **Transport mode: end-to-end**
- **Security Gateways can use transport mode only for connections originating/terminating there**
  - Not when they are intermediary between host and server!
  - Tunnel mode used in almost all the cases





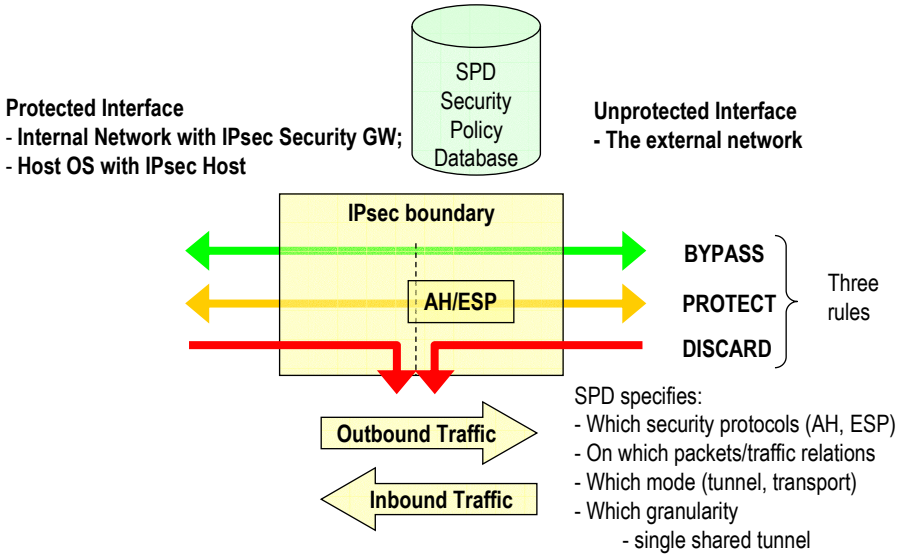

# A note on host-to-gw tunnels




**Remote Worker**
**Outer IP address: public – 213.1.1.4**
**Inner IP address: private – 160.80.80.34 (/24)**
**typically assigned by the security GW**

- **Using a private IP address inside the tunnel:**
  - Allows to access to all services provided in the intranet, exactly like in the case the worker is connected inside the corporate
  - Allows to be protected by the corporate firewall (all traffic destined to 160.80.80.34 MUST be first routed to the corporate subnet and then routed to the end user in a protected fashion

# IPsec protection & access control




SPD specifies:
- Which security protocols (AH, ESP)
- On which packets/traffic relations
- Which mode (tunnel, transport)
- Which granularity
  - single shared tunnel
  - different tunnels per TCP conn



# Lecture 5.2:

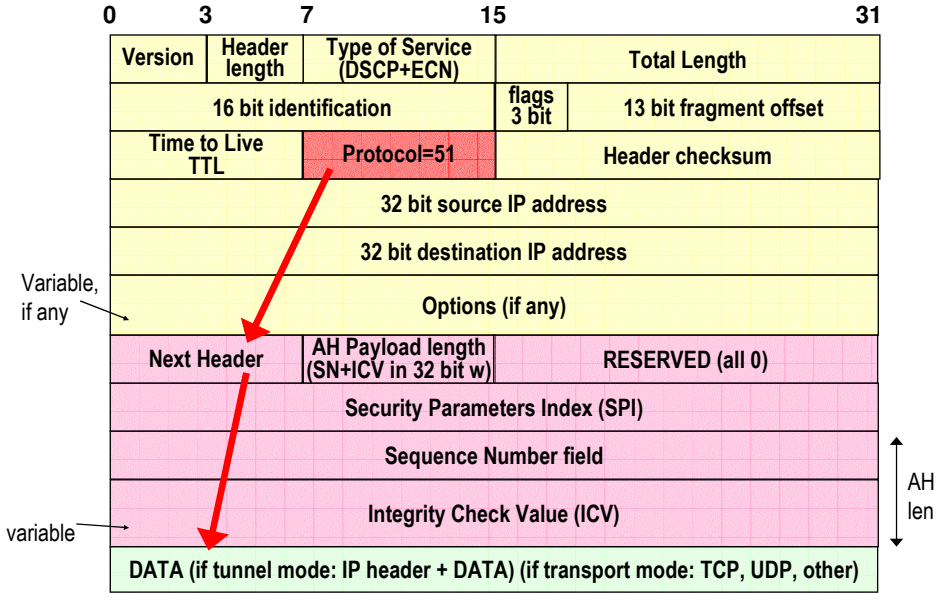
# IPsec Security Protocols
# AH/ESP
**(IPv4 only)**

# Services provided

- **Authentication Header**
  - Integrity and data origin authentication
    - Authentication covers both payload and parts of IP header that do not modify in transfer
  - Protection against replays
    - Optional, through extended sequence numbers
- **Encapsulated Security Payload**
  - Same services as AH
    - Though authentication limited to IP payload
  - Confidentiality through encryption
  - Traffic flow confidentiality
    - Improved privacy against eavesdropping
    - Through padding and dummy traffic generation

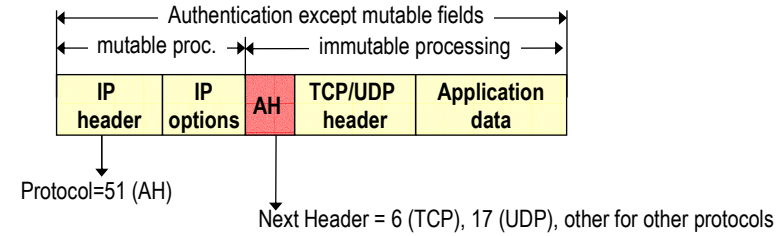


# Authentication Header

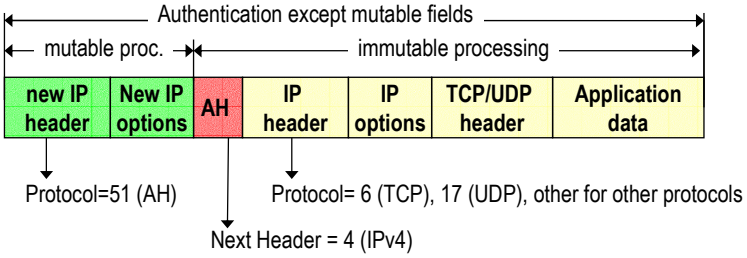


# Transport mode, tunnel mode

**Transport mode:**

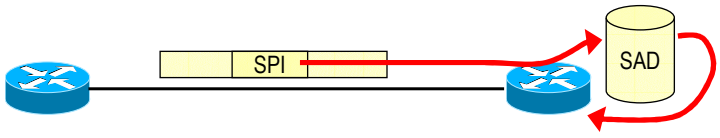



**Tunnel mode:**






# Security Parameters Index




- **➔32 bit index**
- **➔Used to lookup the SAD at destination**
  - ⇨Lookup also uses
    - ➔destination address
    - ➔source address
    - ➔security protocol (AH/ESP)
- **➔Retrieves algorithms and parameters that allow to process received packet**

# Integrity Check computation

➔ **Only on immutable fields in the IP header**

⇨ Or mutable but predictable

➔ e.g. destination address with strict/loose source routing option

➔ **Mutable fields set to 0 during MAC computation**

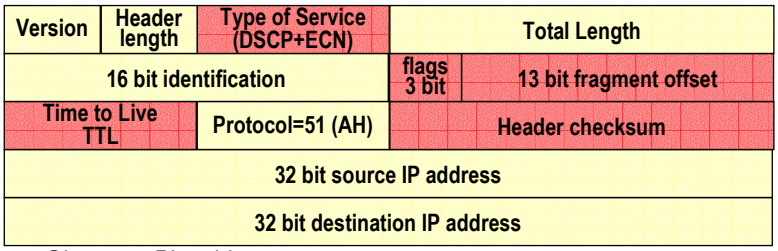
⇨ Highlighted in red in next figure

➔ Note: AH apply before fragmentation, and checked after reassembly

➔ **Options classified as either mutable or not**

➔ Mutable options: details in appendix A RFC 4302

➔ mutable options = all zeroed






# auth algorithm and ICV

➔ **ALL implementations must mandatorily support:**

⇨ HMAC-SHA-1-96

➔ Standard HMAC-SHA-1 ➔ 20 bytes ➔ 160 bits

➔ Use only first 96 bits

➔ **A modern (>2005) implementation MAY support:**

⇨ HMAC-MD5-96

➔ Standard HMAC-MD5 ➔ 16 bytes ➔ 128 bits

➔ Use only first 96 bits

➔ MAY because MD5 now considered weak

➔ **A modern (>2005) implementation SHOULD support:**

⇨ AES-XCBC-MAC-96

➔ see description in RFC 3566

➔ **ICV must be multiple of 32 bits**

⇨ Padding necessary if MAC not multiple of 32

➔ Arbitrary padding chosed at sender

## Why sequence number?

- **IP header DOES NOT contain a sequence number!**
  - Hence replay of an authenticated IP packet is possible
    - And may alter in an unpredictable manner the overlaying service (e.g. ICMP replies can be dangerous ☺)
- **Sequence number: 32 bit counter**
  - Initialized to 0 when the Security Association is established
  - Increments of 1 per each transmitted packet
    - First transmitted packet: SN=1
  - Maximum value $2^{32}$-1, afterwards Security Association must be terminated
    - No counter cycling allowed when anti-replay service active
    - Anti-replay: optional (but default = on)
      - » Anti-replay typically OFF when manual (static) keys configured

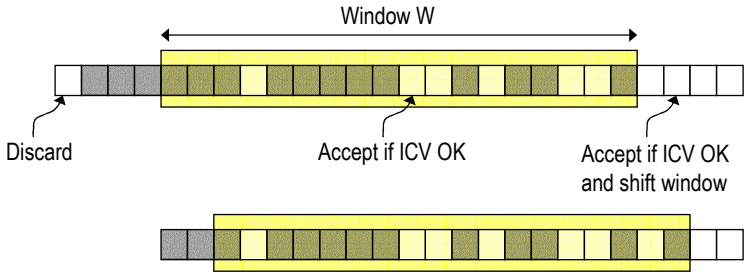


## Extended Sequence Number

- **$2^{32}$ ~ 4.3 billion**
  - A lot, but not REALLY al lot!
    - Packet size = 1500 (1460 bytes payload)
    - $2^{32}$ x 1460 bytes = 6270 GB
    - About 14 h transmission of a 1 gbps link
- **Extended Sequence Number:**
  - 64 bits - this should be enough, now ☺
  - Transmit only low order 32 bits
  - But use high order 32 bits in ICV computation!

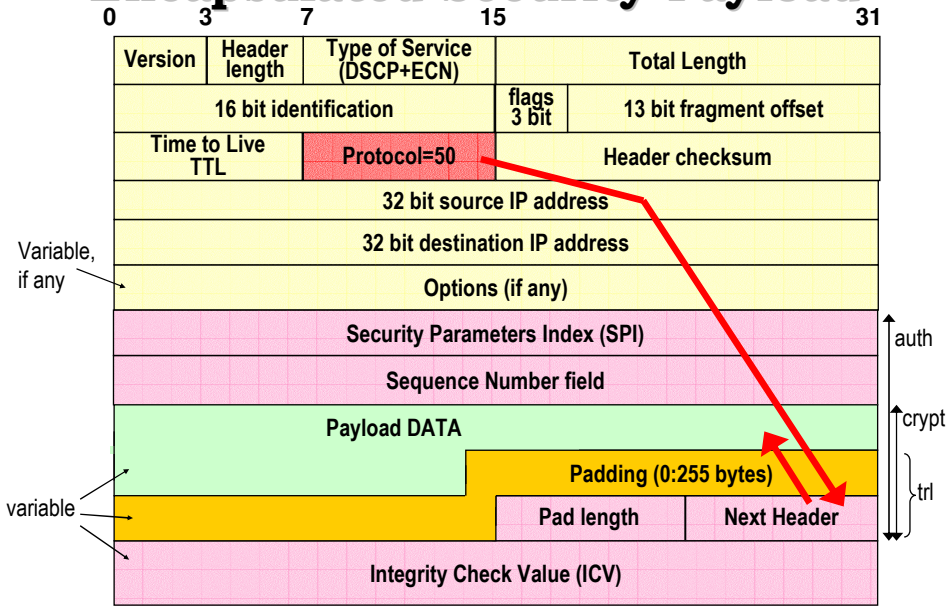
# Anti-replay

- **Sliding Window W**
  - Size locally decided at receiver
    - Minimum = 32; default = 64; higher values recommended for high speed links
    - eventually very large: maximum $2^{31}$-1 with SN and $2^{32}$-1 with ESN
  - Window right margin = highest NS packet received
- **Duplicates discarded**
- **Packets out of left window edge discarded**
- **Packets greater than right window margin make W shift**



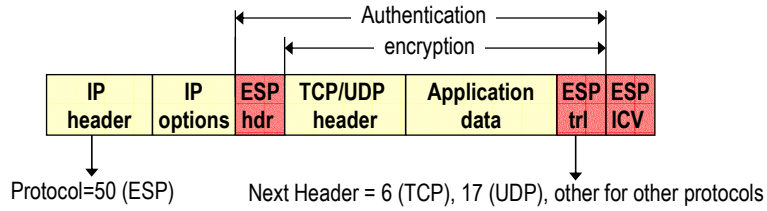



# Encapsulated Security Payload

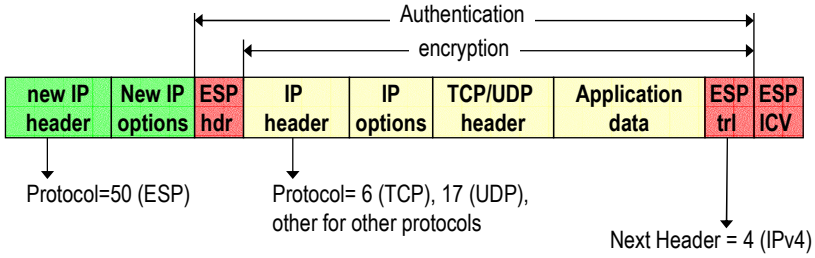


# Transport mode, tunnel mode

**Transport mode:**




**Tunnel mode:**






# ESP Header & Trailer

- **ESP header**
  - 8 bytes, SPI + SN
    - Same as AH (including ESN and optional anti-replay)
    - Plain text
- **ESP Trailer**
  - Padding, variable size + pad length + next header
    - Pad length = 1 byte = 0:255
    - Next Header = 1 byte = type of data payload
      - » Next Header = 4 → Tunnel mode (IP header inside)
      - » Next Header = 59 → Dummy packet!!
  - Padding: for two reasons
    - → Encryption algorithm may require plaintext to be a multiple of some number of bytes
      - » E.g. block size of a block cipher
    - Resulting ciphertext must terminate on a 4-byte boundary

## Encryption & authentication algos

- **Authentication algorithms:**
  - Same as AH
- **Encryption algorithms**
  - No algorithm strictly mandatory
    - Though 3DES-CBC labeled as MUST- ☺
  - Recommended (>2005)
    - AES-CBC (128 bit key), RFC 3602
    - AES-CTR, RFC 3686
- **Combined mode**
  - Novel approach: both authentication and encryption in a single shot
    - AES-CCM specified in RFC 3686 and used also in 802.11i
    - Not suggested here, but mentioned as "appealing in future"
  - No trailer ICV used

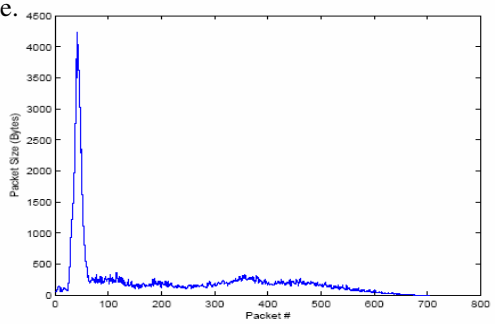


## Encryption & IP unreliability

- **A problem: some good encryption algorithms need to maintain synchronization**
  - E.g. Cipher Block Chaining, etc
- **However IP packets may arrive out of order or may be lost!**
- **Consequence: for some encryption algorithms, packets must carry data needed to resync the deciphering**
  - If such data (per-packet Initialization Vector) necessary, IV prepended to DATA payload
    - Not an ADDITIONAL ESP header! i.e. it is "invisible" to ESP

# Traffic Flow Confidentiality Service

- **Encryption does not guarantee unlinkability!**
- **Linkability via statistical traffic pattern analysis**
- **Traffic source (e.g. web-site) fingerprinting**
  - E.g. sample size profile for www.amazon.com
    - Details in "Privacy Vulnerabilities in Encrypted HTTP Streams, by Bissias, Liberatore, Levine.
  - Many other traffic/protocol fingerprinting approaches (literature + pre-prints)






# ESP Traffic Flow Confidentiality

- **ESP TFC: Two counter-measures against traffic analysis attacks**
  - Ability to alterate packet size
    - Supplementary padding facility added to data payload
      - » Easy to manage in tunnel mode, as inner packet size is known in the IP/TCP header
    - Native (up to) 255 bytes padding insufficient
      - » And why messing up by mixing TCF padding with that necessary for encryption?

| IV (variable) | DATA PAYLOAD | TFC PADDING |
|---|---|---|

  - Ability to generate "dummy" packets
    - Example: one can transform a VBR traffic into CBR
      - » Heavy on the network load, though: reduces statistical multiplexing effectiveness

# Lecture 5.3:

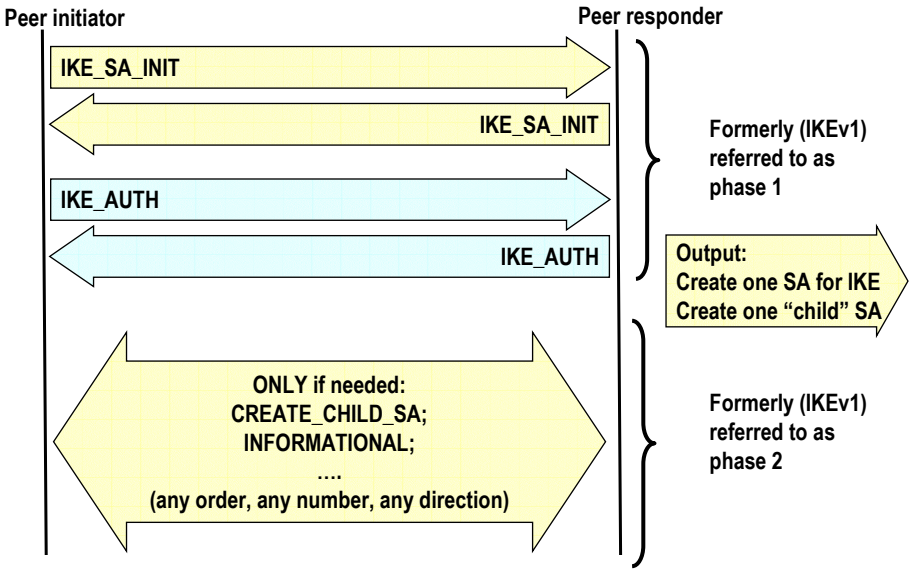
# IKEv2



## Rationale for IKE

- **shared state must be maintained between source and sink**
  - Which security services (AH, ESP)
  - Which Crypto algorithms
  - Which crypto keys
- **Manual maintenance not scalable**
  - Partially OK only for small scale VPNs
  - In any case, weak approach
    - Infinite lifetime SA → no rekeying!
- **IKE = Internet Key Exchange protocol**
  - Goal: dynamically establish and maintain SA
  - IKE now (december 2005, RFC 4306) in version 2
    - Replaces protocols specified in RFCs 2407, 2408, 2409 (IKE, ISAKMP, DOI)
    - IKEv2 quite different (and much cleaner!!) than former specifications

# IKE phases at a glance






# IKE SA and CHILD SA

➔**IKE SA:**

⇨Security association to exchange IKE messages

➔**CHILD SA**

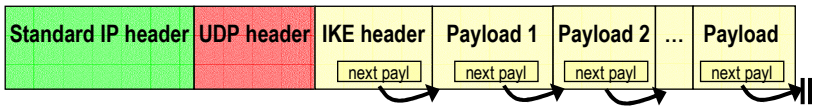
⇨Security association to exchange data messages

➔Making use of AH or ESP

⇨Many CHILD SA may be set up between two peers

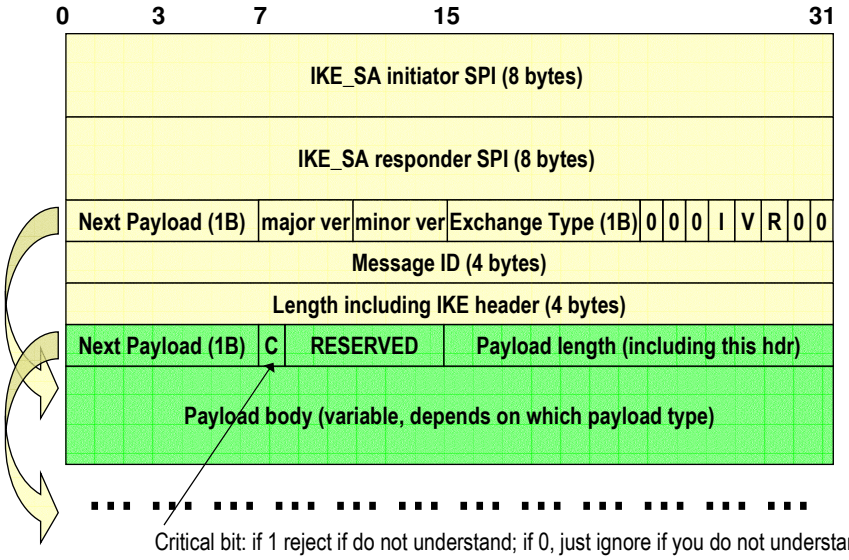
# IKE message format

- **UDP encapsulated**
  - Ports 500 and/or 4500
  - Reliable delivery managed by IKE through retransmission
    - A new important feature of IKEv2!
    - No details here… see RFC 4306 for a thorough discussion
- **IKE header first**
- **Followed by one or more IKE payloads**
  - Brilliant idea (for a perhaps stretched analogy think to AVP concept; a more appropriate analogy is with the next header concept of IPv6)
    - flexible approach: new payloads added at later stages



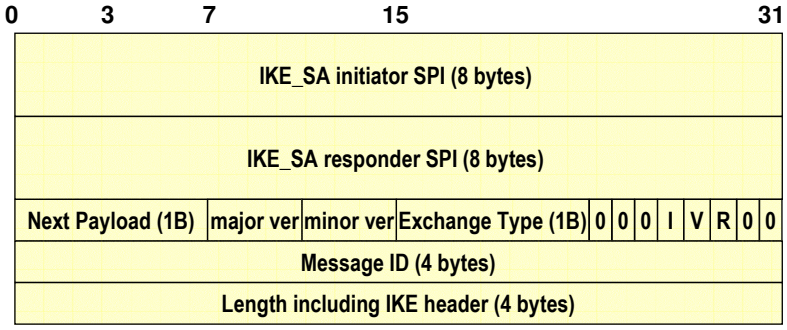



# IKE hdr, generic payload hdr




Critical bit: if 1 reject if do not understand; if 0, just ignore if you do not understand

# IKE header (explanation)

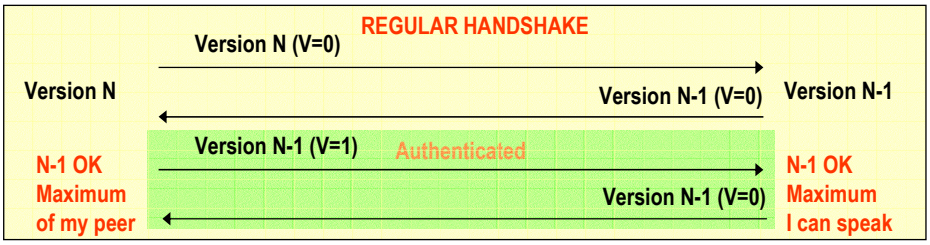


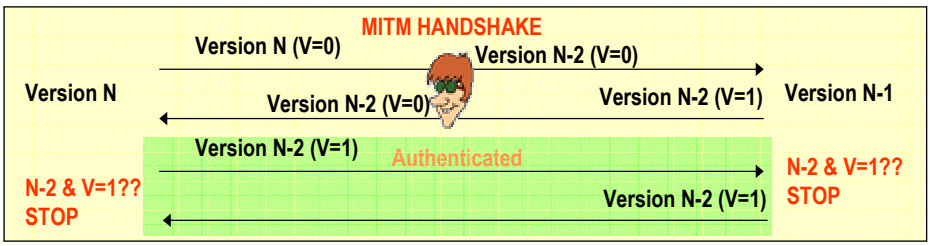

- **Initiator SPI, responder SPI:**
  - SA id for the IKE protocol. Initially, responder SPI set to 0; responder fills it in
    - don't confuse with the SPI for the CHILD AH/ESP SA
- **Versions: major = 2, minor = 0**
  - Bit V=1 says that implementation can "speak" higher version than what written in the hdr
- **Exchange type: one of 4 messages: IKE_SA_INIT, IKE_AUTH, CREATE_CHILD_SA, INFORMATIONAL**
  - Direction: bit R (response=1, request=0); bit I (original initiator=1, original responder=0; needed to properly match SPIs
- **Message ID: Sequence number counter, to match requests with responses, to manage retransmission, to combat replay attacks**

Giuseppe Bianchi

# Version bit
# Protection against version rollback

**Different approach than SSL: based on V (version) flag**




Too bad v1 does NOT include this flag!!! (Hence rollback to IKEv1 is not protected)

Giuseppe Bianchi

# IKE_SA_INIT phase

➔**Clear Text Request followed by Clear Text response**

⇨Negotiates security parameters for the IKE_SA

⇨sends nonces

⇨sends Diffie-Hellman values

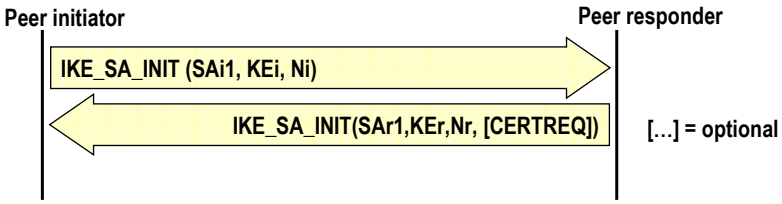
➔**Output:**

⇨Generate a session key from which all the other encryption and authentication keys will be generated

➔Called SKEYSEED



# Exchanged Info




➔ **SAi1/SAr1 (Security Association) payloads:**

⇨ Initiator sends list of crypto algorithms supported

⇨ SAr1 choose one algo per each function (encryption, authentication, pseudo-random function)

➔ **KEi/KEr (Key Exchange) payloads:**

⇨ Diffie-Hellman Ephemeral values

➔ **Ni/Nr payloads:**

⇨ Random values used in the crypto parameters derivation, to protect replay

⇨ At least 16 bytes, up to 256 bytes

➔ **CERTREQ (optional payload)**

⇨ Request of a certificate

# Key generation

➔ **SKEYSEED:**

- ⇨ Contruction based upon a negotiated prf
  - → Note the difference with TLS (where the PRF is explicitly given)
- ⇨ SKEYSEED = prf(Ni, Nr, g^ir)
  - → g^ir = Diffie-Hellman shared key

➔ **7 keys generated:**

- → Through prf+ extended construction (similar to TLS)
- → prf+(SKEYSEED, Ni | Nr | SPIi | SPIr)
- ⇨ $SK_{ai}$ $SK_{ar}$ for authentication at initiator and responder
- ⇨ $SK_{ei}$ $SK_{er}$ for encryption at initiator and responder
- ⇨ $SK_{pi}$ $SK_{pr}$ for generating AUTH payload in SA_AUTH phase
- ⇨ $SK_{d}$ to derive new keys for the CHILD_SA

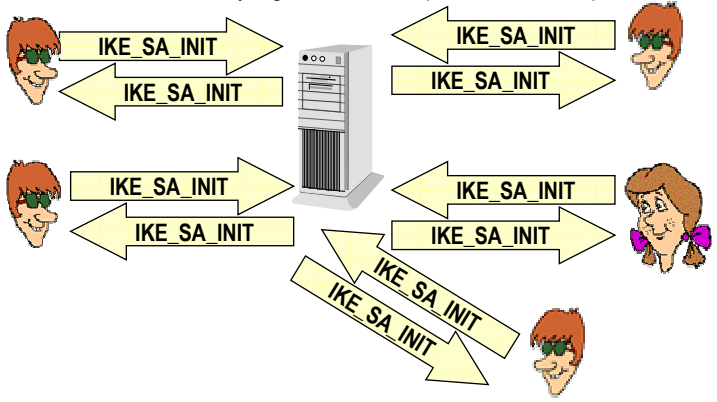


# Protection against DoS attacks

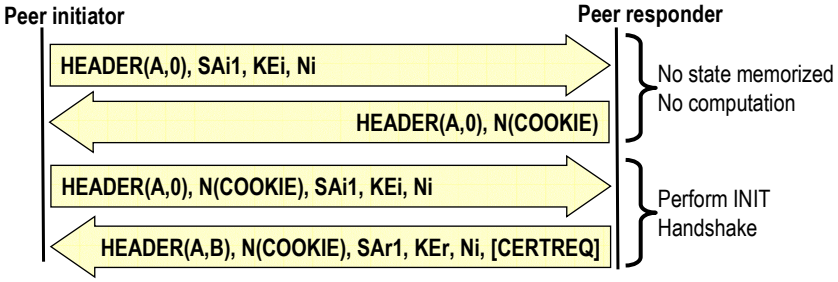
➔ **DH computation and key generation is a computational expensive process; state must be memorized**

➔ **Denial of Service Attack: spoof INIT requests (using forged IP addresses) and overload server (exhaust CPU and memory)**

⇨ When load is sufficiently high, "Normal" requests cannot be processed anymore



## Solution: cookie-based 4-way INIT handshake




➔ **Idea:**
- ⇨ Responder first replies with a cookie
- ⇨ Real initiators will reformulate the request including the cookie provided
- ⇨ Only at this point a state (SPI=B) is instantiated

➔ **Fools DoS attacks based on spoofed IPs**
- ⇨ Typical attacks!



## Is this a real solution?

➔ **Q: What if DoS attacker spoofs cookies too**
- ⇨ E.g. using random cookies?

➔ **A: cookies must be "valid", i.e. issued by the responder**

**…. but ….**

➔ **Server must recognize valid cookies**
- ⇨ Hence it must store a state for the cookies, e.g. a database
- ⇨ And hence it must use memory!!

**… is this true?**

# Idea: use stateless cookies

➔ **Use cookies which do not require any state memorization**

⇨ i.e. the validity of the cookie may be checked without any lookup at a database of issued cookies, but only looking at the request and at the cookie itself

➔ **Example:**

**Cookie = <VersionIDofSecret> | Hash(Ni | IPi | SPIi | <secret>)**

➔ **where:**

⇨ Ni, IPi, SPIi is information that is available in the IKE_SA_INIT request (nonce, IP address, SPI)

➔ Ni included to avoid that an attacker who sees only the server response may spoof the INIT+COOKIE message!

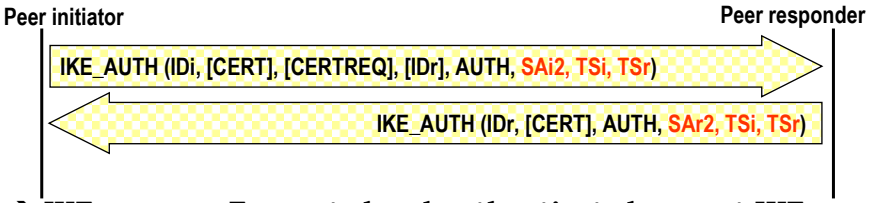
⇨ <secret> is an information available only at the responder

⇨ <secret> changes over time (hence VersionIDofSecret initial label)

➔ Refresh to avoid build up of cookie dictionaries



# IKE_AUTH phase




➔ **IKE message Encrypted and authenticated, except IKE header**

⇨ Using previously derived SKe as encryption key

⇨ Using previously derived SKa as authentication key

⇨ Using previously negotiated encryption/authentication algorithms

➔ **Sends AUTH payload**

⇨ Authenticates previous message + Identity of initiator and responder (IDi, IDr – could be IP addresses, domains, email addresses or else)

➔ Combats downgrade attacks

⇨ When certificates exchanged, authentication uses corresponding private key

## CHILD_SA generation

- **First CHILD_SA directly incorporated in AUTH messages**
  - Transmits new Security Association payloads to negotiate security parameters for the CHILD_SA
  - Transmits the Traffic Selector (TS) parameters, i.e. the IP addresses, port numbers & protocols to which the SA must be applied
    - E.g. IP addresses in the range 192.168.1.1-12
    - TS parameters included in the Security Policy Database
- **Other CHILD_SA may follow**
  - Further include new nonces



## INFORMATIONAL

- **Notification messages**
- **Configuration messages**
  - E.g. assign internal IP address to remote terminal tunneled into an IPsec SA

DETAILS OMITTED FOR LACK OF TIME ☹