

Reti di Elaboratori

Sicurezza nelle reti – email, TLS, VPN

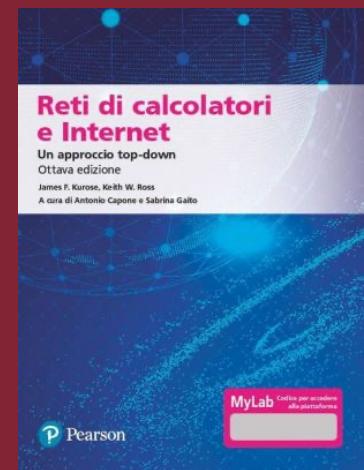


SAPIENZA
UNIVERSITÀ DI ROMA

Alessandro Checco

alessandro.checco@uniroma1.it

Capitolo 8



Sicurezza - sommario

- Che cos'è la sicurezza della rete?
- Principi di crittografia
- **Integrità del messaggio, autenticazione**
- Protezione della posta elettronica
- Protezione delle connessioni TCP: TLS
- Sicurezza a livello di rete: IPsec
- Sicurezza nelle reti wireless e mobili
- Sicurezza operativa: firewall e IDS

Autenticazione zero-knowledge

- generalizzazione di identity challenge

Find this guy

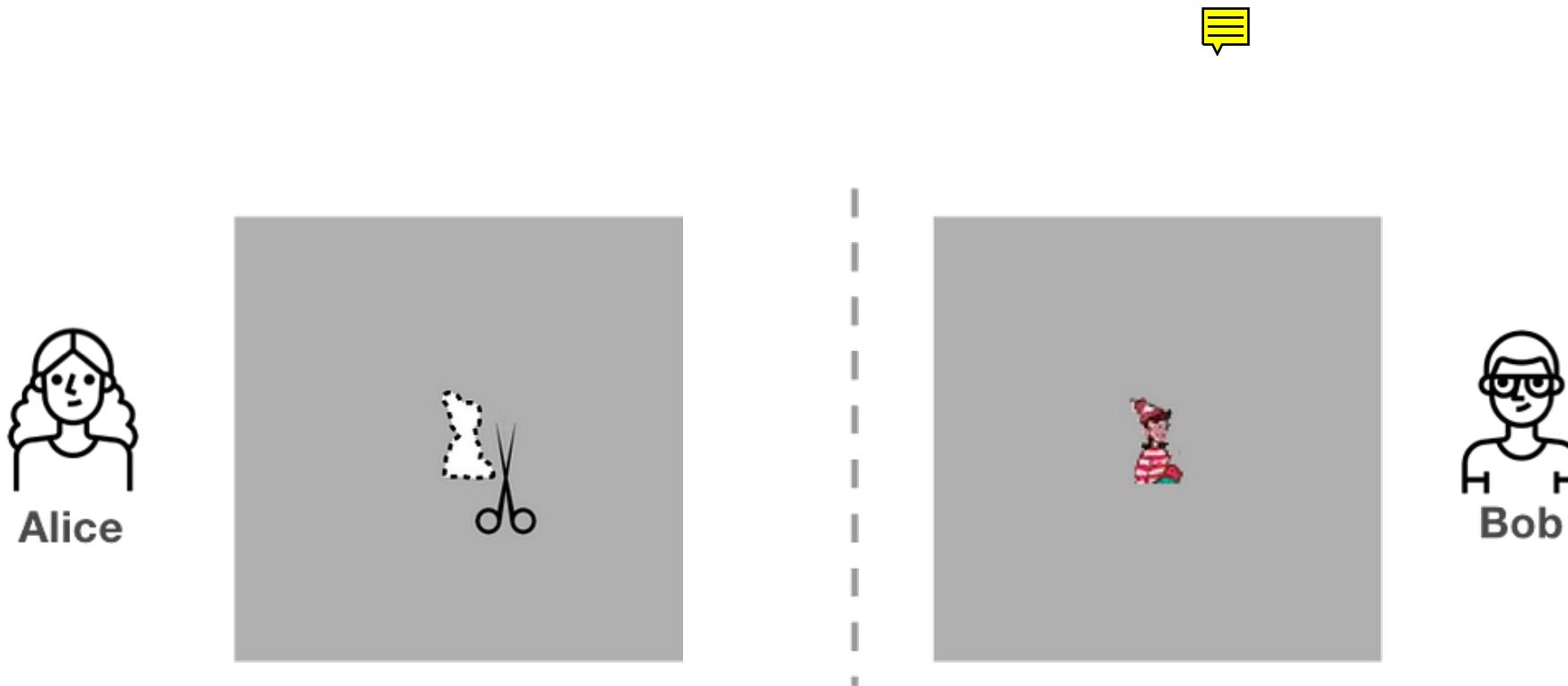


In this scene



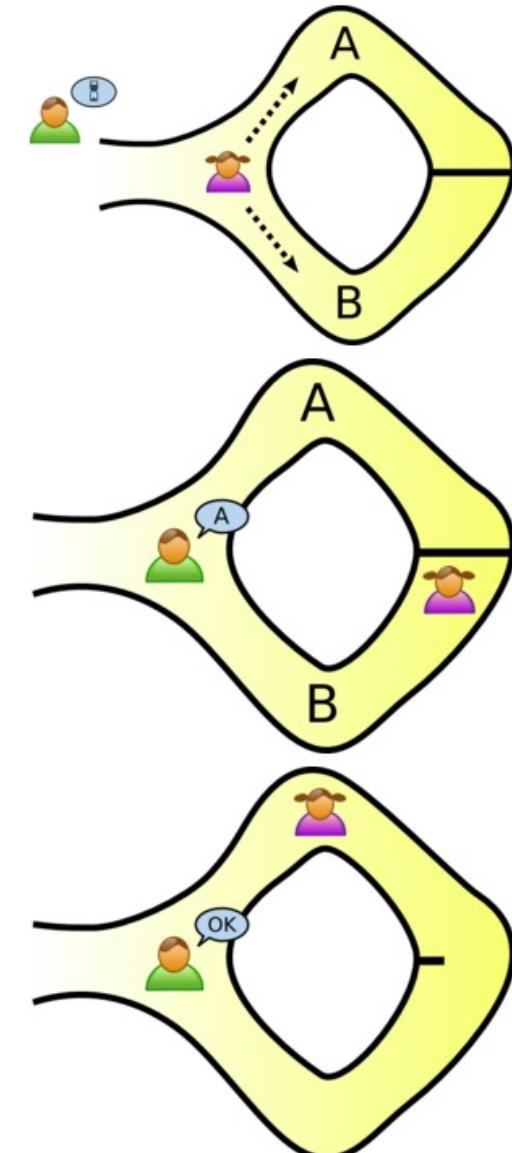
Autenticazione zero-knowledge

- generalizzazione di identity challenge



Autenticazione zero-knowledge

- generalizzazione di identity challenge
- https://it.wikipedia.org/wiki/Dimostrazione_a_conscenza_zero
- Si può usare per contesti più generali della autenticazione a chiave asimmetrica
 - proof-of-stake cryptocurrencies
 - distributed machine learning with private data (federated learning)

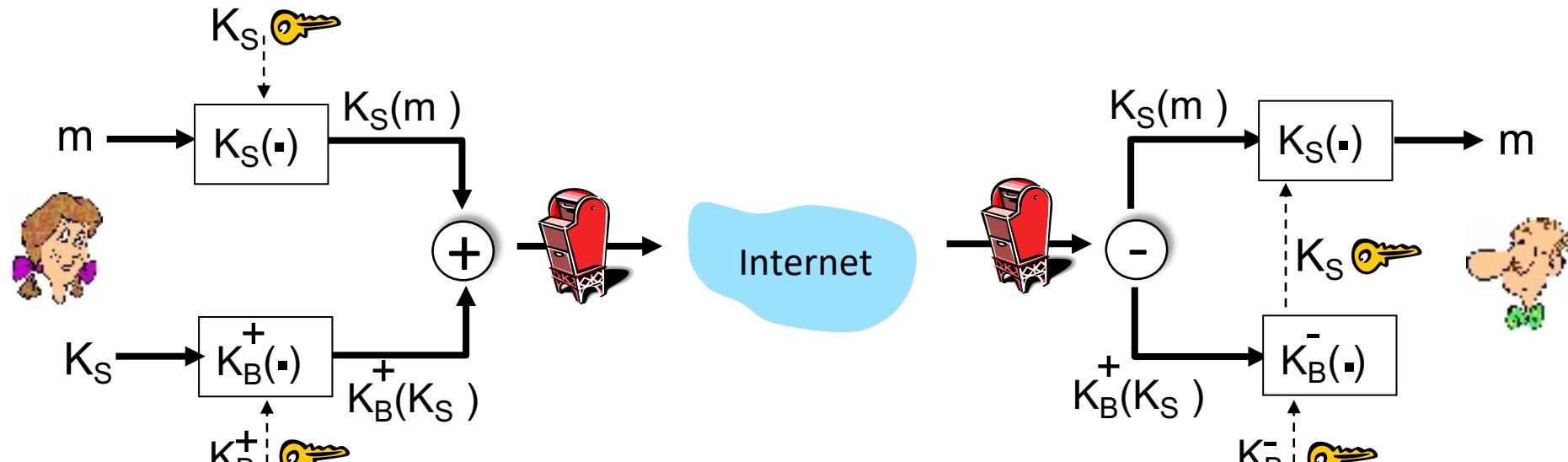


Sicurezza - sommario

- Che cos'è la sicurezza della rete?
- Principi di crittografia
- Integrità del messaggio, autenticazione
- **Protezione della posta elettronica**
- Protezione delle connessioni TCP: TLS
- Sicurezza a livello di rete: IPsec
- Sicurezza nelle reti wireless e mobili
- Sicurezza operativa: firewall e IDS

Posta elettronica sicura: riservatezza

Alice desidera inviare un'e-mail riservata, m , a Bob

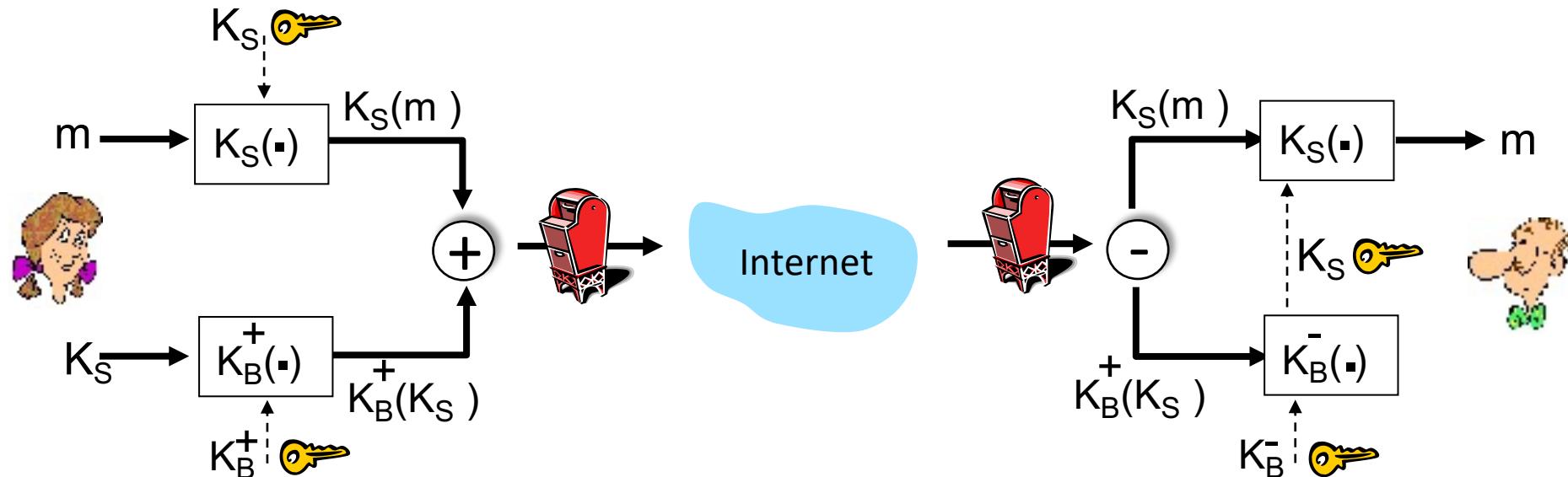


Alice:

- genera una chiave privata *simmetrica* casuale, K_S
- crittografa il messaggio con K_S (per efficienza)
- crittografa la chiave K_S con la chiave pubblica di Bob
- invia sia $K_S(m)$ che $K_B^+(K_S)$ a Bob

Posta elettronica sicura: riservatezza

Alice desidera inviare un'e-mail riservata, m , a Bob

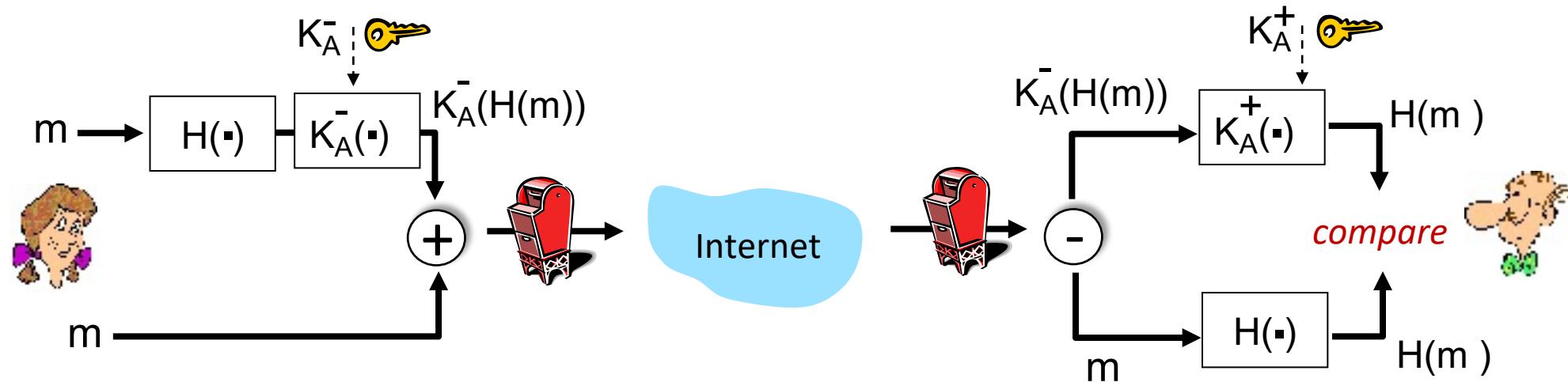


Bob:

- usa la sua chiave privata per decifrare e recuperare K_S
- usa K_S per decifrare $K_S(m)$ e ottenere m

Posta elettronica sicura: integrità, autenticazione

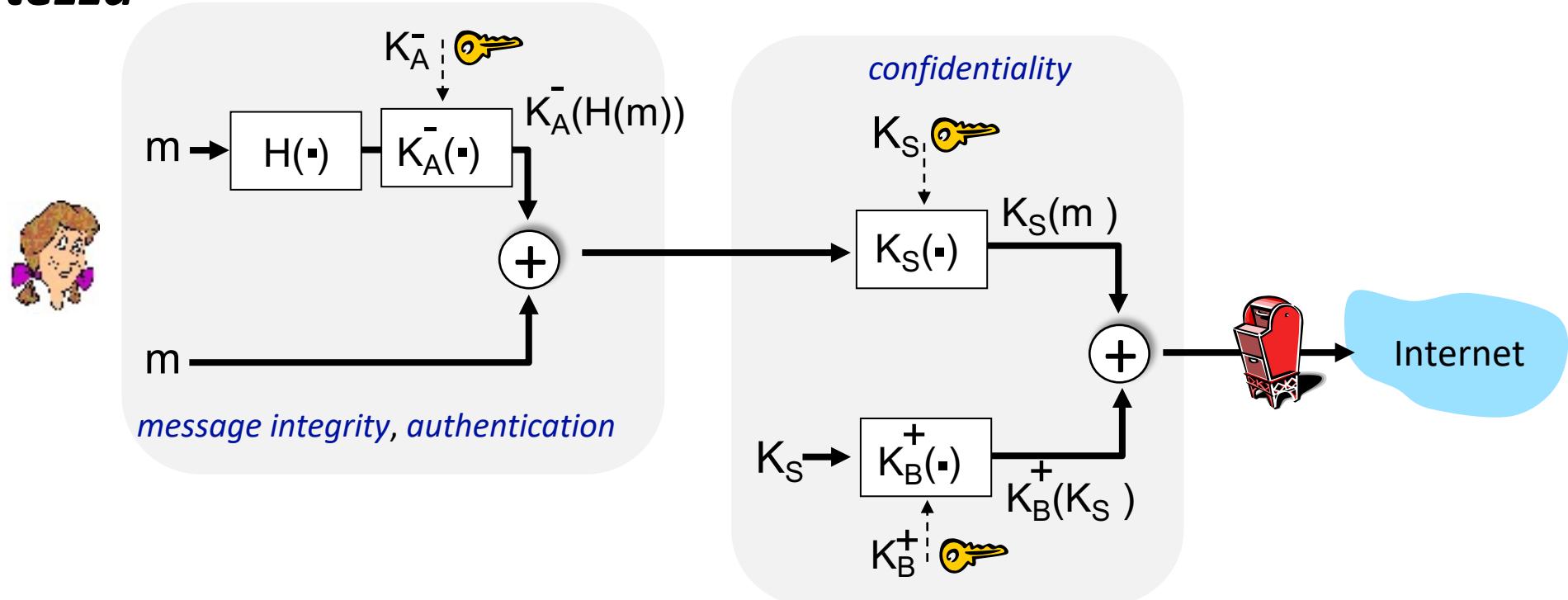
Alice vuole inviare m a Bob, con *integrità del messaggio + autenticazione* (*non è interessata alla riservatezza*)



- Alice firma digitalmente un digest del suo messaggio con la sua chiave privata, fornendo integrità e autenticazione
- invia sia il messaggio (in chiaro) che la firma digitale

Posta elettronica sicura: integrità, autenticazione

Alice vuole inviare m a Bob, con *integrità del messaggio + autenticazione + riservatezza*



Alice usa tre chiavi: la sua chiave privata, la chiave pubblica di Bob, la chiave simmetrica creata al momento

Quale sono le azioni di Bob?

Esempio

-----BEGIN PGP SIGNED MESSAGE-----

Hash: SHA512

Dear Alice,

I hope this message finds you well. I just wanted to let you know that I received the package you sent and everything arrived in good condition. Thank you so much for your help!

Best regards,

Bob

-----BEGIN PGP SIGNATURE-----

iQIzBAEBCgAdFiEEz1lN2l5C7HCX9fIfB6IOYVL63q8FAmCh
Jn0ACgkQB6IOYVL63q+Npg//X+Itu+jZaRvD8WgRLkF43bxF
f+GKrmJq3e0qwGsc9tSupz7YcAJQibJf+4Mw1uLfbVgE/hJW
LOj9MeeuMtyiKvZVWfcE8/1x7hLmj2cJyKmED3+o9PcZZxeJ
Cil+Y04NRK2+/4ErN4J1f4E7yWadD8Xr505j11rLSM5eB6M8
IgFE1AvDITK5EAP1A2QX9KzgI+QDLrmHYz0C3q0KUcUP6IgR
6UOss61kfVzXjKohPX2Aa55mPnWq3KjFCwAFw/Jd/+UshMn1
rXdc5jz47IvBbiZmpoG51jCYeP30gKjKj81hOoCLOv3z1X9O
ONsy0BwdjBDvHGRmAXxwGmL8Y/0GzJftI+9t0pTK8N5Wd99
vG5Q5g5w5N+1E6AOrqnMTTQX91docnf18ngn+VT1/TwnfCg6
n1FYEL26fT6T1T7UeJ6UfOG6FQD6jE3BJAxGTz1NWV7CNFr2
9e8aQWvkBLGgckBJ1+Jp1n8CPfhGdxIJLp1NTxGzv+NtYD
tNNZb+Eg67XgL99F1G8/C1Wwxsyj5I5+Bveapv+BYJbYPsb/
SVhQQD2Jg1007b/PmL1Z+z7J44OKxTgT/8+RdUJg6fyhTx6N
q4HFFu+QyN13s+qxWhsSv6T+WWdRzU6ZF6Xoj3mjB4ffOCQs
U7V4g6WhIYkVgmYn1IWphZMNuXJXBoo7UI8==Jaxf

-----END PGP SIGNATURE-----

-----BEGIN PGP MESSAGE-----

Version: GnuPG v1

hQIMA43ZNMN81fi0AQ/9Hcs8+qsUvekm+OQV6ELUWh3HoQE6tYAAmlVpv6mYV5Ng
3XMzoEi0RFr5KUMJ+jNHy50NJAKclM2KcuM71MM1tY1GybtfW5HDYhXes6esu28S
zgAuizoQtqDbmDVCLtloxne35aQtch1djFh0cpdvNWXJ0WAlRx8pltUoYhsTf5F6
MhBhM14MHZaSU2Pj+iSsipW50zTbX5C+94ffm2gk4XA+t/b0NtyPq+jzEZeBMVhp
50/1AMTKEBW3Kt6QrYou2IB0rjYT1hstKaGuExde5cShUnH0QmTDyirH7qUx5Th3
oKsEm4wWwKwlZvr73cVvrSJ/MuwprcStXfmceGUtnKRN/qaQtk/tHkILbYGghkm
F/JtLbroCtgFZW60xWHDraiFJx/7fvXPNDMkaNPNPg5Xluu9x0ejL3i9VdnP+uObq
MNnWUqIj4f61uw93PVileCaMywfW8cTyICAw1Mp03Vxk1/ErqHT3RYS6H7C5a+X
1W7jd+JTtaxnIyH9r69ejUDPgHted54D1uIp+RWak2rEH0wggNeLo1zDKh77kMzw
kmKxyy1mjpG+Y0d52r4eXm1WYhTdXW2DupNLz9mYHDFb0wTEBSL020p85Sxei7K
ZylLbwvqFtYBPUaaPUmqqgWukP2zueqC/FnNPrFSrtayG4n6L17c2N/hx/VT4kiF
Agw0tayCikVbL/ABD/oAeyCEti51khFsYs0pdYHvUXwZHte0Icv15mIi4ko
0vgmrlUh5ehayL/8IT9k1R/A4VB1PiLBt1EKv05xMoF3qejax6sY/AyBeijPT9W
ZdhqcoqYFa2WYg05+xegpTF0MMldUNCUHe53SX1JgB1bLnwMme3g45aVzLMtVtW
N/7QTWVZKPEfwdxGN2AfM2WQ05WzcfMRU8+ntvkUT7uYIxIMg0uro2mL0UcZoCn
b5ZDT4pMaM8Y51Bfdpi9gHREZTMfh3EGCxkt5oVLC7y8UtvCTOTrPIWdbLAZed
+g/sat5if20yUk0v7h9s/ucx+jC330dskw0AhRHU6Zeip0b1zK+x4qbIBjkB0kc
1jYP3Yj4Q9qMj40h6iy2XVkfFBNCrwzXkogpQN2VeLarK3450f3Gt15yZpxjTc17/
8GslxnxWq3aPNQ2nq8crni71WL/3HquNzdMBzDcZus9N+ti+uTPJHogrg1y10Qo
upP543PVZNGYWXJ10Usy6U7AgcmsMBLkC1qkZVeq+v121C50dRITXGAsVC0WjkX9
PN9KdH0B+6a1R2np5S4s1hanudtooMMVp19xd9wTz1nMiC5hAbsKH1aF+SrdihF
/Cye5tgEs7SkpP6Qd1i1kPApVz71CnQujVukV+gexJzKNZTXTE8WgwG5Wdv2ytLp
AR/EHNKuImE9M5UdxRSbbFIueDn8mrW/9wKtTyTrwSsSjbZ01bc5No9bIjSYZxf
6JMrU8GirtrgTBfz13HSNjntFmoQ3MnuuYjbyebAwLshGuvRLMUy/X0k+Nkubp
rGpTTa28cjnP4NtBf68TfVQ/ZqlOe37Ljt5Y9zLWEz5R2md+gor1CXXJjV85/r
0iK/sN8VNxDVvU7j20iNwgvjvKaaE236megmW08pT1b/keank30F8vx6Fyc8
EQqeAL/uR1lg10UiRD/0CQHygvb2eCNV7D3gogubK+3zZN4r//rEH1w89Cgey3V
3VBib1HtoEA7o+X+bf1S0neZWHiu/tXo3Akqay4nTt0rf3r00+QaSebWn771CkMq
dji81ztunjoGNjhCruA+HMkqwyFa7PJ2wib6WhIhr0j7T6bq/5fk0pCwhiLWZk
0rQuDbjZJGfAyH0/fuUnShqg2jTT38Kh4TWT1A5+n3sPwK+GLxhexK2zCkz9XiCz
cKuWitgPoYHnsD/0xjiz1fD/pA3tFsgTc3H2ghw51m5Vw0e1mQRP2Lk4oAv/HtQ
hRYCd9X100ZXQx3y607S8sfvUq41VN6uPjVMf8jqaGiUvvJKUqHZ7ZqPjCVaj
JespRVFETfnmtaDTTrRfYgYKLH/nxeuHTV/vTwirp0AmPMLEkMfaBxTxPF9qgnC
RQDjie6kDpgjzqtUuzLazezmoVWD5517MG7KbP/21MG2szKrzcrnTfxno28u60J1Z
NOrJK6XHw2CWH1wiTzMGEvCIv0Vh1gHAu1SewLgx1CcvxhGm84TNT/GmMz4IZ
yHrA33hx8MPrvru+AAlaB6JTHbM+Z7x1gGLTRdbq+0Mehb4HQ/Ekt+/Lhnkj94nw
/diyHo0jSt8Ktaej1J1k5X5uLh+QiQr+8nhJM4+RhmJ71HxckfdmihGQRatmyoV
08N7VGP1rM9hCf3b0zQJKo2jeaNtiraADahDofIW7jm3+AemfzUTteM5exUdIx+7
ozxsK57gtHyd4KRPXc8+a70amhdRpfcicoX9aC7Kk0FECZBswgz5klFgGhfLL76e
3KcoiaWJBFYdiCxzWQk3s+jMj1wcLgWA3DRIn18MRK4/sb33VYw/xS
=vdvU

-----END PGP MESSAGE-----

oppure



Sicurezza - sommario

- Che cos'è la sicurezza della rete?
- Principi di crittografia
- Integrità del messaggio, autenticazione
- Protezione della posta elettronica
- **Protezione delle connessioni TCP: TLS**
- Sicurezza a livello di rete: IPsec
- Sicurezza nelle reti wireless e mobili
- Sicurezza operativa: firewall e IDS

Sicurezza a livello di trasporto (TLS)

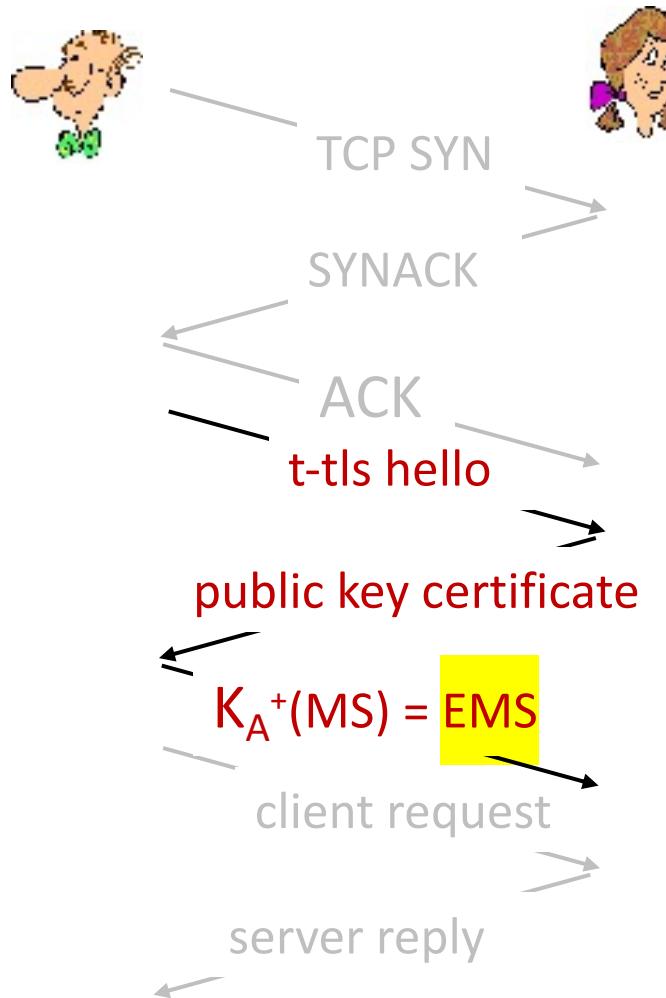
- protocollo di sicurezza ampiamente diffuso al di sopra del livello di trasporto
 - supportato da tutti i browser, server web: https (porta 443)
- fornisce:
 - **riservatezza**: tramite *crittografia simmetrica*
 - **integrità**: tramite *hashing crittografico*
 - **autenticazione**: tramite *crittografia a chiave pubblica*
- storia:
 - inizialmente sviluppato da ricercatori nei campi di programmazione di rete sicura e implementato via socket sicuri
 - secure socket layer (SSL) deprecato [2015]
 - TLS 1.3 : RFC 8846 [2018]



Sicurezza a livello di trasporto: cosa serve?

- costruiamo un protocollo TLS giocattolo, t-tls, per vedere cosa è necessario
- abbiamo già visto tutte le parti necessarie:
 - **handshake**: Alice, Bob usano i loro certificati + chiavi private per autenticarsi a vicenda e scambiarsi o creare "shared secrets" (chiave master che permette di generare chiavi simmetriche di sessione)
 - **key derivation**: Alice e Bob usano lo shared secret per derivare chiavi di sessione
 - **data transfer**: stream data transfer. I dati vengono visti come una serie di record e non più come un bytestream
 - necessario tenere traccia dello stato della connessione
 - **chiusura della connessione**: messaggi speciali per chiudere in modo sicuro la connessione

t-tls: initial handshake



fase di stretta di mano t-tls:

- Bob stabilisce una connessione TCP con Alice
- Bob verifica che Alice sia davvero Alice
- Bob invia ad Alice una chiave segreta principale MS (Master Secret), utilizzata per generare tutte le altre chiavi per la sessione TLS
- potenziali problemi:
 - 3 RTT prima che il client possa iniziare a ricevere i dati (includendo l'handshake TCP)

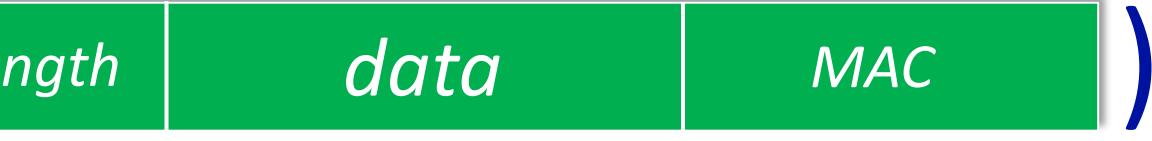


t-tls: chiavi crittografiche

- È considerato insicuro utilizzare la stessa chiave per più di una funzione crittografica
 - chiavi diverse per il codice di autenticazione del messaggio (MAC) e la crittografia del messaggio
- quattro chiavi simmetriche:
 - 🔑 K_c : chiave di cifratura per i dati inviati dal client al server
 - 🔑 M_c : chiave MAC per autenticare dati inviati dal client al server
 - 🔑 K_s : chiave di cifratura per i dati inviati dal server al client
 - 🔑 M_s : chiave MAC per autenticare i dati inviati dal server al client
- chiavi derivate dalla funzione di derivazione delle chiavi (KDF)
 - prende il segreto principale (shared secret scambiato con chiave pubblica) e alcuni dati casuali aggiuntivi per creare nuove chiavi

t-tls: cifratura dei dati

- richiamo: TCP fornisce *byte di dati astrazione del flusso*
- D: possiamo crittografare i dati man mano che arrivano al socket TCP?
 - R: dove andrebbe la "firma" MAC? Se alla fine, nessuna integrità del messaggio finché tutti i dati non sono stati ricevuti e la connessione chiusa!
 - soluzione: spezzettare il bytestream in una serie di "record"
 - ogni record client-server contiene un MAC, creato utilizzando M_c
 - il ricevitore può agire su ogni record man mano che arrivano
- record t-tls crittografato utilizzando la chiave simmetrica, K_c , passato a TCP:
(esempio semplificato usando un'unica chiave K_c)

$K_c($  $)$

header TCP non è cifrato (TLS è un servizio al di sopra di TCP)

t-tls: cifratura dei dati

- possibili attacchi al bytestream
 - *riordino*: man-in middle intercetta i segmenti TCP e riordina (manipolando i numeri di sequenza nell'intestazione TCP non crittografata)
 - *replay attack* (salvare il bytestream e riutilizzarlo dopo)
- soluzioni:
 - utilizzare numeri di sequenza TLS (TLS-seq-# incorporati in MAC)
 - usa il nonce (la chiave MAC cambia a ogni record) per rendere il replay attack inefficace



t-tls: chiusura connessione

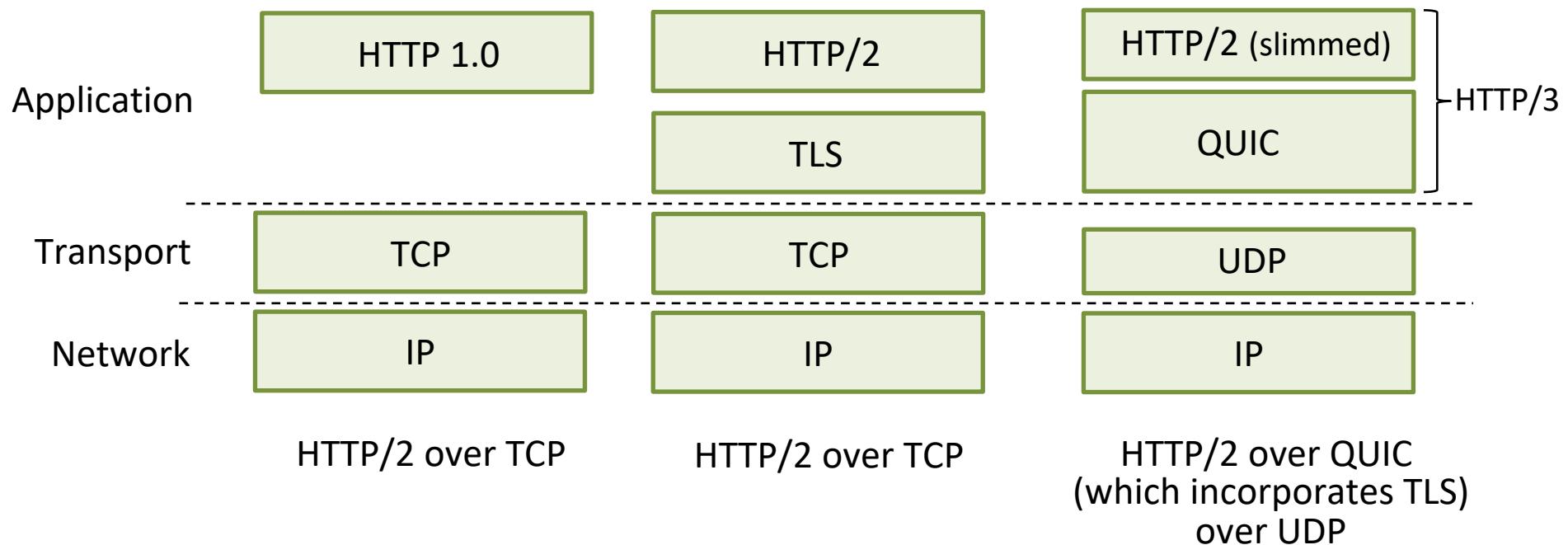
- truncation attack:
 - l'attaccante potrebbe falsificare il segmento di chiusura della connessione TCP (header in chiaro)
 - una o entrambe le parti pensano che ci siano meno dati di quanti ce ne siano in realtà
- soluzione: inserire un campo *type* nel record, con un tipo per la chiusura
 - type 0 for data; type 1 for close
- MAC calcolato utilizzando data, type, # sequenza

$K_C(\begin{array}{|c|c|c|} \hline length & type & data \\ \hline \end{array}) M_C(\begin{array}{|c|} \hline MAC \\ \hline \end{array})$



Sicurezza a livello di trasporto (TLS)

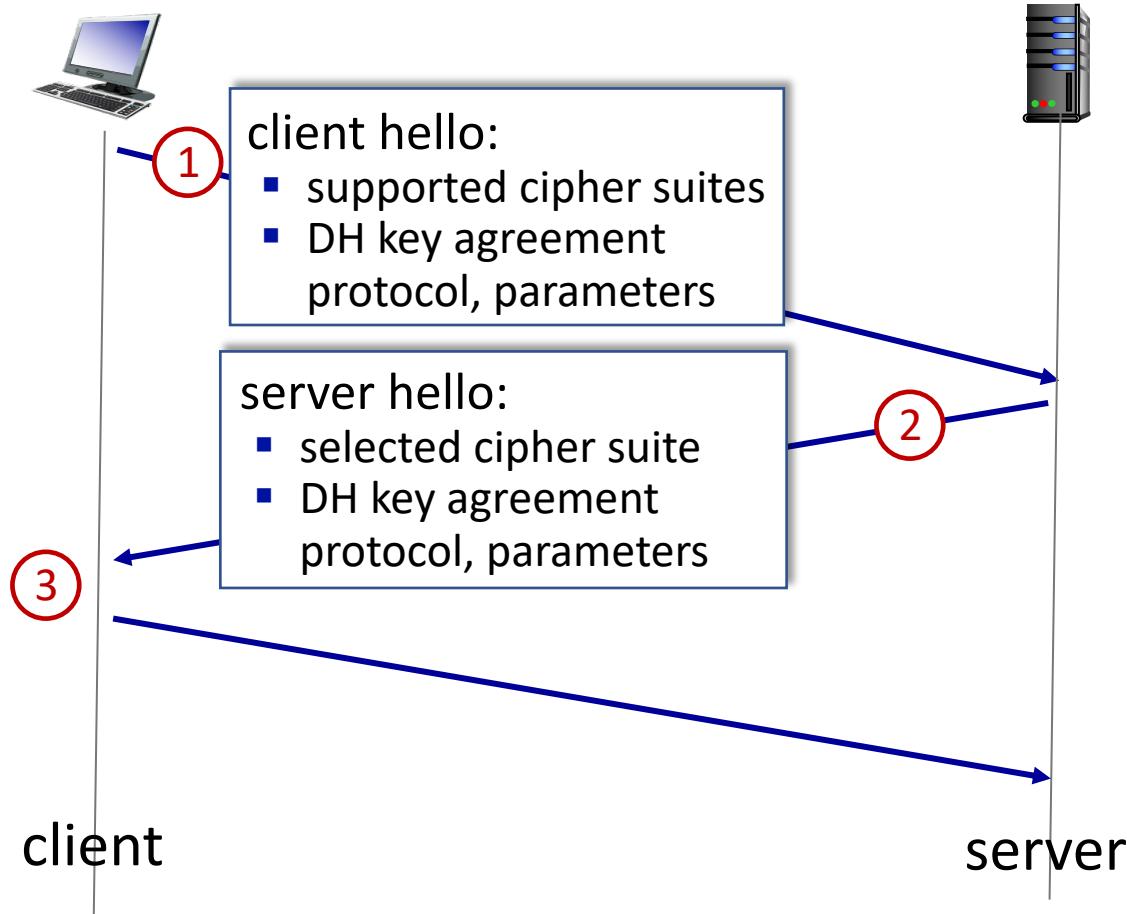
- TLS fornisce un'API che *qualsiasi* applicazione può utilizzare
- QUIC incorpora TLS per utilizzarlo sopra UDP



TLS: 1.3 cipher suite

- “cipher suite”: diversi algoritmi utilizzabili per la generazione di chiavi, cifratura, MAC, firma digitale
- TLS: 1.3 (2018) : scelte tra una selezione più limitata rispetto a TLS 1.2 (2008)
 - solo 5 scelte, anziché 37 scelte
 - *richiede* Diffie-Hellman (DH) per lo scambio di chiavi, piuttosto che DH o RSA
 - unisce crittografia e algoritmo di autenticazione ("crittografia autenticata") per i dati invece che cifratura seriale + autenticazione con MAC
 - basato su AES
 - HMAC utilizza la funzione hash crittografica SHA (256 o 284) per creare il digest del record

TLS 1.3 handshake: 1 RTT



① client TLS hello msg:

- propone protocollo di generazione chiavi e parametri
- indica le suite di cifratura supportate
- invia un "client random"

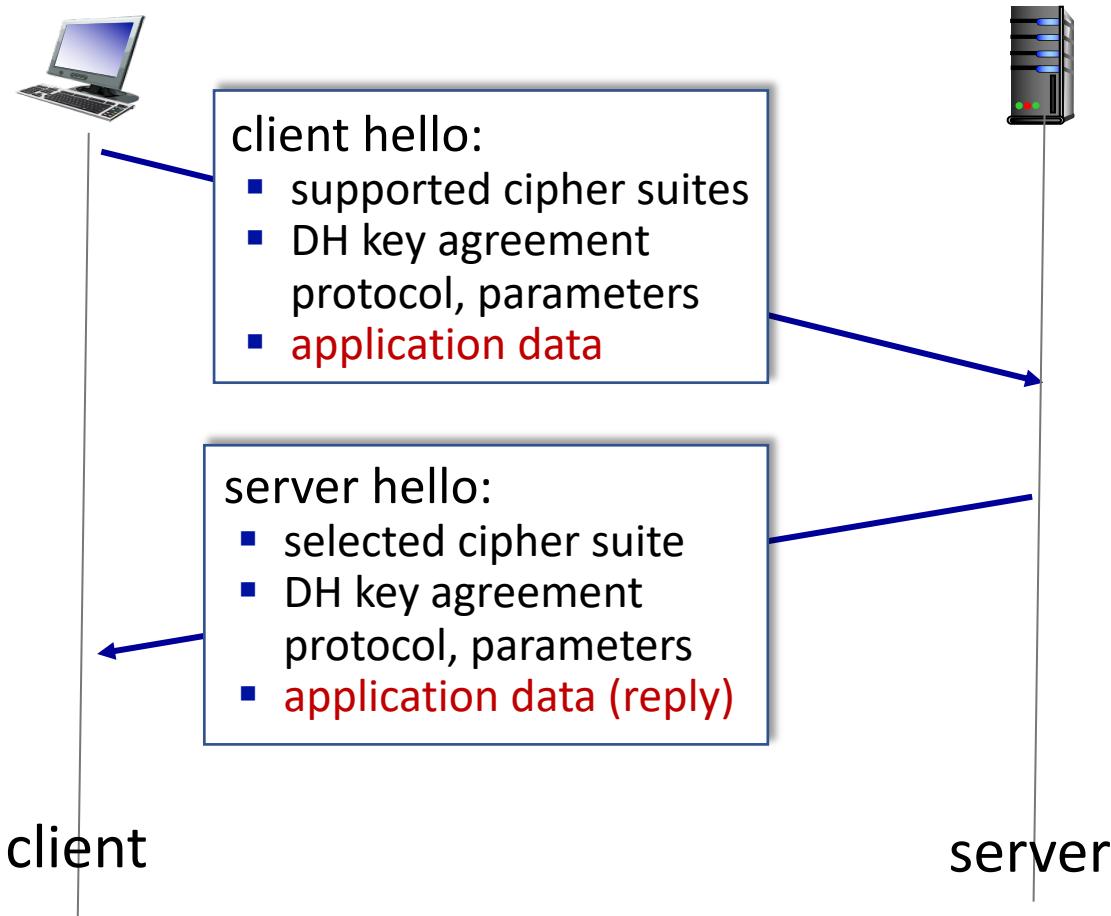
② server TLS hello msg

- sceglie protocollo di accordo chiave e parametri
- sceglie suite di cifratura
- invia certificato firmato
- invia "server random"

③ client:

- controlla il certificato del server usando la public key di CA
- genera Master Key (usando i due "random")
- ora può effettuare una richiesta di applicazione (ad es. HTTPS GET) che userà chiavi di sessione

TLS 1.3 handshake: 0 RTT (ripresa connessione)



- il messaggio hello iniziale già contiene dati cifrati!
 - "ripresa" della connessione precedente tra client e server
 - dati dell'applicazione crittografati utilizzando la Master Key della connessione precedente
- vulnerabile agli attacchi di replay!
 - si usa solo per richieste HTTP che non modificano lo stato del server

TLS e QUIC in Wireshark

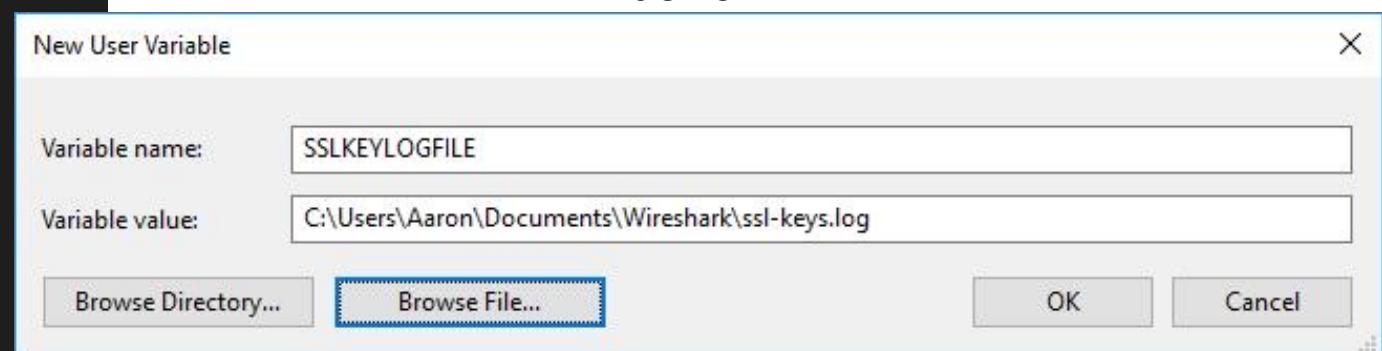


- lato server o client:
 - registrare le chiavi ssl di sistema e riavviare Chrome
- altrimenti: man-in-the-middle!

mac

```
● ● ● alessandro — alessandro@arm64-apple-darwin20 — ~ — zsh — 80x24
Last login: Thu May 16 15:07:01 on ttys001
(base) [~]$ export SSLKEYLOGFILE=~/ssl-key.log
(base) [~]$ open -n /Applications/Google\ Chrome.app
```

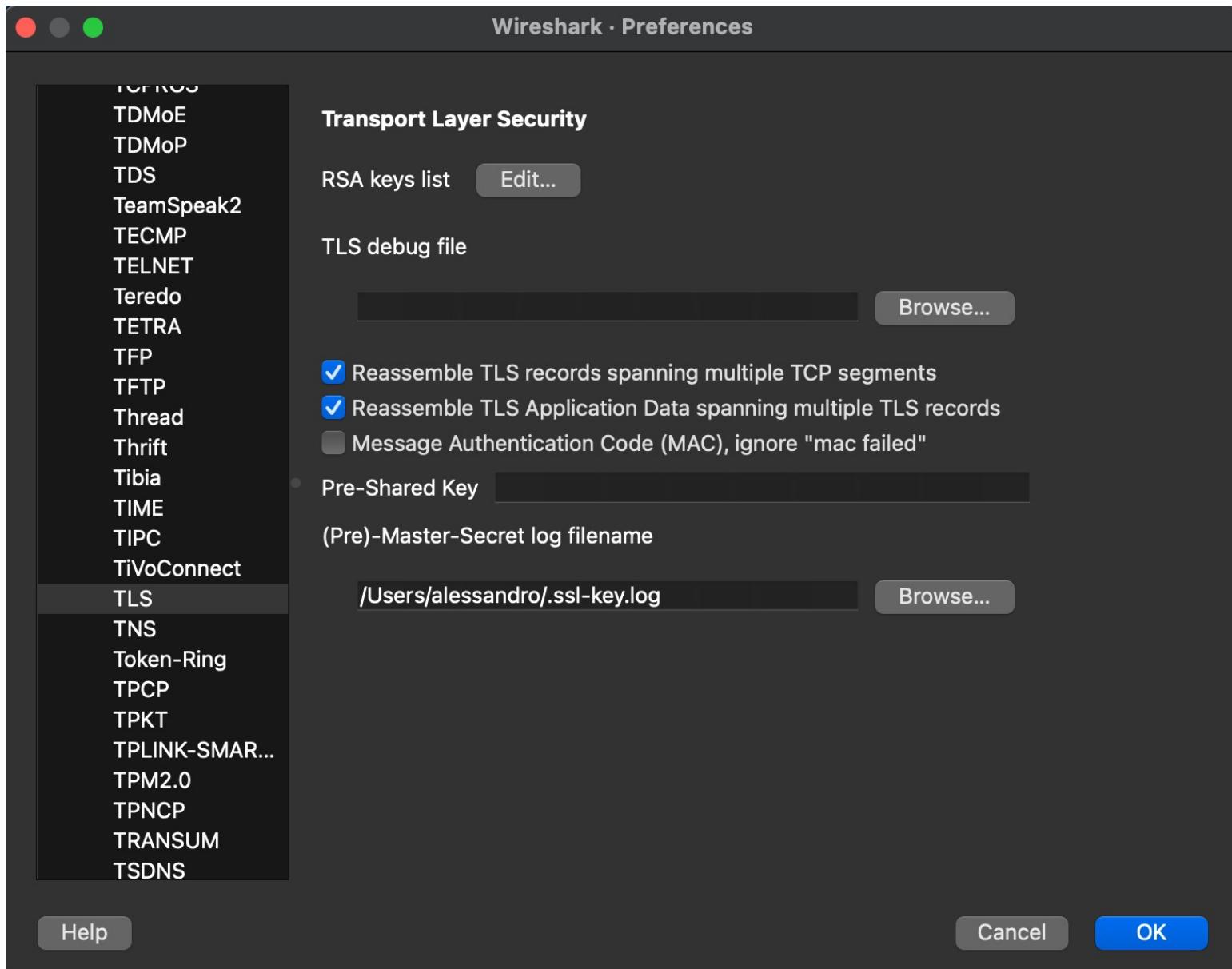
windows



Chiavi TLS

- ogni connessione TLS ha una nuova chiave

TLS e QUIC in Wireshark



TLS e QUIC in Wireshark

The screenshot displays a Wireshark capture of QUIC traffic. The packet list shows two QUIC Initial frames (763 and 764) exchanged between 10.90.108.5 and 142.251.209.4. The details pane provides a detailed breakdown of the QUIC frame structure, including fields like Destination Connection ID, Source Connection ID, and Token Length. The bytes pane shows the raw hex and ASCII data. A yellow callout points to the TLSv1.3 Record Layer: Handshake Protocol: Client Hello section in the details pane.

Packet 763 (Initial Frame):
Source: 10.90.108.5
Destination: 142.251.209.4
Protocol: QUIC
Length: 1292
Info: Initial, DCID=a89bbbcaf6890a13, PKN: 1, CRYPTO

Packet 764 (Initial Frame):
Source: 10.90.108.5
Destination: 142.251.209.4
Protocol: QUIC
Length: 1292
Info: Initial, DCID=a89bbbcaf6890a13, PKN: 2, PING, PADDING, CRYPTO, PADDING, PING, PADDING, PING, PADDING

Details pane (TLSv1.3 Record Layer: Handshake Protocol: Client Hello):
Handshake Protocol: Client Hello (last fragment)
[2 Reassembled Handshake Fragments (2257 bytes): #763(1211), #764(1046)]
[Frame: 763, payload: 0-1210 (1211 bytes)]
[Frame: 764, payload: 1211-2256 (1046 bytes)]
[Handshake Fragment count: 2]
> Handshake Protocol: Client Hello

Packets: 9092 · Displayed: 6415 (70.6%)

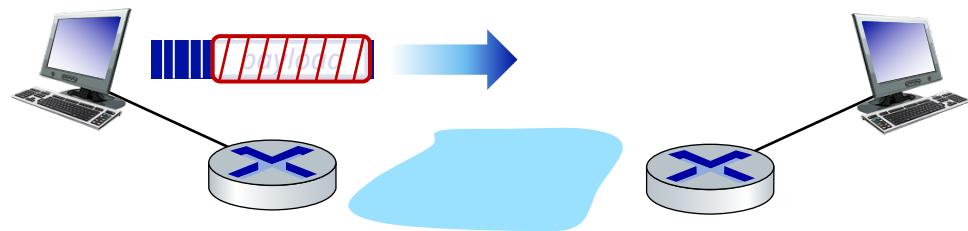
Profile: Default

Sicurezza - sommario

- Che cos'è la sicurezza della rete?
- Principi di crittografia
- Integrità del messaggio, autenticazione
- Protezione della posta elettronica
- Protezione delle connessioni TCP: TLS
- **Sicurezza a livello di rete: IPsec**
- Sicurezza nelle reti wireless e mobili
- Sicurezza operativa: firewall e IDS

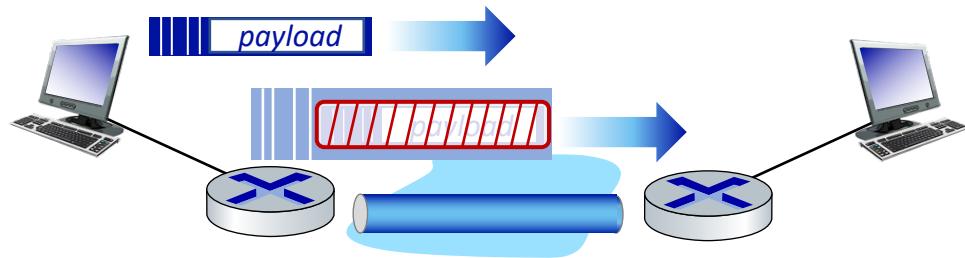
IP Sec

- fornisce crittografia, autenticazione e integrità a livello di datagramma
 - sia per il traffico utente che per il traffico di controllo (ad es. BGP, messaggi DNS)
- due “modalità”:



modalità di trasporto:

- *soltanto il payload* del datagramma è cifrato e autenticato



modalità tunnel (VPN):

- l'intero datagramma è cifrato e autenticato
- datagramma cifrato incapsulato in un nuovo datagramma con una nuova intestazione IP, incanalato verso la destinazione
- chi osserva il traffico vede traffico cifrato tra client e server, senza poter sapere la sorgente iniziale e la destinazione finale dei pacchetti incapsulato



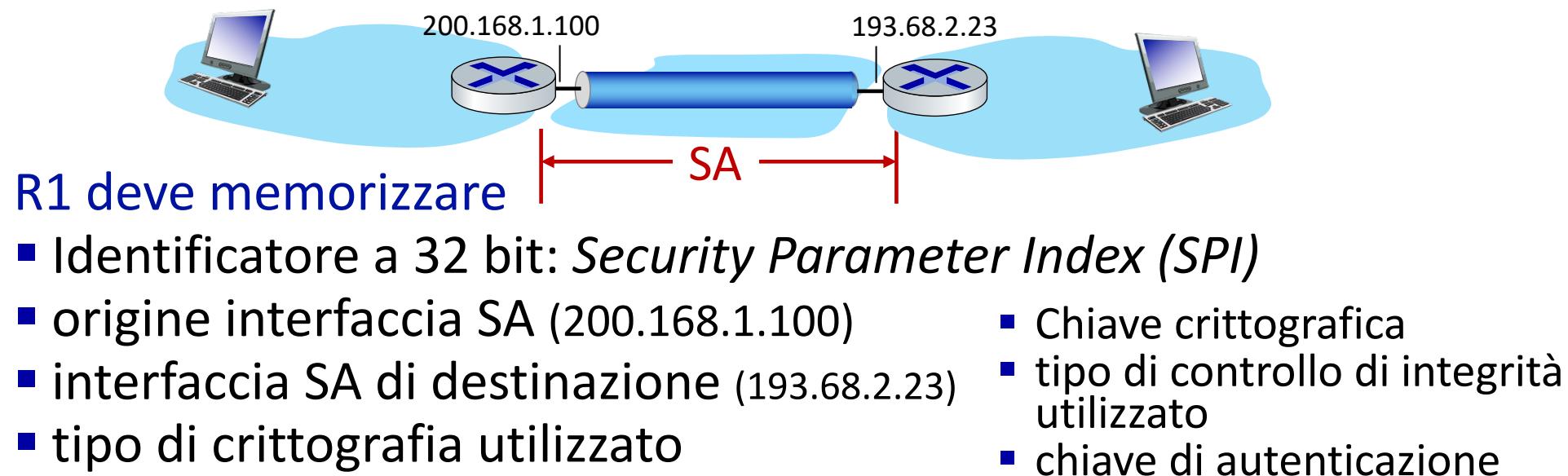
Due protocoli IPsec

- Authentication Header (AH) protocol [RFC 4302]
 - provides source authentication & data integrity but *not* confidentiality
- Encapsulation Security Protocol (ESP) [RFC 4303]
 - provides source authentication, data integrity, *and* *confidentiality*
 - **more widely used than AH**

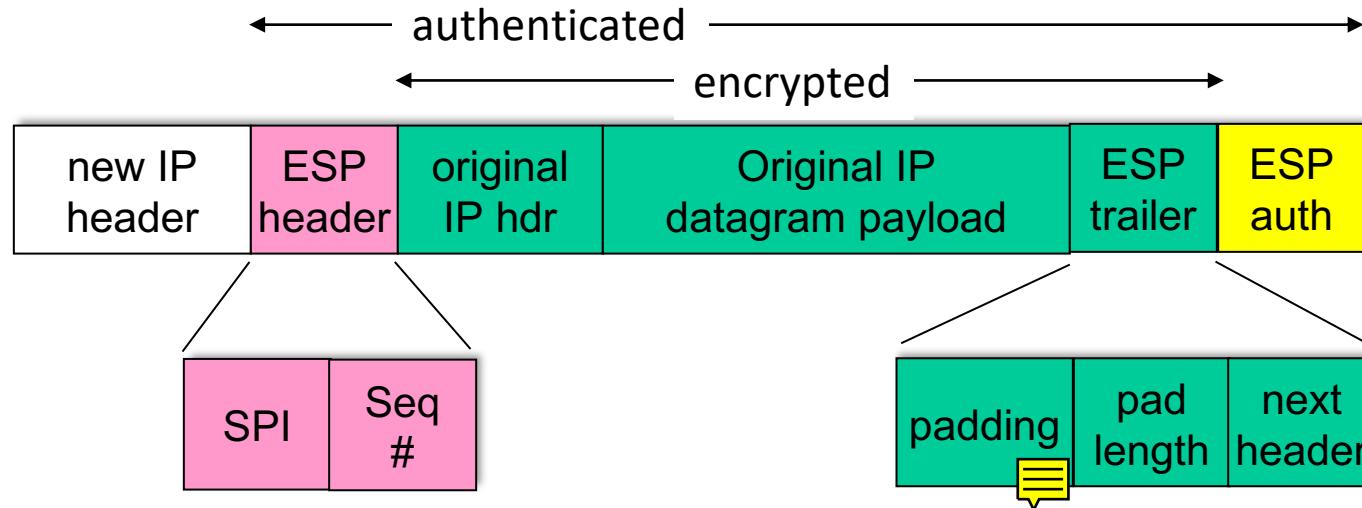


Security associations (SAs)

- prima dell'invio dei dati, **associazione di sicurezza (SA)** stabilita dall'entità di invio all'entità ricevente (direzionale)
- i due router devono memorizzare *le informazioni sullo stato* su SA
 - richiamo: anche gli endpoint TLS memorizzano informazioni sullo stato ma
 - **IP è senza connessione; IPsec è orientato alla connessione!**



IPsec datagram



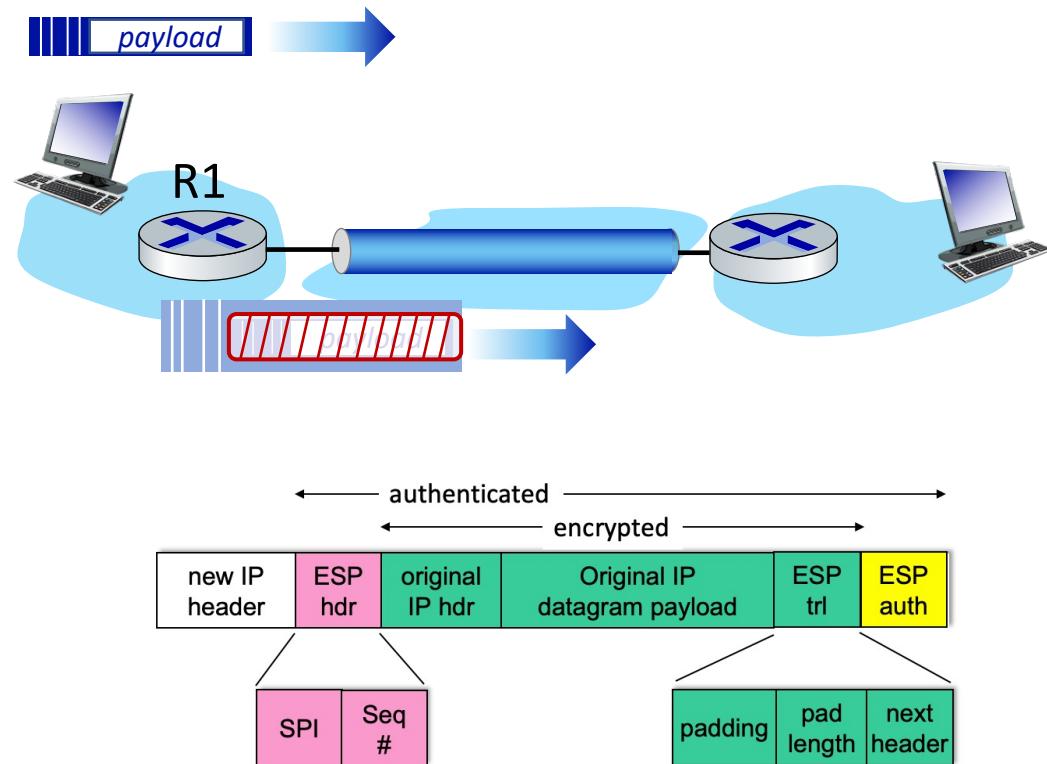
*tunnel mode
ESP*

- ESP trailer: padding per avere blocchi di lunghezza fissa
- ESP header:
 - SPI, per riconoscere a quale tunnel ci riferiamo
 - numero di sequenza per bloccare attacchi replay
- MAC nel campo auth di ESP creato con la chiave crittografica condivisa tra R1 e R2

ESP tunnel mode: actions

in R1:

- aggiunge il trailer ESP al datagramma originale (che include i campi di intestazione originali!)
- cifra il **risultato** utilizzando l'algoritmo e la chiave specificati da SA
- aggiunge l' **intestazione ESP** davanti a questa parte cifrata
- crea **'l'autenticazione MAC'** utilizzando l'algoritmo e la chiave specificati in SA 
- *inserisce MAC alla fine creando il payload IP completo*
- crea una nuova intestazione IP, nuovi campi di intestazione IP, indirizzi all'endpoint del tunnel



Numeri di sequenza IPsec

- per la nuova Security Association, il mittente inizializza seq. # a 0
- ogni volta che il datagramma viene inviato su SA:
 - mittente incrementa seq # counter
 - inserisce il valore nel campo seq # (che è autenticato, cioè firmato)
- obiettivo:
 - impedire all'aggressore di sniffare e riprodurre un pacchetto
 - la ricezione di pacchetti IP duplicati autenticati può interrompere il servizio
- metodo:
 - la destinazione verifica la presenza di duplicati
 - non tiene traccia di *tutti i* pacchetti ricevuti; utilizza invece una finestra

IPsec security databases

Security Policy Database (SPD)

- policy: per ogni datagramma, il mittente deve sapere se usare IP sec
- policy immagazzinata nel **security policy database (SPD)**
- necessario sapere quale SA usare
 - può usare: IP e numero di protocollo di sorgente e destinazione

SPD: “cosa” fare

Security Assoc. Database (SAD)

- l'endpoint mantiene lo stato di una SA nel **security association database (SAD)**
- quando viene inoltrato un datagramma IPsec, R1 accede a SAD per sapere come processare il datagramma
- quando il datagramma IPsec raggiunge R2, R2 esamina nell'header IPsec lo Security Parameter Index (SPI) e cerca in SAD come processare il datagramma correttamente

SAD: “come” comunicare

Sommario: servizi IPsec



Trudy si trova da qualche parte tra R1, R2. E non conosce le chiavi

- Trudy sarà in grado di vedere i contenuti originali del datagramma?
- Trudy sarà in grado di vedere origine, indirizzo IP di destinazione, protocollo di trasporto, porta dell'applicazione?
- flip bit senza essere scoperta?
- impersonare R1 usando l'indirizzo IP di R1?
- riprodurre un datagramma?



NO!