Part 1:

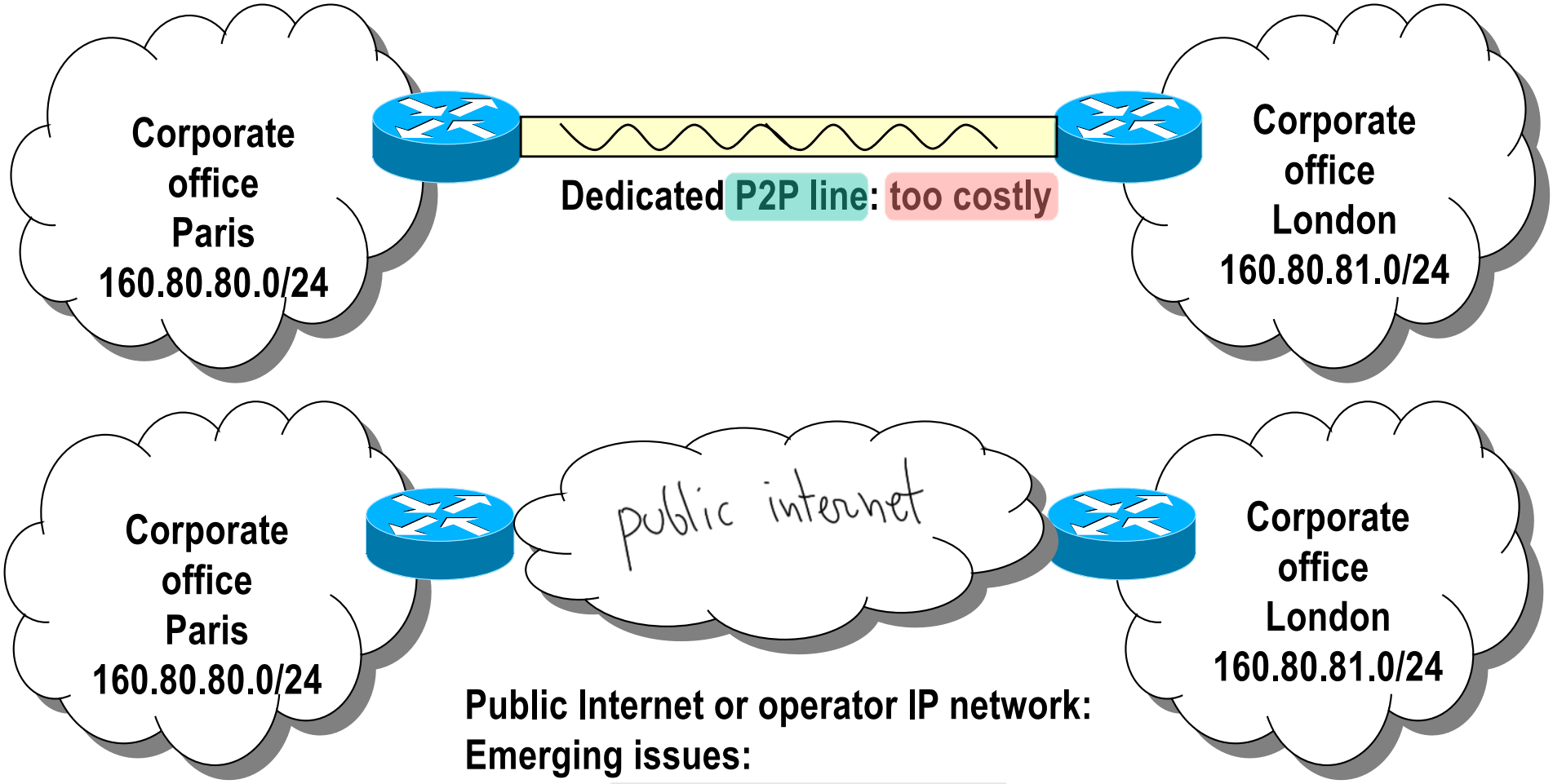
IPsec Basics

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

A parenthesis
VPNs: what they are

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

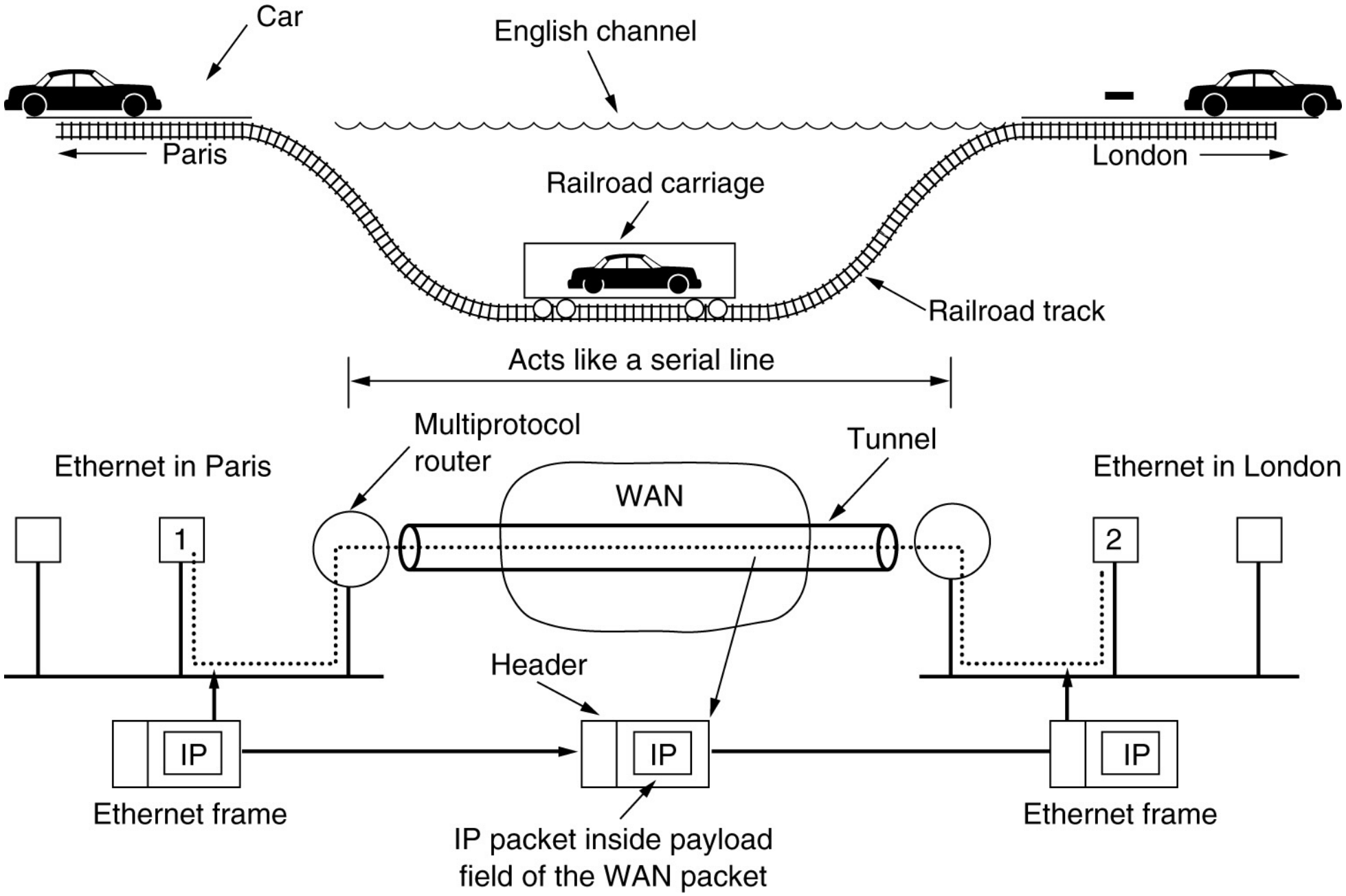
Virtual Private Networks: why?



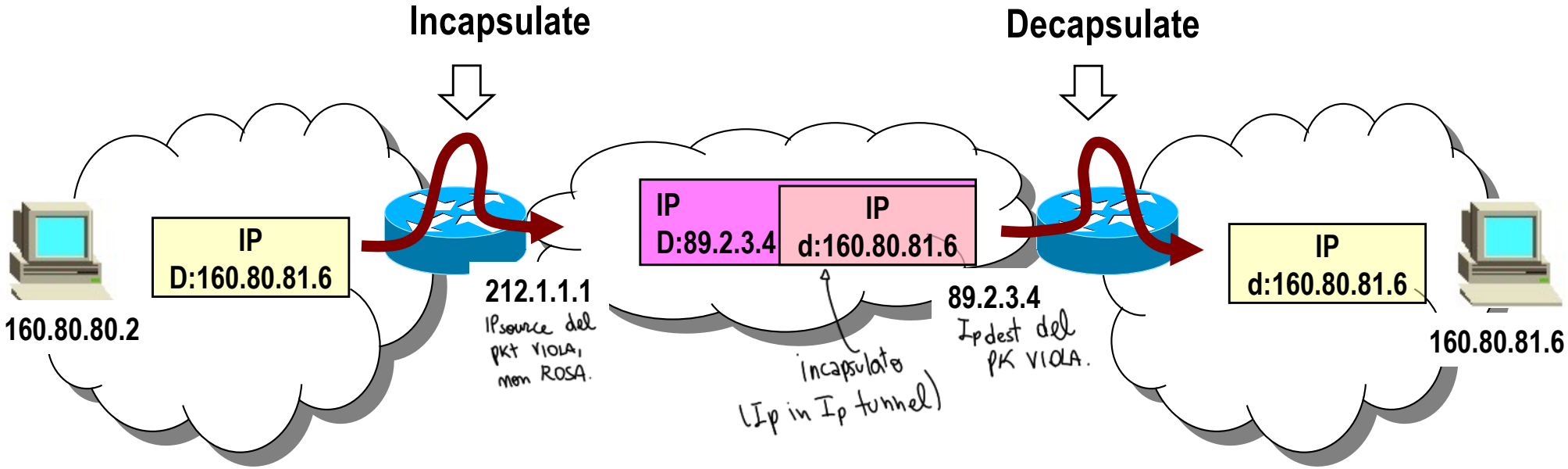
Public Internet or operator IP network:
Emerging issues:
- How to manage private address space across distributed sites?
- How to protect data in transit (especially if public Internet)?

1° elemento Fondamentale

VPN main ingredient → tunnels



VPN main ingredient → tunnels



➔ **IP in IP tunnels**

⇨ Not the most effective approach!

Multi Protocol Label Switching, NON GARANTISCE encryption, devo "fidarmi del network operator".

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

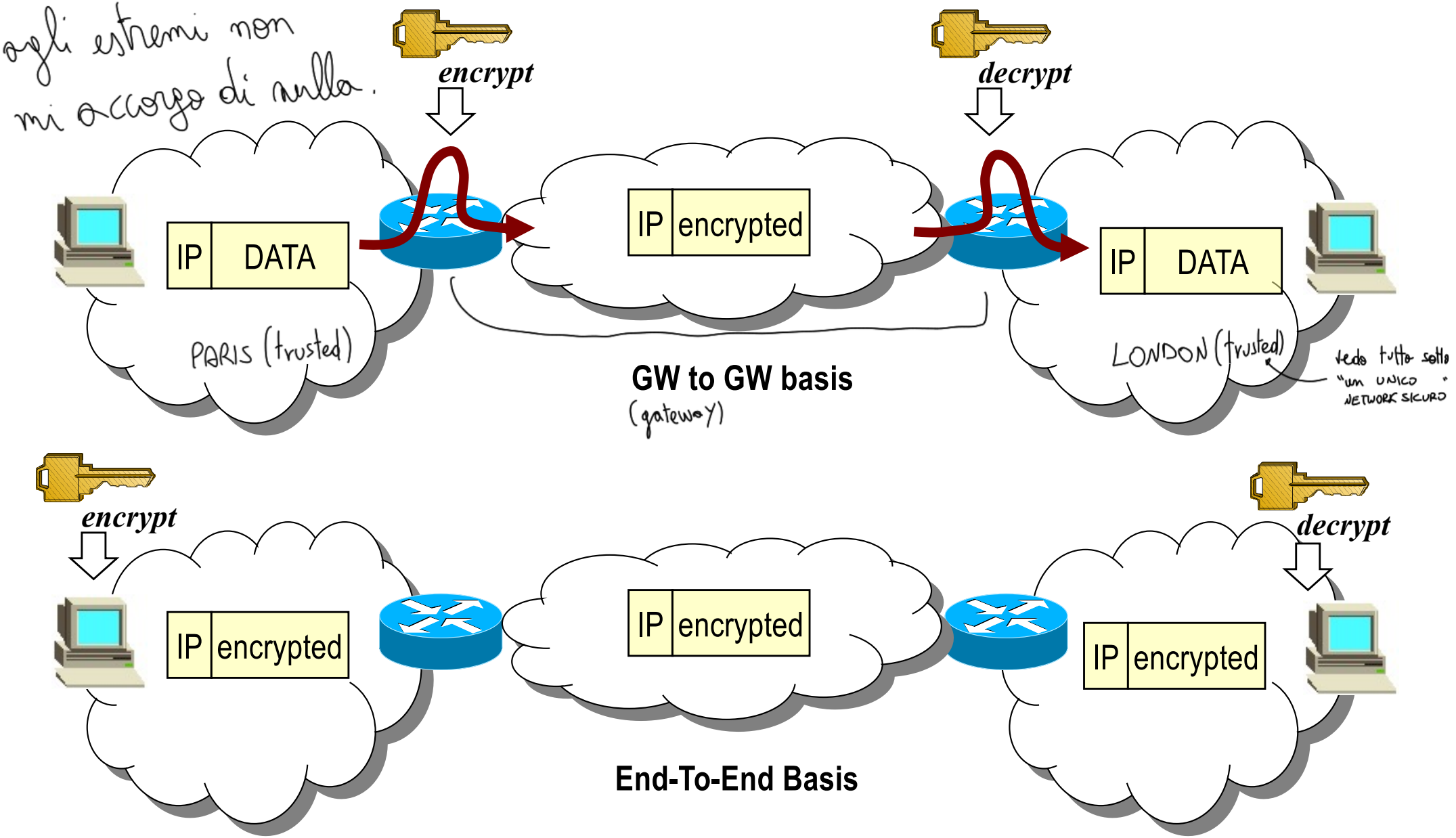
⇨ Typical VPN offer from today operators

» MPLS tunnels alone = no crypto protection

» Trust: in the Operator's infrastructure

VPN 2° ingredient → encryption

agli estremi non mi accorgo di nulla.



VPN and IPsec

- **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

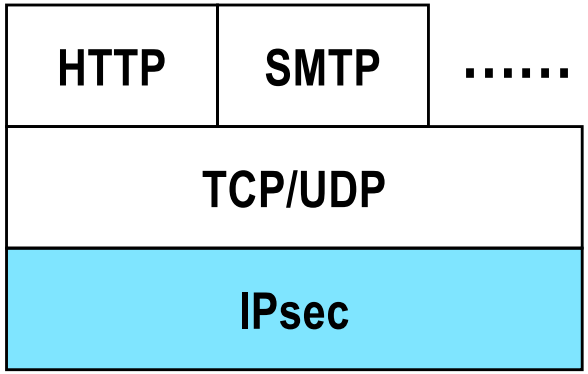
- **VPN alternatives:**
  - Layer 2: GRE/PPTP, L2TP (più recente, opera su Network)
  - Layer 3 (actually 3-): MPLS (offerto dagli operatori)
  - Layer 4 (actually between 4 and 7): (D)TLS tunnels (openVPN ...)
  - Layer 7: SSH tunnels

IPsec Components

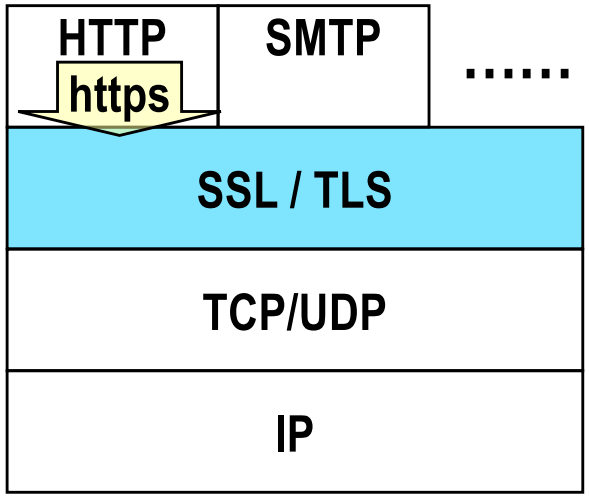
Cosa è !?

IPsec: layered view

Level 3:
tutti pkt fuori dal mio sist. op. sono PROTETTI,
IPsec non specifico per singola APP, ma tutto il traffico.



IPsec opera su L3, ma può proteggere payload IP pkt e IP header



TLS opera tra layer 4 e 'Application', protegge SOLO IL PAYLOAD di TCP. Non protegge TCP.

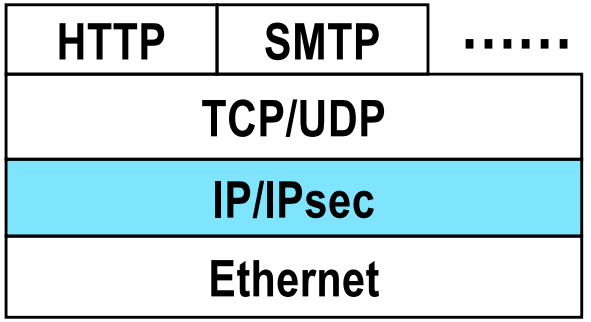
Network layer security (in IP) | Transport layer security (inside IP)

- ➔ **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

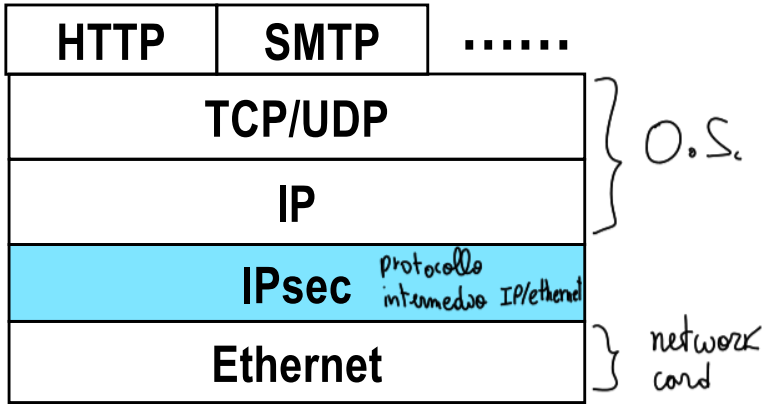
➔ Inside the native IP code (Linux)

- ⇨ Best approach (inside Kernel)
- ⇨ But hard to deploy as requires to access and modify IP source code



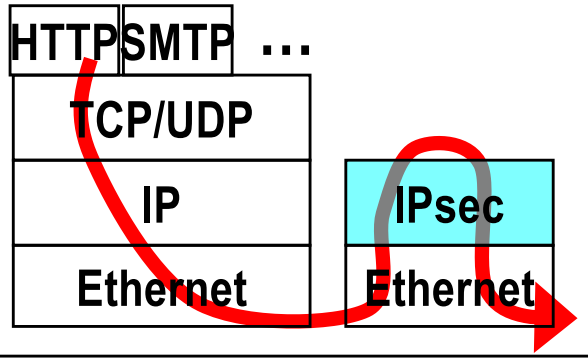
➔ Bump in the Stack (BITS)

- ⇨ 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 (come gateway)



Separa handshake part da transport part (2 protocolli). in TLS faccio "tutto insieme"

IPsec standardization History

➔Three major "series" of RFCs

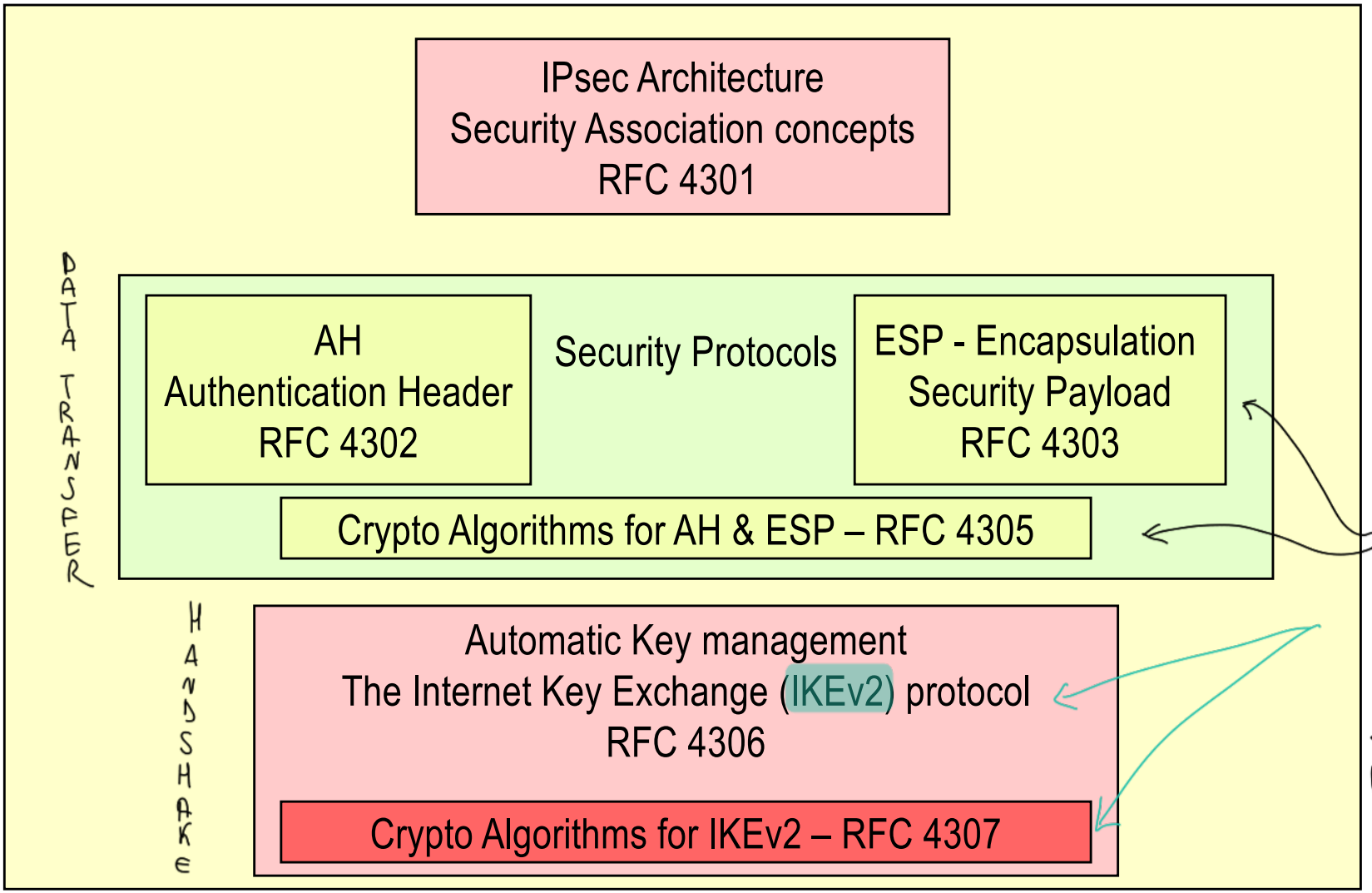
- ⇨Serie 1: RFC 1825-1827 (august 1995)
  - ➔IPsec concepts first drafted
- ⇨Serie 2: RFCs 2401-2412 (november 1998)
  - ➔Significant revision of ALL the IPsec architecture
  - ➔Describes IPsec as we know it today
- ⇨Serie 3: RFC 4301-4307 (december 2005)
  - ➔Born after long discussion in WG (almost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

➔Lots (!) of RFCs – see RFC6071 for a snapshot of IPsec- and IKE-related RFCs

↳ protocol of IPsec family. Ike equivalente a handshake (non uguali, ma per capirel

IPsec RFCs

gestisco filtri, network .. più complesso a livello RFC rispetto TLS.



distinzione algoritmi e protocollo. Importante perché posso aggiornare solo gli algoritmi (no hardcoded)

Security Association

esempio

in TLS

Giuseppe ⟷ Server (session, data transfer)

ALTERNATIVA (es Wifi)

Giuseppe *
- PSK 1234
- alg: AES-GCM-128

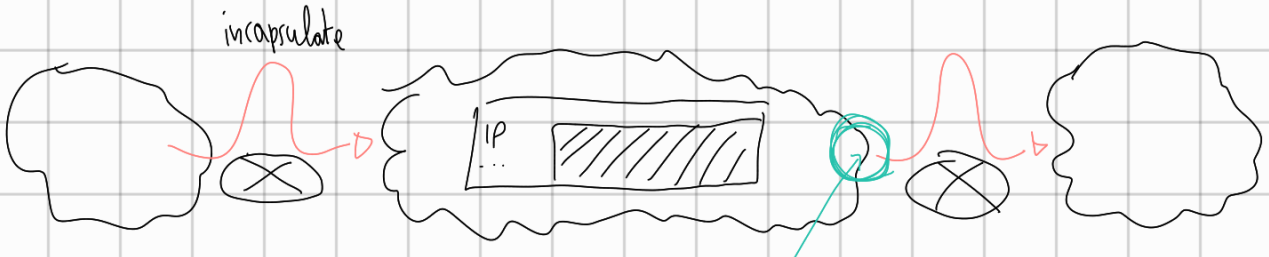
Server
PSK 1234

non serve scambiare dati, configuro solo telefono e router

Chiamo la config Security Parameter Index, SPI = 0001 ≡ ✍

e SPI = 0002 (es: cambio segreto e alg!). Fattibile offline.

Ho creato Security Association, relazione tra nodi in cui specifico come operare.

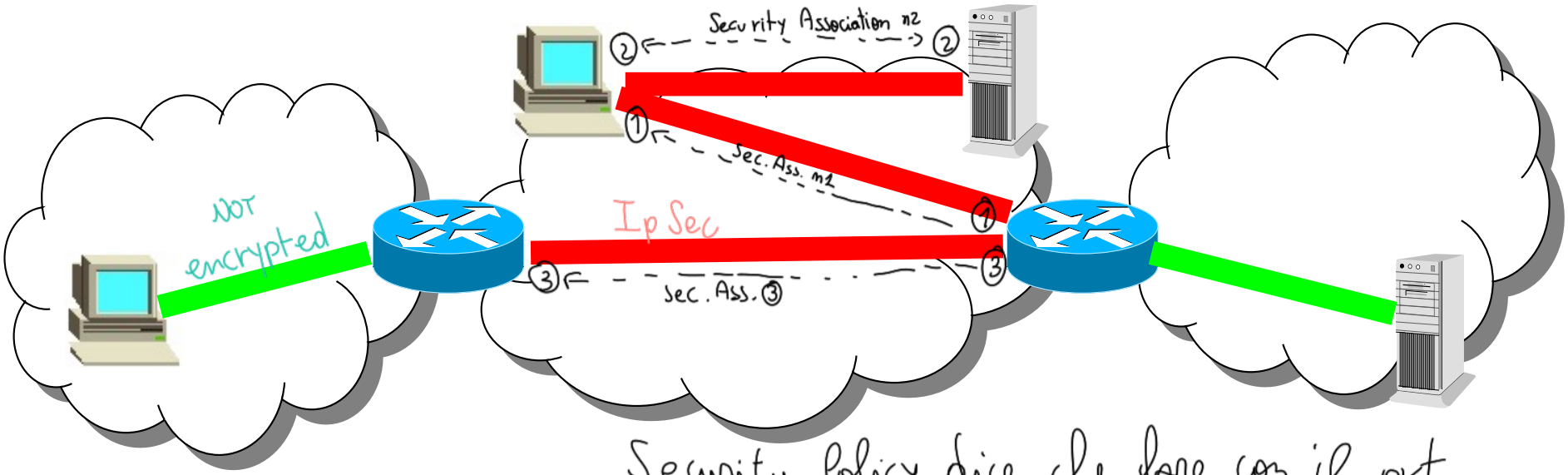


come decripto la parte interna?

aggiungo SPI label, quindi esso mi specifica segreti etc per decriptare il msg! Security association

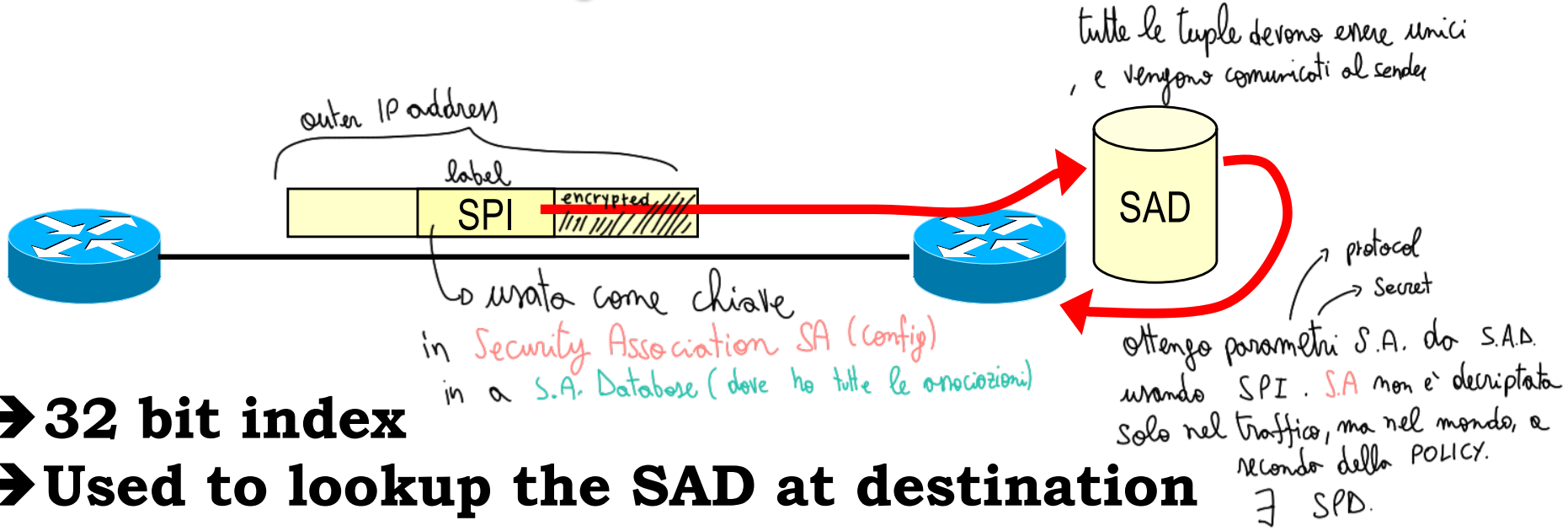
(è una config) # Security Association (non è connection)

- **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



Security Policy dice che fare con il pkt

Security Parameters Index and Security Association DB



(NON PARLO DI TRAFFICO)

➔ **32 bit index**

➔ **Used to lookup the SAD at destination**

⇨ Lookup also uses

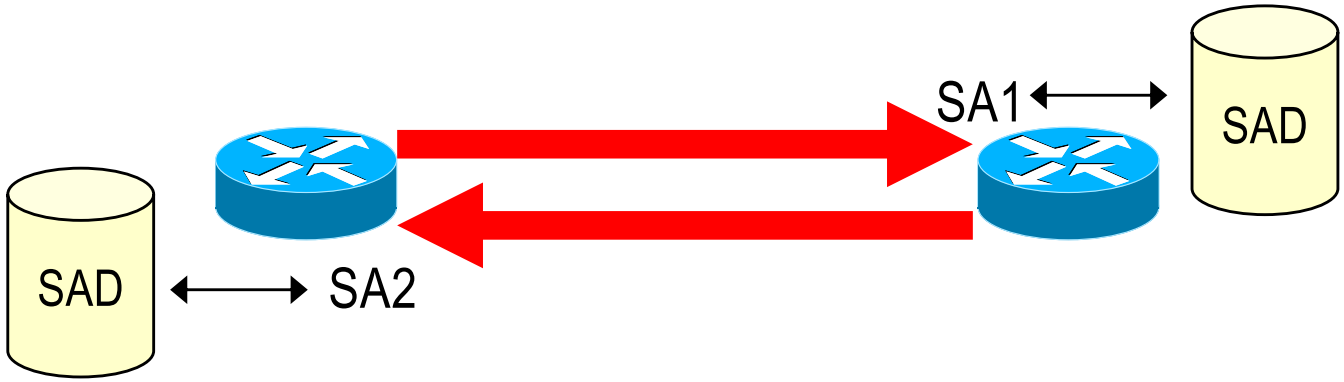
➔ destination address

➔ source address

➔ security protocol (AH/ESP)

➔ **Retrieves algorithms and parameters that allow to process received packet**

SA: monodirectional!



➔**SPI = Security Parameters Index**

⇨32 bit “name” of an SA

➔**Destination Address based**

⇨Security Association managed at the receiver side

➔**SAD = Security Associations Database**

⇨SPI = search key (at least)

➔But IPsrc and Ipdest can also be used in the (non trivial) lookup

⇨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

Security Association and Key management

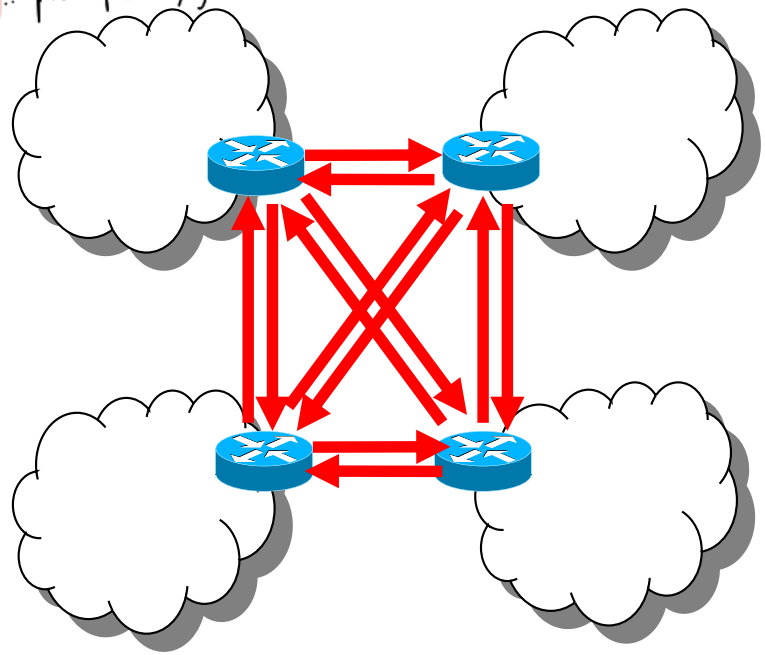
➔ **Manual** (per piccole reti, ∀ LINK 2 S.A. (monodirezionali) ... più policy)

- ⇨ 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
- ⇨ On-demand SA creation
- ⇨ Session-oriented keying/rekeying

& Setup → secret, → algorithm, → ....



config + rekey. (come handshake)

Ipsec security protocols

➔ **Authentication Header** (realizza integrity)

- ⇨ Integrity and data origin authentication
  - ➔ Authentication covers both payload and parts of IP header that do not modify in transfer
- ⇨ Protection against replays (integrity solo non mi protegge)
  - ➔ Optional, through extended sequence numbers

➔ **Encapsulated Security Payload** ≡ integrity + encryption ⇒ $A_{uthenticated}$ $E_{ncryption}$

applicato a IP payload.
Come faceva in TLS

- ⇨ 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

"small protection" (padding fino a 255) e dummy packet.
IPsec non fornisce supporto a Δt e fragmentation (complessi). in TLS ho nulla!

IPsec Security protocols

➔ **AH (Authentication Header)** oggi poco importante, uso il tunnel

- ⇨ Authentication (whole packet, including IP header) only

➔ **ESP (Encapsulated Security Payload)**

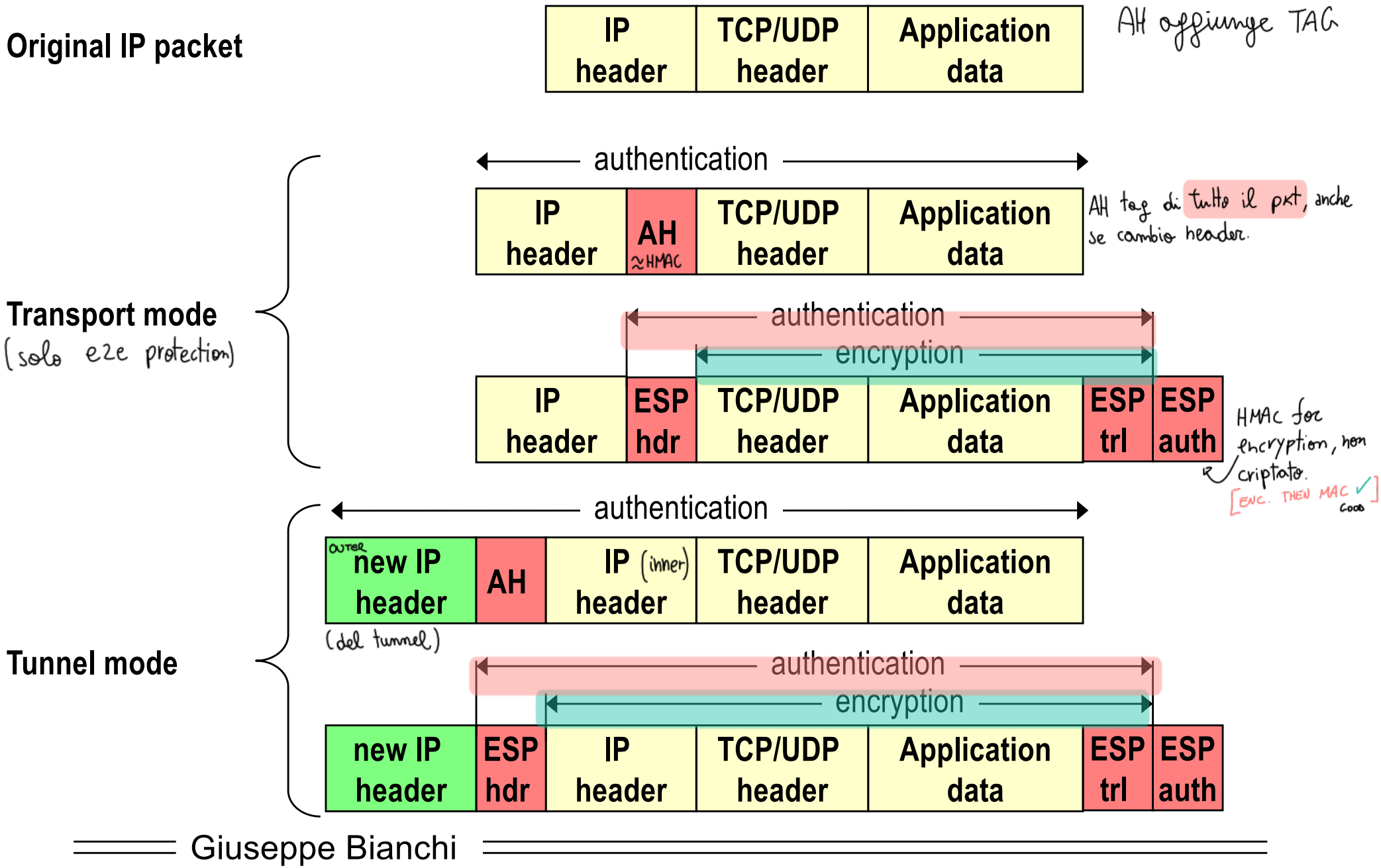
- ⇨ Payload-only protection (no IP header!)
- ⇨ Encryption, Authentication, both, AEAD

➔ **ESP in most cases is the only one needed**

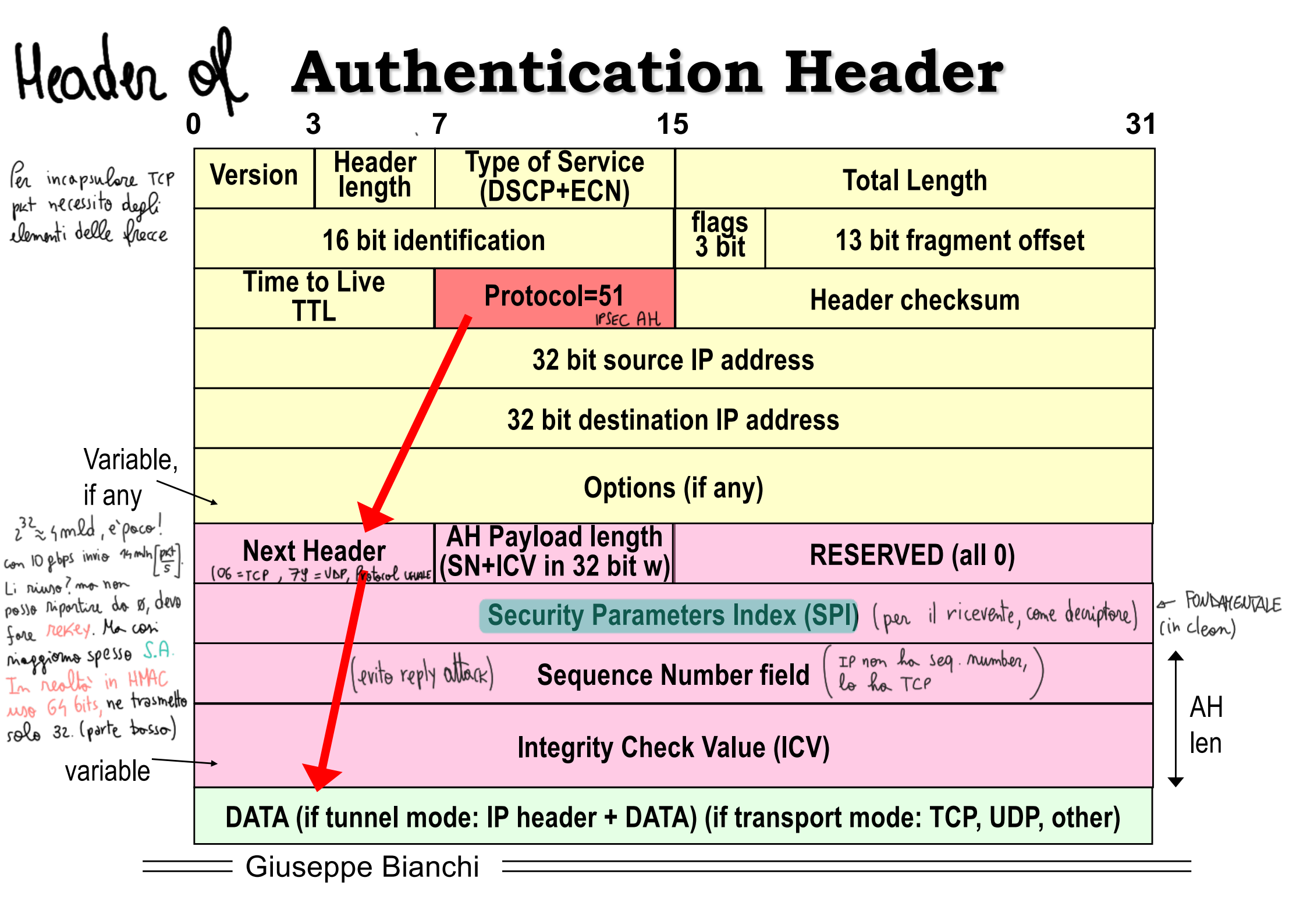
- ⇨ Mandatorily supported in any IPsec implementation, unlike AH
  - ➔ RFC 4301: AH downgraded: from MUST to MAY (be supported)
    - » Quoting from RFC 4301: *"Experience has shown that there are very few contexts in which ESP cannot provide the requisite security services. Note that ESP can be used 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**

Transport vs Tunnel – AH and ESP



Header of Authentication Header

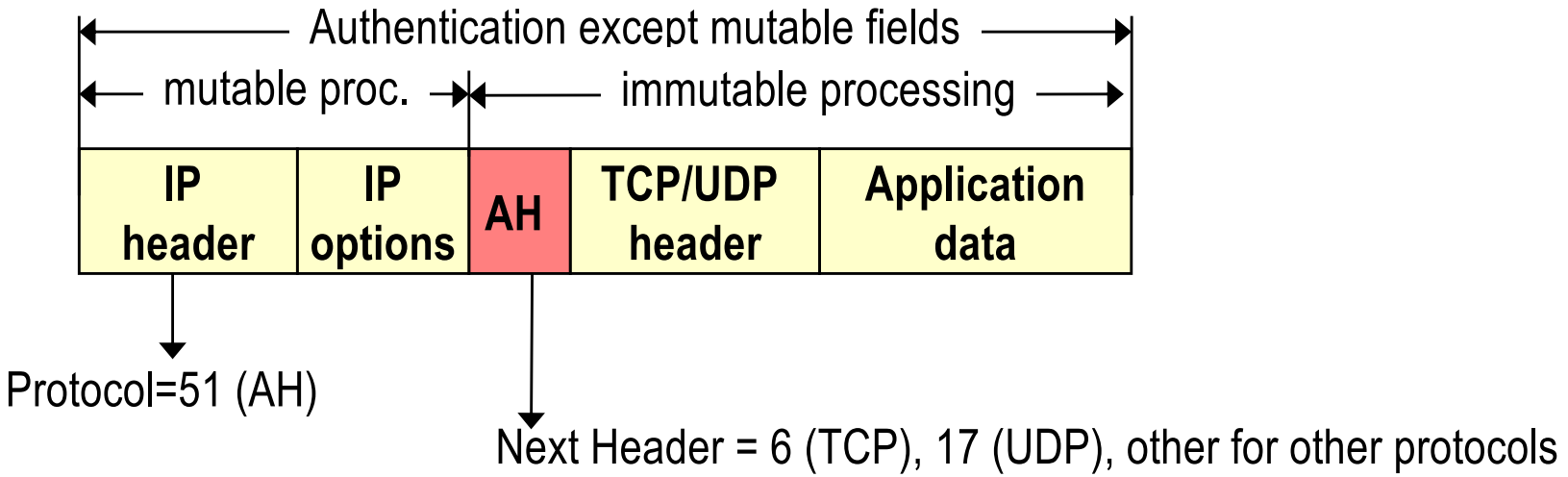


Per incapsulare TCP pkt necessito degli elementi delle frecce

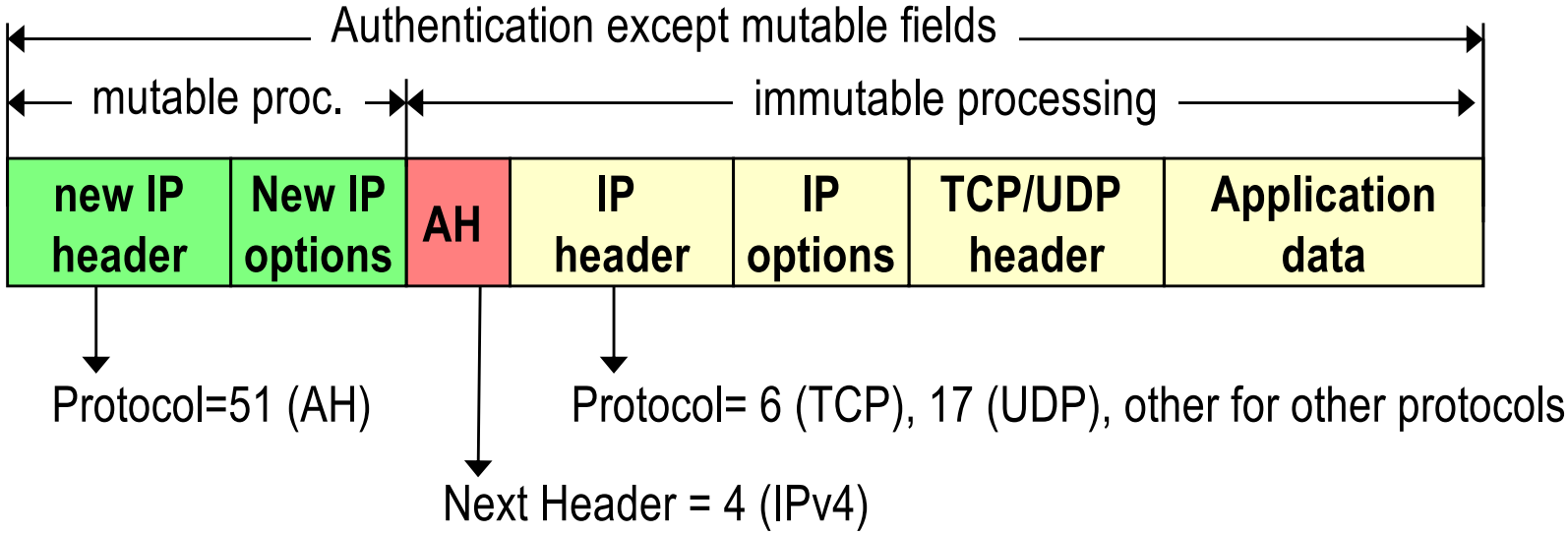
$2^{32} \approx 4$ mld, è poco! con 10 gbps invio 4 mln [pkt/s]. Li riuso? ma non posso ripartire da 0, devo fare rekey. Ma così aggiorno spesso S.A. In realtà in HMAC uso 64 bits, ne trasmetto solo 32. (parte bassa)

Transport mode, tunnel mode (SKIP)

Transport mode:



Tunnel mode:



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

fornito nel network (operatori), no e2e. Specifica, ad es, come Tim gestisce pkt da Fastweb etc....

variano
statico

Se applico integrity protection, non posso fare HMAC di tutto, ma solo delle cose statiche (sennò sembra attacco), e metto maschere per i dinamici.

Version	Header length	Type of Service (DSCP+ECN)	Total Length	
16 bit identification			flags 3 bit	13 bit fragment offset
Time to Live (cambia Yhop) TTL		Protocol=51 (AH)	Header checksum (cambia TTL)	cambia Yhop
32 bit source IP address				
32 bit destination IP address				

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 umpredictable 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

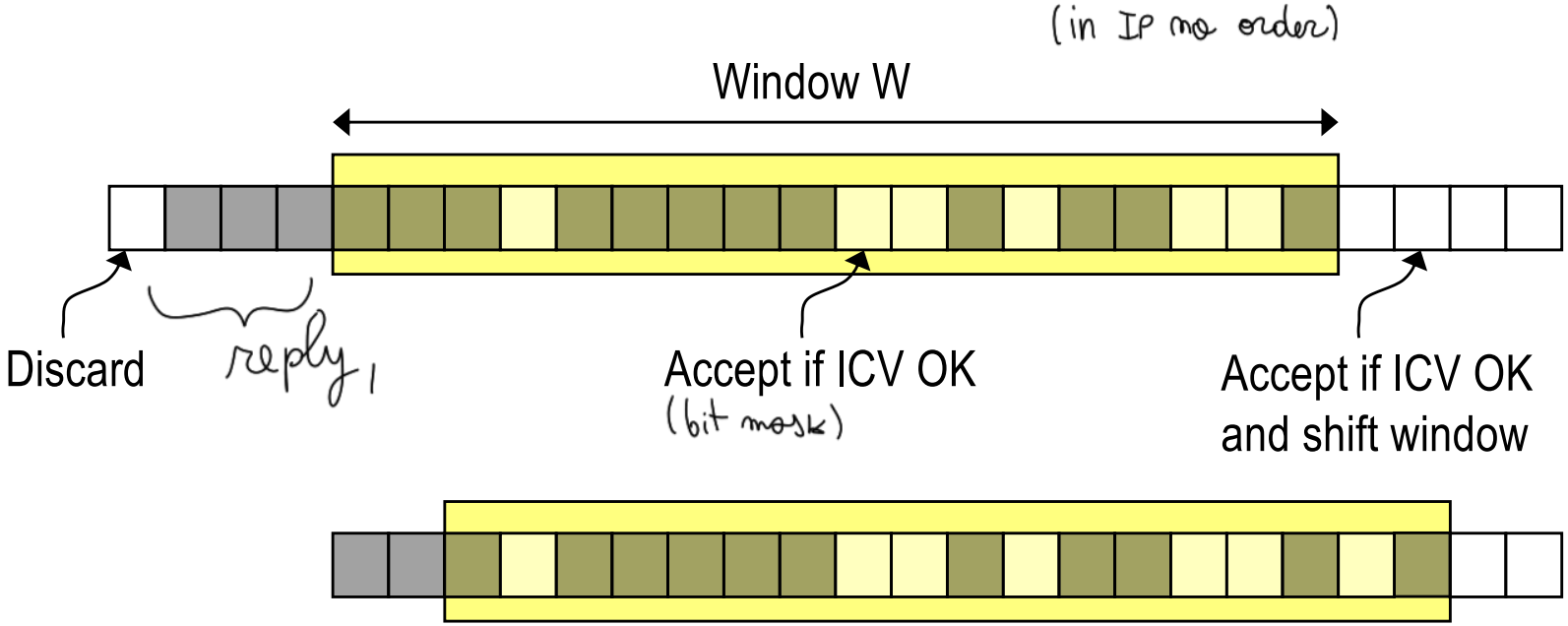
Extended Sequence Number

- **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 (gli altri 32 stanno nella S.A.)
  - But use high order 32 bits in ICV computation!

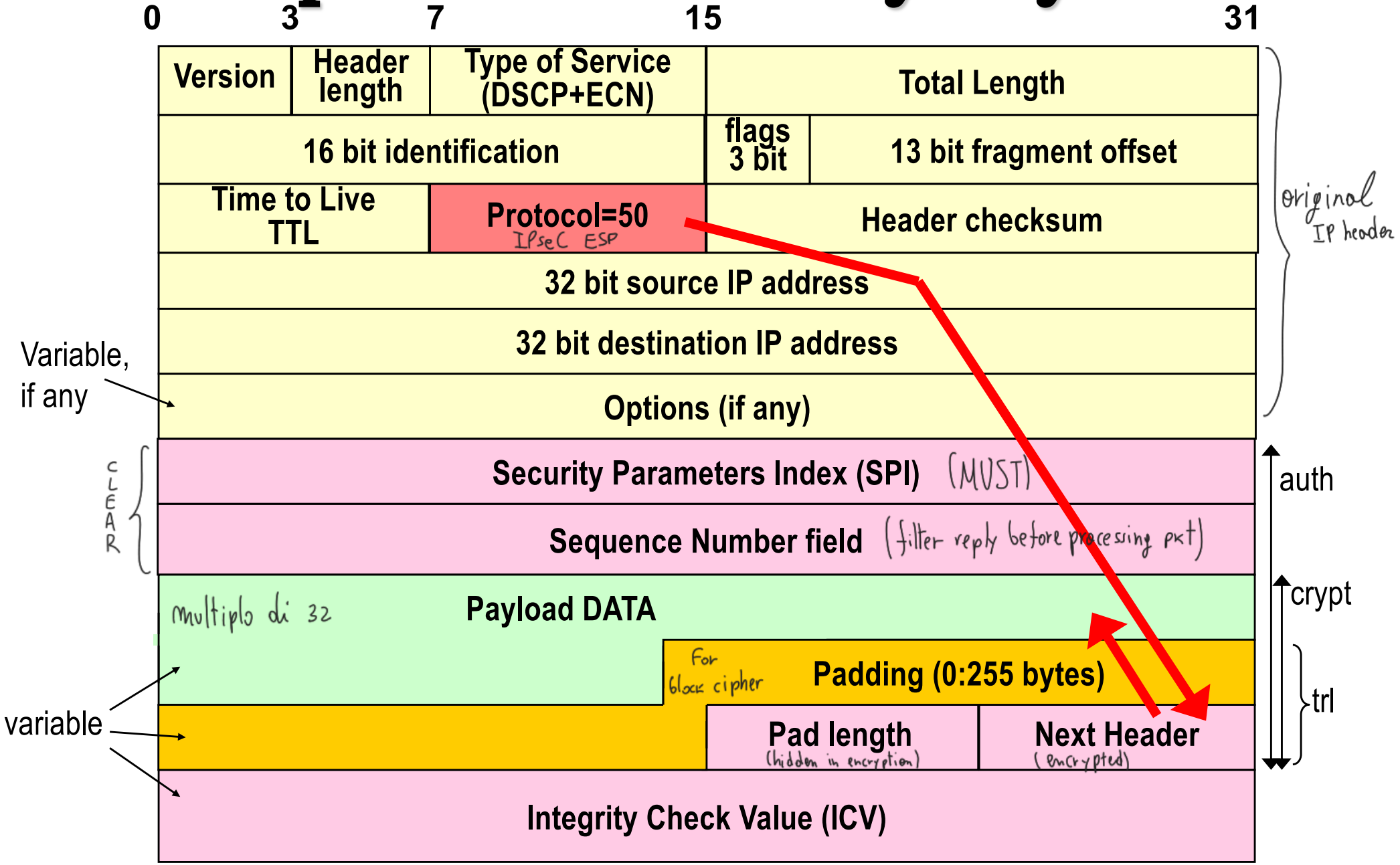
Integrity Check Value, perché non MAC?
IPSec è low level, MAC fa pensare all' ethernet, ma come MAC.

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**



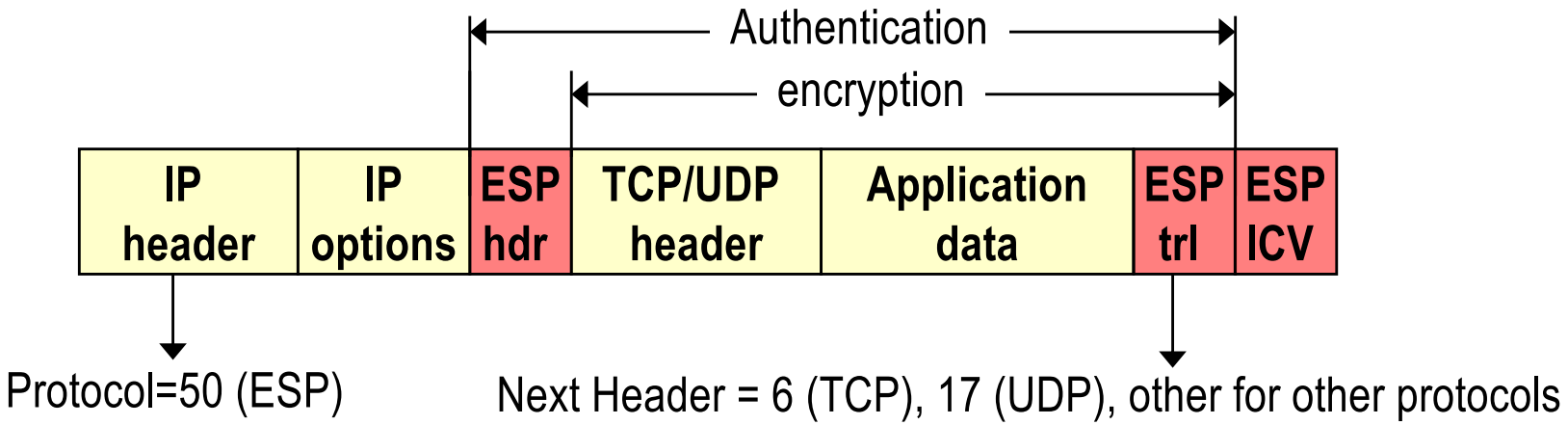
Header Encapsulated Security Payload



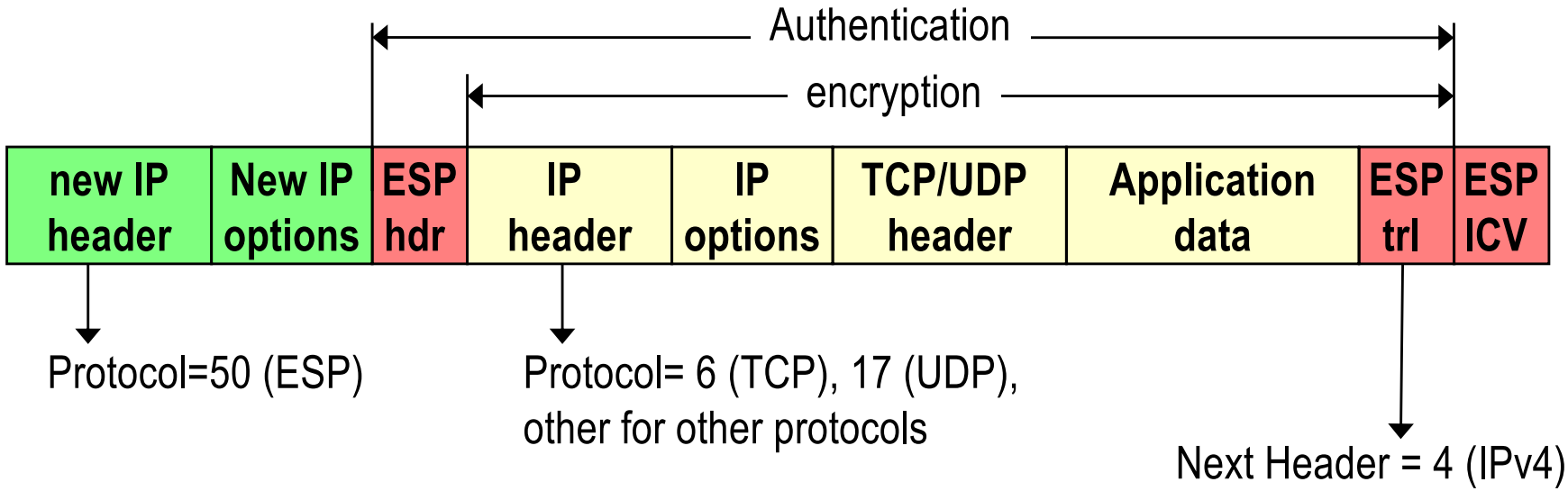
Vedo solo IP header, protocol field = 50 (cioè contiene ESP-IPSEC payload). NON SO NULLA DELL'INNER PKT.
INTRODUCE TRAFFIC FLOW CONFIDENTIALITY

Transport mode, tunnel mode

Transport mode:

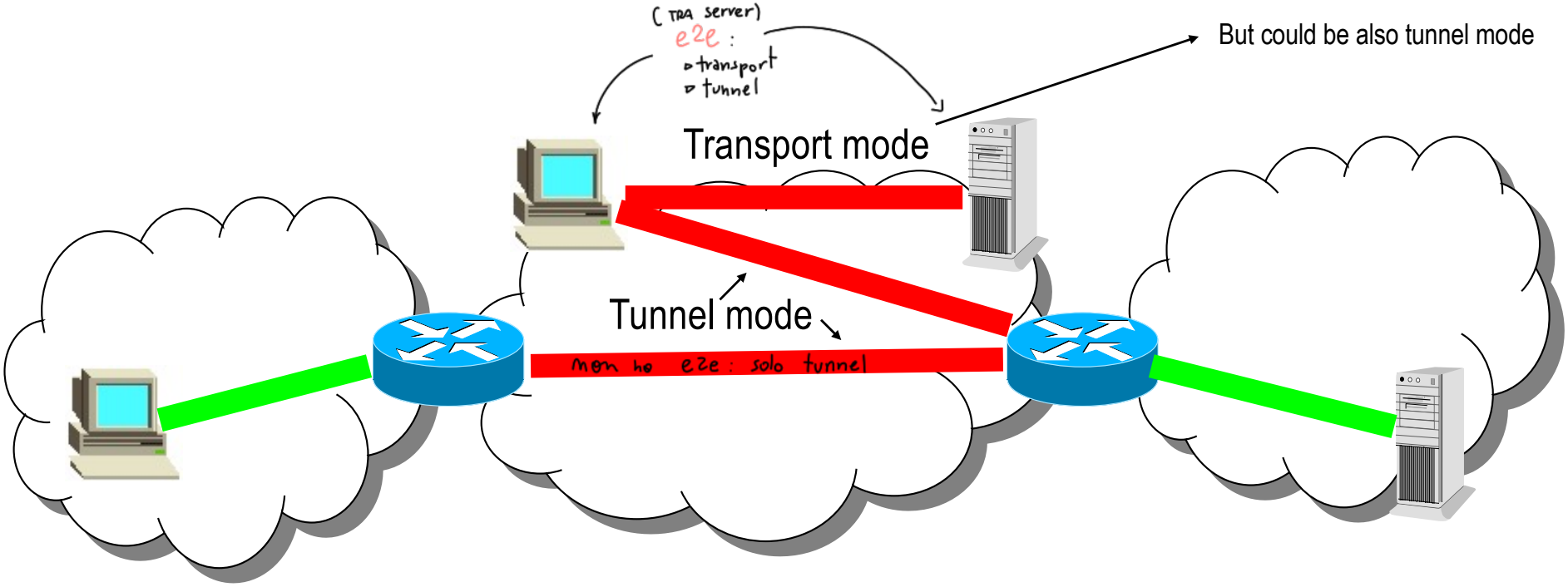


Tunnel mode:

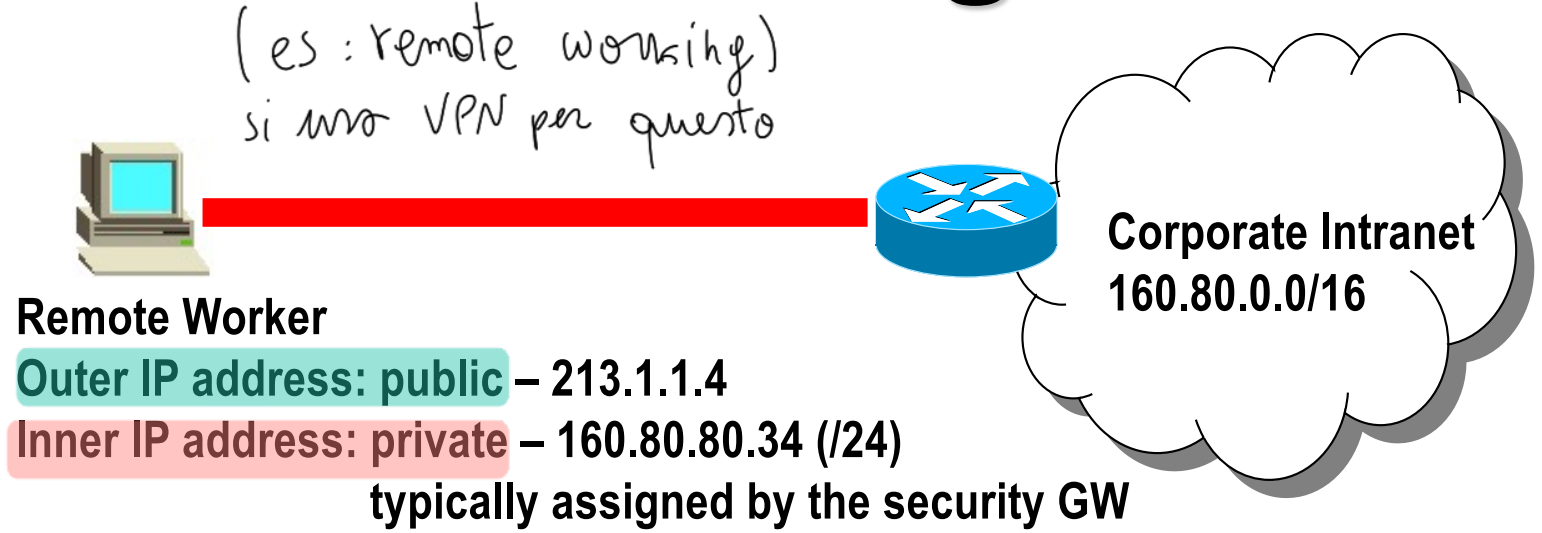


When transport? When Tunnel?

- **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

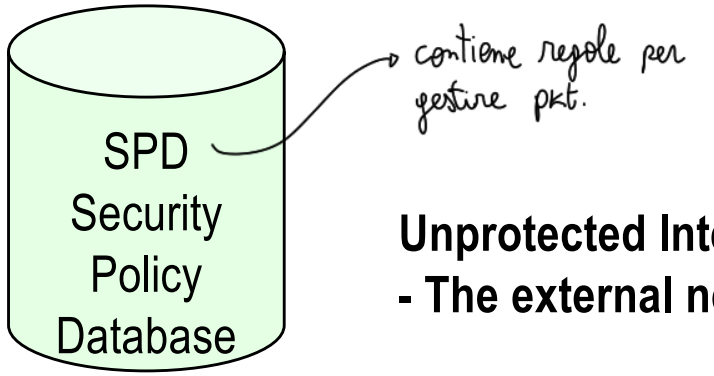


➔Using a private IP address inside the tunnel:

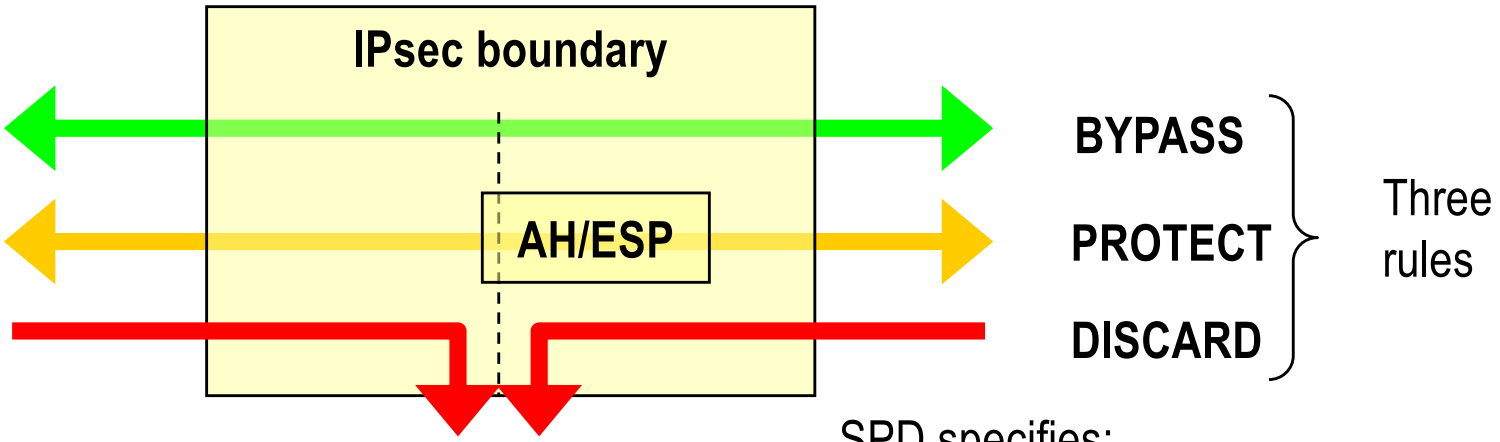
- ⇨Allows to access to all services provided in the intranet, exactly like in the case the worker is connected inside the corporate
- ⇨Allows to be protected by the corporate firewall (all traffic destined to 160.80.80.34 MUST be first routed to the corporate subnet and then routed to the end user in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

IPsec protection & access control

Protected Interface
- Internal Network with IPsec Security GW;
- Host OS with IPsec Host



Unprotected Interface
- The external network



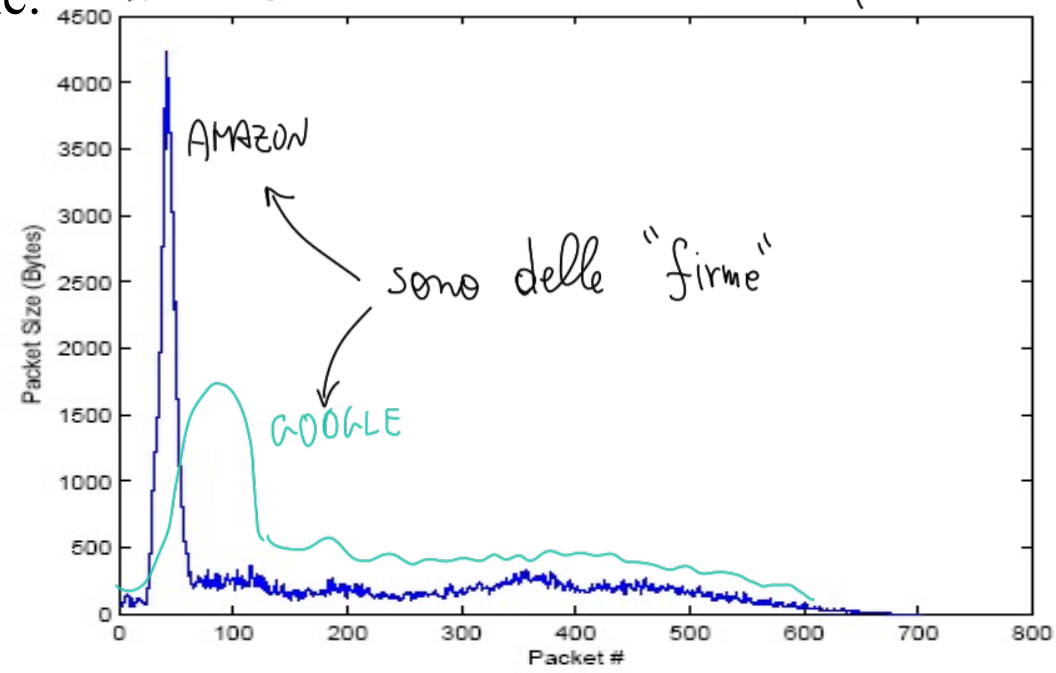
SPD specifies:
- Which security protocols (AH, ESP)
- On which packets/traffic relations
- Which mode (tunnel, transport)
- Which granularity
  - single shared tunnel
  - different tunnels per TCP conn

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.

Analizzando il fingerprint hanno ottenuto info sul traffico (anche se encrypted), in modo molto accurato!

Si è analizzato Pkt size (soprattutto all'inizio) e Δtime (tra pkt_i e pkt_{i+1})



- Many other traffic/protocol fingerprinting approaches (literature + pre-prints)

▶ dovrei equalizzare size & Δt, ma c'è anche pkt absence/presence (cioè se sono attivo o no!). IpSec può creare DUMMY TRAFFIC (next header = 59). Δtime NON gestito da IPSec, bisogna operare su Kernel. Size parzialmente, incremento PADDING (fino a 255, definito in Pad lenght (8 bit). Grandezza "limitata".

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?

  - Ability to generate "dummy" packets
    - Next Header = 59 → Dummy packet!!
    - Example: one can transform a VBR traffic into CBR
      - » Heavy on the network load, though: reduces statistical multiplexing effectiveness

attacker non lo vede, è encrypt, viene decriptato, NON INVIATO al sist. op

Part 2:

IKEv2

Note: minor updates in RFC 5996, following material based on RFC 4306

Rationale for IKE (come handshake per TLS)

➔ **shared state must be maintained between source and sink**

negoziazione
- ⇨ Which security services (AH, ESP)
- ⇨ Which Crypto algorithms
- ⇨ Which crypto keys

contenuto della S.A.

➔ **Manual maintenance not scalable**

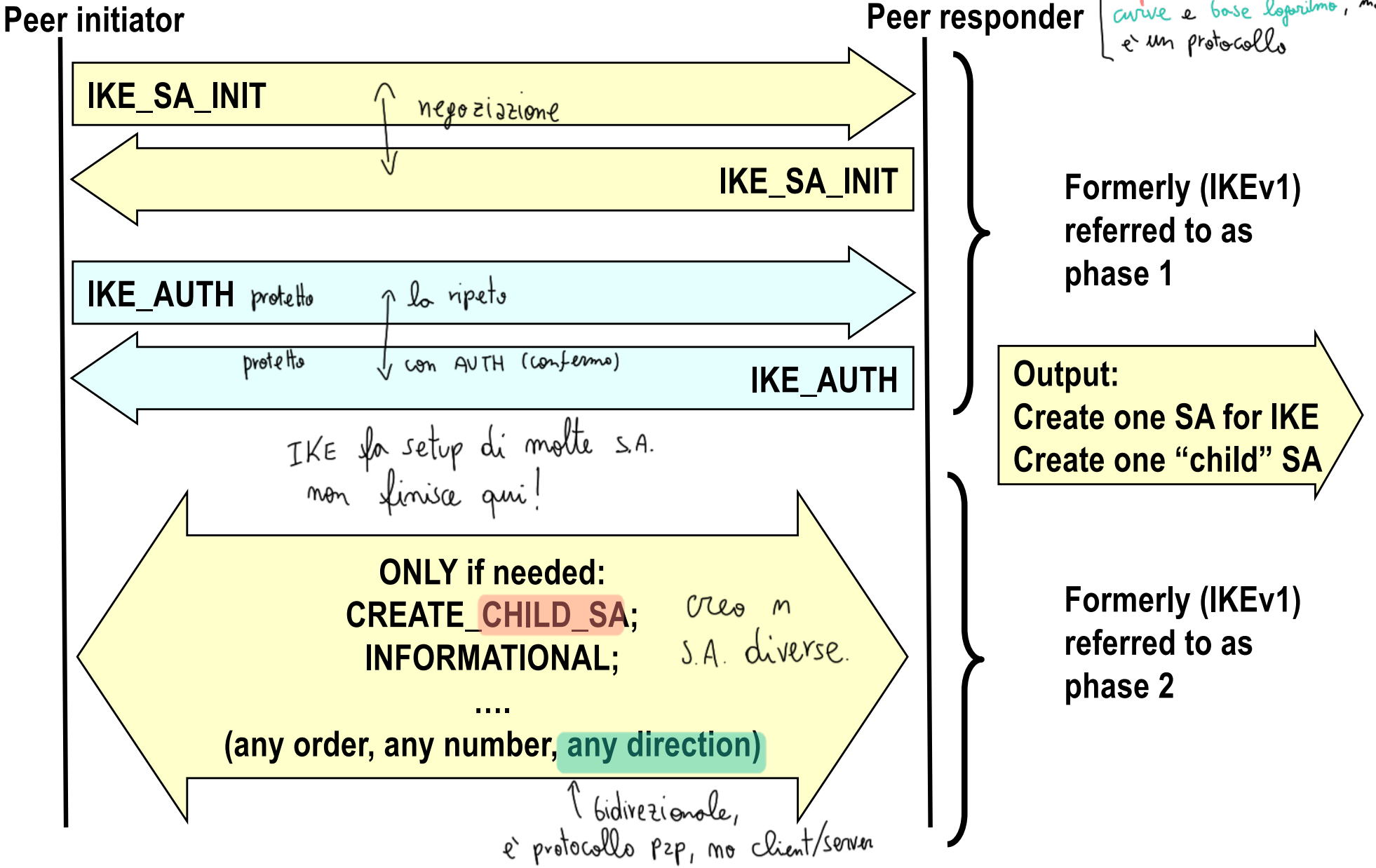
- ⇨ Partially OK only for small scale VPNs
- ⇨ In any case, weak approach
  - →Infinite lifetime SA → no rekeying!

➔ **IKE = Internet Key Exchange protocol**

- ⇨ Goal: dynamically establish and maintain SA
- ⇨ IKE now (2010, RFC 5996 → 2014, RFC 7296) in version 2
  - →Replaces protocols specified in RFCs 2407, 2408, 2409 (IKE, ISAKMP, DOI)
  - →extends IKEv2 in RFC 4306
  - →IKEv2 quite different (and much cleaner!!) than former specifications

IKE phases at a glance

DH hardcoded ha parte algoritmica dipendente dalle curve e base logaritmo, ma è un protocollo



IKE SA and CHILD SA

➔IKE SA:

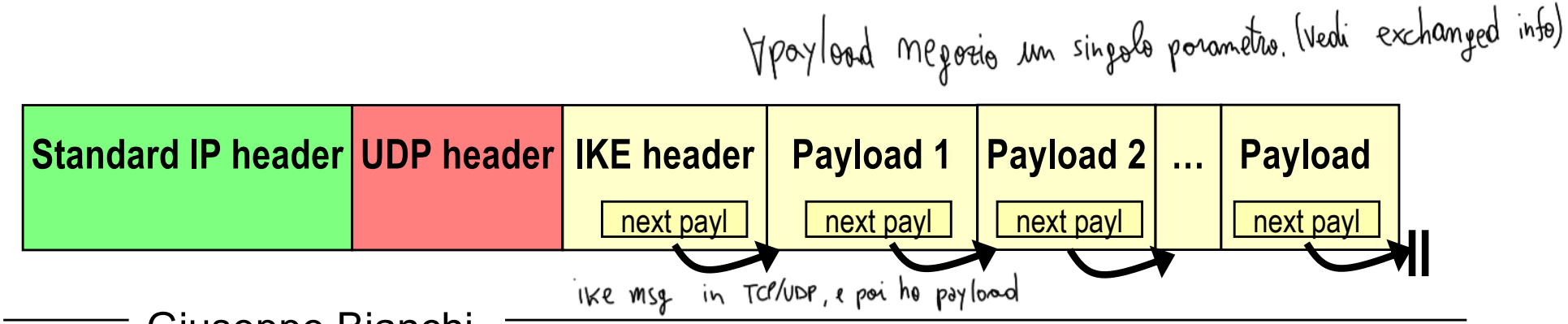
- ⇨Security association to exchange IKE messages (control messages)

➔CHILD SA

- ⇨Security association to exchange data messages
  - →Making use of AH or ESP
- ⇨Many CHILD SA may be set up between two peers

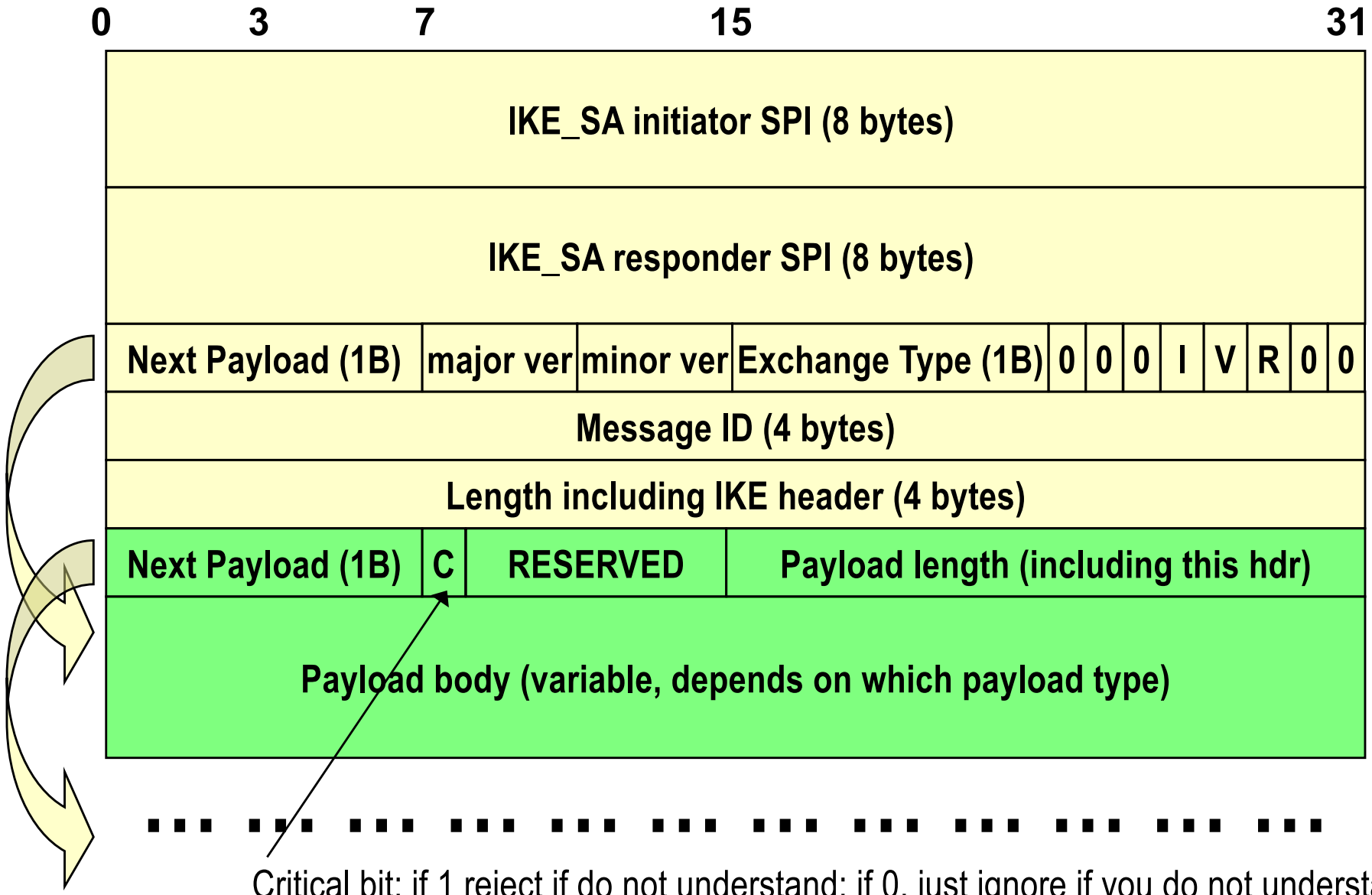
IKE message format

- **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



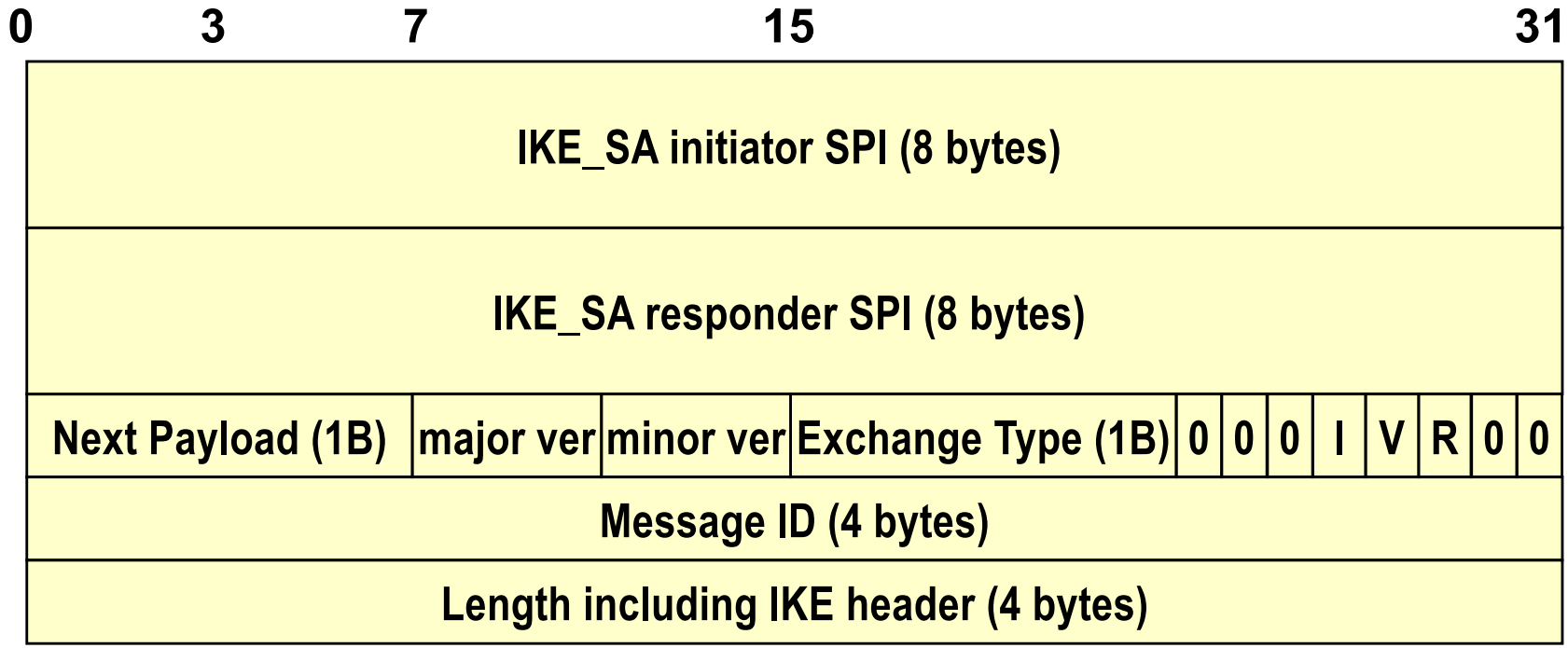
skip

IKE hdr, generic payload hdr



SKIP

IKE header (explanation)



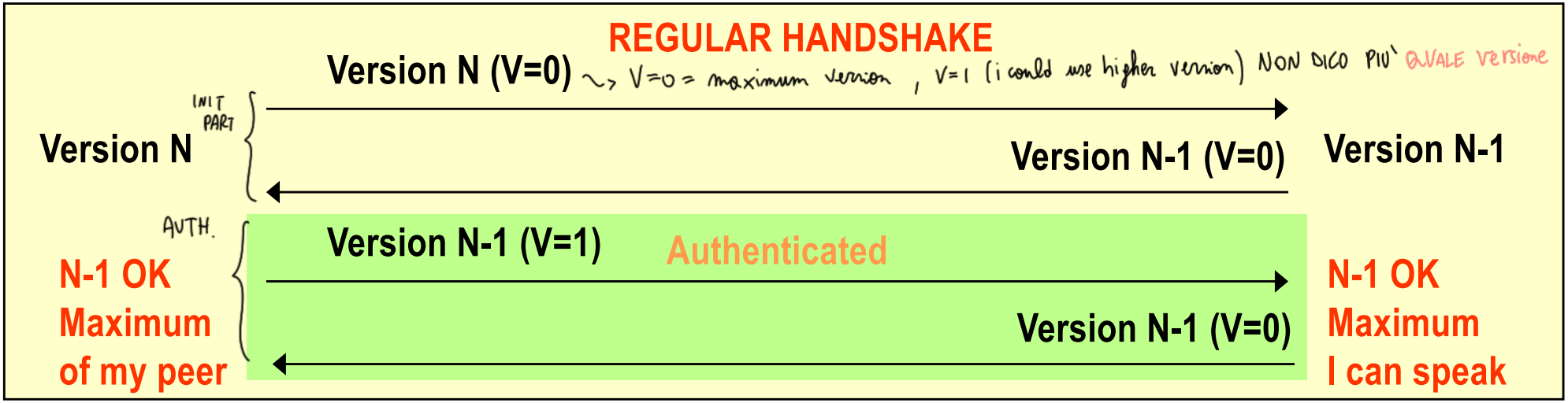
- **Initiator SPI, responder SPI:**
  - SA id for the IKE protocol. Initially, responder SPI set to 0; responder fills it in
    - don't confuse with the SPI for the CHILD AH/ESP SA
- **Versions: major = 2, minor = 0**
  - Bit V=1 says that implementation can "speak" higher version than what written 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**

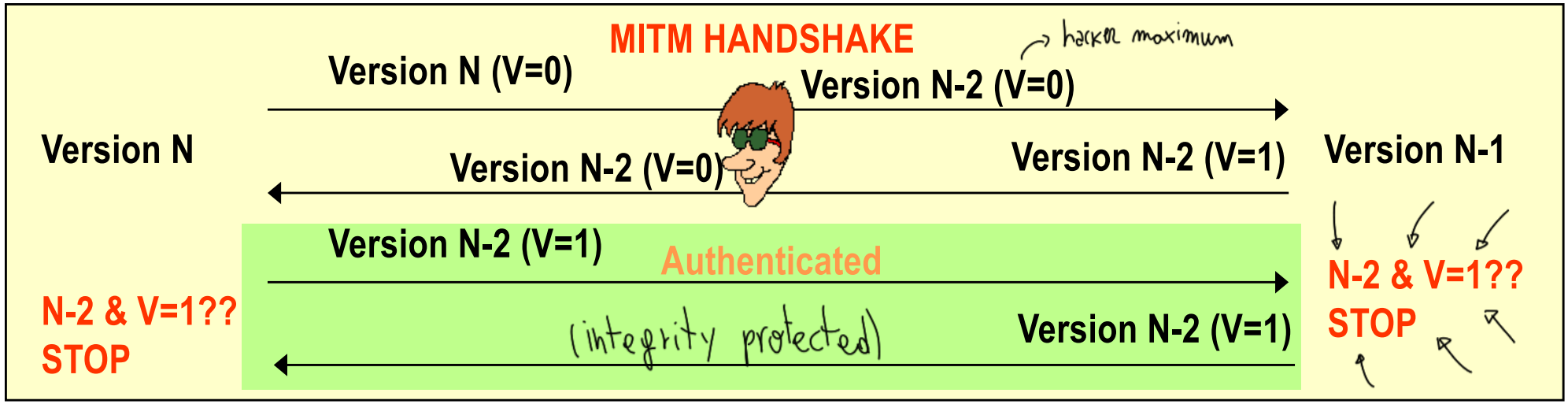
Version bit

posso usare 1 bit invece che 2 byte (TLS)

Protection against version rollback

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



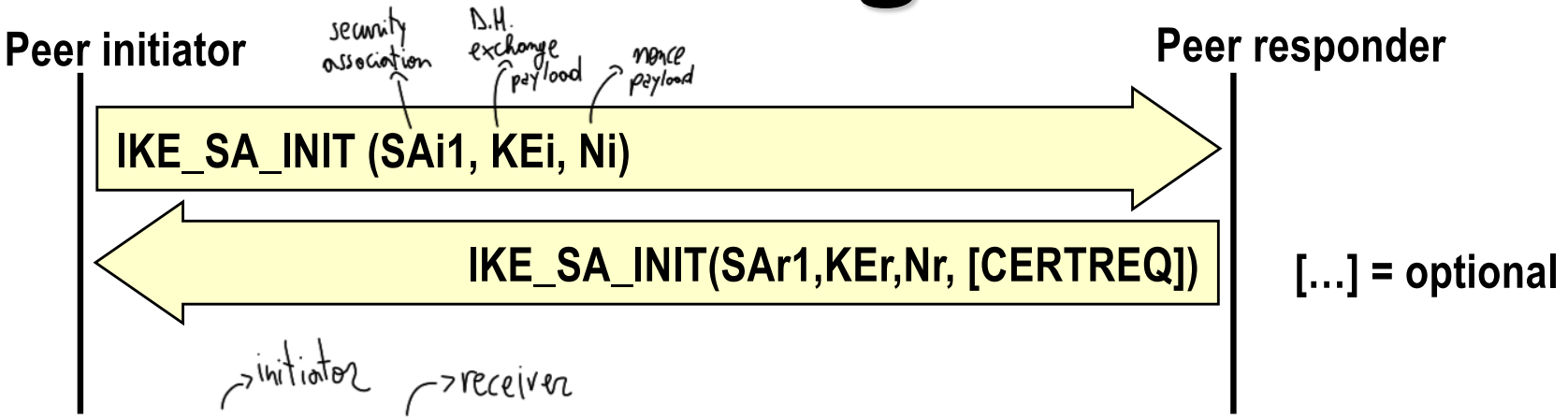


Too bad v1 does NOT include this flag!!! (Hence rollback to IKEv1 is not protected)

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:** (DH parameters)

- ⇨ 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

like TLS

Security Association Payload example

```
SA Payload
    |
    +--- Proposal #1 ( Proto ID = ESP(3), SPI size = 4,
    |      |             7 transforms,      SPI = 0x052357bb )
    |      |
    |      +-- Transform ENCR ( Name = ENCR_AES_CBC )
    |      |     +-- Attribute ( Key Length = 128 )
    |      |
    |      +-- Transform ENCR ( Name = ENCR_AES_CBC )
    |      |     +-- Attribute ( Key Length = 192 )
    |      |
    |      +-- Transform ENCR ( Name = ENCR_AES_CBC )
    |      |     +-- Attribute ( Key Length = 256 )
    |      |
    |      +-- Transform INTEG ( Name = AUTH_HMAC_SHA1_96 )
    |      +-- Transform INTEG ( Name = AUTH_AES_XCBC_96 )
    |      +-- Transform ESN ( Name = ESNs )
    |      +-- Transform ESN ( Name = No ESNs )
    |
    +--- Proposal #2 ( Proto ID = ESP(3), SPI size = 4,
           |             4 transforms,      SPI = 0x35a1d6f2 )
           |
           +-- Transform ENCR ( Name = AES-GCM with a 8 octet ICV )
           |     +-- Attribute ( Key Length = 128 )
           |
           +-- Transform ENCR ( Name = AES-GCM with a 8 octet ICV )
           |     +-- Attribute ( Key Length = 256 )
           |
           +-- Transform ESN ( Name = ESNs )
           +-- Transform ESN ( Name = No ESNs )
```

negozio "separatamente"

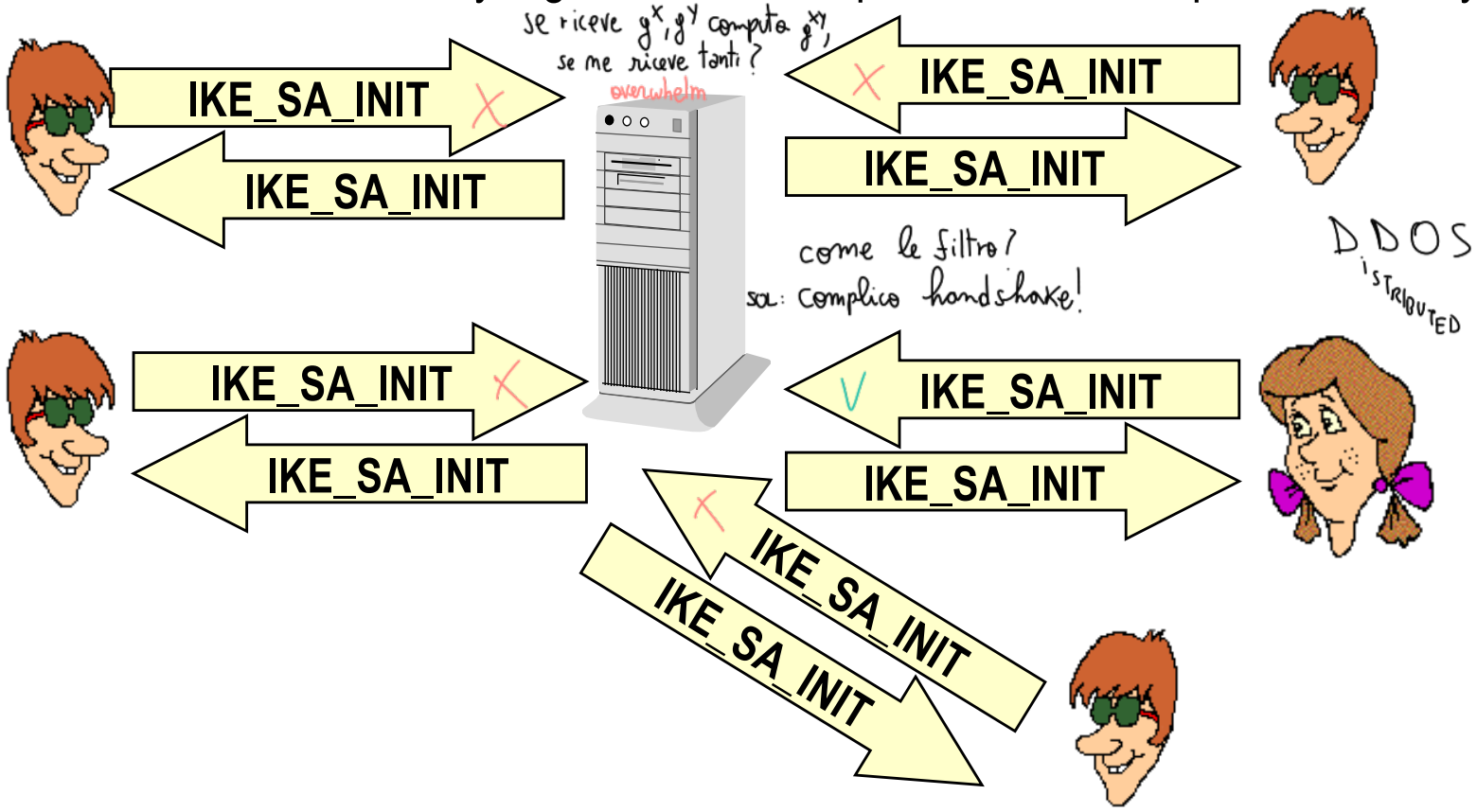
Key generation

- **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

derivo chiavi diverse ∀ direzione, ∀servizio, uso HKDF

Protection against DoS attacks

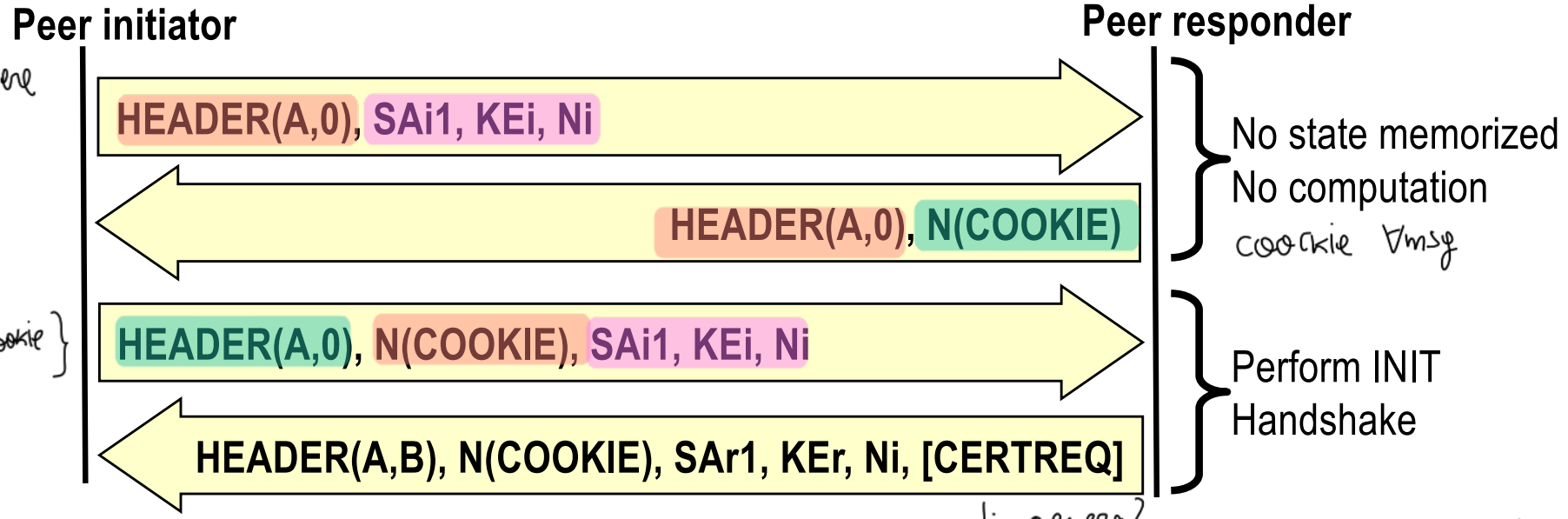
- **DH computation and key generation is a computational expensive process; state must be memorized**
- **Denial of Service Attack: spoof INIT requests (using forged IP addresses) and overload server (exhaust CPU and memory)**
  - ⇨ When load is sufficiently high, "Normal" requests cannot be processed anymore



NB: ipotizzo IP address fake (SPOOF) — NON FISICI
non c'è processo reale che genera i pkt che arrivano al server

SOLUZIONE: chiedo di ritrasmettere

Solution: cookie-based 4-way INIT handshake



come li genero?
- STATICI? NO, attacker forebbe spoof con cookie (anticipa)
- deve essere non prevedibile. Caso dB che ∀ richiesta associa cookie? NO! memory overhead

come faccio!?

➔ 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

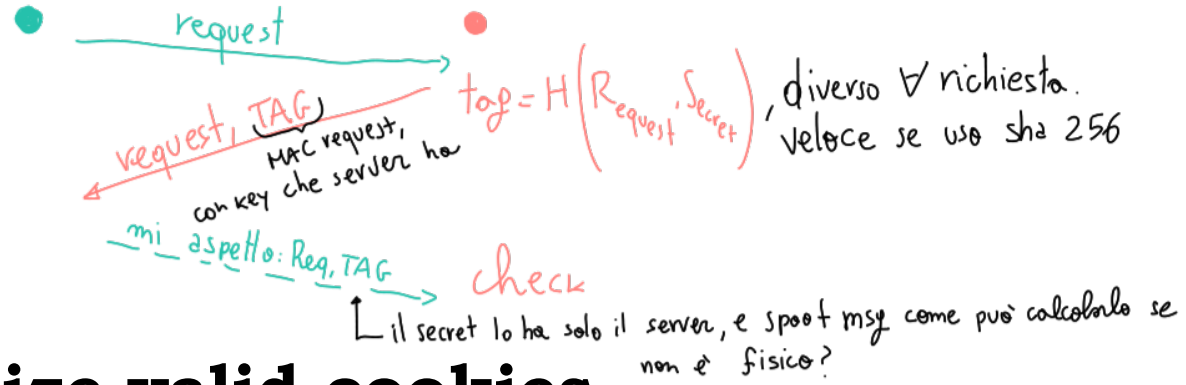
- ⇨ Typical attacks!

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!!

posso cambiare secret?
ho tabella

key version	actual secret
10 a.m.	1234
15 a.m.	7890

… is this true?

Idea: use stateless cookies

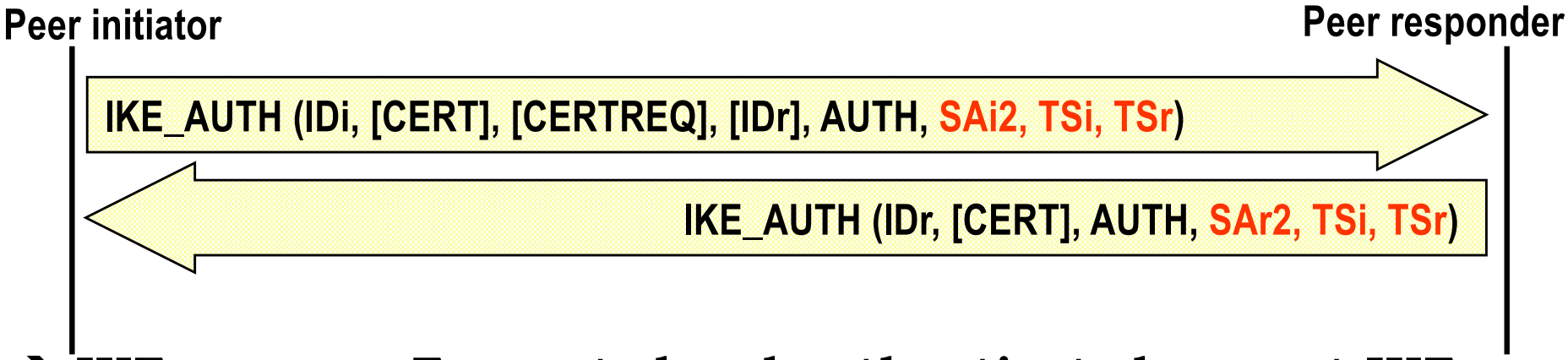
- **Use cookies which do not require any state memorization**
  - ⇨ i.e. the validity of the cookie may be checked without any lookup at a database of issued cookies, but only looking at the request and at the cookie itself
- **Example:**

info in IKE init pkt

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
- **Report error conditions**
- **Empty INFORMATIONAL payload = check for liveness**

Fine parte 2 del corso.