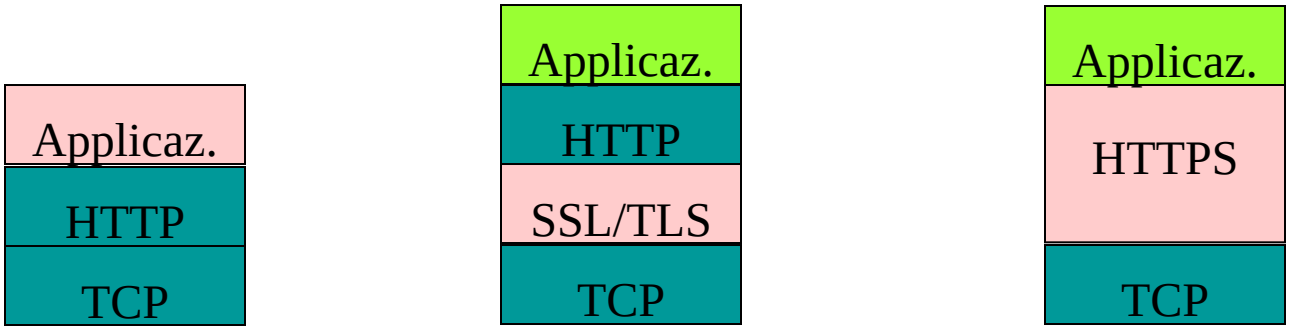
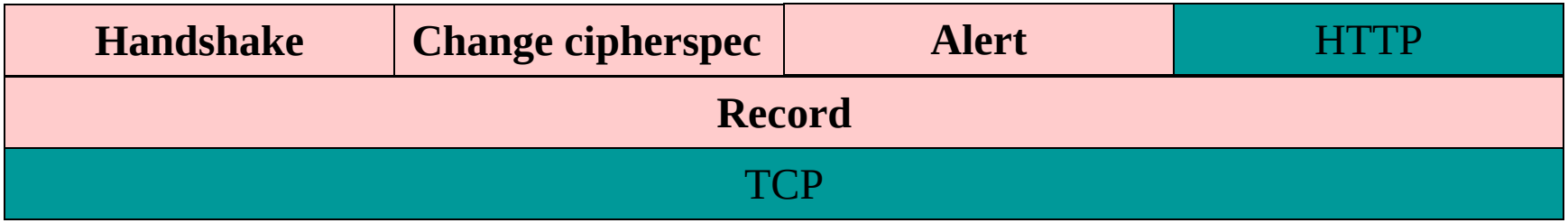

Servizi Sicuri
per le comunicazioni in rete

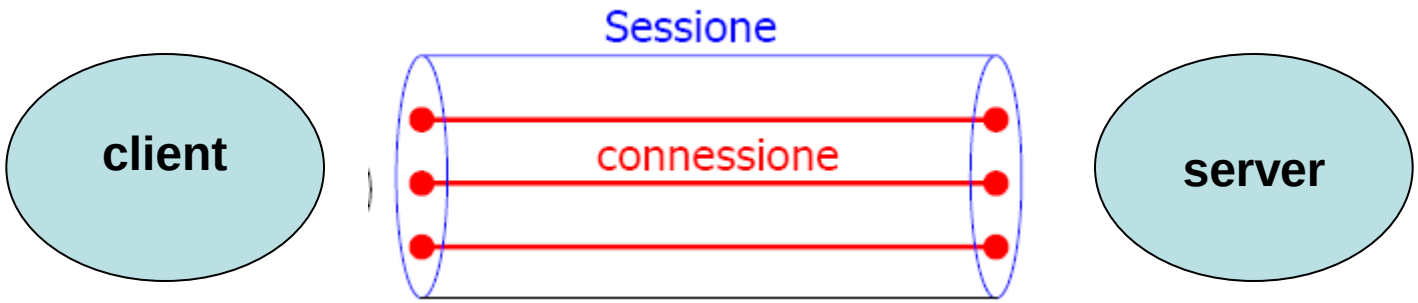
Secure socket layer (SSL)
Transport layer security (TLS)



SSL: Netscape
TLS:RFC 2246

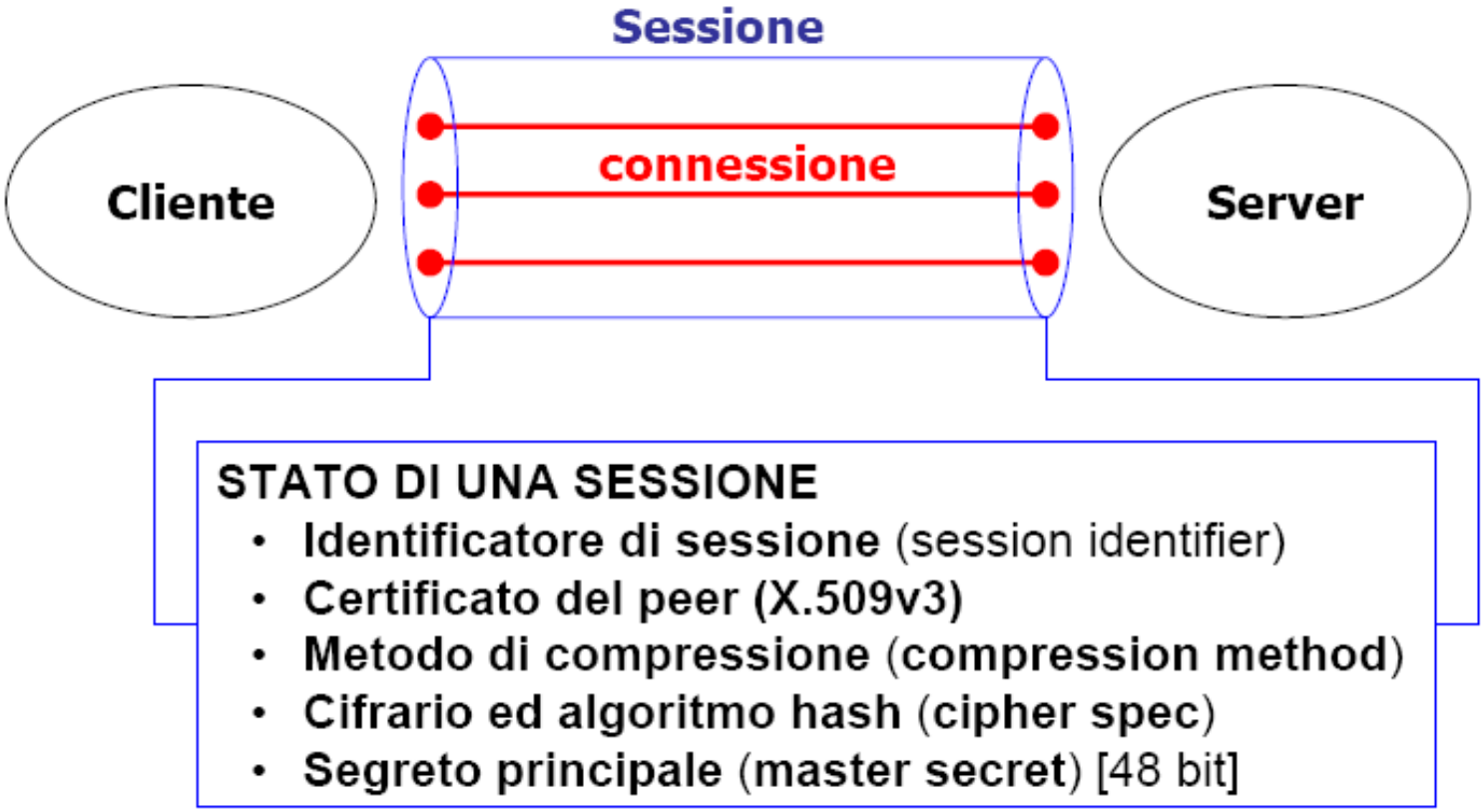


Sessione e Connessione



- La sessione è un'associazione logica tra Cliente e Server.
- Una sessione viene creata dal protocollo di Handshake e definisce un insieme di parametri crittografici che possono essere condivisi fra molteplici connessioni.
- La sessione evita la costosa rinegoziazione dei parametri di sicurezza per ciascuna connessione

Sessione e Connessione



Questi valori definiscono lo stato di ciascun **endpoint** della sessione

Sessione e Connessione

STATO DI UNA CONNESSIONE

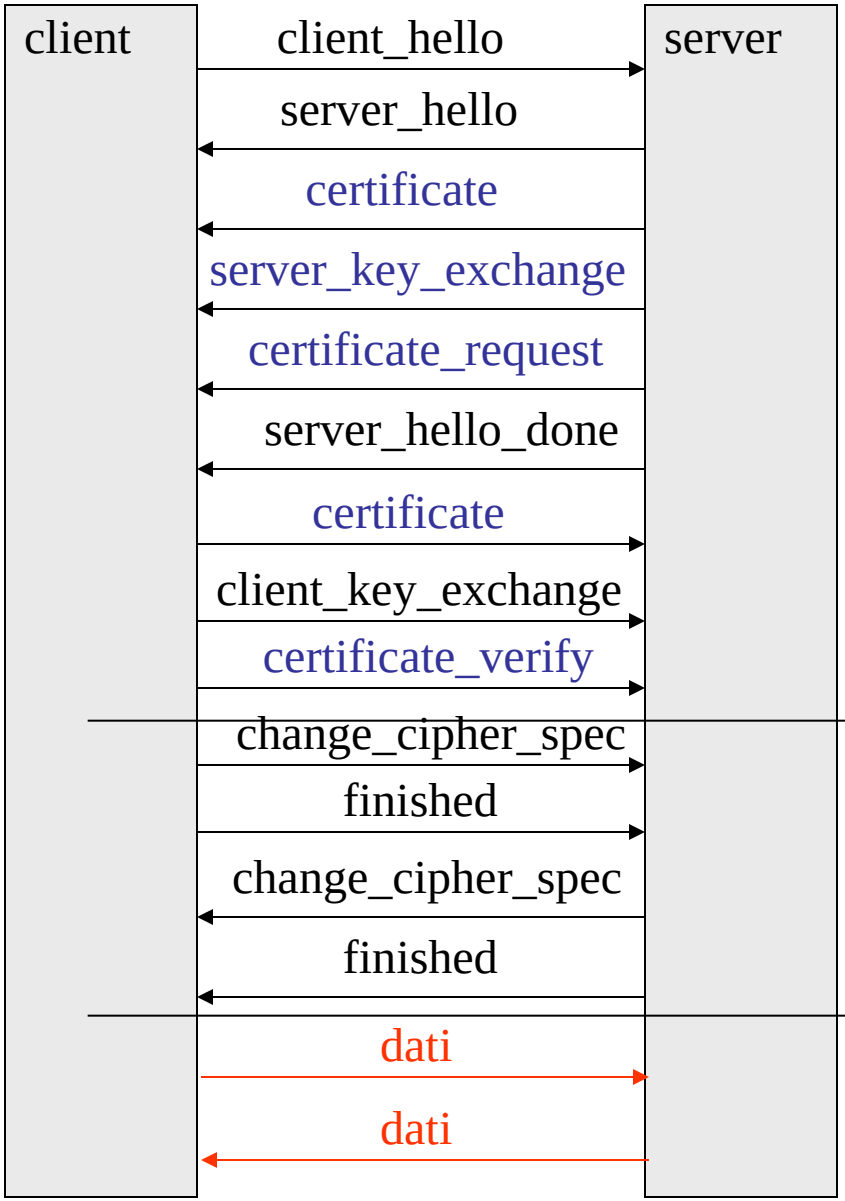
- Server random number (nonce)
- Client random number (nonce)
- Server write MAC secret
- Client write MAC secret
- Server write key
- Client write key
- Initialization vectors
- Sequence numbers

Protocollo di Handshake

Ogni messaggio ha tre campi: tipo, lungh., content (parametri associati al tipo)

TIPO	CONTENUTO
hello_request	nessun paramentro
client_hello	Versione, random, ID di sessione, suite di cifratura, metodo di compressione
server_hello	Versione, random, ID di sessione, suite di cifratura, metodo di compressione
certificate	Catena di certificati X.509v3
server_key_exchange	Parametri, firma
certificate_request	Tipo, autorità
server_hello_done	Nessun parametro
certificate_verify	Firma
client_key_exchange	Parametri, firma
finished	Valore hash

I protocolli Handshake, Change e Record



Protocollo Handshake, Fase 1: selezione dei meccanismi per la riservatezza, per la autenticazione e per l'accordo sul segreto

Protocollo Handshake, Fase 2 : autenticazione del server e invio di chiavi. Messaggio certificate obbligatorio tranne per scambio anonymous. Messaggio server_key_exchange obblig. per DH anon., ephemeral, RSA se server solo chiavi di firma RSA

Protocollo Handshake, Fase 3 : autenticazione del client e accordo sul segreto

Segreto condiviso

Protocollo Hanshake, Fase 4 (Change) controllo di tutti i dati scambiati

Chiavi installate

Protocollo Record: scambio sicuro dei dati

I Messaggi Hello

In questa fase il client ed il server si dicono cosa sanno fare ed il client autentica il server

- Messaggio client_hello e server_hello
  - Versione di SSL
  - Random: timestamp (32 bit) + random byte[28]; client e server generano quantità random diverse
  - ID di sessione: in base al valore (1) viene creata una nuova sessione, (2) viene creata una connessione in una sessione esistente (3) vengono rinegoziati i parametri di una sessione esistente
  - Suite di cifratura (Cipher suite) specifica la lista degli algoritmi di cifratura supportati dal cliente in ordine di gradimento; ciascun elemento della lista definisce sia il metodo per lo scambio delle chiavi sia una suite di cifratura (algoritmi, hash, vettori di inizial.); il server sceglie
  - Metodo di compressione: lista dei metodi di compressione supportati dal client; il server sceglie

Hello messages

- client_hello
  - client_version
    - the highest version supported by the client
  - client_random
    - current time (4 bytes) + pseudo random bytes (28 bytes)
  - session_id
    - empty if the client wants to create a new session, or
    - the session ID of an old session within which the client wants to create the new connection
  - cipher_suites
    - list of cryptographic options supported by the client ordered by preference
    - a cipher suite contains the specification of the
      - key exchange method, the encryption and the MAC algorithm
      - the algorithms implicitly specify the hash_size, IV_size, and key_material parameters (part of the Cipher Spec of the session state)
    - exmaple: SSL_RSA_with_3DES_EDE_CBC_SHA
  - compression_methods
    - list of compression methods supported by the client

Hello messages cont'd

- server_hello
  - server_version
    - min(highest version supported by client, highest version supported by server)
  - server_random
    - current time + random bytes
    - random bytes must be independent of the client random
  - session_id
    - session ID chosen by the server
    - if the client wanted to resume an old session:
      - server checks if the session is resumable
      - if so, it responds with the session ID and the parties proceed to the finished messages
    - if the client wanted a new session
      - server generates a new session ID
  - cipher_suite
    - single cipher suite selected by the server from the list given by the client
  - compression_method
    - single compression method selected by the server

Supported key exchange methods

- RSA based (SSL_RSA_with...)
  - the secret key (pre-master secret) is encrypted with the server's public RSA key
  - the server's public key is made available to the client during the exchange
- fixed Diffie-Hellman (SSL_DH_RSA_with… or SSL_DH_DSS_with…)
  - the server has fix DH parameters contained in a certificate signed by a CA
  - the client may have fix DH parameters certified by a CA or it may send an unauthenticated one-time DH public value in the client_key_exchange message
- ephemeral Diffie-Hellman (SSL_DHE_RSA_with… or SSL_DHE_DSS_with…)
  - both the server and the client generate one-time DH parameters
  - the server signs its DH parameters with its private RSA or DSS key
  - the client may authenticate itself (if requested by the server) by signing the hash of the handshake messages with its private RSA or DSS key
- anonymous Diffie-Hellman
  - both the server and the client generate one-time DH parameters
  - they send their parameters to the peer without authentication
- Fortezza - Fortezza proprietary key exchange scheme

Server certificate and key exchange messages

- certificate
  - required for every key exchange method except for anonymous DH
  - contains one or a chain of X.509 certificates (up to a known root CA)
  - may contain
    - public RSA key suitable for encryption, or
    - public RSA or DSS key suitable for signing only, or
    - fix DH parameters
- server_key_exchange
  - sent only if the certificate does not contain enough information to complete the key exchange (e.g., the certificate contains an RSA signing key only)
  - may contain
    - public RSA key (exponent and modulus), or
    - DH parameters (p, g, public DH value), or
    - Fortezza parameters
  - digitally signed
    - if DSS: SHA-1 hash of (client_random | server_random | server_params) is signed
    - if RSA: MD5 hash and SHA-1 hash of (client_random | server_random | server_params) are concatenated and encrypted with the private RSA key

Certificate request and server hello done msgs

- certificate_request
  - sent if the client needs to authenticate itself
  - specifies which type of certificate is requested (rsa_sign, dss_sign, rsa_fixed_dh, dss_fixed_dh, …)
- server_hello_done
  - sent to indicate that the server is finished its part of the key exchange
  - after sending this message the server waits for client response
  - the client should verify that the server provided a valid certificate and the server parameters are acceptable

Client authentication and key exchange

- certificate
  - sent only if requested by the server
  - may contain
    - public RSA or DSS key suitable for signing only, or
    - fix DH parameters
- client_key_exchange
  - always sent (but it is empty if the key exchange method is fix DH)
  - may contain
    - RSA encrypted pre-master secret, or
    - client one-time public DH value, or
    - Fortezza key exchange parameters
- certificate_verify
  - sent only if the client sent a certificate
  - provides client authentication
  - contains signed hash of all the previous handshake messages
    - if DSS: SHA-1 hash is signed
    - if RSA: MD5 and SHA-1 hash is concatenated and encrypted with the private key

MD5(master_secret | pad_2 | MD5(handshake_messages | master_secret | pad_1))

SHA(master_secret | pad_2 | SHA(handshake_messages | master_secret | pad_1))

Finished messages

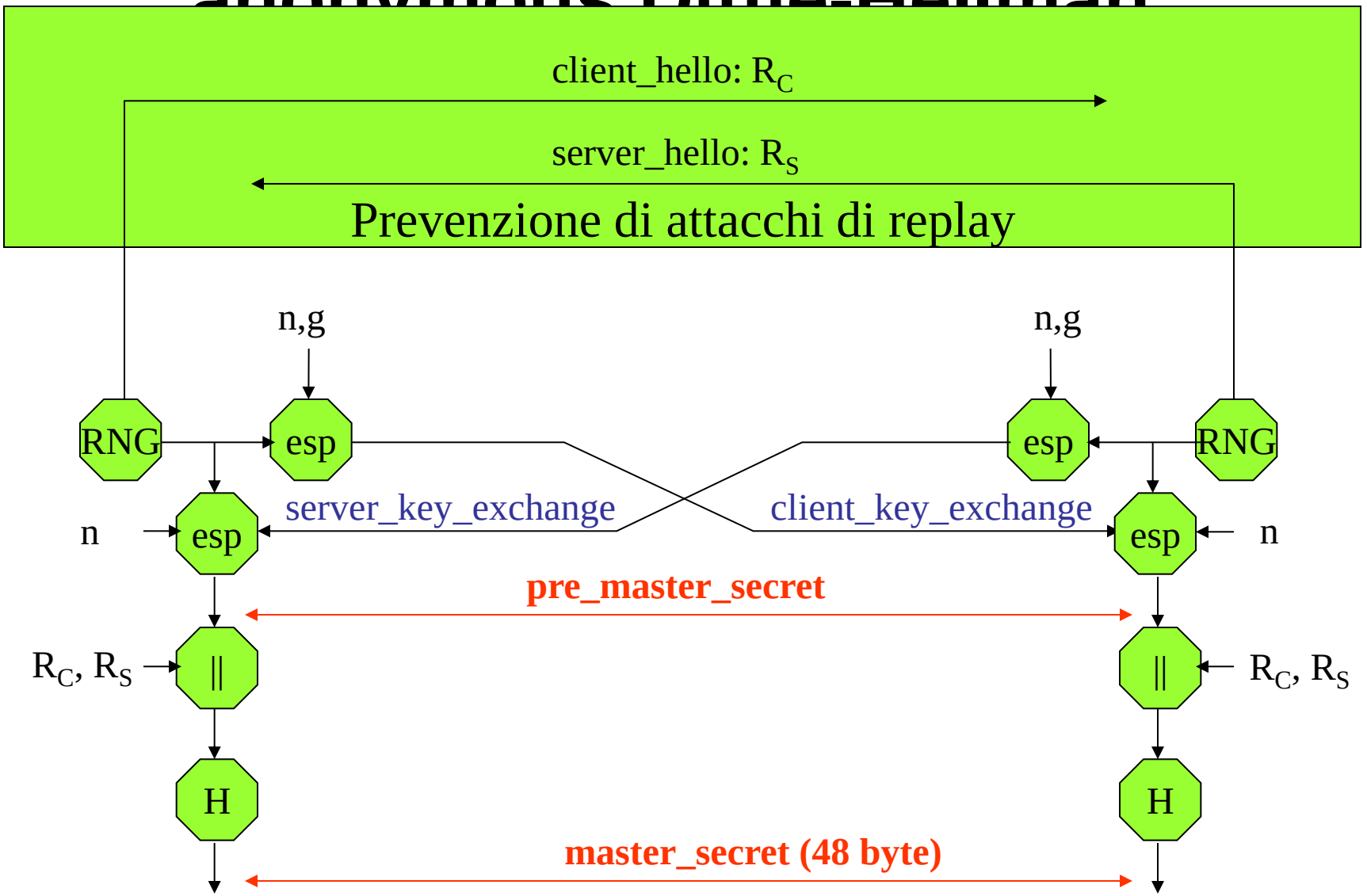
- finished
  - sent immediately after the change_cipher_spec message
  - first message that uses the newly negotiated algorithms, keys, IVs, etc.
  - used to verify that the key exchange and authentication was successful
  - contains the MD5 and SHA-1 hash of all the previous handshake messages:

    MD5(master_secret | pad_2 | MD5(handshake_messages | sender | master_secret | pad_1)) |

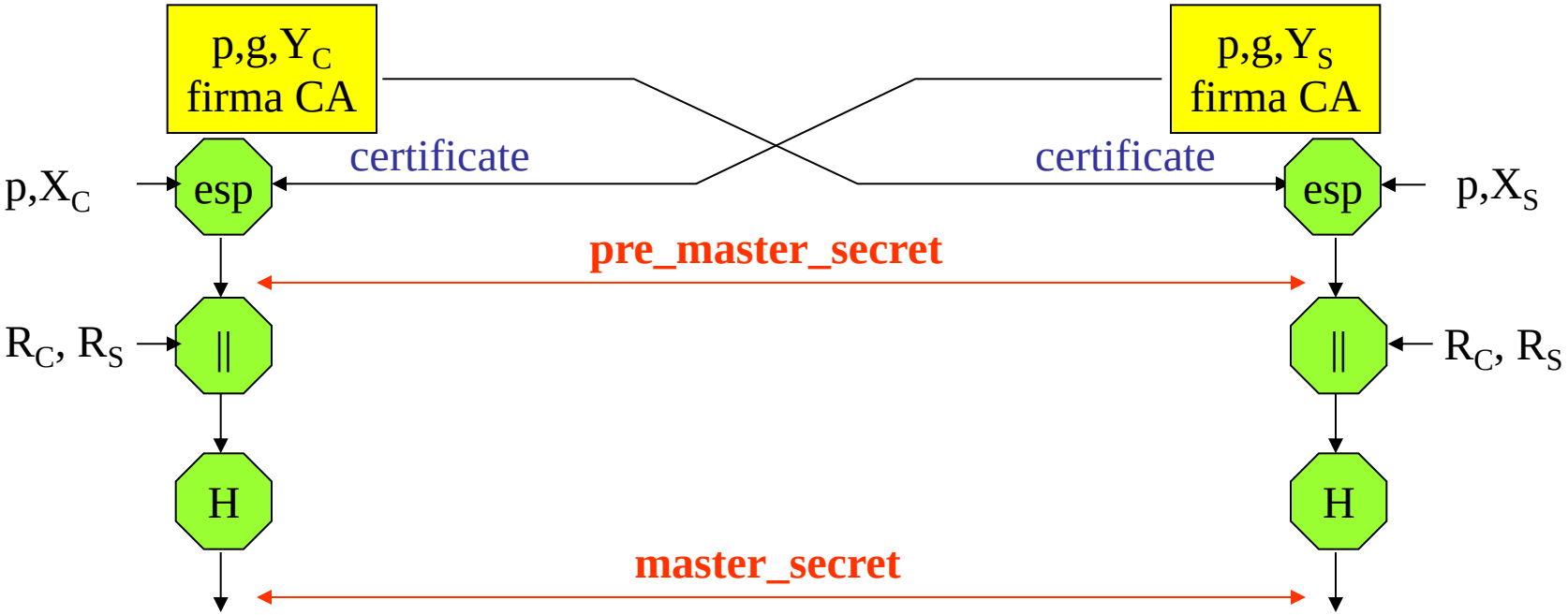
    SHA(master_secret | pad_2 | SHA(handshake_messages | sender | master_secret | pad_1))

    where “sender” is a code that identifies that the sender is the client or the server (client: 0x434C4E54; server: 0x53525652)

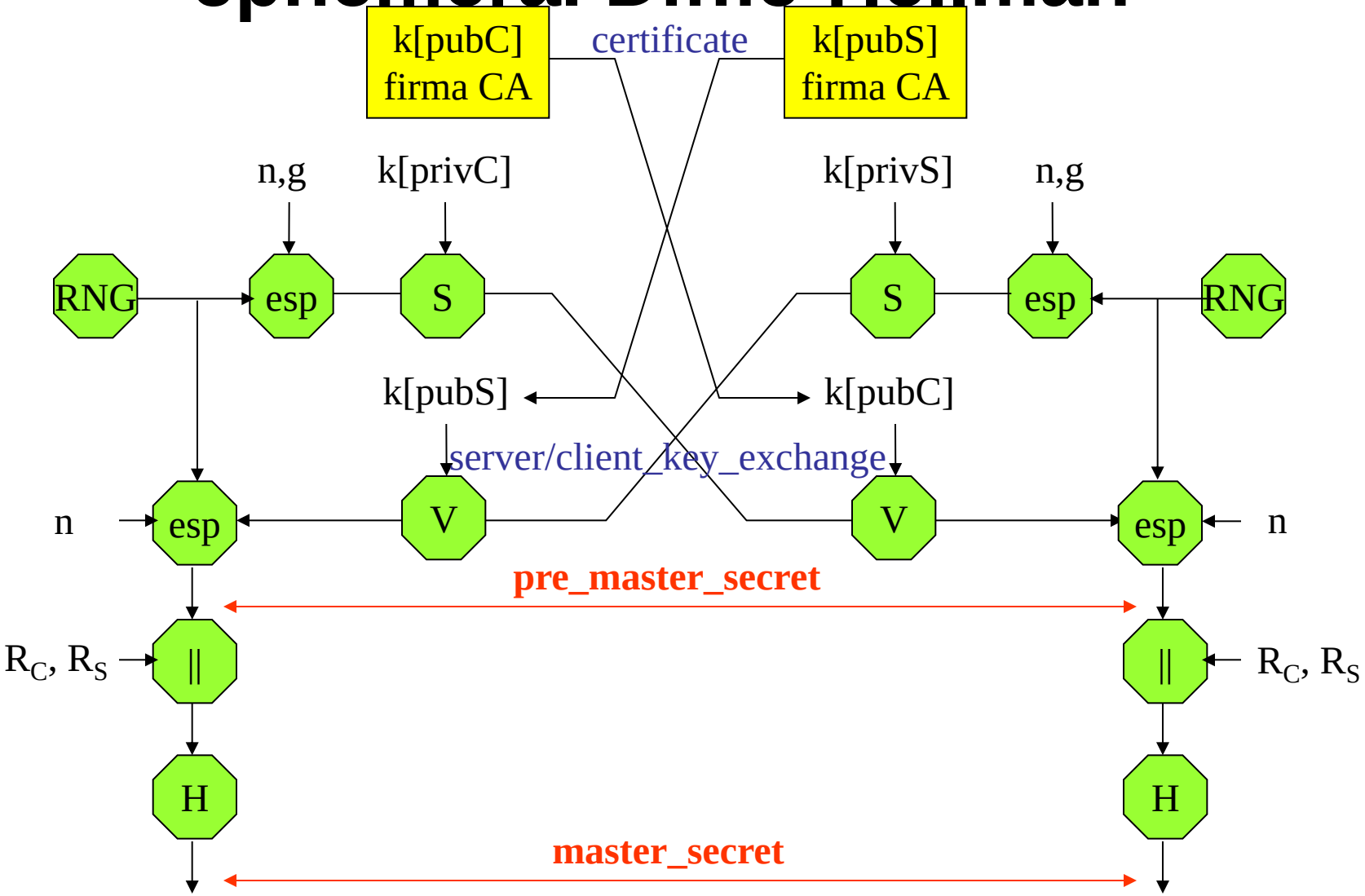
L'accordo sul segreto (Handshake 2&3): anonymous Diffie-Hellman



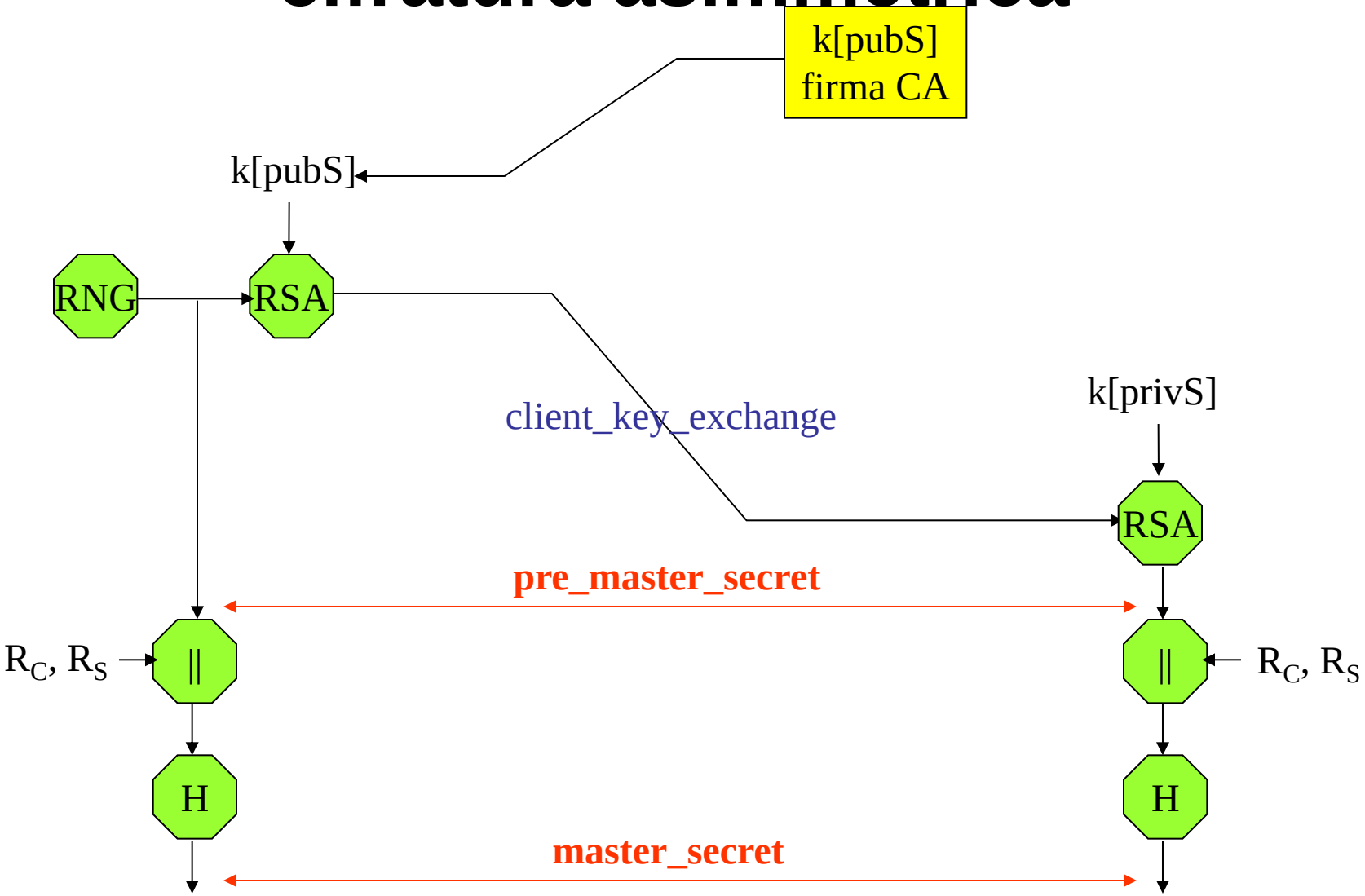
L'accordo sul segreto (Handshake 2&3): fixed Diffie-Hellman



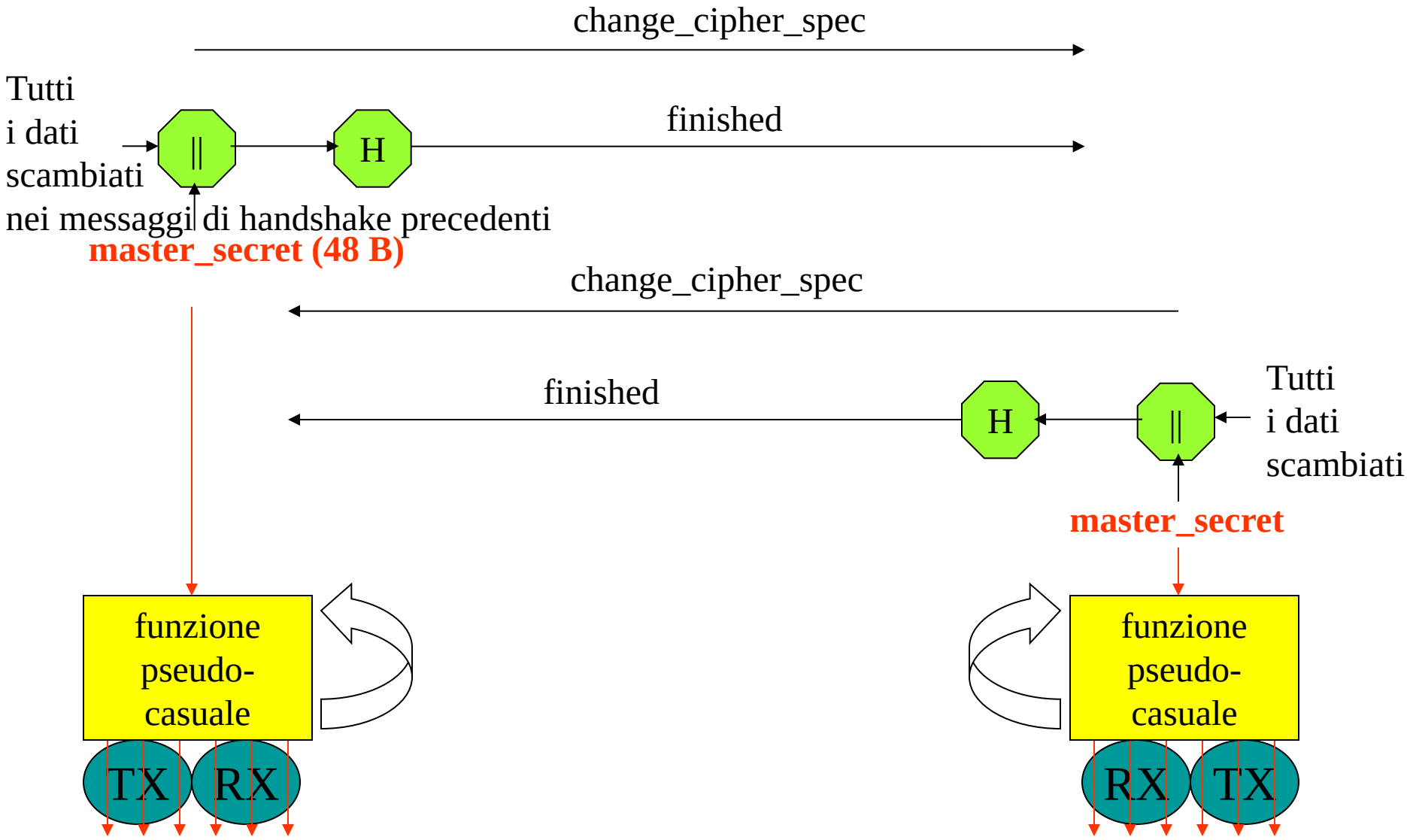
L'accordo sul segreto (Handshake 2&3): ephemeral Diffie-Hellman

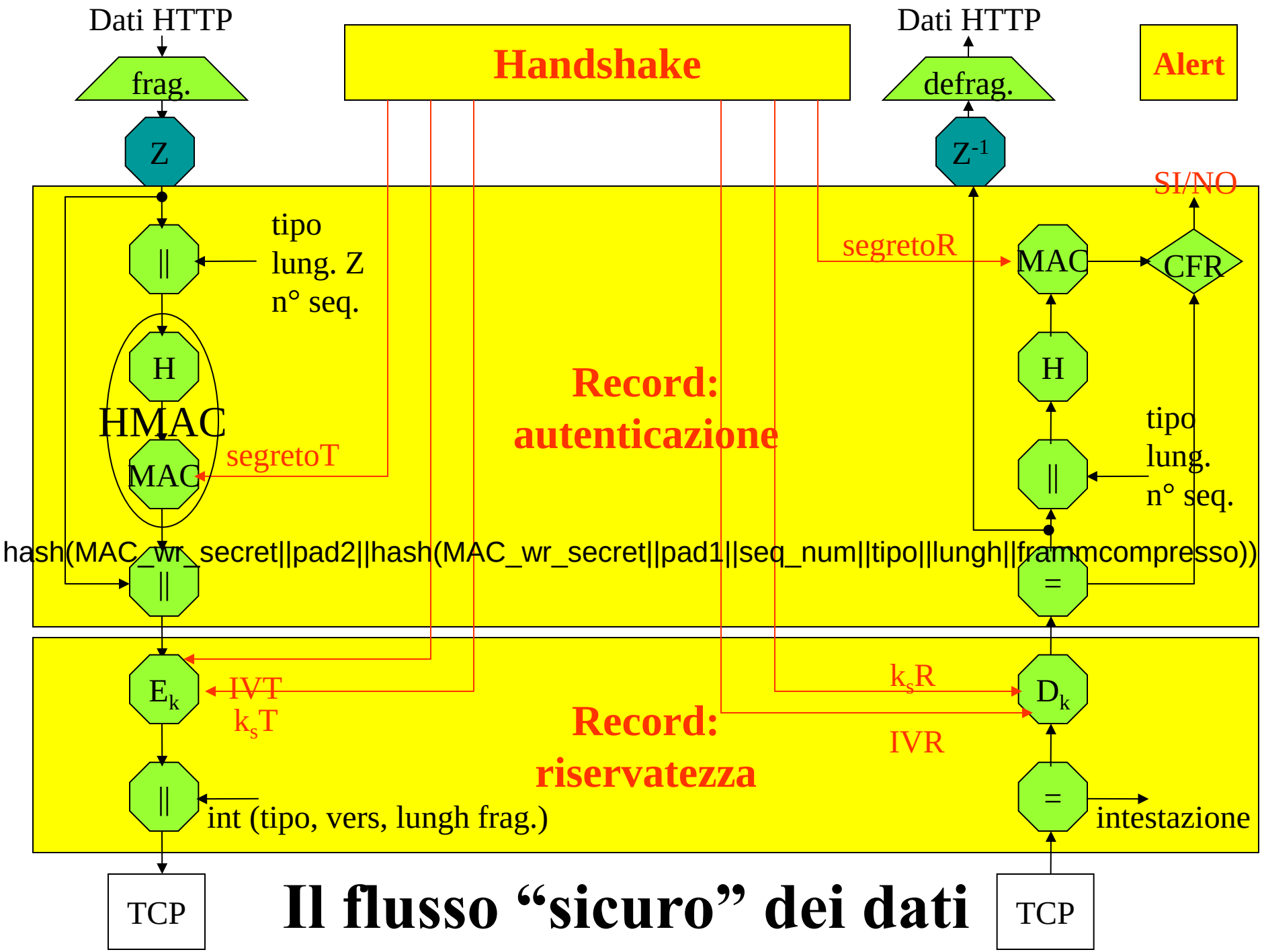


L'accordo sul segreto (Handshake 2&3): cifratura asimmetrica



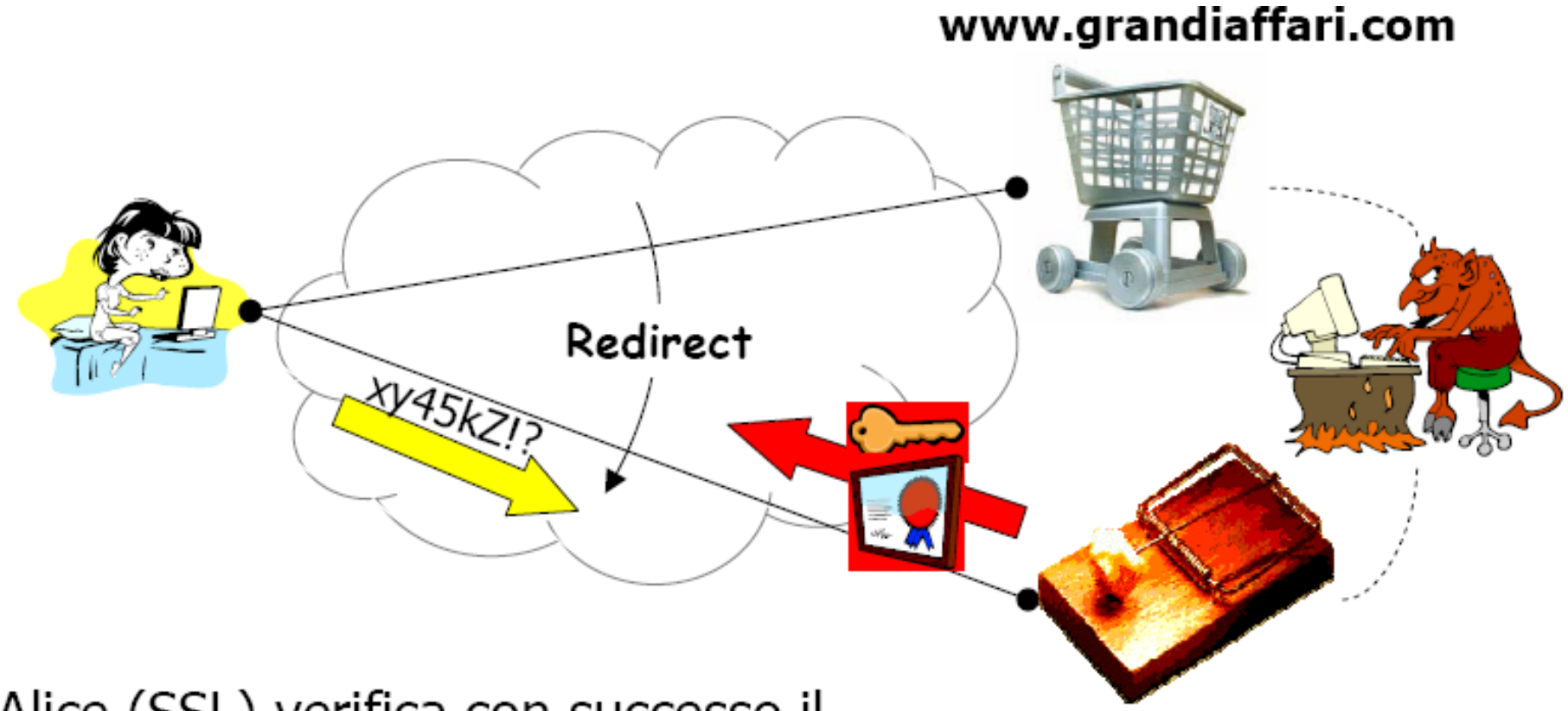
La verifica del segreto e la generazione delle chiavi





Il flusso “sicuro” dei dati

Il certificato è quello giusto?



- Alice (SSL) verifica con successo il certificato dell'avversario, stabilisce la connessione ed invia la propria PWD alla banca

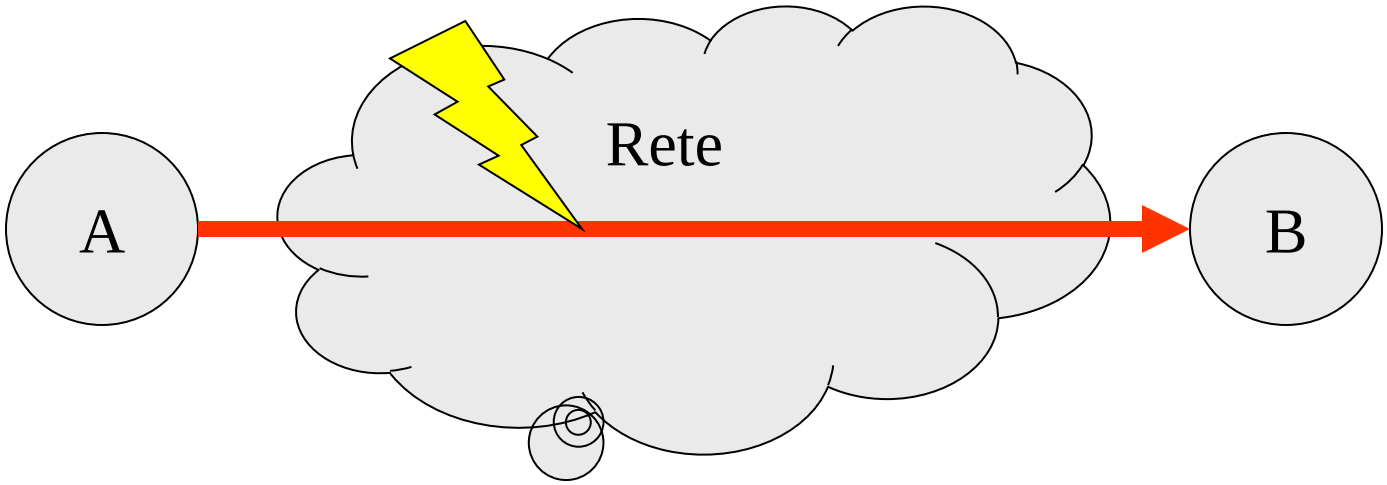
- Il problema è che SSL opera ad un livello più basso di quello applicativo

➔ È l'applicazione che deve (indurre l'utente a) verificare che il nome richiesto sia uguale al nome contenuto nel certificato verificato

- ESEMPIO: Netscape
  - Il browser *notifica* all'utente se l'URL specificato dal browser e quello contenuto nel certificato del server sono diversi
  - L'utente decide se proseguire la connessione oppure no (interfaccia utente!!!)
  - In linea di principio non è detto che il controllo eseguito dal browser Netscape sia sufficiente per ogni tipo di applicazione Web-based


Servizi di sicurezza
a livello di rete

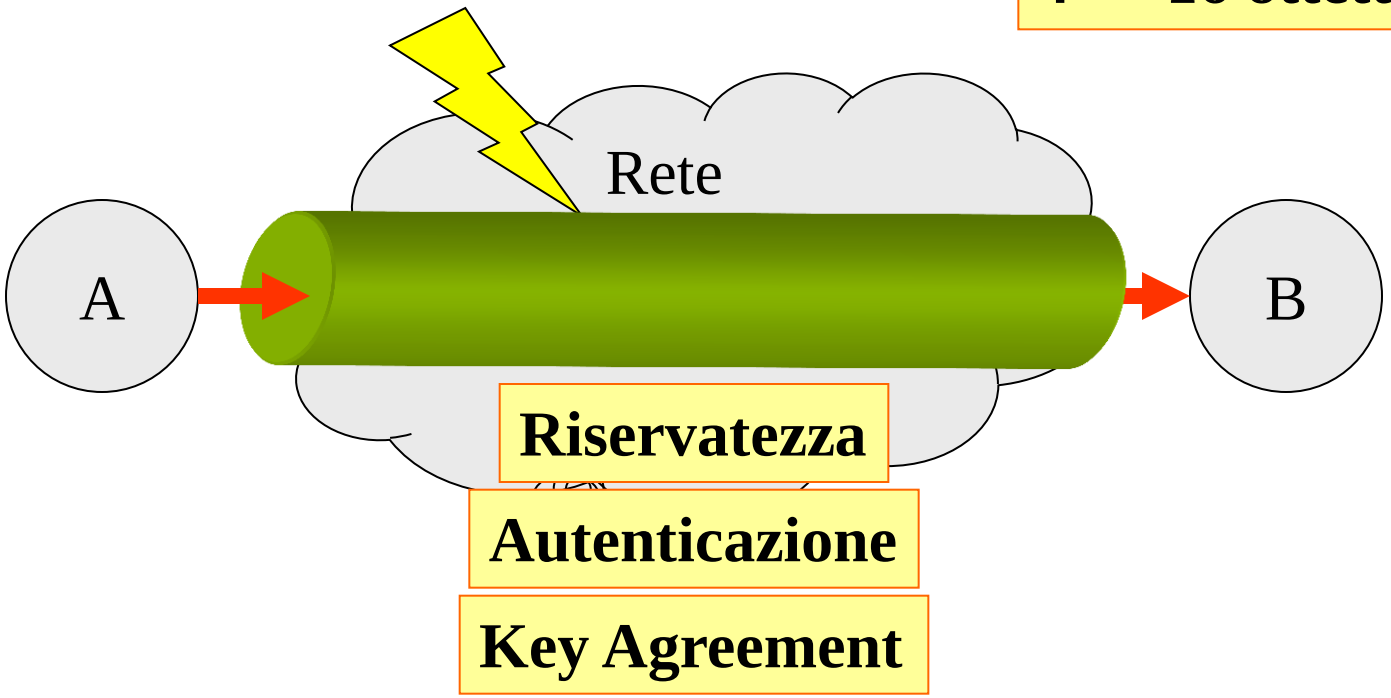
IPv4



- **Packet sniffing**: lettura dei pacchetti in transito
- **IP spoofing**: falsificazione dell'indirizzo del mittente
- **Connection hijacking**: inserimento di dati nei pacchetti in transito
- **Clogging**: generazione di un carico di lavoro inutile ed oneroso

IPSec

Indirizzi IP:
4 → 16 ottetti



RFC 1636 (1994), RFC 2401, 2402,2406, 2408 (1998)
IPSEC estende sia IPv4 sia IPv6 con funzioni di sicurezza

Riferimento anche per LAN e WAN

Servizi di Sicurezza Offerti

- Integrità dei datagrammi
- Autenticazione dell'origine dei dati
- Rifiuto dei pacchetti di 'replica'
- Confidenzialità
- Confidenzialità parziale del flusso di traffico

IPsec è costituito da tre protocolli:

- Authentication Header (AH) per autenticazione dei pacchetti
- Encapsulating Security Payload (ESP) per confidenzialità ed autenticazione dei pacchetti
- Internet Key Exchange (IKE) per negoziazione dei parametri di sicurezza, autenticazione e distribuzione delle chiavi

Applicazioni IPSEC:

Connettività sicure delle sedi locali via Internet
Accesso remoto sicuro via Internet
Routing sicuro

Servizi e Protocolli

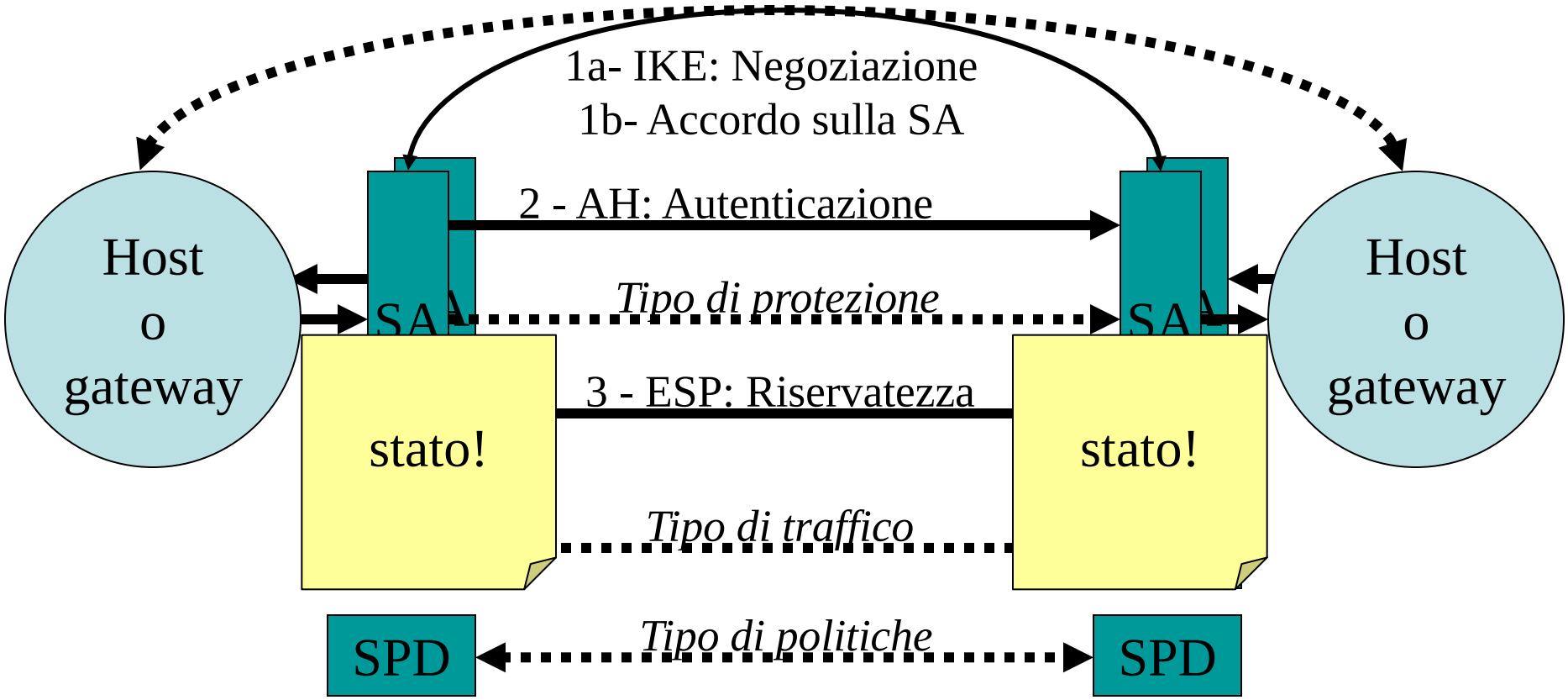
	AH	ESP (cifr.)	ESP (cifr. & aut.)
Integrità dei datagrammi	X		X
Autenticazione dell'origine dei dati	X		X
Rifiuto di pacchetti di replica	X	X	X
Confidenzialità		X	X
Confidenzialità parziale del traffico		X	X

Componenti di IPSec

IPSec è un'architettura per dare sicurezza al livello IP.

IPSec prevede:

- una suite di **protocolli per svolgere servizi**: IKE, AH, ESP
- un insieme di **entità per istanziare i servizi**: SA, SAD, SPD



Gli elementi di IPSec

• **SA(Security Association):** struttura che identifica in maniera univoca una connessione unidirezionale tra due interlocutori, specificando il servizio di sicurezza offerto con i relativi parametri(chiavi, algoritmi..)

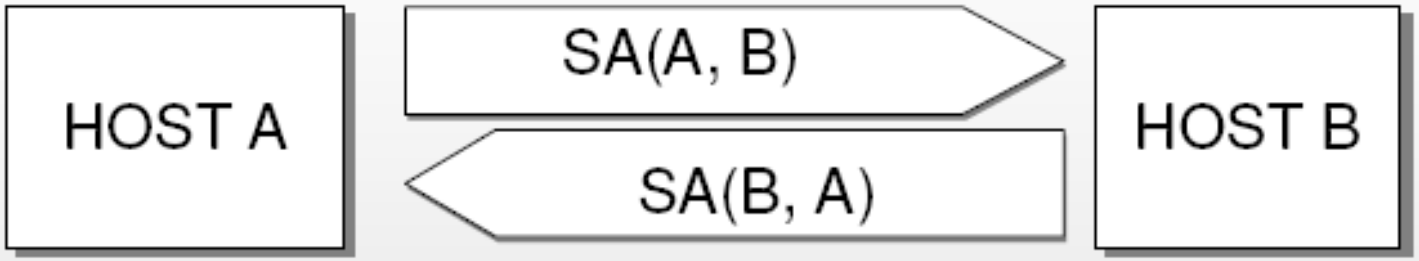
• **SPD(Security Policy Database)**: entità che esamina tutto il traffico IP, sia in ingresso che in uscita, per decidere quali pacchetti debbano usufruire dei servizi IPSec e per approntare di caso in caso il tipo di servizio più adatto

• **SAD(Security Association Database):** entità che tiene traccia di quale tipo di servizio per una SA sia stato espletato e in quale modo

Security Association

- Connessione logica **unidirezionale** tra due host

  (ne occorrono due per avere protezione completa di un canale bidirezionale)



- Una SA è identificata da tre parametri
  - Security Parameter Index (SPI)
  - IP Destination Address
  - Security Protocol Identifier (AH o ESP)

Valore di SPI nelle intest. di AH, ESP per poter selezionare la SA con cui elaborare il pacchetto ricevuto

Parametri di una SA

- Sequence number counter
- Sequence counter overflow
- Anti-replay window
- AH information
- ESP information
- Lifetime of this association
- IPSec protocol mode (tunnel, transport, wildcard)
- Path MTU

per gestire l'anti replay

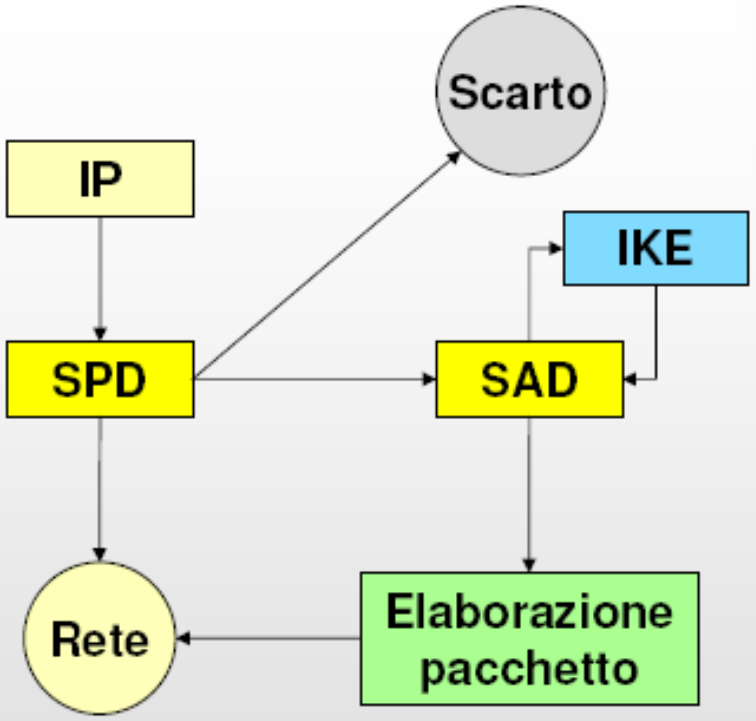
Database di Sicurezza

- **Security Association Database**

  SAD definisce i parametri associati a ciascuna SA

- **Security Policy Database**

  SPD mette in relazione una porzione del traffico IP ad SA specifiche (oppure nessuna SA, nel caso al traffico IP sia consentito aggirare I controlli IPSec)



La funzionalità di SAD deve essere disponibile pero' il modo in cui viene fornita è lasciata all'implementazione

Esempi di Politiche

- Tutto il traffico verso 192.168.2.3 deve protetto da ESP in modalità trasporto usando DES-CBC
- Tutto il traffico FTP (TCP, porta 20) verso 192.168.2.3 deve essere protetto da ESP in modalità tunnel usando 3DES-CBC
- Tutto il traffico verso 192.168.2.3 non deve essere protetto
- Tutto il traffico verso 192.168.2.3 deve essere scartato

SPD

- Un SPD contiene un insieme di *politiche* (*policy entries*) ciascuna delle quali specifica una porzione del traffico IP e la SA per quella particolare porzione di traffico
- Una porzione di traffico IP è specificato per mezzo di un insieme di valori detti *selettori* (*selectors*)
  - Destination IP Address
  - Source IP Address
  - Userid
  - Data Sensitivity Level (Classified,…)
  - Transport Layer Protocol
  - Ipsec Protocol
  - Source And Destination Port
  - Ipv6 Class
  - Ipv6 Flow Label
  - TOS, Ipv4 Type Of Service

Singolo indirizzo, elenco o intervalli di indirizzi

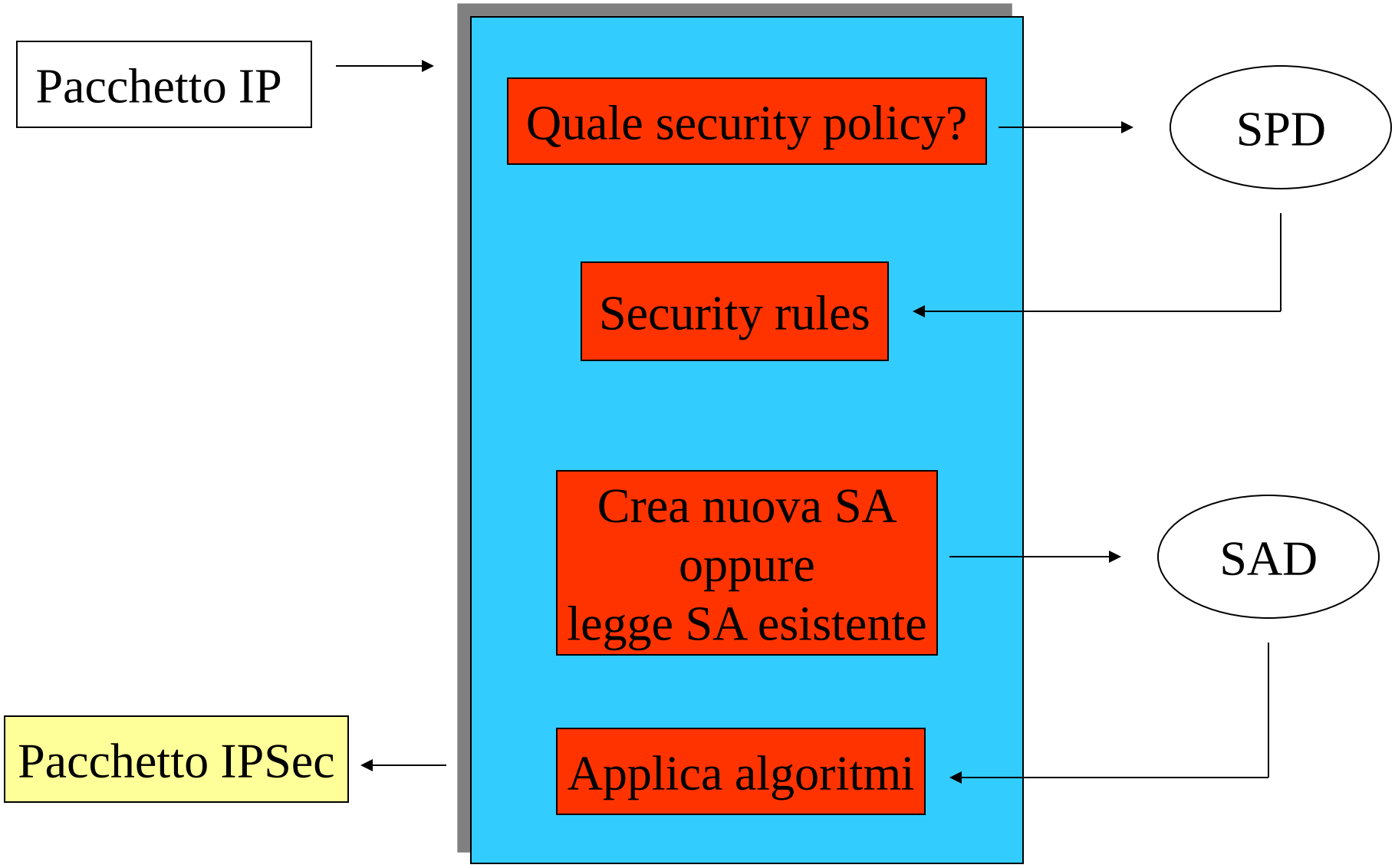
ident. utente ottenuto dal S.O

usato per sistemi che gestiscono la sicurezza del flusso di informazioni

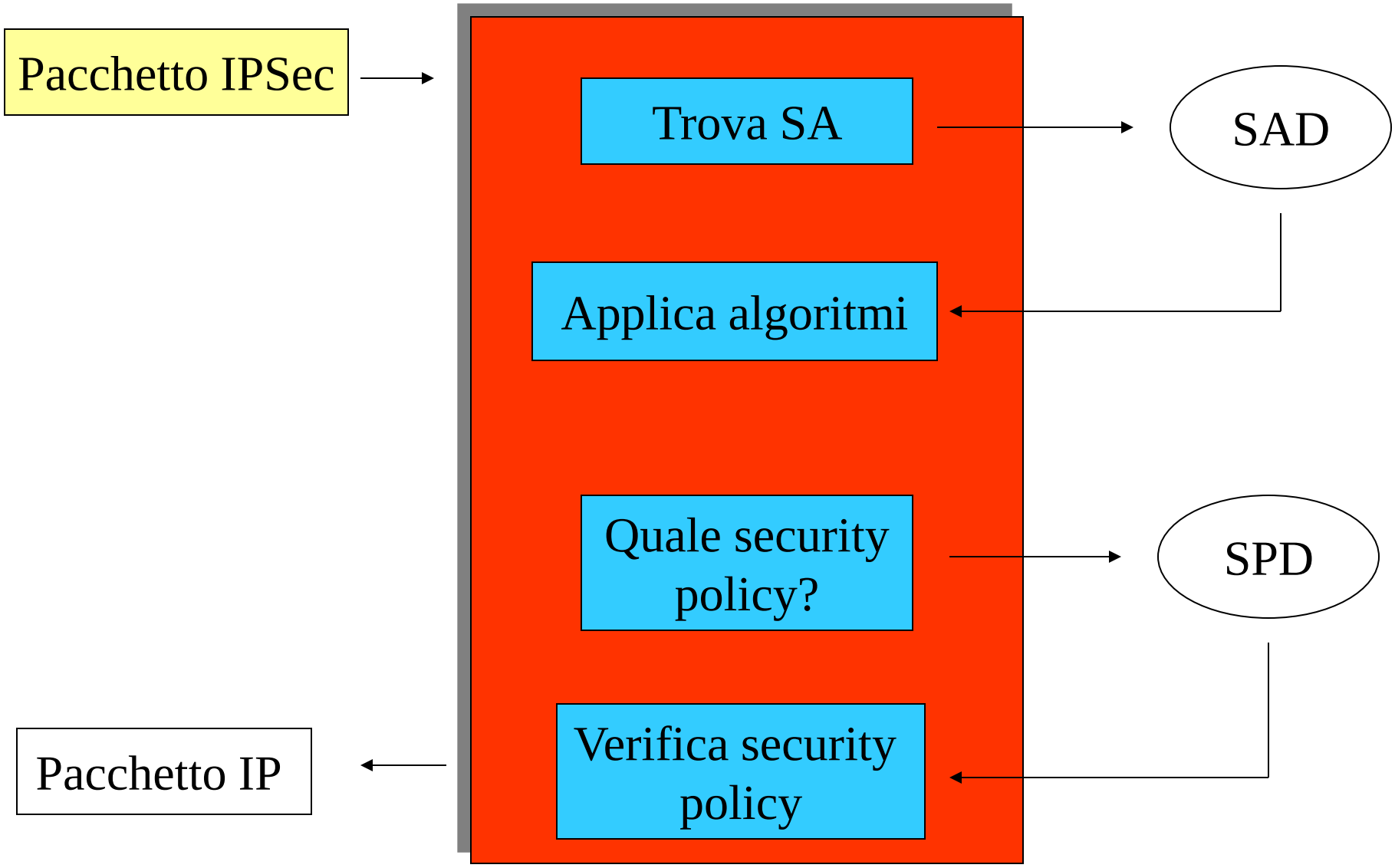
- Il SPD viene acceduto utilizzando i selettori come chiave

I selettori consentono di filtrare il traffico in uscita per associarlo ad un'opportuna SA

Invio di un messaggio



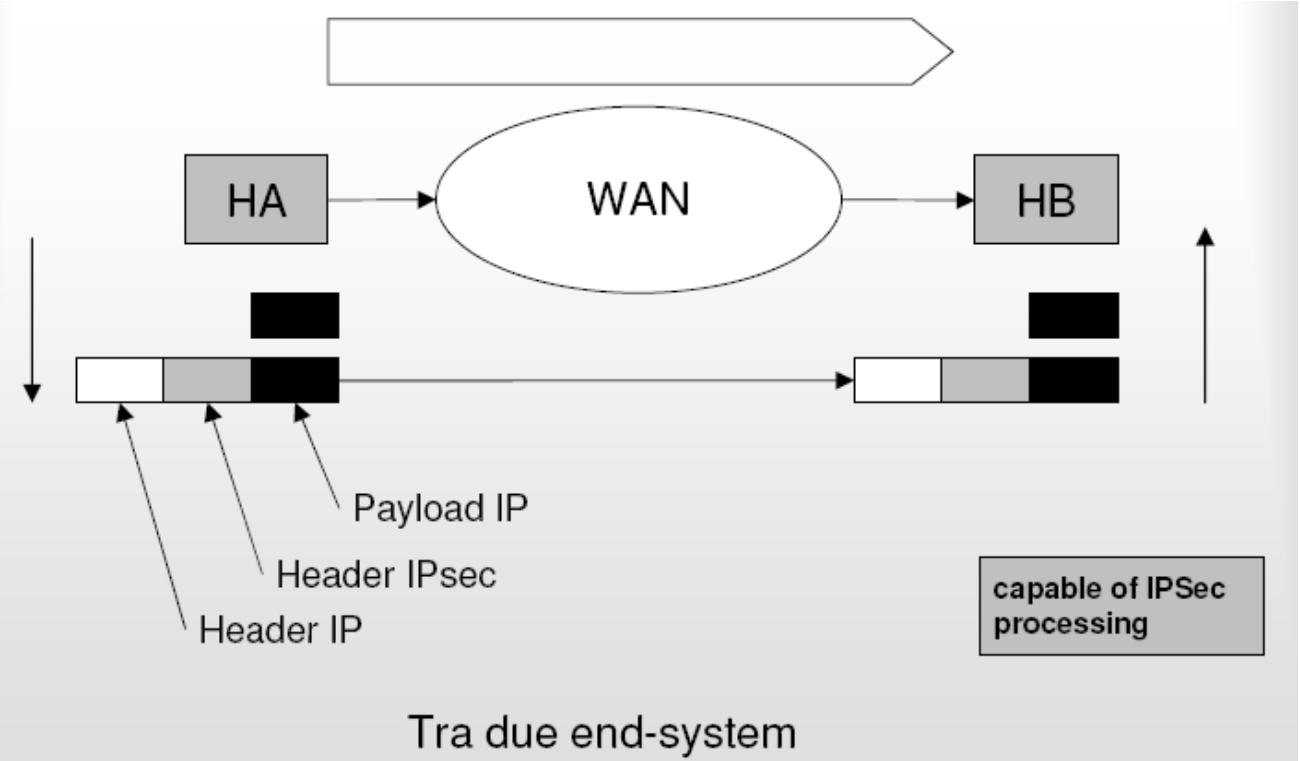
Ricezione di un messaggio

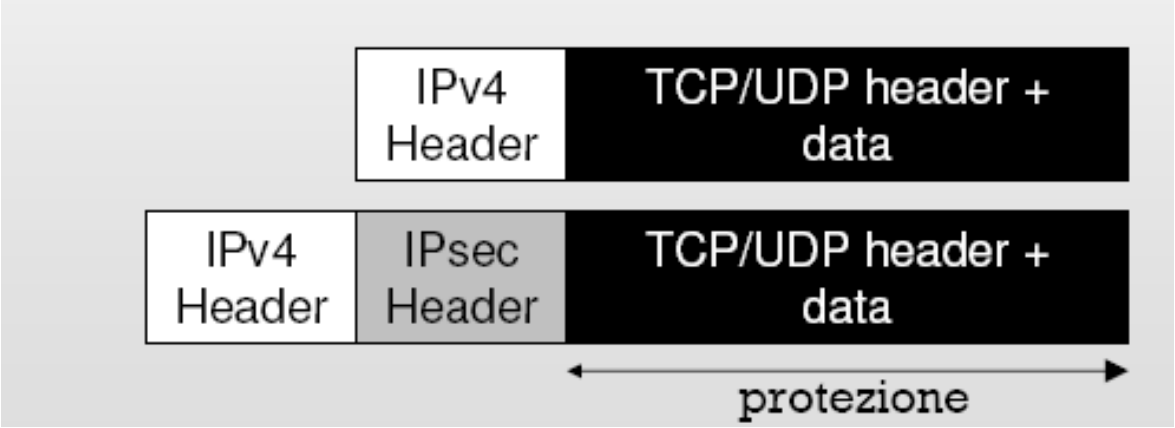


Modalità di incapsulamento

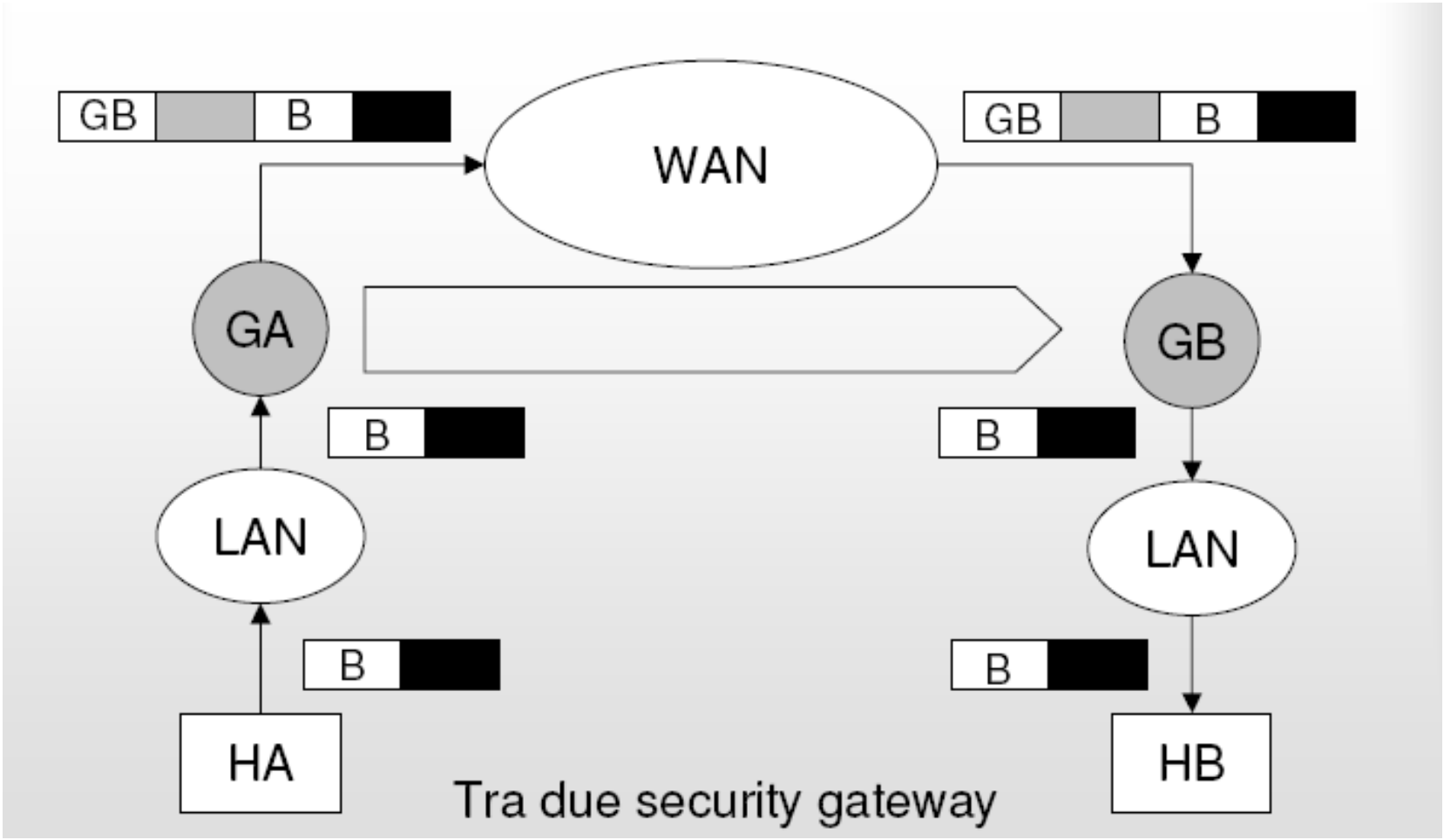
- *Modalità trasporto (transport mode)* → Protezione payload pacchetto IP
  - L'intestazione (header) del pacchetto originale che deve essere protetto (confidenzialità e/o autenticità) viene utilizzata per instradare il pacchetto protetto
  - Questa modalità di incapsulamento richiede che l'intestazione originale contenga indirizzi instradabili sulla rete pubblica
- *Modalità tunnel (tunnel mode)* → Protezione intero pacchetto IP
  - Il pacchetto originale (interno) viene trasportato da un pacchetto IP esterno il cui header specifica due gateways
  - Questa modalità di incapsulamento permette di stabilire un VPN che si estende attraverso Internet e che comprende host aventi indirizzi privati
  - La modalità tunnel è la più utilizzata

IPSEC Trasporto

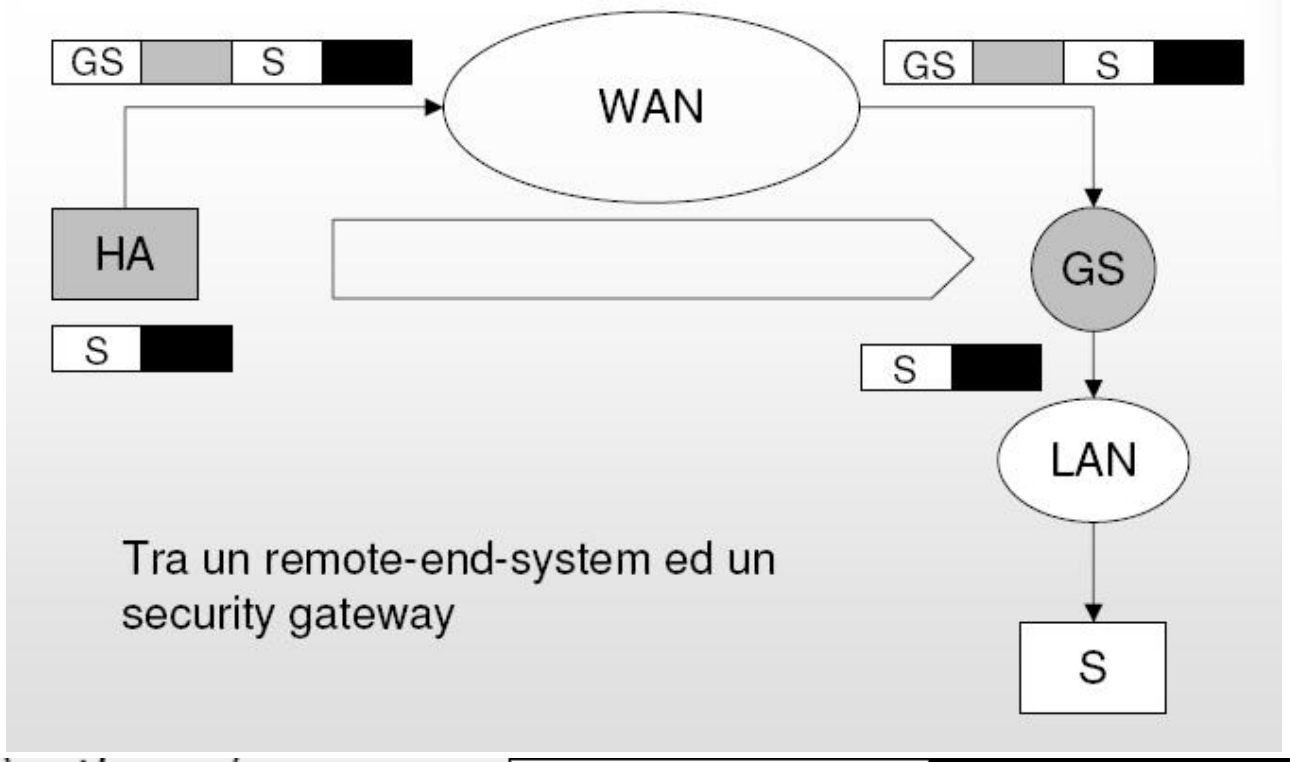


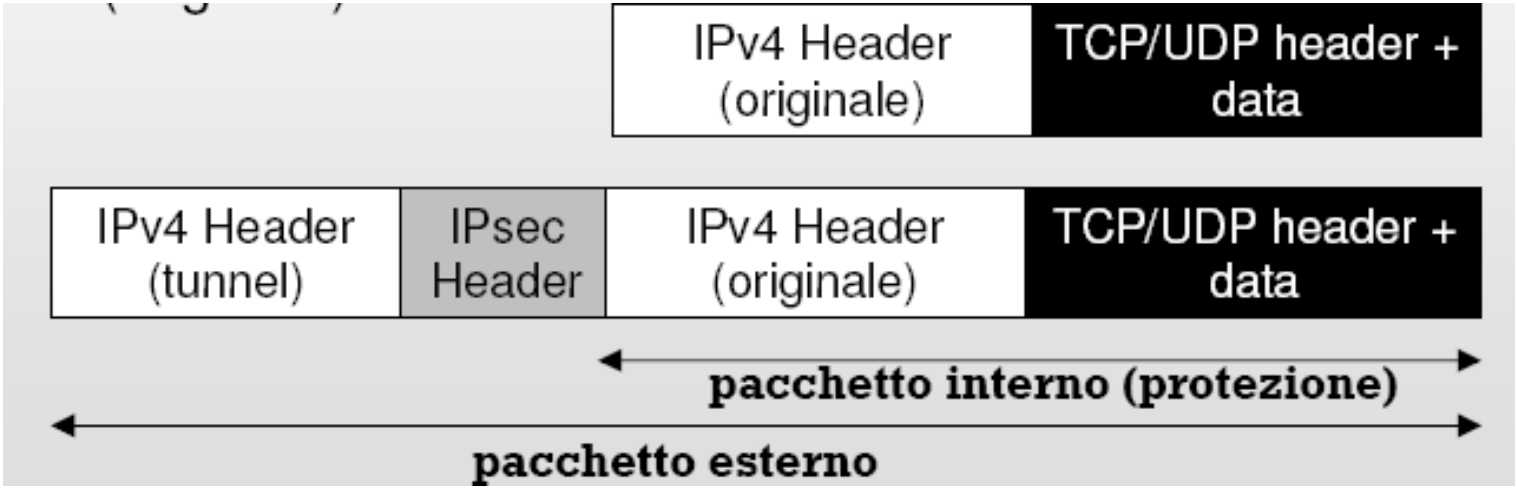


IPSEC Tunnel 1.



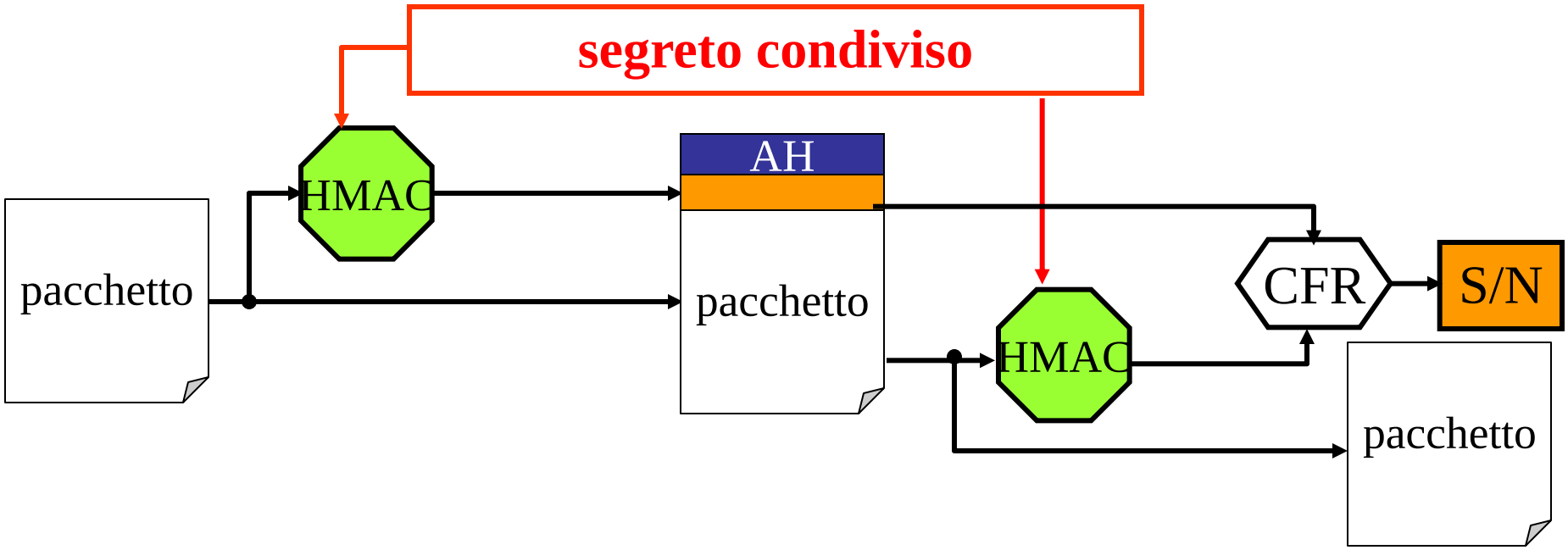
IPSEC Tunnel 2.





Il servizio per l'autenticazione (AH)

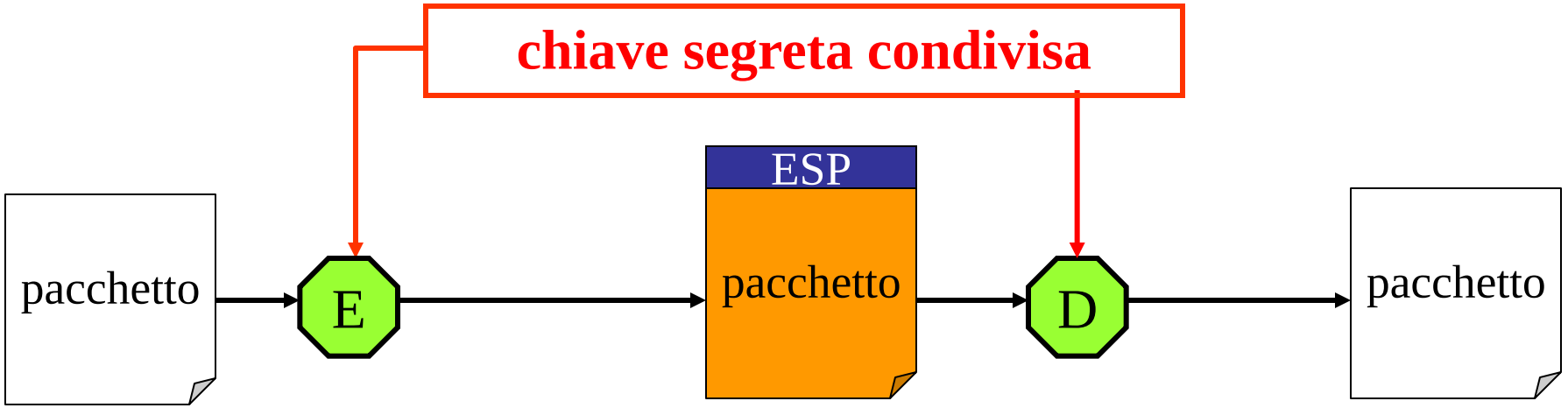
segreto → origine
hash → integrità



HMAC - MD5 96 bit
HMAC - SHA-1 96 bit
……

Il servizio per la riservatezza (ESP)

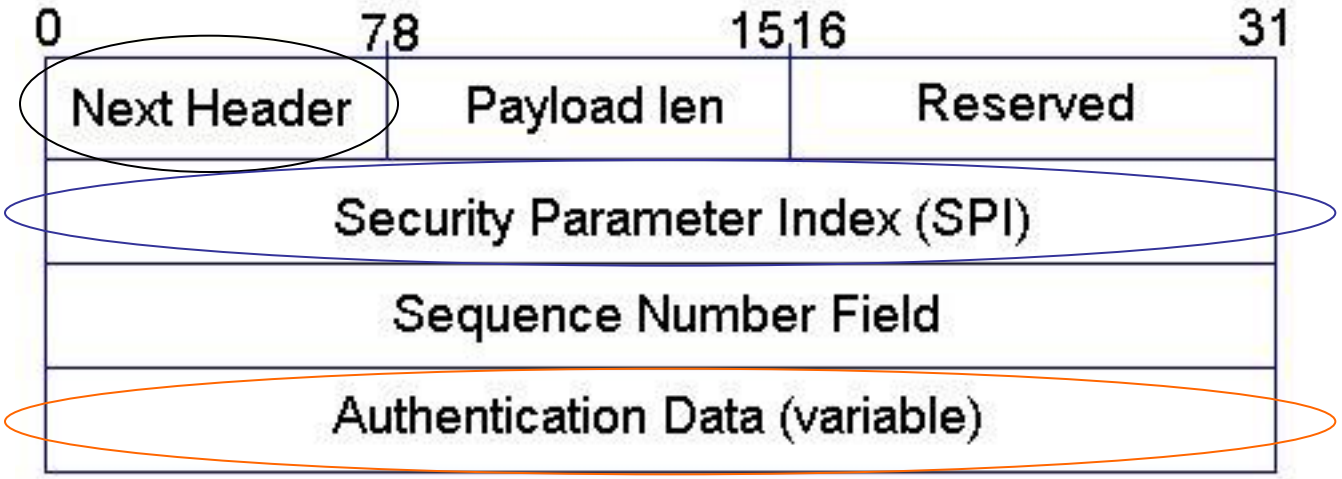
chiave → origine
cifrario → segretezza



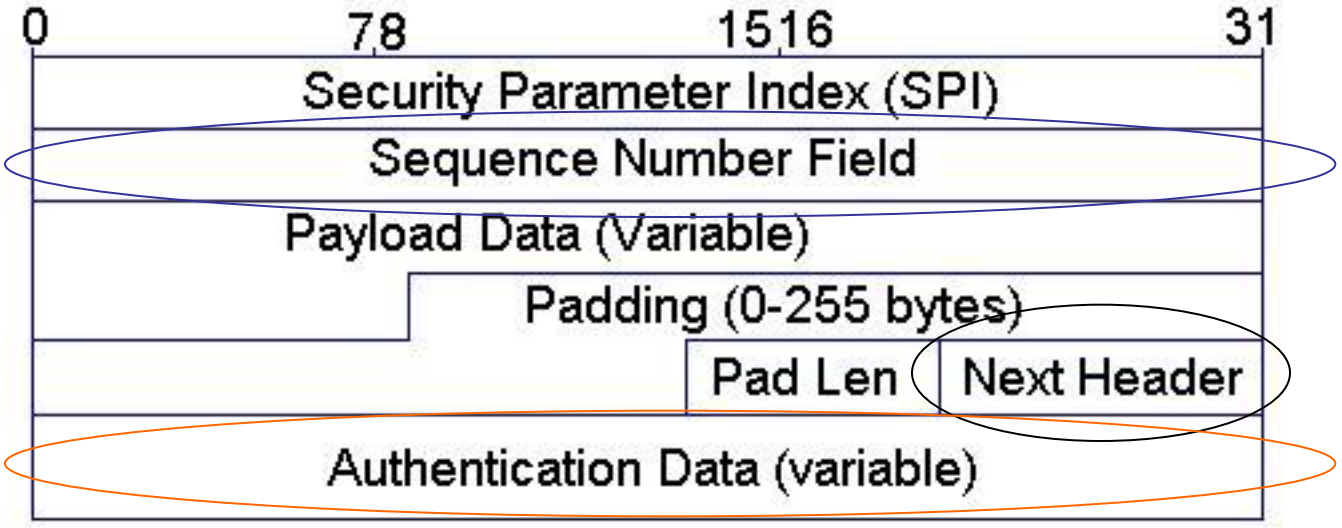
DES, TDES, IDEA,
RC5, CAST, Blowfish
……

Intestazioni

AH (Authentication Header)



ESP (Encapsulating Security Payload)



Campi comuni

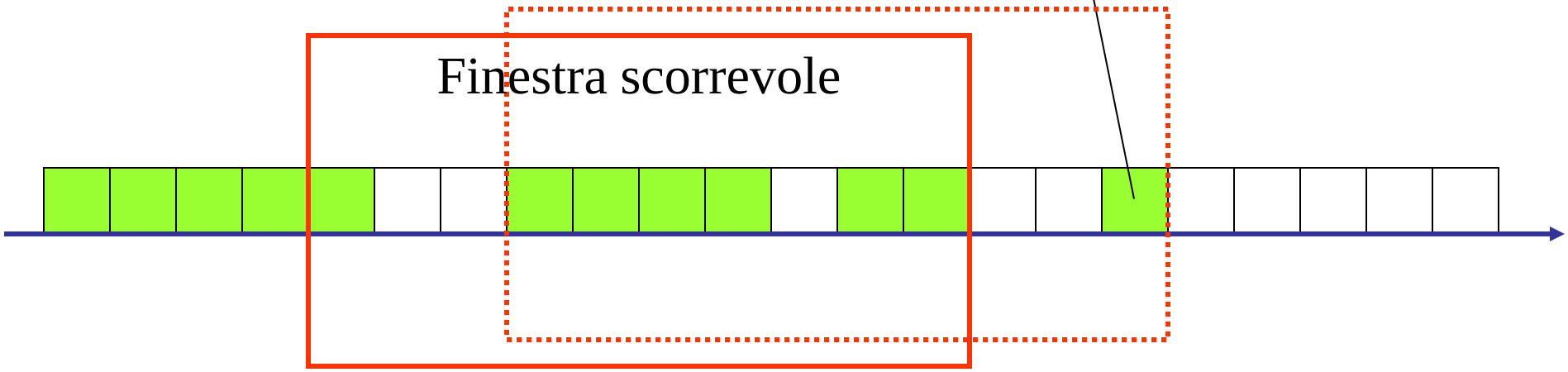
- **Security Parameter Index (SPI):** questo campo di 32 bit, insieme all'indirizzo IP di destinazione ed all'identificatore di protocollo, identifica in maniera univoca la SA relativa al datagramma.

- **Sequence number field:** espleta il servizio di anti-replay tramite la numerazione progressiva dei datagrammi

- **Autentication Data:** contiene l'ICV (*Integrity Check Value)*, che consente di verificare l'integrità del pacchetto in fase di ricezione.

Servizio anti-replay

All’arrivo di ogni nuovo pacchetto valido, la destinazione lo marca e sposta la finestra in modo da disporlo all’estremità destra



Lunghezza di default della finestra: 64 pacchetti

Costruzione di AH

1. Costruire l'header AH con Authentication Data inizialmente impostato a zero
2. Appendere il payload
   - modalità trasporto: pacchetto del livello di trasporto
   - modalità tunnel: pacchetto IP originale
3. Prependere l'header IP *normalizzato* (i campi mutabili sono impostati a zero)
   - I campi *mutabili* (es. TTL, hop count) sono poi ripristinati dopo il calcolo del MAC
4. Calcolare l'ICV (il MAC) su l'header IP normalizzato, l'header AH ed il payload
5. Copiare l'ICV nel campo Authentication Data

Costruzione di ESP

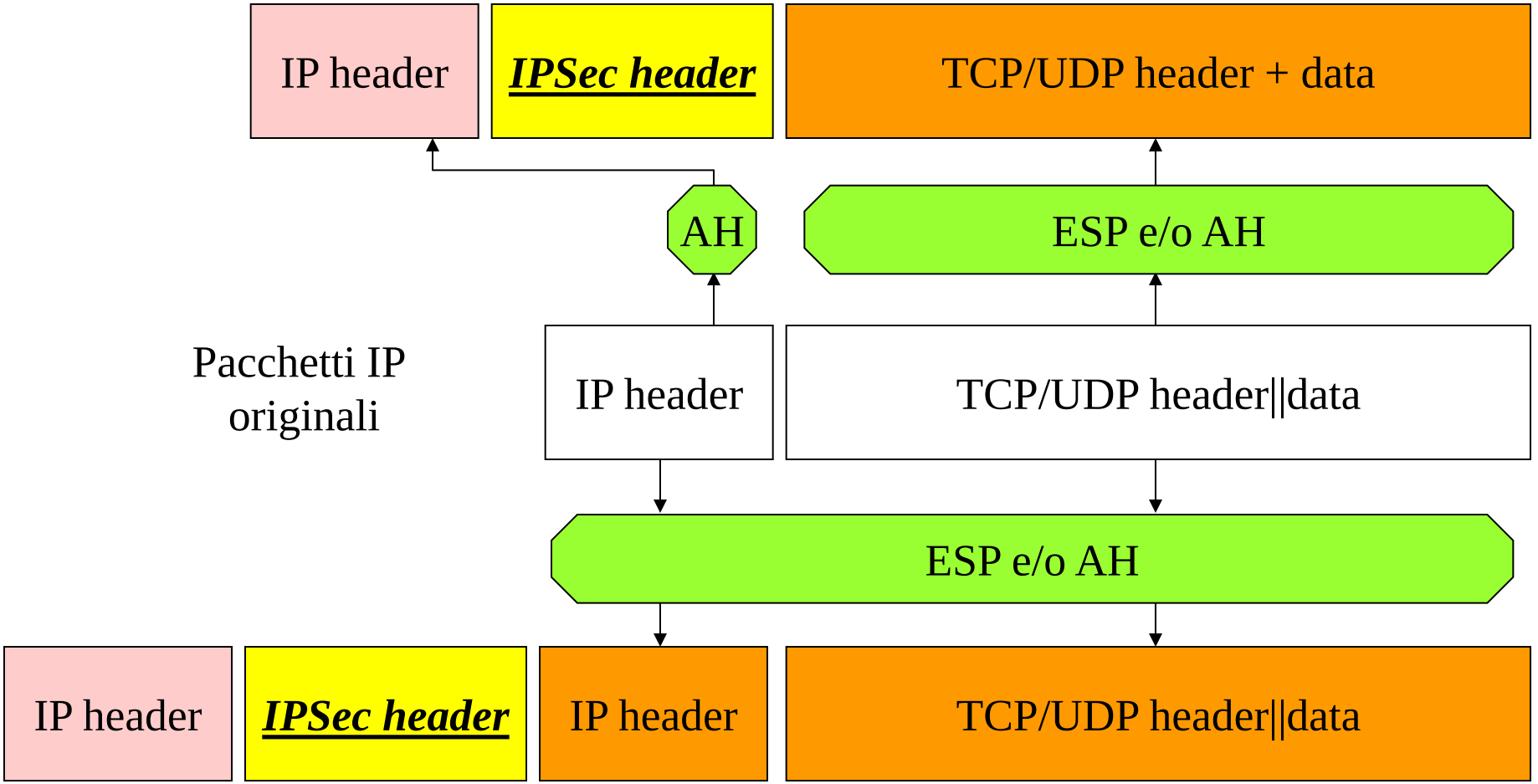
1. Costruire l'header ESP
2. Appendere il payload
   - in modalità trasporto: il pacchetto del livello di trasporto
   - in modalità tunnel: il pacchetto IP originale
3. Costruire il trailer ESP ed appenderlo al payload
4. Cifrare il payload ed il trailer ESP
5. Se richiesto, calcolare il MAC su header ESP ed il testo cifrato ed appendere l'ICV al testo cifrato (ESP Authentication)

Funzionalità

	modalità trasporto	modalità tunnel
AH	Autentica il payload e porzioni selezionate dell'header IP e degli extension header IPv6	Autentica l'intero pacchetto IP interno e porzioni selezionate del pacchetto esterno
ESP	Cifra il payload IP e gli extension header IPv6	Cifra il pacchetto interno
ESP con autenticazione	Cifra il payload IP e gli extensione header IPv6. Autentica il payload IP ma non l'header IP	Cifra il pacchetto interno. Autentica il pacchetto interno

Trasporto e Tunnel

Pacchetti IPSec con SA in modalità **trasporto**



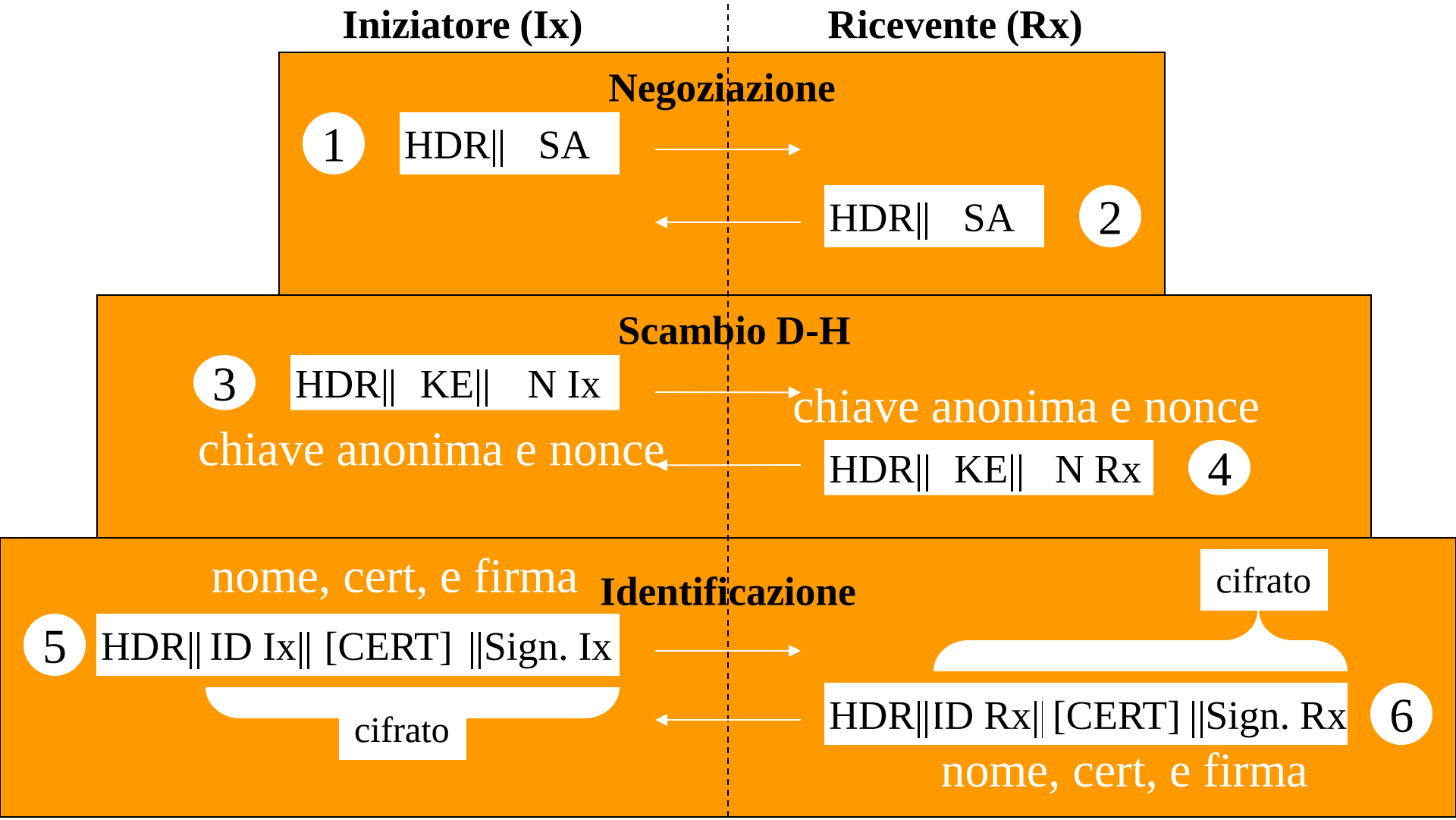
Pacchetti IPSec con SA in modalità **tunnel**

Protocollo IKE

IKE (Internet Key Exchange): meccanismo con cui due interlocutori possono accordarsi sui parametri relativi ad una SA
(creazione, cancellazione, AH o ESP, ecc.).

- *framework* **ISAKMP** (gestione chiavi)
- *protocollo* **Oakley** (distribuzione chiavi basato su DH)
- *protocollo* **Skeme** (autenticazione)

IKE fase 1: ISAKMP SA



IKE fase 2: creazione di una SA