NETWORK SECURITY

Nell'internet odierno, si fa sicurezza praticamente a tutti i livelli, specialmente a livello 4 e 5. Tuttavia, si fa anche sicurezza a livello 3 (IPSec) e, in particolare per reti wireless, anche a livello 2.

• Cos'è la Sicurezza?

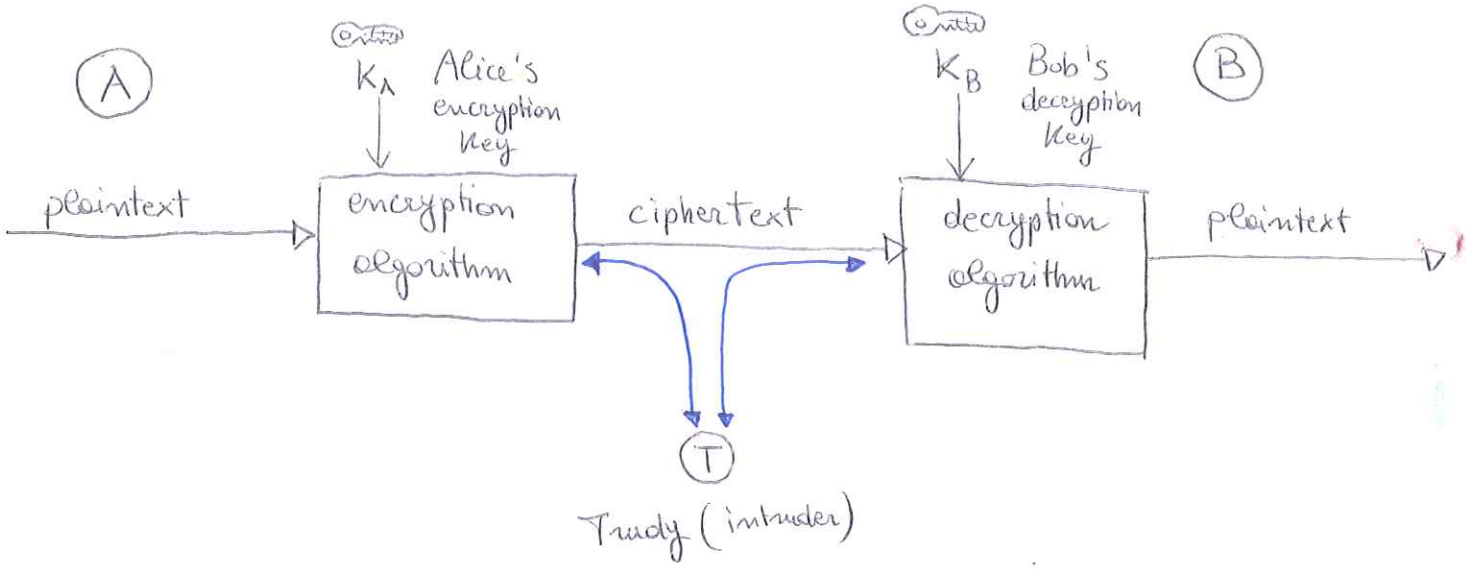
Ci sono diversi aspetti che riguardano la sicurezza informatica:

- CONFIDENZIALITÀ: solo il sender e il receiver dovrebbero "capire" i contenuti dei messaggi:
  - il sender fa CIFRATURA (encryption)
  - il receiver DECIFRA (decryption)
- AUTENTICAZIONE: Sender e receiver vogliono confermarsi reciprocamente la propria identità (e.g. firma digitale)
- INTEGRITÀ DEI MESSAGGI: Sender e receiver vogliono assicurarsi che i messaggi non vengano alterati durante il transito
- ACCESSO E DISPONIBILITÀ: i servizi devono essere accessibili agli utenti e disponibili; è ciò che cercano di negare gli attacchi DoS (Denial of Service)

• Cosa può fare un Intruder?

1) Eavesdrop: intercettare i messaggi
2) Aggiungere messaggi attivamente nella connessione
3) Impersonation: può falsificare i campi degli header dei pacchetti, specialmente il source address → SPOOFING (falsificazione)
4) Hijacking: (dirottamento) "impossessarsi" della connessione, rimuovendo il sender o il receiver e sostituendosi ad esso
5) Denial of Service: prevenire che un servizio sia usato dagli altri utenti, ad esempio sovraccaricando i server di richieste e occupando risorse.

- IL LINGUAGGIO DELLA CRITTOGRAFIA

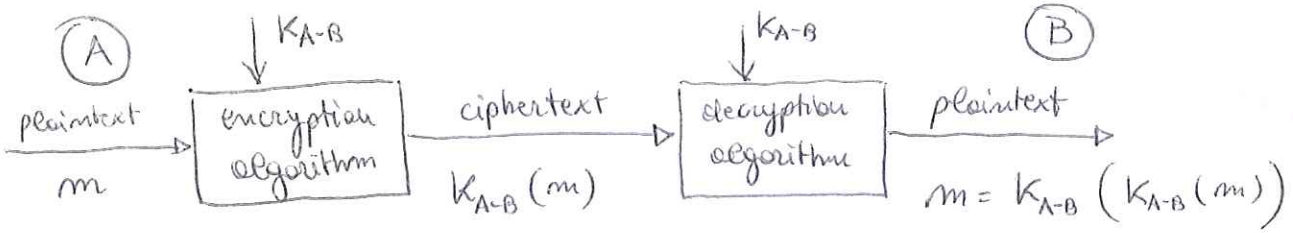


- CRITTOGRAFIA A CHIAVE SIMMETRICA : Sender e receiver usano la stessa chiave di cifratura → $K_A \equiv K_B$

- CRITTOGRAFIA A CHIAVE PUBBLICA (o ASIMMETRICA) ; Ogni utente ha 2 chiavi { pubblica, privata

La chiave pubblica è nota a tutti, ed è usata per decifrare ; quella privata per cifrare.

CRITTOGRAFIA A CHIAVE SIMMETRICA



* A e B condividono la STESSA CHIAVE di cifratura K_{A-B}, che viene usata sia per cifrare che per decifrare.
Ovviamente K_{A-B} dipende dal meccanismo di cifratura.

- Ma come fanno A e B a scambiarsi la chiave K_{A-B} in sicurezza?
  ↳ Lo vedremo in seguito, ma usano l'RSA, un meccanismo a chiave pubblica!

Esempi :

1) Cifratura di Cesare : traslare l'alfabeto e fare il mapping → 26 possibili chiavi
   ⇒ Brute Force : 26 tentativi massimo

2) Substitution Cipher : permutare a caso l'alfabeto e fare il mapping → 26! possibili chiavi
   ⇒ Brute Force : 26! tentativi massimo

   ↯ Analisi statistica : conoscendo i simboli e le parole più probabili, è molto facilmente ricostruibile e sfruttabile!

3) Substitution Cipher with cycling pattern:

- n cifrari diversi: $M_1, M_2, \dots, M_n$
- pattern ciclico di utilizzo dei cifrari per ogni nuovo simbolo in plaintext; ad esempio, per $n=4$, si può avere: $M_1, M_3, M_4, M_3, M_2, M_4, \dots$

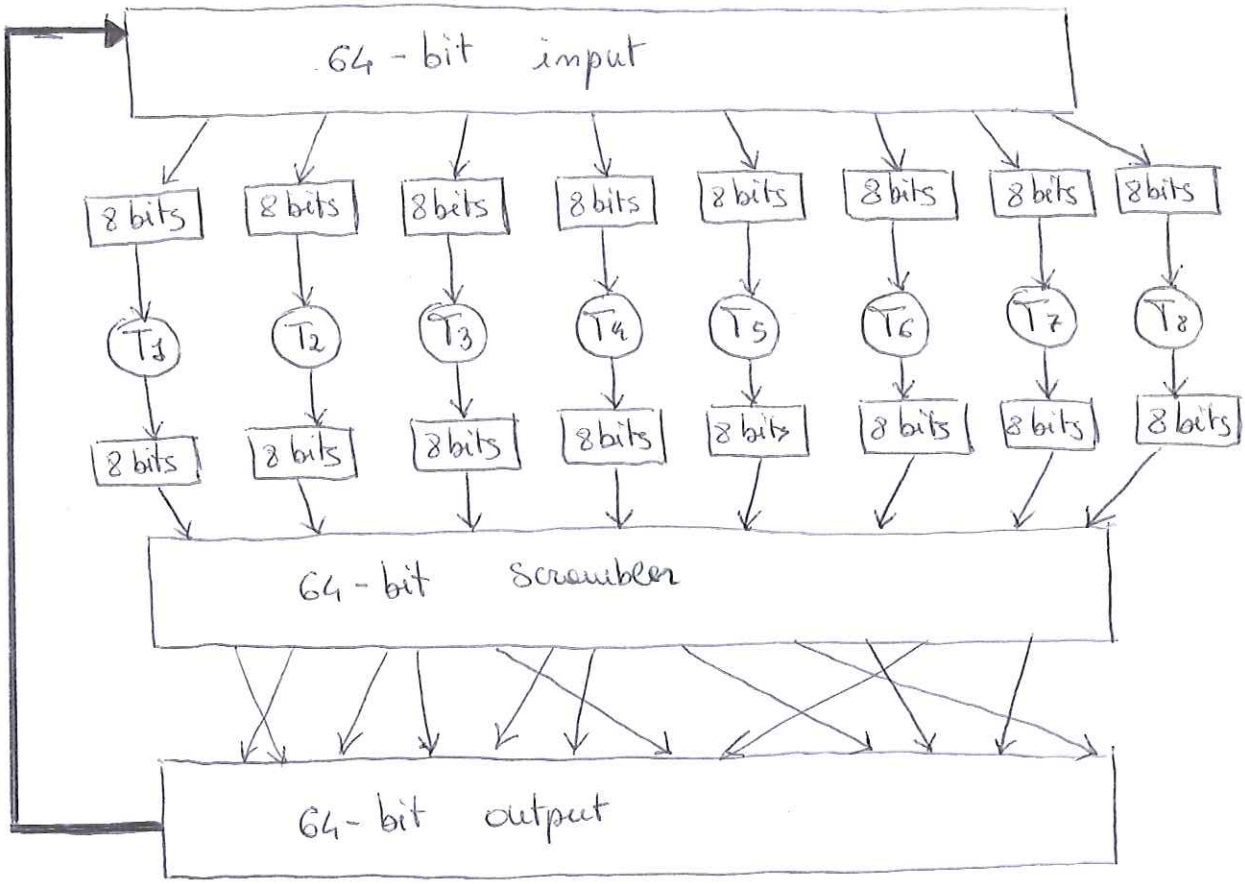
* Simboli uguali vengono cifrati secondo cifrari diversi

ENCRYPTION KEY: gli n cifrari + il pattern ciclico (di solito più lungo di n)

↳ La chiave non è un pattern fisso di n-bit, ma è più grande e difficile da gestire!

- **BLOCK CIPHER:**

Cifratura A BLOCCHI → Si divide l'input a blocchi e si cifrano singolarmente, trasformandoli tramite funzioni diverse.
Degli output si fa SCRAMBLING secondo uno schema fisso! Eventualmente, tutto il processo viene ripetuto più volte. (Lo scrambling si fa bit a bit, non a blocchi).

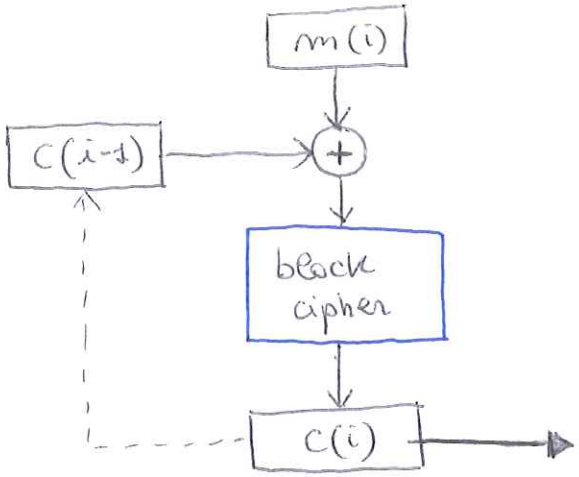


- Block ciphers: DES, 3DES, AES

PROBLEMI: Se in input vengono dati blocchi uguali, in uscita ci saranno gli STESSI OUTPUT! → L'uscita cifrata che si ripete è un'informazione per gli attaccanti!!!

- **CIPHER BLOCK CHAINING:**

Si fa la concatenazione (chaining) tra il blocco da cifrare $m(i)$ e quello cifrato precedentemente $C(i-1)$.
Di solito, si fa lo XOR tra $m(i)$ e $C(i-1)$.
Per il 1° blocco, si utilizza un vettore $C(0)$ generato randomicamente, detto INITIALIZATION VECTOR, che viene inviato in chiaro al receiver!



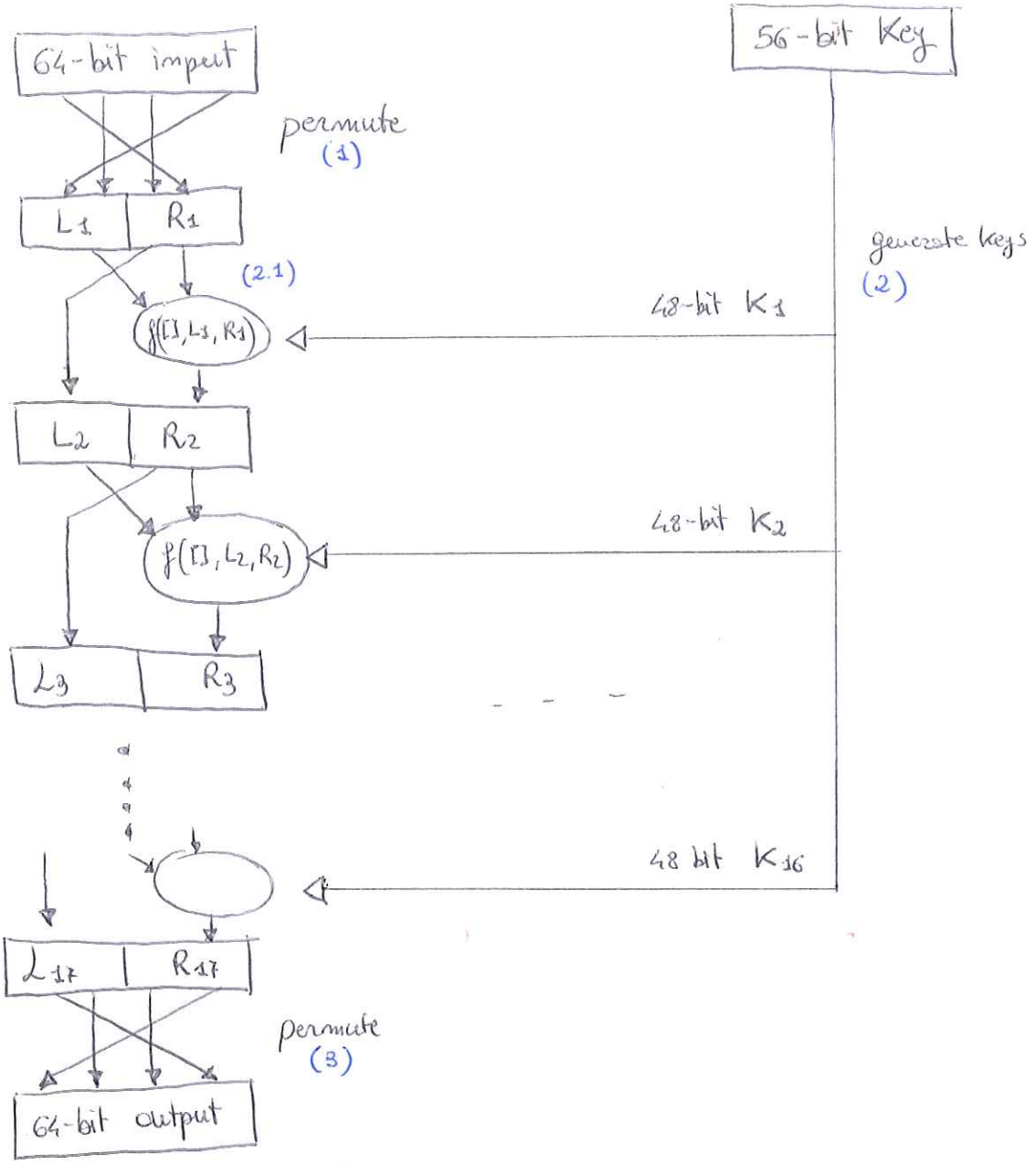
DES

- DES = Data Encryption System
- Use crittografia a CHIAVE SIMMETRICA, con una chiave a 56-bit e input in plaintext a 64-bit
- Use cipher-block chaining
- Per renderlo più sicuro → 3DES: use 3 chiavi sequenzialmente su ogni dato

Funzionamento:

1) Permutazione iniziale dei 64 bit
2) Si svolge quanto segue per 16 ROUND, in ognuno dei quali si use una chiave di 48-bit diverse, generate dalla chiave a 56-bit;
   2.1) Si dividono i 64-bit a metà; la parte destra viene messa a sinistra e, per generare la nuova parte destra, si use una funzione che prende in input la chiave, la parte SX e la parte DX.
3) Permutazione finale dei 64 bit

- Vediamolo anche con uno schema:



- **AES: ADVANCED ENCRYPTION STANDARD**
  - Processe i dati in blocchi da 128 bit ; il doppio di DES e 3DES.
  - Può usare chiavi a 128, 192 o 256 bit
  - Decifrature Brute Force : se ci volesse 1 secondo per decriptare DES brute force, per AES ci vorrebbero 149 TRILIONI di ANNI !!!

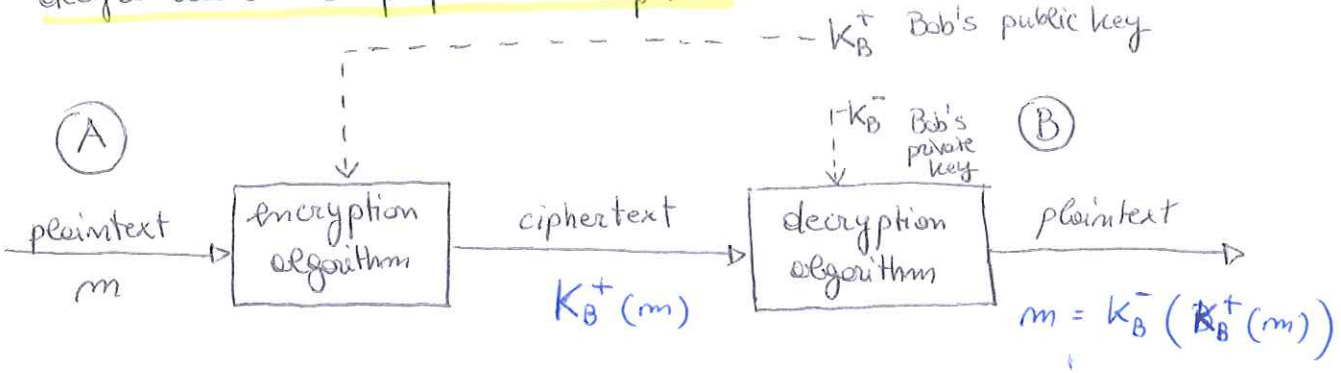
* Rimane sempre il solito problema :

Essendo la chiave simmetrica, bisogna scambierla, e accordarsi sui protocolli di sicurezza da usare.

Come scambiare queste informazioni in sicurezza ?

CRITTOGRAFIA A CHIAVE PUBBLICA

- Sender e receiver NON hanno bisogno di scambiarsi chiavi segrete!
- Ogni utente ha: { 1 chiave pubblica K^+ → NOTA A TUTTI ; 1 chiave privata K^- → NOTA SOLO A SE STESSO }
- Si cifra usando la chiave pubblica del destinatario (!) e il destinatario decifra usando la propria chiave privata



- REQUISITI:

1. Avere chiavi $K_B^+(\cdot)$ e $K_B^-(\cdot)$ tali che:

$$K_B^-(K_B^+(m)) = m$$

2. Data la chiave pubblica K_B^+, dovrebbe essere impossibile calcolare la chiave privata K_B^-.

* RSA = Rivest, Shamir, Adleman algorithm

- PREREQUISITI: ARITMETICA MODULARE:
  - $[(a \bmod n) + (b \bmod n)] \bmod n = (a+b) \bmod n$
  - $[(a \bmod n) - (b \bmod n)] \bmod n = (a-b) \bmod n$
  - $[(a \bmod n) \cdot (b \bmod n)] \bmod n = (a \cdot b) \bmod n$

Così, vale anche la seguente:

$$(a \bmod n)^d \bmod n = a^d \bmod n$$

- Dato che un messaggio è una sequenza di bit, possiamo interpretarlo come un numero e cifrarlo tramite l'aritmetica modulare, ottenendo in output un altro numero, che è il messaggio cifrato.

• RSA: SCEGLIERE LE CHIAVI:

1) Scegliere 2 numeri primi p e q molto grandi (almeno 1024 bit)

2) Calcolare $n = p \cdot q$ e $z = (p-1) \cdot (q-1)$

3) Scegliere e, con $e < n$, tale che non abbia fattori comuni con z; e e z sono primi tra loro!

4) Scegliere d, tale che $(e \cdot d - 1)$ sia esattamente divisibile per z:

$$e \cdot d \bmod z = 1 \Rightarrow (e \cdot d - 1) \bmod z = 0$$

5) Dunque, la CHIAVE PUBBLICA $K^+ = (n, e)$;
la CHIAVE PRIVATA è $K^- = (n, d)$.

• RSA: CIFRATURA/DECIFRATURA:

Data $K^+ = (n, e)$ e $K^- = (n, d)$:

- CIFRARE: Per cifrare m, calcolare → $c = m^e \bmod n$ usando K^+
- DECIFRARE: Per decifrare c, calcolare → $m = c^d \bmod n$ usando K^-

Esempio:

$p = 5$, $q = 7 \Rightarrow n = p \cdot q = 35$; $z = (p-1)(q-1) = 24$

Scegliamo $e = 5$, così e e z sono primi tra loro

$e \cdot d \bmod z = 1 \Rightarrow 5d \bmod 24 = 1$ → ad esempio, scegliamo $d = 29$

encrypt → Se $m = 12 \Rightarrow m^e = 1.248.832 \Rightarrow c = m^e \bmod n = 17$

decrypt → $c = 17 \Rightarrow c^d =$ numero enorme $\Rightarrow m = c^d \bmod n = 12$ ✓

• RISULTATO TEORIA DEI NUMERI:

Se p e q sono primi e $n = p \cdot q$, allora: $x^y \bmod n = x^{y \bmod (p-1)(q-1)} \bmod n$

Quindi: $(m^e \bmod n)^d \bmod n = m^{ed} \bmod n = m^{ed \bmod (p-1)(q-1)} \bmod n =$

$= m^1 \bmod n = m$

$ed \bmod z = 1$

per n molto grande

* Occhio che vale anche la seguente proprietà, sfruttata nelle firme digitali:

$$K_B^-(K_B^+(m)) = m = K_B^+(K_B^-(m))$$

↳ Si può usare la Chiave Privata per cifrare e quella pubblica per decifrare!

- Questa proprietà segue direttamente dall'algebra modulare:

$$(m^e \bmod n)^d \bmod n = m^{ed} \bmod n = (m^d \bmod n)^e \bmod n$$

- **PERCHÉ RSA È SICURO?**

Supponiamo di conoscere la chiave pubblica di Bob $K_B^+ = (n, e)$. Quanto è difficile calcolare d?

↳ Bisogna trovare i fattori primi di n.

* Se trovare i fattori primi p e q di un numero molto grande è computazionalmente "quasi impossibile" ⟶ È su questo che si base la sicurezza di questo sistema!

- **SESSION KEY:**

PROBLEMA: Calcolare gli esponenziali con numeri così grandi è molto costoso e dispendioso!
DES è almeno 2 ordini di grandezza più veloce.

↳ SESSION KEY, K_S

- Si usa RSA per comunicare la chiave simmetrica K_S, detta "di sessione"
- Con K_S usano un meccanismo di CRITTOGRAFIA A CHIAVE SIMMETRICA!

MESSAGE INTEGRITY

Un receiver vuole assicurarsi 2 cose:

- Che il messaggio ricevuto provenga davvero dal sender
- Che il messaggio inviato dal sender non sia stato modificato

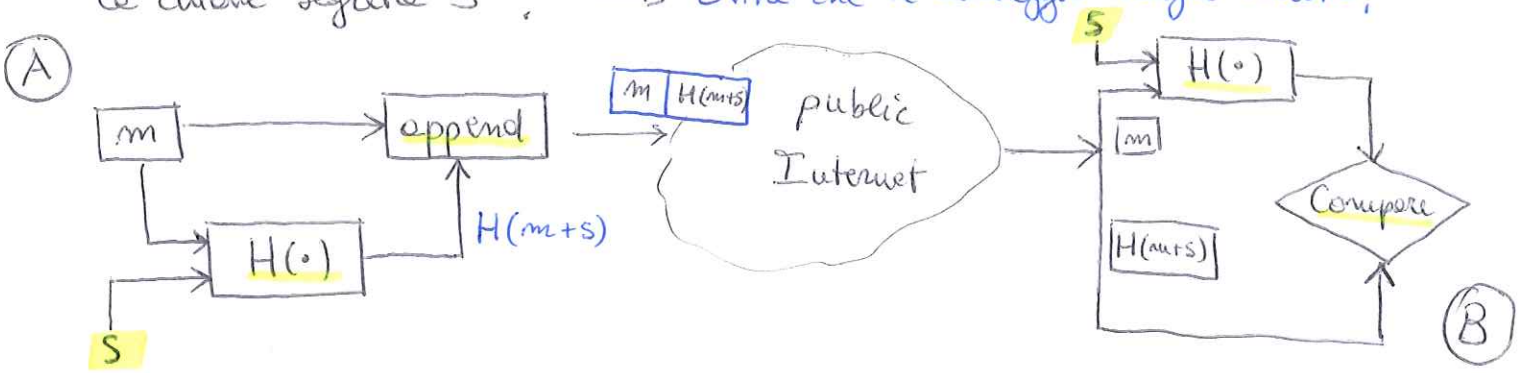
- **HASH CRITTOGRAFICO:**

- Dato un input m, una funzione di Hash $H(\cdot)$ produce $H(m)$, un DIGEST di dimensione fissa! ⟶ Il checksum è un esempio
- È computazionalmente improbabile trovare 2 messaggi x e y diversi, tali che $H(x) = H(y)$.

- Il checksum di Internet ha alcune proprietà di una funzione hash:
  1) Produce un digest di dimensione fisse (16 bit)
  2) È many-to-one

* Tuttavia, dato un messaggio con il suo hash value, è FACILE trovare un altro messaggio con stesso hash value!

- **MESSAGE AUTHENTICATION CODE:**
  - Si applica l'hash al messaggio + una CHIAVE SEGRETA condivise S (scambiate come? → RSA).
  - È difficile modificare m tale che $H'(m+S)$ sia corretta, senza conoscere la chiave segreta S! → Evita che il messaggio venga corrotto!



- Esempi di MACs:
  1) MD5: produce un digest di 128 bit, recentemente attaccato (2005)
  2) SHA-1: standard negli US; produce un digest di 160 bit

- **FIRMA DIGITALE:**

Tecnica crittografica analoga alla firma a mano.

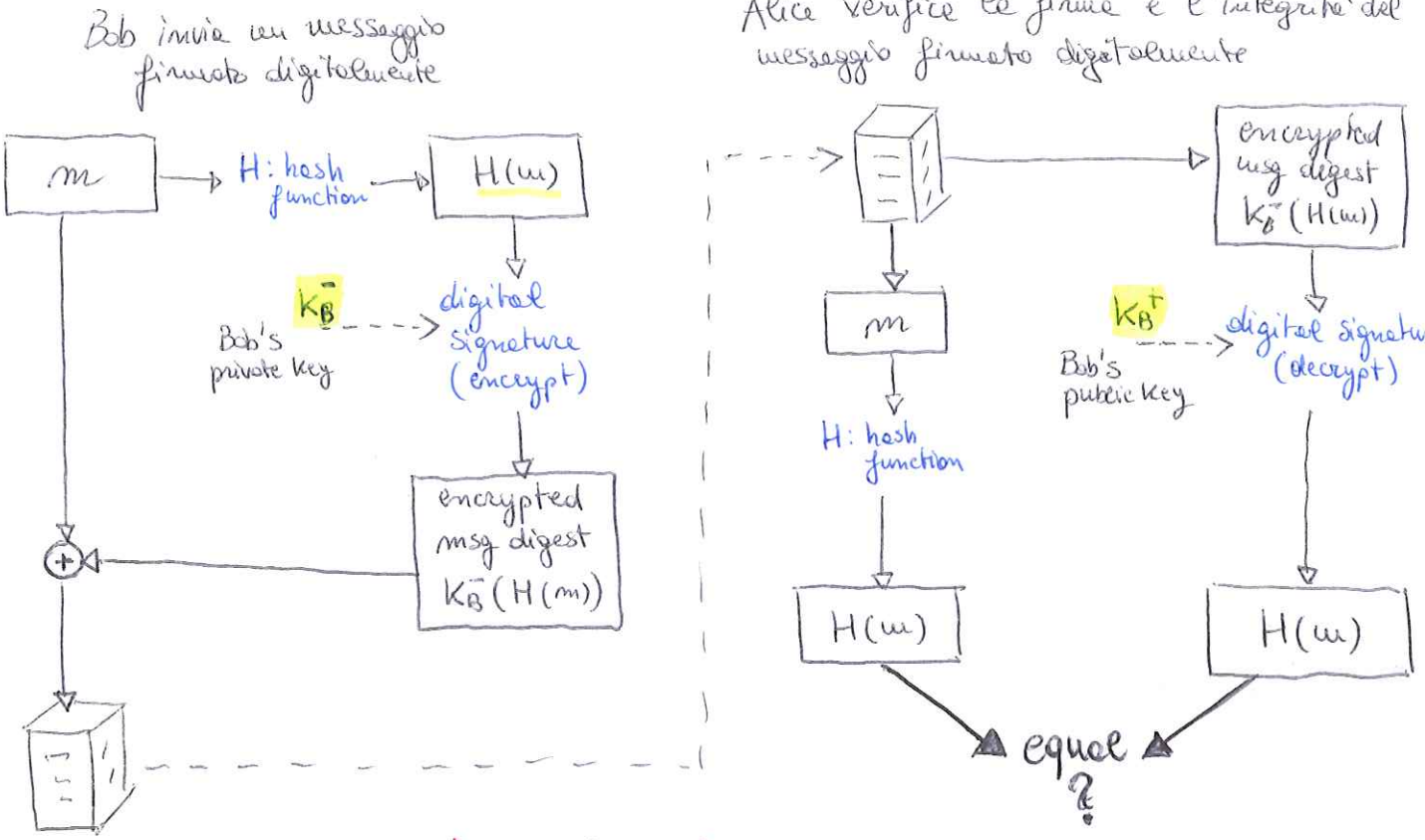
> Si firma un messaggio m, criptandolo usando la propria CHIAVE PRIVATA; messaggio firmato = $K_A^-(m)$.

- Il receiver può verificare l'autenticità, cifrando $K_A^-(m)$ con la chiave pubblica del mittente: vedere che $K_A^+(K_A^-(m)) = m$!
- Il receiver così verifica che:
  1. Il sender ha firmato m
  2. Nessun altro ha firmato m
  3. Il sender ha firmato m e non m'
- Non-repudiation: Si può facilmente verificare che il sender ha firmato m, non può ripudiarlo/negarlo!

PROBLEMA: Cifrare tutto il messaggio m con la Private key e poi decifrarlo è costosissimo!

↳ FINGERPRINT:

- Si usa una funzione Hash $H(\cdot)$ sul messaggio m, che produce un digest di dimensione fisse e molto minore di m (di solito)
- Si firma solo il digest, producendo una FINGERPRINT = $K_A^-(H(m))$
- Si invia m + il fingerprint
- Il receiver si calcola $H(m)$ e decripta il fingerprint, con la chiave pubblica del sender e ottiene $H(m) = K_A^+(K_A^-(H(m)))$.
  Se le 2 $H(m)$ coincidono, ha provato l'autenticità!



AUTENTICAZIONE

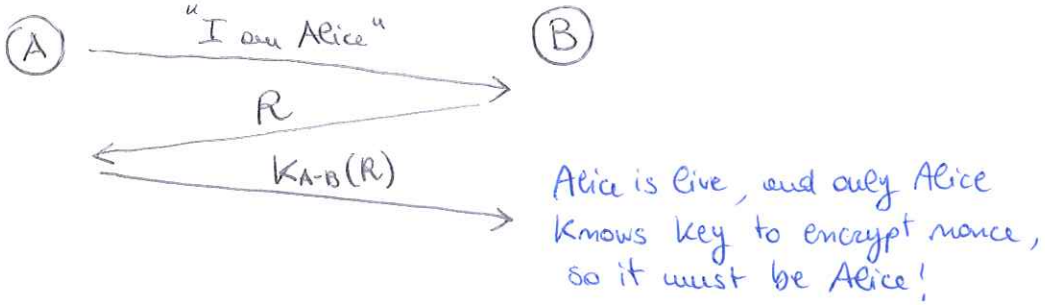
OBIETTIVO: Provare la propria identità. E.g. Bob vuole che Alice gli provi la sua identità.

ATTACCHI:

1) SPOOFING: falsificazione di identità
2) PLAYBACK ATTACK: l'intruder REGISTRA i pacchetti del Sender (anche criptati) e poi li reinvia al receiver, ricevendo risposte!

SOLUZIONE: Per evitare playback attacks, si usano le NONCE

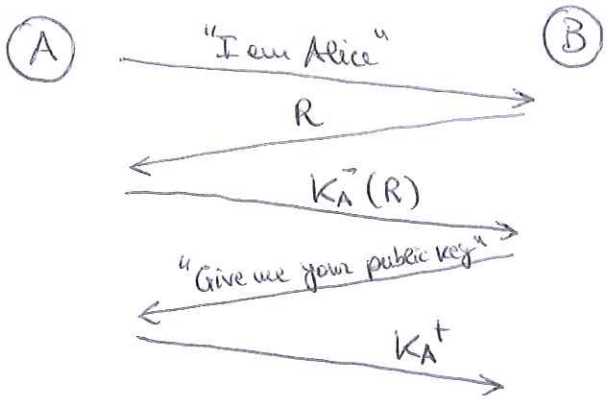
- Una nonce è un numero R generato pseudo-random e usato solo "once-in-a-lifetime"



* Non si può fare playback perché:
  1. Se si cattura R, non si è in grado di calcolare $K_{A-B}(R)$
  2. Se si cattura $K_{A-B}(R)$, non si può riutilizzare, perché R cambia ogni volta!

Svantaggio → Richiede la chiave simmetrica K_{A-B}!

↳ Usiamo la chiave pubblica!



Bob calcola $K_A^+(K_A^-(R)) = R$ e sa che solo Alice ha la chiave privata con cui è stato criptato R tale che $K_A^+(K_A^-(R)) = R$!

* PROBLEMA:

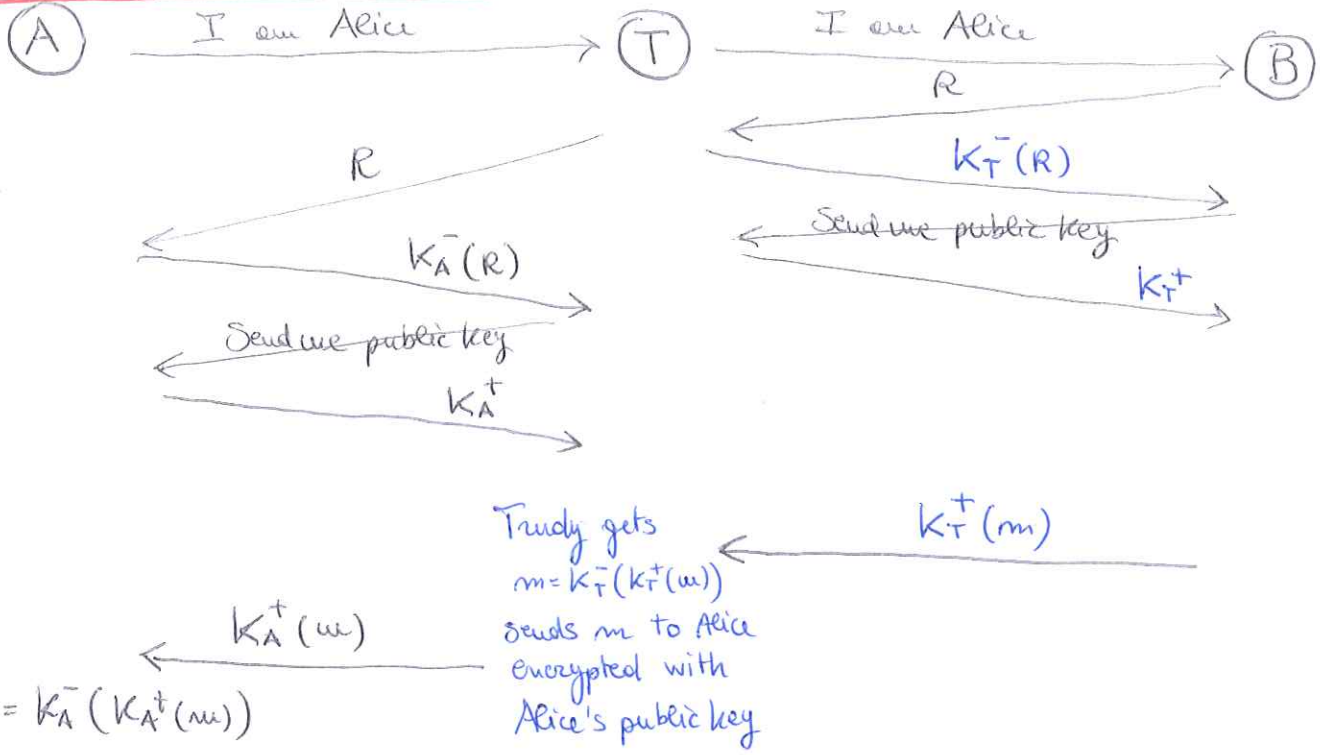
Chi invia la chiave pubblica e $K_A^-(R)$ siamo sicuri che sia Alice e che quelle siano le sue chiavi?

↳ NO → Possibile MAN IN THE MIDDLE ATTACK

Trudy si interpone tra Alice e Bob, filtrando tutto il traffico trasmesso, ma senza che loro si accorgano nemmeno di essere "spiati".

Vediamo come:

- MAN IN THE MIDDLE ATTACK:



- Difficile da rilevare:
  - Bob ed Alice ricevono entrambi tutto correttamente, ma Trudy anche!

* Il problema è, quando Alice ottiene la public key di Bob, come fa ad assicurarsi che sia di Bob e non di Trudy?

↳ SOLUZIONE: CERTIFICATION AUTHORITY (CA)

- CERTIFICATION AUTHORITIES:
  - La certification authority (CA) è un ente che collega le chiavi pubbliche alle entità, fornendo un CERTIFICATO, che viene FIRMATO DIGITALMENTE dalla CA.

$$\text{Certificato} = K_{CA}^-(K_B^+)$$

  - Quando Alice vuole ottenere la chiave pubblica di Bob, deve:
    1) Ottenere il certificato di Bob rilasciato dalla CA (da Bob o altrove)
    2) Decriptare il certificato con la chiave pubblica della CA (NOTA!):

$$K_B^+ = K_{CA}^+(K_{CA}^-(K_B^+))$$

* Funziona perché la chiave pubblica della CERTIFICATION AUTHORITY è NOTA e ben conosciuta!

SECURE E-MAIL

1) CONFIDENZIALITÀ:

Alice vuole inviare e-mail confidenziale (criptata) a Bob:

ALICE:

- Genera una chiave simmetrica K_S random
- Cifra il messaggio con K_S → $K_S(m)$
- Cifra la chiave simmetrica K_S con la chiave pubblica di Bob (RSA) → $K_B^+(K_S)$
- Invia a Bob: $K_S(m) + K_B^+(K_S)$

BOB:

- Decifra la chiave simmetrica usando la propria chiave privata: $K_S = K_B^-(K_B^+(K_S))$
- Usa K_S per decifrare il messaggio → $m = K_S(K_S(m))$

2) AUTENTICAZIONE:

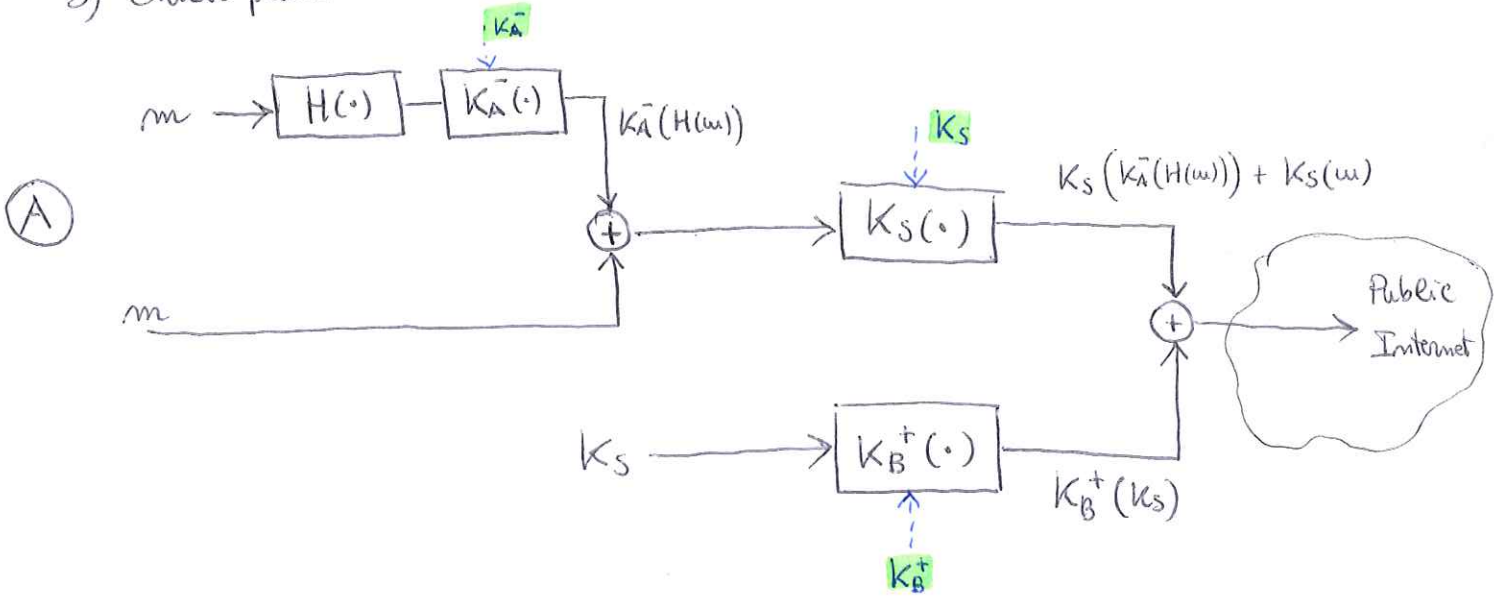
ALICE:

- Firma digitalmente il messaggio, generando un FINGERPRINT ottenuto firmando il digest del messaggio → $K_A^-(H(m))$
- Invia a Bob il messaggio IN CHIARO più il fingerprint → $m + K_A^-(H(m))$

3) CONFIDENZIALITÀ + AUTENTICAZIONE + INTEGRITÀ:

Alice usa 3 chiavi:

1) Chiave privata → Firma Digitale
2) Chiave simmetrica → Confidenzialità
3) Chiave pubblica di Bob → RSA, per comunicare K_S



- **PGP: Pretty Good Privacy**

Standard de facto per la crittografia di e-mail in Internet.
Use crittografia a chiave simmetrica, a chiave pubblica, funzioni hash e firme digitali.
Fornisce confidenzialità, autenticazione e integrità dei messaggi.

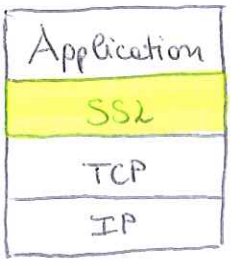
SSL: Secure Socket Layer

- Protocollo largamente diffuso, che dovrebbe supportare il livello 4, ma spesso viene definito come un livello 4 e 1/2.
  E' supportato da quasi tutti i browsers e i Web servers, permette HTTPS.
- Ci sono diverse varianti, tra cui TLS = Transport Layer Security
- Fornisce confidenzialità, integrità e autenticazione.

Obiettivi originari:
- Transazione nell' e-commerce Web
- cifratura, specialmente dei numeri delle carte di credito
- autenticazione del Web server
- autenticazione opzionale del client

* Disponibile per tutte le applicazioni che usano (TCP):

SECURE SOCKET INTERFACE



- SSL è implementato come una LIBRERIA, fornendo delle API alle applicazioni.

- **Toy SSL:**

Ci sono 3 fasi:

(1) Handshake: Alice e Bob usano i loro certificati e chiavi private per autenticarsi e'un e'altro e scambiarsi i segreti condivisi

(2) Key Derivation: Alice e Bob usano i segreti condivisi per derivare e'insieme di chiavi da usare (tipicamente almeno 4 chiavi)

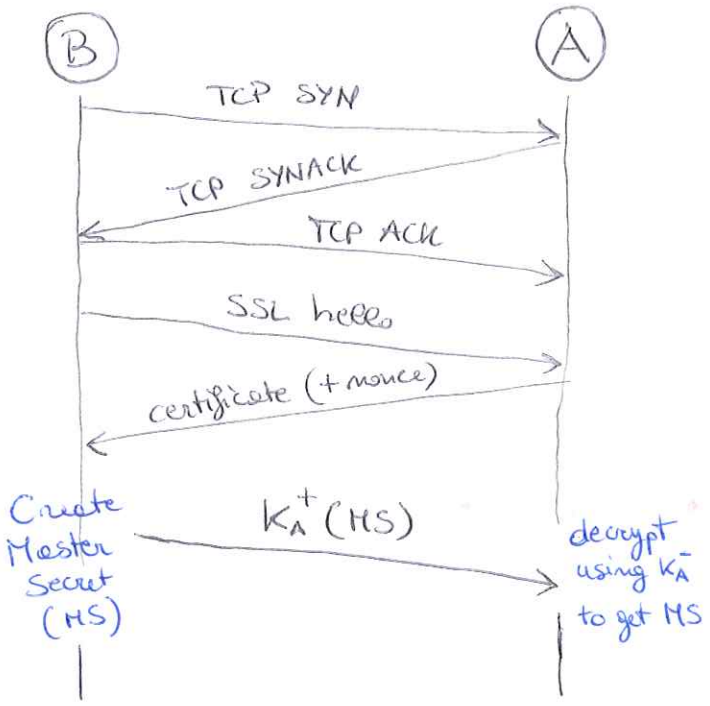
(3) Data Transfer: I dati da trasferire vengono suddivisi in records e trasmessi in sicurezza. Alla fine si inviano messaggi speciali per poter fare Connection Closure

• SSL: le 3 fasi

1. Handshake

- Bob stabilisce una connessione TCP con Alice
- autentica Alice tramite certificato firmato dalla Certification Authority
- Crea una **MASTER KEY**, la cifra con la chiave pubblica di Alice (RSA) e la invia

* Il tutto viene gestito anche con l'invio di nonce, non visibile in figure



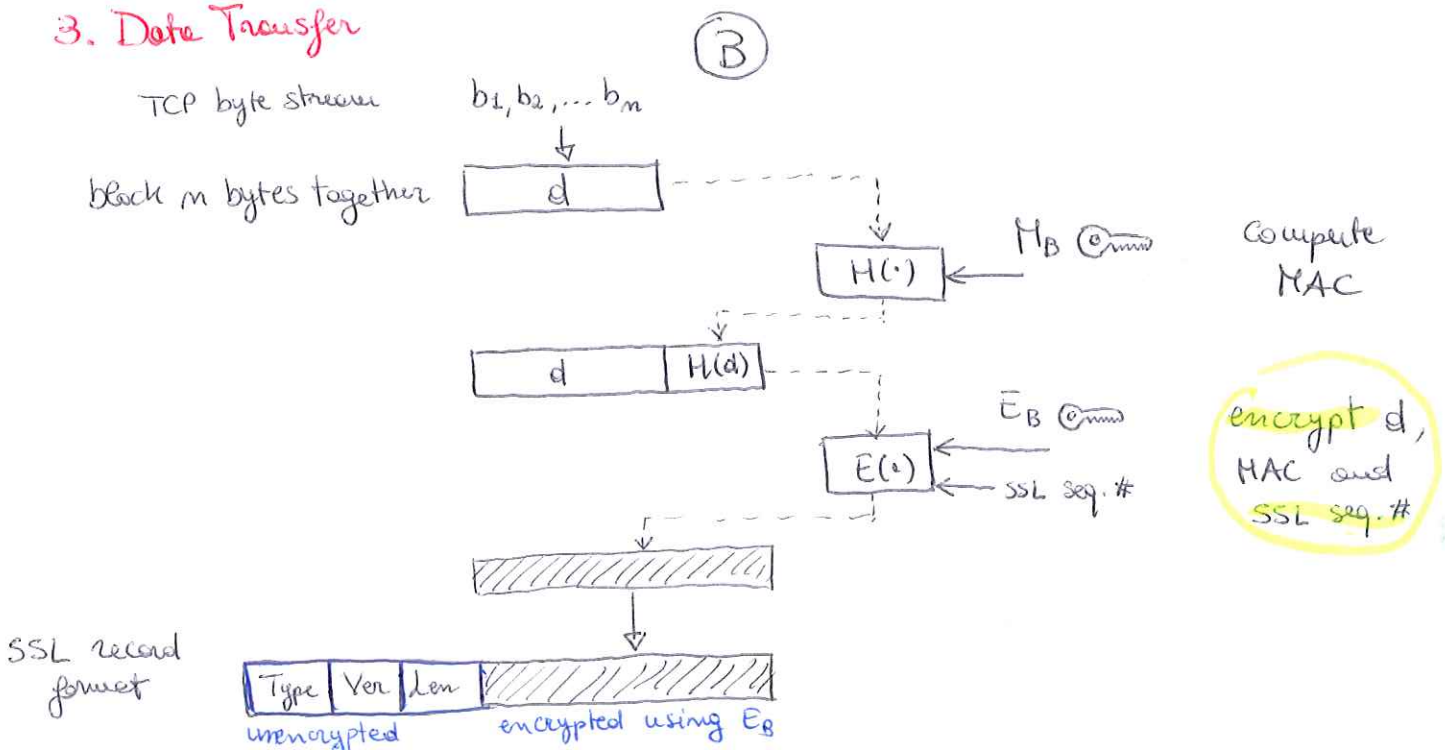
2. Key Derivation

Alice e Bob usano la chiave segreta condivisa (MS) per generare 4 chiavi, 2 a teste;

- E_B: Bob → Alice data encryption key
- E_A: Alice → Bob data encryption key
- M_B: Bob → Alice MAC key (MAC = Message Authentication Code)
- M_A: Alice → Bob MAC key

La cifrature e gli algoritmi di MAC sono negoziabili tra Bob e Alice.

3. Data Transfer



Sicurezza a Livello 3

- Il protocollo di sicurezza a livello 3 è conosciuto come **IPsec**
- Il sender cifra il payload dei datagrammi IP, che può essere un segmento TCP o UDP, un messaggio ICMP oppure OSPF ---

  Così, tutti i dati inviati da un'entità all'altra sarebbero occultati a terze parti
- Spesso, IPsec viene anche usato per creare delle VPN (Virtual Private Network) per delle istituzioni, sulla rete pubblica.

• VPN: Virtual Private Network

Un istituto, che spesso ha diverse sedi in aree geografiche diverse, desidera spesso una propria rete IP privata, in modo che i suoi host e server possano comunicare in maniera sicura. Tuttavia, una rete fisica è molto costosa !!!

↳ Usare una VPN e inviare il traffico sulla rete Internet pubblica:

- Il traffico tra uffici viene CIFRATO prima di essere inviato
- Il traffico è logicamente separato dall'altro traffico

• Servizi di IPsec:

data integrity + origin authentication + replay attack prevention + confidentiality

Ci sono 2 protocolli IP sec che forniscono diversi modelli di servizi:

- AH: Authentication Header
- ESP: Encapsulation Security Payload

- Innanzitutto, alla base, ci sono 2 modalità:

  (1) TRANSPORT MODE ⟶ Si cifra solo il payload

  (2) TUNNEL MODE ⟶ Si cifra tutto il datagramma IP e lo si incapsula nel nuovo datagramma IP, con nuovo header. Non si vedono vere sorgente e destinazione!

• I 2 Protocolli IPsec:

- AH (Authentication Header): fornisce autenticazione della sorgente e integrità dei dati, ma NON confidenzialità (no cifratura)
- ESP (Encapsulation Security Payload): fornisce tutto! Autenticazione + Integrità dati + Confidenzialità

* Combinazione più usata: Tunnel Mode con ESP

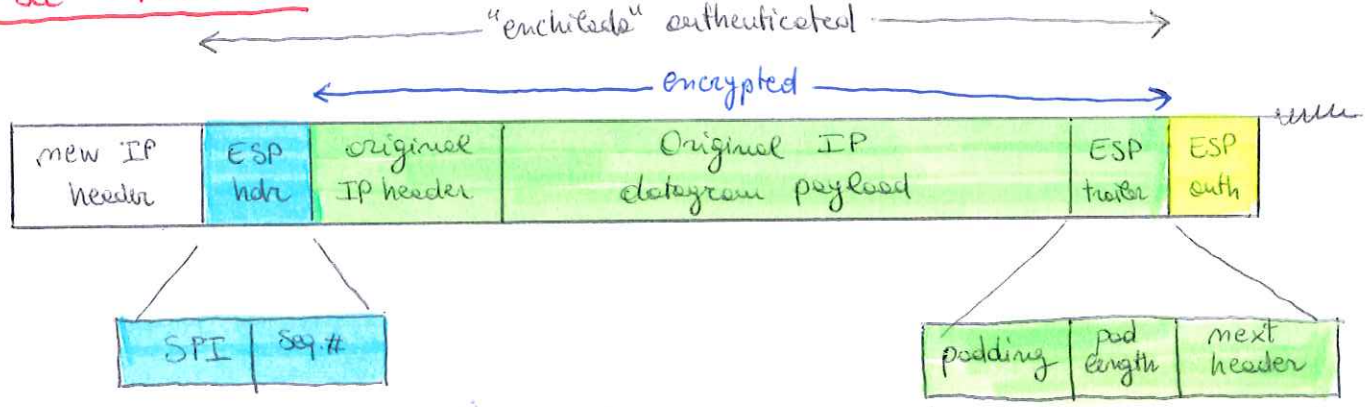
SECURITY ASSOCIATIONS (SAs):

- Una SECURITY ASSOCIATION (SA) è una connessione logica a livello di rete, instaurata tra host sorgente e destinazione.
  È simplex (unidirezionale) ⟹ Per uno scambio reciproco, ce ne vogliono 2
- Le entità / i routers devono mantenere informazioni sulle SAs:
  IPsec è CONNECTION-ORIENTED; IP no, è connectionless!
- Informazioni di stato SA:
  - SPI = Security Parameter Index; identificatore a 32-bit della SA
  - interfaccia d'origine della SA (e.g. 200.168.1.100)
  - interfaccia destinazione della SA
  - tipo di cifratura usata: e.g. 3DES con CBC (Cipher Block Chaining)
  - chiave di cifratura
  - tipo di controllo d'integrità usato: e.g. HMAC con MD5
  - chiave di autenticazione

SECURITY ASSOCIATION DATABASE (SAD):

- Spesso, bisogna mantenere lo stato di molte SAs, quindi viene mantenuto in una strutture dati presente nel Kernel del SO, detta SECURITY ASSOCIATION DATABASE
- Quando si deve inviare un datagramma IPsec, si accede al SAD per vedere come processarlo
- Quando si riceve, si esamina il SPI e lo si cerca nel SAD, per capire come processarlo.

IP sec DATAGRAM



- Seq # → parte da 0 e si incrementa.
  Serve per:
  1) Evitare REPLAY ATTACKS
  2) Risolvere la ricezione dei duplicati (window)

Cifratura a blocchi: serve per raggiungere dimensione fisse del messaggio!

- **SECURITY POLICY DATABASE (SPD):**

Policy = per un dato datagramma, l'entità mittente ha bisogno di sapere se usare o meno IPsec per trasmetterlo; nel qual caso, deve anche sapere quale SA usare

- Nell'SPD vengono salvate proprio queste informazioni

SPD → "COSA" fare con un datagramma in arrivo
SAD → "COME" farlo

- **IKE: INTERNET KEY EXCHANGE**

Decidere tutte le chiavi e i protocolli da usare per ogni SA manualmente è impossibile; c'è bisogno di generarli automaticamente.

Il protocollo IKE gestisce le cose:

- autenticazione reciproca e scambio di PSK (pre-shared key) oppure con PKI (public/private keys and certificates)
- PSK: entrambe le parti iniziano con una chiave segreta:
  - eseguono IKE per avere un canale sicuro per lo scambio di chiave e decisione di algoritmi. si autenticano e generano le SA IPsec in entrambe le direzioni, con cifratura e chiavi
- PKI: entrambe le parti iniziano con la coppia chiave pubblica/privata più il certificato
  - eseguono IKE per ~~creare~~ autenticarsi e creare le 2 SAs. Simile all'handshake SSL.

FIREWALLS

Un FIREWALL isola la rete interna di un'organizzazione dall'Internet pubblico, consentendo il passaggio di alcuni pacchetti e bloccandone altri.

- E' spesso collegato al router d'accesso, ma tipicamente il router NAT svolge anche la funzione di Firewall, così come fa da Local Name Server per il DNS!

- **FIREWALLS: PERCHÉ?**

1) Prevenire attacchi DoS: SYN flooding → l'attaccante invia molti SYN, ma quando riceve il SYNACK non invia l'ACK; così facendo, si occupa uno slot nella struttura dati del Server!
   Cambiando randomicamente ogni volta l'indirizzo IP sorgente, il Server non si accorge che è un attacco!

2) Prevenire modifiche o accessi illegali a dati interni: Ad esempio, sostituzione di pagine Web

3) Consentire solo accesso autorizzato all'interno della rete: insieme di utenti/hosts autenticati

- Ci sono 3 tipologie di FIREWALLS:
  1. STATELESS PACKET FILTERS
  2. STATEFUL PACKET FILTERS
  3. APPLICATION GATEWAYS

- **STATELESS PACKET FILTERING:**
  - La rete interna è connessa ad Internet mediante il router Firewall
  - Il router filtra pacchetto per pacchetto, decidendo se ritrasmetterlo o scartarlo in base a:
    - IP sorgente e IP destinazione
    - port numbers sorgente e destinazione di TCP/UDP
    - tipo di messaggio ICMP
    - bit SYN e ACK nei pacchetti TCP

* **NOTA**: un router, di livello 3, svolge compiti anche guardando informazioni di livello 4! (Indipendenti, ma fino a un certo punto!)

Esempio 1: Bloccare pacchetti entranti e uscenti con IP protocol field = 17 (UDP) o con porta sorgente o destinazione = 23 (TELNET)

Esempio 2: Bloccare pacchetti entranti con segmenti TCP con ACK = 0

→ blocca connessioni TCP dall'esterno verso l'interno, ma le permette dall'interno verso l'esterno.

- **ACCESS CONTROL LIST (ACL):**

ACL = Liste di regole, applicate con priorità top-to-bottom ai pacchetti: ogni regola è una coppia (action, condition)

action	source address	dest address	protocol	source port	dest port	flag bit
allow	222.22/16	outside of 222.22/16	TCP	> 1023	80	any
allow	outside of 222.22/16	222.22/16	TCP	80	> 1023	ACK
allow	222.22/16	outside of 222.22/16	UDP	> 1023	53	---
allow	outside of 222.22/16	222.22/16	UDP	53	> 1023	---
deny	all	all	all	all	all	all

Ad esempio, questa ACL permette solo traffico da e verso Server Web (TCP su porta 80) e da e verso Server DNS (UDP porta 53).

* **PROBLEMA:** Permette il passaggio anche di traffico che "NON ha senso"!
E.g. dest port = 80, ACK bit set, anche se non è stabilita una connessione TCP!

- È STATELESS perché esegue solo le istruzioni della tabella; non mantiene info sulle connessioni (TCP, ad esempio). NON guarda la STORIA del traffico!

- **STATEFUL PACKET FILTERING:**

* Tiene traccia di tutte le connessioni TCP!
  - traccia i SYN e i FIN per capire se i pacchetti che arrivano "hanno senso"
  - se scatta un timeout di connessione inattiva sul firewall, non fa più passare i pacchetti.
  - alla ACL viene aggiunto un campo "check connection" per capire se controllare lo stato della connessione prima di applicare la regola.

- **APPLICATION GATEWAYS:**

Filtrano i pacchetti guardando i campi IP/TCP/UDP, ma anche IL TRAFFICO DI LIVELLO APPLICATIVO

* Permettono di gestire privilegi differenti tra (utenti/hosts) differenti!

Esempio TELNET: il router Gateway fa da Server Telnet per i client interni, facendoli autenticare mediante credenziali e stabilendo se abilitarli o meno, in base alle politiche stabilite.
Poi, fa da Client nei confronti del VERO Server Telnet:
↳ Fa da (PROXY SERVER)!

- Spesso i Proxy Server Web fanno anche da Application Gateways per HTTP.

- **Limitazioni di Firewalls e Gateways:**
  - IP Spoofing: Si possono ingannare i firewalls modificando l'indirizzo IP sorgente. Un router non sa se un pacchetto arrive davvero dalla sorgente dichiarata!
  - Se più apps hanno bisogno di trattamenti speciali, ognuna deve avere il proprio application gateway
  - Il software del client deve sapere come contattare il gateway: e.g. bisogna settare l'indirizzo IP del proxy server nel browser Web
  - I filtraggi per UDP spesso prevedono politiche "all or nothing", eccetto che per il traffico DNS.
- TRADEOFF:

grado di comunicazione col mondo esterno	VS	livello di sicurezza

  - Diversi siti, molto protetti, sono ancora soggetti ad attacchi.

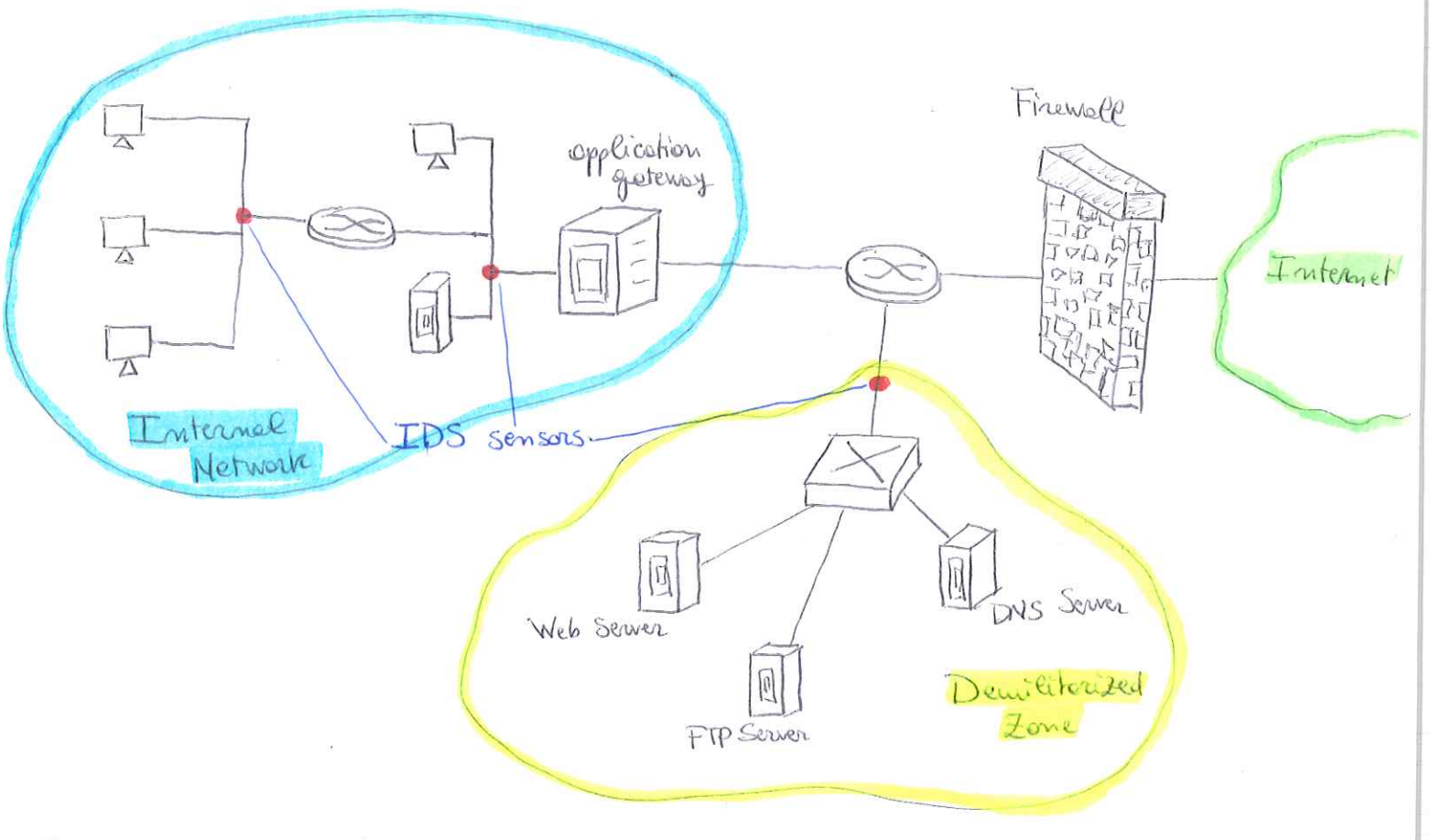
- **INTRUSION DETECTION SYSTEM (IDS):**

Sono degli apparati posti in più zone della rete interna, che fanno un'analisi più profonda e mirata del traffico, cercando di individuare situazioni anomale.

Fanno packet filtering controllando solo gli header TCP/IP, senza tenere traccia di connessioni.

- Fanno però deep packet inspection: controllano anche il PAYLOAD del pacchetto, alla ricerca di virus noti e/o stringhe d'attacco.

- Esaminano anche la correlazione tra pacchetti multipli:
  - port scanning
  - network mapping
  - DoS attack



- Zone demilitarizzate: restrizioni inferiori sugli incoming packets; è una zona in cui ci sono i Servers, e ci aspetto che vengano combattuti anche dall'esterno!