

CIFRATURA SIMMETRICA :

- CIFRARI A BLOCCI E OPERAZIONI
- CIFRARI PERFETTI
- ONE-TIME PAD
- SICUREZZA PRATICA E SICUREZZA INCONDIZIONATA

DATA ENCRYPTION STANDARD :

DISTRIBUZIONE DELLE CHIAVI :

- TERRA ENTRE' FIDATA : KDC, KTC
- PROTOCOLLO DIFFIE-HELLMANN
- L'ATTACCO MAN-IN-THE-MIDDLE

CIFRATURA A CHIAVE PUBBLICA :

- CRYPTO-SISTEMI A CHIAVE PUBBLICA
- RSA E IL PROBLEMA DELLA FATTORIZZAZIONE
- RSA IN PRATICA
- ALTRI CIFRARI ASIMMETRICI

INTEGRITA' E AUTENTICITA' DEI DATI :

- MANIPULATION DETECTION CODE (MDC)
- MESSAGE AUTHENTICATION CODE (MAC)
- INTEGRITA' DEI DATI E AUTENTICOZIONE DELL'ORIGINALE

FIRMA DIGITALE :

- FIRMA DIGITALE CON APPENDICE
- FIRMA DIGITALE CON RECUPERO DEL MESSAGGIO
- FIRMA DIGITALE BASATA SU RSA

INFRASTRUTTURA A CHIAVE PUBBLICA :

- CERTIFICATI
- LE INFRASTRUTTURE A CHIAVE PUBBLICA
- STANDARD X509 v3
- CONVI ALLA NORMATIVA ITALIANA

RANDOM E PSEUDO-RANDOM BIT GENERATORS :

- RANDOM BIT GENERATORS
- PSEUDORANDOM BIT GENERATORS
- SICUREZZA CRIPTOCARICA FISSA PRBG
- TEST STATISTICI

802.11 DATA LINK PROTOCOLS :

- TIPOLOGIE DI RETE
- WIRED EQUIVALENT PRIVACY (WEP)
- AUTENTICAZIONE E CONTROLLO DELL'ACCESSO

LOCAL BAN I

- NEEDHAM-SCHROEDER
- OTWAY REES
- SSL (VERSITA' VERSIONE)
- X509
- GSM

KERBEROS:

- ARCHITETTURA E PROTOCOLLO SEMPLIFICATI
- ARCHITETTURA COMPLETA
 - PRE-AUTHENTICATION
 - DELEGATION
 - ▷ PROXIMABLE TICKETS
 - ▷ FORWARDABLE TICKETS
- REALMS

SECURE SOCKET LAYER (SSL):

- ARCHITETTURA
- IL PROTOCOLLO RECORD
- IL PROTOCOLLO HANDSHAKE
- UTILIZZO DI SSL NEI PAGAMENTI ELETTRONICI
- UNITI DI SSL

CIFRATURA SIMMETRICA:

M: spazio dei messaggi
C: spazio dei cointogrammi
K: spazio delle chiavi

E: $P \times K \rightarrow C$ (cifratura)
D: $C \times K \rightarrow P$ (decifratura)

Proprietà: 1) $\forall m \in M, \forall e \in K, \exists d \in K : m = D(d, E(e, m))$
2) È facile a livello computazionale calcolare d sapendo e e viceversa

In pratica vale e=d.

SICUREZZA: un cifrario simmetrico (E,D) è sicuro se e solo se, dato $c = E(e, m)$ e $m = D(e, c)$:

- = Dato c è difficile determinare m senza conoscere e, e viceversa
- = Dato c ed m è difficile determinare e, a meno che la chiave non venga usata una volta sola

Se Alice manda un messaggio a Bob, Bob può sapere che $m = D(e, c)$ è giusto se:

- Bob conosce m in anticipo (è solo una conferma)
- Bob conosce una parte di m in anticipo (per email, ...)
- Bob sa che m ha delle ridondanze strutturali (testo ASCII)

Se un messaggio è considerato buono sono garantite:

- CONFIDENZIALITÀ (privacy): solo Alice ha visto il messaggio m
- PROVENIENZA: il messaggio viene sicuramente da Alice } non sempre
- INTEGRITÀ: se fosse stato modificato sarebbe illeggibile } sono garantite

Dire che Alice e Bob si fidano l'un l'altro vuol dire che:

- Alice crede che Bob non rivelerà m, e viceversa
- Alice crede che Bob terrà e segreta, e viceversa, ad esempio:
 - Alice crede che Bob sia competente nel gestire le chiavi, e viceversa
 - Alice crede che Bob non rivelerà mai la chiave, e viceversa

Tipi di cifrari: 1) cifroni a blocchi: criptano in insieme di t bit alla volta

2) cifrari a stream: cifrari a blocchi con $t = 1$

Ad ogni t bit corrisponde (o può corrispondere) una funzione di cifratura diversa

CIFRARIO A BLOCCHE:

$|P| = |C| = n$ bit \rightarrow lunghezza del blocco
 $|K| = k$ bit \rightarrow lunghezza della chiave

Per ogni K : $\Rightarrow E(K, P)$ deve essere invertibile
 $\Rightarrow D(K, P)$ è l'inversa di $E(K, P) = E_K(P)$

*TRUE RANDOM CIPHER: ogni chiave definisce una particolare permutazione E_K
= essendo 2^n i possibili messaggi, si ottengono $2^n!$ possibili sostituzioni
= la lunghezza della chiave è $k = \log(2^n!) \approx (n-1.44)2^n$, circa 2^n volte la lunghezza del blocco
d'obiettivo dell'avversario è:
• trovare il testo in chiaro dal testo cifrato (partial break)
• trovare anche la chiave (total break)

In generale si assume che:
• l'avversario ha accesso al metzo di comunicazione
• l'avversario conosce tutti i dettagli della funzione di cifratura
tranne la chiave segreta (kerckhoff assumption)

Classificazione degli attacchi:

- ciphertext-only attack: è noto il solo testo cifrato
- known-plaintext attack: è noto anche il testo in chiaro
- chosen plaintext attack: il testo in chiaro è stato scelto
- chosen ciphertext attack: il testo cifrato è stato scelto

Un cifrario sicuro a CHosen-PLAINTEXT es è anche contro known-plaintext e ciphertext-only attack: è meglio usare cifrari di questo tipo anche quando il chosen plaintext non è praticabile.

Un cifrario è COMPUTAZIONALMENTE SICURO se la complessità delle operazioni necessarie per romperlo usando il BEST KNOWN ATTACK superano di un margine rilevante le risorse computazionali di un ipotetico avversario (risorse limitate sempre!)

Complessità di un attacco si misura tramite:

- complessità dei dati: numero di dati di ingresso necessari (attesi)
- complessità di memorizzazione: numero di dati da memorizzare necessari (attesi)
- complessità di elaborazione: numero di operazioni necessarie per elaborare i dati di ingresso e memorizzare i dati intermedi

Un cifrario a blocchi è COMPUTAZIONALMENTE SICURO quando:

- n è abbastanza grande da evitare un'attacco auspicabile esauriva dei dati
- k è abbastanza grande da evitare una ricerca esauriva della chiave
- nessun attacco conosciuto ha complessità dei dati e complessità di elaborazione minori di 2^n e 2^k rispettivamente

RICERCA ESAUSTIVA DELLE CHIAVI: è un known plaintext attack che può essere considerato anche ciphertext-only attack se il testo in chiaro ha delle ridondanze strutturali. Si tratta di provare una ad una tutte le possibili chiavi (brute force).

ANALISI ESAUSTIVA DEI DATI: è un known plaintext attack in cui si costruiscono copie (testo in chiaro, testo cifrato) per una fissata chiave. Tale attacco al dizionario dovrebbe al più 2^n paia di (testo in chiaro, testo cifrato).

SOSTITUZIONE MONOALEFABETICA: le chiavi sono permutazioni dell'alfabeto

- Cifratura: ogni carattere in chiaro di posizione p nell'alfabeto viene sostituito con il corrispondente carattere di posizione p nella chiave
- Decifratura: ogni carattere del testo cifrato di posizione p nella chiave viene sostituito con il corrispondente carattere di posizione p nell'alfabeto

Il numero di chiavi è $26! - 1$ (26 sono i caratteri nell'alfabeto, 1 è la corrisp. diretta) Sembra sicuro, ma può essere rotto analizzando con un ciphertext-only attack basato sulla statistica della frequenza di ogni singolo carattere nella lingua data.

COTTOANALISI \rightarrow UNIARIA: per cifrari a blocchi e a caratteri (known plaintext attack)
 \rightarrow DIFFERENZIALE: concepita per cifrari a blocchi ma applicabile anche a chars (chosen plaintext attack)

CIFRARIO PERFETTO: quando un crittoanalista analizzando il crittogramma C , non acquisisce sul messaggio m , alcuna informazione di cui non disponeva già prima

Secondo Shannon: Data la variabile aleatoria M che assume valori nello spazio dei messaggi \mathcal{M} , e C la variabile aleatoria che assume valori nello spazio dei crittogrammi \mathcal{C} , si dice CIFRARIO PERFETTO se il cifrario che per ogni $m \in \mathcal{M}$, $\forall c \in \mathcal{C}$, vale $P(M=m | C=c) = P(M=m)$, ossia la probabilità a priori è uguale alla probabilità a posteriori (di un avversario di conoscere il testo in chiaro sapendo il crittogramma).

TEOREMA: In un cifrario perfetto il numero di chiavi deve essere \geq al numero ~~del messaggio~~ dei messaggi possibili

DIMOSTRAZIONE (PER ASSURDO): $N_m = \text{numero dei messaggi}$, $N_k = \text{numero delle chiavi}$; essendo $|M| = N_m \leq |\mathcal{C}|$ altrimenti il cifrario non è invertibile, si suppone per assurdo che $N_k < N_m$, quindi $N_k < N_m \leq |\mathcal{C}|$ implica $N_k < |\mathcal{C}|$. Dato un messaggio m , $P(M=m) \neq 0$, esiste quindi un ~~crittogramma~~ crittogramma $c \in \mathcal{C}$ che non è immagine di m , perciò $P(M=m | C=c) = 0 \neq P(M=m)$ contro l'ipotesi che un cifrario sia perfetto.

MODALITÀ DI CIFRATO GRAFIA:

- a) Electronic Code Book (ECB): ogni blocco in chiaro viene crittato separatamente
 - stesso testo in chiaro corrisponde allo stesso testo crittato, non si nasconde la struttura dei dati
 - i blocchi sono crittati separatamente, è permesso il mixtus e la sostituzione
 - la propagazione degli errori si limita al solo blocco
- b) Cipher Block Chaining (CBC): segue il principio di DIFFUSIONE di Shannon, introducendo dipendenza tra il blocco elaborazione e quelli precedenti. Data la dipendenza, stessi blocchi vengono crittati in modo diverso, eliminando la periodicità.
 - IV e la chiave determinano lo stesso testo in chiaro
 - IV può essere inviato in chiaro, ma garantisce l'integrità
 - il testo crittato dipende dal corrispettivo in chiaro e da tutti quelli precedenti
 - la propagazione degli errori si propaga sul blocco successivo
 - il recupero da errori è autocorrettore
- c) Cipher-feedback (CFB)
- d) Output feedback (OFB)

CRIPTATURA MULTIPLO: se un cifrario è sottoposto ad una ricerca esauriva della chiave allora ~~per~~ crittare un messaggio più volte potrebbe aumentarne la sicurezza.

- cifrari a cascata: ogni cifrario (1^o) ha una chiave indipendente
- cifratura multipla: i cifrari sono i soliti e le chiavi non necessariamente sono indipendenti fra loro
- CIFRATURA POPPA (k_1, k_2): soggetto ad attacchi di tipo known-plaintext chiamati meet-in-the-middle, che richiede 2^k operazioni e 2^k unità di memorizzazione
- CIFRATURA TRIPLO (k, k', k''): usato in operazioni finanziarie, un attacco di tipo chosen plaintext richiede 2^k operazioni, 2^k dati di ingresso e 2^k unità di memorizzazione, mentre un known-plaintext richiede p dati di ingresso, $\frac{2^{k+m}}{p}$ operazioni e una complessità di memorizzazione dell'ordine di $O(p)$. Si ottiene backward compatibility quando $k=k'$
- CIFRATURA TRIPLO (k, k', k''): un attacco di tipo known-plaintext simile al meet-in-the-middle richiede 2^{2k} operazioni e 2^k unità di memorizzazione. In combinazione con DES ($k=56$) il cifrario è praticamente sicuro.

ONE-TIME PAD (Vernam, 1917): dato un messaggio m su t bit e una chiave k vista come una sequenza di altri t bit scelti in modo randomico, si ha che:

- CIFRATURA: $c_i = m_i \oplus k_i, \forall 0 \leq i \leq t$
- DECIFRATURA: $m_i = c_i \oplus k_i, \forall 0 \leq i \leq t$

Se m è pernolico, dopo la cifratura, c non lo è!

TEOREMA: One-Time Pad è un cifrario perfetto se la chiave è scelta in modo perfettamente random per ogni messaggio e:

- 1) tutti i messaggi hanno lunghezza t
- 2) tutte le sequenze di t -bit sono messaggi possibili

DIMOSTRAZIONE: In un cifrario perfetto $P(M=m | C=c) = \frac{P(M=m, C=c)}{P(C=c)} = P(M=m)$.

$P(M=m, C=c)$ implica che esiste una chiave k tale che $c = m \oplus k$ e k è unica, perciò, se K fosse una variabile aleatoria che modella il protocollo generazione della chiave, si avrebbe che $P(M=m, C=c) = P(M=m, C=c, K=k)$, perciò:

$$P(M=m, C=c, K=k) = P(C=c | M=m, K=k) \times P(M=m, K=k)$$

$$= P(C=c | M=m, K=k) \times P(M=m | K=k) \times P(K=k) = P(M=m) \times (1/2)^t$$

Considerando la famiglia di eventi indipendenti $F_m = \{M=m\}$, gli eventi sono olginti e $P(\bigcup_m F_m) = 1$, perciò $P(C=c) = \sum_m P(C=c, M=m) = \sum_m P(M=m) \times (1/2)^t$.

$$P(M=m | C=c) = \frac{P(M=m) \times (1/2)^t}{(1/2)^t} = P(M=m) \text{ per definizione di cifrario perfetto.}$$

TEOREMA: One-Time Pad usa il numero minimo di chiavi

DIMOSTRAZIONE: siccome le chiavi sono sequenze random di t bit, $Nm \leq Nk$, allora $|M| = |K| = 2^t$. (avd.)

One-Time Pad è vulnerabile ad attacchi known-plaintext, in quanto la chiave può essere facilmente ottenuta tramite $k_i = m_i \oplus c_i$

→ per questo motivo, la chiave dev'essere usata una sola volta;

se forse usata due volte, allora avrà dove che $c = m \oplus k$ e $c' = m' \oplus k'$, quindi si ottiene che $m \oplus m' = c \oplus c'$, che aiuta molto al criptanalista (una sequenza di zero in $c \oplus c'$, si ripete anche in $m \oplus m'$)

One-Time Pad richiede una chiave di molti bit random, problema non semplice

SCUREZZA INCONDIZIONATA: si assume che:

- un avversario abbia risorse limitate
- l'avversario a posteriori dev'essere uguale all'avversario a priori (sul ciphertext)
- essendo perfetto, non fornisce informazioni all'avversario

Una condizione necessaria per i cifrari simmetrici per avere sicurezza incondizionata è avere scelto in modo random e indipendente i bit della chiave, di lunghezza maggiore o uguale al numero di bit del messaggio.

One-Time Pad è INCONDIZIONATAMENTE SICURO contro attacchi ciphertext-only.

La lunghezza della chiave, che dev'essere al minimo pari alla lunghezza del messaggio, complica la gestione delle chiavi e la loro distribuzione, perché si usano chiavi pseudo-random generate a partire da un segreto condiviso (piccolo). Ciò non rende più One-Time Pad incondizionatamente sicuro, ma almeno praticamente sicuro.

DATA ENCRYPTION STANDARD (DES): secondo Shannon alla base di un cifrario dev'essere:

- **Diffusione:** alterare la struttura del testo in chiaro "spargendone" i caratteri su tutto il testo cifrato, e si ottiene tramite:
 - Permutazione: rimane immutata la frequenza delle singole lettere, ma si perde l'informazione sulla frequenza dei q-grammi
 - Combinazione: si combinano i caratteri del testo in chiaro in modo che ognuno dei loro dipenda da molti altri essi: si perde l'informazione sulla frequenza delle singole lettere oltre a quella dei q-grammi
- **Confusione:** combinazione complessa del messaggio con la chiave per non permettere al crittanalista di separarle mediante un'analisi del cattogramma
 - Funzione non lineare del messaggio e della chiave per formare il cattogramma

In DES, la DIFFUSIONE si ottiene tramite permutazione ed espansione dei bit, mentre la CONFUSIONE si ottiene tramite sostituzione e successiva compressione dei bit del messaggio e della chiave. (Le stesse idee si trovano in AES).

La chiave è composta da 56 bit che dicontranno 64 aggiungendo un bit di parità ogni 8 bit ($8, 16, \dots, 64$).

La cifratura è composta da 16 round, ognuna con una differente chiave proveniente da un key scheduler: all'inizio e alla fine si operano due permutazioni per generare diffusione. Ogni round opera secondo $L_i = R_{i-1}$ e $R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$, dove F è la funzione non lineare che genera confusione.

Se c'è un criterio preciso con cui sono state disegnate le parti non lineari (S-box) non è completamente noto, ma si pensa che l'NSA le abbia progettate inserendo trascrittelli per decifrare messaggi. Non c'è comunque prova che tali trascrittelli esistano.

DES viene usato per:

- transazioni bancarie
- cifrare PIN e account in ATM
- interbanking in organizzazioni governative

Proprietà di DES:

- 1) Ogni bit del crittogramma dipende da tutti i bit della chiave e da tutti i bit del messaggio
- 2) Non ci sono relazioni statistiche evidenti tra il messaggio ed il crittogramma (per la crittoanalisi)
- 3) d'alterazione di un bit nel messaggio provoca un'alterazione di ciascun bit del crittogramma con probabilità 0.5

Anomalie di DES:

- Chiavi deboli: una chiave è debole se $E_k(E_k(x)) = x$, $\forall x \in P$.
- Chiavi semi-debolli: una coppia di chiavi è semi-debolle se $E_{k_1}(E_{k_2}(x)) = x$, $\forall x \in P$

\Rightarrow DES ha 4 chiavi deboli e 6 coppie di chiavi semi-debolli (note)

- Complementarietà: se $m = E_a(a)$ allora $\bar{m} = E_{\bar{a}}(\bar{a})$
 - rende possibile un chosen-plaintext attack tramite ricerca esauritiva delle chiavi, con spazio delle chiavi dimezzato
- DES non è raggruppabile, ovvero, $\forall m \in M$, $\forall k_1, k_2 \in K$ $\neq k_3 \in K$: $E_{k_1}(E_{k_2}(m)) \neq E_{k_3}(m)$

\Rightarrow cifrare più volte è meglio di cifrare una sola volta

DISTRIBUZIONE DI UNA CHIAVE:

PUNTO-PUNTO:

- ogni utente ha una chiave segreta a priori con un altro utente
- ogni utente memorizza $n-1$ chiavi
- il numero totale di chiavi è $\frac{n \times (n-1)}{2} \approx \frac{n^2}{2} \Rightarrow O(n^2)$

PRO: - Se viene compromesso un utente, vengono compromesse solo le sue comunicazioni

CONTRO: - Scalabilità ridotta: il numero di chiavi aumenta in modo quadratico al numero degli utenti e l'ingresso o l'uscita di un membro si riflette su tutti gli altri

SESSIONE:

- (CHIAVE EFFIMERA)
- le parti si conoscono e condividono a priori una chiave W
- le parti vogliono comunicare tramite una chiave di sessione K
- la chiave W viene usata per stabilire la chiave K

- 1) ONE-PASS: $A \rightarrow B: E_W(t_A, B, K) \dashrightarrow t_A: \text{timestamp (reg. sync.)}$
- 2) CHALL.-RESP.: $B \rightarrow A: m_B \dashrightarrow A \rightarrow B: E_W(m_B, B, K) \dashrightarrow m_B: \text{nonce (fresh)}$
- 3) BOTH PART.: $B \rightarrow A: m_B$
 $A \rightarrow B: E_W(K_B, m_B, m_A, B) \dashrightarrow K = f(K_A, K_B)$
 $B \rightarrow A: E_W(K_B, m_A, m_B, A)$

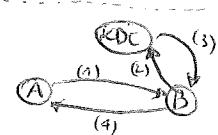
- CON TUTTA ENTITÀ FIDATA:
 - aiuta le parti a stabilire una comunicazione
 - ogni utente mantiene una chiave segreta a priori con T
 - il numero di chiavi segrete a lungo termine è $O(n)$
 - da TTP:
 - mantiene un database $\langle U, K_U \rangle$
 - garantisce l'integrità e la segretezza del database
 - applica correttamente il protocollo

KEY DISTRIBUTION CENTER:



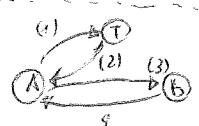
- A e B scambiano con T le chiavi segrete K_{AU} e K_{AB}
- KDC genera la chiave K e la invia ad A e B
- ~~A e B si fidano del metodo di KDC di generare le chiavi.~~

ONE-WAY-KETS PROTOCOL:



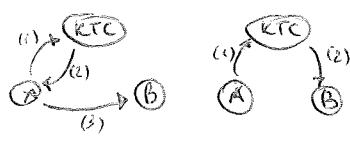
- M1) $A \rightarrow B : M_A, A, B, E_{KAT}(M_A, M_A, A, B)$
 - M2) $B \rightarrow T : M_A, A, B, E_{KAT}(M_A, M_A, A, B), E_{KBT}(M_B, M_B, B, A)$
 - M3) $T \rightarrow B : M_B, E_{KAT}(M_A, K_{AB}), E_{KBT}(M_B, K_{AB})$
 - M4) $B \rightarrow A : M_B, E_{KAT}(M_A, K_{AB})$
- M, M_A, M_B : monces

KERBEROS (UNIX DIRECTORY):



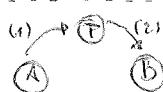
- M1) $A \rightarrow T : T_A, B$
 - M2) $T \rightarrow A : E_{KAT}(T, L, K_{AB}, B), E_{KBT}(T, L, K_{AB}, A)$
 - M3) $A \rightarrow B : E_{KBT}(T, L, K_{AB}, A), E_{KAB}(A, T)$
 - M4) $B \rightarrow A : E_{KAB}(T+1)$
- L: lifetime
T: timestamp

KEY TRANSACTION CENTER:



- A e B scambiano con T le chiavi segrete K_{AU} e K_{AB}
- una delle due parti genera la chiave di sessione K che T trasmette all'altra parte
- da parte si fida dell'altra sul metodo di generare le chiavi

WIDE-MOUNTED KTC Protocol:



- M1) $A \rightarrow T : T_A, E_{KAT}(T_A, B, K_{AB})$
- M2) $T \rightarrow B : T_B, E_{KBT}(T_B, A, K_{AB})$

T: timestamp

PRO: - facile aggiungere o rimuovere entità dalla rete
- ogni entità deve memorizzare solo un segreto a lungo termine

CONTRO: - ogni comunicazione richiede l'interazione iniziale con la TTP
- la TTP deve memorizzare n segreti a lungo termine
- la TTP ha la possibilità di leggere tutti i messaggi
- se la TTP è compromessa, tutte le comunicazioni sono inutile

SISTEMA A CHIAVE PUBBLICA: permette di far comunicare due persone ~~tra~~ ^{in un canale sicuro} attraverso

Problema del logoritmo discreto:

- Sia p un numero primo
- Sia g un numero generato compreso tra $1 \leq g < p$
- $\forall n, 1 \leq n < p, \exists t : g^t \bmod p = n$

→ ESPONENZIALE DISCRETO: dati g e y è facile determinare $g^x \bmod p$
→ LOGARITMO DISCRETO: dati g e y , $1 \leq y \leq p-1$ è computazionalmente difficile determinare x , $0 \leq x \leq p-1$ tale che $y = g^x \bmod p$
 $x \triangleq \log_g y \bmod p$

Con ciò alla base del protocollo DIFFIE-HELLMAN.

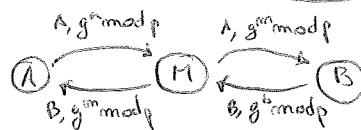
DIFFIE-HELLMAN PROTOCOL



- Sia p un numero primo
- Sia g un generatore, con $1 \leq g < p$
- Siano a e b numeri pubblici o conosciuti
- Alice sceglie un numero random a
- Bob sceglie un numero random b
- M1) $A \rightarrow B : A, Y_A = g^a \text{ mod } p$
- M2) $B \rightarrow A : B, Y_B = g^b \text{ mod } p$
- Alice determina $K_{AB} = (Y_B)^a \text{ mod } p = g^{ab} \text{ mod } p$
- Bob determina $K_{AB} = (Y_A)^b \text{ mod } p = g^{ab} \text{ mod } p$

Sicurezza: un avversario può determinare K_{AB} a partire da Y_A e Y_B tramite il calcolo di $[K_{AB} = Y_A \log_g Y_B \text{ mod } p]$, quindi se $\log \text{ mod } p$ fosse facile da determinare allora l'algoritmo sarebbe rotto. Purtroppo non c'è prova del contrario, cioè che se $\log \text{ mod } p$ fosse difficile da determinare allora il sistema è sicuro. Non è stato trovato alcun modo di calcolare K_{AB} da Y_A e Y_B senza prima ottenere anche a e b .

ATTACCO MAN-IN-THE-MIDDLE



$$\begin{aligned} \text{Alice - Lou C. : } K_{AM} &= g^{am} \text{ mod } p \\ \text{Bob - Lou C. : } K_{BM} &= g^{bm} \text{ mod } p \end{aligned}$$

L'avversario può leggere i messaggi tra Alice e Bob e impersonarli (inietta messaggi).



Anche se la coppia $\langle U, Y_U \rangle$ fosse pubblica e solo leggibile sarebbe comunque possibile l'attacco dell'uomo nel mezzo.

- PRO :
- Non è richiesta la presenza di una TTP
 - Il file pubblico può essere memorizzato da qualsiasi delle parti
 - Solo i file pubblici devono essere memorizzati, quindi l'unico attacco possibile avviene da un'avversario passivo

- CONTRO:
- La gestione delle chiavi diventa più complessa in presenza di un'avversario attivo.

→ **SISTEMA A CHIAVE PUBBLICA**: una TTP può certificare ogni chiave pubblica nel file in modo che l'uomo nel mezzo non la possa impersonare

- PRO :
- non è possibile e' impersonificazione da un'attivo attivo
 - le entità si fidano della TTP solo nell'associare la coppia correttamente, la TTP non può controllare la comunicazione
 - I certificati possono essere memorizzati localmente per eliminare l'interazione con il file pubblico

- CONTRO:
- se la firma delle TTP è compromessa, ogni comunicazione diventa insicura
 - tutta la fiducia è rimandata ad una sola entità

SCHEMA A CIFRATURA ASIMMETRICA: Date due trasformazioni "invertibili": tale che:

Cifratura: $\{E_e : e \in K\}$, $E_e : M \rightarrow C$

Decifratura: $\{D_d : d \in K\}$, $D_d : C \rightarrow M$

- = la chiave di cifratura \leftarrow
e può essere pubblica
- = la chiave di decifratura
d DEVE ESSERE SEGRETA

• $e \in K$, $\exists ! d \in K : D_d = E_e^{-1}$

• Vmelli, $\forall c \in C$, $E_e(m) = D_d(m)$ sono facili da calcolare

- Dati $e \in K$ e $c \in C$, dev'essere praticamente impossibile trovare il messaggio $m \in M$ tale che $E_e(m) = c$
- Dati $e \in K$, dev'essere praticamente impossibile trovare la corrispondente chiave $d \in K$.

obiettivi dell'avversario:

- Rompere il sistema, ossia ottenere il testo in chiaro dal testo cifrato
- Rompere completamente il sistema, ossia trovare la chiave

Tipi di attacchi:

- Nessun cifrario asimmetrico è perfetto
- Dato che la chiave di cifratura è nota, un avversario possiede sempre attivare un chosen-plaintext attack
- Un attacco più forte è il chosen-ciphertext in cui l'avversario invia del testo cifrato a sua scelta e ottiene dalla vittima in qualche modo il testo in chiaro

RIVEST SHAMIR ADLEMAN (RSA, 1978):

- Generazione delle chiavi:
 - si generano due numeri primi diversi molto grandi ($100 \div 200$ cifre decimali) p e q
 - si calcola $n = p \cdot q$ e $\phi = (p-1)(q-1)$
 - si sceglie un numero random e , $1 < e < \phi$, tale che $\text{mcd}(e, \phi) = 1$
 - si calcola l'unico intero d , $1 < d < \phi$, tale che $ed \equiv 1 \pmod{\phi}$
 - (d, n) è la chiave privata
 - (e, n) è la chiave pubblica
- Cifratura: per generare c da m è necessario:
 - ottenere la chiave pubblica autentica della seconda parte (e, n)
 - rappresentare il messaggio come un intero m nell'intervallo $[0, n-1]$, $0 \leq m \leq n-1$, $0 \leq m \leq n$
 - calcolare $c = m^e \pmod{n}$ ed inviarlo alla seconda parte
- Decifratura: per recuperare m a partire da c è necessario:
 - calcolare $m = c^d \pmod{n}$, conoscendo la chiave privata d

DIMOSTRAZIONE (funzionamento); bisogna provare che $D_d(E_e(m)) = m$

$$c^d \equiv_m m \stackrel{\text{def}}{=} m^{k\phi+1} \pmod{n}, \quad \forall k \in \mathbb{Z} \stackrel{\text{def}}{=} (m^\phi)^k \cdot m \pmod{n} \equiv m \pmod{n}$$

applicando
il teorema di Eulero

teorema di Eulero: $\forall m \in \mathbb{Z}, m > 1, \forall a \in \mathbb{Z}_m^*, a^{\phi(m)} \equiv 1 \pmod{m}$

$$\mathbb{Z}_m^* = \{x \mid 1 < x < m, \text{mcd}(x, m) = 1\}$$

Cifrare/decifrare efficientemente:

- RSA richiede il calcolo dell'esponentiale modulare $c^d \text{ mod } n$
 - Se n avesse k bit allora $k = \log n + 1$ in binario
- Grade-school algorithm richiede $(d-1)$ moltiplicazioni modulari
 - d è grande quanto ϕ che aumenta esponenzialmente rispetto a k
 - non è efficiente
- Square-and-multiply algorithm richiede $2k$ moltiplicazioni modulari con k pari al numero di bit della rappresentazione di d
 - essendo $k \leq k$, l'algoritmo ha complessità $O(k^2)$

RSA PROBLEM (RSAP): trovare il testo in chiaro m a partire dal testo cifrato c conoscendo solo le informazioni pubbliche (e, n) .

- La fattorizzazione ha complessità maggiore o uguale a RSAP
- RSAP non è più complesso della FATTORIZZAZIONE

Nonostante sia largamente diffuso che RSAP e la fattorizzazione intera siano computazionalmente equivalenti, non c'è prova di questo.

RADICE E-ESTIMA: un altro modo di decifrare $c = m^e \text{ mod } n$ è calcolare $\sqrt[e]{c}$

- se n fosse primo, il problema è computazionalmente semplice
- se n non è primo, il problema è equivalente alla fattorizzazione

TOTAL BREAK: un modo possibile di rompere totalmente RSA è calcolare $\phi(n)$, ma è computazionalmente equivalente alla fattorizzazione di n

- dati p e q , $n = p \cdot q$, ~~calcolare~~ ottenere $\phi(n)$ è immediato
- Viceversa, se $\phi(n)$ fosse dato, allora $\phi(n) = (p-1)(q-1) = m - (p+q) + 1$
 - si ottiene $x_1 = (p+q) = m + 1 - \phi(n)$. Da $(p-q)^2 = (p+q)^2 - 4m$, quindi $(p-q)(p+q) = (p+q)^2 - 4m$, perciò $x_2 = (p-q) = (p+q)^2 - \frac{4m}{p+q}$. Infine si ottiene $p = \frac{x_1 + x_2}{2}$ e $q = \frac{x_1 - x_2}{2}$.

ATTACCO FLESSIVO ALL'RSA: per rompere totalmente l'algoritmo, ma più difficile della fattorizzazione, perché in base alla scelta di e , d può essere più grande di $p \cdot q$ può essere

ADAPTIVE CHOSEN-CIPHERTEXT: un attacco chosen-ciphertext si realizza scegliendo il testo cifrato e cercando di ottenere quello in chiaro (tramite la parte).

Un attacco adaptive chosen-ciphertext si performa scegliendo il testo cifrato in base ~~a~~ al testo in chiaro ricevuto nelle richieste precedenti!

HOMOMORPHIC PROPERTY OF RSA:

- Siano m_1 e m_2 due testi in chiaro
- Siano c_1 e c_2 i rispettivi testi cifrati
- Si osserva che $(m_1 \cdot m_2)^e \equiv m_1^e \cdot m_2^e \equiv c_1 \cdot c_2 \pmod{n}$

⇒ Il testo ~~cifrato~~ del prodotto (di due testi in chiaro) corrisponde al prodotto dei singoli testi cifrati

Essendo le cui cifrari assimmetrici si usano anche in combinatorie a quelli simmetrici in sistemi BRIDI, in quanto quelli simmetrici sono più veloci.

RSA IN PRATICA: RSA è lento rispetto ai cifrari simmetrici e viene perciò usato per il trasporto di chiavi di sessione o per piccole quantità

- Regole pratiche:
- p, q devono essere scelti in modo che la fattorizzazione di $n = p \cdot q$ sia impossibile, quindi
 - p, q devono essere abbastanza grandi e circa della stessa quantità di bit (per evitare l'attacco **NUMBER CRUNCHING**)
 - p, q non devono essere troppo piccoli
 - l'esponente e dev'essere piccolo o con un numero parso di bit o 1
 - l'esponente d dev'essere della stessa lunghezza di n circa

Well Knowns Attack against RSA:

- COMMON MODULUS ATTACK: l'avversario richiede una chiave privata (d_x, n) da cui può determinare efficientemente la fattorizzazione di n grazie a d_x . In questo modo può ottenere tutte le chiavi d_i che hanno lo stesso n
 - SMALL ~~MESSAGE~~ MESSAGE SPACE ATTACK: messaggi di piccole dimensioni, rengono modificati per ottenere il risultato sperato (es. asta.)
 - CHINESE REMAINDER THEOREM (CRT): dato l'insieme di numeri primi interi positivi m_1, m_2, \dots, m_r e gli interi positivi a_1, a_2, \dots, a_r , allora il sistema di congruenze $x \equiv a_i \pmod{m_i}, 1 \leq i \leq r$, ha un'unica soluzione modulo $M = m_1 \times m_2 \times \dots \times m_r$ data da:
- $$x = \sum_{i=1}^r a_i M_i y_i \pmod{M}, \text{ con } M_i = \frac{M}{m_i}, y_i = M_i^{-1} \pmod{m_i} \quad (1 \leq i \leq r)$$

Si vuole determinare $m = c^d \pmod{n}$:

- CRT ci aiuta a trovare $x \pmod{pq}$ da $x \pmod{p}$ e $x \pmod{q}$
- Il piccolo teorema di Fermat afferma che $a^{p-1} \pmod{p} = 1$.

Algoritmo:

- (ottimizzato)
 - calcola $m_1 = c^d \pmod{p}$ e $m_2 = c^d \pmod{q}$
 - determina (Fermat) $m_1 = c^{d \pmod{p-1}} \pmod{p}$ e $m_2 = c^{d \pmod{q-1}} \pmod{q}$
 - determina (CRT) $a_1 = q^{-1} \pmod{p}$ e $a_2 = p^{-1} \pmod{q}$
 - infine $m = a_1 m_1 q + a_2 m_2 p$

$\Rightarrow b_1 = c^d \pmod{p} = c^{d \pmod{p-1}} \pmod{p}$, dove d è su k bit, mentre $p-1$ è su $\frac{k}{2}$ bit, perciò è necessario $O(k^2/4)$

- FAULT INJECTION ATTACK: l'implementazione di RSA con CRT causando:
 - danni hardware nel calcolare m_1 , producendo m_1' e quindi $m' = a_1 m_1' q + a_2 m_2 p$
 - Ne segue che $m - m' = a_1 (m_1' - m_1) q$, perciò $\text{gcd}(m - m', n) = q$ può essere efficientemente determinato dall'algoritmo di Euclideo

- \Rightarrow circuito compromesso
- \Rightarrow sistemi embedded con processore fisico
- \Rightarrow l'unica contromisura è il controllo dei risultati

- **LOW EXPONENT ATTACK:** se tre utenti avessero 3 differenti parametri rispettivamente m_1, m_2, m_3 , ma con lo stesso esponente di cifratura $e=3$ (piccolo), allora per mezzo del CRT si trova $x = m^3 \pmod{m_1 m_2 m_3}$. Dato che $m < m_i$ per definizione di cifratura in RSA, segue che $m^3 < m_1 m_2 m_3$, perciò vale che $x = m^3$. Di conseguenza un avversario può recuperare m solamente determinando la radice cubica di x .

ALTRI CIFRATI ASIMMETRICI:

- **SISTEMA A LOGARITMO DISCRETO:** dato un numero primo p e un divisore primo q di $p-1$, dato un $g \in [1, p-1]$ ($1 \leq g \leq p$) di ordine q , si ottiene la chiave privata x scegliendola in modo random da $1 \leq x \leq q$, e la chiave pubblica y scegliendola come $y = g^x \pmod{p}$.

PROBLEMA DEL LOGARITMO DISCRETO (DLP): ottenere x dati (p, q, g) e y .

- **ELGAMAL:**
 - si sceglie un numero k random
 - $c_1 = g^k \pmod{p}$, $c_2 = m \cdot g^k \pmod{p}$
 - invia al destinatario il messaggio cifrato (c_1, c_2)

Decifratura:

- $c_1^x = g^{kx} \pmod{p} = y^k \pmod{p}$
- $m = c_2 \cdot y^{-k} \pmod{p}$

Sicurezza: l'avversario ha bisogno di determinare $y^k \pmod{p}$ a partire da (p, q, g) e y , il che è equivalente a DH P , ovvero DLP in \mathbb{Z}_p .

- **CURVA ELLITICA:** generazione delle chiavi:

- Sia dato $p \in \mathbb{F}_p$
- Sia E la curva ellittica definita da $y^2 = x^3 + ax + b \pmod{p}$ con $a, b \in \mathbb{F}_p$ e $4a^3 + 27b^2 \neq 0$
- L'insieme di punti $E(\mathbb{F}_p)$ che punta ad ∞ forma un gruppo additivo
- Sia P di ordine n , allora il sottogruppo $\langle P \rangle$ generato da P è dato da $\langle \infty, P, 2P, \dots, (n-1)P \rangle$
- p, E, P sono i parametri pubblici
- la chiave privata è scelta in modo random tra $1 \leq d < n$
- la chiave pubblica è data da $Q = dP$

Cifratura: se il messaggio M rappresenta un punto

- $c_1 = kP$, $c_2 = M + kQ$
- invia al destinatario (c_1, c_2)

Decifratura:

- $dC_1 = d(kP) = kQ$
- $M = c_2 - dC_1$

Sicurezza: la complessità nel calcolo di kQ dati i parametri di dominio, Q e $C_1 = kP$, è EC $\mathbb{B}H\mathbb{P}$.

FUNZIONI HASH E INTEGRITÀ DEI DATI:

- Integrità del messaggio: proprietà per la quale i dati non sono stati alterati in modo non autorizzato dal momento della generazione, della trasmissione, o della memorizzazione
- Autenticazione dell'origine del messaggio: è un tipo di autenticazione in cui si conferma l'originalità del mittente dei dati creati in qualche momento nel passato
- \Rightarrow l'autenticazione dell'origine dei dati include l'integrità dei dati

FUNZIONI HASH: devono essere FACILI da calcolare, UNICHE, DIFFICILI da invertire.
Il messaggio hash può essere usato per garantire l'integrità e l'autenticazione del messaggio, e rappresentarlo univocamente.

- Proprietà:
 - **COMPRESSESSO**: mappare un numero arbitrario di bit di ingresso in un insieme fisso di bit di uscita
 - **FACILITÀ DI CALCOLO**: dato x , si trova facilmente $h(x)$
 - **COLLISIONI POSSIBILI**: essendo una funzione many-to-one

Proprietà per la sicurezza:

- **PREIMAGINE RESISTENZA**: dev'essere impossibile dato un output specificato trovare l'input corrispondente
- **2ND-PREIMAGINE RESISTENZA**: dato un ingresso, dev'essere impossibile trovare un altro ingresso che si mappa sulla stessa uscita
- **COLLISION RESISTANCE**: dev'essere impossibile trovare una coppia di valori di ingresso che mappano la stessa uscita

\Rightarrow Motivazioni delle proprietà per la sicurezza:

- \rightarrow Preimage resistance (es.): nella firma digitale basata su RSA, (m, d) è la chiave privata, (n, e) è la chiave pubblica, mentre la firma si ottiene da m come $s = h(m)^d \text{ mod } n$. Se non fosse resistente ad attacchi di preimage, un avversario potrebbe selezionare uno $z < n$, calcolare $y = z^e \text{ mod } n$ e trovare un m' tale che $h(m') = y$, offrendo che z è la firma del messaggio m' .
- \rightarrow 2nd-Preimage resistance (es.): nella firma digitale con appendice la firma è $s = S(h(m))$. Una terza entità fidata sceglie un messaggio m da far firmare al mittente $s = S_A(h(m))$, perciò, se non fosse resistente ad attacchi di 2nd-preimage, un avversario può determinare una seconda immagine m' tale che $h(m') = h(m)$ e affermare che il mittente ha firmato m' invece di m .
- \rightarrow Collision resistance (es.): nella firma digitale con appendice, la firma è $s = S(h(m))$; se h non è resistente ad attacchi di tipo collision, un avversario può scegliere due messaggi m ed m' tali che $h(m) = h(m')$ e determinare $s = S_A(h(m))$, ~~involandosi~~ $\langle m, s \rangle$ ad un mittente e poi affermando di aver inviato $\langle m', s \rangle$

MANIPULATION DETECTION CODE (MDC): classificazione:

- (OWHF) One Way Hash Function: funzione hash resistente ad attacchi di PREIMAGE e 2ND-PREIMAGE.
- (CRHF) Collision Resistant Hash Function: funzione hash resistente a: 2ND-PREIMAGE e COLLISION.

Collision resistance implica anche 2nd-preimage resistance, ma non preimage resistance, anche se in pratica CRHF sono anche resistenti a questo tipo di attacco.

La gravità delle conseguenze pratiche di un attacco dipende dal grado di controllo che ha l'avversario sul messaggio per il quale l'MDC può essere controllato:

- CONTRAFFAIZIONE SELETTIVA: l'avversario ~~non~~ controlla l'intero messaggio o parte di esso
(selective forgery)
- CONTRAFFAIZIONE ESISTENZIALE: l'avversario non ha alcun controllo sul messaggio (al massimo ce l'ha sull'hash)
(existential forgery)

Classificazione in base alla comprensione attuata:

- = hash basate su cifrari a blocchi: invertibili, quindi insicure
- = hash personalizzate: quelle realmente utilizzate (ad hoc)
- = hash basate sull'aritmetica modulare (es. RSA): fallimenti totali

MESSAGE AUTHENTICATION CODE (MAC): servono a garantire l'autenticazione del messaggio usando tecniche simmetriche

Proprietà:

- insieme di funzioni h_k parametrizzate da un segreto (chiave) k
- FACILITÀ DI CALCOLO: data h_k e k , è facile ottenere $h_k(x)$ da x
- COMPRESSEONE: mappa un numero finito di bit in un numero fisso
- RESISTENTE ALLA COMPUTAZIONE: per ogni chiave k , date zero o più copie $(x_i, h_k(x_i))$, dev'essere computazionalmente impossibile determinare $(x, h_k(x))$ per ogni possibile ingresso $x \neq x_i$, compreso il caso in cui $h_k(x) = h_k(x_i)$ per qualche i .

→ Se vale allora non c'è modo di trovare la chiave

La definizione di MAC non dice niente a proposito della resistenza a PREIMAGE e a 2ND-PREIMAGE per le parti che conoscono la chiave k . Per un avversario che NON conosce la chiave k :

- h_k dev'essere resistente a 2ND-PREIMAGE e a COLLISION
- h_k dev'essere anche PREIMAGE RESISTANT per evitare attacchi chosen-text

L'avversario ha come obiettivo quello di determinare una nuova coppia testo-MAC $(x, h_k(x))$ senza conoscere k , per qualche $x \neq x_i$, date una o più copie del tipo $(x_i, h_k(x_i))$

CONTRAFFAzione DEL MAC: permette ad un avversario di ottenere testo contraffatto accettato come autentico

- **CONTRAFFAzione SOTTIVA:** l'avversario può produrre testo e MAC a sua scelta
- **CONTRAFFAzione ESISTENZIALE:** l'avversario può produrre copie testo-MAC, ma senza il controllo sul testo

Il recupero della chiave permette entrambe le contraffazioni (*forgery*)

per un avversario che non conosce la chiave k , h_k è resistente, esposto ad attacchi di tipo chosen-text, a:

- **2ND-PREIMAGE:** la resistenza alla computazione (proprietà del MAC) implica che se MAC non può essere determinato senza conoscere k .
- **PREIMAGE:** per assurdo, il recupero da una preimage x (~~input~~) di un'uscita hash generica y viola la resistenza alla computazione

In assenza di **BEST KNOWN ALG**, un avversario può:

- individuare la chiave : 2^t tentativi (t bit della chiave)
- individuare il MAC di un messaggio : 2^m tentativi (m bit di uscita)

IMPLEMENTAZIONI PER LA SICUREZZA:

- MAC basato su cifrario a blocchi (CBC)
- MAC basato su MDC:
 - a panino : $h_k(x) = h(k \parallel \text{pad.} \parallel x \parallel k)$
 - hash-based : $h_k(x) = h(k \parallel \text{pad}_1 \parallel h(k \parallel \text{pad}_2 \parallel x))$
- customized MAC (HMAC, MD5-MAC, ...)
- MAC for stream ciphers

INTEGRITÀ DEI DATI:

- usando solo il MAC : $x \parallel h_k(x)$
- usando MDC su canale sicuro (autenticato) : il ~~messaggio~~ messaggio viene inviato sul canale sicuro, mentre l'MDC viene trasmesso su un canale autenticato (telefono, giornale, ...)
- usando la CRITTOGRAFIA: la crittografia sola non garantisce l'integrità dei dati, quindi si unisce con:
 - riordinamento dei blocchi ECB
 - cifratura dei dati random
 - ...

⇒ **CRITTOGRAFIA +**:
1) meccanismi per garantire l'integrità (ECB block reord., ...)
2) MDC : $C = E_k(x \parallel h_k(x))$, proprietà di sicurezza di $h(x)$ più debole rispetto a quelle delle firme digitali
⇒ la sicurezza dipende dal *avversario*!!!
• $(x, E_k(h(x)))$: h deve essere COLLISION RESISTANT, senz'essere determinata senza conoscere k
• $E_k(x) \parallel h(x)$: risparmio computazionale rispetto a cifrare $x \parallel h(x)$; h deve essere COLLISION RESISTANT
3) MAC : $C = E_{k_1}(x \parallel h_{k_2}(x))$. E può essere scrittito, h garantisce l'integrità; E impedisce una ricerca esauriente della chiave k_2 di h . CONTRO: ci sono 2 chiavi.
• $E_{k_1}(x), h_{k_2}(E_{k_1}(x))$: permette l'autenticazione senza la conoscenza del testo del messaggio
• $E_{k_1}(x), h_{k_2}(x)$

I meccanismi basati su chiavi condivise non garantiscono la NON RIPUDIABILITÀ.
Nonostante le MAC (e la firma digitale) garantiscono l'AUTENTICAZIONE DELL'ORIGINÉ, non forniscono l'integrità (dei risultati) o garanzie inerenti la tempestività.
Per poterle ottenere, occorre potenziare tali meccanismi di origine dei dati di timestamp, numeri di sequenza, numeri random, ...

FIRMA DIGITALE: numero dipendente da un segreto noto solo al firmatario e dal contenuto del messaggio.

Dev'essere:
• **VERIFICABILE**: in caso di disputa, una terza parte dev'essere in grado di risolvere la controversia senza richiedere il segreto al firmatario.

Classificazione:
- **CON APPENDICE**: richiede in ingresso il messaggio per la verifica della firma; usa una funz. hash

- **CON RECUPERO DEL MESSAGGIO**: il messaggio originale viene recuperato direttamente dalla firma (es. RSA)

FIRMA DIGITALE CON APPENDICE: M : spazio dei messaggi M_h : spazio dell'immagine di h
 h : funzione hash S : spazio delle firme

Generazione delle chiavi:

- Si sceglie una chiave privata che definisce l'algoritmo di firma $S_A : M_h \rightarrow S$
- Si definisce la corrispondente chiave pubblica grazie all'algoritmo di verifica V_A tale che $V_A(m^*, s) = \text{vero}$ se $s = S_A(m^*)$, falso altrimenti,
 $\forall m^* \in M_h$, dove $m^* = h(m)$, $\forall m \in M$. V_A è costruita in modo tale da poterla calcolare senza conoscere la chiave privata del firmatario
- S_A è detta chiave privata; V_A è detta chiave pubblica

Procedura di firma:
1) $m^* = h(m)$
2) $s = S_A(m^*)$
3) invio di (m, s)

Procedura di verifica:
1) ricezione di (m, s)
2) $m^* = h(m)$
3) $v = V_A(m^*, s)$
4) v è il risultato

Proprietà di V_A e S_A :
- S_A efficiente da determinare
- V_A efficiente da determinare
- dev'essere praticamente impossibile trovare m e $s \in S$ tali che $V_A(m^*, s) = \text{true}$, $m^* = h(m)$ per un'altra entità diversa dal firmatario originale

FIRMA DIGITALE CON RECUPERO DEL MESSAGGIO: M : spazio dei messaggi
 M_s : spazio delle firme
 S : spazio delle forme

Generazione delle chiavi:

- si sceglie una chiave ~~privata~~ che definisce l'algoritmo di firma $S_A : M_S \rightarrow S$
- si definisce la chiave pubblica che definisce l'algoritmo di verifica V_A tale che $V_A \cdot S_A$ si mappi su M_S ; V_A è costruita in modo da poterla compiere senza saper conoscere la chiave privata del firmatario
- S_A è detta chiave privata; V_A è detta chiave pubblica

Processo di firma:

- 1) $m^* = R(m)$ funzione di ridondanza (invertibile)
- 2) $s = S_A(m^*)$
- 3) invio di s

Processo di verifica:

- 1) ricezione di s
- 2) $m^* = V_A(s)$
- 3) se $m^* \in M_S$ la firma è valida
- 4) recupera $m = R^{-1}(m^*)$

Proprietà di V_A e S_A :

- S_A efficiente da calcolare
- V_A efficiente da calcolare
- dev'essere praticamente impossibile trovare $s \in S$ tale che $V_A(s) \in M_S$ per un'entità diversa dal firmatario

Funzione di ridondanza:

- R e R^{-1} sono pubblici e conosciuti
- la scelta di R è critica per la sicurezza del sistema

→ cattiva scelta di R : $M_R \neq M_S$

R e S_A sono bivette, quindi M e S hanno lo stesso numero di elementi, perciò $\forall s \in S, V_A(s) \in M_R$. Si ottiene dunque che è "semplice" trovare un m per il quale s è la firma, $m = R^{-1}(V_A(s))$; s è una firma valida per m (contraffazione estensiva)

→ buona scelta di R : $R(m) = m \parallel m$

Quando n è grande (m è la cardinalità di m), $|M_R|/|M_S| = (1/2)^n$ è piccolo, perciò non è facile per un avversario trovare un s che determini $V_A(s) \in M_R$.

FIRMA DIGITALE CON APPENDICE DA RECUPERO DEL MESSAGGIO:

Generazione della firma:

- $m^* = R(h(m))$, $s = S_A(m^*)$
- invio di (m, s)

Verifica della firma:

- ricezione di (m, s)
- $m^* = R(h(m))$, $m' = V_A(s)$, $u = (m' == m^*)$
- la firma è valida se $u = \text{true}$

In questo caso, R non è critico per la sicurezza del sistema.

ROTTURA DELLA FIRMA:

- ROTURA TOTAL: l'avversario è in grado di determinare la chiave privata del firmatario
- CONTRAFFAzione SOVETTIVA: l'avversario controlla il messaggio la cui firma è stata contraffatta
- CONTRAFFAzione ESISTENZIALE: l'avversario non ha il controllo sul messaggio la cui firma è contraffatta

TIPI DI ATTACCO: • KEY ONLY ATTACKS: l'avversario conosce solo la chiave pubblica del firmatario

• MESSAGE ATTACKS:

- KNOWN-MESSAGE ATTACK: l'avversario ha le firme di un insieme di messaggi conosciuti, ma non scelti da lui
- CHOSEN-MESSAGE ATTACK: l'avversario ha scelto anche il messaggio
- ADAPTIVE CHOSEN-MESSAGE ATTACK: l'avversario sceglie il messaggio adattandolo ai vari contesti per migliorare l'efficacia
È il più difficile da prevenire, anche se in pratica può essere impossibile da ottenerlo, monché occorre progettare uno schema che eviti la possibilità di farlo.

Il livello di sicurezza può variare con il tipo di applicazione.

Quando la funzione hash viene usata nella firma digitale (come spesso accade) questa dev'essere una parte fissa del processo di firma, in modo da evitare che un'avversario possa riempirla con una più debole e mettere su un attacco di contraffazione selettiva.

FIRMA DIGITALE BASATA SU RSA: essendo la crittografia un processo reversibile, la firma digitale può essere creata da questa invertendo i ruoli di cifratura e decifratura. Nella firma digitale con recupero del messaggio si riporta $M_S \equiv S \equiv \mathbb{Z}_m = [0, m-1]$, con funzione di ridondanza $R: M \rightarrow \mathbb{Z}_m$ scelta e nota pubblicamente

- Generazione delle chiavi:
- 1) p, q numeri primi diversi e molto grandi
 - 2) $m = p \times q$, $\phi = (p-1) \times (q-1)$
 - 3) scelta di un numero random $1 < e < \phi$ tale che $\text{gcd}(e, \phi) = 1$
 - 4) calcolo dell'intero UNICO $1 < d < \phi$ tale che $ed \equiv 1 \pmod{\phi}$
 - 5) (d, m) è la chiave privata
 - 6) (e, m) è la chiave pubblica
 - 7) p e q devono essere distrutti!

- Processo di firma:
- 1) $m^* = R(m)$, come un intero in $[0, m-1]$
 - 2) $S = m^{*d} \pmod{m}$
 - 3) invio di S

- Processo di verifica:
- 1) ricezione di S
 - 2) $m^* = S^e \pmod{m}$
 - 3) la firma è valida se $m^* \in M_S$
 - 4) il messaggio si ottiene da $m = R^{-1}(m^*)$

Dimostrazione della validità: se S è la firma per il messaggio m , allora vale che $S = m^{*d} \pmod{m}$, con $m^* = R(m)$. Essendo $ed \equiv 1 \pmod{\phi}$ allora vale $S^e = (m^{*d})^e \equiv m^{*(ed)} = m^* \pmod{n}$, perciò $R^{-1}(m^*) = R^{-1}(R(m)) = m$!

- Possibili attacchi:
- **RATTOZZAZIONE INTESA:** porta alla rotura totale dell'algoritmo, p e q devono essere scelti in modo che sia impossibile attuarla (praticamente)
 - **PROPRIETÀ MONTIPLICATIVA:** R non deve soddisfare tale proprietà affinché non sia possibile la contraffazione selettiva

AUTENTICAZIONE & NON RIPUBBLICO:

- **AUTENTICAZIONE** (basata sulla crittografia simmetrica) permette le parti di convincersi, o di convincere una terza entità, sull'integrità e l'autenticità di un messaggio dato dal tempo t_0
 - **NON RIPUBBLICO** (basato sulla crittografia a chiave pubblica) permette le parti di far convincere gli altri dell'integrità e dell'autenticità di un messaggio al tempo t_0 , dal tempo $t_0 > t_0$.
- ~~→ evita che una persona che firma un documento possa successivamente negare di averlo fatto~~

L'autenticazione dell'origine tramite firma digitale è valida solo finché il segreto privato del firmatario è mantenuto come tale

CERTIFICATI

- PROBLEMI: rendere la chiave pubblica di un soggetto disponibile ad altri per verificarne l'autenticità e la validità
- ROTOSI: ogni soggetto può essere identificato tramite un identificatore
- AUTORITÀ DI CERTIFICAZIONE: è una TTP che attesta l'autenticità delle chiavi
- CERTIFICATO: (public-key certificate): è una struttura dati che lega l'identificatore dell'utente con la sua chiave pubblica ed è formato dall'autorità di certificazione

CREAZIONE DEL CERTIFICATO: dopo aver verificato l'identità della parte e la sua chiave pubblica, l'autorità genera un certificato $C(T, A) = e_A, A, L, S(e_A, A, L)$ con il periodo di validità.

- da verifica dell'identità del soggetto attraverso per metodi non crittografici
- da verifica dell'autenticità della chiave può avvenire tramite challenge-response oppure fornendo la coppia di chiavi alla CA stessa
- Altre informazioni nel certificato possono essere: info agg sul soggetto, sulla firma, sui metodi di identificazione, di generazione, di verifica ...

MODULO CON SINCROLA CA: controlla un sottosistema detto SECURITY DOMAIN e mantiene i certificati memorizzati all'interno di una CERTIFICATE DIRECTORY in sola lettura e gestito da una terza parte (NSICUCA)

USO DI VERIFICA:

- 1) ottieni la chiave pubblica del Trento (e_T)
- 2) ottieni l'identificatore univoco del mittente A
- 3) ottieni il certificato $C(T, A)$
- 4) verifica che:
 - la chiave di Trent sia valida
 - il certificato $C(T, A)$ sia ancora valido
 - la firma su $C(T, A)$ sia corretta usando ep
 - il certificato $C(T, A)$ non sia stato revocato
- 5) Se le verifiche sono positive, allora es è autentico

⇒ DIRETTO DA FLORIAN (metodo sul quale si basa la certificazione): ogni soggetto che utilizza un certificato delega alla CA la fiducia nella verifica dell'identità di un altro soggetto e dell'autenticità della chiave.

~~→~~ Occorre che ci sia fiducia anche sulla chiave pubblica e_T .

⇒ CERTIFICATO DEGLI ATTRIBUTI: permette di legare una chiave a delle informazioni generalmente sull'utilizzo della chiave stessa: $AC(R, A) = e_A, \alpha, S_R(\alpha, e_A), C(T, A)$ con α attributi. Può servire a:

- assegnare autorizzazioni sull'utilizzo della chiave
- visualizzare l'uso della firma digitale

REVOCA DELL'UTENTE: quando scade il periodo di validità della chiave, scade anche il certificato (EXPIRED CERTIFICATE)

se per qualunque motivo una chiave diventa invalida prima della scadenza, allora il certificato dev'essere revocato, ad es:

- chiave compromessa
- utente non me necessita più
- ...

la revoca dev'essere:
- CORRETTA: richiesta solo da chi è autorizzato, cioè il possessore o l'emittitore (subject or issuer)
- TEMPESTIVA: la revoca deve essere raggiunta da tutti gli interessati il prima possibile

La gestione può avvenire tramite:

- DATA DI SCADENZA NEI CERTIFICATI: si limita l'esposizione quando compromessa
- NOTIFICA MANUALE: utenti avvisati tramite canali speciali, NON SCALABILE!
- FILE PUBBLICO DI REVOCA DELL'UTENTE: contiene le chiavi revocate e va controllato prima dell'utilizzo della chiave
 - CERTIFICATE REVOCATION LIST (CRL): metodo con file pubblico di chiavi revocate
- REVOCATION CERTIFICATES: alternativa al CRL, è un certificato con revocation flag attivo e sostituisce quella nella directory del certificato originale

CERTIFICATE REVOCATION LIST: lista di certificati revocati che non vengono riconosciuti se il certificato non ha subito modifiche. In modalità push, viene pubblicata ad intervalli regolari per evitare replay attack (firmare + data pubb.)

Quando diventa troppo grande si può trasmettere a pezzi:

- delta-CRL: l'utente finale ha mantenuto in modo sicuro
- partitionamento in base al motivo di ricerca
- segmentazione in pezzi a cui viene assegnato un certificato; ogni certificato specifica i segmenti a cui è stato pre-assegnato

I CRL permettono la verifica offline dei certificati.

Svantaggi:
• un avversario utilizza una chiave finta alla distribuzione della nuova CRL
• I CRL sono spesso l'ultima componente realizzata

- i vecchi browser non implementavano l'accesso
- inizialmente Microsoft fece causa a Verisign perché non aveva il puntatore alla CRL nei certificati

APPROCCI ALTERNATIVI ALLA CRL:

- controllo on-line dei certificati: accurato, ma poco affidabile
- Timeline certification (short-term certificate): certificati con marca temporale in cui è l'utente a specificare quanto tempo dev'essere il certificato $S_{\text{ca}}(C, t)$, nel intervallo di validità

INFRASTRUTTURE A CHIAVE PUBBLICA: affinché entità appartenenti a domini di sicurezza differenti possano interagire è necessario che sia possibile stabilire una relazione di fiducia fra le rispettive CA del dominio (TRUST RELATIONSHIP)

• MODELLO DI TRUST CENTRALIZZATO: albero in cui ogni entità conosce a priori la chiave pubblica della radice. La CATENA DI CERTIFICATI definisce la catena di fiducia che ponte dalla CA fidata fino a quella della quale si vuole ottenere fiducia, seguendo il CAMMINO DI CERTIFICAZIONE.

- Unico dominio di sicurezza
- Trust affidato alla chiave pubblica della radice
- Una catena è necessaria anche per entità sotto la stessa CA
- Le catene possono essere anche notevolmente lunghe
- Nel modello più naturale, il trust inizia dal CA padre piuttosto che dalla CA radice

- **MODELLO A RADICI MULTIPLEX**: i certificati incrociati (cross-certificates) permettono ad un'entità di ottenere fiducia nei certificati rilasciati da una CA esterna al proprio dominio di sicurezza
- **MODELLO CON CERTIFICATI INVERSI**: ogni CA crea un certificato inverso (reverse certificate) per la CA padre, ed un certificato diretto (forward certificate) per la CA figlia; in questo modo ogni entità conosce a priori la chiave pubblica della CA che ha creato il suo certificato.
 - catene "lunghe" anche tra CA che comunicano frequentemente
 - si può migliorare con la cross-certification
- **MODELLO DISTRIBUITO**: ogni CA può cross-certificare ogni altra CA, ogni CA può certificare gli utenti finali; inoltre, ogni entità conosce a priori la chiave pubblica della propria CA locale
 - se CA_x cross-certifica CA_y, allora CA_x ripone fiducia in CA_y e in tutte le CA raggiungibili da CA_y
 - CA_x può limitare la fiducia imponendo dei vincoli sui certificati:
 - sulla lunghezza massima della catena
 - sull'insieme dei domini validi

RANDOM E PSEUDO-RANDOM BIT GENERATORS: la sicurezza di molti sistemi crittografici dipende dalla generazione di quantità imprevedibili, che devono essere sufficientemente grandi e casuali, tali da ottenere una probabilità per un avversario di scegliere tale quantità sufficientemente bassa da evitare di montare un attacco basato su tale probabilità.

- **RANDOM BIT GENERATORS (RBG)** necessitano di una "sorgente di casualità" per emettere una sequenza di bit indipendenti (la probabilità che un bit sia 0 o 1 non dipende dai bit precedenti) ed equiprobabili (probabilità che esca 0 o 1 è 0,5).
 - **RBG BASATI SU HARDWARE**: cercano la casualità nei fenomeni fisici, come il rumore termico dei semiconduttori, la frequenza instabile degli oscillatori a cristallo, il suono da un microfono, ...
 - **RBG BASATI SU SOFTWARE**: creati via software, come il clock del sistema, il tempo tra la pressione di due tasti o il movimento del mouse, il contenuto dei buffer di ingresso o di uscita, ... un buon RBG basato su software tiene più fonti differenti quanto disponibili.

Ovviamente non devono essere soggetti ad osservazioni o modifiche da parte di avversari, anche perché sono influenzati da fattori esterni o malfunzionamenti.

- **PSEUDO-RANDOM BIT GENERATORS (PRBG)**: dal momento in cui la generazione di bit random è inaffidabile nella maggior parte dei sistemi pratici, si cerca di usare algoritmi deterministici (PRBG) con ingressi casuali, in modo che l'avversario non possa distinguere tra bit realmente randomi e la sequenza di uscita del PRBG
 - k (bit di ingresso) dev'essere sufficientemente grande da evitare una ricerca esauriva di k ingressi
 - un PRBG passa tutti i test statistici di tempo polinomiale se non esiste alcun algoritmo di tempo polinomiale che riesce a distinguere l'uscita da una quantità realmente randomica con probabilità < 0,5
 - un PRBG passa il test del next-bit se non esistono algoritmi polinomiali che, dati n ingressi i primi l-bit della sequenza di uscita, riescano a determinare il (l+1)-esimo bit con probabilità maggiore di 0,5

→ UN PRBG CHE PASSA QUESTI TEST È CRIPTOGRAFICAMENTE SICURO

• AD-HOC PRBG:

- si usa una OWF come algoritmo deterministico, tipo un hash o la crittografia



Anche se non è stato dimostrato che mani crittograficamente sicure, sembrano che sia sufficiente per la maggior parte delle applicazioni pratiche.

- ANSI X9.17: usato per generare chiavi pseudorandom e vettori di initializzazione in DES
- FIPS 186: standard DSA per la firma digitale e usato per generare la chiave privata in DSA e il segreto per ogni messaggio, sempre in DSA

• CRIPTOCRATICALLY SECURE PRBG (CSPRNG): la sicurezza dei CSPRNG si basa sulla presunta intrattabilità di problemi numerici:

- RSA-PRBG è sicuro sotto l'ipotesi che RSA PROBLEM è intrattabile
- BWM-BLUM-STUB-PRBG è sicuro sotto l'ipotesi che la fattorizzazione intera è intrattabile
- Questi CSPRNG fanno uso della moltiplicazione modulare, il che è sensibilmente più lento dei PRBG ad-hoc.

802.11 DATA LINK PROTOCOLS: protocollo di livello data-link (che ingloba anche quello fisico) in cui occorre garantire la COLLISION AVOIDANCE e la SICUREZZA.

TIPOLOGIE DI RETE: • **MODALITÀ AD-HOC:** Insieme di Servizi di Base Indipendenti (BSS), i client comunicano direttamente con i client all'interno delle loro celle, ognuno lavora da gateway per il routing.

• **MODALITÀ INFRASTRUCTURE:** ogni station invia le comunicazioni ad un access point che funge da Bridge Ethernet. Prima di comunicare devono definire l'associazione:

- l'AP invia ad intervalli regolari il SSID (beacon)
- il client sceglie il set di Servizi di base (BSS)
- AP e client si autenticano mutualmente
- il client richiede di stabilire una connessione

WIRED EQUIVALENT PRIVACY (WEP):

è un protocollo standard a livello data-link in grado di aumentare la confidenzialità (obiettivo principale), l'autenticazione e l'integrità (obiettivi secondari), usando un cifrario RC4.

$$k \rightarrow \boxed{\text{KSG}} \xrightarrow{\oplus} c_i = p_i \oplus m; \quad \text{ENCRYPTION: } C = P \oplus \text{KSG}(k) \quad \text{KSG: key sequence generator}$$

$$k \rightarrow \boxed{\text{KSG}} \xrightarrow{\oplus} p_i = z_i \oplus c_i; \quad \text{DECRYPTION: } P = C \oplus \text{KSG}(k)$$

d'access point condivide la chiave k con tutti i nodi mobili.

- Il messaggio viene preso, calcolato il CRC e allegato, poi fatto lo XOR con il key stream uscente da RC4(k, v), in cui v è un vettore d'initializzazione nato che viene allegato in testa al messaggio cifrato: $v \parallel \text{RC4}(k, v) \oplus (m \parallel \text{CRC}(m))$.
- In ricezione, si fa lo XOR tra il messaggio ricevuto (nella v) e $\text{RC4}(k, v)$ e poi si controlla che $\text{CRC}(m)$ corrisponda con quelli ricevuti.

Il vettore d'initializzazione è fisso di 24 bit, mentre l'implementazione può essere STANDARD a 64 bit o ESTESA a 128 bit. La distribuzione della chiave non è specificata e si appoggia su meccanismi esterni, così come la gestione delle chiavi, che nello standard è unica per tutti i nodi.

IMPLEMENTAZIONE STANDARD: 64 bit = 40 bit (k) + 24 bit (v)

IMPLEMENTAZIONE ESTESA: 128 bit = 104 bit (k) + 24 bit (v)

KEYSTREAM REUSE ATTACK: cifrare due messaggi con lo stesso keystream può rivelare informazioni su entrambi.

$$\begin{cases} C_1 = P_1 \oplus \text{RC4}(K, v) \\ C_2 = P_2 \oplus \text{RC4}(K, v) \end{cases} \quad \text{allora:}$$

$$C_1 \oplus C_2 = P_1 \oplus \text{RC4}(K, v) \oplus P_2 \oplus \text{RC4}(K, v) = P_1 \oplus P_2$$

se P_1 è noto allora $P_2 = C_1 \oplus C_2 \oplus P_1$
 $\text{e } \text{RC4}(K, v) = C_1 \oplus P_1$

Se parole reali hanno abbastanza ridondanza da poter determinare P_1 e P_2 avendo solo $P_1 \oplus P_2$. L'attacco diventa più efficiente se si hanno diversi testi cifrati provenienti dello stesso keystream.

SOLUZIONI: = vettore d'inizializzazione diverso per ogni pacchetto

→ il riuso del vettore di inizializzazione può causare la probabilità di un keystream reuse attack

- ALTRÒ:
 - WEP raccomanda di cambiare IV per ogni pacchetto, ma non lo rispetta
 - WEP non dice niente su come scegliere IV
 - WEP dice solo che IV dev'essere di 24 bit, e che quasi garantisce che lo stesso IV verrà utilizzato per diversi messaggi

Trovare uno dei messaggi P_1 o P_2 non è complicato: un avversario può inviare messaggi dall'esterno alla rete e leggere dall'interno il corrispettivo messaggio cifrato quando l'AP lo comunicherà via wireless. Con il tempo l'avversario può costruire un dizionario $\langle IV, \text{keystream} \rangle$ e attaccare, ma con 40 bit di chiave è più conveniente una ricerca esauritiva della chiave, perché:

⇒ i censori hanno iniziato a usare l'implementazione estera

MESSAGE AUTHENTICATION ATTACK: WEP usa il checksum CRC-32 per assicurarsi che il messaggio non sia stato modificato dopo l'invio, ma non è sufficiente a garantire l'integrità contro avversari malintenzionati, in quanto la vulnerabilità aumenta insieme all'uso di RF.

Proprietà: 1) Il checksum di WEP è una funzione lineare del messaggio rispetto allo XOR, \forall coppia x, y vale $c(x \oplus y) = c(x) \oplus c(y)$

→ MESSAGE MODIFICATION ATTACK: se: $C = \text{RC4}(K, v) \oplus \langle M, c(M) \rangle$, si definisce

$$C' = C \oplus \langle \Delta, c(\Delta) \rangle, \text{ dove } \Delta \text{ è una modifica arbitraria. Allora:}$$

$$C' = \text{RC4}(K, v) \oplus \langle M, c(M) \rangle \oplus \langle \Delta, c(\Delta) \rangle$$

$$= \text{RC4}(K, v) \oplus \langle M \oplus \Delta, c(M \oplus \Delta) \rangle$$

$$= \text{RC4}(K, v) \oplus \langle M \oplus \Delta, c(M \oplus \Delta) \rangle = \text{RC4}(K, v) \oplus \langle M', c(M') \rangle$$

C' è il testo cifrato del nuovo messaggio $M' = M \oplus \Delta$.

2) Il checksum di WEP è una funzione che non necessita della chiave può essere determinata da chiunque conosca il messaggio

3) È possibile usare nuovamente IV senza "allarmare" il destinatario
 Il riuso non risiede che l'avversario blocca la ricezione

→ MESSAGE INJECTION ATTACK: se un avversario si impone il testo cifrato o del testo in chiaro di un pacchetto, allora può:

• recuperare sia keystream che il vettore IV

• creare un nuovo pacchetto con lo stesso IV (prop. 2)

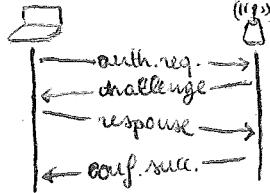
• ripetere l'attacco all'infinito (prop. 3)

Si può evitare disabiliando il riuso del IV oppure usando un MAC

CONTROLLO DEGLI ACCESSI: si può implementare in diversi modi

- OPEN SYSTEM AUTHENTICATION: chiunque può entrare nella rete senza autenticazione
- CLOSED NETWORK AUTHENTICATION: solo i clienti che conoscono il nome della rete o l'SSID possono entrare, il beacon non viene mai inviato e l'SSID funge da segreto condiviso. Il problema è che molti pacchetti inviano l'SSID in chiaro in broadcast (SNIFF!)
- ETHERNET MAC ADDRESS ACL (Access Control List): non fa parte di WEP, ma è lo stesso usata lista di MAC address che possono entrare ⇒ si possono sniffare e impostare via SW
- SHARED KEY AUTHENTICATION: solo chi conosce la chiave segreta può entrare nella rete

L'autenticazione a chiave condivisa usa un protocollo CHALLENGE-RESPONSE



Il challenge su 128 bit è una quantità random
Se modo mobile offre il challenge con la chiave ed invia il risultato (response) all'AP
Se l'AP conferma il risultato allora invia la notifica

N'avversario può ottenere la coppia (challenge, response) e recuperare il keystream tramite lo XOR (keystream = challenge ⊕ response) in modo da autenticarsi all'∞.

[AUTHENTICATION SPOOFING]

IP REDIRECTION ATTACK: serve a decriptare un messaggio senza dover conoscere la chiave

- l'avversario usa un modo all'interno della rete per sniffare i pacchetti inviati attraverso l'etere
- modifica il pacchetto per cambiare il destinatario, esterno alla rete, da cui stava controffatto
- = l'AP riceve il pacchetto, lo decripta e lo invia al destinatario desiderato

I problemi sono:

- indovinare l'IP del destinatario (reale)
- modificare l'IP del pacchetto (Message Modification Attack)
- assicurarsi che il checksum sia ancora corretto \Rightarrow DIFFICILE

D: messaggio originale

D': messaggio modificato

X: checksum originale

X': checksum modificato

D_H: bit più significativo di D

D_L: bit meno significativo di D

$$\Rightarrow X' = X + D_H + D_L - D_H - D_L \text{ (complemento a 1)}$$

L'avversario fa cosa sommare non con cosa fare lo XOR!

- Se l'avversario conosce X il problema è banale, calcola X', cambia X in X' e compensa la modifica cambiando qualche altro campo (es. Source Addr.)
- Se l'avversario NON conosce X il problema è calcolare $\Delta = X' \oplus X$ sapendo $\xi = X' - X$.

Creati protocolli sicuri non è facile e richiede competenze acquisite nell'ing. dei protocolli:
Alcuni principi pericolosi per le sicurezza sono:

- privilegiare le performance
- essere permissivi in ciò che si può accettare
- non memorizzare le info sullo stato

Bisognerebbe invece:

- usare le esperienze passate
- sotto porre il protocollo alla comunità

CONTROMIURE: • A BREVE TERMINE:

- WPA è retrocompatibile, basta cambiare firmware e driver
- TKIP introduce nuovi paradigmi, come funzioni hash, cifrari a blocco,...
- le procedure di autenticazione rimane basata!

• A LUNGO TERMINE:

- CCMP (802.11i) è una soluzione crittografica sicura che richiede però nuovo hardware e un nuovo protocollo
- 802.1x è un metodo di autenticazione basato su porte e la distribuzione delle chiavi

LOGICA BAN: POSTULATI:

- Message meaning rule: $\bullet A \in A \leftrightarrow B, A \in \{X\}_K \text{ allora } A \models B \vdash X$
 $\bullet A \models \overset{K}{\leftrightarrow} B, A \in \{X\}_K \text{ allora } A \models B \vdash X$
 $\bullet A \in A \overset{K}{\neq} B, A \in \{X\}_K \text{ allora } A \models B \vdash X$
- Nonce verification rule: $A \models \#(X), A \models B \vdash X \text{ allora } A \models B \models X$
- Jurisdiction rule: $A \models B \models X, A \models B \Rightarrow X \text{ allora } A \models X$

- OBIETTIVI DI UN PROTOCOLLO:
- Key Authentication : $A \models A \overset{K}{\leftrightarrow} B, B \models A \overset{K}{\leftrightarrow} B$
 - Key Confirmation : $A \models B \models A \overset{K}{\leftrightarrow} B, B \models A \overset{K}{\leftrightarrow} A$
 - Key Freshness : $A \models \#(A \overset{K}{\leftrightarrow} B), B \models \#(A \overset{K}{\leftrightarrow} B)$

PROTOCOLLO NEEDHAM-SCHROEDER:

- Protocollo reale:
- M1) $A \rightarrow T : A, B, N_a$
 - M2) $T \rightarrow A : E_{K_a}(N_a, B, K_{ab}, E_{K_b}(K_{ab}, A))$
 - M3) $A \rightarrow B : E_{K_b}(K_{ab}, A)$
 - M4) $B \rightarrow A : E_{K_{ab}}(N_b)$
 - M5) $A \rightarrow B : E_{K_{ab}}(N_b - 1)$

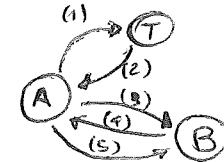
Protocollo idealizzato: M1) ...

- M2) $T \rightarrow A : \{N_a, (A \overset{K_{ab}}{\leftrightarrow} B), \#(A \overset{K_{ab}}{\leftrightarrow} B), \{A \overset{K_{ab}}{\leftrightarrow} B\}_{K_b}\}_{K_a}$
- M3) $A \rightarrow B : \{A \overset{K_{ab}}{\leftrightarrow} B\}_{K_b}$
- M4) $B \rightarrow A : \{N_b, A \overset{K_{ab}}{\leftrightarrow} B\}_{K_{ab}}$
- M5) $A \rightarrow B : \{N_b, A \overset{K_{ab}}{\leftrightarrow} B\}_{K_{ab}}$

Ipotesi: $A \models A \overset{K_a}{\leftrightarrow} T, B \models B \overset{K_a}{\leftrightarrow} T, T \models A \overset{K_a}{\leftrightarrow} T, T \models B \overset{K_b}{\leftrightarrow} T, T \models A \overset{K_{ab}}{\leftrightarrow} B$
 $A \models T \Rightarrow A \overset{K_a}{\leftrightarrow} B, B \models T \Rightarrow A \overset{K_a}{\leftrightarrow} B, A \models T \Rightarrow \#(A \overset{K_a}{\leftrightarrow} B)$
 $A \models \#(N_a), B \models \#(N_b), T \models \#(A \overset{K_a}{\leftrightarrow} B), B \models \#(A \overset{K_a}{\leftrightarrow} B)$

Obiettivi: key authentication e key confirmation

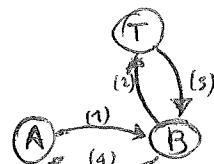
Belief: Dopo M2: $A \models \{N_a, A \overset{K_a}{\leftrightarrow} B, \#(A \overset{K_a}{\leftrightarrow} B), \{A \overset{K_a}{\leftrightarrow} B\}_{K_b}\}_{K_a}, A \models A \overset{K_a}{\leftrightarrow} T \text{ allora } A \models T \vdash (N_a, A \overset{K_a}{\leftrightarrow} B, \dots)$
 $A \models \#(N_a), A \models T \vdash (N_a, A \overset{K_a}{\leftrightarrow} B) \text{ allora } A \models T \models A \overset{K_{ab}}{\leftrightarrow} B \quad (A \models T \models \#(A \overset{K_a}{\leftrightarrow} B))$
 $A \models T \models A \overset{K_a}{\leftrightarrow} B, A \models T \Rightarrow A \overset{K_a}{\leftrightarrow} B \text{ allora } A \models A \overset{K_a}{\leftrightarrow} B \quad (A \models \#(A \overset{K_a}{\leftrightarrow} B))$
Dopo M3: $B \models \{A \overset{K_a}{\leftrightarrow} B\}_{K_b}, B \models B \overset{K_a}{\leftrightarrow} T \text{ allora } B \models T \vdash A \overset{K_a}{\leftrightarrow} B$
 $B \models \#(A \overset{K_a}{\leftrightarrow} B), B \models T \vdash A \overset{K_a}{\leftrightarrow} B \text{ allora } B \models T \models A \overset{K_{ab}}{\leftrightarrow} B$
 $B \models T \models A \overset{K_a}{\leftrightarrow} B, B \models T \Rightarrow A \overset{K_a}{\leftrightarrow} B \text{ allora } B \models A \overset{K_{ab}}{\leftrightarrow} B$
Dopo M4: $A \models \{N_b, A \overset{K_b}{\leftrightarrow} B\}_{K_b}, A \models A \overset{K_b}{\leftrightarrow} B \text{ allora } A \models B \vdash (N_b, A \overset{K_b}{\leftrightarrow} B)$
 $A \models \#(A \overset{K_b}{\leftrightarrow} B), A \models B \vdash A \overset{K_b}{\leftrightarrow} B \text{ allora } A \models B \models A \overset{K_{ab}}{\leftrightarrow} B$
Dopo M5: $B \models \{N_b, A \overset{K_b}{\leftrightarrow} B\}_{K_b}, B \models A \overset{K_b}{\leftrightarrow} B \text{ allora } B \models A \vdash (N_b, A \overset{K_b}{\leftrightarrow} B)$
 $B \models \#(N_b), B \models A \vdash (N_b, A \overset{K_b}{\leftrightarrow} B) \text{ allora } B \models A \models A \overset{K_{ab}}{\leftrightarrow} B$



PROTOCOLLO OTWAY-REES:

- Protocollo reale:
- M1) $A \rightarrow B : M, A, B, E_{K_a}(N_a, M, A, B)$
 - M2) $B \rightarrow T : M, A, B, E_{K_a}(N_a, M, A, B), E_{K_b}(N_b, M, A, B)$
 - M3) $T \rightarrow B : M, E_{K_a}(N_a, K_{ab}), E_{K_b}(N_b, K_{ab})$
 - M4) $B \rightarrow A : M, E_{K_a}(N_a, K_{ab})$

- Protocollo idealizzato:
- M1) $A \rightarrow B : \{N_a, M, A, B\}_{K_a}$
 - M2) $B \rightarrow T : \{N_a, M, A, B\}_{K_a}, \{N_b, M, A, B\}_{K_b}$
 - M3) $T \rightarrow B : \{N_a, A \overset{K_a}{\leftrightarrow} B, B \vdash M\}_{K_a}, \{N_b, A \overset{K_b}{\leftrightarrow} B, A \vdash M\}_{K_b}$
 - M4) $B \rightarrow A : \{N_a, A \overset{K_a}{\leftrightarrow} B, B \vdash M\}_{K_a}$



Ipotesi: $A \xrightarrow{k_a} T, B \xrightarrow{k_b} T, T \xrightarrow{k_a} A, T \xrightarrow{k_b} B, T \xrightarrow{k_{ab}} A \Leftrightarrow B$
 $A \xrightarrow{k_{ab}} T \Rightarrow A \xleftarrow{k_{ab}} B, B \xrightarrow{k_{ab}} T \Rightarrow A \xleftarrow{k_{ab}} B$
 $A \xrightarrow{k_a} T \Rightarrow B \sim M, B \xrightarrow{k_a} T \Rightarrow A \sim M$
 $A \in \#(N_a), B \in \#(N_b), A \in \#(M)$

Obiettivo: key authentication, $A \in B \sim M, B \in A \sim M$

Belief: Dopo M2: $T \notin \{N_a, M, A, B\}_{k_a}, \{N_b, M, A, B\}_{k_b}$
 $T \notin \{N_a, M, A, B\}_{k_a}, T \xleftarrow{k_a} A \text{ allora } T \xrightarrow{k_a} (N_a, M, A, B)$
 $T \notin \{N_b, M, A, B\}_{k_b}, T \xleftarrow{k_b} B \text{ allora } T \xrightarrow{k_b} (N_b, M, A, B)$
Dopo M3: $B \notin \{N_a, A \xleftarrow{k_b} B, B \sim M\}_{k_a}, \{N_b, A \xleftarrow{k_a} B, A \sim M\}_{k_b}$
 $B \notin \{N_a, A \xleftarrow{k_b} B, A \sim M\}_{k_a}, B \xrightarrow{k_b} T \text{ allora } B \xrightarrow{k_b} (N_b, A \xleftarrow{k_b} B, A \sim M)$
 $B \in \#(N_b), B \xrightarrow{k_b} T \xleftarrow{k_a} (N_b, A \xleftarrow{k_b} B, A \sim M) \text{ allora } B \in \#(N_b, A \xleftarrow{k_b} B, A \sim M)$
 $B \xrightarrow{k_b} T \Rightarrow A \xleftarrow{k_b} B, B \xrightarrow{k_b} T \xleftarrow{k_a} B \text{ allora } B \in A \xleftarrow{k_b} B$
 $B \xrightarrow{k_b} T \Rightarrow A \sim M, B \xrightarrow{k_b} T \xleftarrow{k_a} A \sim M \text{ allora } B \in A \sim M$
Dopo M4: $A \notin \{N_a, A \xleftarrow{k_b} B, B \sim M\}_{k_a}$
 $A \notin \{N_a, A \xleftarrow{k_b} B, B \sim M\}_{k_a}, A \xrightarrow{k_a} T \text{ allora } A \in T \sim (N_a, A \xleftarrow{k_b} B, B \sim M)$
 $A \in T \sim (N_a, A \xleftarrow{k_b} B, B \sim M), A \in \#(N_a) \text{ allora } A \in T \sim (N_a, A \xleftarrow{k_b} B, B \sim M)$
 $A \in T \sim (N_a, A \xleftarrow{k_b} B), A \xrightarrow{k_a} T \text{ allora } A \in A \xleftarrow{k_b} B$
 $A \in T \sim B \sim M, A \xrightarrow{k_a} T \text{ allora } A \in B \sim M$
 $A \in B \sim M, A \in \#(M) \text{ allora } A \in B \sim M$

POSTULATO PER LE FUNZIONI HASH: $P \in X \in P \in Q \vdash h(X) \text{ allora } P \in Q \vdash X$
in generale vale: $P \in (X_1, \dots, X_m) \in P \in Q \vdash h(X_1, \dots, X_m) \text{ allora } P \in Q \vdash (X_1, \dots, X_m)$

KERBEROS: diverse infrastrutture:

- SISTEMA CENTRALIZZATO CHIUSO:
 - informazioni ed elaborazioni centralizzate
 - accesso tramite terminali "stupidi"
 - unico dominio amministrativo
 - protezione fisica

Mechanismi per la sicurezza:

- sistema operativo
- meccanismi hardware

- SISTEMA DISTRIBUITO CHIUSO:
 - condivisione di risorse (file, stampanti, CPU, ...)
 - Workstation
 - unico dominio amministrativo
 - utenti noti e fidati

Mechanismi per la sicurezza:

- sistema operativo, meccanismi hardware
- autenticazione basata sull'indirizzo

- SISTEMA DISTRIBUITO APERTO:
 - domini applicativi differenti
 - utenti sconosciuti o inaffidabili
 - la rete è insicura

Mechanismi per la sicurezza:

- sistema operativo, meccanismi hardware
- crittografia

ARCHITETTURA E PROTOCOLLO SEMPLIFICATI: è costituita da un sistema distribuito aperto in cui il server deve consentire solo gli accessi autorizzati e autenticare le richieste di servizio.

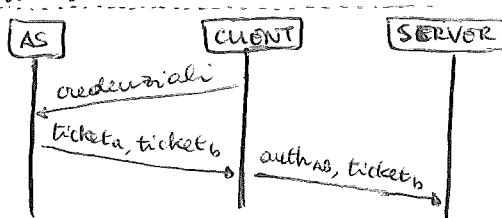
Metodi per l'autenticazione:

- 1) affidarsi alle workstation per identificare gli utenti ed affidarsi al server per autenticarli tramite VID (sist. dist. chiuso)
- 2) affidarsi alle workstation per identificare gli utenti ed affidarsi al server per autenticare le workstation (sist. dist. chiuso)
- 3) richiedere che l'utente fornisca al server una prova della sua identità ogni volta che richiede un servizio; richiedere che anche il server fornisca una prova della sua identità (sist. dist. aperto)

Omettivi:

- **SICUREZZA**: un avversario in ascolto sulla rete non deve acquisire informazioni
- **AFFIDABILITÀ**: senza Kerberos è tutto il sistema, quindi dovrà essere affidabile
- **TRASPARENZA**: l'autenticazione deve essere trasparente per l'utente (escluso password)
- **SCALABILITÀ**: gestione di un numero elevato di client e server.

Protocollo di autenticazione:



KRB_AS_REQ: $C \rightarrow AS : A, B, t, L, Na, WS$

KRB_AS_RESP: $AS \rightarrow C : \{A, B, t, L, Kab, WS\}_{K_A}, \{B, t, L, Na, Kab\}_{K_A}$

KRB_AP_REQ: $C \rightarrow S : \{A, B, t, L, Kab, WS\}_{K_B}, \{A, t_A, \dots\}_{K_AB}$

KRB_AP_RESP: $S \rightarrow C : \{t_A, \dots\}_{K_AB}$



Ticket: contiene la chiave di sessione codificata con la master key (K_A, K_B)

Authenticator: ovviene al server la presenza del client e ricevi-

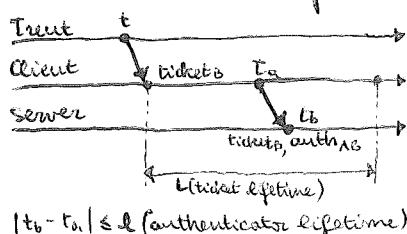
t_A : intervallo di validità del ticket, permette al client di fare più autenticazioni con il server senza passare dall'AS (due messaggi in meno)

t_A : marca temporale generata dal client, in modo che il server verifichi che sia "recente"; ad ogni autenticazione il client genera un nuovo autenticatore con stessa K_B e t_A diversa

WS : identificatore della workstation, permette al server di controllare quali computer possono usare il ticket

$subkey_A, subkey_B$: chiavi da inserire negli autenticatori per l'espletamento del servizio

Ticket e authenticator lifetime:



Kerberos utilizza i timestamp, quindi richiede che i clock siano sincronizzati, perché:

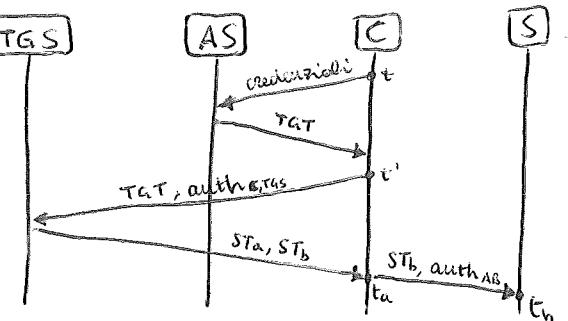
→ clock non sincronizzati e timestamp non cached rendono possibile un REPLAY ATTACK

→ $l = 5$ min, l'autenticatore può essere replicato in questo periodo di vulnerabilità

Se K_A e K_B derivano da password, allora lo schema è tanto sicuro quanto la loro regalanza o la loro resistenza ad un password-guessing attack

ARCHITETTURA COMPLETA: occorre trovare una soluzione per tutti quei client che usano molti servizi, perciò si può pensare di:

- far inserire la password/master key all'utente per ogni nuova autenticazione e rimossa subito dopo (poco usabile)
- memorizzare la password/master key sulla workstation dell'utente per un lungo periodo di tempo (poco sicuro)



KRB_AS_REQ: $C \rightarrow AS: A, TGS, t, L, Na, WS$.
KRB_AS_RESP: $AS \rightarrow C : \{A, TGS, t, L, TK, WS\}_{K_{TGS}}, \{TGS, t, L, TK, WS\}_{K_{AS}}$
KRB_TGS_REQ: $C \rightarrow TGS: \{A, t\}_{K_T}, B, t', L', Na, \{A, TGS, t, L, TK, WS\}_{K_{TGS}}$
KRB_TGS_RESP: $TGS \rightarrow C : \{A, B, t', L', Kab, WS\}_{K_B}, \{B, t', L', Kab, WS, Na\}_{K_{AS}}$
KRB_AP_REQ: $C \rightarrow S : \{A, B, t', L', Kab, WS\}_{K_B}, \{A, ta, ... \}_{K_{AB}}$
KRB_AP_RESP: $S \rightarrow C : \{ta, ... \}_{K_{AB}}$

Periodicità dei messaggi:

• KRB_AS: uno per sessione

• KRB_TGS: uno per ogni servizio

• KRB_AP: uno per richiesta di servizio

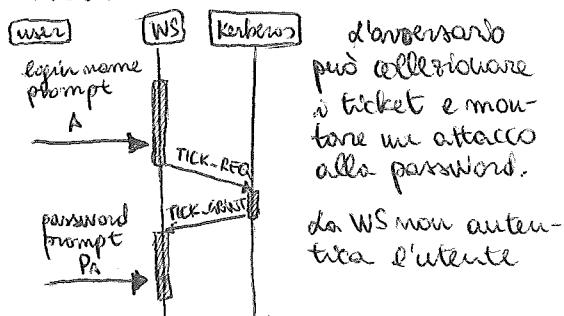
- t : ticket lifetime, tempo di vita del TGT entro il quale si può fare una richiesta al TGS
- L' : ticket lifetime, tempo di vita del service ticket, entro il quale si può fare richiesta di servizi al server S, inviando ST_b (e l'autenticator)
- l : authenticator lifetime, $l_b - t_a$ deve essere minore di l (in kerberos 5 è impostato a 5ms)

Autenticazione degli utenti e delle workstations:

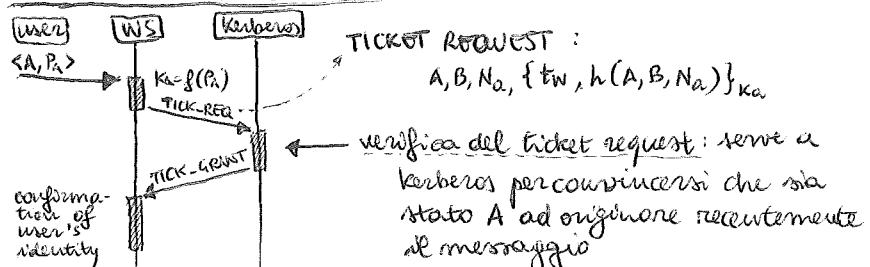
Kerberos autentica gli utenti rispetto ai servizi di rete, ma:

- NON AUTENTICA gli utenti rispetto al KDC, infatti:
 - chiunque può contattare il KDC e chiedere ed ottenere un ticket per il client A (messaggio 1, KRB_AS_REQ)
 - messimo oltre ad client A può utilizzare tale ticket (criptato con K_A)
 - si può sfruttare tale caratteristica per lanciare un password attack (quando K_A dipende dalla password di A)
- NON AUTENTICA gli utenti rispetto alle workstation (indirect authentication)
 - ma WS è il mezzo attraverso il quale gli utenti accedono ai servizi
 - le WS sono uniformi, interchangeable, thus client

Autenticazione tradizionale:



KRB PRE-AUTHENTICATION:



DELEGATION: si considera lo scenario in cui il Mail Server (MS) deve interagire con il File System (FS) per conto dell'utente (A) secondo il principio del minimo privilegio.

Kerberos fornisce due meccanismi che permettono all'utente di delegare il MS per suo conto:



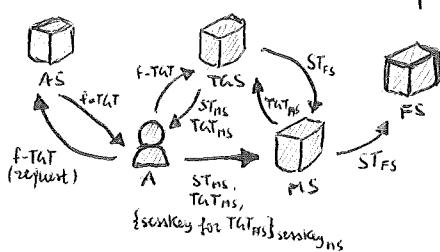
PROXY TICKET: permette al MS di richiedere un servizio per conto di A tramite un ticket che gli ha dato l'utente stesso



$$\begin{aligned} PT_{FS} &= \{K_{A,FS}, N_A, t, L, FS\}_{K_A}, \{K_{A,FS}, MS, t, L\}_{K_{AS}} \\ \{setKey_{FS}\}_{setKey_{MS}} &= \{..., K_{A,FS}, ...\}_{K_{A,MS}} \end{aligned}$$

Per permettere un corretto funzionamento, il client A deve conoscere in anticipo quanti (e quali) proxy ticket deve inviare al MS, oppure sia capace di negoziarli volta per volta. Si può ovviare a questo inconveniente usando un forwardable ticket.

► FORWARDABLE TICKET: si permette al server delegato MS di richiedere i ticket di cui ha bisogno volta per volta.



Quando il cliente A invia al TGS il forwardable-ticket, riceve un service ticket per richiedere il servizio ad MS e un TGT che può usare MS per richiedere servizi al suo posto.

$$ST_{ms} = \{ \dots, K_{ams}, \dots \}_{ka}, \{ \dots, K_{ams}, \dots \}_{K_{ams}}$$

$$TGT_{ms} = \{ \dots, TK_a, \dots \}_{ka}, \{ \dots, TK_a, MS, \dots \}_{KTGS}$$

Il client invierà i due ticket al mail server MS, compresa la ticketing key TK_a cifrata con K_{ams} .

Proxy Ticket

- PRO:** il client conosce quali permessi delegare al server

- CONTRO:** il client deve sapere di quali ticket ci sarà bisogno

Forwardable Ticket

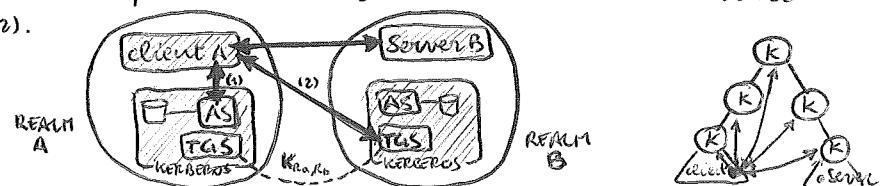
- PRO:** il server determina di quali ticket ha bisogno

- CONTRO:** un server compromesso può abbracciare di tutti i ticket che riceve

Limitazioni alla delega: un ticket ha un tempo di vita massimo (Lifetime) e specifica un insieme ristringito di permessi (capability).

REALMS: Kerberos autentica server e client appartenenti allo stesso dominio amministrativo (realm), quando però appartengono a due realm differenti:

- il client A richiede a kerberos A un REFERRAL TGT per il realm B (1).
- Kerberos A crea un referral TGT usando una inter-realm key $K_{ra,rb}$
- il client A usa il referral TGT per richiedere a kerberos B un service ticket per il server B (2).



I realms sono organizzati gerarchicamente (ad albero), perciò se i due domini non sono collegati direttamente il client A deve richiedere un ticket per ogni nodo successivo (su ogni nello nell'albero) finché non raggiunge il realm del server B. $\Rightarrow O(\log n)$ operazioni.

SECURE SOCKET LAYER (SSL): famiglia di protocolli di livello trasporto

- SESSION**: associazione logica fra client e server creata dal protocollo Handshake che definisce un insieme di parametri crittografici che possono essere condivisi fra molteplici connessioni. Grante ad essa stessa la costanza rinegoziazione dei parametri di sicurezza per ciascuna connessione.

PROTOCOLLO RECORD: incapsula i dati inviati di livelli superiori assicurando la confidenzialità e l'integrità della comunicazione

- **FRAGMENTAZIONE**: in blocchi di al più 2^{14} byte

- **COMPRESSESIONE**: senza perdite e non deve aumentare il blocco di più di 1024 byte

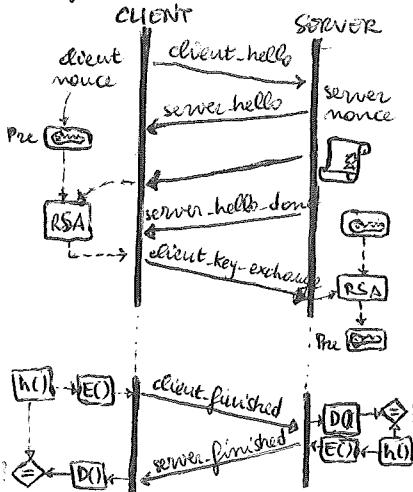
- **AGGIUNTA DEL MAC**: del MAC secret, del numero di sequenza, del padding, ...

- **CIFRATURA**: utilizzando la Server/Client write key, può essere a blocchi o a caratteri, ma non deve far aumentare le dimensioni di un blocco di più di 1024 byte

PROTOCOLLO HANDSHAKE: permette di stabilire una SESSIONE sicura, quindi:

- client e server si autenticano a vicenda
- negoziare la suite di cifratura (cipher suite), ossia:
 - metodo per lo scambio delle chiavi
 - algoritmo di cifratura (da usare nel protocollo Record)
 - algoritmo per il MAC (da usare nel protocollo Record)
- stabilire un segreto condiviso, o MASTER SECRET

Metodo per lo scambio delle chiavi (basato su RSA), autenticazione del solo server, in genere il client rimane anonimo:



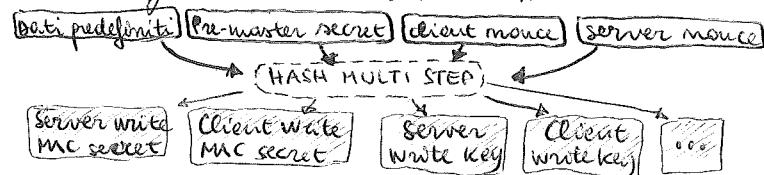
client-hello e server-hello contengono:

- Versione di SSL
- session ID
- cipher suite
- Random: timestamp [32b]
- più randombyte [28b]
- metodo di compressione

client-key-exchange: contiene il segreto condiviso cifrato

con la chiave pubblica del server

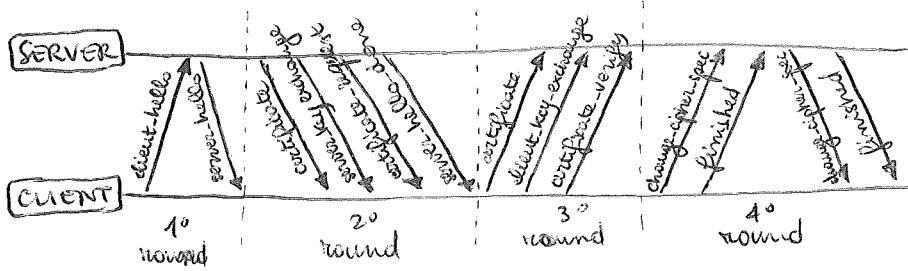
A questo punto vengono generate le chiavi ottenute in ingresso i nonces ed il segreto condiviso ad una HASH MULTI-STEP.



Per essere sicuro di parlare con il server (autenticare il server) il client deve attendere di ricevere un messaggio dal server cifrato con una chiave generata dal master secret.

- AUTENTICAZIONE DEL CLIENT: si rimanda al livello applicativo sfruttando il canale SSL appena instaurato per mettere di username, password, ...
- SICUREZZA DEL CANALE: nei browser meno recenti sono supportate solo chiavi a 40bit invece dei 128 bit di quelli attuali a causa di restrizioni imposte dalla normativa statunitense in materia di esportazione di materiale critografico
- AUTENTICAZIONE DEL CLIENT: il server richiede l'autenticazione del client con un messaggio certificate-request dopo server-hello \Rightarrow CHALLENGE-RESPONSE
- SICUREZZA DEL PROTOCOLO:

- i nonces in client-hello e server-hello servono a generare master secret franchi, quindi evitando attack di tipo replay
- l'utilizzo di certificati protegge dal MAN-IN-THE-MIDDLE
- le sequenze casuali debono essere unpredictibili (pre-master secret e i nonce)
- il protocollo Record numera il blocco, lo autentica tramite il MAC e poi cifrato tutto: così facendo si evitano replay attack, riordino & sostituzione di un blocco, ma se uno non arriva occorre riceverlo tutt'è e spedirlo nuovamente
- nel protocollo Record, il cifrario "protegge" il MAC



Il protocollo CHANGE CIPHER SPEC è costituito da un solo messaggio in chiaro per rendere operativa la suite di cifratura.

Il protocollo ALERT è utilizzato per comunicare al peer gli allarmi relativi a SSL.