I seguenti appunti sono frutto del lavoro di una studentessa, non dei docenti.

Come tali, potrebbero contenere imprecisioni, errori o mancanze in numero arbitrario.

MODULO 2

SLIDE 01

INTRODUZIONE

CHIAVE SIMMETRICA

m = messaggio originale (Plain Text)
c = messaggio cifrario (Cipher Text)
K = chiave segreta condivisa

E = algoritmo di criptazione
D = algoritmo di decriptazione } PUBBLICI !

casi d'uso

(1) single-use key (one-time key) : la chiave simmetrica segreta viene usata per cifrare un solo messaggio.
(2) multiple-use key (many-time key) : la chiave simmetrica segreta viene usata per cifrare più messaggi.

CHIAVE ASIMMETRICA

m = messaggio originale (Plain Text)
c = messaggio cifrario (Cipher Text)
PU = chiave pubblica di Bob
PR = chiave privata di Bob

E = algoritmo di criptazione
D = algoritmo di decriptazione } PUBBLICI !

PROBABILITÀ DISCRETA

DEFINIZIONI

- U : insieme universo
- una **distribuzione di probabilità** su U è $P : U \rightarrow [0,1]$ tale che $\sum_{x \in U} P(x) = 1$
- un **evento** è un sottoinsieme A di U
- la **probabilità di un evento** A è $P[A] = \sum_{x \in A} P(x)$
- una **variabile random** è una funzione $X : U \rightarrow V$
- una **variabile random uniforme** r su S è tale che $\forall a \in S$, $P[r = a] = \frac{1}{|S|}$ si scrive $r \xleftarrow{R} S$

XOR

X	Y	X ⊕ Y
0	0	0
0	1	1
1	0	1
1	1	0

PROPRIETÀ :

sia X una variabile random $\in \{0,1\}^n$ con distribuzione uniforme.
sia Y una variabile random $\in \{0,1\}^n$ con distribuzione arbitraria.
siano X e Y indipendenti.
$\Longrightarrow$ allora $Z = X \oplus Y$ è una variabile random uniforme $\in \{0,1\}^n$.

SLIDE 02

SICUREZZA PERFETTA, OTP, CIFRARI A FLUSSO, PRG, Adv

CIFRARIO SIMMETRICO

UN CIFRARIO SIMMETRICO DEFINITO SU (K, M, C) È UNA COPPIA DI ALGORITMI EFFICIENTI (E, D) DOVE:

- $E : K \times M \to C$
- $D : K \times C \to M$

TALE CHE $\forall m \in M, \forall K \in K : D(K, E(K, m)) = m$

E È SPESSO RANDOMIZZATO.

D È SEMPRE DETERMINISTICO.

CIFRARIO SICURO - SHANNON

SUPPONENDO CHE L'ATTACCANTE ABBIA ACCESSO SOLO AL CT (CIPHERTEXT) = ATTACCO "CT-ONLY ATTACK"

UN CIFRARIO (E, D) DEFINITO SU (K, M, C) È PERFETTAMENTE SICURO SE:
$\forall m_0, m_1 \in M$ CON $len(m_0) = len(m_1)$ E $\forall c \in C$
$$P[E(K, m_0) = c] = P[E(K, m_1) = c]$$

DOVE K È UNIFORME IN K.   SICUREZZA PERFETTA $\Longrightarrow |K| \geq |M|$!

CIFRARIO One-Time Pad (OTP)

$K = M = C = \{0,1\}^n$

$E(K, m) = K \oplus m$

$D(K, c) = K \oplus c$

K VIENE USATA UNA SOLA VOLTA (ONE-TIME KEY) ED È RANDOMICA ($\Rightarrow$ DISTRIBUZIONE UNIFORME SU K)

PRO: VELOCE SIA LA CRIPTAZIONE CHE LA DECRIPTAZIONE.

CONTRO: CHIAVI MOLTO LUNGHE SU MESSAGGI MOLTO LUNGHI (CHIAVE LUNGA COME IL MESSAGGIO).

Teorema: OTP È SICURO ($|K| \geq |M|$)

$\Longrightarrow$ DIFFICILE DA USARE NELLA PRATICA.

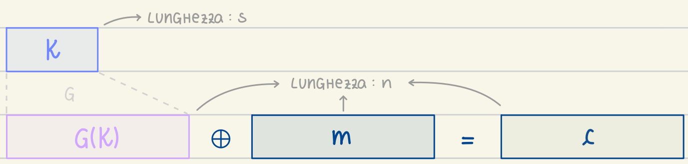
RENDIAMO OTP PRATICO $\Longrightarrow$ CIFRARI A FLUSSO

PER FARE CIÒ SI SOSTITUISCE LA CHIAVE RANDOMICA CON UNA CHIAVE PSEUDORANDOMICA.

Pseudorandom Generator (PRG):

PRG È UNA FUNZIONE $G : \underbrace{\{0,1\}^s}_{\text{spazio dei seed}} \longrightarrow \underbrace{\{0,1\}^n}_{\text{spazio delle chiavi}}$ DETERMINISTICA ED EFFICIENTEMENTE COMPUTABILE CON $n \gg s$ T.C.:

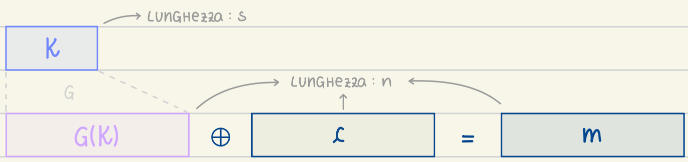
CRIPTAZIONE: $E(K, m) = G(K) \oplus m = c$



OSS:

- K DEVE ESSERE RANDOM
- K DEVE ESSERE USATA UNA SOLA VOLTA

DECRIPTAZIONE: $D(K, c) = G(K) \oplus c = m$



I cifrari a flusso non sono perfettamente sicuri poiché $|K| \ngeq |M|$.
Diamo quindi una nuova definizione di sicurezza che si basa sui PRG.

PRG deboli

- Linear Congruential Generator: parametri a, b (interi), p (primo) $\longrightarrow$

```
r[0] := seed
r[i] ← a r[i-1] + b mod p
output few bits of r[i]
i++
```

- Random $\longrightarrow$

```
glibc random():
    r[i] ← ( r[i-3] + r[i-31] ) % 2^32
    output r[i] >> 1
```

attacchi su cifrari a flusso quindi anche OTP

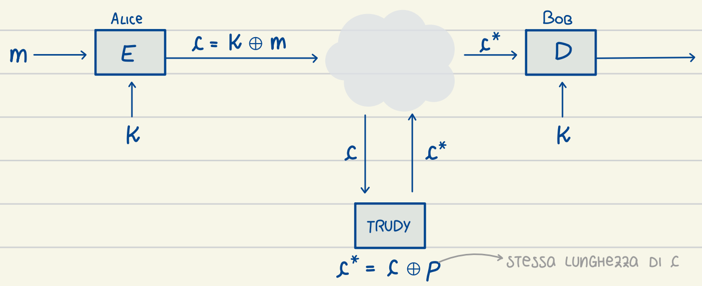
(1) usare la chiave K più di una volta su un cifrario a flusso è insicuro:

$c_1 \longleftarrow m_1 \oplus PRG(K)$ → a parità di K, PRG genera la stessa chiave pseudorandomica

$c_2 \longleftarrow m_2 \oplus PRG(K)$

$$c_1 \oplus c_2 \longrightarrow m_1 \oplus m_2 \oplus \underbrace{PRG(K) \oplus PRG(K)}_{\emptyset} \longrightarrow m_1, m_2$$

(2) integrità non assicurata: OTP è malleabile



$K \oplus c^* =$

$K \oplus c \oplus p =$

$K \oplus K \oplus m \oplus p = m \oplus p$

non è possibile riconoscere eventuali modifiche sul CT. Queste modifiche possono essere fatte in modo tale da modificare prevedibilmente il PL.

sapendo che cosa c'è scritto in m, ad esempio 'Alice', Trudy può modificare m scrivendoci 'Maria' mettendo p = 'Alice ⊕ Maria'.

RC4 (tipo di PRG)

usato per inizializzare l'array S come una pseudo-random permutazione dei numeri $0, \dots, 255$.
L'inizializzazione avviene con il seguente algoritmo con input una stringa di bytes s:

```
for i=0 to 255 do:
   S[i] = i
j =0

for i=0 to 255 do:  ←
   k = s[i % |s|]   //extract one byte from seed
   j = (j + S[i] + k) % 256   ←
   swap(S[i], S[j])  ←
```

i scorre l'array linearmente ←

j si sposta all'interno dell'array secondo la formula $j + S[i] + K$ ←

si scambiano i valori $S[i]$ e $S[j]$ ←

Debolezze:

- anche assumendo che l'algoritmo di inizializzazione sia perfetto, il 2° bit sarà molto spesso in chiaro poiché la probabilità che sia 0 è molto alta.
- attacco "chiavi correlate"

CSS (tipo di PRG)

Content Scrambling System

array chiamato Linear Feedback Shift Register (LFSR)

il seed è lo stato iniziale di LFSR.

obsoleto e molto rotto.

eStream (tipo di PRG)

$PRG : \{0,1\}^s \times R \longrightarrow \{0,1\}^n$ con $n \gg s$

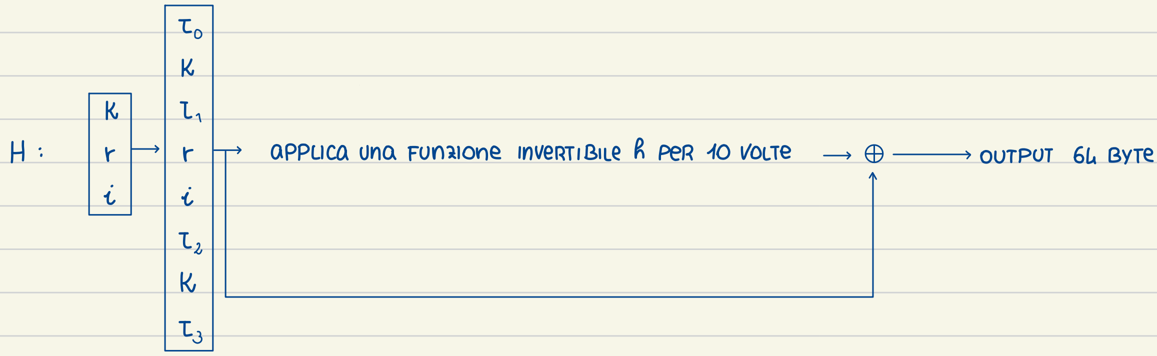
R ↓ nonce : valore non ripetibile per la stessa chiave

$E(K, m, r) = m \oplus PRG(K, r)$

eStream Salsa 20

$Salsa20(K, r) = H(K, (r, 0)) \parallel H(K, (r, 1)) \parallel \ldots$

siano τ_i costanti prefissate



sicurezza dei PRG

un PRG è sicuro quando è impredicibile :

supponendo di avere i primi i bit di $G(K)$ a disposizione, è possibile definire un algoritmo che scopra i restanti.

> $G : K \longrightarrow \{0,1\}^n$ è **predicibile** se :
>
> $\exists$ un algoritmo efficiente A e $\exists\ 1 \leq i \leq n-1$ tale che
>
> $\underset{K \leftarrow K}{P}[A(G(K)|_{1,\ldots,i}) = G(K)|_{i+1}] > \frac{1}{2} + \varepsilon$ per un ε non trascurabile

un PRG è impredicibile se non è predicibile.

test statico

un test statico su $\{0,1\}^n$ è un algoritmo A tale che $A(x)$ ritorna '0' o '1', cioè $A : \{0,1\}^n \longrightarrow \{0,1\}$.

advantage (Adv)

sia $G : K \longrightarrow \{0,1\}^n$ un PRG

sia $A : \{0,1\}^n \longrightarrow \{0,1\}$ un test statico su $\{0,1\}^n$ che testa se lo spazio delle chiavi è indistinguibilmente uniforme : 0 se non lo è, 1 se lo è

$$Adv_{PRG}[A, G] = \underbrace{\left| \underset{K \leftarrow K}{P}[A(G(K)) = 1] - \underset{r \leftarrow \{0,1\}^n}{P}[A(r) = 1] \right|}_{\text{distanza tra la probabilità di una chiave generata e di una effettivamente randomica}} \in [0,1]$$

ritorna 1 se ritiene che l'input sia randomico

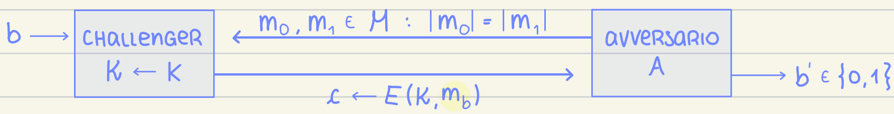
distanza tra la probabilità di una chiave generata e di una effettivamente randomica nuova definizione di sicurezza:

> $G : K \longrightarrow \{0,1\}^n$ è un PRG sicuro se per ogni test statico A efficiente, $Adv_{PRG}[A,G]$ è trascurabile.

sicurezza semantica

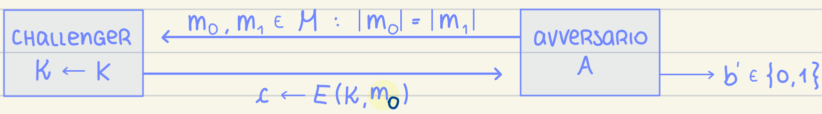
su un cifrario $Q = (E,D)$ e un avversario A, si definisce il seguente gioco:

per $b = 0,1$ definisce due esperimenti $EXP(0)$ e $EXP(1)$ come segue:

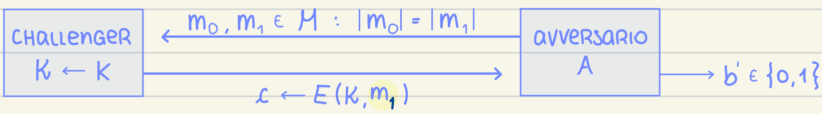


$$Adv_{SS}[A,Q] := \left| P[EXP(0)=1] - P[EXP(1)=1] \right|$$

EXP(0) :



EXP(1) :

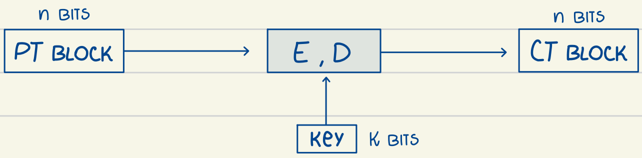


Q è semanticamente sicuro se $Adv_{SS}[A,Q]$ è trascurabile $\forall A$ efficiente.

Teorema: se G è un PRG sicuro, allora il cifrario a flusso Q derivato da G è semanticamente sicuro.

SLIDE 03

CIFRARI A BLOCCHI



esempi di cifrari a blocchi:

- DES $n = 64$ $K = 56$
- 3DES $n = 64$ $K = 168$
- AES $n = 128$ $K = 128, 192, 256$

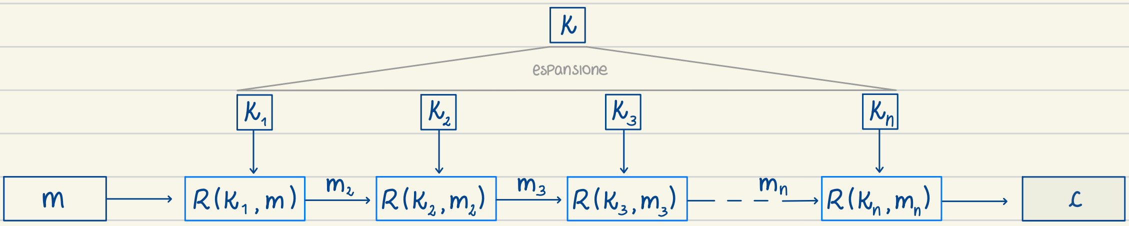
Cifrari a blocchi costruiti su iterazione

La chiave K viene espansa per ottenere n chiavi $K_1, K_2, \dots, K_n$.

Il messaggio m viene dato in input alla prima round function $R(K_1, m)$ che ritorna m_1 che sarà l'input della seconda round function e così via.

Vengono eseguite n round function con input chiave K_i e messaggio m_i : $R(K_i, m_i)$ con $1 \leq i \leq n$

L'output dell'ultima round function sarà il messaggio cifrato c.



Ogni round deve essere facilmente invertibile.

PRF e PRP

PRF : PseudoRandom Function

è una funzione $F : K \times X \longrightarrow Y$ tale che esiste un algoritmo efficiente per calcolare $F(K, x)$.

PRP : PseudoRandom Permutation

è una funzione $E : K \times X \longrightarrow X$ tale che

(1) esiste un algoritmo deterministico efficiente per calcolare $E(K, x)$.

(2) la funzione $E(K, \cdot)$ è iniettiva $(\forall K)$.

(3) esiste un algoritmo efficiente $D(K, y)$ inverso ad E.

OSS: qualsiasi PRP è anche un PRF.

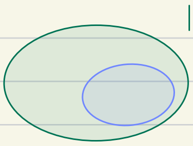
notazioni PRF e PRP

sia $F : K \times X \longrightarrow Y$ un PRF, allora:

- $Funs[X,Y]$ è l'insieme di tutte le funzioni da X a Y
- $S_F = \{ F(K, \cdot) \mid K \in K \} \subseteq Funs[X,Y]$

$|Funs[X,Y]| = |Y|^{|X|}$

$|S_F| = |K|$



sia $E : K \times X \longrightarrow X$ un PRP, allora:

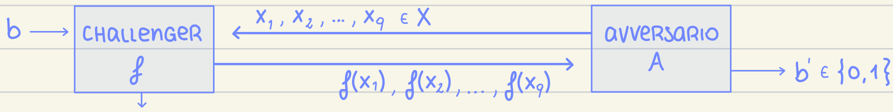
- $Perms[X]$ è l'insieme di tutte le funzioni iniettive da X a X (permutazioni)
- $S_E = \{ E(K, \cdot) \mid K \in K \} \subseteq Perms[X]$

Sicurezza dei PRF

Intuitivamente: un PRF è sicuro se una random function $\in Funs[X,Y]$ è indistinguibile da una $\in S_F$.

Praticamente: sia $b = 0,1$ e definisco l'esperimento $EXP(b)$ come segue:

1) A invia al challenger le x.
2) Sulla base dell'input b, calcola $f(x_i)$ e le invia ad A.
3) Sulla base di ciò che riceve, A deve dire cosa pensa sia b, se 0 o 1.



$b = 0$: $k \leftarrow K$, $f = F(k, \cdot)$

$b = 1$: $f \leftarrow Funs[X,Y]$

F è un PRF sicuro se per ogni avversario efficiente A:

$$Adv_{PRF}[A,F] := |P[EXP(0)=1] - P[EXP(1)=1]|$$

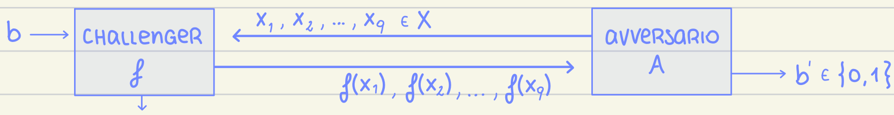
è trascurabile

Sicurezza dei PRP

Intuitivamente: un PRP è sicuro se una random function $\in Perms[X]$ è indistinguibile da una $\in S_E$.

Praticamente: sia $b = 0,1$ e definisco l'esperimento $EXP(b)$ come segue:

1) A invia al challenger le x.
2) Sulla base dell'input b, calcola $f(x_i)$ e le invia ad A.
3) Sulla base di ciò che riceve, A deve dire cosa pensa sia b, se 0 o 1.



$b = 0$: $k \leftarrow K$, $f = E(k, \cdot)$

$b = 1$: $f \leftarrow Perms[X]$

F è un PRP sicuro se per ogni avversario efficiente A:

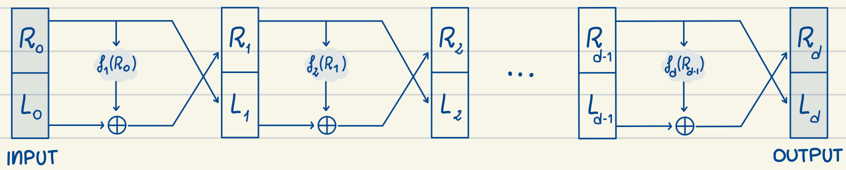
$$Adv_{PRP}[A,F] := |P[EXP(0)=1] - P[EXP(1)=1]|$$

è trascurabile

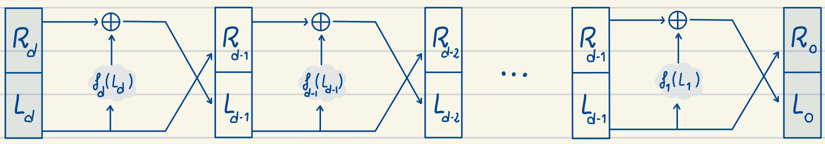
DES

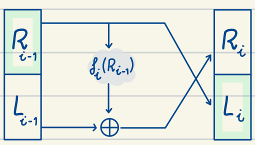
DES: Data Encription Standard

Date le funzioni $f_1, \dots, f_d : \{0,1\}^n \longrightarrow \{0,1\}^n$ non necessariamente invertibili, si vuole creare una funzione invertibile $F: \{0,1\}^{2n} \longrightarrow \{0,1\}^{2n}$



inverso:

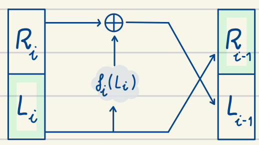




$R_i = f_i(R_{i-1}) \oplus L_{i-1}$

$L_i = R_{i-1}$

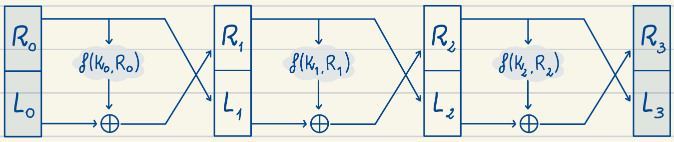
inverso:



$R_{i-1} = L_i$

$L_{i-1} = f_i(L_i) \oplus R_i$

Teorema: sia $f: K \times \{0,1\}^n \longrightarrow \{0,1\}^n$ un PRF sicuro.
allora un DES a 3-round $F: \{0,1\}^{2n} \longrightarrow \{0,1\}^{2n}$ è un PRP sicuro.
siano K_1, K_2, K_3 chiavi indipendenti:



Ricerca esaustiva della chiave di un cifrario a blocchi

date tre coppie input-output $\langle m_i, c_i = E(K, m_i) \rangle$ con $i = 1,2,3$ si vuole trovare la chiave K.

3DES : Data Encription Standard - 3

considera un cifrario a blocchi DES :
- $E: K \times M \longrightarrow M$
- $D: K \times M \longrightarrow M$

un 3DES è definito come $3E: K^3 \times M \longrightarrow M$ ovvero $3E(K_1, K_2, K_3, m) = E(K_1, D(K_2, E(K_3, m)))$

proprietà:
- lunghezza chiave $= 3 \cdot 56 = 168$ bits
- 3 volte più lento di DES
- se $K_1 = K_2 = K_3$ allora si ottiene un DES normale
- buona resistenza agli attacchi

perché 3DES e non 2DES

considero un cifrario a blocchi DES,
definisco $2E(K_1, K_2, m) = E(K_1, E(K_2, m))$.
$2DES(K_1, K_2, m) = E(K_1, E(K_2, m))$.
problema: attacco "meet in the middle" $\longrightarrow$ dati m (PT) e c (CT) si vogliono trovare le chiavi K_1 e K_2
tali che $E(K_1, E(K_2, m)) = c$
$\Rightarrow D(K_1, E(K_1, E(K_2, m))) = D(K_1, c)$
$\Rightarrow E(K_2, m) = D(K_1, c)$

$$\boxed{m} \longrightarrow \boxed{E(K_2, m)} \longrightarrow \boxed{E(K_1, E(K_2, m))} \longrightarrow \boxed{c}$$

meet in the middle

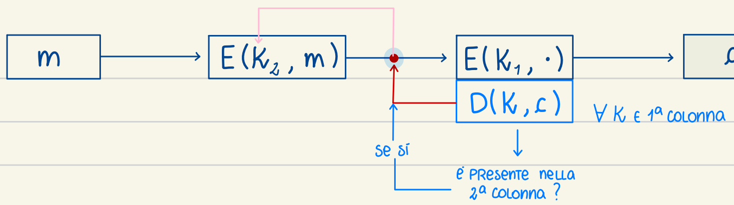
step 1) costruisco una tabella con 2 colonne :

tutte le possibili chiavi di un DES $(0, .., 2^{56} = N)$ — 1ª

1ª	2ª
$K^0 = 00 \dots 00$	$E(K^0, m)$
$K^1 = 00 \dots 01$	$E(K^1, m)$
$K^2 = 00 \dots 10$	$E(K^2, m)$
$\vdots$	$\vdots$
$K^N = 11 \dots 11$	$E(K^N, m)$

2ª — il messaggio criptato con tutte le possibili chiavi di un DES

step 2) controllo per ogni chiave K se $D(K, c)$ è presente nella 2ª colonna;
se sì ho trovato $\longrightarrow K_1$ (la chiave usata in $D(K, c)$)
$\longrightarrow K_2$ (corrispettivo nella 1ª colonna di K_1 appena trovata)



$K^0 = 00\dots00$	$E(K^0, m)$
$K^1 = 00\dots01$	$E(K^1, m)$
$K^2 = 00\dots10$	$E(K^2, m)$
$\vdots$	$\vdots$
$K^i = 11\dots11$	$E(K^i, m)$
$\vdots$	$\vdots$
$K^N = 11\dots11$	$E(K^N, m)$

lo stesso attacco su 3DES richiede moolto più tempo.

DESX

considera un cifrario a blocchi DES:

- $E: K \times M \longrightarrow M$
- $D: K \times M \longrightarrow M$

un DESX è definito come $EX(K_1, K_2, K_3, m) = K_1 \oplus E(K_2, m \oplus K_3)$

attacchi a DES

- attacchi lineari (DES) $\longrightarrow$ richiedono tempo 2^{43}
- attacchi quantistici (DES) $\longrightarrow$ super easy

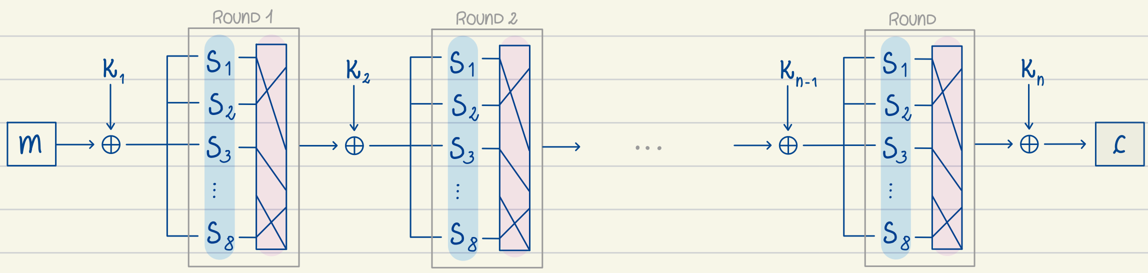
AES

AES: Advanced Encription Standard (PRP)

Dimensione delle chiavi: 128, 192, 256 bit

Dimensione dei blocchi: 128 bit

ogni round è composto da due layer: sostituzione e permutazione



ogni round deve essere singolarmente invertibile per consentire l'invertibilità (non c'è invertibilità globale)

Funzione Round

sostituzione: viene applicata S-box ad ogni byte dell'input m $(m[i,j] = S[m[i,j]]$ per $1 \le i,j \le 4)$

permutazione: ShiftRows + MixColumns

attacchi ad AES

- key recovery attack
- related key attack (quando le chiavi possono essere molto simili)

PRF $\Leftrightarrow$ PRG

PRF $\Rightarrow$ PRG

sia $F : K \times \{0,1\}^n \longrightarrow \{0,1\}^n$ un PRF

t volte n

definisco il PRG $G : K \longrightarrow \{0,1\}^{n \cdot t}$ come segue: $G(K) = F(K, \langle 0 \rangle_n) \,\|\, F(K, \langle 1 \rangle_n) \,\|\, \dots \,\|\, F(K, \langle t-1 \rangle_n)$

valore 0 scritto su n bit — valore 1 scritto su n bit — valore t-1 scritto su n bit

proprietà:

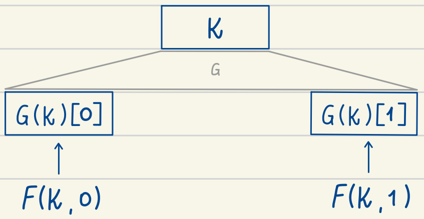
- se F è un PRF sicuro, allora G è un PRG sicuro
- i processi di ciascun blocco sono parallelizzabili

PRF $\Leftarrow$ PRG

gli elementi del codominio sono lunghi il doppio di quelli del dominio

sia $G : K \longrightarrow K^2$ un PRG

definisco il PRP $F : K \times \{0,1\} \longrightarrow K$ come segue: $F(K, x \in \{0,1\}) = G(K)[x]$



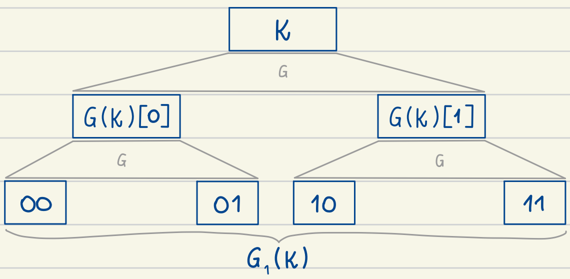
proprietà:

- se G è un PRG sicuro, allora F è un PRF sicuro

per costruire un PRF con dominio maggiore è possibile estendere il PRG ricorsivamente:

sia $G : K \longrightarrow K^2$ un PRG

costruisco il PRG $G_1 : K \longrightarrow K^4$ estendendo G:



$G_1(K) = G(G(K)[0]) \,\|\, G(G(K)[1])$

definisco così un PRF $F : K \times \{0,1\}^2 \longrightarrow K$ come: $F(K, x \in \{0,1\}^2) = G_1(K)[x]$

SLIDE 04

COME USARE UN CIFRARIO A BLOCCHI SU MESSAGGI SPEZZATI IN PIÙ BLOCCHI.

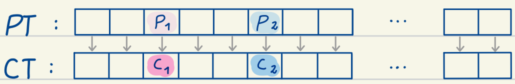
OBIETTIVO: COSTRUIRE UNA CRIPTAZIONE SICURA PARTENDO DA UN PRP SICURO.

one-Time key

one-Time key

L'AVVERSARIO HA A DISPOSIZIONE SOLO IL CT E IL SUO OBIETTIVO É DI SCOPRIRE INFORMAZIONI SUL PT.

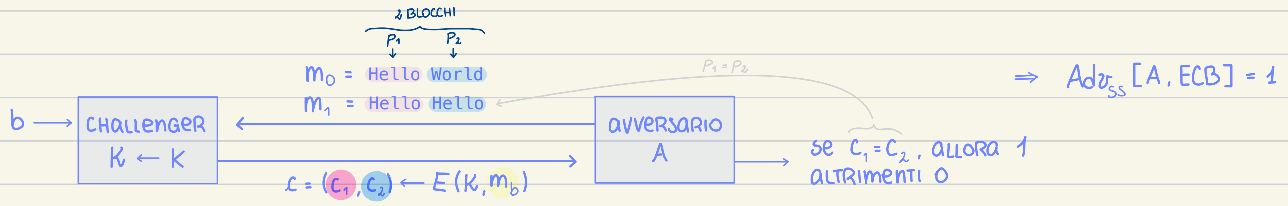
ECB Electronic Code Book



OGNI CARATTERE DEL PT VIENE CONVERTITO NEL CARATTERE CORRISPONDENTE IN CT.

PROBLEMA: SE $P_1 = P_2$ ALLORA $C_1 = C_2$

ECB NON É SEMANTICAMENTE SICURO PER MESSAGGI CHE CONTENGONO PIÙ DI UN BLOCCO:

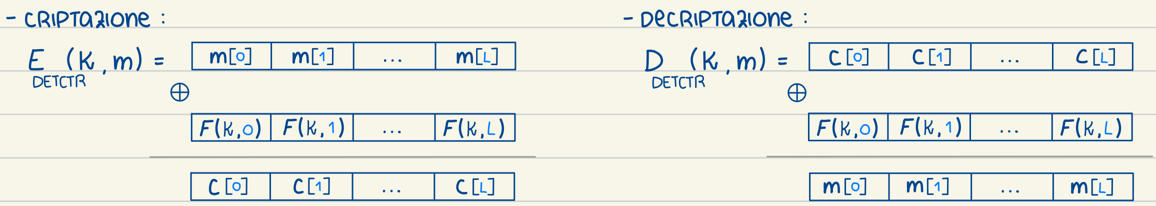


SE DUE BLOCCHI p IN PT SONO UGUALI, ANCHE I DUE BLOCCHI c IN CT SARANNO UGUALI RENDENDOLI DISTINGUIBILI.

DETCTR Deterministic Counter Mode

DATO UN PRF $F: K \times \{0,1\}^n \longrightarrow \{0,1\}^n$

COSTRUISCO UN DETERMINISTIC COUNTER MODE:



OSS: NON É NECESSARIO INVERTIRE F PER DECRIPTARE.

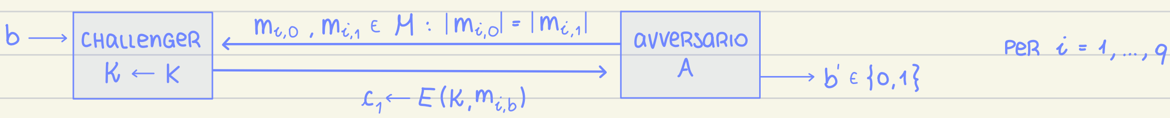
$\forall L > 0$, SE F É UN PRF SICURO SU (K, X, X) ALLORA DETCTR É SEMANTICAMENTE SICURO SU (K, X, X).

many-Time key

many-Time key

L'AVVERSARIO PUÓ OTTENERE IL CT DI MESSAGGI SCELTI DA LUI (CPA: CHOSEN-PLAINTEXT ATTACK), IL SUO OBIETTIVO É DI ROMPERE LA SICUREZZA SEMANTICA.

DEFINISCO $Q = (E, D)$ SU (K, M, C). $EXP(b)$, PER $b = 0, 1$ É DEFINITO COME:



CPA $\longrightarrow$ SE L'AVVERSARIO VUOLE $c = E(K, m')$ ESEGUE LA RICHIESTA AL CHALLENGER CON $m_{j,0} = m_{j,1} = m'$

Q É UN CIFRARIO SEMANTICAMENTE SICURO SE PER OGNI AVVERSARIO EFFICIENTE A:

$$Adv_{CPA}[A, Q] := |P[EXP(0) = 1] - P[EXP(1) = 1]| \text{ é trascurabile.}$$

PROBLEMA: SUPPONGO CHE A PARITÁ DI MESSAGGIO E CHIAVE IL CPA RITORNI SEMPRE LO STESSO CT. ALLORA L'AVVERSARIO PUÓ CAPIRE QUANDO DUE MSG CRIPTATI SONO UGUALI.

(GRAVE QUANDO LO SPAZIO DEI MSG M É PICCOLO)

⇒ Bisogna fare in modo che quando le chiavi sono usate più volte (many-time key), dato lo stesso PT venga prodotto ogni volta un CT diverso.

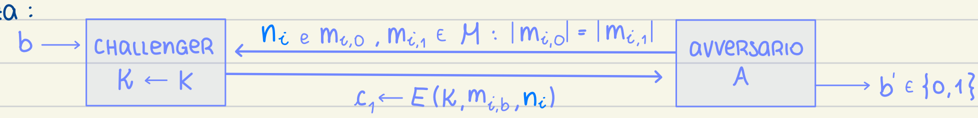
Soluzione 1) Randomized encription . $len(CT) = len(PT) + \#$random bits (il CT è più lungo del PT)

Soluzione 2) nonce-based encription : viene usato un valore nonce n che cambia ad ogni messaggio, il mittente, oltre ad indicare il messaggio m, indica anche il nonce n.
Il nonce non deve essere per forza né segreto né random.
La coppia (K, n) non viene mai usata più di una volta.

2 tipologie di nonce :

- counter : usato quando E tiene traccia dello stato da msg a msg. Inoltre se il destinatario sta tenendo traccia anch'esso dello stesso stato, non è necessario inviargli n assieme al CT.
- Random nonce : $n \leftarrow \mathcal{N}$. $\mathcal{N}$ deve essere definito grande abbastanza per fare in modo che lo stesso n non si ripeta con alta probabilità.

Sicurezza :



Tutti i nonce $\{n_1, \dots, n_q\}$ devono essere diversi tra loro.

Q è un cifrario nonce-based semanticamente sicuro se per ogni avversario efficiente A :
$$Adv_{nCPA}[A, Q] := |P[EXP(0)=1] - P[EXP(1)=1]| \text{ è trascurabile.}$$

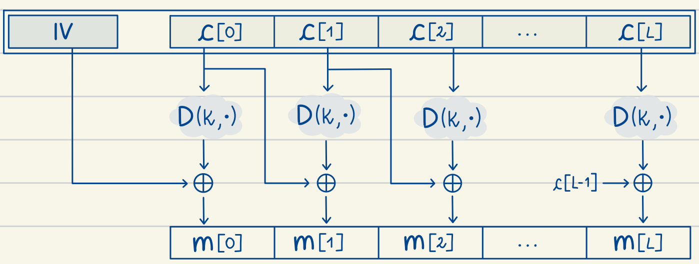
CBC : Cipher Block Chaining

Versione 1) CBC con random IV (Inizializzation Vector)

$D = K \times \{0,1\}^n \longrightarrow \{0,1\}^n$ è l'algoritmo inverso di E.

Decriptazione

$D_{CBC}(K, c)$:



$m[0] = D(K, c[0]) \oplus IV$

$m[i] = D(K, c[i]) \oplus c[i-1]$

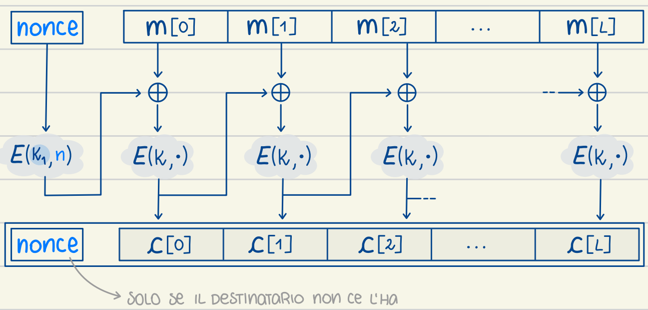
attacco : se l'avversario può predirre l'IV, allora non è più sicuro.

Versione 2) CBC nonce-based

chiave $= (K, K_1)$?

Criptazione

$E_{nCBC}(K, m, n)$.

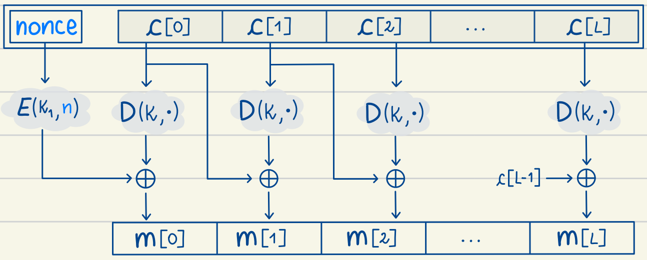


$c[0] = E(K, E(K_1, n) \oplus m[0])$

$c[i] = E(K, c[i-1] \oplus m[i])$

DECRIPTAZIONE

$D_{nCBC}(K, c, n)$:



$m[0] = D(K, c[0]) \oplus E(K_1, n)$

$m[i] = D(K, c[i]) \oplus c[i-1]$

SLIDE 05

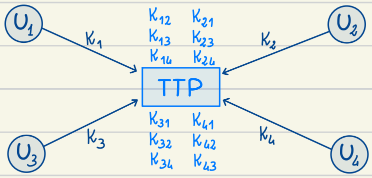
SCAMBIO DELLE CHIAVI

TRUSTED 3RD PARTIES

KEYS MANAGER : C'É LA NECESSITÁ DI MEMORIZZARE LE CHIAVI : OGNI UTENTE DEVE MEMORIZZARE UNA CHIAVE PER OGNI ALTRO UTENTE. $O(n)$ CHIAVI $\forall$ UTENTE $\Rightarrow$ $O(n^2)$ CHIAVI IN TOTALE. SONO TROPPE, É UN PROBLEMA.

SOLUZIONE : TTR (TRUSTED THIRD PARTY) : MEMORIZZA LE CHIAVI AL POSTO DEGLI UTENTI.

OGNI UTENTE MEMORIZZA UNA SOLA CHIAVE DI "ACCESSO" AL TTP.



COME IL TTP GENERA LE CHIAVI :

(1) IL MITTENTE A (K_A) DICE AL TTP CON CHI VUOLE COMUNICARE : B (K_B)

(2) IL TTP SCEGLIE UNA CHIAVE RANDOM K_{AB}

(3) IL TTP MANDA AD A IL MESSAGGIO « "A,B" ‖ K_{AB} » CRIPTATO CON LA CHIAVE DI A : $E(K_A, \text{"A,B"} \| K_{AB})$

(4) A DECRIPTA IL MESSAGGIO RICEVUTO DAL TTP E LO INOLTRA A B CRIPTANDOLO CON LA CHIAVE DI B : $E(K_B, \text{"A,B"} \| K_{AB})$

IN QUESTO MODO SIA A CHE B HANNO LA CHIAVE K_{AB}, QUESTA CHIAVE PERMETTE AD A E B DI AVERE UNA SOLA COMUNICAZIONE, SUCCESSIVAMENTE SARÁ NECESSARIO GENERARNE UN'ALTRA PER POTER COMUNICARE ANCORA

MARKLE PUZZLES

PERMETTE DI GENERARE UNA CHIAVE CONDIVISA SENZA DOVER USARE UN ONLINE TTP.

I PUZZLES SONO PROBLEMI RISOLVIBILI CON UN PO' DI SFORZO.

É QUINDI POSSIBILE CHE A E B COMUNICHINO CON UNA CHIAVE CONDIVISA USANDO UNA SEMPLICE CRITTOGRAFIA SIMMETRICA, INTEGRANDO NEL MESSAGGIO CIFRATO UN PUZZLE TRA UN POOL DI POSSIBILI PUZZLE. RISULTA PERÓ ESSERE INEFFICIENTE.

PROTOCOLLO DIFFIE-HELLMAN

PERMETTE DI GENERARE UNA CHIAVE CONDIVISA SENZA DOVER USARE UN ONLINE TTP.

PARTENDO DALLA SITUAZIONE IN CUI A E B NON CONOSCONO ALCUNA INFORMAZIONE SEGRETA, A E B SI SCAMBIANO DEI MESSAGGI E AL TERMINE DI QUESTO SCAMBIO ENTRAMBI CONOSCONO K :

(1) A E B SI SCAMBIANO p, NUMERO PRIMO, E g, NUMERO INTERO $\{2, \dots, p-2\}$

(2) A SCEGLIE UN VALORE RANDOM $a \in \{1, \dots, p-2\}$ CALCOLA $g^a \pmod p$ E LO INVIA A B

(3) B SCEGLIE UN VALORE RANDOM $b \in \{1, \dots, p-2\}$ CALCOLA $g^b \pmod p$ E LO INVIA A A

(4) A CALCOLA LOCALMENTE $(g^b)^a \pmod p$ E B CALCOLA LOCALMENTE $(g^a)^b \pmod p$ COSÍ FACENDO ENTRAMBI AVRANNO LA CHIAVE $g^{ba} \pmod p$

UN POSSIBILE EAVESTOPPER AVREBBE DIFFICOLTÁ PERCHÉ DOVREBBE ESEGUIRE TUTTE QUESTE COMPUTAZIONI COMPLESSE IN LOCALE, MENTRE A E B SE LE SONO DIVISE. (BANALE, FORSE ERA UTILE NEGLI ANNI '80 AL MASSIMO)

SLIDE 06

ARITMETICA MODULARE

N = INTERO POSITIVO

p = NUMERO PRIMO

$\mathbb{Z}_N = \{0, 1, \dots, N-1\}$

$5-7 = ?$ in $\mathbb{Z}_{12}$ $\longrightarrow$ $5-7 \pm K\,12 = \{0,1,\dots,11\}$

$-2 \pm K \cdot 12 = \{0,1,\dots,11\}$

$-2 + 1 \cdot 12 = 10 \in \{0,1,\dots,11\}$ ✓

MASSIMO COMUNE DIVISORE TRA x e y : $\exists\, a, b$ INTERI T.C. $a \cdot x + b \cdot y = mcd(x,y)$

SE $mcd(x,y) = 1 \implies x$ e y SONO RELATIVAMENTE PRIMI.

L'INVERSO DI x IN $\mathbb{Z}_N$ È L'ELEMENTO $x^{-1} \in \mathbb{Z}_N$ TALE CHE $x \cdot x^{-1} = 1$

2^{-1} ? IN $\mathbb{Z}_{12}$ $\longrightarrow$ $2^{-1} = \frac{N+1}{2}$ POICHÉ $2 \cdot \frac{N+1}{2} = N+1 = (N+1) \% N = 1$

$x \in \mathbb{Z}_N \iff mcd(x, N) = 1$

$\mathbb{Z}_N^* = \{x \in \mathbb{Z}_N \mid mcd(x, N) = 1\}$ GLI ELEMENTI INVERTIBILI IN $\mathbb{Z}_N$

PER I NUMERI PRIMI p VALE CHE $\mathbb{Z}_p^* = \mathbb{Z}_p \setminus \{0\} = \{1, 2, \dots, p-1\}$

$\mathbb{Z}_{12}^* \subseteq \{0,1,\dots,11\} = \{1,5,7,11\}$

PER RISOLVERE x^{-1} CON $x \in \mathbb{Z}_N^*$ SI PUÒ USARE L'ALGORITMO DI EUCLIDE ESTESO.

$a \cdot x + b\, y = 0$ IN $\mathbb{Z}_N$ $\implies$ $x = -b \cdot a^{-1}$ IN $\mathbb{Z}_N$

TEOREMA DI FERMAT : $\forall\, x \in \mathbb{Z}_p^*$, $x^{p-1} = 1$ IN $\mathbb{Z}_p$

$p = 5 \Rightarrow 3^{5-1} = 3^4 = 81 = 1$ IN $\mathbb{Z}_5$

$\implies x \in \mathbb{Z}_p^* \Rightarrow x \cdot x^{p-1} = 1 \implies x^{-1} = x^{p-2}$ IN $\mathbb{Z}_p$

GENERARE NUMERI PRIMI RANDOM

PER GENERARE RANDOMICAMENTE UN NUMERO PRIMO LUNGO 1024 BITS :

(1) SCEGLI UN NUMERO INTERO RANDOM $p \in [2^{1024}, 2^{1025} - 1]$

(2) VERIFICA SE $2^{p-1} = 1$ IN $\mathbb{Z}_p$:

SE SÌ RITORNA p, ALTRIMENTI RITORNA A (1)

TEOREMA DI EULERO : $\mathbb{Z}_p^*$ È UN 'GRUPPO CICLICO' : $\exists\, g \in \mathbb{Z}_p$ TALE CHE $\{1, g, g^2, g^3, \dots, g^{p-2}\} = \mathbb{Z}_p^*$

QUESTO VALORE g SI CHIAMA GENERATORE DI $\mathbb{Z}_p^*$

$p = 7$ $\quad \mathbb{Z}_7 = \{1, 3, 3^2, 3^3, 3^4, 3^5\} = \{1, 3, 2, 6, 4, 5\}$ DOPO RICOMINCIA DA CAPO : $3^6 = 243 \% 7 = 1$, $3^7 = 729 \% 7 = 3$, ...

$3 \cdot 3 = 9 \% 7 = 2$ $\quad 3 \cdot 3 \cdot 3 \cdot 3 = 81 \% 7 = 4$

L'INSIEME $\{1, g, g^2, g^3, \dots, g^{p-2}\}$ VIENE CHIAMATO GRUPPO GENERATO DA g E SI INDICA CON $<g>$.

L'ORDINE DI $g \in \mathbb{Z}_p$ È LA DIMENSIONE DI $<g>$: $ORD_p(g) = |<g>|$ = IL VALORE MINORE $a > 0$ T.C. $g^a = 1$ IN $\mathbb{Z}_p$.

TEOREMA DI LAGRANGE : $\forall\, g \in \mathbb{Z}_p^*$, $ORD_p(g)$ DIVIDE $p-1$.

DEFINISCO UN INTERO N. LA FUNZIONE DI EULERO $\varphi(N) = |\mathbb{Z}_N^*|$

Teorema di Eulero : $\forall x \in \mathbb{Z}_N^*$, $x^{\varphi(N)} = 1$ in $\mathbb{Z}_N$.

Risolvere le equazioni lineari : $a \cdot x + b y = 0$ in $\mathbb{Z}_N$ $\Longrightarrow$ $x = -b \cdot a^{-1}$ in $\mathbb{Z}_N$

Risolvere quelle con grado polinomiale ?

L'elemento $x \in \mathbb{Z}_p$ tale che $x^n = c$ in $\mathbb{Z}_p$ è chiamato radice n-esima di c.

$7^{\frac{1}{3}} = \sqrt[3]{7} \;\%\; 11 = 6 \iff 6^3 = 216 \;\%\; 11 = 7$

Quando esiste $c^{\frac{1}{n}}$ in $\mathbb{Z}_p$?

- caso semplice :

  se $mcd(n, p-1) = 1$, allora $\forall c \in \mathbb{Z}_p^* : c^{\frac{1}{n}} \in \mathbb{Z}_p$ ed è facile da trovare.

- caso $n = 2$: $mcd(2, p-1) \neq 1$ (se p è un numero primo dispari)

  $x \in \mathbb{Z}_p$ è un residuo quadratico (Q.R) se ha radice quadrata $\in \mathbb{Z}_p$ ($\sqrt{x} \in \mathbb{Z}_p$).

  p è numero primo dispari $\Rightarrow$ il numero di Q.R. in $\mathbb{Z}_p$ è $\frac{p-1}{2} + 1$

Teorema di Eulero : $x \in \mathbb{Z}_p^*$ è un Q.R. $\iff$ $x^{\frac{p-1}{2}} = 1$ in $\mathbb{Z}_p$ con p numero primo dispari

Risolvere le equazioni quadratiche : $a \cdot x^2 + b \cdot x + c = 0$ in $\mathbb{Z}_p$ $\Longrightarrow$ $x = \dfrac{-b \pm \sqrt{b^2 - 4 \cdot a \cdot c}}{2}$ in $\mathbb{Z}_p$

SLIDE 06.1

Crittografia asimmetrica

mittente B
destinatario A

Chiave pubblica : chiave relativa ad un solo utente A, conosciuta da tutti.

Criptazione con chiave pubblica : usata da B quando vuole inviare un messaggio ad A, successivamente A decifrerà il messaggio con la sua chiave privata. (non autenticato confidenziale)

Decriptazione con chiave pubblica : usata da A quando riceve un messaggio da B, precedentemente criptato con la chiave privata di B. (autenticato ma non confidenziale)

Chiave privata : chiave relativa ad un solo utente A, sconosciuta a tutti tranne che ad A.

Criptazione con chiave privata : usata da B quando vuole inviare un messaggio ad A, successivamente A decifrerà il messaggio con la chiave pubblica di B. (autenticato ma non confidenziale)

Decriptazione con chiave privata : usata da A quando riceve un messaggio da B, precedentemente criptato con la chiave pubblica di A. (non autenticato confidenziale)

FIRMA DIGITALE

RSA

è molto difficile trovare la chiave privata avendo la pubblica.

Confidenziale : messaggio criptato con la chiave pubblica del destinatario, decifrabile solo con la chiave privata del destinatario.

Chiave pubblica

Un sistema di criptazione a chiave pubblica è una tripla di algoritmi (G, E, D) dove :

G() : genera randomicamente una coppia di chiavi : PK (private key) e SK (secure key)

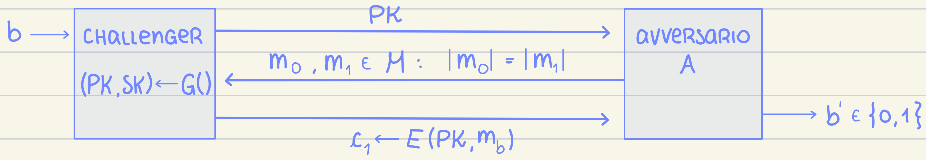
E(PK, m) : ritorna $c \in C$ randomicamente

D(SK, c) : ritorna $m \in M$ o $\perp$ deterministicamente

Proprietà di consistenza : $\forall (PK, SK)$ ritornate da G , $\forall m \in M$: $D(SK, E(PK, m)) = m$

Sicurezza semantica

Per $b = 0,1$ definisco $EXP(0)$ e $EXP(1)$ come:



$\mathbb{E} = (G, E, D)$ è semanticamente sicuro se per ogni avversario efficiente A:

$$Adv_{SS}[A, \mathbb{E}] := |P[EXP(0) = 1] - P[EXP(1) = 1]| \text{ è trascurabile.}$$

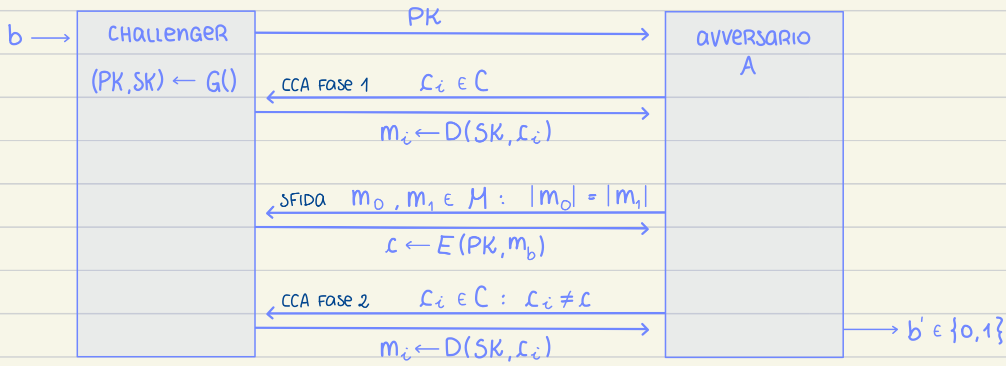
OSS: one-time security $\Longrightarrow$ many-time security

mentre per i cifrari simmetrici: one-time security $\not\Longrightarrow$ many-time security

CCA: Chosen Ciphertext Security

Sia $\mathbb{E} = (G, E, D)$ un cifrario a chiave pubblica definito su (M, C).

Per $b = 0,1$ definisco $EXP(0)$ e $EXP(1)$ come:



$\mathbb{E} = (G, E, D)$ è CCA-sicuro se per ogni avversario efficiente A:

$$Adv_{CCA}[A, \mathbb{E}] := |P[EXP(0) = 1] - P[EXP(1) = 1]| \text{ è trascurabile.}$$

TrapDoor Permutation

TrapDoor Permutations

TDF: TrapDoor Function

La funzione TDF: $X \longrightarrow Y$ è una tripla di algoritmi efficienti (G, F, F^{-1}) dove:

$G()$: genera randomicamente una coppia di chiavi: PK (private key) e SK (secure key).

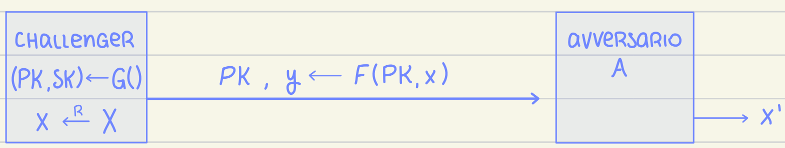
$F(PK, \cdot)$: definisce deterministicamente una funzione $X \longrightarrow Y$.

$F^{-1}(SK, \cdot)$: definisce una funzione $Y \longrightarrow X$ che inverta $F(PK, \cdot)$.

Proprietà di consistenza: $\forall (PK, SK)$ ritornate da G, $\forall m \in M$: $F^{-1}(SK, F(PK, x)) = x$

TDFs: secure TrapDoor Function

(G, F, F^{-1}) è sicura se $F(PK, \cdot)$ è una funzione "one way": facile da computare, ma molto difficile da invertire se non si ha SK.



$\xleftarrow{R}$: Random

(G, F, F^{-1}) è un TDF sicuro se per ogni avversario efficiente A:

$Adv_{OW}[A, F] := P[x = x'] <$ trascurabile.

Hash function

Hash function

input di lunghezza arbitraria

output di lunghezza fissata (tipicamente molto più corto dell'input)

algoritmo one-way hash

sia H( ) l'algoritmo hash one-way, allora:

- input: stringa m di lunghezza arbitraria.
- output: stringa binaria H(m) lunga L bits, chiamata "hash di m su H".
- la lunghezza L è fissata per la funzione hash one-way data.

MD5 ha L = 128 bits

SHA-1 ha L = 160 bits

proprietà di un buon algoritmo hash one-way:

- facile da calcolare — l'algoritmo deve essere veloce
- difficile da invertire — non ci deve essere un algoritmo inverso fattibile
- difficile che generi collisioni — deve essere infattibile trovare due input che generino lo stesso hash
- un minimo cambiamento nell'input deve generare un cambio radicale nel valore hash

sistema di criptazione a chiave pubblica creato su un TDFs

siano: (G, F, F^{-1}) un TDF sicuro $X \to Y$

(E_s, D_s) un cifrario simmetrico definito su (K, M, C)

$H : X \to Y$ una funzione hash

costruiamo un sistema di criptazione a chiave pubblica (G, E, D) dove il generatore di chiavi G è lo stesso per le TDF

criptazione: $E(PK, m) \longrightarrow x \xleftarrow{R} X$

$y \leftarrow F(PK, x)$

$k \leftarrow H(x)$

$c \leftarrow E_s(k, m) \longrightarrow$ output: (y, c)

decriptazione: $D(SK, (y, c)) \longrightarrow x \leftarrow F^{-1}(SK, y)$

$k \leftarrow H(x)$

$m \leftarrow D_s(k, c) \longrightarrow$ output: m

se (G, F, F^{-1}) è un TDF sicuro, (E_s, D_s) fornisce una cifratura autenticata e $H : X \to Y$ è un oracolo randomico, allora (G, E, D) è CCA^{RO} sicuro

oss: MAI applicare $F(PK, \cdot)$ direttamente al PT poiché non sarebbe semanticamente sicuro ed esistono numerosi attacchi.

RSA

RSA (TDP)

Si basa sulle proprietà dei numeri primi e sulla teorei dei numeri.

$\underline{G()}$: (1) sceglie randomicamente due numeri primi p e $q \approx 1024$ bits. Fissa $N = p \cdot q$
(2) sceglie due interi e e d tali che $e \cdot d = 1 \% \varphi(N)$
(3) ritorna $PK = (N, e)$, $SK = (N, d)$

$\underline{F(PK, x)}$: $\mathbb{Z}_N^* \longrightarrow \mathbb{Z}_N^*$ è $RSA(x) = x^e$ in $\mathbb{Z}_N$
(1) il mittente cerca la chiave pubblica del destinatario $PK = (N, e)$
(2) calcola il CT $c = m^e \% N$ e lo invia al destinatario

$\underline{F^{-1}(SK, y)} = y^d$ dove $y^d = RSA(x)^d = x^{ed} = x^{K \cdot \varphi(N) + 1} = (x^{\varphi(N)})^K \cdot x = x$
(1) il destinatario riceve il CT
(2) calcola il PT usando la sua chiave privata $SK = (N, d)$: $m = c^d \% N$

chiavi di B : (1) sceglie due numeri primi : $p = 5$ e $q = 11$
(2) moltiplica p e q . $N = p \cdot q = 55 \implies \varphi(N) = 40$
(3) sceglie un numero : $e = 3$ t.c. $mcd(e, 40) = 1$
(4) calcola d t.c. soddisfi $(e \cdot d) \% \varphi(N) = 1$ ovvero $(3 \cdot d) \% 40 = 1$, $d = 27$
chiave pubblica di B : $(N, e) = (55, 3)$
chiave privata di B : $(N, d) = (55, 27)$

A mittente : vuole mandare un msg $m = 13$ a B
trova la chiave pubblica di B : $(55, 3)$
calcola c nel seguente modo : $c = m^e \% N$
$= 13^e \% 55$
$= 13^3 \% 55$
$= 2197 \% 55$
$= 52$
manda il CT $c = 52$ a B

criptazione

B destinatario : riceve il CT $c = 52$ da A
usa la sua chiave privata $(55, 27)$ per calcolare m :
$m = c^d \% N$
$= 52^{27} \% 55$
$= 13$

decriptazione

$\Rightarrow$ $E(PK, m)$: (1) sceglie randomicamente x in $\mathbb{Z}_N$
(2) $y \leftarrow RSA(x) = x^e$ e $K \leftarrow H(x)$
(3) ritorna $(y, E_S(K, m)) = (y, c)$

$D(SK, (y, c))$: ritorna $D_S(H(RSA^{-1}(y)), c) = m$

L'RSA TDP non è uno schema di cifratura poiché semanticamente non sicuro

RSA è davvero una funzione one way ?
per invertire la funzione senza avere d a disposizione , l'attaccante deve calcolare x da c , ovvero deve calcolare $x^e \% N$ ed è fattibile.

FIRMA DIGITALE

RSA nella pratica

Per accelerare la criptazione bisogna scegliere un e piccolo, il minimo è $e = 3$
Il valore consigliato è $e = 65537 = 2^{16}+1$

attacchi

- Timing attack: il tempo necessario per calcolare $c^d \% N$ può rivelare qualcosa sul valore d
- Power attack: la potenza necessaria per calcolare $c^d \% N$ può rivelare qualcosa sul valore d
- Fault attack: un errore di computazione nel calcolo di $c^d \% N$ può rivelare qualcosa sul valore d

Problemi nella generazione delle chiavi

Un generatore di chiavi $G()$ è OpenSSL.
Questo algoritmo:
(1) genera il numero primo p partendo dal seed di inizializzazione;
(2) aggiunge al seed un numero di bits per modificare la randomizzazione;
(3) genera il numero primo q dal seed "modificato".
(4) calcola $N = p \cdot q$

Supponendo di avere un seed iniziale con pochi elementi, diversi device genereranno lo stesso valore p, ma diversi q producendo così diversi N: N_1 e N_2 t.c. $mcd(N_1, N_2) = p$!

Firma digitale

$G()$ genera la chiave pubblica e privata degli utenti come per RSA.
Mittente:
(1) il mittente calcola la firma digitale s del messaggio m usando la sua chiave privata d: $s = m^d \% N$.
(2) concatena m e s: (m, s) indica il messaggio m e la firma digitale s del mittente su m.
(3) il mittente invia al destinatario la coppia (m, s).
Destinatario:
(1) il destinatario riceve (m, s) e cerca la chiave pubblica (e, N) del mittente.
(2) calcola $t = s^e \% N$.
(3) controlla se $t = m$: se sì significa che m non è stato modificato lungo il tragitto e che il mittente è certificato, altrimenti rifiuta il messaggio: verifica della firma.

La firma digitale non assicura la confidenzialità ma assicura l'autenticità e la non ripudiabilità.

Se i messaggi da inviare sono lunghi si esegue un'operazione intermedia:
(1) si calcola l'hash del messaggio per renderlo di una lunghezza prestabilita
(2) si firma digitalmente l'hash

PRO

- praticamente impossibile da codificare
- non ripudiabile
- verificabile universalmente
- è diversa per ogni messaggio poiché si basa sul messaggio stesso

altri sistemi crittografici asimmetrici

ElGamal

G() : (1) sceglie un numero primo p e una "radice primitiva modulo p" g

$\forall a$ numero coprimo di p, $\exists k$ un intero t.c. $g^k = a \% p$

due interi sono coprimi se il loro mcd = 1

(2) sceglie un esponente random a in $[0, \dots, p-2]$

(3) computa $A = g^a \% p$

Publick Key = (p, g, A)

Secret Key = a

mittente :

(1) cerca la chiave pubblica del destinatario (p, g, A)

(2) sceglie un esponente random b in $[0, \dots, p-2]$

(3) calcola $B = g^b \% p$ e $c = A^b \cdot m \% p$ $\longrightarrow$ $CT = (B, c)$

(4) invia il CT al destinatario

destinatario :

(1) riceve il CT (B, c) dal mittente

(2) usa la sua chiave privata a per calcolare m nel modo seguente:

- $x = p-1-a$
- $m = B^x \cdot c \% p$

contro

espansione del messaggio : il CT è lungo $2 \cdot PT$

Rabin

G() : (1) genera randomicamente due numeri primi molto grandi p e q tali che : $p \% 4 = q \% 4 = 3$

(2) calcola $N = p \cdot q$

(3) $PK = N$, $SK = (p, q)$

mittente :

(1) cerca la chiave pubblica del destinatario N

(2) calcola il CT $c = m^2 \% N$

(3) invia il CT al destinatario

destinatario :

(1) riceve il CT c dal mittente

(2) usa la sua chiave privata (p, q) per calcolare m nel modo seguente:

- $m_p = c^{(p+1)/4} \% p$
- $m_q = c^{(q+1)/4} \% q$
- trova y_p e y_q tali che $y_p \cdot p + y_q \cdot q = 1$
- $r = (y_p \cdot p \cdot m_q + y_q \cdot q \cdot m_p) \% N$
- $s = (y_p \cdot p \cdot m_q - y_q \cdot q \cdot m_p) \% N$
- uno tra $r, -r, s, -s$ è il messaggio originale m