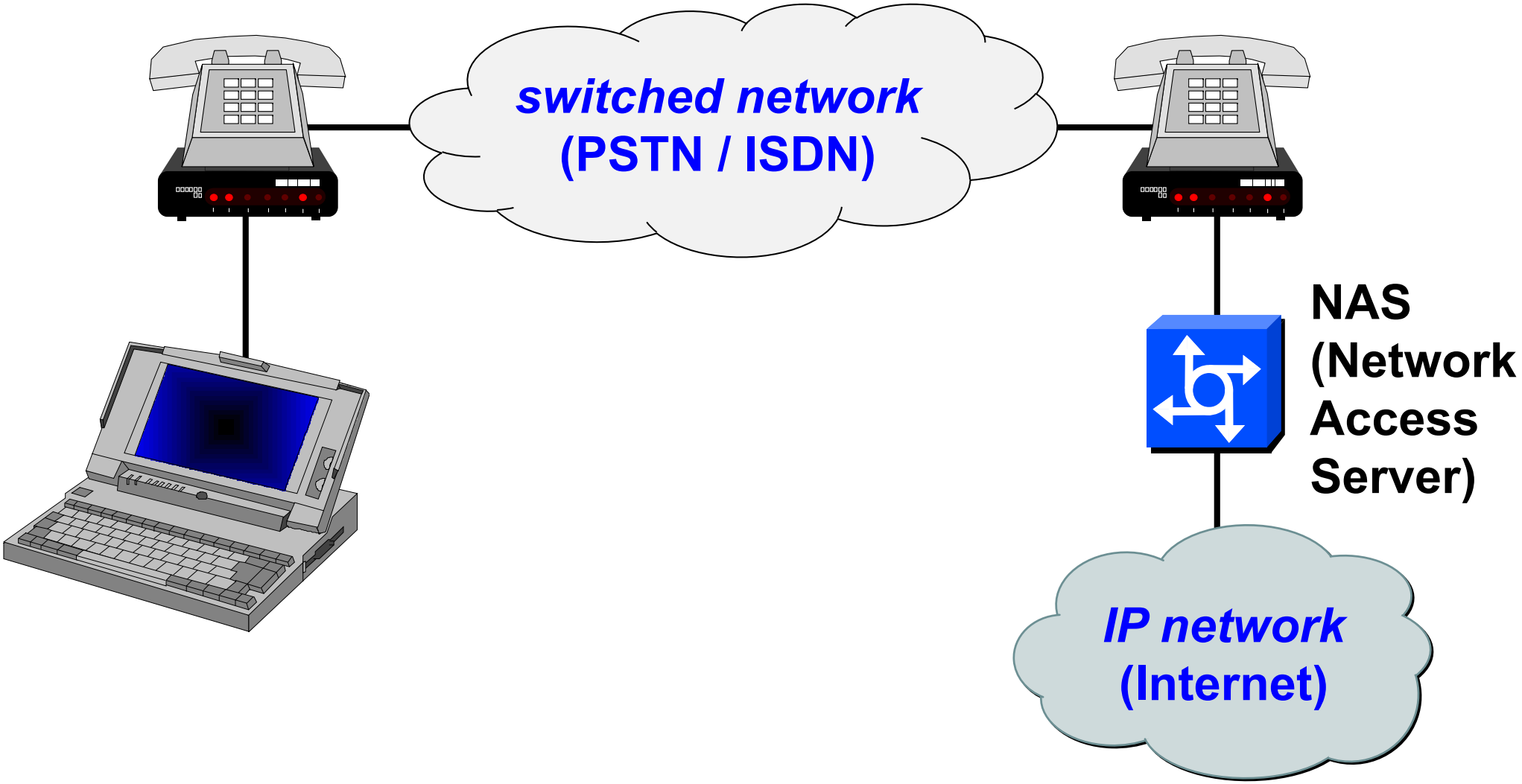
Security of IP networks

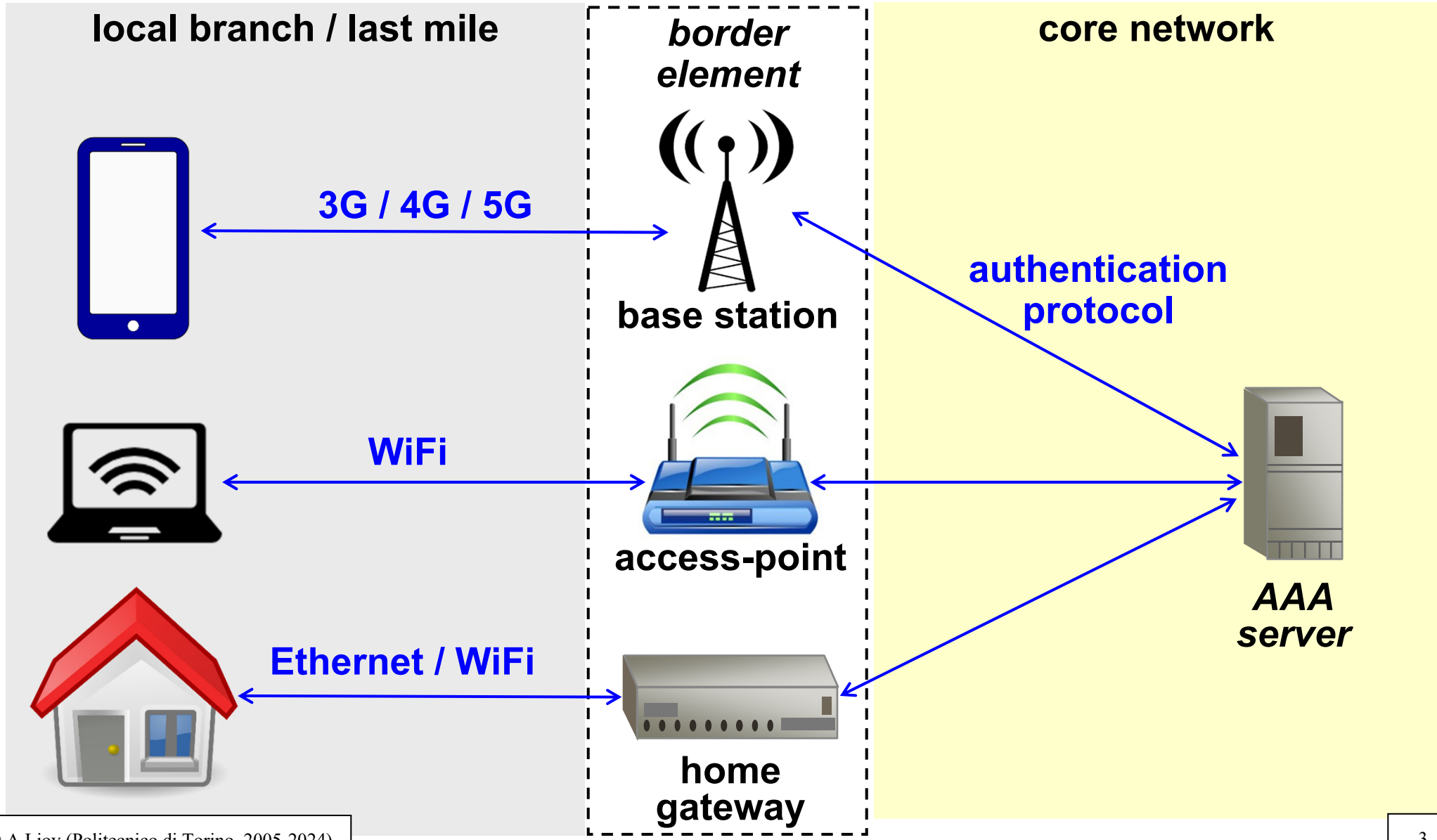
Antonio Lioy
< lioy @ polito.it >

Politecnico di Torino
Dip. Automatica e Informatica

Network Access Control (for connection via dial-up lines)



Network Access Control (modern way)



Authentication of PPP channels

- **PPP (Point-to-Point Protocol) is able to encapsulate network packets (L3, e.g. IP) and carry them over a point-to-point link**
  - physical (e.g. PSTN, ISDN)
  - virtual L2 (e.g. xDSL with PPPoE)
  - virtual L3 (e.g. L2TP over UDP/IP)
- **activated in three sequential steps:**
  - LCP (Link Control Protocol)
    - establishing, configuring, and testing the L2 connection
    - can negotiate also authN protocol and algorithm
  - authentication (optional; PAP, CHAP, or EAP)
  - L3 encapsulation via various NCPs (Network Control Protocols)
    - e.g. IPCP (IP Control Protocol) for IP packets

Authentication of network access

- **protocols used not only for PPP**
- **PAP**
  - Password Authentication Protocol
  - user password sent in clear
- **CHAP**
  - Challenge Handshake Authentication Protocol
  - symmetric challenge-response (based on user password)
- **EAP**
  - Extensible Authentication Protocol
  - it's an authentication framework
  - … to use external techniques (e.g. challenge, OTP, TLS)

LCP Authentication-Protocol Configuration Option

- **this option contains:**
  - Type (8bit) = option type
  - Length (8bit) = length of option in bytes
  - Authentication Protocol (16bit) = protocol identifier
  - [Algorithm (8bit)] = algorithm identifier (needed when a protocol supports various ones)
- **for PAP:**
  - Type=3, Length=4, Protocol=0xC023
- **for CHAP:**
  - Type=3, Length=5, Protocol=0xC223, Algorithm=5 (for MD5)

PAP

- Password Authentication Protocol
- RFC-1334 "PPP Authentication Protocols" (Oct 1992)
  - note: defines also initial version of CHAP
- user-id and password sent in clear between Peer and Authenticator
- authentication only once when the channel is created
- very dangerous!

PAP: 2-way handshake protocol

- **(Peer > Authenticator) Authenticate-Request (code=1)**
  - Code (8bit) + Identifier (8bit) + Length (16bit)
  - Peer-ID Length (8bit) + Peer-ID (0-255B)
  - Passwd-Length (8bit) + Password (0-255B)
- **(Authenticator > Peer) Authenticate-Response (code=2, 3)**
  - Code (8bit) + Identifier (8bit) + Length (16bit)
  - Msg-Length (8bit) + Message (0-255B)
  - code=2 (ACK), code=3 (NAK)
- **Identifier needed to match Request and Response**
- **Authenticate-Request or -Response may be lost**
  … so, Authenticator MUST permit multiple requests

CHAP

- **RFC-1994 “PPP Challenge Handshake Authentication Protocol (CHAP)” (Aug 1996)**
- **symmetric challenge (password-based)**
  - initial challenge compulsory (at channel creation)
  - authentication request optionally repeated (with a different challenge) during transmission – decision taken by the NAS
  - challenge MUST be a nonce
- **the Authenticators that support both PAP and CHAP *must* offer CHAP first**

CHAP: 3-way handshake protocol

- **(Authenticator > Peer) Challenge (code=1)**
  - Code (8bit) + Identifier (8bit) + Length (16bit)
  - Challenge-Size (8bit) + Challenge-Value (0-255B)
- **(Peer > Authenticator) Response (code=2)**
  - Code (8bit) + Identifier (8bit) + Length (16bit)
  - Response-Size (8bit) + Response-Value (0-255B)
- **(Authenticator > Peer) Result (code= 3 Success, 4 Failure)**
  - Code (8bit) + Identifier (8bit) + Length (16bit)
- **Response-Value = md5 (Identifier || pwd || Challenge-Value)**
- **Identifier needed to match Request and Response**
- **Challenge or Response may be lost … so, Authenticator MUST resend Challenge if no Response (until retry limit)**

MS-CHAP

- **MS-CHAPv1**
  - RFC 2433 "Microsoft PPP CHAP Extensions" (oct.1998)
  - dropped by MS starting with Windows Vista
- **MS-CHAPv2**
  - RFC 2759 "Microsoft PPP CHAP Extensions, v2" (jan.2000)
  - dropped by MS starting with Win11 22H2
- **LCP negotiates CHAP algorithm 0x80 (v1) or 0x81 (v2) for option 3 (Authentication Protocol)**
- **it's a MS-specific implementation of the CHAP concepts**
  - yet supported by many other vendors (e.g. CISCO)

MS-CHAP: extensions over CHAP

- **similar principles, different protocol**
- **common (v1 and v2):**
  - authenticator-controlled password change
  - authenticator-controlled authentication retry
  - specific failure codes
- **MS-CHAPv2 provides mutual authentication:**
  - by piggybacking a peer challenge on the Response packet
  - plus an authenticator response on the Success packet
- **each peer must know the plaintext password, or an MD4 hash of the password (thus not compatible with most password storage formats)**

The MS-CHAPv2 protocol

[PEER] **[AUTHENTICATOR]**

Hello >>

<< (Server Challenge, 16 byte) SC

(Client Challenge, 16 byte) CC

(Challenge-Hash) H = sha1(SC || CC || username)[0…7]

(NT-Hash) K = md4(password)

(Challenge-Response) R = des(K[0…6],H) || des(K[7…13],H) || des(K[14…20],H)

R + H + username >>

R1 ***R2*** ***R3***

decrypt R and verify if the result matches H

(NT-Hash-Hash) NHH = md4(md4(password))

(Digest) D = sha1(NHH || R || M1)

(Authentication-Response) A = sha1(D || H || M2)

<< A

compute A' and verify if it matches A

M1 = `"Magic server to client signing constant"`

M2 = `"Pad to make it do more than one iteration"`

MS-CHAPv2: divide-and-conquer (2012)

- **we have a known ciphertext-plaintext pair, R and H**
- **we need to find the three keys, K[0-6] K[7-13] K[14-20]**
- **brute-force the pwd? too much time …**
- **brute force the keys? 2^56 + 2^56 + 2^56 operations**
- **but K is just 128 bit (MD4 output) i.e. 16 B**
  - so K[14-20] has got only TWO bytes of K[14-15] padded with 0's
  - we need 2^16 operations to find K[14-20]
- **to find K[0-6] and K[7-13] we perform only 2^56 operations and then compare the result with R1 and R2**
- **2^56 DES operations (T<=23 hours with a specific FPGA)**
- **conclusion: MS-CHAPv2 should NEVER be used anymore**
  - finally cancelled in Win11 22H2 … but still used by many nets

EAP

- **RFC-3748 (extended by RFC-5247)**
  "PPP Extensible Authentication Protocol (EAP)"
- **a flexible L2 authentication framework**
- **authentication predefined mechanisms:**
  - MD5-challenge (similar to CHAP)
  - OTP
  - generic token card
- **other mechanisms may be added:**
  - RFC-2716 "PPP EAP TLS authentication protocol"
  - RFC-3579 "RADIUS support for EAP"

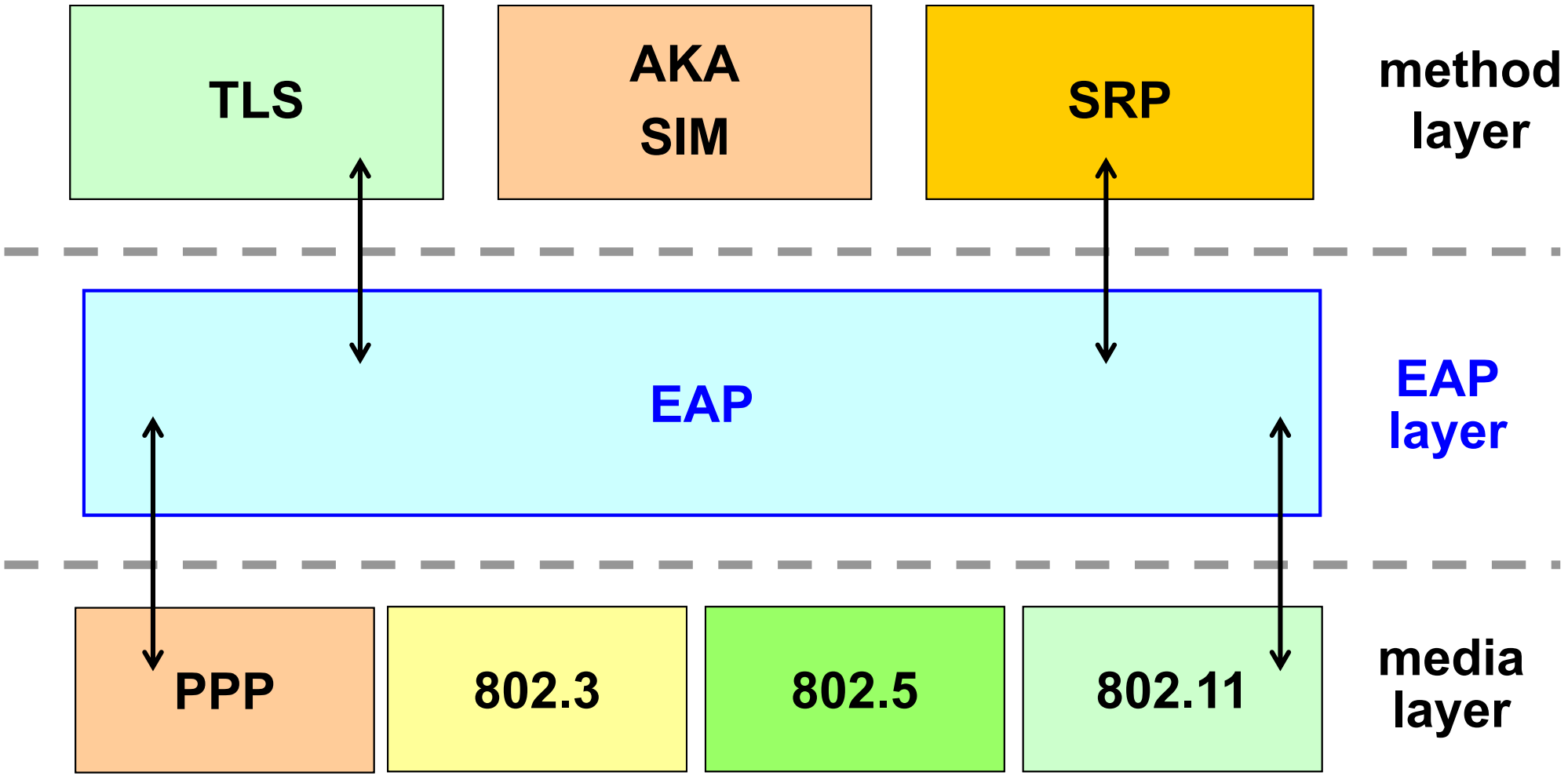
EAP - encapsulation

- **authentication data are transported via a specific EAP encapsulation protocol**
  - because L3 packets are not yet available …
- **features of EAP encapsulation:**
  - independent of IP
    - supports any link layer (e.g. PPP, 802, …)
  - explicit ACK/NAK (no windowing)
    - assumes no reordering (PPP guarantees ordering, UDP and raw IP do not!)
  - retransmission (max 3-5 retransmissions)
  - no fragmentation (must be provided by EAP methods for a payload greater than the minimum EAP MTU)

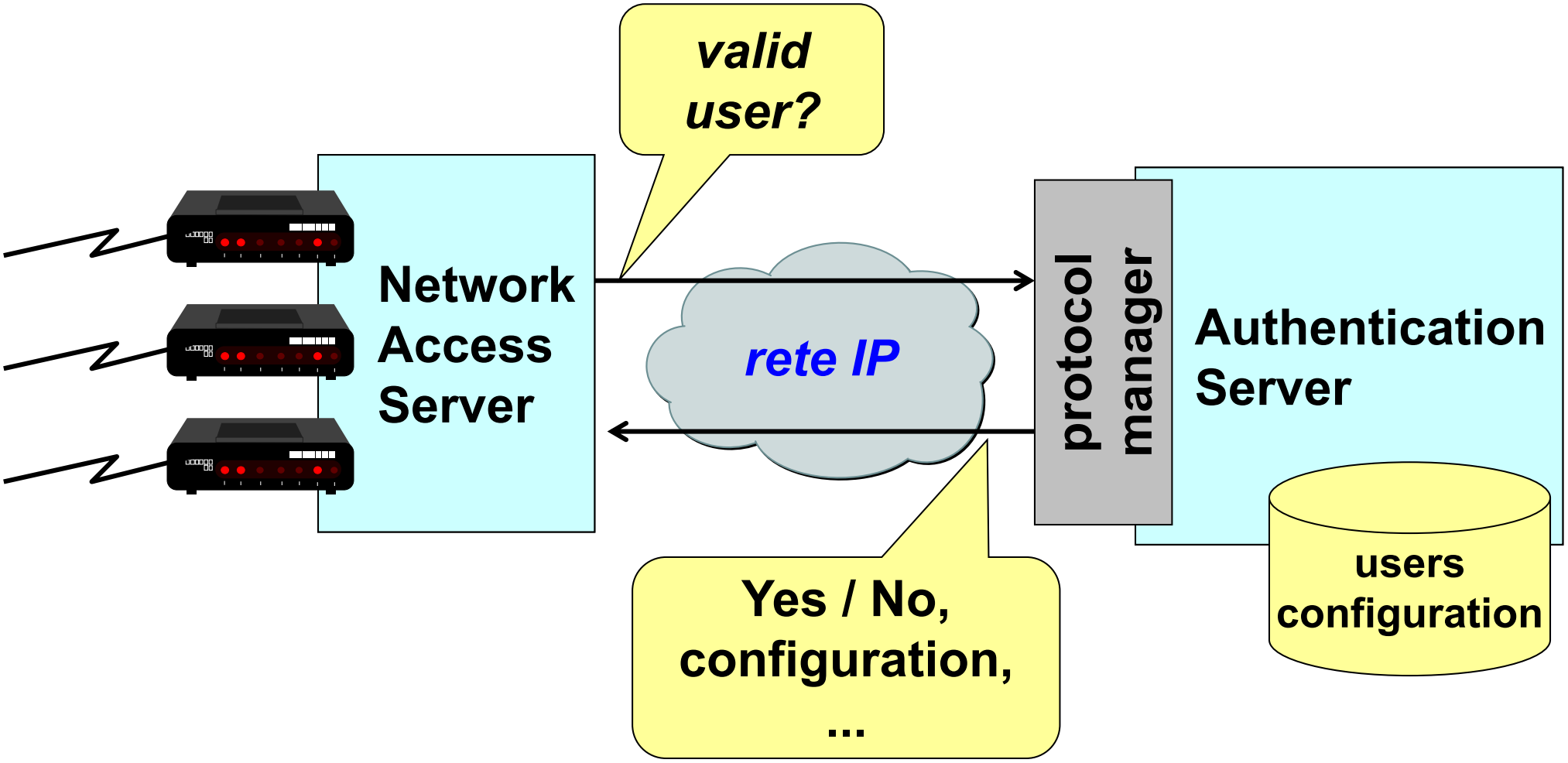
EAP

- **the link is not assumed to be physically secure**
  - EAP methods must provide security on their own
- **some EAP methods:**
  - EAP-TLS (RFC-5216) – TLS mutual authentication
  - EAP-MD5 (RFC-3748) – only EAP client authentication
  - EAP-TTLS = tunnelled TLS (to operate any authentication method protected by TLS, e.g. PAP, CHAP)
  - PEAP = TLS tunnel to protect an EAP method
  - EAP-SRP (Secure Remote Password)
  - GSS_API (includes Kerberos)
  - AKA-SIM (RFC-4186, RFC-4187)

EAP - architecture



Authentication for network access



AAA

- **the NAS manufacturers claim that security needs three functions:**
  - *Authentication* – entity's identity is authenticated based on credentials (e.g. password, OTP)
  - *Authorization* – determining whether an entity is authorized to perform a given activity or gain access to resources/services
  - *Accounting* – tracking network resource usage for audit support, capacity analysis or cost billing
- **the AS performs exactly these three functions talking with one or more NAS via one or more protocols**

Network authentication protocols

- **RADIUS**
  - the de-facto standard
  - proxy towards other AS
  - low security (pwd obfuscation, some authN/replay protection)
- **DIAMETER**
  - evolution of RADIUS
  - emphasis on roaming among different ISP
  - takes care of security (IPsec, TLS)
- **TACACS+ (TACACS, XTACACS)**
  - originally technically better than RADIUS, achieved smaller acceptance because it was a proprietary solution (Cisco)

RADIUS

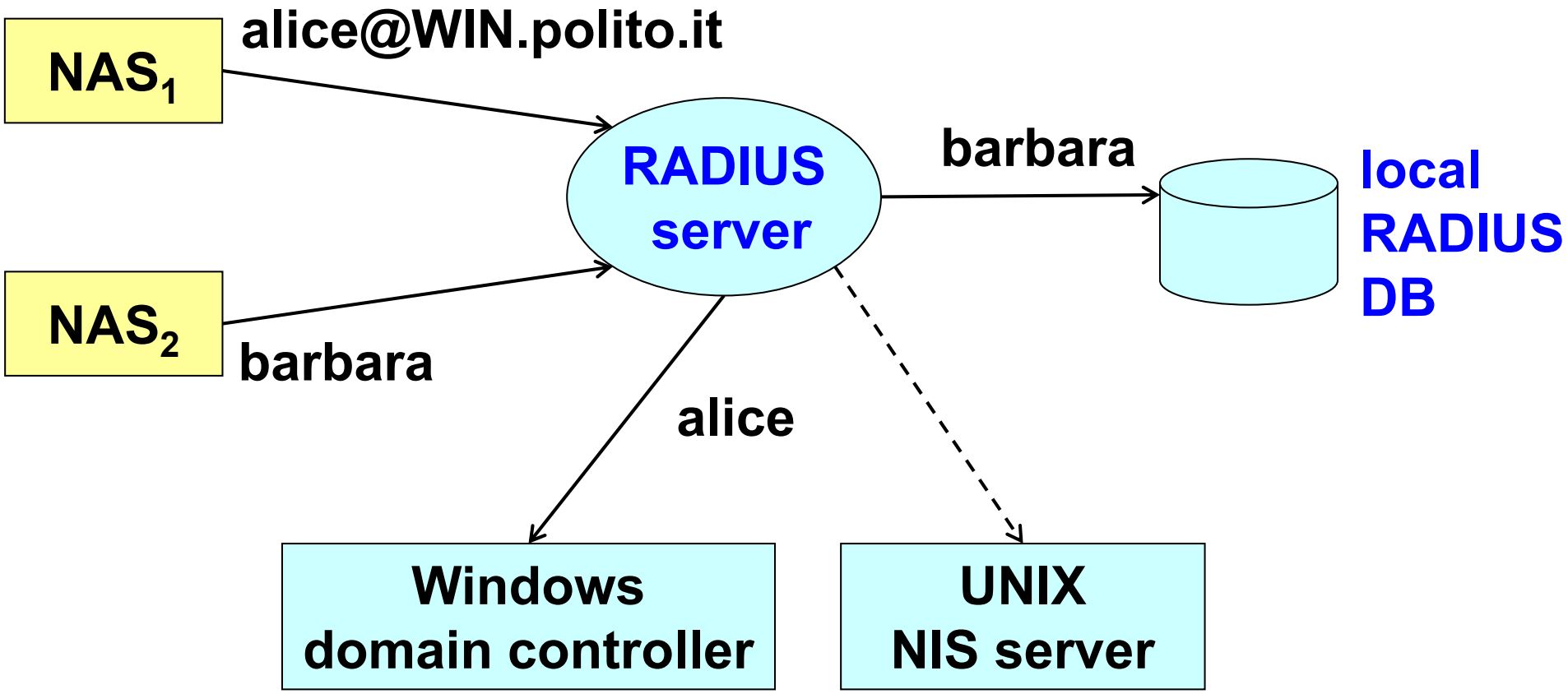
- **Remote Authentication Dial-In User Service**
- **Livingston Technologies (1991) then IETF**
- **supports authentication, authorization and accounting to control network access:**
  - physical ports (analogical, ISDN, IEEE 802)
  - virtual ports (tunnel, wireless access)
- **centralized administration and accounting**
- **client-server schema between NAS and AS**
  - port 1812/UDP (authentication) and 1813/UDP (accounting) ; unofficial ports: 1645 & 1646/UDP
  - timeout + retransmission
  - secondary server

RADIUS - RFC

- RFC-2865 (protocol)
- RFC-2866 (accounting)
- RFC-2867/2868 (tunnel accounting and attributes)
- RFC-2869 (extensions)
- RFC-3579 (RADIUS support for EAP)
- RFC-3580 (guidelines for 802.1X with RADIUS)

RADIUS proxy

- the RADIUS server may act as a proxy towards other authentication servers



Which security functionalities for Radius?

- **sniffing NAS req (if contains pwd)**
  - confidentiality, privacy
- **fake AS resp (to block valid or allow invalid user)**
- **changing AS resp (Y > N or N > Y)**
  - authN & integrity of AS resp
- **replay of AS resp (if not properly tied to NAS req)**
  - anti-replay of AS resp
- **pwd enumeration (from fake NAS)**
  - authN of NAS req
- **DoS (many NAS req from fake NAS)**
  - server scalability

RADIUS: data protection

- **packet integrity and authentication via keyed-MD5:**
  - key = shared-secret
  - client without key are ignored
- **password transmitted "encrypted" with MD5 (after padding with NUL bytes to a multiple of 128 bit):**

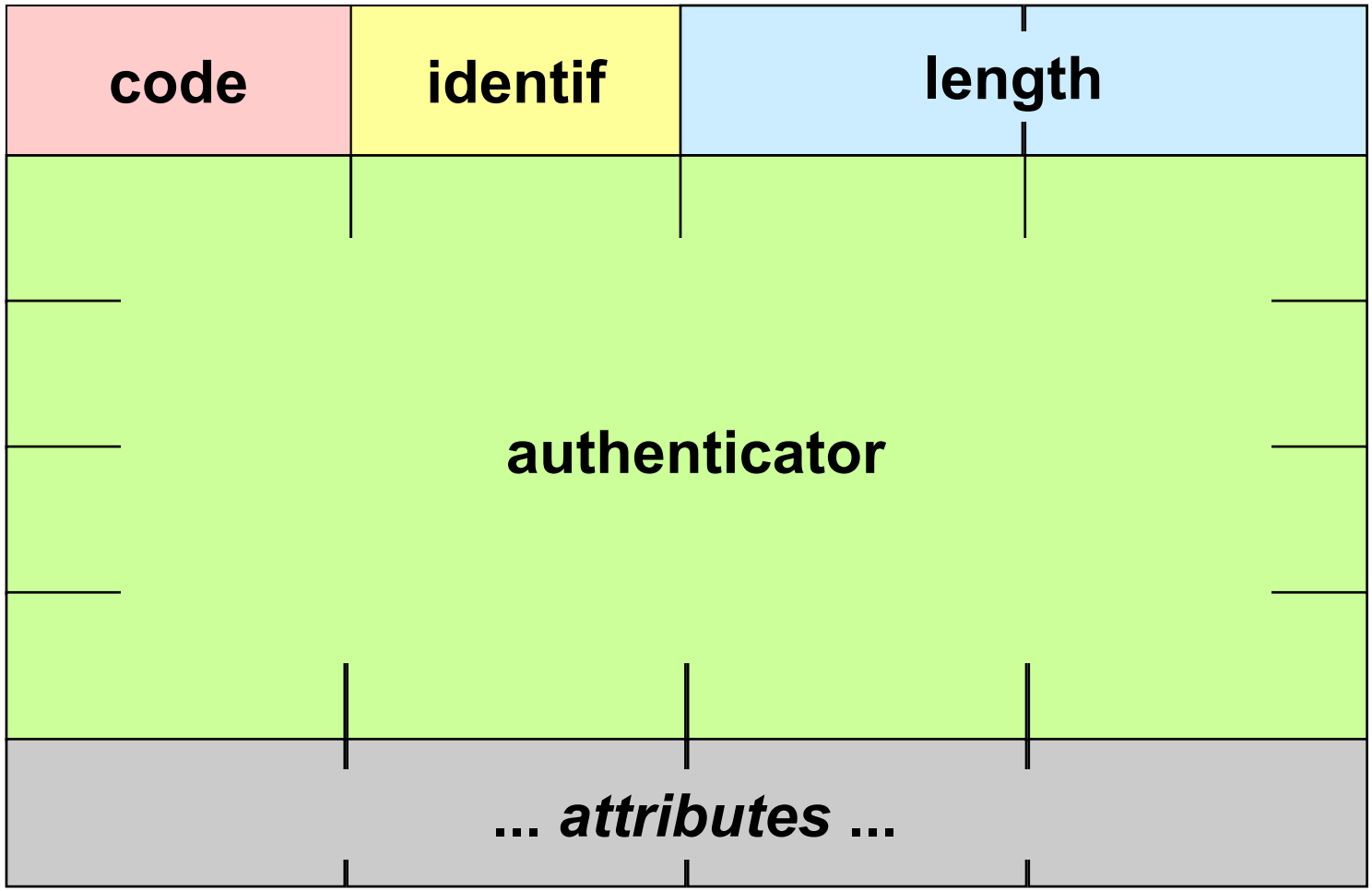
password ⊕ md5(key+authenticator)

RADIUS

- **user authentication via PAP, CHAP, token-card and EAP**
  - CISCO provides a free server for CryptoCard
  - others support SecurID
- **attributes in TLV form, easily extensible without modification to installed base (by ignoring any unknown Type):**

  attribute type – length – value

RADIUS - format



RADIUS – packet types

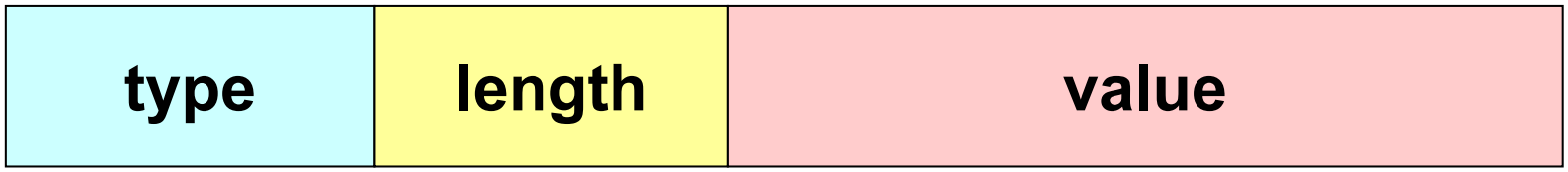
- **ACCESS-REQUEST**
  - contains access credentials (e.g. username + pwd)
- **ACCESS-REJECT**
  - access is denied (e.g. due to bad username/pwd)
- **ACCESS-CHALLENGE**
  - requests additional info from the user (e.g. a PIN, token code, secondary password)
- **ACCESS-ACCEPT (*parameters*):**
  - access is granted + network parameters are given
    - for SLIP/PPP: Framed-Protocol, Framed-IP-Address, Framed-IP-Netmask, MS-primary-DNS-server, MS-Secondary-DNS-server, …
    - for terminal: host, port

RADIUS - authenticator

- **double purpose:**
  - server reply authentication and no replay
  - masking the password
- **in Access-Request:**
  - it is named Request Authenticator
  - 16 byte randomly generated by the NAS
- **in Access-Accept / Reject / Challenge**
  - it is named Response Authenticator
  - it is computed via a keyed-digest:

md5(code || ID || length || RequestAuth || attributes || secret)

RADIUS - some attributes

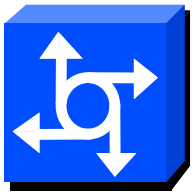


- **type = 1 (User-Name)**
  - value = text, network access identifier (NAI), DN
- **type = 2 (User-Password)**
  - value = password ⊕ md5 (key || RequestAuthent.)
- **type = 3 (Chap-Password)**
  - value = user CHAP response (128 bit)
- **type = 60 (CHAP-Challenge)**
  - value = challenge from the NAS to the user

NAI (Network Access Identifier)

- RFC-2486
- NAI = username [@ realm]
- all devices must support NAI up to 72 byte long
- the exact syntax for username and realm is in the RFC (note that only ASCII characters < 128 are allowed, but all of them are allowed)
- note that the username is the one used in the PPP authentication phase (does not necessarily match the application username)

Example - CHAP + RADIUS



RADIUS server

CHAP / Challenge-Request

CHAP / Challenge-Response

RADIUS / Access-Request:
- CHAP-Username
- CHAP-Challenge
- CHAP-Password

RADIUS / Access-Accept:
- parameters, …

CHAP / Success

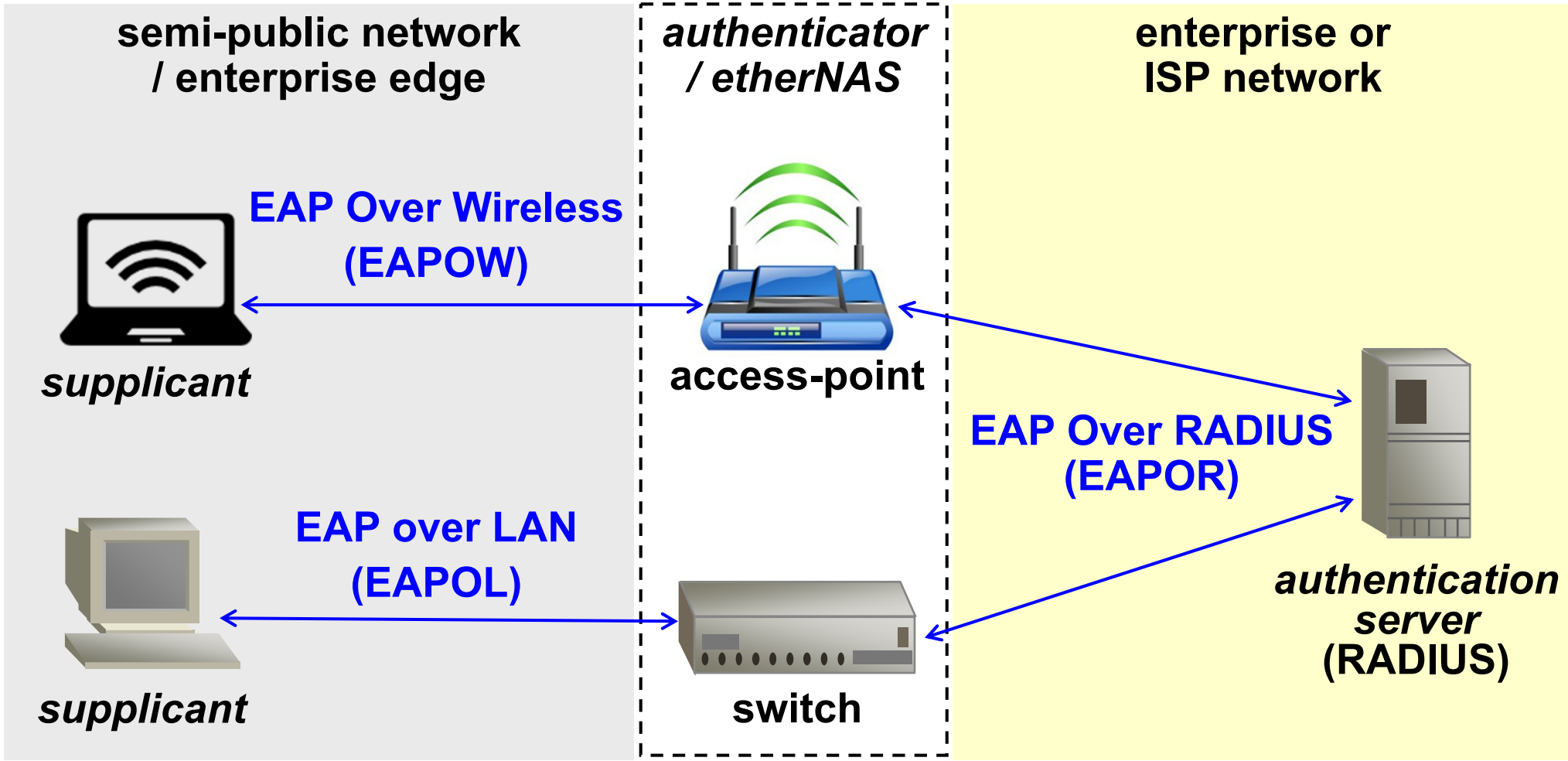
IPCP

IEEE 802.1x

- **Port-Based Network Access Control:**
  - L2 authentication architecture
  - useful in a wired network to block access
  - absolutely needed in wireless networks
- **first implementations (long ago):**
  - Windows-XP and Cisco wireless access-points

http://standards.ieee.org/getieee802/download/802.1X-2001.pdf

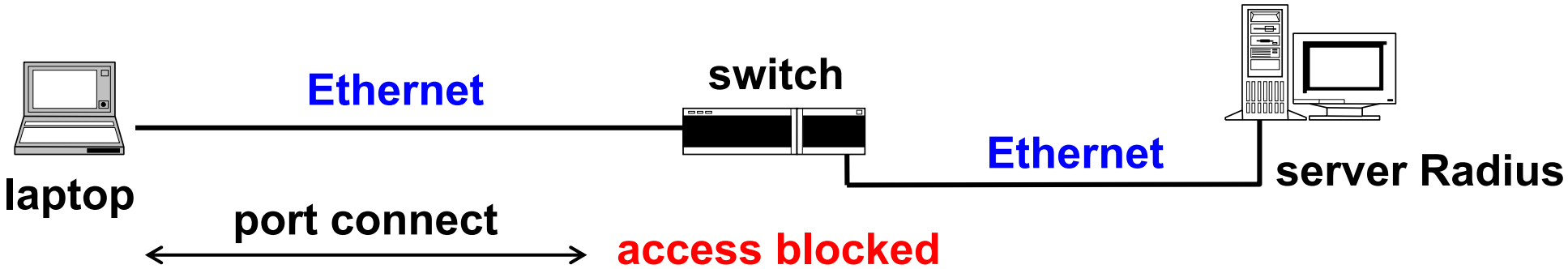
802.1x - architecture

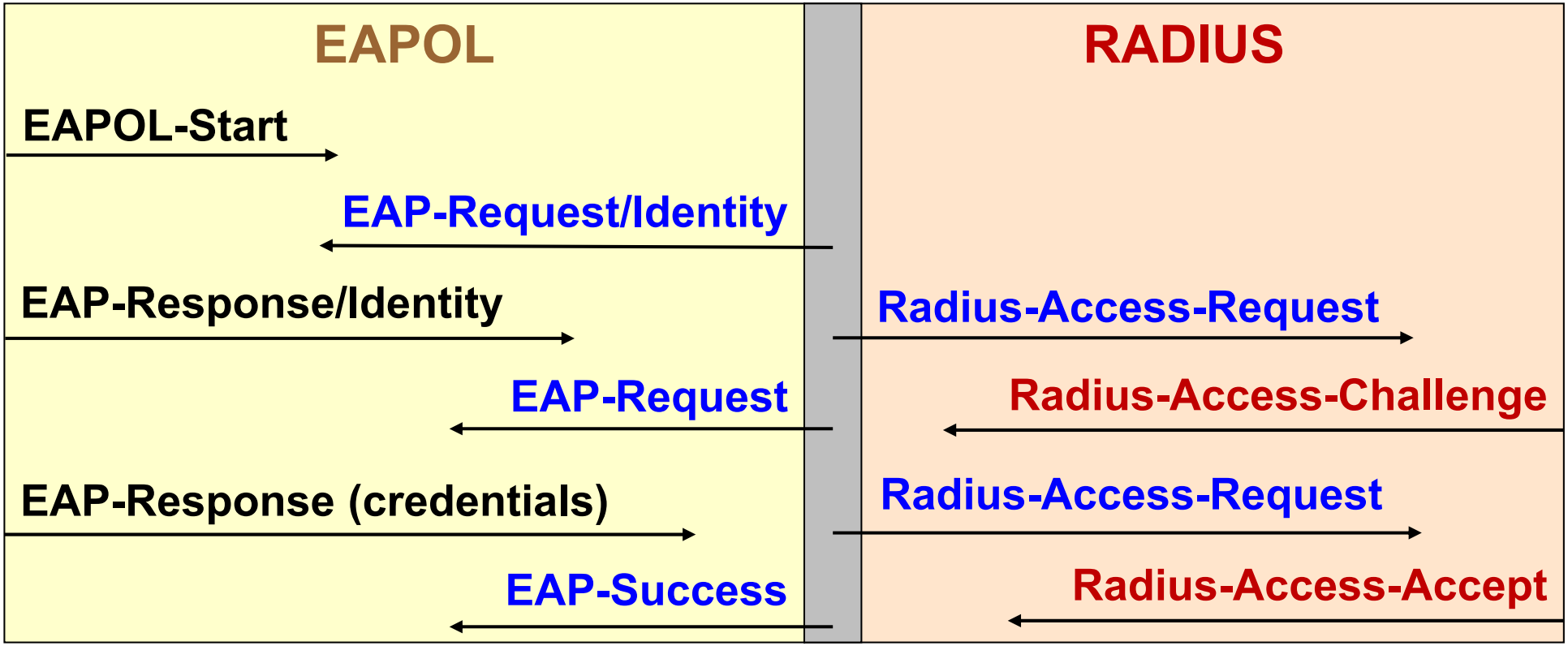


IEEE 802.1x

- **authentication and key-management framework**
- **supplicant authentication:**
  - with EAP methods, end-to-end supplicant-AS
- **key management (for supplicant and etherNAS):**
  - may derive session keys for use in packet authentication, integrity, and confidentiality
  - standard algorithms for key derivation (e.g. TLS, SRP, …)
- **exploits the application level for the actual implementation of the security mechanisms**
  - direct dialogue between supplicant and AS
    - NIC and NAS operate as “pass-through device”
  - no change needed on NIC and NAS for new mechanisms

802.1x - messages

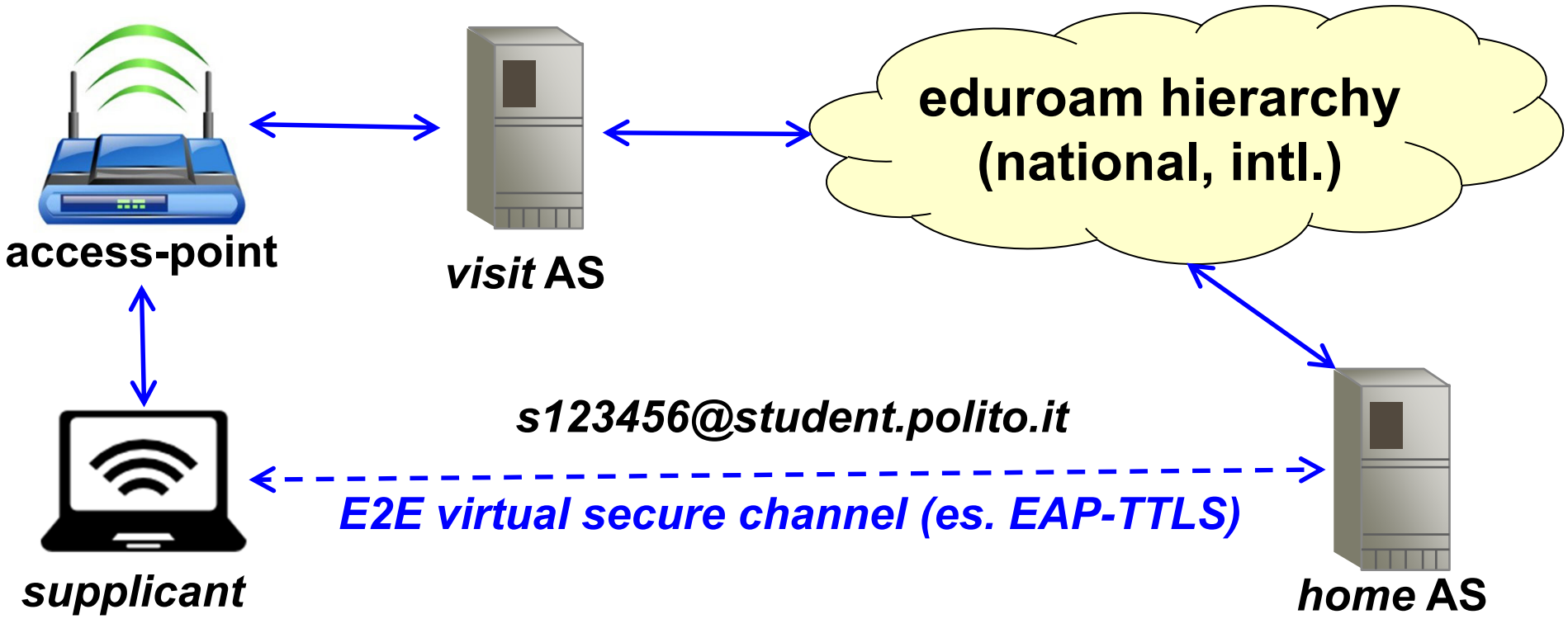




access allowed

eduroam

- **WiFi access at research institutes (Italy, Europe, …) and other places (e.g. some airports)**
  - (10/11/2024) 106 countries
  - uses 802.1x + RADIUS federation



Which is the best OSI level to implement security?

Application
Presentation
Session
Transport
Network
Data Link
Physical

firewall? IPSEC?
smart-card?
encryption box?
guards?

Optimal level?

- the upper we go in the stack, the more specific are the security functions (e.g. it's possible to identify the user, commands, data) and independent from the underlying network … but we leave more room for DoS attacks

- the lower we go in the stack, the more quickly we can "expel" the intruders … but the fewer the data for the decision (e.g. only the MAC or IP addresses, no user identification, no commands)

DHCP (in)security

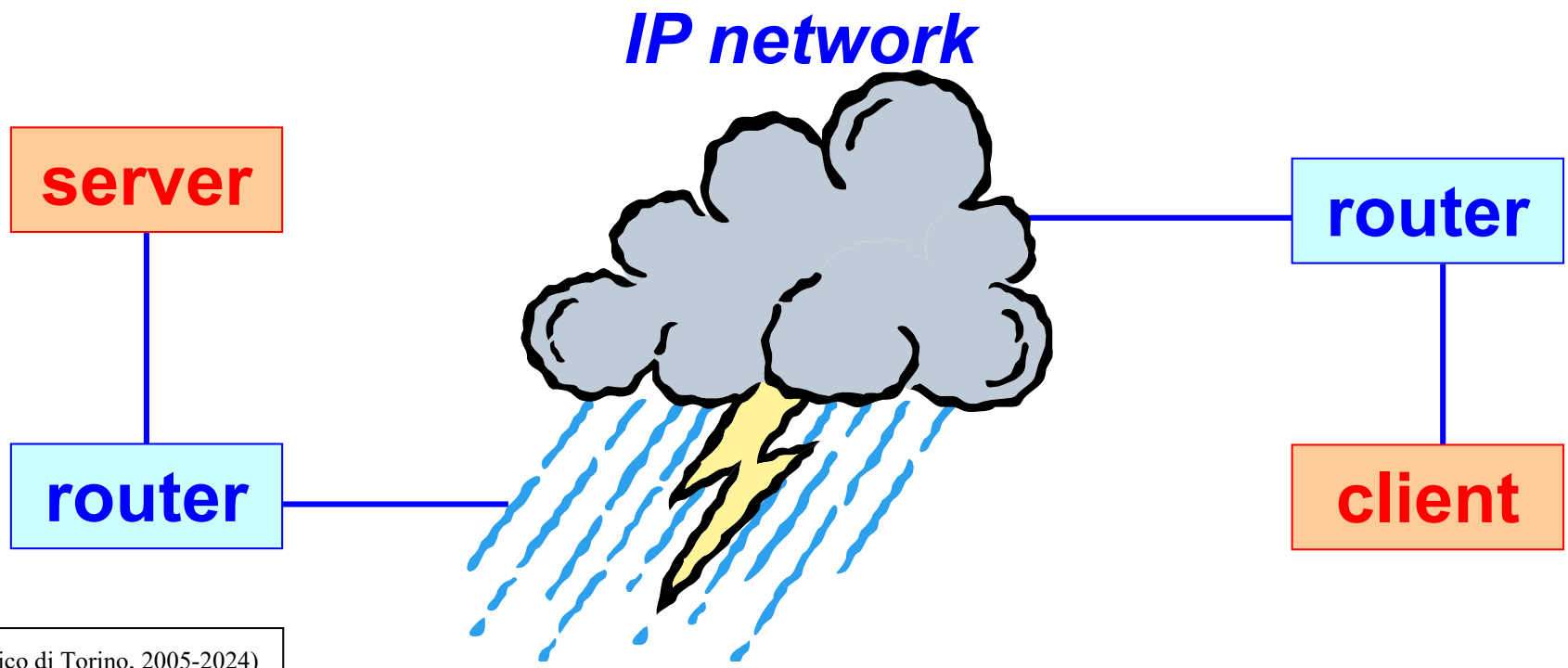
- **non-authenticated (!!) broadcast (!) protocol providing:**
  - IP address, netmask, default gateway
  - local nameserver, local DNS suffix
- **activation of a fake DHCP server is trivial**
  - because the DHCP request is L2 broadcast
- **possible attacks from the fake DHCP server:**
  - denial-of-service
    - provides a wrong network configuration
  - MITM
    - provide configuration with 2-bit subnet + gw equal attacker
    - if we activate NAT, we can intercept the replies too
  - malicious name-address translation (e.g. for phishing, pharming)

DHCP protection

- **some switches (e.g. Cisco) offer:**
  - DHCPsnooping = only transmit replies from “trusted ports”
  - IP guard = switching only IPs got from a DHCP server (but there is a limit to the number of recognized addresses)
- **RFC-3118 “Authentication for DHCP messages”**
  - use of HMAC-MD5 to authenticate the messages
  - problem = key distribution and management (shared key!)
  - rarely adopted

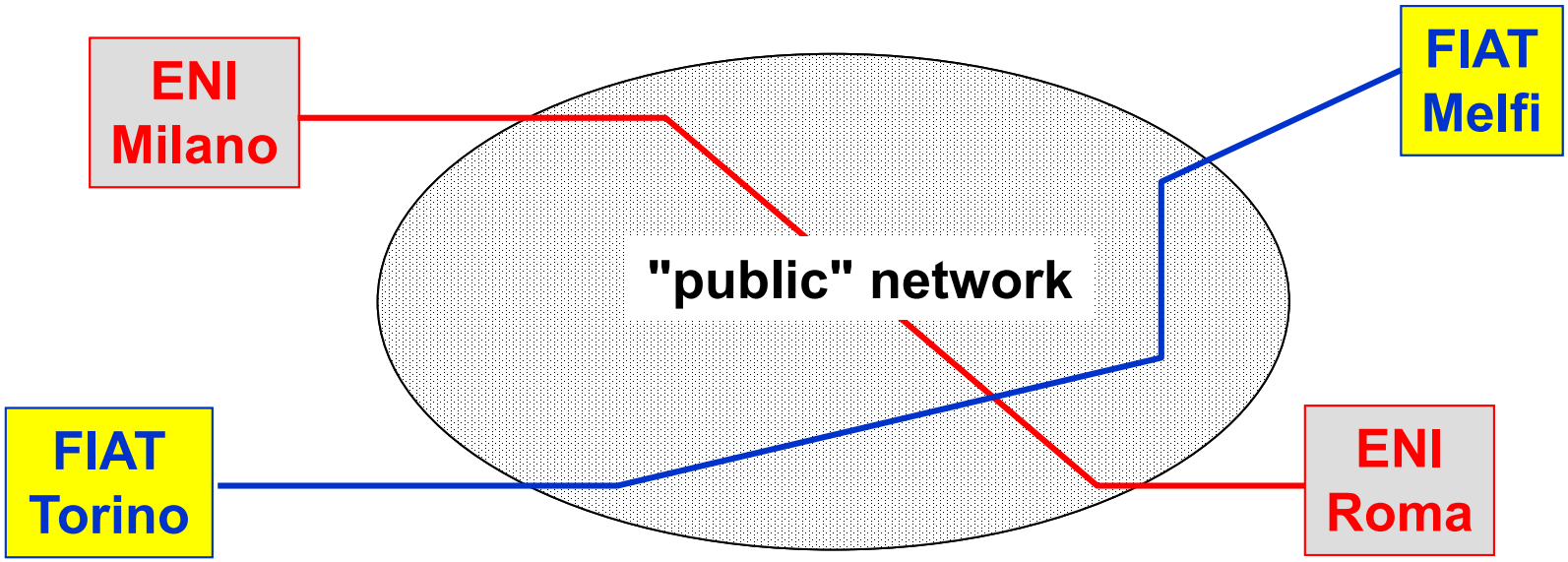
Security at network level (L3)

- end-to-end protection for L3-homogeneous networks (e.g. IP networks)
- creation of VPN (Virtual Private Network)



What is a VPN?

- a technique (hardware and/or software) to create a private network ...
- ... while using shared (or anyway untrusted) channels and transmission devices



Techniques to create a VPN

- via private addressing
- via protected routing (IP tunnel)
- via cryptographic protection of the network packets (secure IP tunnel)

1. VPN via private addresses

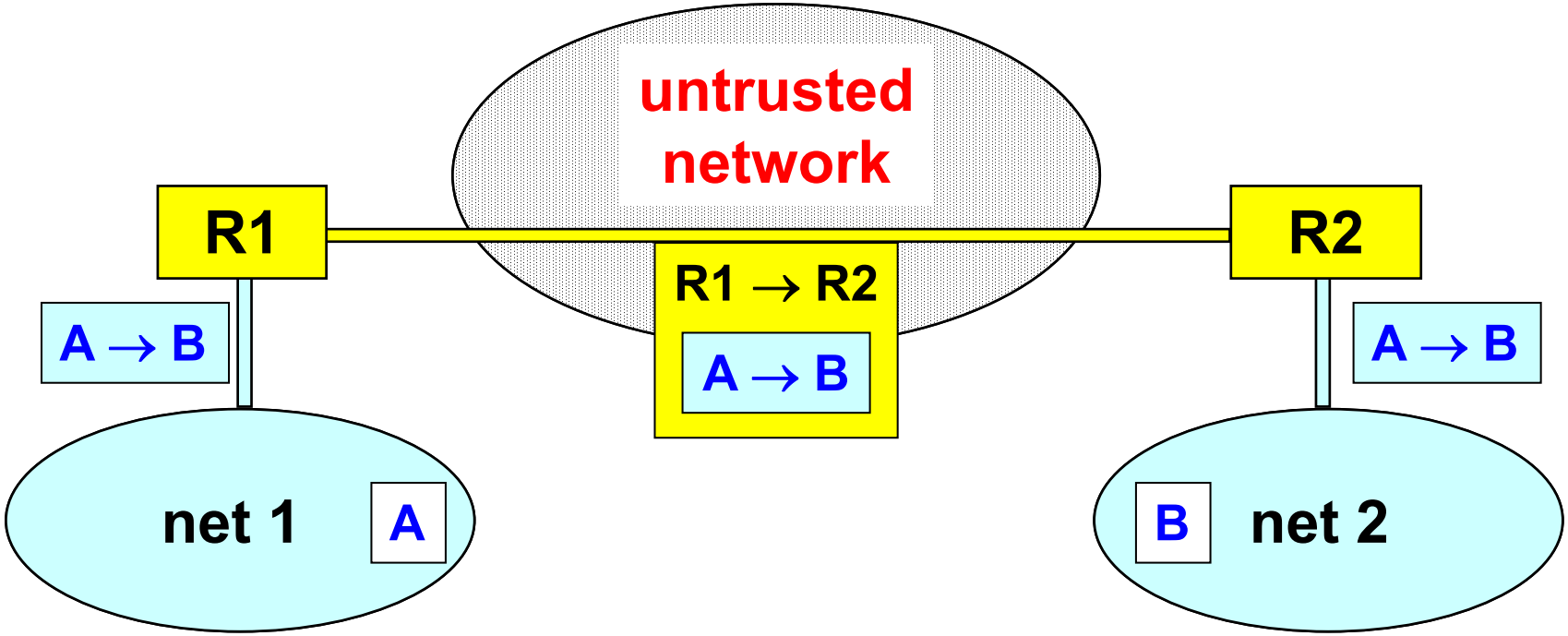
- **the networks to be part of the VPN use non-public addresses so that they are unreachable from other networks (e.g. private IANA networks as per RFC-1918)**

- **this protection can be easily defeated if somebody:**
  - guesses or discovers the addresses
  - can sniff the packets during transmission
  - has access to the communication devices

2. VPN via tunnel

- **the routers encapsulate whole L3 packets as a payload inside another packet**
  - IP in IP
  - IP over MPLS
  - other (e.g. IP over TLS)
- **the routers perform access control to the VPN by ACL (Access Control List)**
- **this protection can be defeated by anybody that manages a router or can sniff the packets during transmission**

VPN via IP tunnel



IPv4 header (end-to-end)	TCP/UDP header + data

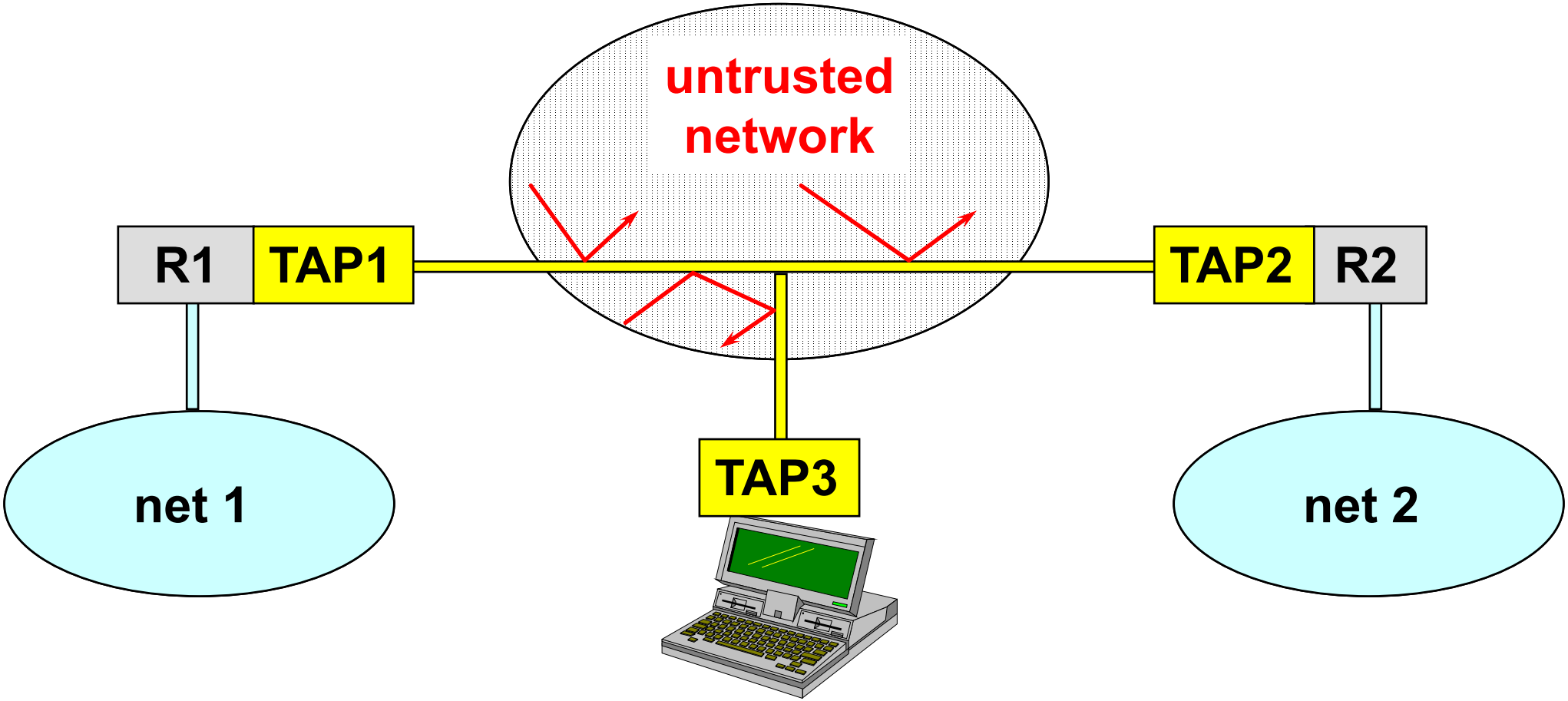
IPv4 header (tunnel)	IPv4 header (end-to-end)	TCP/UDP header + data

3. VPN via secure IP tunnel

- **before encapsulation, the packets are protected with:**
  - MAC (integrity + authentication)
  - encryption (confidentiality)
  - numbering (to avoid replay)

- **if the cryptographic algorithms are strong, then the only possible attack is to stop the communications**
- **also known as S-VPN (Secure VPN)**

VPN via secure IP tunnel



IPsec

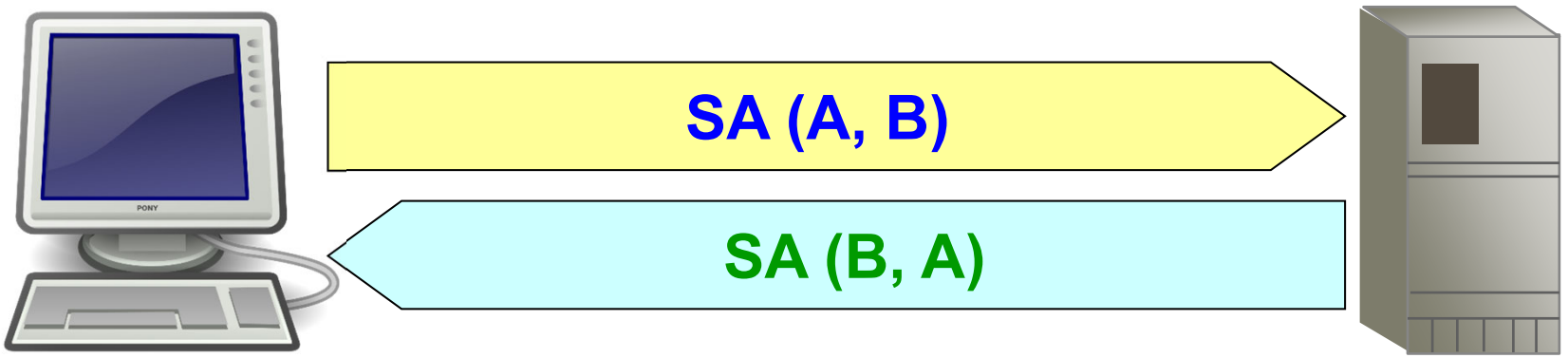
- **IETF architecture for L3 security in IPv4 / IPv6:**
  - to create S-VPN over untrusted networks
  - to create end-to-end secure packet flows
- **definition of two specific packet types:**
  - AH (Authentication Header)
    for integrity, authentication, no replay
  - ESP (Encapsulating Security Payload)
    for confidentiality (+AH)
- **protocol for key exchange:**
  - IKE (Internet Key Exchange)

IPsec security services

- **authentication of IP packets:**
  - computation of a keyed-digest with a shared key
  - provides:
    - data integrity and sender authentication
    - (partial) protection against “replay” attacks as the packet contains a sequence number
- **confidentiality of IP packets:**
  - payload encryption with a symmetric algo and a shared key
  - provides data privacy
- **peer authentication when creating the SA:**
  - key agreement after authN (shared key or digital signature)

IPsec Security Association (SA)

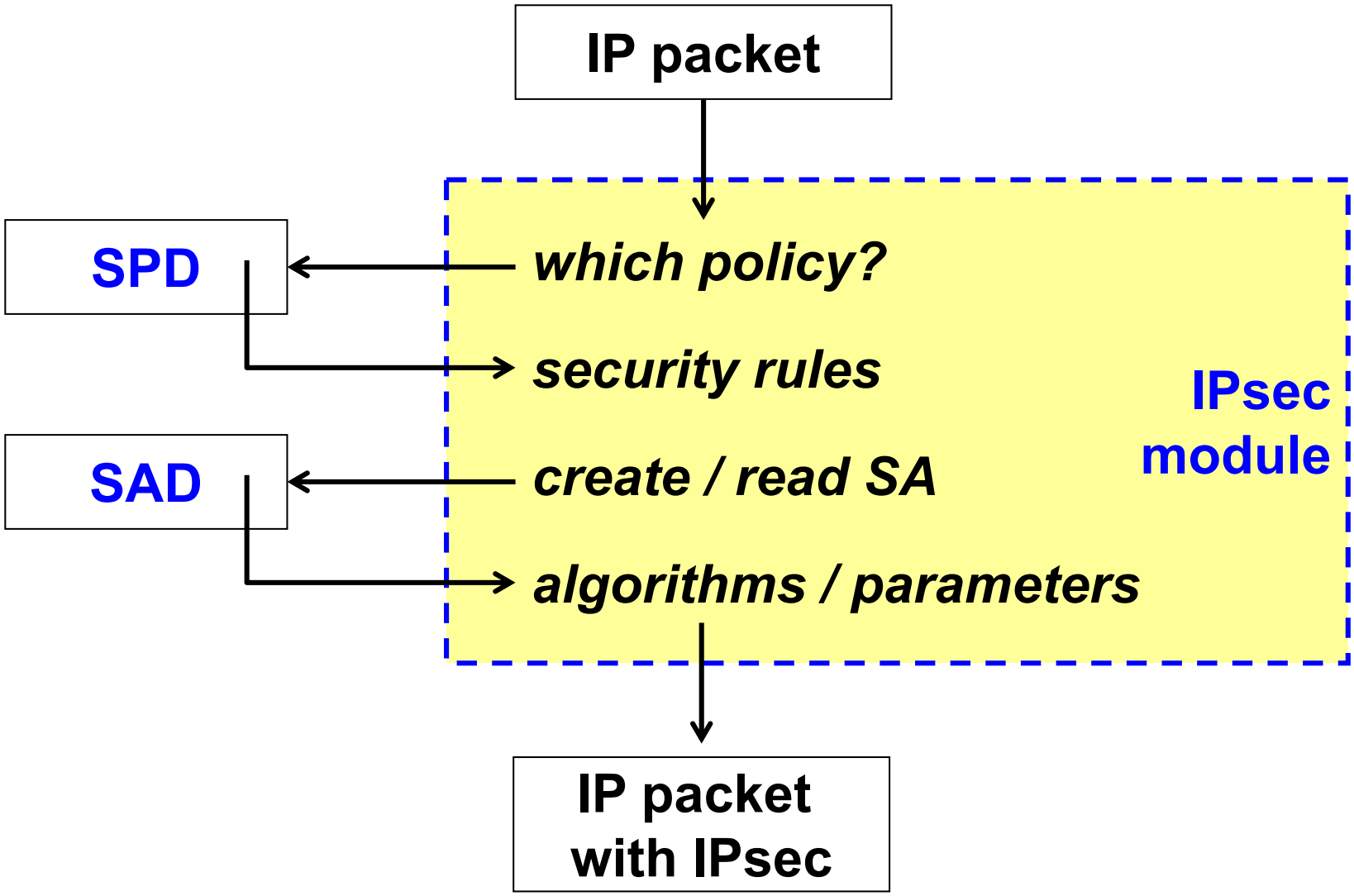
- unidirectional logic connection between two IPsec systems
- each SA has associated different security services
- two SA are needed to get complete protection of a bidirectional packet flow



IPsec local database

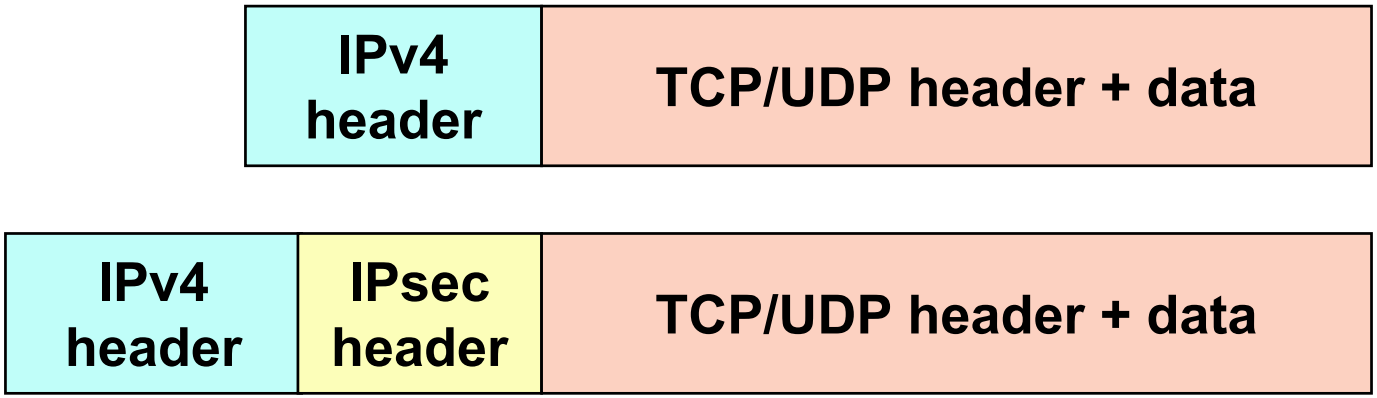
- **SPD (Security Policy Database)**
  - list of security policy to apply to the different packet flows
  - a-priori configured (e.g. manually) or connected to an automatic system (e.g. ISPS, Internet Security Policy System)
- **SAD (SA Database)**
  - list of active SA and their characteristics (algorithms, keys, parameters)

How IPsec works (sending)



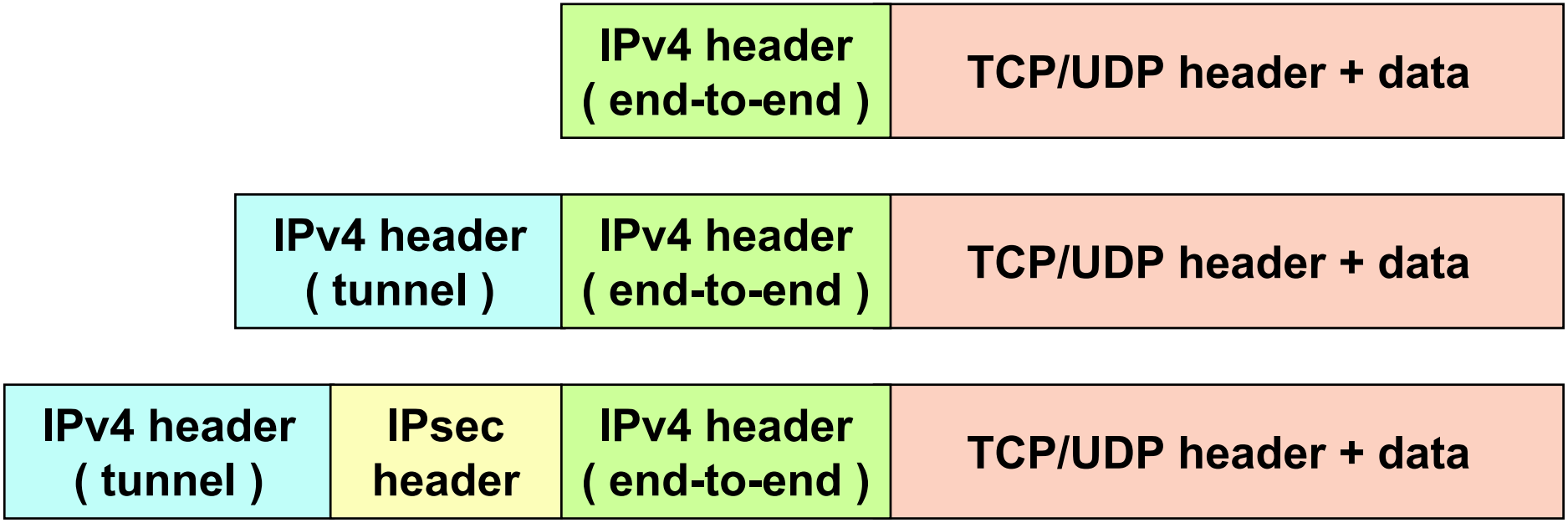
Transport mode IPsec

- used for end-to-end security, that is used by hosts, not gateways (exception: traffic for the gateway itself, e.g. SNMP, ICMP)
- pro: computationally light
- con: no protection of header variable fields



Tunnel mode IPsec

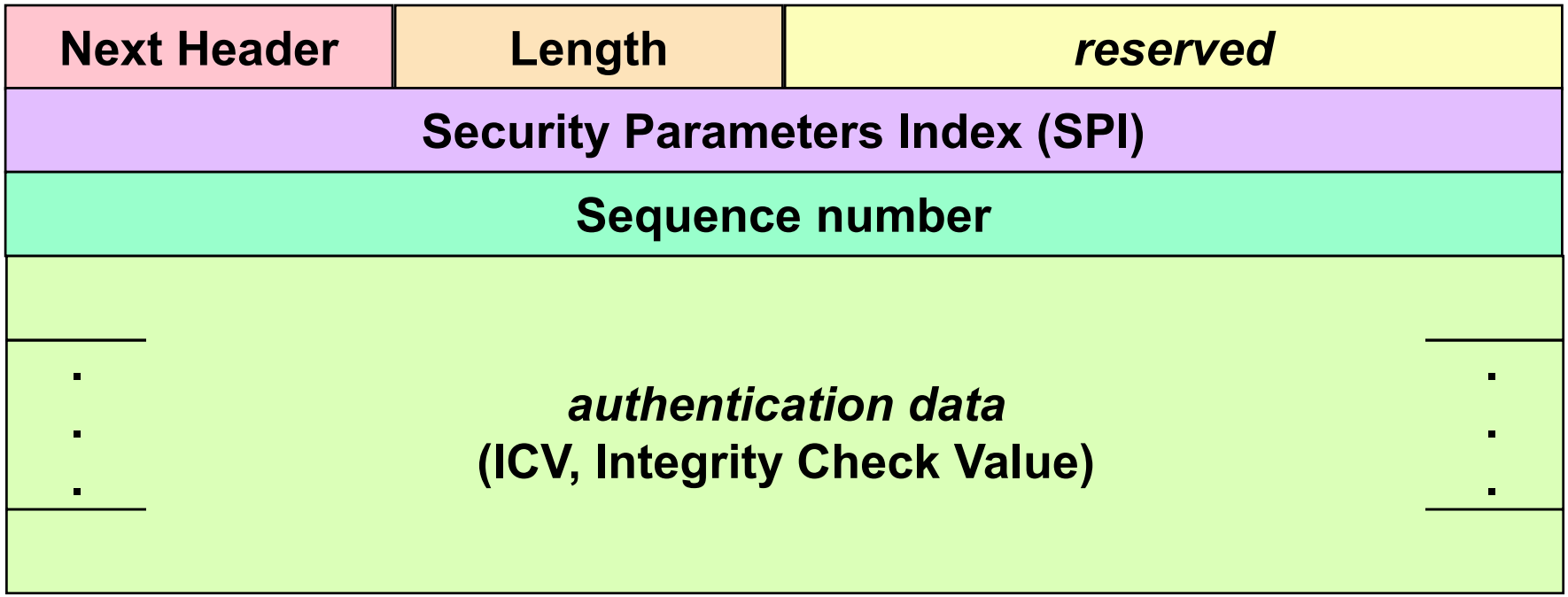
- used to create a VPN, usually by gateways
- pro: protection of E2E header variable fields
- con: computationally heavy

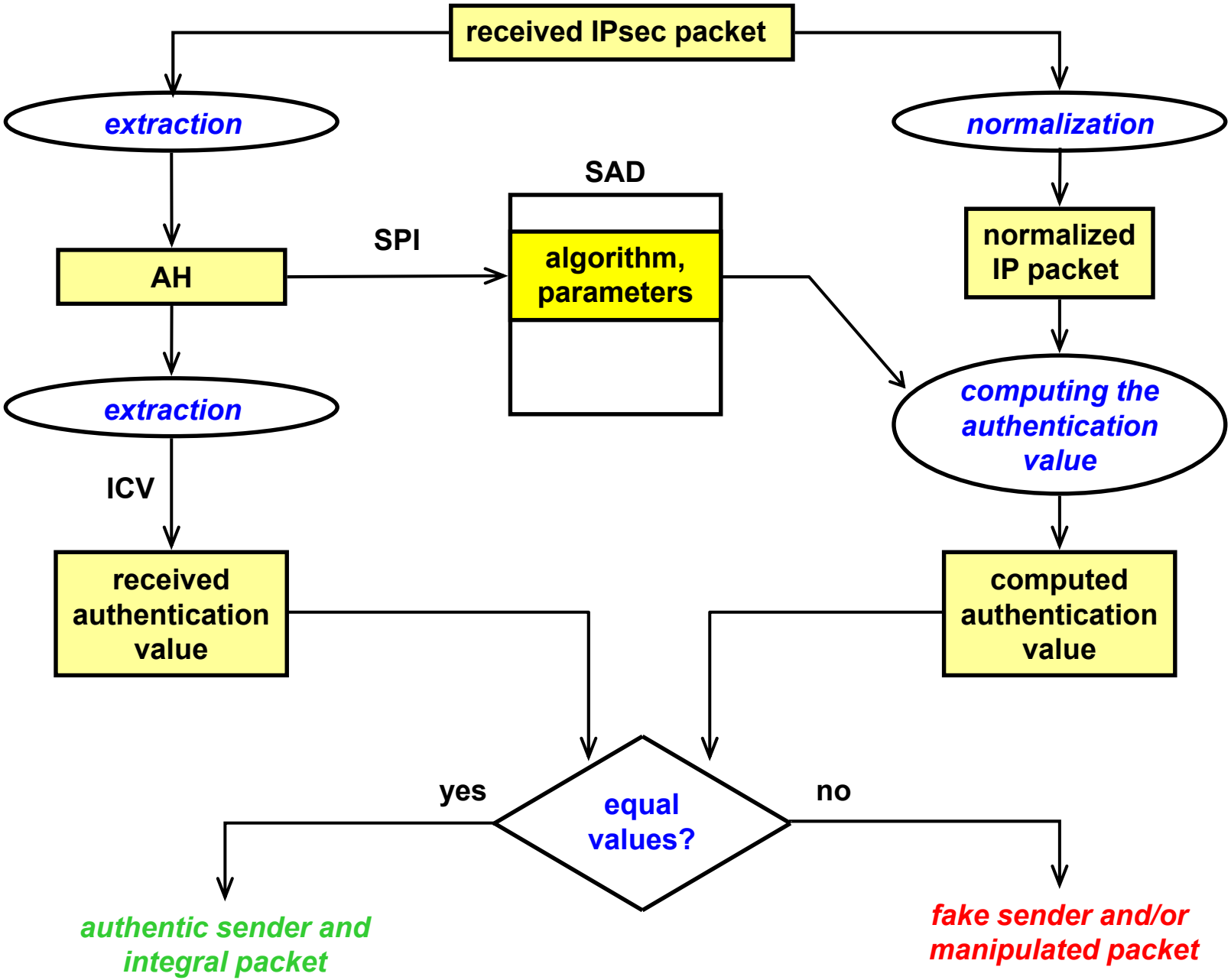


AH

- **Authentication Header**
- **mechanism (first version, RFC-1826):**
  - data integrity and sender authentication
  - compulsory support of keyed-MD5 (RFC-1828)
  - optional support of keyed-SHA-1 (RFC-1852)
- **mechanism (second version, RFC-2402):**
  - data integrity, sender authentication and (partial) protection from replay attack
  - HMAC-MD5-96
  - HMAC-SHA-1-96

AH - format (RFC-4302)





HMAC-SHA1-96

- given M normalize it to generate M'
- compute the authentication base:
  B = HMAC-SHA1 (K, M')
- ICV = 96 leftmost bits of B

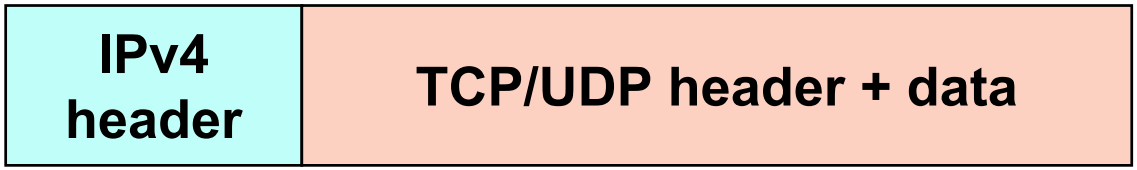
- in general, for any truncated MAC:
  - (pro) less information for the attacker
  - (con) less bits to predict for the attacker
  - (do) never truncate to less than half of the hash size (to match the birthday attack length)
  - (do) never truncate to less than 80 bit (too short)

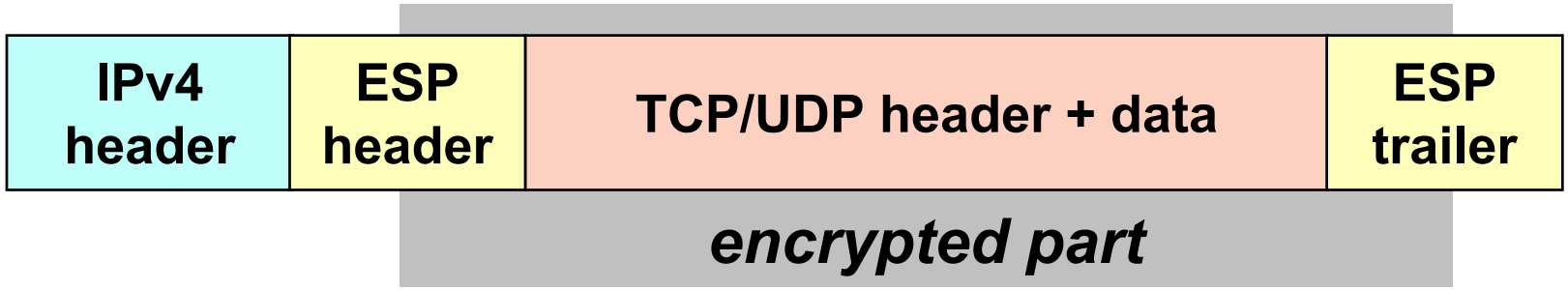
ESP

- **Encapsulating Security Payload**
- **first version (RFC-1827) gave only confidentiality**
- **base mechanism: DES-CBC (RFC-1829)**
- **other mechanisms possible**
- **second version (RFC-2406):**
  - provides also authentication (but the IP header, so the coverage is not equivalent to that of AH)
  - the packet dimension is reduced and one SA is saved

ESP in transport mode

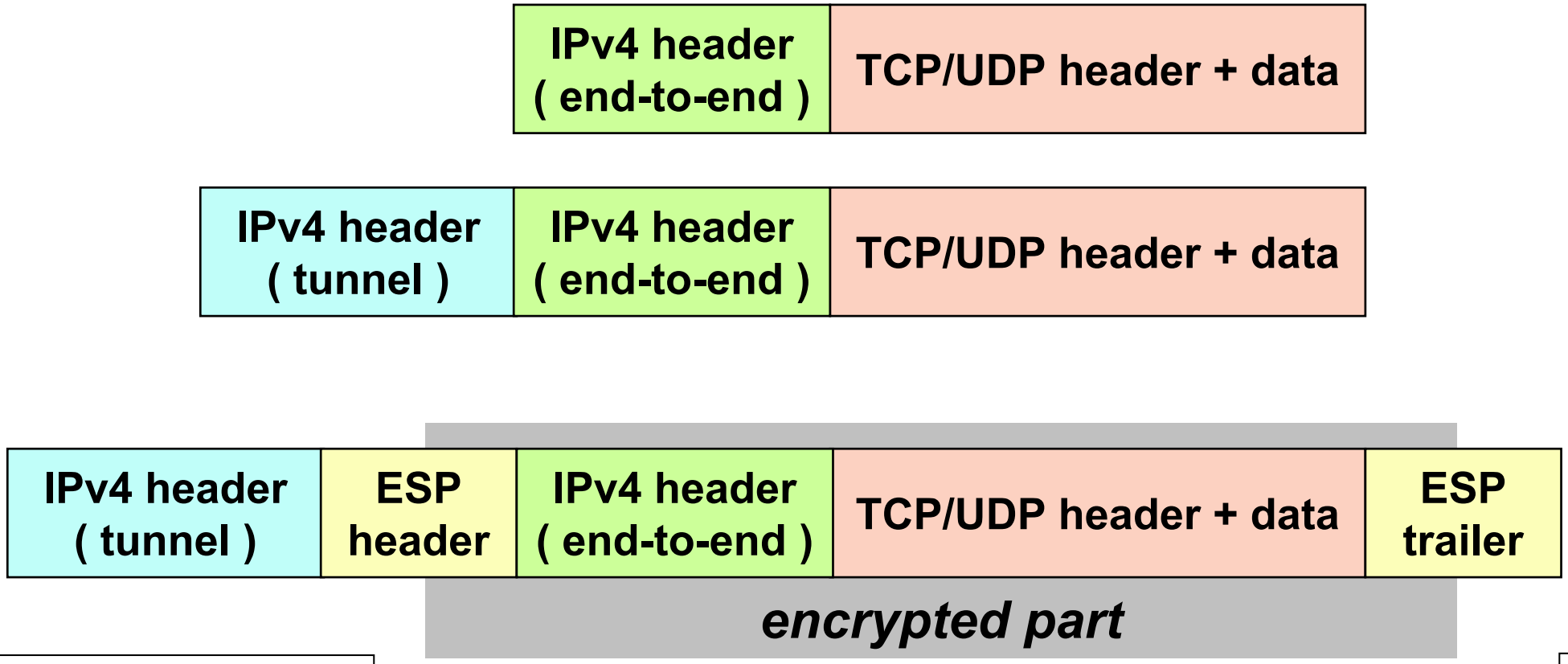
- pro: the payload is hidden (including info needed for QoS, filtering, or intrusion detection!)
- con: the header remains in clear





ESP in tunnel mode

- pro: hides both the payload and (original) header
- con: larger packet size

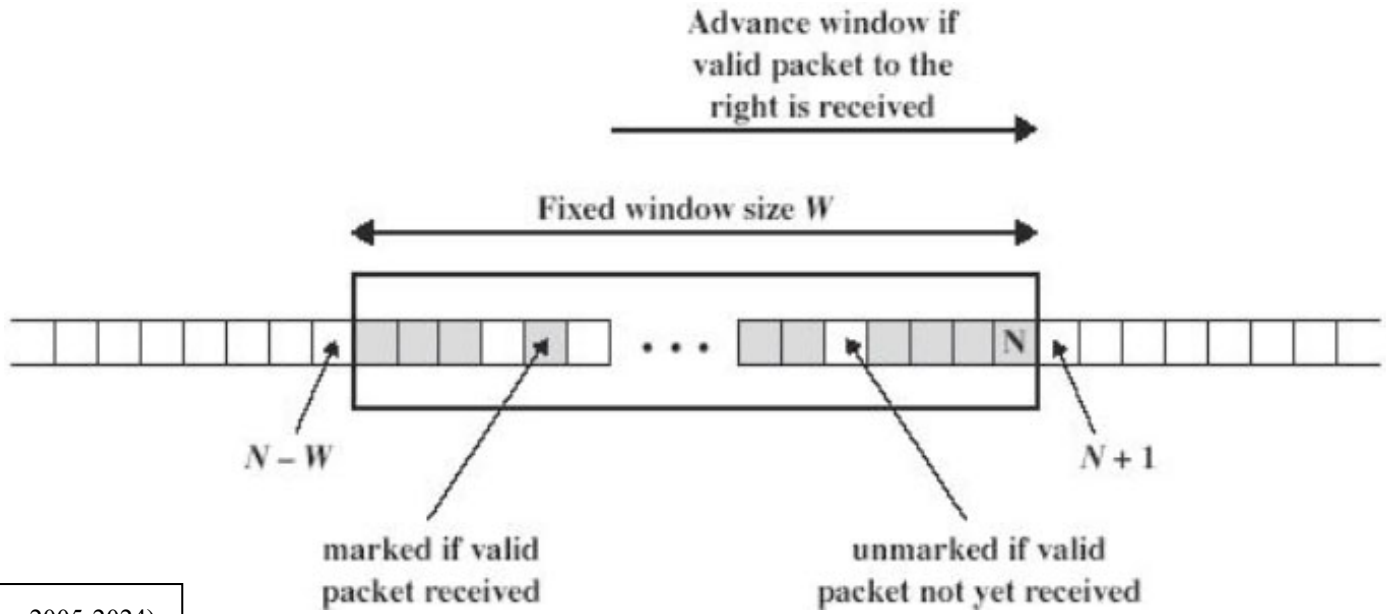


IPsec implementation details

- **UI crypto-suites (RFC-4308) for interoperability**
  - VPN-A = ESP/3DES-CBC/HMAC-SHA1-96
  - VPN-B = ESP/AES-128-CBC/AES-XCBC-MAC-96
- **NULL algorithms for ESP:**
  - for authentication or privacy (but not simultaneously!)
  - protection vs. performance trade-off
- **sequence number:**
  - (partial) protection from replay
  - minimum window of 32 packets (64 suggested)
  - creation mandatory for the sender but verification optional for the receiver (announced during SA negotiation)

IPsec replay protection

- at SA creation, sender initializes sequence number to 0
- when sending a packet, increment the sequence number
- when the sequence number 2^{32}-1 is reached, a new SA should be negotiated
- moving window: outside it, no replay protection



IPsec v3

- **AH is optional, ESP mandatory**
- **support for single source multicast**
- **ESN (Extended Sequence Number):**
  - 64 bit (but only the 32 least significant ones are transmitted)
  - default when using IKEv2, but it's usage must be explicitly negotiated
- **support for authenticated encryption (AEAD)**
- **clarifications about SA and SPI (for faster lookup)**

IPsec v3 – algorithms (RFC-4305)

- **for integrity and authentication:**
  - (MAY) HMAC-MD5-96
  - (MUST) HMAC-SHA-1-96
  - (SHOULD+) AES-XCBC-MAC-96
  - (MUST) NULL (only for ESP)
- **for privacy:**
  - (MUST) NULL
  - (MUST–) TripleDES-CBC
  - (SHOULD+) AES-128-CBC
  - (SHOULD) AES-CTR
  - (SHOULD NOT) DES-CBC

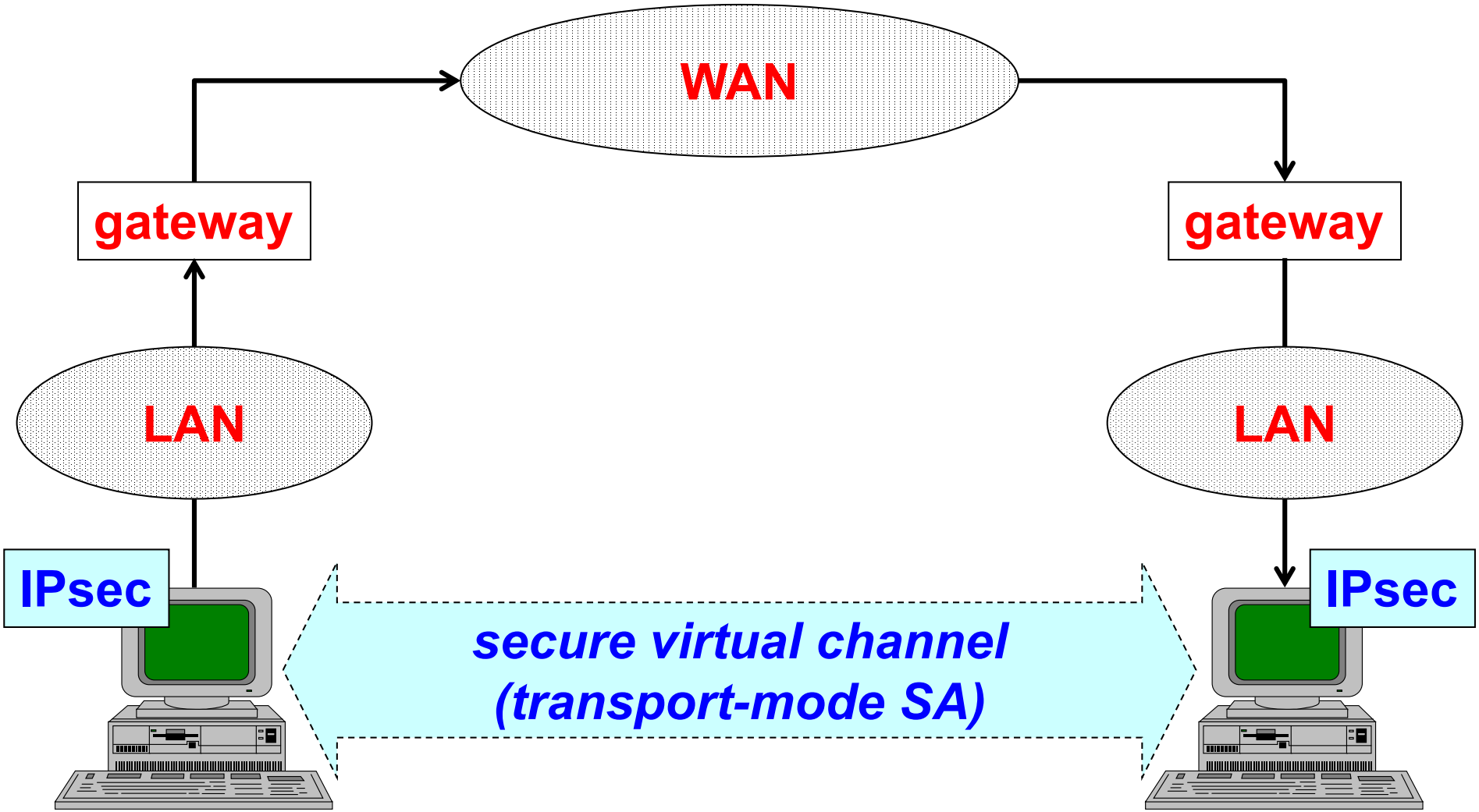
IPsec v3 – other algorithms

- **for authenticated encryption (AEAD mode):**
  - AES-CCM
  - AES-CMAC
  - ChaCha20 w/ Poly1305
- **for authentication and integrity:**
  - HMAC-SHA-256-128
  - HMAC-SHA-384-192
  - HMAC-SHA-512-256

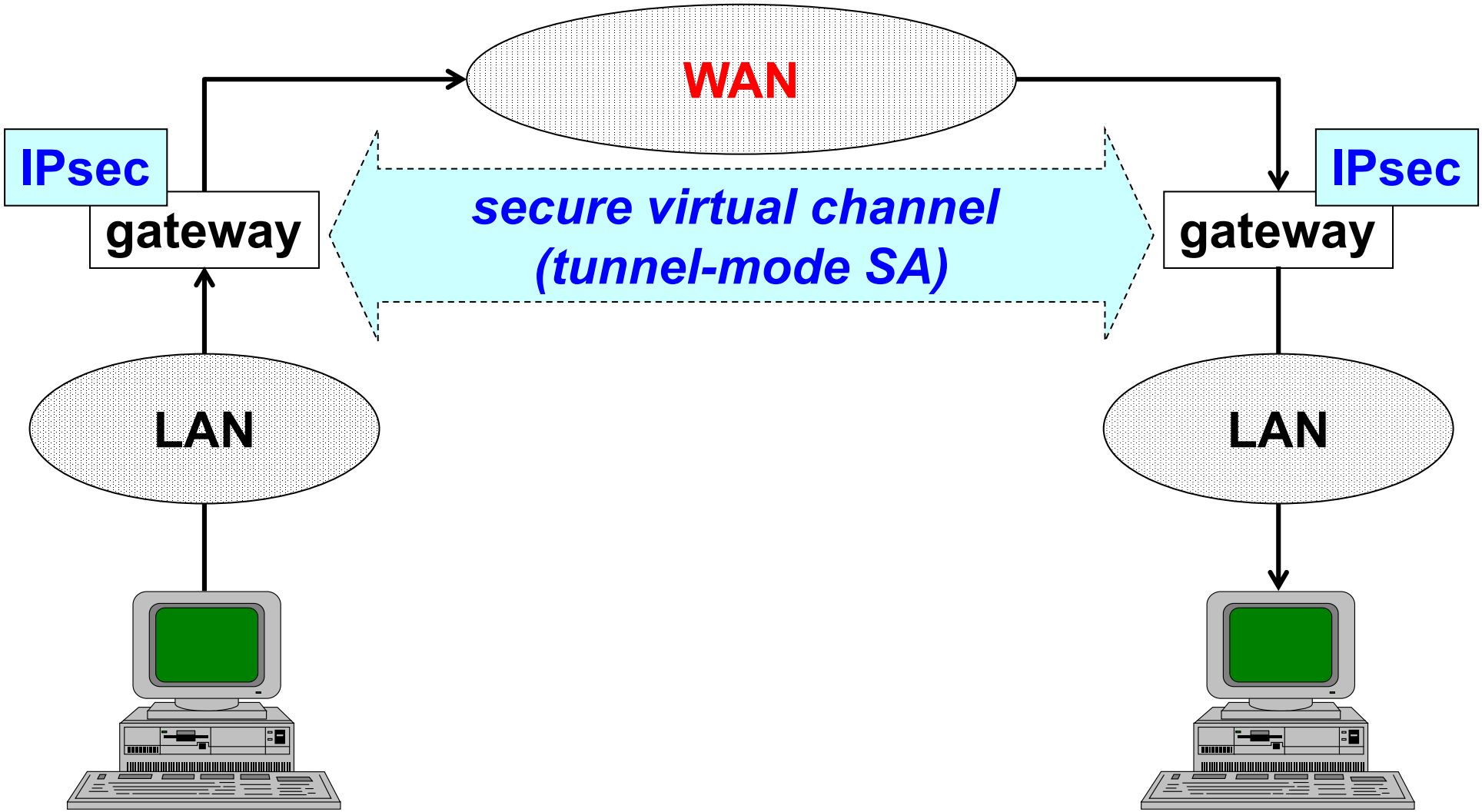
IPsec v3 – TFC

- **TFC (Traffic Flow Confidentiality) padding in ESP**
  - after the payload and before the normal padding
  - the receiver must be able to compute the original size of the payload (e.g. possible with IP, UDP, and ICMP payloads)
- **support for "dummy packets" (next header 59)**
  - useful only if encrypted …

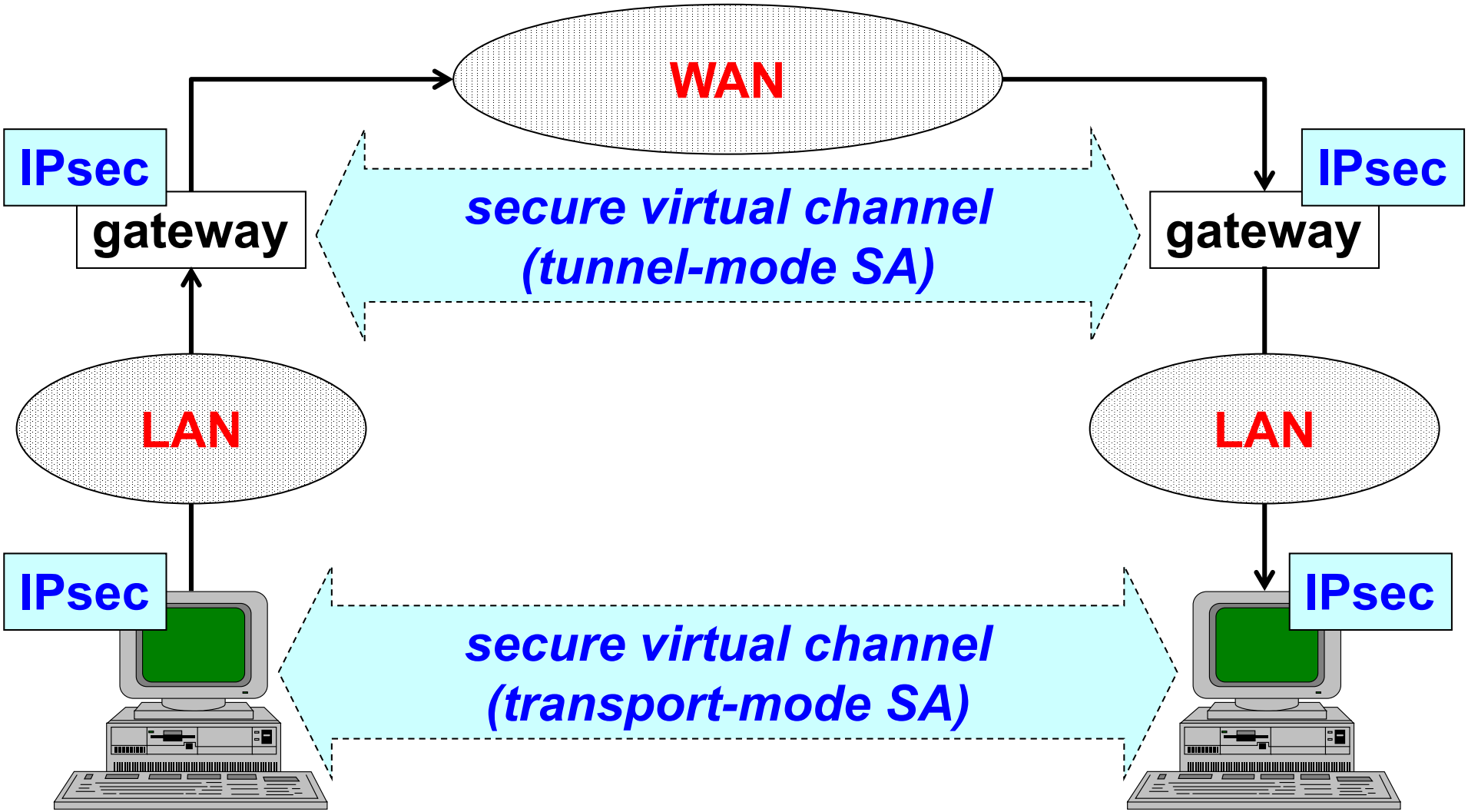
End-to-end security



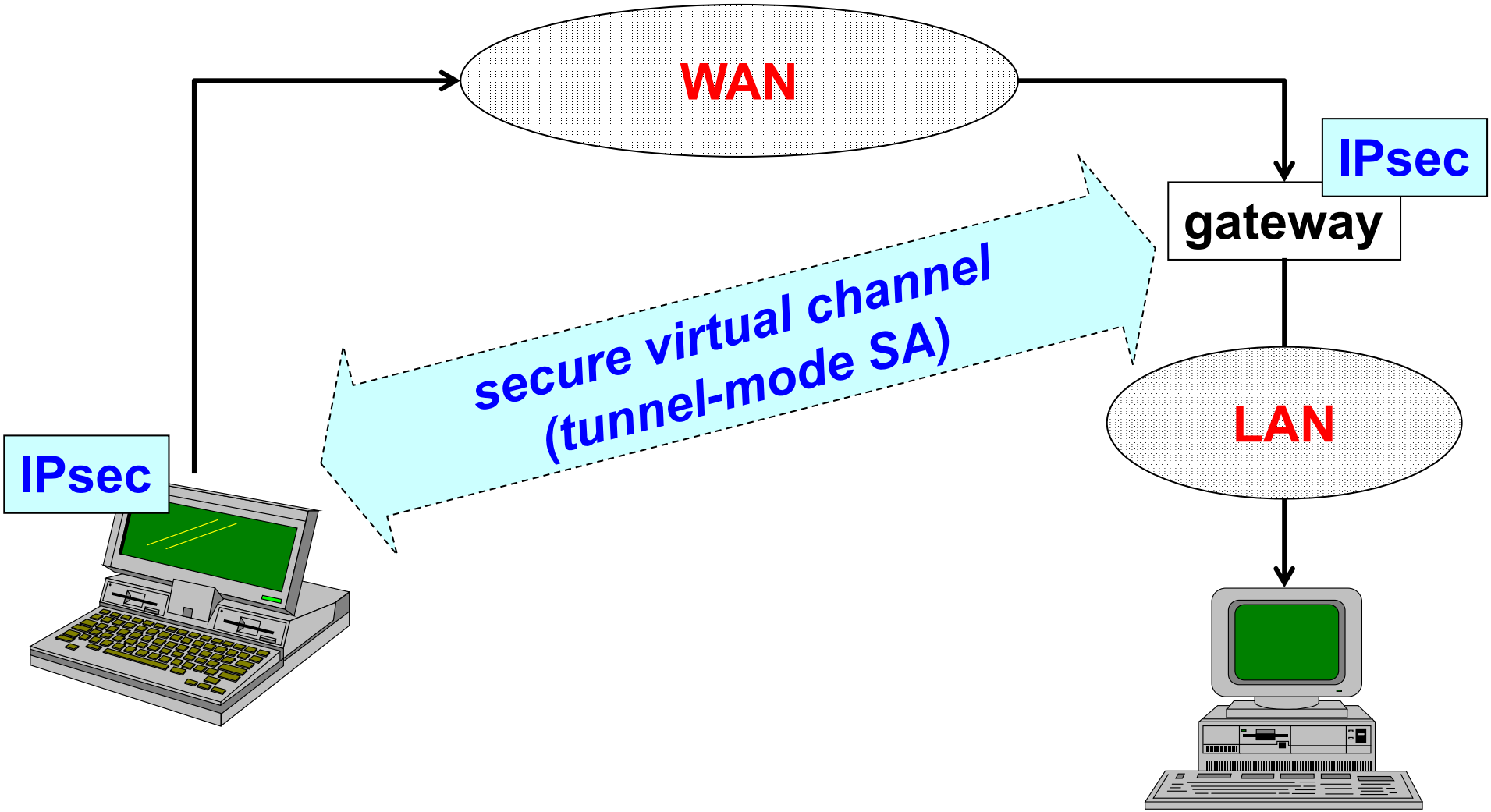
Basic VPN



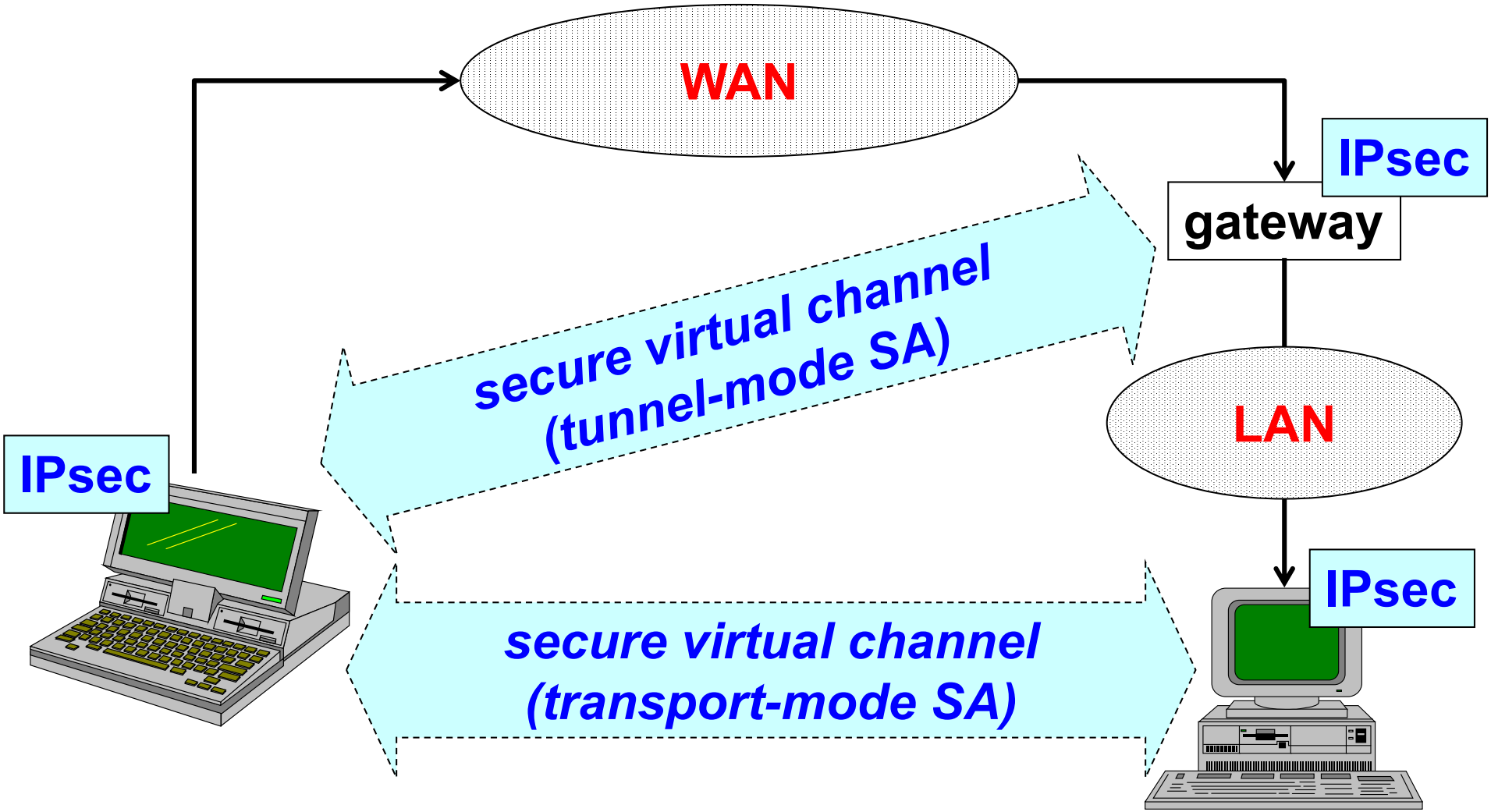
End-to-end security with basic VPN



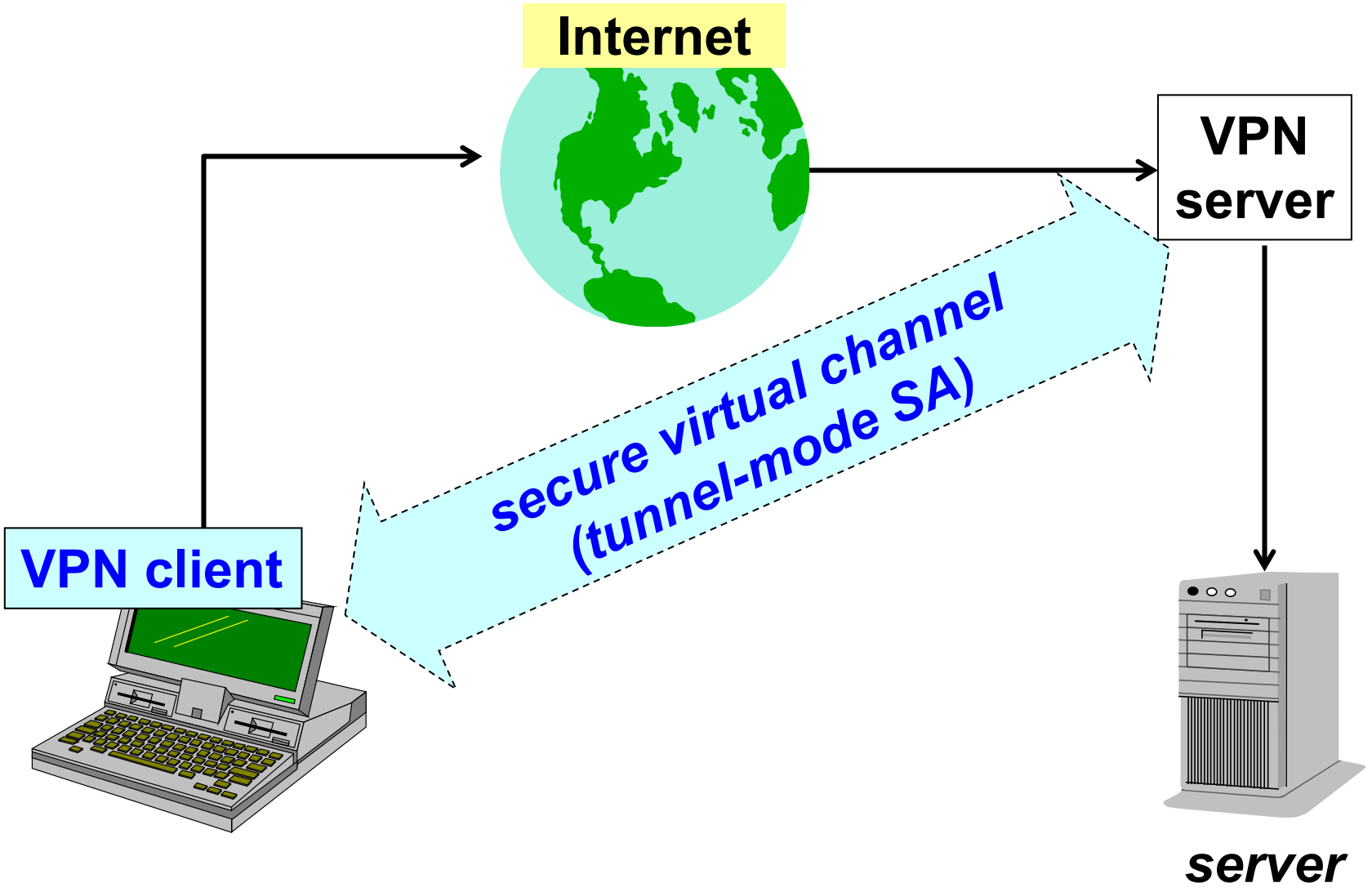
Secure gateway



Secure remote access



Anonymous navigation



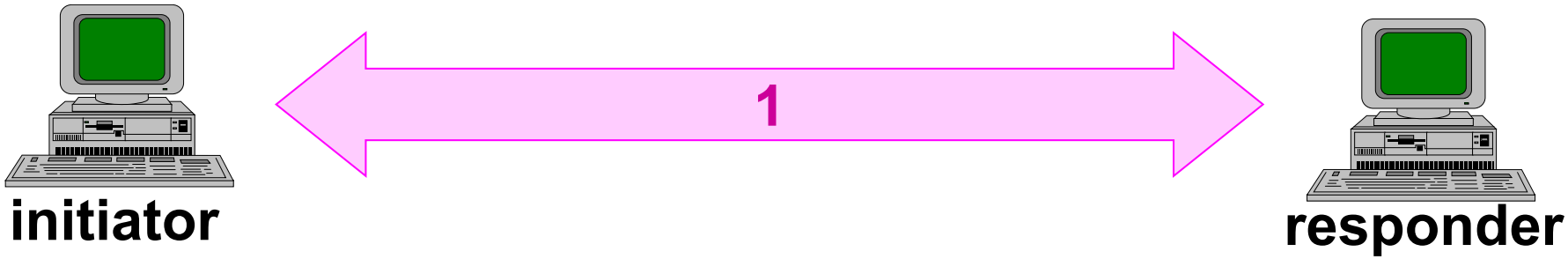
IPsec key management

- **very important component of IPsec**
- **provides to the IPsec parties the symmetric keys used for packet authentication and/or encryption**
- **what about key distribution?**
  - OOB (e.g. manual)
  - automatic in-band (which protocol?)

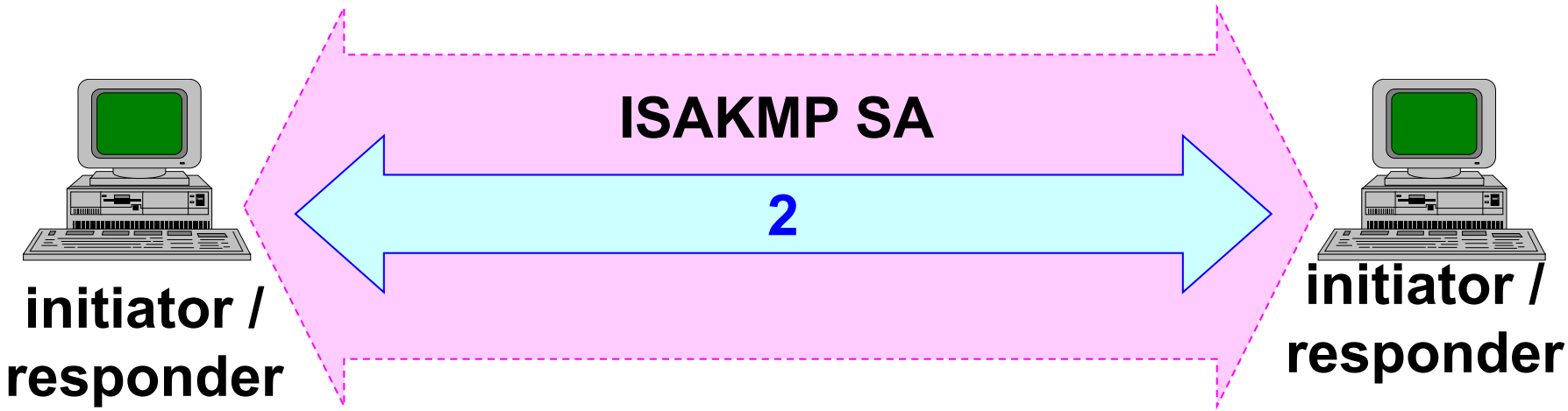
ISAKMP, OAKLEY, and IKE

- **ISAKMP, Internet Security Association and Key Management Protocol (RFC-2408)**
  - procedures to negotiate, set-up, modify and delete a SA
  - key exchange method not fixed:
    - OAKLEY (RFC-2412): protocol for authenticated exchange of symmetric keys
- **IKE, Internet Key Exchange (RFC-2409) = ISAKMP + OAKLEY**
  - creation of one SA to protect the ISAKMP exchange
  - this SA is used to protect the negotiation of the SA needed by IPsec traffic
- **the same ISAKMP SA may be reused several times to negotiate other IPsec SA**

IKE: operations



IKE phase 1 - negotiation of a bidirectional ISAKMP SA:
"main mode" or "aggressive mode"



IKE phase 2 - negotiation of the IPsec SA: "quick mode"

IKE: “modes” of operation

- **Main Mode:**
  - 6 messages
  - protects the parties identities
- **Aggressive Mode:**
  - 3 messages (but doesn’t protect the parties identities)
- **Quick Mode:**
  - 3 messages
  - negotiation only of the IPsec SA
- **New Group Mode:**
  - 2 messages

IKE: authentication methods

- **Digital Signature**
  - non-repudiation of the IKE negotiation
- **Public Key Encryption**
  - identity protection in the aggressive mode
- **Revised Public Key Encryption**
  - less expensive, only 2 public-key operations
- **Pre-Shared Key**
  - the party ID may only be its IP address (problem with mobile users)

VPN concentrator

- **special-purpose appliance that acts as a terminator of IPsec tunnel:**
  - for remote access of single clients
  - to create site-to-site VPN
- **very high performance with respect to the costs (low)**

Applicability of IPsec

- **only unicast packets (no broadcast, no multicast, no anycast)**
- **between parties that activated a SA:**
  - by shared keys
  - by X.509 certificates
- **… therefore in “closed” groups**

IP (in)security

- **addresses are not authenticated**
- **packets are not protected:**
  - integrity
  - authentication
  - confidentiality
  - replay
- **therefore all protocols using IP as carrier can be attacked, mainly relevant for the "service" protocols (i.e. the non-application ones, such as ICMP, IGMP, DNS, RIP, …)**

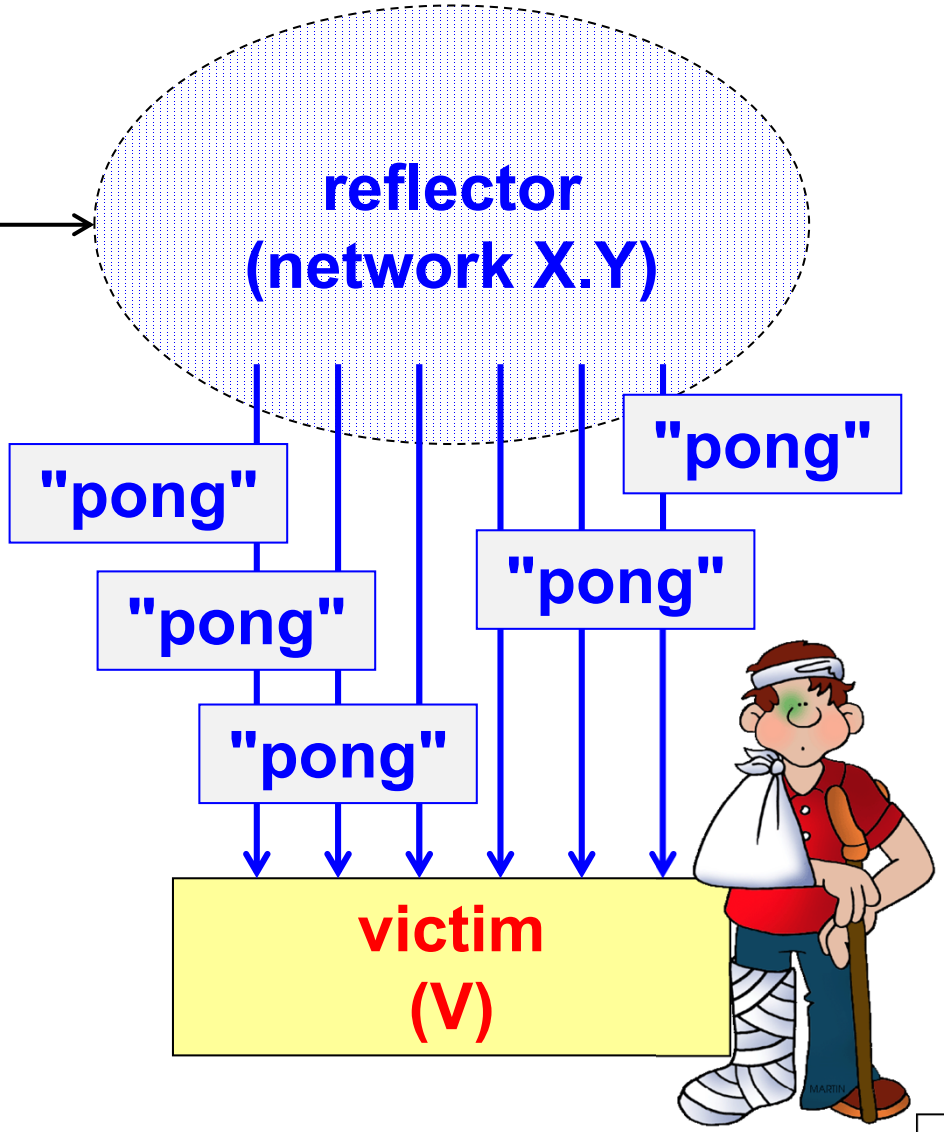
ICMP security

- **Internet Control and Management Protocol**
- **vital for network management**
- **many attacks are possible because it has no authentication**
- **ICMP functions:**
  - echo request / reply
  - destination unreachable (network / host / protocol / port unreachable)
  - source quench
  - redirect
  - time exceeded for a datagram

Smurfing attack

src = V
dst = X.Y.255.255
ICMP echo request

ping



Anti-smurfing countermeasures

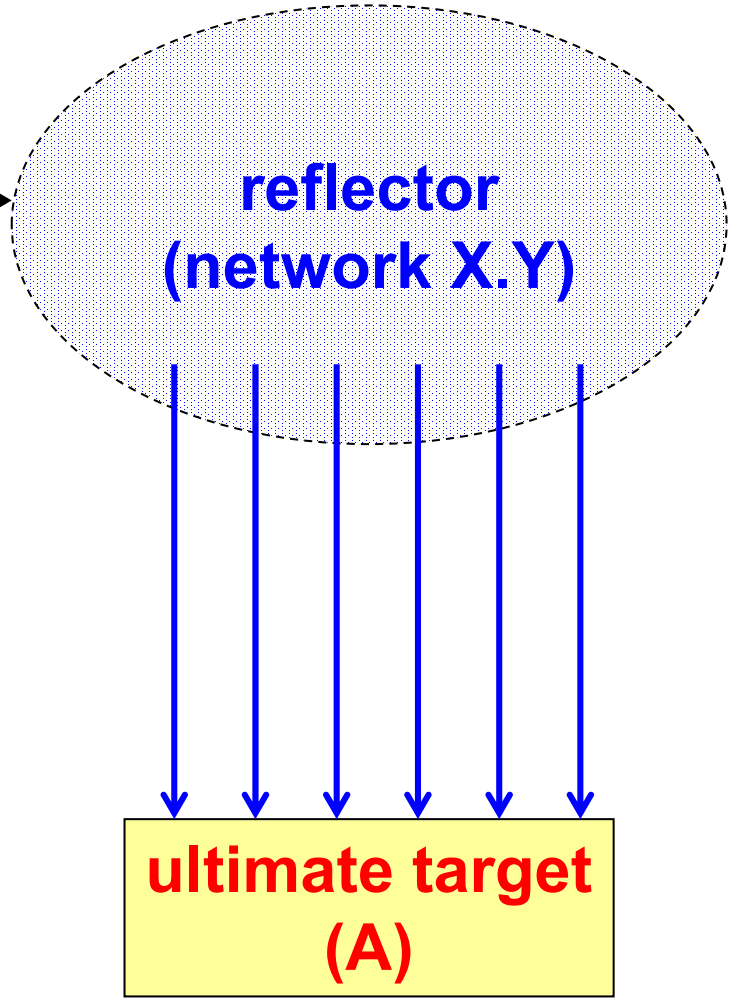
- **for external attacks:**
  - reject IP broadcast packets at your border
  - example (CISCO syntax)

```
interface serial0
no ip directed-broadcast
```

- **for internal attacks:**
  - identify the attacker via network management tools

Fraggle attack

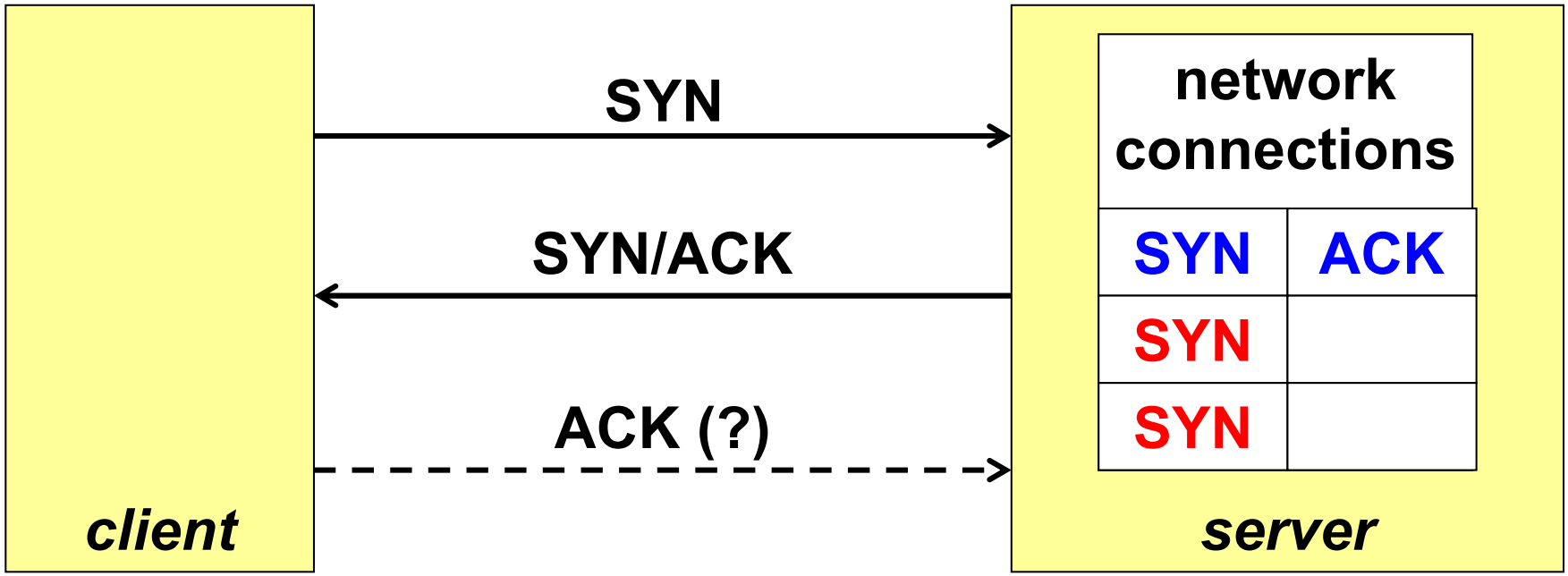
src = A
dst = X.Y.255.255
UDP echo request



ARP poisoning

- **ARP = Address Resolution Protocol (RFC-826)**
  - used to discover the L2 address of a node when knowing its L3 address
  - result stored in the ARP table
- **ARP poisoning:**
  - nodes accept ARP reply without ARP request
  - nodes overwrite static ARP entries with the dynamic ones (obtained from ARP reply)
  - the “ar$sha” ARP field (sender hw address) may differ from the src field in the 802.3 packet
  - used by attack tools (e.g. Ettercap)

TCP SYN flooding

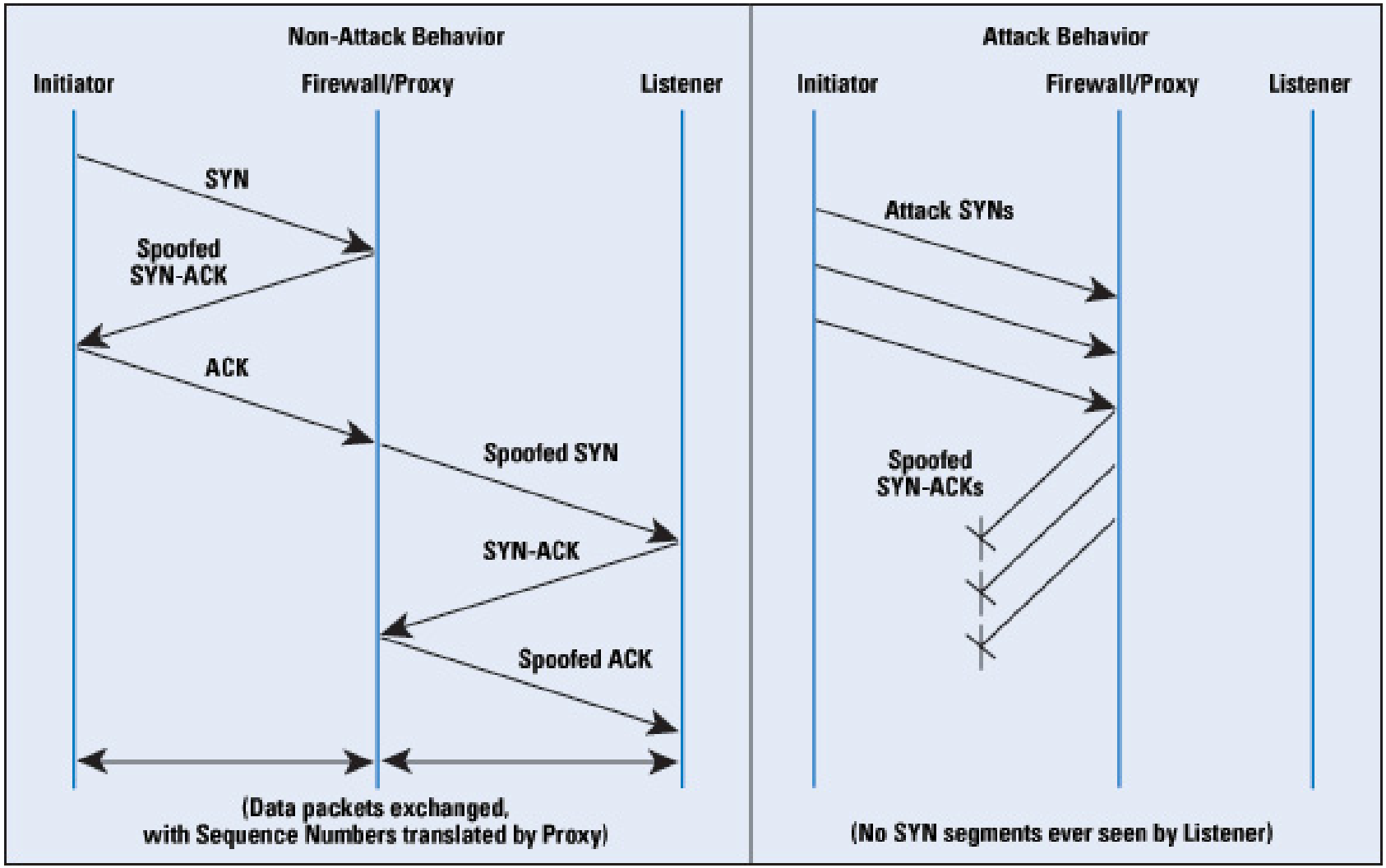


- multiple requests with IP spoofing
- the connection table is saturated until half-open connections timeout (typical value: 75”)

Protection against SYN flooding

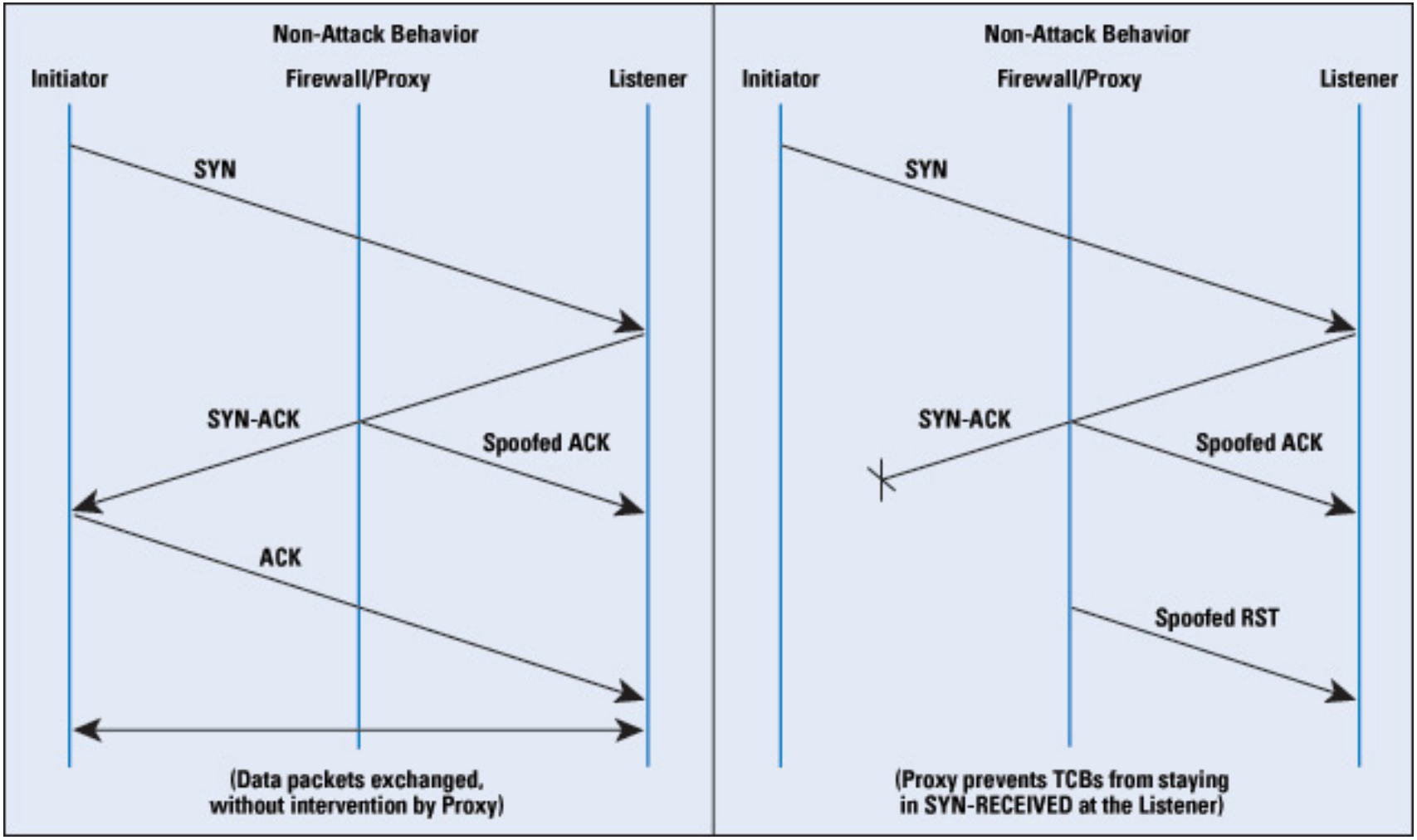
- **decrease the timeout**
  - risk to delete requests from valid but slow clients
- **increase the table size**
  - can be circumvented by sending more requests
- **use a router as “SYN interceptor”:**
  - substitutes the server in the first phase
  - if the handshake completes successfully, then transfers the channel to the server
  - “aggressive” timeout (risky!)
- **use a router as “SYN monitor”:**
  - kills the pending connection requests (RST)

SYN interceptor (or firewall relay)



http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_9-4/syn_flooding_attacks.html

SYN monitor (or firewall gateway)



http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_9-4/syn_flooding_attacks.html

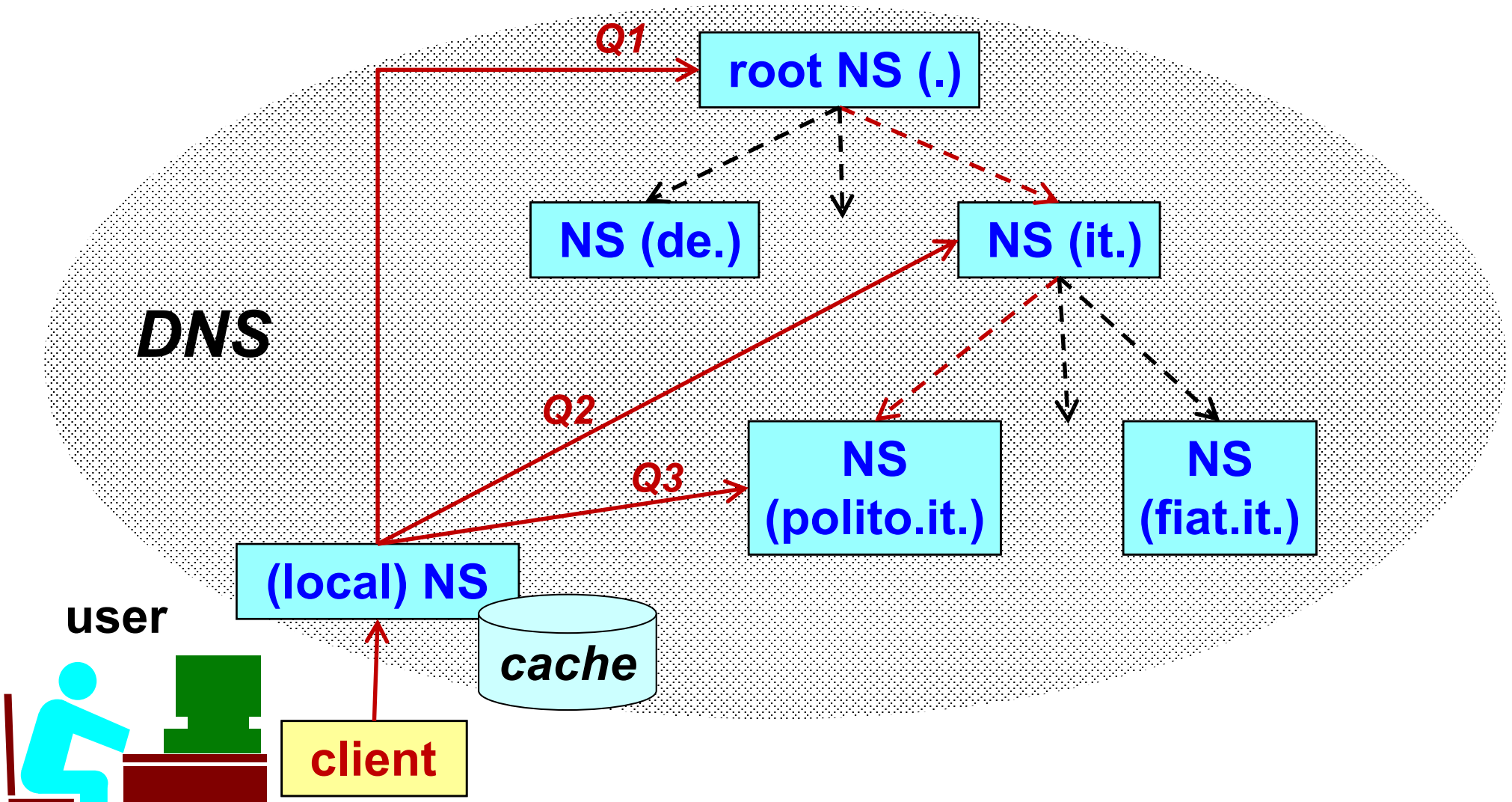
SYN cookie

- idea of D.J.Bernstein (http://cr.yp.to)
- the only approach really effective to completely avoid the SYN flooding attack
- uses the TCP sequence number of the SYN-ACK packet to transmit a cookie to the client and later recognize the clients that already sent the SYN without storing any info about them on the server
- available on Linux and Solaris

DNS security

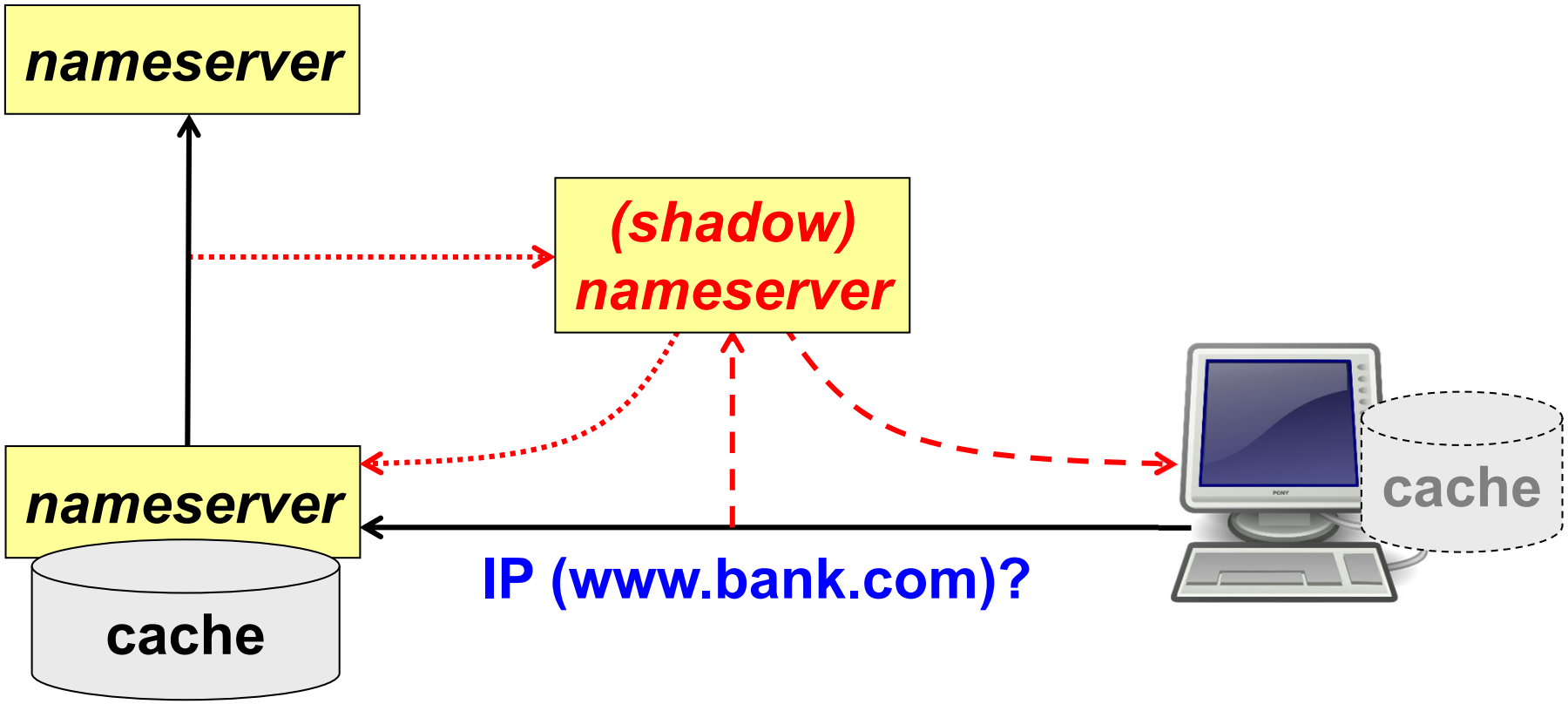
- **DNS (Domain Name System)**
- **translation:**
  - from names to IP addresses
  - from IP addresses to names
- **vital service**
- **queries over port 53/UDP**
- **zone transfers over port 53/TCP**
- **no security**
- **DNS-SEC under development / deployment**

DNS architecture (iterative query)



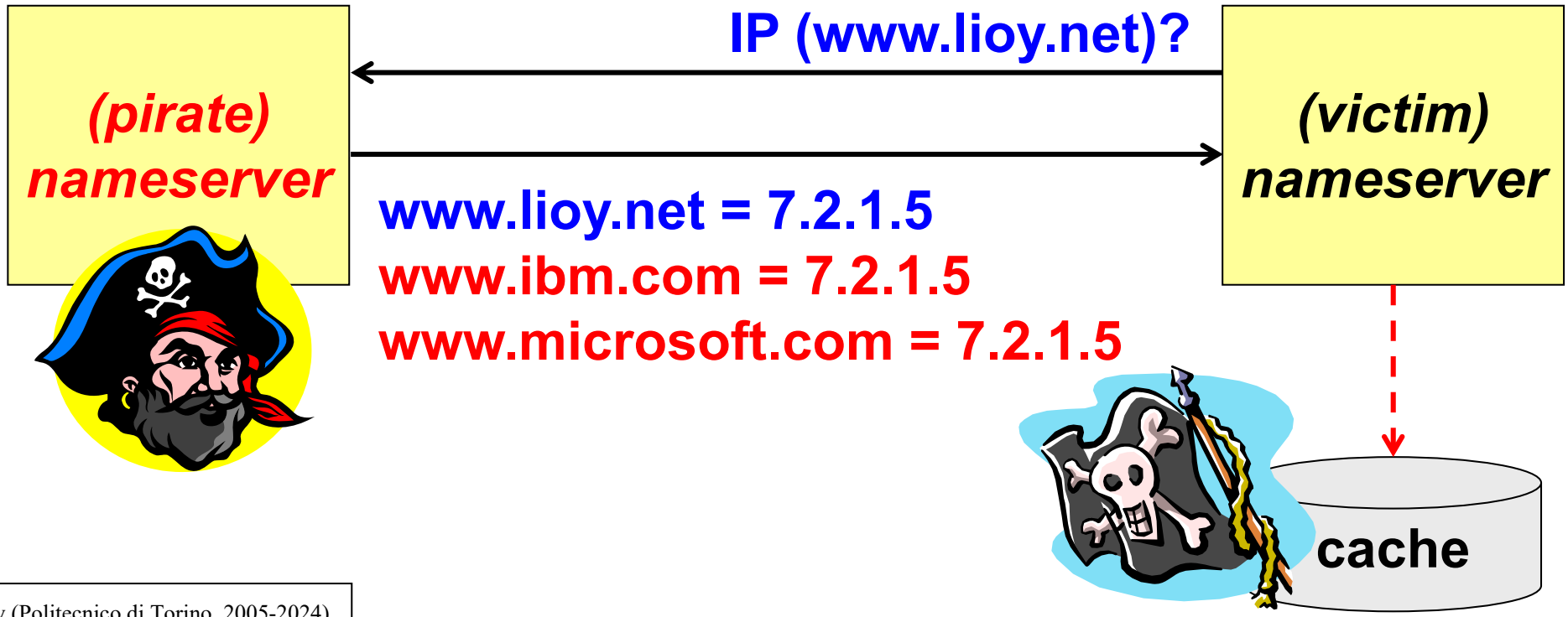
DNS shadow server

- sniffing to intercept the queries
- spoofing to generate fake answers (DoS or traffic redirection to fake sites)



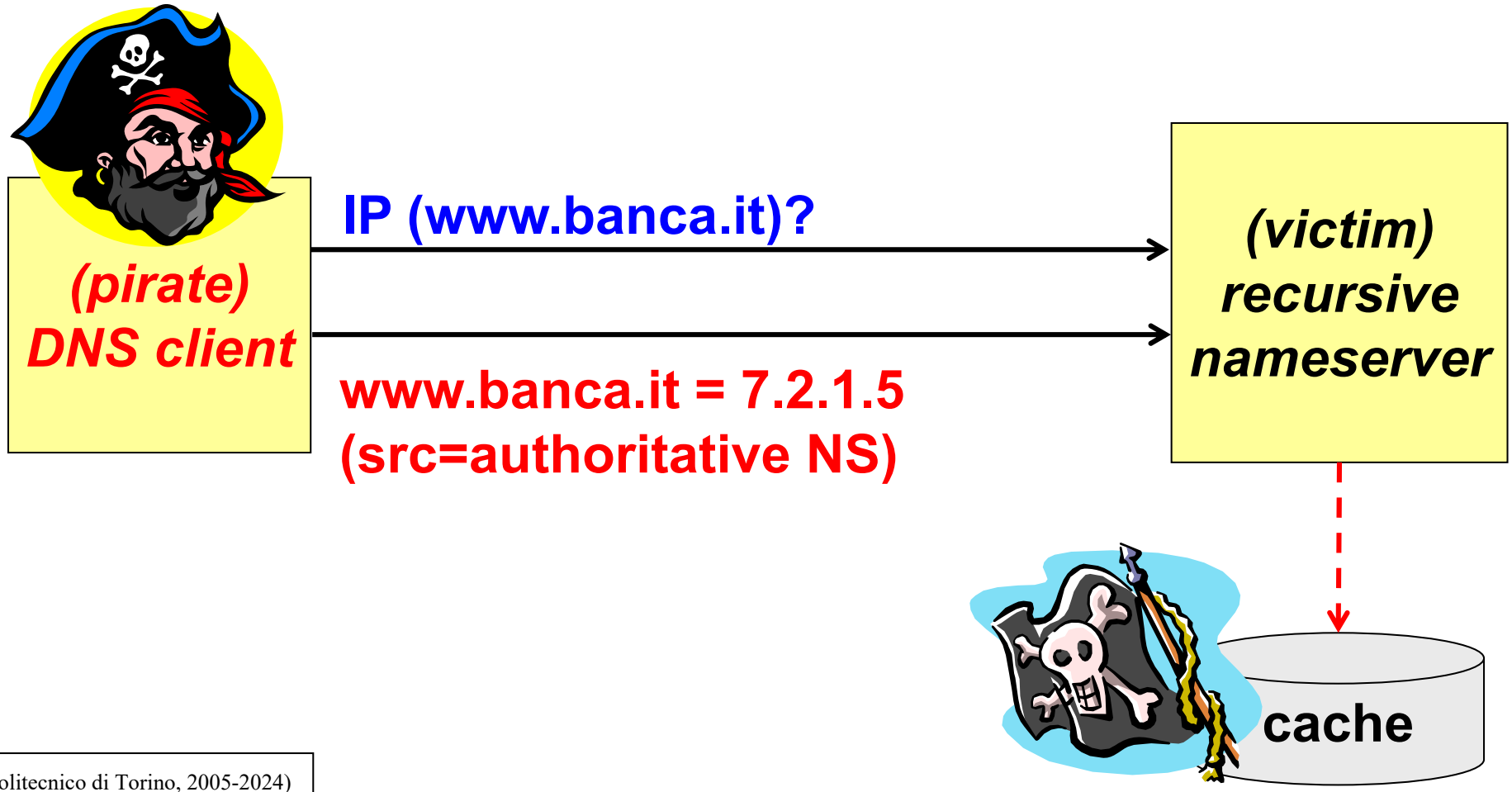
DNS cache poisoning

- attract the victim to make a query on my NS
- provide answers also to queries never done to push / overwrite the victim's cache

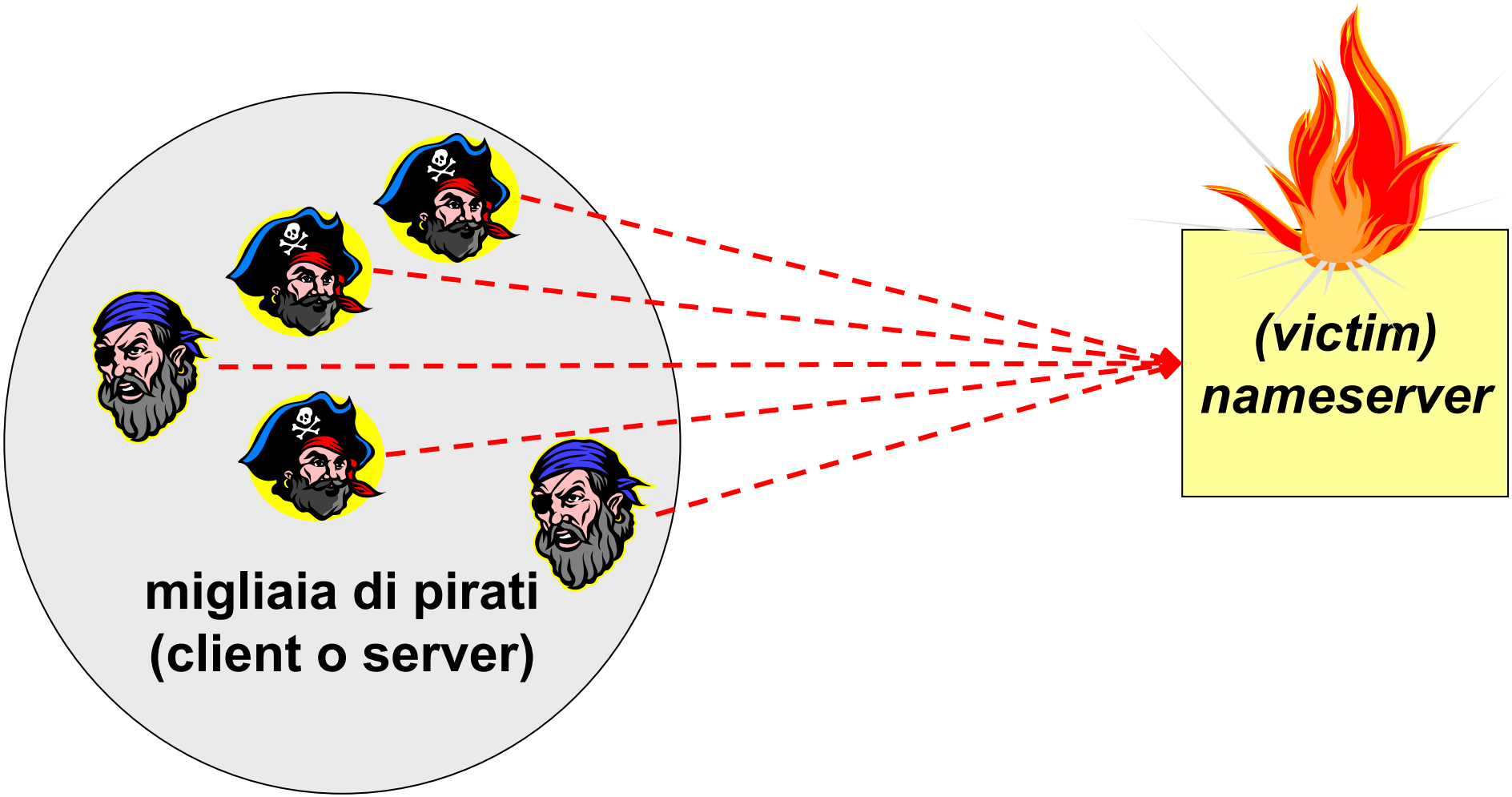


DNS cache poisoning (2nd version)

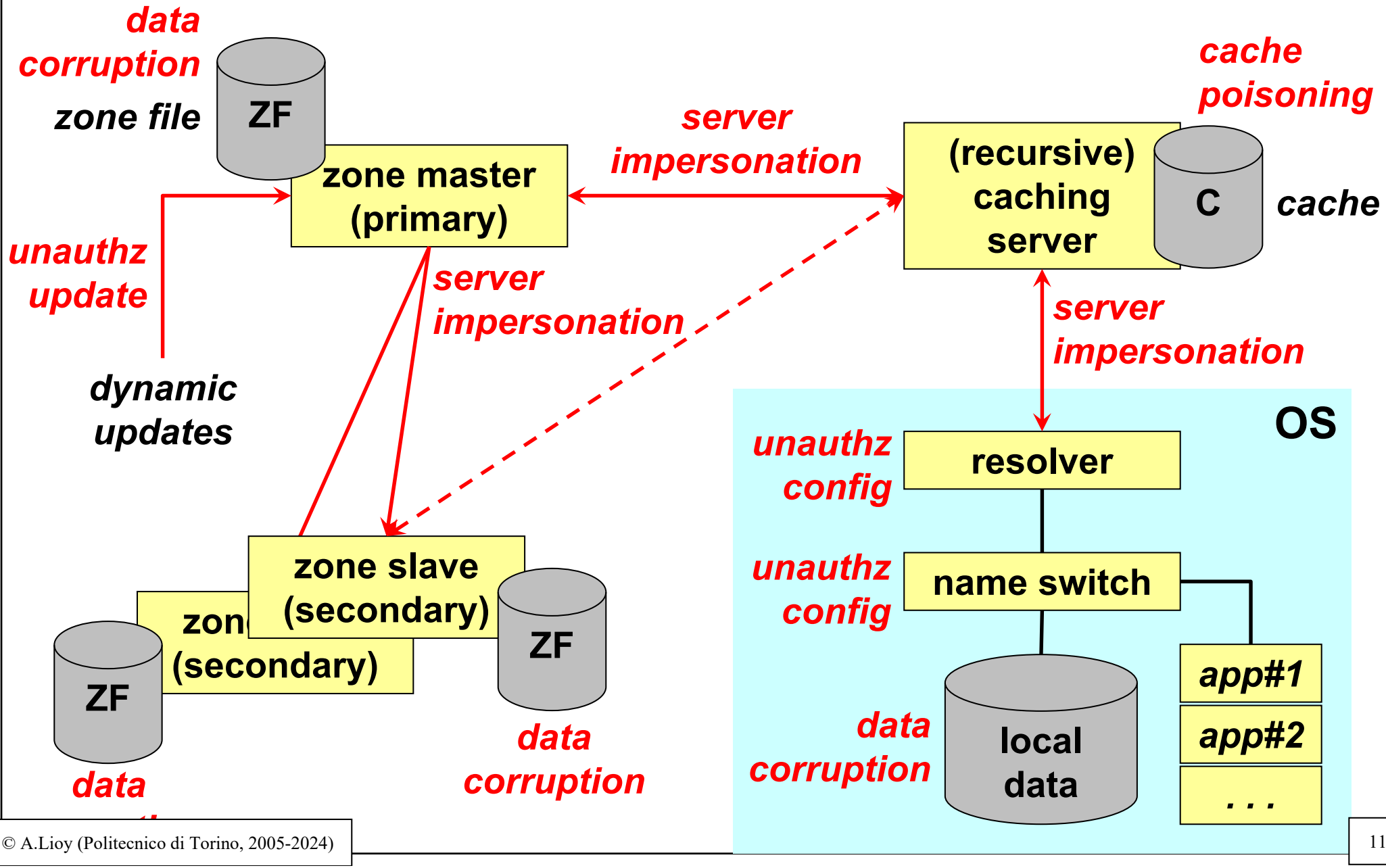
- make a query and self-provide the (wrong) answer too, to insert it into the victim's cache



(DNS) flash crowd



Name-address translation



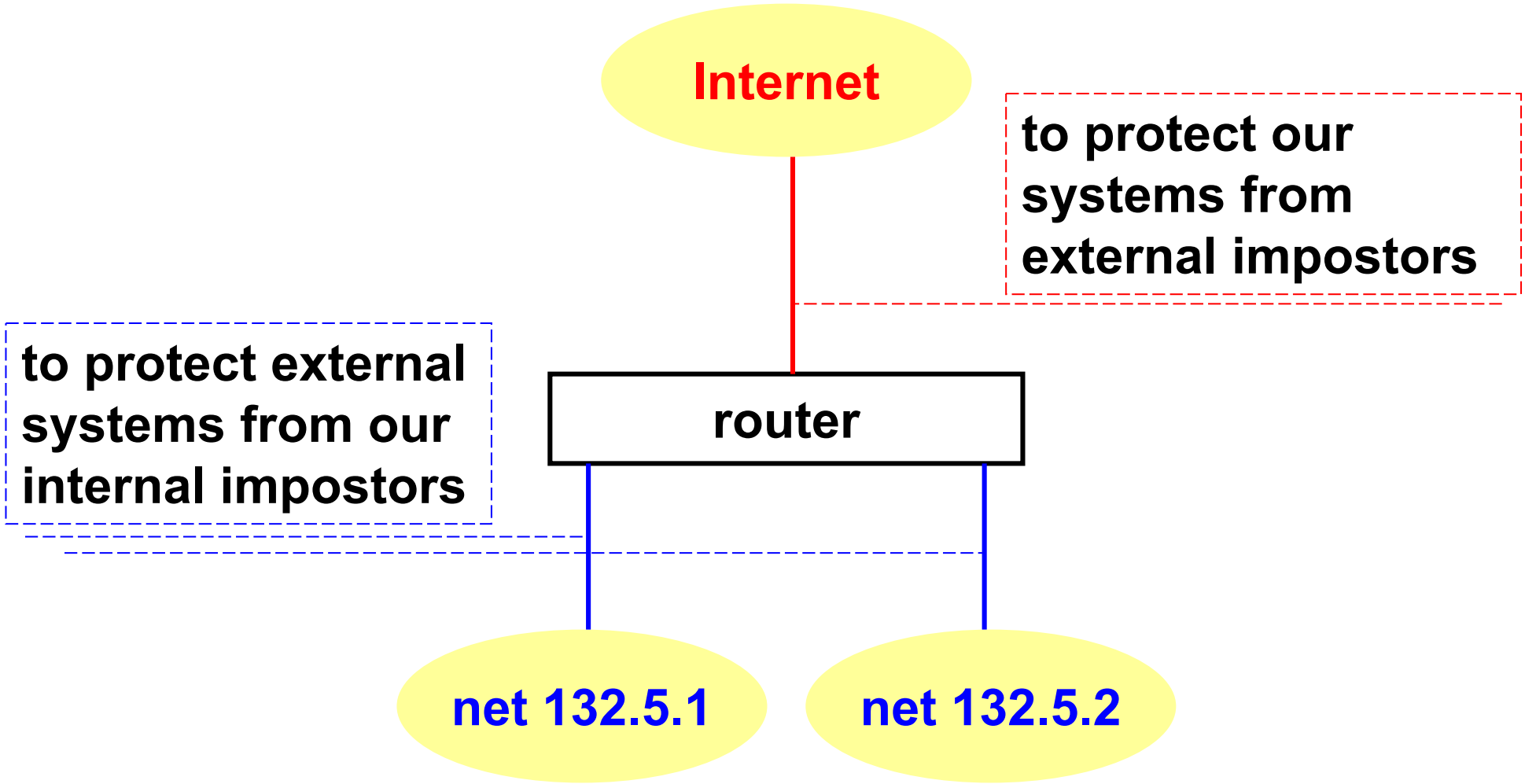
DoT and DoH

- **apart from attacks against the nameservers, DNS has got a user privacy problem for the queries:**
  - can be read while in transit
  - can be read and logged by the nameserver
- **DNS-over-TLS (DoT)**
  - query and response encapsulated in a secure TLS tunnel
  - but it is still evident that it's a DNS exchange
- **DNS-over-HTTPS (RFC-8484)**
  - query and response are part of a normal HTTPS exchange
  - externally it looks like visiting a secure web page
- **well-known service providers of DoH/DoT:**
  - Cloudflare (1.1.1.1) and Google (8.8.8.8 and 8.8.4.4)

Protection from IP spoofing

- to protect ourselves from external impostors
- also to protect the external world from our internal impostors (=net-etiquette)
- RFC-2827 “Network ingress filtering: defeating Denial of Service attacks which employ IP source address spoofing”
- RFC-3704 “Ingress filtering for multihomed networks”
- RFC-3013 “Recommended Internet Service Provider security services and procedures”

Filters for IP spoofing protection



Example of IP spoofing protection

```
access-list 101 deny ip
  132.5.0.0 0.0.255.255 0.0.0.0 255.255.255.255
interface serial 0
ip access-group 101 in

access-list 102 permit ip
  132.5.1.0 0.0.0.255 0.0.0.0 255.255.255.255
interface ethernet 0
ip access-group 102 in

access-list 103 permit ip
  132.5.2.0 0.0.0.255 0.0.0.0 255.255.255.255
interface ethernet 1
ip access-group 103 in
```