

CRITTOGRAFIA

CRITTOGRAFIA = Metodi di cifratura
↑

la crittografia è lo studio di tecniche matematiche sofisticate per manutenere i messaggi o

tentare di svelarli → CRIPTOANALISI = Metodi di interpretazione

Quindi il mondo della crittografia si divide in : 1) Persone che applicano i metodi di cifratura

2) Persone che applicano metodi di criptoanalisi

CRITTOLOGIA: CRITTOGRAFIA + CRITTOLOGIA = Studio della comunicazione su canali non sicuri e relativi problemi.

Per poter inviare un messaggio su un canale non sicuro, bisogna cifrare il messaggio m in un crittogramma C , leggibile solo tramite decifratura da parte del destinatario

CIFRATURA: Operazione che trasforma un messaggio in chiave in un crittogramma.

$$C : MSG \rightarrow \text{CRIPTOGRAFIA}$$

DECIFRATURA: Operazione che permette di ricavare il messaggio dal crittogramma

$$D : \text{CRIPTOGRAFIA} \rightarrow MSG$$

le funzioni C e D sono uno l'inverso dell'altra.

$$D(C(m)) = m$$

↳ C deve essere iniettiva cioè 1 crittogramma = 1 solo messaggio

Livelli di segretezza: Classificazione metodi cifranti in base al livello di segretezza.

1) Cifranti per uso riservato: le funzioni C e D sono tenute segrete. (Usate per cose militari/politiche. Non per cose di mano)

2) Cifranti per uso generale: le funzioni C e D sono note, ma c'è una informazione aggiuntiva detta "chiave" che è segreta nota solo dai due che stanno comunicando.

CIFRARI PER USO GENERALE:

- 1) Funzioni C e D note
- 2) Chiave segreta K
 - a) Diversa per ogni coppia di utenti.
 - b) Inserita come parametro in C e D $\Rightarrow c = C(m, K)$
 - c) Dimensione delle chiavi molto grande $m = D(c, K)$
 - d) Chiave scelta in modo casuale.

Vantaggi:

- 1) Tenere segreta la chiave è più facile che tenere segreto C e D
- 2) Tutti possono usare C e D con chiavi diverse
- 3) Se un crittanalista entra in possesso di una chiave \Rightarrow Basta generare una nuova

Tanti cifrari usano questa tecnica: 3DES, RC5, IDEA, AES

CRITTO ANALISTA (EVE)

Un crittanalista può avere due comportamenti:
1) Passivo: Ascolta la comunicazione
2) Attivo: Ascolta la comunicazione, la disturba e modifica

Attacchi a un sistema crittografico:

La pericolosità dipende dalle info in possesso del crittanalista:

- 1) Cipher text attack: Un crittanalista rileva solo i messaggi crittati c_1, \dots, c_n
- 2) Known Plain-text Attack: Un crittanalista conosce una serie di coppie: $(m_1, c_1), \dots, (m_n, c_n)$
- 3) Chosen Plain-text Attack: Un crittanalista si procura una serie di coppie $(m_1, c_1), \dots, (m_n, c_n)$ relative a messaggi in chiave da lui scelti

Han man in the middle: Un crittanalista si mette sul canale di comunicazione senza che il mittente ed il destinatario se ne accorgano. Pensano di comunicare tra di loro ma è il crittanalista che gestisce la comunicazione.

Le cifrari diffusi adesso, non sono perfetti ma sono sicuri perché richiedono la risoluzione di problemi matematici estremamente difficili. \Rightarrow Tempi lunghi di calcolo
 Il problema è che non sappiamo se:
 1) I problemi richiedono necessariamente tempi enormi
 2) Non sono stati individuati metodi più efficienti di risoluzione

Cifrari a chiave privata (sistemi simmetrici)

Stessa chiave privata per cifrare e decifrare

Cifrari a chiave pubblica (sistemi Asimmetrici)

C'è una chiave pubblica per cifrare k_{pub} e una chiave privata per decifrare k_{priv}
 nota solo al destinatario.

La funzione di cifratura deve essere **one-way trap door** cioè calcolare $C(m, k_{pub})$

dove essere facile ma decifrare C deve essere difficile a meno di non conoscere il meccanismo
 segreto rappresentato da k_{priv} (**trap-door**)

Un crittanalista non può decifrare C nemmeno se conosce k_{pub} , C e D
 ma deve conoscere k_{priv}

Vantaggi: a) Se ho n utenti necessito di n chiavi pubbliche/ private rispetto

$$a \frac{n(n-1)}{2}$$

b) Non è richiesto lo scambio segreto di chiavi

Svantaggi: a) Sistemi molto più lenti

b) Esposti ad attacchi di tipo chosen plain text \rightarrow Cifranti

Possedere un tot
 di messaggi diversi
 → confrontarli con
 gli altri inviati
 dagli utenti

CIFRARI IBRIDI

1) Si usa AES per la comunicazione con chiavi private (cifratura simmetrica)

ADVANCED ENCRYPTION STANDARD (AES) → Cifrario Simmetrico a blocchi

- a) Stessa chiave per cifrare e decifrare
 - b) Messaggio è diviso in blocchi lunghi come la chiave
 - c) La chiave viene usata per trasformare un blocco del messaggio in un blocco del cattogramma
 - d) Le chiavi vengono stabilite dai mezzi elettronici usati.
- 2) Si usa la cifratura asimmetrica per comunicare la chiave privata a mittente e destinatario.

Risolvendo l'attacco chosen-plain-text se la chiave per l'AES rimasta imprendibile!

Sistemi crittografici:

Dovono garantire:

- 1) Segretezza → Come visto sopra
- 2) Identificazione dell'utente → Accertare l'identità di chi accede ai servizi
- 3) Autenticazione del messaggio → Capire se il messaggio non sia stato modificato/cambiato
- 4) Firma digitale → Poter dimostrare chi è il mittente del messaggio

Rappresentazione degli oggetti:

Un insieme finito di caratteri o simboli:

Un oggetto è rappresentato da una sequenza ordinata di caratteri dell'alfabeto Γ , $|\Gamma| = s$

Se ho N oggetti da rappresentare:

- a) $d(s, N)$ = lunghezza della sequenza di caratteri più lunga che rappresenta un oggetto dell'insieme
- b) $d_{\min}(s, N)$ = Valore minimo di $d(s, N)$ tra tutte le rappresentazioni possibili

Un metodo di rappresentazione è tanto migliore, tanto più $d(s, N)$ si avvicina a $d_{\min}(s, N)$

Rappresentazione unaria

$S = 1, \Gamma = \{0\} \Rightarrow$ le rappresentazioni possono avere solo sequenze di 0

Quindi se ho N oggetti allora $d_{\min}(1, N) = N$

N oggetti
0, 00, 000, ..., $\underbrace{00...00}_N$

Rappresentazione binaria

$S = 2, \Gamma = \{0, 1\}$

Sia $k > 1$ lunghezza delle sequenze \rightarrow Abbiamo 2^k sequenze di lunghezza $k \rightarrow$

Quindi: il numero di sequenze binarie lunghe da 1 a k è $2^{k+1}-2 = \sum_{i=1}^k 2^i$

$$k \\ \begin{array}{l} 100 \\ 01 \\ 110 \\ 10 \end{array} \\ u=2$$

In fine per rappresentare N oggetti vogliamo che $2^{k+1}-2 > N$

$$\Rightarrow k \geq \log_2(N+2) - 1$$

Quindi:

$$d_{\min}(2, N) = \log_2(N+2) - 1 \Rightarrow \lceil \log_2 N \rceil - 1 \leq d_{\min}(2, N) \leq \lceil \log_2 N \rceil$$

\uparrow
lunghezza sequenze

- 1) Bastano $\lceil \log_2 N \rceil$ caratteri binari per contenere N sequenze differenti
- 2) Si possono contenere N seq. differenti tutte di $\lceil \log_2 N \rceil$ caratteri

Rappresentazione S-aria

$s =$ lunghezza insieme

- 1) Bastano $\lceil \log_s N \rceil$ caratteri per contenere N sequenze differenti
- 2) Si possono contenere N seq. differenti tutte di $\lceil \log_s N \rceil$ caratteri

Rappresentazioni efficienti

Una rappresentazione è efficiente se usa un numero minimo di caratteri di ordine logaritmico nella cardinalità di N dell'insieme da rappresentare.

Attenzione: l'alfabeto deve contenere almeno 2 caratteri $\Rightarrow S = 2$

Conviene inoltre usare una lunghezza prefissata per ogni sequenza cioè $d = \lceil \log_2 N \rceil$,
dove $\lceil \log_2 N \rceil \leq d \leq \lceil \log_s N \rceil + 1 = d_{\min}(s, N) + 1$

Teoria delle calcolabilità \rightarrow Si occupa delle questioni fondamentali su potenza e limitazioni dei sistemi di calcolo

Calcolabilità: Classificare i problemi in "risolvibili" e "Non risolvibili"

Complessità: Classificare i problemi in "facili" e "difficili"

Problemi: 1) Decidibili: Che possono essere risolti (risolvibili)

- Costo polinomiale (trattabili)

- Costo esponenziale (nottrattabili)

2) Non Decidibili: Che non possono essere risolti (Non risolvibili)

Ordinare le sequenze generate da un alfabeto:

1) Si definisce un ordine fra i caratteri dell'alfabeto

2) Si ordinano le sequenze in ordine di lunghezza crescente, in pari lunghezza, in ordine alfabetico

Una sequenza σ arbitraria si troverà tra quelle di 1 o 1 caratteri, in posizione alfabetica,

tra queste. Essa può essere messa in relazione con un numero intero che indica la posizione

nell'elenco. Le sequenze sono infinte e numerabili!

Questo è possibile perché esse sono di lunghezza finita anche se illimitata cioè per un qualunque

intero d esistono sequenze di lunghezza maggiore di d

Esempio: $\Gamma = \{A, \dots, z\} \Rightarrow$

⁰ A.	²⁰ $\dots z$,
²¹ AA, ... , A ²⁰ ,	²¹ BA, ... , B ²⁰ ,
²² AAA, ... , AA ²⁰ ,	²² ... , z ²⁰

Potete numerare le sequenze

Problemi computazionali



Possono essere visti come funzione matematica $f: \mathbb{N}^k \rightarrow \mathbb{N}^l$

teorema: L'insieme dei problemi computazionali non è numerabile

Dimo: Sia $F = \{ \text{Funzioni } | f: \mathbb{N} \rightarrow \{0,1\}^*\}$

Diagonalizzazione



1	2	3	4	5	6	7	8	...
$p_1(x)$	0	1	0	1				

$p_2(x)$	1	1	0	0				
----------	---	---	---	---	--	--	--	--

$p_3(x)$	0	0	1	1				
----------	---	---	---	---	--	--	--	--

:

$\begin{cases} 0 & \text{se } p_x(x) = 1 \\ 1 & \text{se } p_x(x) = 0 \end{cases}$

$\Rightarrow g$ non corrisponde a nessuna p_i della tabella

il che vuol dire che qualunque enumerazione scelta, \exists almeno una f esclusa $\Rightarrow F$ non numerabile

finito

Algoritmo: Unione di operazioni, completamente e univocamente determinate.

Devono essere rappresentati da sequenze finite di caratteri di un alfabeto finito
→ Gli algoritmi sono formalmente infiniti, ma numerabili.

Osservazione:

I problemi computazionali sono maggiori rispetto agli algoritmi perché non c'è detto che abbiano algoritmi che li risolvono

$$|\text{problem}i| \gg |\text{algoritmi}|$$

Problema dell'arresto (undecidibile)

Non è possibile stabilire se un programma, dato un input, termina in un tempo finito.

Dim: Se l'algoritmo "ARRESTO" fosse decidibile

$$\rightarrow \text{Arresto}(A, D) = 1 \quad \text{se } A(D) \text{ termina}$$

↓
algoritmo ↓
input

$$\text{Arresto}(A, D) = 0 \quad \text{se } A(D) \text{ non termina}$$

Se scegliamo $D = A$, cioè consideriamo $A(A) \Rightarrow \text{Arresto}(A, A) = 1 \Leftrightarrow A(A) \text{ termina}$

Quindi \exists consideriamo Paradosso(A) }

Paradosso(Paradosso)

while ($\text{Arresto}(A, A)$) } \Rightarrow

termina
 \Leftrightarrow

} :

$$x = \text{arresto}(A, A) = 0$$

\Leftrightarrow

$A(A)$ non termina

quindi: Paradosso(Paradosso) termina

\Leftrightarrow

$$x = \text{arresto}(\text{Paradosso}, \text{Paradosso}) = 0$$

\Leftrightarrow

Paradosso(Paradosso) Non termina

→ ASSURDO

TESE DI CHURCH-TURING

Tutti i modelli di calcolo risolvono esattamente lo stesso classe di problemi, dunque si equivalgono nella possibilità di risolvere problemi, operando con diversa efficienza.

La decidibilità è una proprietà del problema non dipende da quale modello utilizziamo.

Un miglioramento ai modelli (struttura di una macchina o istruzioni di un linguaggio di programmazione) serve solo a : 1) Abbassare il tempo di esecuzione
2) rendere più agevole la programmazione

Complessità computazionale

Un algoritmo si dice efficiente in base a quanto è veloce in tempo.

ordini di grandezza:

1) $\Theta(g(n))$: Unieme delle funzioni $f(n)$ t.c date 3 costanti $c_1, c_2, n_0 > 0$

$$c_1 g(n) \leq f(n) \leq c_2 g(n) \quad \text{per } \forall n \geq n_0$$

ovvero la funzione è delimitata da $c_1 g(n)$ e $c_2 g(n)$

Sintesi: $f(n) = \Theta(g(n))$

$$\text{esempio: } f(n) = 3n^2 + n + 1 = \Theta(n^2)$$

2) $O(g(n))$: Unieme delle funzioni $f(n)$ t.c date 2 costanti c e $n_0 > 0$

$$f(n) \leq c g(n) \quad \forall n \geq n_0$$

ovvero la funzione è limitata superiormente da $g(n)$ a partire da una costante moltiplicativa

Sintesi: $f(n^k) = O(n^i)$ con $i \geq k$

$$\text{esempio: } f(n) = 3n^2 + n + 1 = O(n^2) \quad \text{ma anche } f(n) = 3n^2, n + 1 = O(n^3)$$

3) $\Omega(g(n))$: Unieme delle funzioni $f(n)$ t.c date 2 costanti c e $n_0 > 0$

$$c g(n) \leq f(n) \quad \forall n \geq n_0$$

Ovvero la funzione è limitata inferiormente da $g(n)$ a parte una costante moltiplicativa

Sintesi: $f(n^i) = O(n^i)$ con $i \leq k$

Esempio: $f(n) = 3n^2 + n + 1 = O(n^2)$ ma anche $f(n) = 3n^2 + n + 1 = O(n^{1.5})$

Attenzione: se $f(n) = \Theta(g(n)) \Leftrightarrow f(n) = O(g(n))$ e $f(n) = \Omega(g(n))$

Classi P, NP, co-NP, NPC

Certificato: Attestato di esistenza y della soluzione con dati d'ingresso x usando l'algoritmo A

$$\hookrightarrow A(x, y) = 1$$

Attenzione: se $A(x, y) = 0$ non vuol dire che non esiste soluzione per x , ma solo che y non ne è un certificato

1) P = classe dei problemi decisionali risolvibili con un algoritmo polinomiale

↳ Problemi trattabili

2) NP = classe dei problemi decisionali verificabili con un algoritmo polinomiale

↳ Problemi intrattabili

Certificato polinomiale: 1) Ogni x di lunghezza n ammette un certificato y di lunghezza polinomiale n

2) Esiste un algoritmo di verifica A polinomiale in n applicabile a ogni coppia $\langle x, y \rangle$

Attenzione: $P \subseteq NP$: Esiste un certificato vuoto y per ogni sequenza di ingresso x e definire $A(x, y)$ che ignora y e verifica l'accettabilità di x risolvendo direttamente il problema.

3) CO-NP = classe dei problemi decisionali P per cui P -complementare $\in NP$

$$\text{Se } P \in NP \Rightarrow \overline{P} \in CO-NP \quad \text{e quindi} \quad P \subseteq (NP \cap CO-NP)$$

↓
Contiene le istanze non accettabili per P

4) NPC = classe dei problemi decisionali t.c. $P \in NP$

↓
Problemi NP-completi $\forall P' \in NP$ Si ha $P' \leq_p P$

↓
 P' si riduce polinomialmente a P

Riduzione polinomiale: $P' \leq_p P \iff$ Esiste un algoritmo R t.c. trasforma ogni istanza

x_1 di P' in una istanza x_2 di P in modo che x_1 sia accettabile

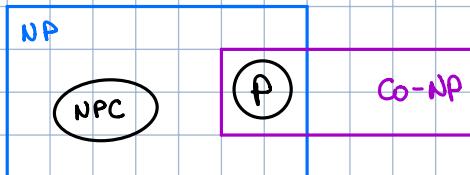
in P' e x_2 è accettabile in P . L'algoritmo R permette di

risolvere P' in tempo polinomiale se si sa risolvere P in tempo polinomiale

Si calcola $x_2 = R(x_1)$ $\forall x_1$,

e poi si risolve P' sulle x_2

Relazione tra le varie classi



OSSERVAZIONE: Se si scoprisse un algoritmo che risolve in tempo polinomiale un problema $P \in \text{NPC}$ \Rightarrow Tutti i problemi in NP potrebbero essere risolti in tempo polinomiale. Cioè che $P = \text{NP}$.

Se si dimostrasse il contrario, cioè che non esiste $\Rightarrow P \neq \text{NP}$

5) NPH: classe di problemi P t.c. $\forall P' \in \text{NP}$ si ha $P' \leq_p P$
 \hookrightarrow **NP-HARD**

Un problema P è NP-hard $\Leftrightarrow \forall P' \in \text{NPC}$ si ha $P' \leq_p P$

OSSERVAZIONE: I problemi NP-hard sono più difficili o di uguale difficoltà a quelli NP

Ge ruolo del caso

Complessità della sequenza casuale (secondo Kolmogorov):

Si definisce la complessità (secondo Kolmogorov) della sequenza casuale h rispetto al sistema di calcolo

S; come:

$$K_{S_i}(h) = \min \left\{ |P| \mid S_i(P) = h \right\} \rightarrow \text{cioè la lunghezza del più breve programma che genera } h.$$

dove P è la sequenza binaria che rappresenta un algoritmo per generare la sequenza binaria h in un sistema di calcolo S_i . $|P|$ è la lunghezza della sequenza P .

Si noti quindi che il programma P contiene la sequenza h al suo interno se h non obbedisce a nessuna legge semplice, cioè:

$$K_{S_i}(h) = |h| + c \quad \begin{array}{l} \text{costante per trasferire } h \text{ nel risultato da parte del prog.} \\ \text{esso dipende da } S_i \text{ e non da } h \end{array}$$

Sistema di calcolo universale

Esiste un sistema universale S_u tra tutti gli S_i t.c. può simulare il funzionamento di qualsiasi altro sistema di calcolo data la coppia $\langle i, p \rangle$

dove $q = \langle i, p \rangle$ per S_u ha lunghezza $|q| = |p| + \lceil \log_2 i \rceil$

\rightarrow programma che genera h in S_u

indice Sistema
di calcolo (S_i)

programma
che genera
 h in S_i

quindi abbiamo: $h = S_u(p) = S_u(\langle i, p \rangle)$

inoltre abbiamo che:

$$\forall h. \quad K_{S_u}(h) \leq K_{S_i}(h) + c_i$$

\rightarrow programma $|p|$ più breve tra tutti i sistemi S_i

$=$: Vale per le sequenze generate per simulazione di S_i

$<$: Vale per le sequenze generate con programmi più corti (Ad esempio per simulazione di un altro sistema S_j t.c. $S_i \neq S_j$)

Quindi dato che la complessità di h nel sistema S_i può essere identificata

Come $K_{S_u}(h)$ a meno di una costante. Poniamo trascurare il sistema usato ed indicare $K(h) = K_{S_u}(h)$

Abbiamo quindi:

Una sequenza h è casuale se \Rightarrow

\downarrow
la casualità è una
proprietà delle sequenze

$$K(h) \geq |h| - \lceil \log_2 |h| \rceil = |p| + 1$$

\downarrow
sottoggo perché
sempre avrei uno def.
+ troppo restrittivo

Teorema: $\forall n, \exists$ seq. casuali (secondo Kolmogorov) di lunghezza n .

Dim: Dato n , diamo:

$$1) \quad S = 2^n \Rightarrow \text{Sequenza binaria di lunghezza } n$$

\triangleright Vogliamo che $T < S$

2) $T = \#$ sequenze di lunghezza n , NON CASUALI

3) $N = \#$ sequenze binarie di lunghezza $< n - \lceil \log_2 n \rceil$

$$N = \sum_{i=0}^{n - \lceil \log_2 n \rceil - 1} 2^i = 2^{n - \lceil \log_2 n \rceil} - 1$$

Tra queste N sequenze ci sono anche prof. che generano le T sequenze
non casuali di lunghezza n , cioè:

$$T \leq N < S \Rightarrow T < S$$

$\forall n$, \exists certamente sequenze casuali di lunghezza n

e sono più numerose di quelle non casuali:

$$\frac{T}{S} < \frac{N}{S} = \frac{2^{n - \lceil \log_2 n \rceil} - 1}{2^n} = \frac{1}{2^{\lceil \log_2 n \rceil}} - \frac{1}{2^n} < \frac{1}{2^{\lceil \log_2 n \rceil}}$$

quindi se $\lim_{n \rightarrow \infty} \frac{T}{S} = 0$ perché $\lim_{n \rightarrow \infty} \frac{1}{2^{\lceil \log_2 n \rceil}} = 0$

Determinare se una sequenza h è casuale è un problema indecidibile

Sia h la sequenza e $\text{random}(h)$ un programma che verifica se h è random

Dato un algoritmo paradoxo:

Funzione Paradosso:

for binary $h = 1$ to ∞ do

if $(|h| - \log_2 |h| > |P|)$ and $(\text{Random}(h) = 1)$

then return h

Con: P = Sequenza binaria che rappresenta paradoxo e random

$$\hookrightarrow |P| = |\text{Paradoxo}| + |\text{Random}|$$

P è una costante che non dipende da h (appare come variabile)

abbiamo: Paradosso restituisc la prima seq. casuale h di lunghezza t.c

$$|h| - \lceil \log_2 |h| \rceil > |P|$$

essendo che esistono sequenze casuali di una qualsiasi lunghezza se
esiste una che soddisfa ① e ②

Ma:

① Il programma rappresentato da P è breve e genera h

$$u(h) \leq |P| < |h| - \lceil \log_2 |h| \rceil$$

$\Rightarrow h$ non casuale

② h è casuale

Assurdo!

Sorgente binaria casuale

Sorgente che genera una seq. di bit con le seguenti proprietà:

- 1) $P(0) > 0$, $P(1) > 0$ e prob. immutabili durante il processo di generazione
- 2) la generazione per ogni bit deve essere indipendente dagli altri.

E' quasi impossibile avere le complete indipendenza e casualità, si usano generatori pseudo-casuali.

Generatore di numeri pseudo-casuali:

È un algoritmo che partendo da un valore di input detto seed, genera una sequenza arbitrariamente lunga di numeri contenente una sequenza detta periodo che si ripete indefinitivamente (più è lungo il periodo, più è efficiente il generatore).

La limitazione di questi generatori è che possono generare al massimo 2^m sequenze diverse nel caso binario. ($m = \# \text{ bit del seed}$)

Sia n il numero di bit della sequenza generata e m i bit del seed
abbiamo che $m < n$ quindi:

$$2^m \ll 2^n$$

↓ ↓
 seq. diverse seq. diverse
 di n bit con n bit
 con il seed

Per essere usato in ambito crittografico deve superare 4 test:

- 1) **Test di frequenza:** Verifica se i diversi elementi appaiono in S approssimativamente lo stesso numero di volte
- 2) **Poker test:** Verifica l'equidistribuzione di sottosequenze di lunghezza arbitraria ma prefissata.
- 3) **Test di autocorrelazione:** Verifica il numero di elementi ripetuti a distanza prefissata
- 4) **Ran test:** Verifica che le sottosequenze massimali contenenti elementi tutti uguali hanno distrib. esponenzialmente negativa ($P(11) > P(111) > P(1111) > P(11111) \dots$)

Generatori che rispettano i test:

- 1) **Generatore lineare:** $x_i = (ax_{i-1} + b) \bmod m$
- x_0 è il seed

↓
Periodo

Proprietà:

- non critto sicuri
- a) $\text{mcd}(b, m) = 1$
 - b) $(a-1)$ divisibile per ogni fattore primo di m
 - c) $(a-1)$ multiplo di 4 se m è divisibile per 4

- 2) **Versione polinomiale:** $x_i = (a_1 x_{i-1}^t + a_2 x_{i-1}^{t-1} + \dots + a_t x_{i-1} + a_{t+1}) \bmod m$

Efficienti ma esistono algoritmi polinomiali che possono fare previsioni degli elementi generati

Un generatore pseudo-casuale per essere **CRIPTOGRAFICAMENTE SICURO** deve superare il **TEST DI PROSSIMO BIT** (che implica anche i 4 precedenti):

TEST DI PROSSIMO BIT:

Un generatore pseudo-casuale supera il TEST di Prossimo Bit se non è possibile fare previsioni sugli elementi della sequenza prima che essi vengano generati, cioè:

È un algoritmo polinomiale in grado di prendere l' $(i+1)$ -esimo bit della sequenza a partire dalla conoscenza degli i bit precedentemente generati, con probabilità $> 1/2$.

Funzioni one-way:

Le funzioni one-way sono tali da poter permettere il calcolo $y = f(x)$ in tempo polinomiale ma $x = f^{-1}(y)$ in tempo esponenziale.

Quindi se si considera

$$S = x \cdot f(x) \cdot f(f(x)) \cdot \dots$$

Ogni bit delle seq. S è facile da calcolare da quello precedente ma difficile dai valori successivi.

Quindi calcolerò la seq. S per un certo numero di parti $s_0, s_1, s_2, \dots, s_i$ ma li comunicherò in ordine inverso al destinatario $s_i, s_{i-1}, s_{i-2}, \dots, s_0$

$$x_{i+1} = f(x_i)$$

Polinomiale

$$x_i = f^{-1}(x_{i+1})$$

Esponenziale

Solo tramite una trapdoor è possibile calcolare $x_i = f^{-1}(x_{i+1})$ in tempo polinomiale.

Generatori Binari Criptograficamente Sicuri

Si usano "PREDICATI HARD CORE" delle funzioni one-way cioè:

$b(x)$ è Predicato HC di una funzione one way $f(x)$ se:

- 1) $b(x)$ è facile da calcolare conoscendo x
- 2) $b(x)$ è difficile da prevedere con probabilità $> 1/2$ conoscendo $f(x)$

Esempio: $f(x) = x^2 \bmod n$ con n non primo

dato $n=77$ e $x=10$ è facile calcolare $y = 10^2 \bmod 77 = 23$

ma è difficile ritrovare al valore $x=10$ dato $y=23$

Quindi dato il predicato $b(x) = \begin{cases} 1 & \text{se } x \text{ pari} \\ 0 & \text{se } x \text{ dispari} \end{cases}$ sulla funzione $f(x)$

Se conosco x posso calcolare $b(x)$ per $f^{-1}(y)$ è esponenziale

Quindi anche $b(x)$ richiede tempo esponenziale se conosco x

Generatore BSS: Lento (elevamento al quadrato di n grandi) e usato per creazioni di chiavi corse

Sia $n = p \cdot q$ dove p e q numeri primi grandi t.c.

a) $p \bmod 4 = 3$ e $q \bmod 4 = 3$

b) $2 \cdot \left\lfloor \frac{p}{4} \right\rfloor + 1$ e $2 \cdot \left\lfloor \frac{q}{4} \right\rfloor + 1$ Siano primi

c) $x_0 = y^2 \bmod n$ per qualche y

 need

Il generatore BSS impiega x_0 come seed, calcola una successione x_i di $m \leq n$ numeri e genera in corrispondenza una sequenza binaria b , secondo la legge:

- $x_i = (x_{i-1})^2 \bmod n$

- $b_i = 1 \Leftrightarrow x_{m-i} \text{ è disponibile}$ es: $b_0 = 1 \Leftrightarrow x_m \text{ è disponibile}$
 $b_1 = 1 \Leftrightarrow x_{m-1} \text{ è disponibile}$

Il generatore parte da x_0 a cui corrisponde b_m e procede al calcolo degli i-uni

x_1, \dots, x_m memorizzando i corrispondenti bit b_{m-1}, \dots, b_0 che comunica successivamente all'esterno in ordine inverso (Prima b_0 , poi b_1, \dots)

Comuniciamo la sequenza al contrario perché ciascun elemento della sequenza deve superare il test del prossimo bit.

Generatori di numeri pseudocasuali basati su cifrari simmetrici

- Cifrario simmetrico
- Chiave
- Si sostituisce il messaggio con un valore iniziale legato al generatore

Esempio (USA il DES)

Algoritmo per produrre numeri pseudocasuali in modo efficiente

$n = \#$ bit delle parole prodotte

$s =$ seme casuale di n bit

$h = \#$ parole da produrre

$k =$ chiave segreta

Generazione (s, m)

{ $d =$ rappresentazione su n bit di data e ora

$y = C(d, k);$
↓
cifrario

$z = s;$

for ($i = 1; i \leq m; i++$) {

$$x_i = C(y \oplus z, k);$$

$$z = C(y \oplus x_i, k);$$

Comunicare x_i all'esterno

}

}

Algoritmi Randomizzati

1) **Monte Carlo**: Algoritmi che generano un risultato probabilmente corretto in un tempo sicuramente breve ($\text{Prob. correttezza} \geq 3/4$)

2) **Las Vegas**: Algoritmi che generano un risultato sicuramente corretto in un tempo probabilmente breve

Algoritmo randomizzato di Miller e Robin (usato per risolvere $P_{\text{primo}}(N)$ che ha costo originale pari a $O(n^6)$)

Sia N dispari (fare pari sarebbe sicuramente comodo)

1) Poniamo $\underbrace{N-1}_{\text{Pari}} = 2^w z$ con: a) z dispari

b) $w = \text{esponente più grande della potenza}$
di 2 che divide $N-1$

OSS: z e w si calcolano in tempo polinomiale rispetto # cifre di N .

2) Sia y un intero arbitrario con $2 \leq y \leq N-1$

Se N è primo deve essere vero che:

P₁) $\text{gcd}(N, y) = 1 \Rightarrow$ Dalla definizione di numero primo

Da proprietà dell'algebra modulare

P₂) $(y^z \bmod N = 1)$ o $(\exists i \text{ con } 0 \leq i \leq w-1 \text{ t.c. } y^{2^i z} \bmod N = -1)$

Lemme 1: Se N è composto, il numero di interi compresi tra 2 e $N-1$ che soddisfano P_1 e P_2 è minore di $N/4$.

Quindi:

a) Se uno dei due predicati è falso $\Rightarrow N$ è certamente composto

b) Se i predicati sono entrambi veri:

N è composto con prob. $< 1/4$, quindi è primo con prob. $> 3/4$

Osservando k volte: la prob. di errore $< \left(\frac{1}{4}\right)^k$

Verifica del certificato: Function Verifica(N, u):

↓
Intervallante se la
sua verifica richiede
tempo polinomiale

if ($P_1 = \text{falso}$) or ($P_2 = \text{falso}$)

then return 1 $\Rightarrow N$ composto

else return 0 $\Rightarrow N$ non composto

(Prob. errore $< 1/4$)

Verifica di primalità di N : Function test-MR(N):

↓

La funzione può indicare

N primo sbagliando con

probabilità $(1/4)^k$

ma se restituisce 0 è
un risultato certo

Value scelto dall'utente

for $i = 1$ to u do

Scegli y a caso tra 2 e $N-1$;

if Verifica(N, u) = 1 then return 0

return 1 \Rightarrow Se N è

probabilmente primo
(Prob. err. $< (1/4)^k$)

↓
Se N non
è primo

Nel calcolo di Verifica(N, u):

- P_1 si calcola in tempo polinomiale con l'algoritmo di Euclide

- P_2 si calcola in tempo polinomiale usando questo metodo:

Metodo: (Algoritmo delle quadrate successive (esponentiazione veloce)) ($N-1 = 2^w z$)

Se si deve calcolare $x = y^z \bmod s$ con y, z, s interi

1) Si decompone z in somme di potenze di 2 cioè

$$z = k_1 \cdot 2^{w_1} + \dots + k_t \cdot 2^{w_t} = \sum_{w=0}^t k_w \cdot 2^w \quad \text{dove } k_w \in \{0,1\}$$

$$t = \lfloor \log_2 z \rfloor = \Theta(\log z)$$

e $w_1 < \dots < w_t$

Esempio: $z = 45 = 2^5 + 2^3 + 2^2 + 2^0 = 32 + 8 + 4 + 1$

2) Si calcolano $y^{2^j} \bmod s$ per $j = w_1, w_2, \dots, w_t$

Esempio: $x = g^{45} \bmod 11$

a) $y^2 \bmod 5 = g^2 \bmod 11 = 4$

a) $y^{16} \bmod 11 = 3^2 \bmod 11 = 9$

b) $y^4 \bmod 5 = 4^2 \bmod 11 = 5$

5) $y^{32} \bmod 11 = g^2 \bmod 11 = 9$

c) $y^8 \bmod 5 = 5^2 \bmod 11 = 3$

3) Si calcola $x = y^{2^{w_1}} \cdot y^{2^{w_2}} \cdots \cdot y^{2^{w_t}} \bmod s = \prod_{w: k_w \neq 0} y^{2^w} \bmod s$

Esempio:

$$y^z \bmod 5 = g^{45} \bmod 11 = g^{32+8+4+1} \bmod 11 = (4 \cdot 3 \cdot 5 \cdot 9) \bmod 11 = 1$$

In totale si eseguono $\Theta(\log_2 z)$ quadrature

$\Theta(\log_2 z)$ moltiplicazioni (costo al più quadratico nel numero di cifre)

Quindi in totale ho un costo polinomiale nella dimensione dei dati

Generazione di numeri binari grandi primi:

Possiamo alternare la generazione casuale di un intero con un test probabilistico di primalità, finché mi incontra un numero dichiarato primo. ($\text{Prob} > 3/4$)

Bisogna però acciunarsi di non esaminare troppi numeri.

Proprietà: Gli numeri primi godono di una proprietà della densità interessante,

Cioè :

IMPORTANTE

In un intorno di N di lunghezza $\log_e N$ cade mediamente un numero primo.

Essendo che $\log_e N$ è proporzionale a N in dimensione

\Rightarrow Numero di prove attese è polinomiale

Algoritmo di Generazione:

Generazione di un numero binario di almeno n bit (scelto dall'utente).

Seme: N con n bit di cui i 2 estremi sono a 1 e gli altri bit sono generati a caso

Function Primo(n):

$N = 1S1$, se S è una sequenza di $n-2$ bit prodotti da un generatore binario

while $\text{test_MR}(N) = 0$ do $N = N + 2$
return N

} Ripetuto $O(n)$ pseudo - casuale
Volte con $n = \log N$

$P \subseteq \text{Classe RP}$ (Random Polynomial) $\subseteq NP$

RP = Classe dei problemi decisionali verificabili in tempo polinomiale randomizzato

Ti $x = \text{Input}$

$y = \text{Certificato probabilistico di } x \text{ se:}$

a) y è di lunghezza al più polinomiale in $|x|$

b) y è estratto perfettamente a caso da un insieme associato a x

$A(x, y) = \text{Algoritmo di verifica polinomiale}$

Un tempo polinomiale attesta che

a) x non possiede la proprietà ($\pi(x)=0)$

oppure

b) x la possiede con prob. $> 1/2$

Cifrari storici

Cifrario di Cesare: Sostituire le lettere del messaggio con 3 posizioni avanti nell'alfabeto



es: $\begin{array}{c} \text{A B C} \\ +2 \downarrow \downarrow \downarrow \\ \text{D E F} \end{array}$

Versione generale: Sia k una chiave con $1 \leq k \leq 25$, ma $\text{pos}(x)$ la posizione nell'alfabeto di una generica lettera.

- la cifratura di x sarà y t.c $\text{pos}(y) = (\text{pos}(x) + k) \bmod 26$
- Per decifrare $\text{pos}(x) = (\text{pos}(y) - k) \bmod 26$

Crittanalisi: Basta conoscere la struttura del cifrario per forzarlo

Proprietà commutativa:

date u_1, \dots, u_n e una seq. di operazioni di cifratura o decifrazione che impiegano u_1, \dots, u_n si ha che l'ordine di tali operazioni può essere permuto senza modificare il citoogramma finale.

OSS: Si nota inoltre che $c(c(s, u_2), u_1) = c(s, u_1 + u_2)$ e

$$d(d(s, u_1), u_2) = d(s, u_1 + u_2)$$

cioè più operazioni possono essere ridotte a una sola, quindi i cifrari composti non indicano maggiore sicurezza

1) Cifrari a sostituzione: Sostituiscono ogni lettera del messaggio in chiaro con una o più lettere dell'alfabeto

a) Monoalfabetica: A una lettera del mex. corrisponde sempre la stessa lettera del crittogramma

- Esempio cifrario affine:

Cifratura: $\text{pos}(y) = (a \cdot \text{pos}(x) + b) \bmod 26$ con a e b parametri

Decifratura: $\text{pos}(x) = a^{-1} \cdot (\text{pos}(y) - b) \bmod 26$

Vincoli:

- $K = (a, b)$ chiavi:

Totale chiavi = $12 \times 26 = 312 - 1 = 311$
 ↓
 Valori di a possibili
 ↓
 Valori di b possibili

- $\text{mcd}(a, 26) = 1 \Rightarrow$ l'inverso di a esiste
 ed è unico $\Leftrightarrow \text{Mcd}(a, m) = 1$

Scegliere la chiave $(1, 0)$ che non cambia il messaggio

Se la segretezza dipende dalla chiave segreta \Rightarrow la possibile scelta della chiave deve essere ampia

- Esempio cifrario completo

Prendo una permutazione arbitraria dell'alfabeto come chiave

lettera in chiaro di posizione i
 ↓
 lettera di posizione i nella permutazione

ABCDEFGHIJKLMNOPQRSTUVWXYZ
 SDTKBJOHZRZCUNYE PXVFWAGQILM

testo: BOBSTAIAATTENTOAeve

messaggio cifrato: DEDFWSRSWWBYWESBGB

la chiave è di 26 lettere \Rightarrow # chiavi = $26! - 1$

Crittoanalisi: Non c'è niente perché si può analizzare la frequenza dei caratteri.

Per ogni lettera ripetuta nel mex il # chiav: da testare diventa:

$$\frac{26!}{26 - \# \text{ lettere non ripetute}}$$

Calcolo inverso

$$a = x^{-1} \bmod n$$

1) Trovo: fattori primi di n
 $f_1, f_2, f_3, \dots, f_r$

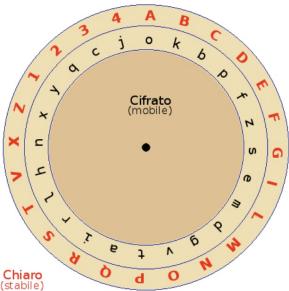
$$2) \text{Calcolo } \phi(n) = n \left(1 - \frac{1}{f_1}\right) \cdot \dots \cdot \left(1 - \frac{1}{f_r}\right)$$

$$3) a = x^{\phi(n)-1}$$

b) Polialfabetica: Una stessa lettera in punti diversi del menafgio

ammette un insieme di lettere sostitutive possibili.

1) Cifrario di Alberti: Mittente e destinatario possedevano un disco cifrante uguale composto da 2 dischi contenenti ciascuno 1 alfabeto diverso la corrispondenza veniva fatta tramite notazione di ens.



a) Disco esterno: Un insieme di caratteri limitato in ordine alfabetico seguito da alcuni numeri

per far variare le chiavi di cifratura

b) Disco interno: Un insieme di caratteri ampio senza numeri → Usato per costituire il codice greco

Indice fino

A	B	C	D	E	F	G	H	I	L	M	N	O	P	Q	R	S	T	U	V	Z	1	2	3	4	5
S	D	T	K	B	J	O	H	R	Z	C	U	N	Y	E	P	X	V	F	W	A	Q	I	L	M	

Chiave: A-S

Messaggio: NON FIDARTI DI EVE

m = NON FIDA 2 R T I D I E V E

c = UNUJRKSQ



qui la chiave diventa A-Q

Perché in corrispondenza della Q c'è il z (carattere speciale) che indica il cambio

Indice Mobile

A	B	C	D	E	F	G	H	I	L	M	N	O	P	Q	R	S	T	U	V	Z	1	2	3	4	5
E	Q	H	C	W	L	M	V	P	D	N	X	A	O	G	Y	I	B	Z	R	J	T	S	K	U	

m: I L D 2 E L P F I N O
c: P D C S W D O O I R J

Il numero 2, ottenuto decifrando S, indica che dopo due caratteri la chiave verrà cambiata

O, decifrato in P, indica la nuova chiave A-P

A	B	C	D	E	F	G	H	I	L	M	N	O	P	Q	R	S	T	U	V	Z	1	2	3	4	5
P	D	N	X	A	O	G	Y	I	B	Z	R	J	T	S	K	U	F	E	Q	H	C	W	L	M	

A	B	C	D	E	F	G	H	I	L	M	N	O	P	Q	R	S	T	U	V	Z	1	2	3	4	5
Q	I	L	M	S	D	T	K	B	J	O	H	R	Z	C	U	N	Y	E	P	X	V	W	A	Q	I

Chiave: A-Q

Messaggio: NON FIDARTI DI EVE

m = NON FIDA 2 R T I D I E V E

c = UNUJRKSQUYBMBSPS



qui la chiave diventa A-Q

OSS: Se si cambia → perdi la chiave diventa

difficile da attaccare Perché rende

inutili gli attacchi sulla freq. di

caratteri

2) Ci proviamo di Vigevano:

la chiave è una parola segreta che serve per cifrare il messaggio

Se è più corta del menaggio, viene ripetuta più volte (lettera, messaggio)

Esempio:

$\mathcal{K} = \text{ABRA}$

$\downarrow \downarrow \downarrow \downarrow$ = Di quanto sono state spostate le lettere nell'alfabeto
 $0 \ 1 \ 21 \ 0$

H: I L D e L F I Z O

clique

κ : A B R A A B R A A

C : I M U E L G Z N O

8 +0 9+1 acc ...

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Z
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Z
B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Z	A
C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Z	A	B
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Z	A	B	C
E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Z	A	B	C	D
F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Z	A	B	C	D	E
G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Z	A	B	C	D	E	F
H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Z	A	B	C	D	E	F	G
I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Z	A	B	C	D	E	F	G	H
J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Z	A	B	C	D	E	F	G	H	I
K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Z	A	B	C	D	E	F	G	H	I	J
L	M	N	O	P	Q	R	S	T	U	V	W	X	Z	A	B	C	D	E	F	G	H	I	J	K
M	N	O	P	Q	R	S	T	U	V	W	X	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	O	P	Q	R	S	T	U	V	W	X	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	P	Q	R	S	T	U	V	W	X	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	Q	R	S	T	U	V	W	X	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	R	S	T	U	V	W	X	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	S	T	U	V	W	X	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	T	U	V	W	X	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	U	V	W	X	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	V	W	X	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X

Incrocio tra l'elenco delle chiavi m_i e della chiave $n_i = c_i$

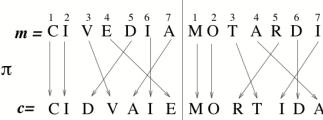
Critto analisi: Per essere efficiente la chiave deve essere lunga perché

senso rischio di trasformare le stesse lettere nella medesima cifratura e la distanza tra $|P_2 - P_1| = |K|$:

Se usiamo una chiave lunga come un testo, casuale e non riutilizzabile, il cipario diviene inattaccabile (One Time Pad).

2) Cifrari a trasposizione : Permutazione dell'ordine delle lettere e possibile inserimento di alte ignrate nelle deciphering

1) Cifrario a permutazione semplice: \Rightarrow



$$= \left\{ \begin{matrix} h = 7 \\ n = \end{matrix} \right\} \{ 12, 53, 76, 9 \}$$

Finato un intero h e una permutazione π degli interi $\{1, 2, 3, \dots, h\}$ che costituisce

la chiave, il processo di cifratura consiste nel dividere il messaggio in blocchi.

di h lettere ciascuno, e permutare le lettere di ciascun blocco in accordo con π .

Se $|m|$ non è divisibile per $h \Rightarrow$ aggiungo lettere casuali che contrib. nella permutazione

Questo cifrario ha tante chiavi possibili quante le permutazioni di h elementi - 1

Cioè $\#K = h! \quad - 1$

↓
permutazione identica

Più cresce h , più sono difficili gli attacchi; però è anche più difficile ricordarli :-)

2) Cifrario a permutazione di colonne:

La chiave $K = \langle c, r, \pi \rangle$ dove

- c e r denotano il numero di colonne e righe in una Tabella di lavoro T
- π rappresenta la permutazione degli indici $\{1, 2, \dots, c\}$

Il messaggio si viene decomposto in blocchi: m_1, m_2, \dots, m_n da $c \times r$ caratteri ciascuno

I caratteri sono distribuiti nella Tabella e poi vengono permutate le colonne

es: $c = 6, r = 3$ e $\pi = \{215346\}$. $m \Rightarrow$ NON SONO IL COLPEVOLI

N	O	N	S	O	N
O	I	O	C	O	L
P	E	V	O	L	E

Tabella T

⇒

O	N	O	N	S	N
I	O	O	O	C	L
E	P	L	V	O	E

Tabella T permutata

→ CRIPTOGRAFIA = OIENOPOOLNOVSCONLE.

Si fa questo passaggio per ogni blocco m_1, \dots, m_n del messaggio

Chiavi = Numero esponenziale nella lunghezza del messaggio non tenendoci vincoli per r e c

3) Cifrario a griglia: Deriva dal cifrario di Richelieu il quale usava una scheda perforata e l'indicazione di una pagina del libro per poter sovrapporre la scheda sulle pagine e trovare le lettere del messaggio.

la variante courante nell' avere:

- Chiave segreta come griglia quadrata di dim. q con q pari
 - $S = q^2/4$ celle della griglia trasparenti, le altre opache
 - Scrivere: primi S caratteri del menaggio nelle posizioni corrispondenti alle celle trasparenti

La griglia viene rotata di 90° in senso orario, e mi ripete per ogni rotazione l'operazione di scrittura di tre succ. gruppi di lettere.

$$q = 6 \quad s = 9$$

m = L'ASSASSINO È ARCHIMEDES TARRINGTON

Numbers diar:

$G = \mathbb{U}^S$ griglie cioè chiavi segrete disponibili

Rot. 1	Rot. 2
L A S S S I S N	O A R C H I M E
D E T A R R I N	G T O * * * *
Rot. 3	Rot. 4

*: caratteri a caso

D	L	G	A	T	O
E	O	S	S	S	E
N	*	T	*	A	A
A	*	R	S	C	H
*	S	R	R	*	I
I	M	I	E	N	N

CRITTOGRAMMA

Vincoli:

- 1) la chiave deve essere scelta in modo che le positioni corrispondenti alle celle trasportanti non si sovrappongano mai nelle 4 notazioni.
 - 2) Se la lunghezza del messaggio è minore di 4s, le positioni della pagina f rimane vuote e riempiono con caratteri a caso!
 - 3) Se la lunghezza del messaggio è maggiore di 4s, il messaggio viene decomposto in blocchi di 4s caratteri ciascuno e ogni blocco è cifrato indipendentemente dagli altri.
 - 4) La decifrazione di f è eseguita sovrapponendomi quattro volte la chiave

la sicurezza di un cifrario è legata alla dimensione dello spazio delle chiavi che deve essere sufficientemente ampio da evitare attacchi esaurienti (provare tutte le chiavi).

Però i vecchi cifrari sono stati attaccati in un altro modo cioè l'attacco statistico di tipo cipher text (il crittanalista ha a disp. solo il citoigramma)

CRIPTOANALISI STATISTICA:

INFORMAZIONI NOTE DAL CRIPTOANALISTA:

- 1) Il metodo impiegato per cifratura/decifratura.
- 2) Linguaggio naturale in cui è stato scritto il messaggio.
- 3) Messaggio abbastanza lungo per ricevere alcuni dati statistici sui caratteri che compongono il citoigramma.

ATTACCO:

Si basa sull'impiego di un insieme di tavole note cioè la frequenza in ogni lingua con cui appaiono in media le varie lettere dell'alfabeto.

Dati simili sono note anche per sequenze di lettere come:

- bigrammi = gruppi di 2 lettere consecutive
- trigrammi = gruppi di 3 lettere consecutive

e così via \Rightarrow q-grammi = gruppi di q lettere consecutive

1) Attacco a sostituzione monoaffonica:

y nel citoigramma corrisponde a x nel messaggio quindi

$$\text{Frequenza}(y) \approx \text{Frequenza}(x)$$

Si confrontano le frequenze delle lettere e si provano permutazioni per lettere con seq. assai pronome.

È sufficiente individuare una corrispondenza (x,y) e impostare un sist. di due eq. in a e b incognite

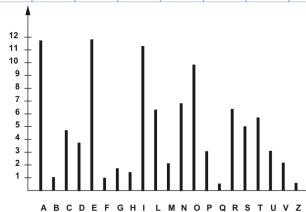


Figura 5.8: La frequenza in percentuale delle lettere nell'italiano.

2) Cifrario a sostituzione polialfabetica

Cifrario di Vigenère:

Basta scoprire il valore di h per scomporre il ciptogramma e continuare con la decifrizzazione con il metodo monoalfabetico.

Scoprire il valore di h :

Il menaggio contiene vicinamente gruppi di lettere adiacenti ripetuti più volte, perché sottoseguenze allineate con la stessa porzione della chiave sono trasl. nel ciptogramma in sottoseguenze identiche.

Si cercano nel ciptogramma coppie di posizioni p_1, p_2 in cui iniziano sottoseq. identiche.

la distanza tra queste posizioni $d = p_2 - p_1$ è probabilmente uguale alla lunghezza h della chiave o un suo multiplo.

Cifrario di Alberti: Immune da questi attacchi se la chiave viene cambiata spesso evitando pattern ripetitivi perché mantenendole a lungo metterebbe a rischio il cifrario perché in quell'intervallo la srt. è monoalfabetica

3) Cifrario a trasposizione:

Non si basano gli attacchi sulla freq. delle lettere ma si studiano i q-grammi

1) Cifrario a permutazione semplice:

Se si conosce la lunghezza h :

1) Si divide il ciptogramma in porzioni di lunghezza h

2) In ciascuna si cercano i q-grammi più diffusi nel linguaggio

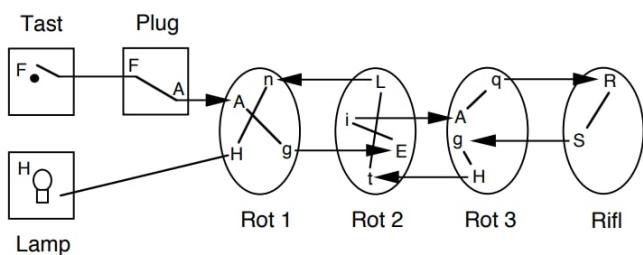
3) Se un gruppo deriva da un q-gramma, si trova parte della permutazione

Conclusione:

- 1) Nei cifrari a trasposizione l'istogramma delle frequenze coincide approssimativamente con quello proprio del linguaggio
- 2) Nei cifrari a rot. monoalfabetica i due istogrammi coincidono a meno di una permutazione delle lettere
- 3) Nei cifrari a rot. polialfabetica, l'istogramma del citoogramma è assai più appiattito di quello del linguaggio (le freq. delle lettere variano assai meno tra loro)

Macchina Enigma

Estensione elettronica del cifrario di Alberti usato durante la seconda guerra mondiale.



Come funzionava prima delle modifiche

- I rotori non mantenevano la stessa posizione reciproca durante la cifratura
- Per ogni lettera battuta sulla tastiera
 - Il primo rotore avanzava di un passo
 - Dopo 26 passi il rotore era tornato sulla posizione iniziale, e avanzava di un passo il secondo rotore
 - Dopo la rotazione completa del secondo rotore, avanzava di un passo il terzo rotore
- La corrispondenza tra caratteri cambiava ad ogni passo (la chiave cambia a ogni passo)

Numero di permutazioni

- 26 con le rotazioni del primo rotore rispetto al secondo
- 26 con le rotazioni del secondo rotore rispetto al terzo
- 26 con le rotazioni del terzo rotore rispetto al riflettore
- $26 \times 26 \times 26 = 17576$ chiavi diverse

Problemi:

- Rotori immutabili
- 26^3 permutazioni sono sempre le stesse, applicate nello stesso ordine
- Note a tutti i proprietari di una macchina Enigma
 - Alberti aveva previsto, per ogni coppia di utenti, una coppia di dischi diversa da tutte le altre

Modifiche:

- Possibilità di permutare tra loro i tre rotori:

permutazioni: $(3!) \times 26^3 > 10^5$

- Aggiunta del **plugboard** tra tastiera e primo rotore

Consente di scambiare tra loro i caratteri di 6 coppie scelte arbitrariamente in ogni trasmissione

- Ogni cablaggio è descritto da una sequenza di 12 caratteri (le 6 coppie da scambiare)

Combinazioni possibili: $\binom{26}{12} \sim 10^7$

- Ogni gruppo di 12 caratteri si può presentare in $12!$ permutazioni diverse, ma non tutte producono effetti diversi:

AB CD EF GH IJ KL

CD AB EF GH IJ KL

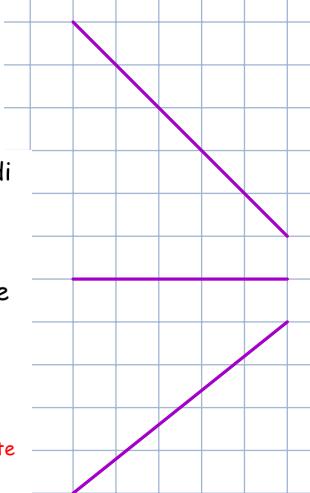
producono lo stesso effetto, e con queste anche tutte le 6! permutazioni delle 6 coppie

Infine si devono considerare i possibili scambi tra gli elementi delle coppie, che producono lo stesso effetto

AB CD EF GH IJ KL

BA CD EF HG IJ KL

Dobbiamo dividere per un ulteriore fattore $2^6 = 64$



Il numero di chiavi dei rotori ($> 10^5$) si moltiplica per un fattore

$$\binom{26}{12} \frac{12!}{6! \cdot 64} > 10^{11}$$

per un totale di più di 10^{16} chiavi

(10 milioni di miliardi di combinazioni possibili)

Cifrari Perfetti

Sono cifrari in grado di nascondere l'informazione con certezza assoluta ma a un costo così alto da mettere in dubbio la loro praticità.

Vengono usati solo per comuni cabiui sporadiche di massima segretezza.

Siano: M_{sg} = Spazio dei messaggi

C_{itto} = Spazio dei cattogrammi

$P(M=m | C=c)$ = Probabilità che sia in transitato il messaggio $m \in M_{\text{sg}}$ sapendo che il cattogramma è $c \in C_{\text{itto}}$

$P(M=m)$ = Probabilità che il messaggio $m \in M_{\text{sg}}$

Definizione: Un cifrario è perfetto se $\forall m \in M_{\text{sg}}$ e $\forall c \in C_{\text{itto}}$ vale la relazione

$$P(M=m | C=c) = P(M=m)$$

Finato un messaggio m , la probabilità che esso sia in transitò è la stessa qualunque sia il crittogramma C in transitò.

Cioè la conoscenza del crittoanalista non cambia dopo aver visto transitare il crittogramma C .

Se le due prob. non fossero uguali implicherebbe che il crittoanalista ottiene informazioni nuove dal crittogramma.

Teorema di Shannon

In un cifrario perfetto il numero delle chiavi deve essere maggiore o uguale al numero dei messaggi possibili, cioè:

$$\# \text{ chiavi} \geq \# \{ \text{Msg} \}$$

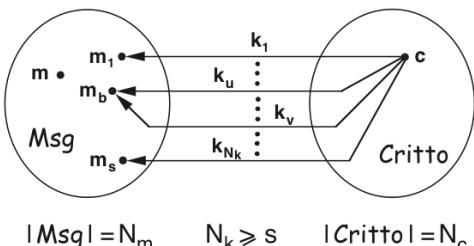
Dim:

Siano $N_m = \#$ dei messaggi possibili, cioè $m \in \text{Msg}, P(M=m) > 0$

$N_k = \#$ delle chiavi;

Poniamo per assurdo $N_m > N_k$

A un crittogramma C , con $P(C=c) > 0$, corrispondono $s \leq N_k$ messaggi (non necessariamente distinti) ottenuti decifrando C con tutte le chiavi.



\Rightarrow Non tutti i messaggi sono ottenibili da C , cioè non hanno tutti la stessa prob., quindi il crittoanalista ottiene info in più!

Poiché $N_m > N_k > s$, esiste almeno un messaggio m con $P(M=m) > 0$ non ottenibile da C .

Questo implica $P(M=m | C=c) = 0 \neq P(M=m)$, ovvero per

$N_m > N_k$ il cifrario non è perfetto.

OSS: Questo dimostra che i cifrari perfetti sono molto costosi perché richiedendo più chiavi che mex ci sarà difficoltà nella gestione e scambio segreto di esse.

Cifrario one-Time Pad

È un cifrario perfetto inventato da Mauborgne e Vauven nel 1917.

Il suo costo alto è legato all'impiego di una sequenza casuale di bit di lunghezza non limitata a priori la quale costituisce la chiave segreta.

Per utilizzarne il cifrario dobbiamo ammettere che i messaggi, chiavi e crittogrammi siano sequenze binarie arbitrariamente lunghe.

Le funzioni di cifratura e di decifrazione eseguono trasformazioni tra sequenze di bit preservando la loro lunghezza cioè $\text{length}(\text{mess}) = \text{length}(\text{critto})$.
Le funzioni di cifratura e di decifrazione sono basate sull'operazione

binaria XOR definita così'

n_1	n_2	out
0	0	0
0	1	1
1	0	1
1	1	0

Lo XOR è commutativo e gode delle seguenti proprietà:

$$x \oplus y \oplus y = x$$

$\downarrow \quad \downarrow$
 $x \oplus y \quad x \oplus y$

1) Generazione chiave segreta:

Si costruisce una sequenza $K = k_1, k_2, \dots, k_n$ di bit, di lunghezza \geq di quella del messaggio. Ogni k_i di K è scelto perfettamente a caso tra 0 e 1.

2) Cifratura:

Se il messaggio da trasmettere è di m bit $m = m_1, \dots, m_m$ il cifrogramma è $c = c_1, \dots, c_m$ con $c_i = m_i \oplus k_i$ per $i = 1, \dots, m$

3) Decifratura:

Prenzi gli m bit iniziali della chiave k_1, \dots, k_m e $c = c_1, \dots, c_m$.

Il messaggio è $m = m_1, \dots, m_m$ con $m_i = c_i \oplus k_i$ per $i = 1, \dots, m$

OSS: le chiavi non sono riutilizzabili perché:

$$c' \oplus c'' = (m' \oplus k) \oplus (m'' \oplus k) = m' \oplus m'' \oplus \underbrace{k \oplus k}_{\text{O}} = m' \oplus m''$$

Siano vere le seguenti ipotesi:

- 1) Tutti i messaggi hanno la stessa lunghezza n (messaggi più lunghi vengono spezzati in più sequenze mentre quelle più corte si riempiono con bit inutili)
- 2) Tutte le sequenze di n bit sono messaggi possibili

Allora:

Teorema: Sotto le ipotesi 1 e 2, utilizzando una chiave scelta perfettamente a caso per ogni messaggio ($P(k_i = k) = \frac{1}{2^n}$) il cifrario One-Time Pad è perfetto e impiega un numero minimo di chiavi:

$$|\text{key}| = |\text{msg}| = 2^n$$

Dim:

Bisogna verificare che valga la relazione: $P(M = m | C = c) = P(M = m)$

Applicando la def. di prob. condizionale ottengo:

$$P(M = m | C = c) = \frac{P(M = m \wedge C = c)}{P(C = c)}$$

Per la def. di XOR si ha: chiavi diverse \Rightarrow critogrammi diversi

e ogni chiave può essere generata con prob. $(1/2)^n$

Quindi anche $P(C=c) = (1/2)^n \quad \forall c \in \{ \text{critto} \}$ quindi gli eventi

$\{M=m\}$ e $\{C=c\}$ sono indipendenti

Ottieniamo che:

$$P(M=m \wedge C=c) = P(M=m) \times P(C=c)$$

Sostituendo questa eq. a quella della prob. condizionale ottieniamo

$$P(M=m | C=c) = P(M=m)$$

Quindi il cifrario One-Time Pad è perfetto.

Generazione della chiave

Il mittente e destinatario devono accordarsi preventivamente su una sequenza n^* di bit casuali molto lunga, da condividere man mano che procede la comunicazione.

Il mittente esegue la cifratura con i primi n bit della sequenza n^* ed uguale viene fatto per la decifrazione da parte del destinatario. Poi viene scartata quella parte di chiave.

Problema: Generare molti bit casuali per garantire sia che la chiave sia scelta a caso, sia di non usare gli stessi bit per la cifratura /decifratura di messaggi diversi.

Si utilizzano generatori pseudo-casuali apportando 2 scelte:

- 1) Generatore pubblicamente noto e il quale costituisce l'informazione segreta del cifrario.

Necessita quindi che il segnale sia molto lungo e che
sia numeroso per evitare un attacco brute force.

Il mittente e destinatario devono accordarsi su un gran
numero di segnali lunghi, messaggio per messaggio.

2) Il generatore e i 2 segnali costituiscono la chiave segreta.

Essendo i generatori pseudo-casuali programmi brevi
può essere ammesso lo scambio in secretezza.

Normalmente il generatore privato deve essere definito
dal mittente e destinatario e non un generatore già noto

Osservazione:

Se rinnoviamo l'ipotesi che le seq. di n bit devono essere meno
possibili ottenendo che saranno α^n (per la lingua inglese $\alpha \approx 1.1$) t.c.

$$\alpha^n \ll 2^n$$

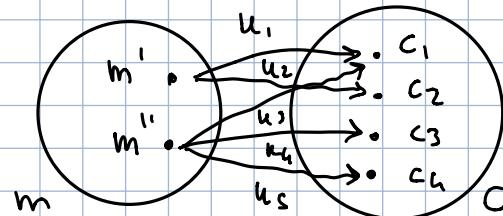
Sia $N_m = \alpha^n$ e N_k abbiamo che:

$$N_k \geq N_m \quad \text{cioè} \quad N_k \geq \alpha^n$$

Sia la chiave una seq. di t bit, ottenendo:

$$2^t \geq \alpha^n \Rightarrow t \geq \log_2 \alpha^n = n \cdot \log_2 \alpha = 0.12 \cdot n$$

E' opportuno tuttavia che coppie diverse $\langle m, k \rangle$ producano lo
stesso critogramma c



4) che vuol dire che ~~#~~ coppie(m, k) $\gg \#$ cifogrammi

$$d^n \cdot 2^t \gg z^n$$



$$t \gg 0.88 n$$

Cioè, pur emendo che non tutte le seq. siano possibili con lunghezza n , la chiave deve rimanere prossima alla lunghezza di n per mantenere la sicurezza del cifrario.

Le cifrari simmetrici standard

proposti da Shannon

Criteri importanti ^V alla base dei cifrari a chiave segreta o simmetrici:

1) Diffusione: Alterare la struttura del testo in chiaro "spargendone" i caratteri su tutto il testo cifrato.

cifrari a trasposizione

Si ottiene permutando i caratteri (o bit) del menaggio.

Così facendo la frequenza delle singole lettere rimane invariata ma si perde l'informazione sulla frequenza dei q-gramm.

Si può anche combinare tra loro i caratteri del menaggio in modo che ciascun carattere del cifrogramma venga a dipendere da molti di essi, così facendo si perde l'informazione sia sulla freq. delle singole lettere, che sulla freq. dei q-grammi.

2) Confusione: Combinare in modo complesso il menaggio e la chiave per

non permettere al crittoanalista di separare queste due sequenze mediante un'analisi del citoigramma.

Si ottiene nei cifrari a sostituzione Polialfabetica e One-Time Pad

Soluzione adottata per poter combinare i due criteri:

Eseguire trasformazioni che dipendono **non linearmente** dalle sequenze in ingresso.

Data Encryption Standard (DES)

Il DES è un cifrario simmetrico di uso generale che presenta la seguente struttura:

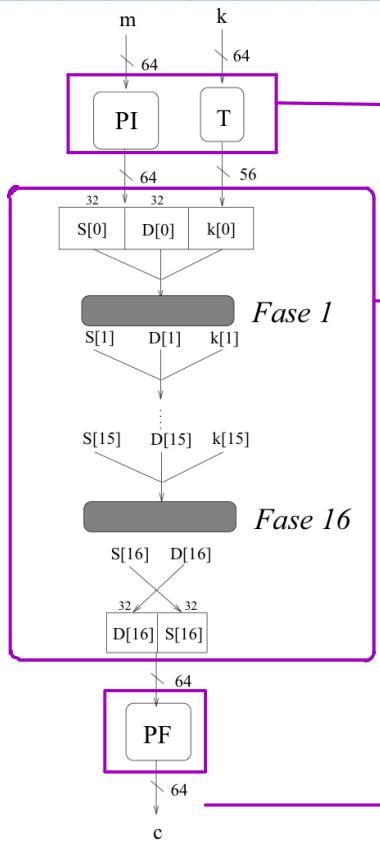
- Il messaggio è suddiviso a blocchi (64 bit). Ogni blocco viene cifrato e decifrato indipendentemente dagli altri.
- Cifrazione e Decifrazione procedono attraverso $r = 16$ fasi successive (round) in cui si ripetono le stesse operazioni.
- La chiave K è composta da 8 byte, ogni byte ha 7 bit scelti e 1 bit di parità. In totale quindi sono 56 bit scelti e 8 di parità su 64 bit totali (8 byte)
- Della chiave K vengono create n sottochiavi $K[0], K[1], \dots, K[n-1]$ usate una per fase

Come funziona il DES

m = blocco del messaggio

c = corrispondente blocco del citoigramma

K = chiave segreta, con : bit di parità



Fasi:

1) Permutazioni iniziali

Permutazioni iniziali. PI permuta i 64 bit del messaggio in chiaro. T depura la chiave dai bit di parità e permuta gli altri 56 bit per generare la prima sottochiave $k[0]$. Le operazioni sono specificate completamente nelle tabelle di figura 7.2.

2) Trasformazione Centrale

Trasformazione centrale. La sequenza permutata da PI viene decomposta in un blocco sinistro $S[0]$ e un blocco destro $D[0]$ di 32 bit ciascuno. Su questi due blocchi si eseguono, nelle successive fasi del DES , sedici trasformazioni strutturalmente uguali, ciascuna con una diversa sottochiave $k[i]$ di 56 bit derivata dalla chiave iniziale k . La i -esima fase riceve in ingresso tre blocchi $S[i-1], D[i-1], k[i-1]$ rispettivamente di 32, 32 e 56 bit, e produce in uscita tre nuovi blocchi $S[i], D[i], k[i]$ che costituiscono l'ingresso alla fase successiva. Dopo l'ultima fase i due blocchi $S[16]$ e $D[16]$ vengono scambiati e concatenati tra loro in un unico blocco di 64 bit da cui si costruirà il crittogramma finale.

3) Permutazione Finale

Permutazione finale. PF genera la permutazione inversa di PI (figura 7.2) rimescolando nuovamente i bit del crittogramma.¹

OSS:

la Decifrazione consiste nel ripetere il processo invertendo l'ordine delle chiavi.

Per ogni fase $i = 1 \dots 16$ effettuo le seguenti operazioni:

$$1) \quad s[i] = D[i-1]$$

$$2) \quad D[i] = s[i-1] \oplus f(D[i-1], k[i-1])$$

↓
Funzione non lineare (S-box)

legenda

s = parte sinistra

D = parte destra

Quiudi:

- Con le permutazioni e scambi effettuo la diffusione
- Con le s-box effettuo la confusione

Permutazioni

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	⑦
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	②

Permutazione PI

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	①	41	9	49	17	57	25

Pos. 60

Pos. 50

Permutazione PF

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	52	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

Trasposizione T

Le tabelle vanno lette per righe. La permutazione PI riordina i bit del messaggio $m = m_1, m_2, \dots, m_{64}$ come $m_{58}, m_{50}, \dots, m_7$ (porta in posizione 40 il bit in posizione 1)

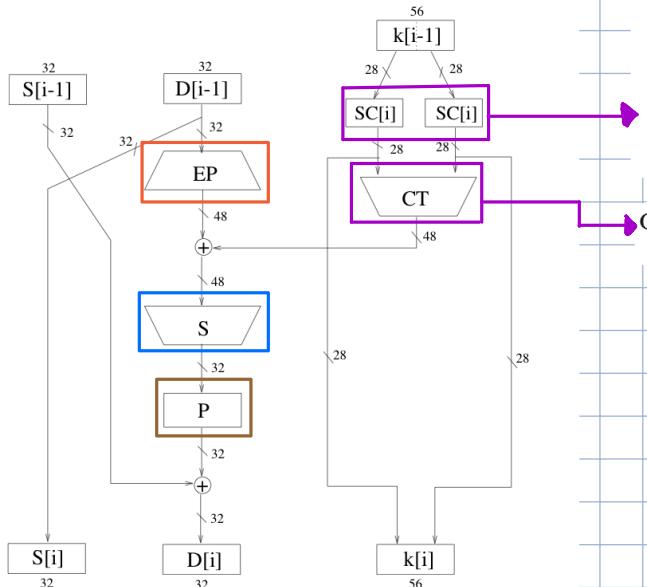
PF è la permutazione inversa di PI, cioè riporta in posizione 1 il bit in posizione 40, etc.

T provvede anche a scartare dalla chiave $k = k_1, k_2, \dots, k_{64}$ i bit per il controllo di parità $k_8, k_{16}, \dots, k_{64}$, generando una sequenza di 56 bit che costituisce la prima sottochiave $k[0]$.

le tabelle dicono: un posizone + deve andare il bit s8-erimo, pos z il bit s0 erimo

ecc... $m = m_1, m_2, \dots, m_{64}$ diventa $m_{58}, m_{50}, \dots, m_7$

Fare i-erima del DES



Shift ciclico $SC[i]$. La sottochiave $k[i-1]$ di 56 bit, ricevuta dalla fase precedente, viene suddivisa in due metà di 28 bit ciascuna. Su ognuna di queste la funzione $SC[i]$ esegue uno shift ciclico verso sinistra di un numero di posizioni definito come segue: $SC[i] = 1$ per $i = 1, 2, 9, 16$, $SC[i] = 2$ altrimenti. Le due parti così traslate vengono concatenate in un unico blocco di 56 bit che costituisce la sottochiave $k[i]$ per la fase successiva.

Compressione e trasposizione CT . Il blocco di 56 bit prodotto dall'operazione precedente viene ulteriormente elaborato per produrre un nuovo blocco di 48 bit utilizzato nel processo di cifratura dei blocchi $S[i-1]$ e $D[i-1]$. La funzione CT esegue una permutazione del blocco e una selezione di 48 bit da esso come indicato nella figura 7.4. La combinazione delle funzioni $SC[i]$ e CT garantisce che in ogni fase venga estratto dalla sottochiave un diverso sottoinsieme di bit per la cifratura. Si calcola che nella cifratura ogni bit della chiave originale k partecipi in media a quattordici fasi.

14	17	11	24	01	05
03	28	15	06	21	10
23	19	12	04	26	08
16	07	27	20	13	02
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Espansione e permutazione EP. Il blocco $D[i-1]$ di 32 bit generato nella fase precedente viene espanso a 48 bit duplicando sedici bit in ingresso e spostandone altri (figura 7.5), per ottenere un blocco della stessa dimensione di quello estratto dalla sottochiave e poter eseguire lo XOR tra i due. Si incrementa così la dipendenza tra ingresso e uscita della fase, poiché i bit duplicati influenzano due delle sostituzioni operate dalla funzione S del blocco seguente.

la duplicazione aumenta la diffusione.

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

Figura 7.5: La funzione EP. Sedici bit di ingresso vengono duplicati: per esempio il bit 32 è copiato nelle posizioni 1 e 47 dell'uscita.

Permutazione P. È una permutazione di 32 bit che genera il blocco finale $D[i]$ (figura 7.7).

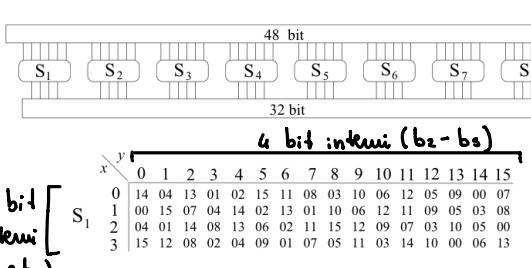
16	07	20	21	29	12	28	17
01	15	23	26	05	18	31	10
02	08	24	14	32	27	03	09
19	13	30	06	22	11	04	25

Figura 7.7: La permutazione P.

"le bit in pos 16
delle sq. dopo l's-box
va in pos 1"

Sostituzione S (o S-box). Questa funzione, progettata con molta cura dalla IBM e modificata con altrettanta cura dalla NSA, costituisce la parte cruciale e "magica" su cui si basa la sicurezza del cifrario, come spiegheremo nel seguito. Essa consta di otto sottofunzioni combinatorie S_1, S_2, \dots, S_8 (figura 7.6). L'ingresso

di 48 bit viene decomposto in otto blocchi B_1, B_2, \dots, B_8 di 6 bit ciascuno che costituiscono l'ingresso alle sottofunzioni di pari indice. Sia $B_j = b_1 b_2 b_3 b_4 b_5 b_6$. Questi bit vengono divisi in due gruppi $b_1 b_6$ e $b_2 b_3 b_4 b_5$ che definiscono due numeri x, y , con $0 \leq x \leq 3$ e $0 \leq y \leq 15$, utilizzati per accedere alla cella di riga x e colonna y in una tabella che definisce la sottofunzione S_j . Il numero ivi contenuto è compreso tra 0 e 15, ed è quindi rappresentato con 4 bit che costituiscono l'uscita di S_j realizzando una compressione da 6 a 4 bit. Complessivamente gli otto blocchi generano una sequenza di 32 bit. I blocchi EP e S sono studiati in modo che tutti i bit di $D[i-1]$ influenzino l'uscita di S , senza che non sarebbe poi possibile decifrare il messaggio.

Figura 7.6: La struttura generale della S-box (in alto), e la tabella 4×16 che definisce la sottofunzione S_1 . Le sottofunzioni S_2, S_3, \dots, S_8 sono definite in modo simile: per esse rimandiamo ai testi specialistici.

Tutte le funzioni, eccetto la s-box, sono lineari se riferite all'operazione all'operazione di XOR, cioè vale $f(x) \oplus f(y) = f(x \oplus y)$

Per l's-box non vale invece! Se comprendesse solo funzioni lineari l'attacco sarebbe molto più facile

OSS:

Se $c^* = (\bar{m}, \bar{u})$ alla s-box otterrei comunque $m_i + u_i$ questo perché:

$$\bar{m}_i \oplus \bar{u}_i = (1 \oplus m_i) \oplus (1 \oplus u_i) = m_i \oplus u_i \oplus 1 \oplus 1 = m_i \oplus u_i$$

Attacchi al DES

Per poter eseguire un attacco esauriente bisognerebbe provare tutte le 2^{56} chiavi disponibili.

Però è possibile ridurre lo spazio delle chiavi notando la seguente relazione:

$$c_{\text{DES}}(m, u) = c \Rightarrow c_{\text{DES}}(\bar{m}, \bar{u}) = \bar{c}$$

dove la barra indica la complementazione bit a bit di una seq. binaria. Questo perché il blocco s-box ha ingresso e uscita identiche per le coppie complementari (m, u) e (\bar{m}, \bar{u}) . Quindi il blocco XOR ha un ingresso complementato $s[i-1]$ e uno diretto proveniente dal blocco P, e quindi genera l'uscita complementata.

Sapendo ciò è possibile fare un attacco di tipo chosen plain text nel modo seguente:

Date 2 coppie (m, c_1) e (\bar{m}, c_2) scelta una chiave λ calcola $c_{\text{DES}}(m, \lambda)$.

- Se il risultato è c_1 prob. k è la chiave segreta quindi si prova k su altre copie (m, c) .
- Se invece $c_{\text{DES}}(m, k) = \bar{c}_2$ prob. \bar{k} è la chiave segreta perché genererebbe (\bar{m}_2, c_2) e quindi si prova su altre copie.
- Se la chiave non è k o \bar{k} si prova con un'altra.

Quindi lo spazio delle chiavi si riduce a 2^{ss} , quindi posso dimezzare lo spazio delle chiavi.

Altri attacchi:

- Cifra analisi differentiale (chosen plain-text) impiega 2^{47} messaggi in chiave (1990)
- Cifra analisi lineare (known plain-text) permette di inferire alcuni bit e usare brute-force sui restanti 2^{43} (1993)
Usa 2^{43} coppie $\langle m, c \rangle$ non scelte
- Macchina RIVERA rompe il cifrario in meno di 24 ore (2008)

Varianti del DES:

- 1) Le chiavi vengono scelte in modo indipendente una dall'altra per un totale di 768 bit anziché 36.

Problema: Numero troppo grande di bit da generare

z) la cifratura multipla prevede la concatenazione di più copie del DES che utilizzano chiavi diverse.

$$C_{\text{DES}}(C_{\text{DES}}(m, k_1), k_2) \neq C_{\text{DES}}(m, k_3)$$

Questo aumenta la sicurezza e diventa pari a 57 bit di chiave

Attenzione: Non $112 = \#$ spazio delle chiavi, come si potrebbe pensare

Questo perché se ho $c = C(C(m, k_1), k_2)$

e lo decifro con k_2 ottengo $D(c, k_2) = C(m, k_1)$

Si prende una coppia $\langle m, c \rangle$

1) $\forall k_1$, calcolo e salvo $C(m, k_1) = 2^{56}$

2) $\forall k_2$ calcolo $D(c, k_2)$ e lo cerco nella lista delle cifrature ed esiste almeno una corrispondenza

Costo: $N = 2^{56}$ cifrature + $O(n)$ decifrazioni = $2N = 2^{57}$

la cifratura multipla più nota è la Triple DES:

1) 2TDEA : $c = C_{\text{DES}}(D_{\text{DES}}(C_{\text{DES}}(m, k_1), k_2), k_1)$

$$m = D_{\text{DES}}(C_{\text{DES}}(D_{\text{DES}}(c, k_1), k_2), k_1)$$

2) 3TDEA : Uguale ma con 3 chiavi.

2TDEA = 112 bit di sicurezza = # chiavi \Rightarrow abbastanza sicuro

3TDEA = 112 bit di sicurezza + Costo di enumerare le coppie k_2, k_3

\hookrightarrow # chiavi > 112 \Rightarrow Non sicuro

3) AES

Cifrario AES (ADVANCED ENCRYPTION STANDARD) : Il nuovo standard

↳ Chiamato anche Rijndael (Nome derivato dai cognomi dei ricercatori Daemen e Rijmen)

L'AES nasce con l'obiettivo di concentrarsi su:

- Sicurezza \Rightarrow Da attacchi crittanalitici noti, corretto processo di cifratura, ...
- Costo di realizzazione \Rightarrow Economicità dell'hardware, velocità di cifratura, ...
- Caratteristiche Algoritmiche \Rightarrow Flessibilità: Facilmente portabile, diverse chiavi ecc.

L'AES è un cifrario a blocchi il quale prevede blocchi da 128 bit e chiavi da 128, 192, 256 bit.

Opera per fasi come il DES, il numero di fasi dipende dalla lunghezza della chiave

128 \rightarrow 10 fasi

192 \rightarrow 12 fasi

256 \rightarrow 14 fasi

Ha una chiave iniziale e una locale per ogni fase (creata a partire dalla chiave segreta)
↓
Usata per le op. iniziali

Selezioni delleotto chiavi locali di ogni fase

Se la chiave è di 128 bit, gli organizza come matrice bidimensionale di

16 byte:

0-7	32-39	64-71	96-103
8-15	40-47	72-79	104-111
16-23	48-55	80-87	112-119
24-31	56-63	88-95	120-127

w(0) w(1) w(2) w(3)

↑ 1 byte

w(0) = Primi 32 bit della chiave

w(1) = 32-63 //

w(2) = 64-95 //

w(3) = 96-127 //

Caricati per colonne

↓

w(i) = Sequenza di 4 byte

Per ricavare le chiavi di fase, parto da $w(1) \dots w(4)$ e le calcolo

$$\forall t \geq 4$$

$$w(t) = w(t-1) \oplus w(t-4) \quad \text{se } t \nmid 4 \quad (\text{t non div per 4})$$

$$w(t-1) = T(w(t-1)) \oplus w(t-4) \quad \text{se } t \mid 4$$



S-box \rightarrow Non lineare

La chiave dell' i -esima fase sarà:

$$1 \leq i \leq 10$$

$$w(u_i), w(u_{i+1}), w(u_{i+2}), w(u_{i+3})$$

Cifratura Messaggi

Ogni fase opera su 128 bit, organizzato come matrice bidimensionale di 16 byte (Riempita per colonne)

$b_{0,0}$	$b_{0,1}$	$b_{0,2}$	$b_{0,3}$
$b_{1,0}$	$b_{1,1}$	$b_{1,2}$	$b_{1,3}$
$b_{2,0}$	$b_{2,1}$	$b_{2,2}$	$b_{2,3}$
$b_{3,0}$	$b_{3,1}$	$b_{3,2}$	$b_{3,3}$

B

Figura 7.8: La matrice B del cifrario AES. I 16 byte $b_{i,j}$ che costituiscono gli elementi della matrice B sono trasformati nel corso di ogni fase.

Trasformazione iniziale:

Il messaggio m è caricato nella matrice B. Ogni byte di B è posto in XOR (bit a bit) con il corrispondente byte della chiave iniziale.

k_{op}	$k_{0,1}$	$k_{0,2}$	$k_{0,3}$
$k_{1,0}$	$k_{1,1}$	$k_{1,2}$	$k_{1,3}$
$k_{2,0}$	$k_{2,1}$	$k_{2,2}$	$k_{2,3}$
$k_{3,0}$	$k_{3,1}$	$k_{3,2}$	$k_{3,3}$



$b_{0,0}$	$b_{0,1}$	$b_{0,2}$	$b_{0,3}$
$b_{1,0}$	$b_{1,1}$	$b_{1,2}$	$b_{1,3}$
$b_{2,0}$	$b_{2,1}$	$b_{2,2}$	$b_{2,3}$
$b_{3,0}$	$b_{3,1}$	$b_{3,2}$	$b_{3,3}$

Dopo la fase iniziale, vengono eseguite le seguenti op. per 10 volte

1) Ogni byte della matrice B è trasformato mediante una S-box

Come funziona la S-box

Un byte in ingresso $b_{i,j}$ viene diviso in 2 parti da 4 bit

$$b_{i,j} = b_0 \ b_1 \ b_2 \ b_3 \mid b_4 \ b_5 \ b_6 \ b_7$$

La parte a sinistra, che sarà un numero $0 \leq x \leq 15$, è la riga scelta

La parte di destra, che sarà un numero $0 \leq x \leq 15$, è la colonna scelta

Uso poi riga e colonna per accedere alle tabelle definite dalle Sbox

e il numero che ottengo lo ritrasformo in binario

Esempio: $b_{i,j} = 1000 \mid 1011$

$$\begin{array}{c} \downarrow \\ 8 \end{array} \quad \begin{array}{c} \downarrow \\ 11 \end{array}$$

Accedo Sbox $[8, 11] \rightarrow 61 = 00111101 \rightarrow$ trasformazione $b_{i,j}$

Quello che fa la Sbox è:

$$x \underset{\text{byte}}{\underset{\text{Sbox}}{\longrightarrow}} x^{-1} \text{ (inverso moltiplicativo in } GF(2^8))$$

2) La matrice così ottenuta è permutata mediante shift ciclico su righe

Op2. La prima riga (riga 0) della matrice B resta inalterata mentre i byte contenuti nelle righe 1, 2, 3 sono soggetti a uno shift ciclico verso sinistra rispettivamente di 1, 2 o 3 posizioni. Così per esempio il byte $b_{2,2}$ sarà spostato in posizione $b_{2,0}$; il byte $b_{3,1}$ sarà spostato in posizione $b_{3,2}$.

$$B = \begin{array}{|c|c|c|c|} \hline b_{0,0} & b_{0,1} & b_{0,2} & b_{0,3} \\ \hline b_{1,0} & b_{1,1} & b_{1,2} & b_{1,3} \\ \hline b_{2,0} & b_{2,1} & b_{2,2} & b_{2,3} \\ \hline b_{3,0} & b_{3,1} & b_{3,2} & b_{3,3} \\ \hline \end{array}$$

$$\longrightarrow B_{\text{shift}} =$$

$$\begin{array}{|c|c|c|c|} \hline b_{0,0} & b_{0,1} & b_{0,2} & b_{0,3} \\ \hline b_{1,1} & b_{1,2} & b_{1,3} & b_{1,0} \\ \hline b_{2,2} & b_{2,3} & b_{2,0} & b_{2,1} \\ \hline b_{3,3} & b_{3,0} & b_{3,1} & b_{3,2} \\ \hline \end{array}$$

3) le colonne risultanti vengono trasformate con un'operazione algebrica
 (questa operazione non viene fatta alla 10° fase)

Op3. Ogni colonna della matrice, trattata come vettore di 4 elementi, viene moltiplicata per una matrice prefissata M di 4×4 byte, ove la moltiplicazione tra byte è eseguita modulo 2^8 e l'addizione è eseguita come XOR. Anche in questo caso non entriamo nel dettaglio, notando solo che la M è scelta in modo che ogni byte di una colonna di B influenzi, nel prodotto con la M , tutti i byte della colonna stessa.

$$B_j \rightarrow M \circ B_j \quad \text{con } 0 \leq j \leq 3$$

$b_{i,j} \rightarrow$ diventa un valore che dipende da tutti i byte della colonna $b_{0,j}, b_{1,j}, b_{2,j}, b_{3,j}$

Dopo le fasi ② e ③:

Si ottiene che tutti i byte di output dipendono da tutti i byte dell'input

4) Ogni byte della matrice risultante è posto in XOR con un byte della chiave locale per quella fase.

Op4. Ogni byte $b_{i,j}$ della matrice B è trasformato come $b_{i,j} \leftarrow b_{i,j} \oplus k_{i,j}$, ove $k_{i,j}$ è il corrispondente byte della chiave locale.

$$B \oplus K$$

Cifrari a composizione di blocchi

I cifrari come il DES e l'AES potrebbero avere problemi se blocchi uguali producono blocchi cifrati uguali introducendo periodicità nel programma.

Per ovviare al problema, uno dei metodi è il **Cipher block Chaining (CBC)**.

Chiper block Chaining

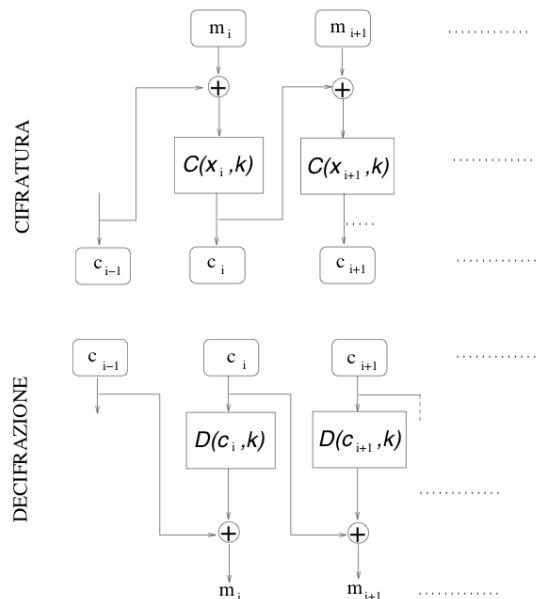
Metodo che nega il principio di diffusione di Shannon inducendo una dipendenza di posizione tra il blocco in elaborazione e quelli precedenti.

Se il messaggio viene diviso in blocchi, ad esempio, da 128 bit.

- Se l'ultimo blocco del msg non è < 128 bit aggiungo una sequenza di bit $S = \dots 00\dots 0$ con tanti 0 quanti bit mi mancano per farlo diventare da 128 bit.
- Se invece l'ultimo blocco è di 128 bit, aggiungo comunque un blocco terminatore composto dalle sequenze $S = \dots 0\dots 0$ di 128 bit

Dopo si sceglie una seq. c_0 iniziale di bit casuali per ogni messaggio inviato.

Quindi effettua la cifratura e decifratura nel seguente modo:



Ogni blocco m_i :

- Viene cifrato come $c_i = C(m_i \oplus c_{i-1}, k)$

xor chiave

blocco $i-1$ -esimo del msg blocco $i-1$ del crittogramma

- Viene decifrato come $m_i = c_{i-1} \oplus D(c_i, k)$

xor

$i-1$ blocco del crittogramma i -esimo blocco del crittogramma

Crittografia a chiave pubblica (Asimmetrica)

Esiste una coppia $(k[\text{Pub}], k[\text{Prv}])$ per ogni utente del sistema.

Y messaggi vengono cifrati come $c = C(m, k[\text{Pub}])$

La decifratura è eseguita come $m = D(c, k[\text{Prv}])$

Proprietà che il processo di Cifratura e Decifratura devono rispettare:

1. Per ogni possibile messaggio m si ha: $D(C(m, k[\text{pub}]), k[\text{prv}]) = m$. Ossia Dest deve avere la possibilità di interpretare qualunque messaggio che gli altri utenti decidano di spedirgli.
2. La sicurezza e l'efficienza del sistema dipendono dalle funzioni C e D , e dalla relazione che esiste tra le chiavi $k[\text{prv}]$ e $k[\text{pub}]$ di ogni coppia. Più esattamente:
 - (a) la coppia $\langle k[\text{prv}], k[\text{pub}] \rangle$ è facile da generare, e deve risultare praticamente impossibile che due utenti scelgano la stessa chiave;
 - (b) dati m e $k[\text{pub}]$, è facile per il mittente calcolare il crittogramma $c = C(m, k[\text{pub}])$;
 - (c) dati c e $k[\text{prv}]$, è facile per il destinatario calcolare il messaggio originale $m = D(c, k[\text{prv}])$;
 - (d) pur conoscendo il crittogramma c , la chiave pubblica $k[\text{pub}]$, e le funzioni C e D , è difficile per un crittoanalista risalire al messaggio m .

Da queste proprietà si ricava che C deve essere una funzione one-way trapdoor cioè facile da eseguire ma difficile da invertire se non si conosce la trapdoor (chiave segreta)

Vantaggi: Crittografia a chiave pubblica

- 1) Bisogna di $2n$ chiavi: invece che $\frac{n(n-1)}{2}$
- 2) Non richiede lo scambio di chiavi:

Svantaggi: Crittografia a chiave pubblica

- 1) Sistemi più lenti rispetto a quelli simmetrici
- 2) Esposti a attacchi di tipo: chosen plain-text

Riciami di algebra modulare

Dati due interi $a, b \geq 0$ e $n > 0$, a è congruo a b modulo n

$$a \equiv b \pmod{n}$$

Se e solo se esiste $k \in \mathbb{Z}$:

$$a = b + kn$$

Proprietà:

- $(a + b) \pmod{n} = ((a \pmod{n}) + (b \pmod{n})) \pmod{n}$.
- $(a - b) \pmod{n} = ((a \pmod{n}) - (b \pmod{n})) \pmod{n}$.
- $(a \times b) \pmod{n} = ((a \pmod{n}) \times (b \pmod{n})) \pmod{n}$.
- $a^{r+s} \pmod{n} = ((a^r \pmod{n})^s) \pmod{n}$, con r e s interi positivi qualunque.

Funzione di Eulero:

Le numeri di interi minori di n e coprimi con esso $\phi(n) = |\mathbb{Z}_n^*|$

In particolare $\phi(n) = n-1$ se n è primo

teorema 1: Se n è il prodotto di due numeri primi p e q

$$\phi(n) = (p-1)(q-1)$$

teorema 2: Per $n > 1$ e per ogni intero a coprimo con n

(teorema di Euler)

$$a^{\phi(n)} \equiv 1 \pmod{n}$$

teorema 3: Per n primo e per ogni $a \in \mathbb{Z}_n^*$

(Piccolo teorema di Fermat)

$$a^{n-1} \equiv 1 \pmod{n}$$

Conseguenza \Rightarrow Se a coprimo con n , $a^{-1} = a^{\phi(n)-1} \pmod{n}$

Equazione: $a \times = b \pmod{n}$

Teorema: L'equazione ammette soluzione $\Leftrightarrow \text{mcd}(a, n)$ è un divisibile di b

Insomma se questo è vero, l'eq. ammette $\text{mcd}(a, n)$ sol. distinte

Corollario: L'eq. ammette unica soluzione

$\Leftrightarrow a$ e n coprimi cioè $\text{mcd}(a, n) = 1$

che è vero \Leftrightarrow esiste a^{-1} di a

Attenzione: L'inverso di a (a^{-1}) sarà la soluzione e

si può calcolare come

$$a^{-1} = a^{\phi(n)-1} \pmod{n} \Rightarrow \begin{array}{l} \text{occorre conoscere} \\ \phi(n) \text{ cioè fattorizzazione} \\ n \Rightarrow \text{difficile} \end{array}$$

Metodo per calcolare l'inverso: $x = a^{-1} \pmod{b}$

Sappiamo che $\text{mcd}(a, b) = 1$

$$ax = t \pmod{b} \Leftrightarrow ax = bz + 1 \quad \text{opportuno scegliere } z$$

$$\Leftrightarrow ax + by = \text{mcd}(a, b) \quad \text{dove } y = -z$$



Value dell'inverso

x è calcolabile applicando la funzione di Eulero Estesa che restituisce la tripla $(\text{mcd}(a, b), x, y)$ in tempo polinomiale nella dim dell'input con x, y t.c. $ax + by = \text{mcd}(a, b)$. Quindi basta prendere il secondo output della tripla restituita dalla funzione.

```
Function Extended.Euclid(a, b):
    if b = 0
        then return (a, 1, 0)
    else (d', x', y') ← Extended.Euclid(b, a mod b);
        (d, x, y) ← (d', y', x' - [a/b]y');
    return (d, x, y).
```

Generatori: $a \in \mathbb{Z}_n^*$ è un generatore di \mathbb{Z}_n^* se la funzione

$$a^k \pmod{n} \quad 1 \leq k \leq \phi(n)$$

genera TUTTI e SOU gli elementi di \mathbb{Z}_n^*

Esempio: $g = 2$ è generatore di $\mathbb{Z}_{13}^* = \{1, 2, \dots, 12\}$

x	1	2	3	4	5	6	7	9	10	11	12
2^x	2	4	8	3	6	12	11	5	10	7	1

OSS:

1) Essendo che il valore 1 viene generato da $k = \phi(n)$ cioè

$a^{\phi(n)} \pmod{n} = 1$ per il teorema di euler,

allora $a^k \not\equiv 1 \pmod{n}$ per ogni $1 \leq k < \phi(n)$ perché
ogni numero deve essere generato una sola volta per far
lo che a sia generatore

2) Non per tutti i valori di n , \mathbb{Z}_n^* ammette generatori

3) Teorema: Se n è primo $\Rightarrow \mathbb{Z}_n^*$ ammette almeno un generatore

Inoltre il numero di generatori sono in totale $\phi(n-1)$

4) 1 non è mai un generatore

Funzioni one-way trapdoor

Facile da eseguire ma difficile da invertire se non si conosce la trapdoor (chiave segreta). Calcolare l'inversa senza trapdoor richiede la soluzione di un problema NP-HARD \Rightarrow comunque algoritmico non polinomiali.

1) Fattorizzazione:

Prodotto n di due interi p e $q \Rightarrow$ tempo polinomiale

Trovare p e q da $n \Rightarrow$ Esponenziale \rightarrow diventa facile se si conosce $p \circ q$ (trapdoor)

2) Calcolo della radice in modulo:

- Calcolare la potenza $y = x^z \bmod s$ con x, z, s interi \Rightarrow tempo Polinomiale
- Se s non primo, invertire la funzione e calcolare $x = y^{1/z} \bmod s \Rightarrow$ tempo Esponenziale
- Se x è primo con s e si conosce $v = z^{-1} \bmod \phi(s)$ (trapdoor)

Si calcola facilmente $x = y^v \bmod s$ infatti $y^v \bmod s = x \bmod s$

3) Calcolo del logaritmo discreto:

- Calcolare la potenza $y = x^z \bmod s \Rightarrow$ tempo Polinomiale
- Invertire rispetto a z cioè trovare $y = x^z \bmod s \Rightarrow$ tempo esponenziale dati x, y, s
- Si può introdurre una trapdoor

Attacco chosen plain-text

Se un crittanalista cifra alcuni messaggi con la chiave pubblica, può intercettare alcuni messaggi confrontando il cifrogramma con quelli fatti da lui.

Se corrispondono \Rightarrow messaggio decifrato. Pericoloso se il crittanalista sta aspettando che venga inviato un messaggio specifico.

Cifrario RSA

Il cifrario RSA è un cifrario a chiave pubblica presentato nel 1978

Come funziona:

1) Creazione della chiave:

Ogni possibile destinatario esegue le seguenti operazioni:

- Sceglie 2 numeri primi p e q molto grandi (Almeno 1024 bit)
- Calcola $n = p \times q$ e la funzione di Eulero $\phi(n) = (p-1)(q-1)$
- Sceglie un intero e minore e coprimo con $\phi(n)$
- Calcola l'intero $d = e^{-1} \pmod{\phi(n)}$
- Rende pubblica la chiave $K[\text{pub}] = (e, n)$ e mantiene segreta la chiave $K[\text{priv}] = d$

Dim. blocchi per la cifratura = $\lceil \log_2 n \rceil$

2) Messaggio:

Sequenza binaria trattata come un intero. Il messaggio viene

diviso in blocchi di $\log_2 n$ bit con dimensione massima che dipende dalla $K[\text{pub}]$ del destinatario

a) Cifratura: $c = C(m, K[\text{pub}]) = m^e \pmod{n}$

b) Decifratura: $m = D(c, K[\text{priv}]) = c^d \pmod{n}$

Dimostrazione di Correttezza del RSA:

Bisogna dimostrare che la scelta di un qualiasi intero e coprimo con $\phi(n)$ \Rightarrow funzione $x^e \pmod{n}$ sia una permutazione di \mathbb{Z}_n in modo da garantire l'invertibilità del processo di

cifratura, poiché qualsiasi coppia di messaggi $m_1 \neq m_2$ risulta

$$C(m_1, K[\text{pub}]) \neq C(m_2, K[\text{pub}])$$

Però non è sufficiente al nostro scopo perché non è ovvio che decifrando un citoogramma si ricostruisca il messaggio che è stato spedito,

cioè che $D(C(m, \kappa[\text{pub}], \kappa[\text{priv}]) = m$,

ovvero $(m^e \bmod n)^d \bmod n = m$ (infatti d è l'inverso di e modulo $\phi(n)$) mentre nel calcolo indicato interviene op. mod n).

Vogliamo dimostrare che questa relazione è valida, cioè il cifrario è corretto.

Teorema 8.5. Per qualunque intero $m < n$ si ha: $(m^e \bmod n)^d \bmod n = m$, ove n, e, d sono i parametri del cifrario RSA.

Per una proprietà del calcolo modulare (paragrafo 8.1) la relazione da dimostrare può essere riscritta come $m^{ed} \bmod n = m$. Per la dimostrazione distinguiamo due casi, relativi ai valori di p e q scelti dal destinatario e al valore di m scelto dal mittente:

p e q non dividono m . Abbiamo $\text{gcd}(m, n) = 1$, quindi per il teorema di Eulero (paragrafo 8.1) risulta $m^{\Phi(n)} \equiv 1 \pmod{n}$. Poiché d è l'inverso di e in $Z_{\Phi(n)}^*$, abbiamo $e \times d \equiv 1 \pmod{\Phi(n)}$, ovvero $e \times d = 1 + r\Phi(n)$, con r intero positivo opportuno. Otteniamo quindi: $m^{ed} \bmod n = m^{1+r\Phi(n)} \bmod n = m \times (m^{\Phi(n)})^r \bmod n = m \times 1^r \bmod n = m$.

p (oppure q) divide m , ma q (oppure p) non divide m . Poiché p divide m abbiamo $m \equiv m^r \equiv 0 \pmod{p}$, ovvero $(m^r - m) \equiv 0 \pmod{p}$, per qualunque intero positivo r . Con un procedimento analogo a quello del punto precedente abbiamo anche: $m^{ed} \bmod q = m^{1+r\Phi(n)} \bmod q = m \times m^{r(p-1)(q-1)} \bmod q = m \times (m^{(q-1)r(p-1)}) \bmod q = m \bmod q$, ove l'ultima uguaglianza è dovuta alla relazione $m^{(q-1)} \equiv 1 \pmod{q}$ per il teorema di Eulero. Dunque $m^{ed} \equiv m \bmod q$, e quindi $(m^{ed} - m) \equiv 0 \pmod{q}$. Ne consegue che $m^{ed} - m$ è divisibile sia per p che per q , quindi è divisibile per il loro prodotto $p \times q = n$; ovvero $(m^{ed} - m) \equiv 0 \pmod{n}$ da cui deriva immediatamente la tesi.

Si noti che p e q non possono dividere entrambi m perché si avrebbe $m \geq n$ contro l'ipotesi sulla dimensione dei blocchi. La correttezza del cifrario è così dimostrata. Vi sono però diverse considerazioni da fare sulla complessità delle operazioni che esso richiede.

Attacchi all'RSA

Gli attacchi possibili a un cifrario a chiave pubblica sono di quelli diversi plain-text (ogni utente solo se il crittanalista non è in grado di prevedere il contenuto dei messaggi che vengono scambiati).

1) Per decifrare un citoigramma C , sarebbe suff. calcolare $m = \sqrt[e]{C} \bmod n$ perché tutti conoscono $\kappa[\text{pub}] = (e, n)$ e sanno che $C = m^e \bmod n$.

Sappiamo però che il calcolo della radice e -esima nell'algebra modulare

può essere eseguito efficacemente se n primo, ma è notevolmente difficile se n è composto

Dimostrazione della

Correctezza

Ne segue che il calcolo di m a partire da e, n, c non è più facile del calcolo dei fattori primi di n , quindi tanto vale calcolare questi ultimi, anche perché ciò implica forzare totalmente il sistema e non la singola decifrazione di un particolare critogramma.

2) Un altro metodo potrebbe basarsi sulla scoperta di $\phi(n)$ senza passare attraverso i valori p e q poiché conoscendo $\phi(n)$ si calcola immediatamente la chiave segreta. Però la situazione non cambia.

Potiamo infatti:

a) Scrivere $\phi(n) = (p-1)(q-1) = n - (p+q) + 1$

b) Conoscendo $\phi(n)$ potiamo calcolare $x_1 = p+q$

c) Osservando che $(p-q)^2 = (p+q)^2 - 4n$, calcoliamo $x_2 = (p-q)$

d) Infine si ha $p = \frac{(x_1 + x_2)}{2}$ e $q = \frac{(x_1 - x_2)}{2}$

Cioè vuol dire che il calcolo di $\phi(n)$ e la fattorizzazione di n sono problemi equivalenti poiché si può passare dall'uno all'altro in tempo polinomiale.

3) Si potrebbe condurre un attacco esauriente sulla chiave segreta d , cioè verificare, per ogni possibile valore di d , se la decifrazione dà origine a un messaggio significativo. Il processo potrebbe essere più costoso della fattorizzazione di n poiché, in dipendenza della scelta di e , il valore di d può essere molto più grande di quello di p e q .

Quindi tutti questi tentativi ci conducono al fatto che per forzare RSA bisogna fattorizzare n .

Accorgimenti con cui sceglie p, q ed e in modo che non ponano essere tutte alcune proprietà

+) p e q molto grandi (un migliaio di bit) \Rightarrow Perché esistono algoritmi particolarmente efficienti e inoltre la fatt. e le log discrete non sono NP-Hard

2) Differenza tra p e q grande

Se $|p-q|$ fosse piccolo, $\frac{(p-q)}{2}$ sarebbe vicino a \sqrt{n} . Esempio $\frac{(p+q)^2}{4} - n = \frac{(p-q)^2}{4}$

e il membro sinistro dell'eq. un quadrato perfetto $\Rightarrow \frac{p+q}{2} > \sqrt{n}$

Dunque si possono scegliere gli interi $> \sqrt{n}$ fino a trovare un intero z t.c

$z^2 - n$ sia un quadrato perfetto, $z^2 - n = w^2$. Si suppone quindi che

$$z = \frac{p+q}{2} \quad \text{e} \quad w = \frac{p-q}{2} \quad \text{da cui trovare } p = z+w \quad \text{e} \quad q = z-w.$$

Infine controllare il cattogramma. Il fatto che $|p-q|$ sia piccolo ci dice che non bisogna allontanarsi troppo da \sqrt{n} .

3) $\frac{p-1}{2}$ e $\frac{q-1}{2}$ primi fra loro (Massimo comun divisore piccolo)

a) Se $u \mid \phi(n)$ e se m, n primi fra loro, per $e = 1 + \frac{\phi(n)}{k}$ si

avrebbe $c = m^e \pmod{n} = m \cdot (m^{\phi(n)})^{1/u} \pmod{n} = m$, cioè non si avrebbe nessuna trasformazione del mex.

5) Si consiglia di scegliere e come numero primo molto piccolo (rapide cifrature) e si impone di aggiungere una seq. casuale di bit diversi per ogni destinatario alla fine di ogni mex.

Attenzione: e non deve essere più piccolo di un certo soglia, non deve avere $m^e < n$ poiché risulterebbe facile trovare \bar{x} e rendendo che non interviene la riduzione in modulo.

RSA soffre degli stessi problemi che si presentano nei cifrari simmetrici a blocchi.

In particolare il modo CBC per la compostizione di blocchi si applica anche all'RSA.

Cifrari ibridi

Cifrari dove le chiavi segrete vengono scambiate tramite i cifrari a chiave pubblica, la quale viene poi usata nei cifrari simmetrici.

Combinazione DES / RSA

La chiave segreta viene trasmessa con l'RSA, denominata chiave di sessione $K[\text{session}]$. La chiave è modificabile ogni volta che si vuole.

I messaggi segreti vengono scambiati nella forma

$$C_{\text{RSA}}(K[\text{session}], K[\text{pub}]), C_{\text{AES}}(m, K[\text{session}])$$

Ha i vantaggi sia del DES che del AES senza soffrire di attacchi chosen plain text.

Non è banale che il mittente debba avere le potere di calcolo e gli strumenti per creare una chiave privata.

Attenzione: Su internet viene data uguale responsabilità sia al mittente che al destinatario.

Protocollo DH (Diffie ed Hellman) per lo scambio pubblico delle chiavi

Sostituisce l'RSA perché più semplice e resistente ad attacchi parziali.

È un algoritmo molto diffuso nei protocolli criptografici usati su internet ad esempio l'SSL (Secure socket layer).

Gli utenti generano la chiave $K[\text{session}]$ inviandosi in modo incrementale parti di essa.

Queste parti possono essere ricevute solo conoscendo alcune informazioni regolarmente in posesso da entrambi:

- 1) Alice e Bob si accordano pubblicamente su un p molto grande ($100+bit$) e su un generatore g di \mathbb{Z}_p^* .

Primo

La generazione di chiavi sicureamente perché p è primo, inoltre si ha $\mathbb{Z}_p^* = \{1, 2, \dots, p-1\}$.

Se Alice e Bob non possono generare la coppia (g, p) possono comunque usare una coppia pubblica messa a disposizione dal sistema.

2) Alice estrae un intero positivo casuale $x < p$, calcola $X = g^x \bmod p$ e spedisce in chiaro questo valore a Bob.

Varrà diversi di x generano valori diversi di $X \in [1, p-1]$

3) Bob estrae un intero positivo casuale $y < p$, calcola $Y = g^y \bmod p$ e spedisce il valore in chiaro ad Alice.

Varrà diversi di y generano valori diversi di $Y \in [1, p-1]$

4) Alice riceve Y e calcola $U[\text{senza}] = Y^x \bmod p = g^{xy} \bmod p$
sfruttando la conoscenza privata di x

5) Bob riceve X e calcola $U[\text{senza}] = X^y \bmod p = g^{xy} \bmod p$
sfruttando la conoscenza privata di y

Entrambi hanno generato la chiave di simmetria che può essere usata per lo scambio di messaggi.

Un crittanalista pur intercettando le varie parti in chiaro, dovrebbe calcolare $X = g^x \bmod p$ rispetto a X e $Y = g^y \bmod p$ rispetto a y , che è un problema computazionalmente difficile e quindi improponibile per valori di p molto grandi.

CRITTOANALISI:

Il crittanalista però può agire con un attacco man-in-the-middle. Infatti può scegliere un intero z qualunque, calcolare $Z = g^z \bmod p$

e soffrire le comunicazioni tra Alice e Bob con le proprie.

Cattura i mez X e Y di Bob e Alice e risponde a Entrambi con Z. Alice e Bob condividono le chiavi con Z (diverse tra loro)

$$k_A = Z^x \bmod p = g^{xz} \bmod p \quad \text{e} \quad k_B = Z^y \bmod p = g^{yz} \bmod p$$

Un attaccante userà k_A per comunicare con Alice e k_B per comunicare con Bob.

CIFRARIO DI EL GAMAL

Alice vuole mandare un messaggio a Bob

Bob

- 1) Sceglie p (numero primo molto grande) e g (generatore per \mathbb{Z}_p^*)
- 2) Sceglie $2 \leq x \leq p-2$ come chiave privata $\Rightarrow k[\text{priv}] = x$
- 3) Calcola $y = g^x \bmod p$
- 4) Pubblica $k[\text{pub}] = \langle p, g, y \rangle$

Alice $0 \leq m < p$

- 1) Si procura $k[\text{pub}] = \langle p, g, y \rangle$
- 2) Sceglie a caso $2 \leq n \leq p-2$
- 3) Calcola $c = g^n \bmod p$
- 4) Calcola $d = m \cdot y^n \bmod p$
- 5) Invia a Bob la coppia $\langle c, d \rangle$

Bob

$$\text{Riceve } \langle c, d \rangle \text{ e decifra } m = \frac{d}{c^x} \bmod p$$

Connettorza:

$$\frac{d}{c^x} \bmod p = \frac{y^n \cdot m}{c^x} \bmod p = \frac{g^{x \cdot n} \cdot m}{g^{nx}} \bmod p = m \bmod p = m$$

\downarrow
 $m < p$

Crittoanalisi:

Conosce p, g, y, c, d

$$1) \text{ Se conosce } x \Rightarrow m = \frac{d}{c^x} \bmod p$$

$$2) \text{ se conosce } n \Rightarrow m = \frac{d}{y^n} \bmod p$$

CRIPTOGRAFIA SU CURVE ELLITTICHE (ECC = Elliptic Curve Cryptography)

Metodo di crittografia a chiave pubblica più sicuro e efficiente di RSA e DH.

Definizione:

Una curva ellittica E su un campo K è definita come l'insieme dei punti $(x, y) \in K^2$ t.c.

$$y^2 + axy + by = x^3 + cx^2 + dx + e$$

dove $a, b, c, d, e \in K$.

OSS: Se la caratteristica di K è $\neq 2, 3$ allora l'equazione che definisce la curva ellittica si può ridurre alla forma normale di Weierstrass:

$$y^2 = x^3 + ax + b \quad a, b \in K$$

Caratteristica di un campo:

Più piccolo numero naturale K f.c. sommando K volte l'elemento neutro moltiplicativo del campo K (indicato con 1), si ottiene l'elemento neutro additivo di K (indicato con 0). Se non esiste un tale K allora è 0 per definizione.

Le curve ellittiche sono usate in crittografia per la possibilità di attribuire all'insieme dei punti di una curva la struttura algebrica di un **gruppo abeliano additivo**, ovvero di definire una legge di composizione interna che permette di associare ad ogni coppia di punti sulla curva, un terzo punto sempre sulla curva.

La legge di composizione chiamata anche somma sulle curve ellittiche, è associativa, commutativa, ammette un elemento neutro, ed c'è t.c. sia definito l'inverso di ogni punto.

CURVE ELLITTICHE SU NUMERI REALI

Supponiamo che il campo sia \mathbb{R} . La curva sarà costituita dall'insieme $E(a,b)$ dei punti $(x,y) \in \mathbb{R}^2$ che soddisfano $y^2 = x^3 + ax + b$.

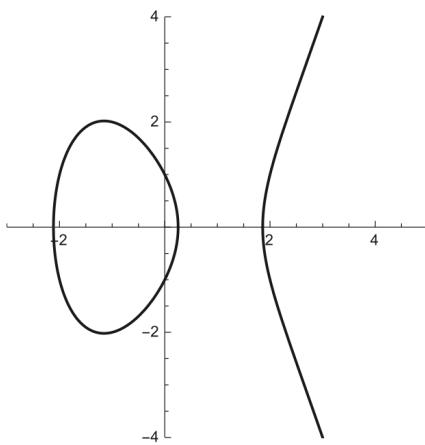
ovvero:

$$E(a,b) = \{(x,y) \in \mathbb{R}^2 \mid y^2 = x^3 + ax + b\} \text{ dove } a,b \in \mathbb{R}$$

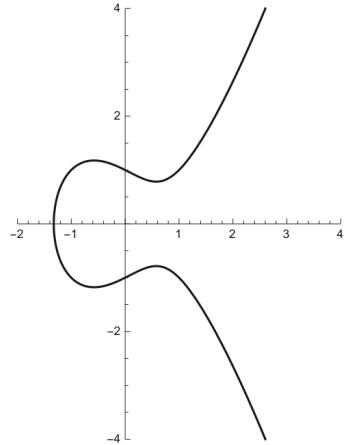
Asumiamo inoltre che $4a^3 + 27b^2 \neq 0 \Rightarrow$ Ci assicura che il polinomio cubico non abbia radici multiple, cioè privo di punti singolari come "cuspidi" o "nodi" dove non sarebbe definita la tangente.

Quindi c'è garantita l'esistenza della tangente in ogni punto della curva.

RAPPresentazione grafice



(a) Curva $y^2 = x^3 - 4x + 1$



(b) Curva $y^2 = x^3 - x + 1$

Come si può notare
esse hanno una
simmetria orizzontale

Forme quando il polinomio
ha 3 radici reali

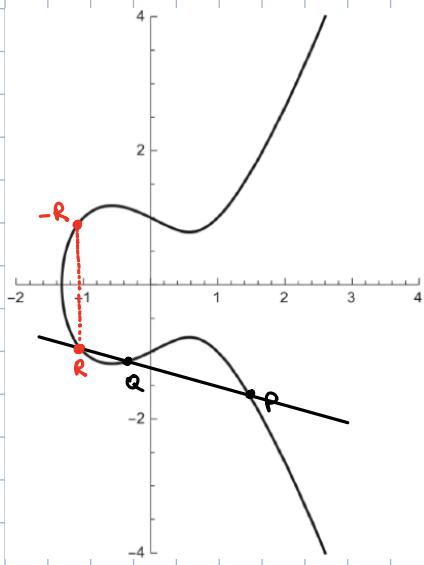
Forme quando il polinomio
ha 1 radice reale

Proprietà : Qui retta interseca una curva ellittica in al più 3 punti

Operazione di addizione su una curva ellittica:

Per sommare due punti P e Q :

- Si considera la retta passante per P e Q , oppure l'unica tangente alla curva in P nel caso P e Q coincidano;
- Si determina il punto di intersezione R tra le curve e la retta per P e Q , oppure tra la curva e la tangente in P per $P=Q$
- Si definisce somma di P e Q il punto simmetrico a R rispetto all'asse delle ascisse, ovvero si pone $P+Q = -R$



Proprietà ($\forall P, Q \in E(a,b)$)

- 1) Chiusura: $P+Q \in E(a,b)$
- 2) Elemento Neutro: $P+O = O+P = P$
- 3) Inverso: $P = (x, y) \Rightarrow -P = (x, -y)$
- 4) Commutativa: $P+Q = Q+P$
- 5) Associativa: $(P+Q)+R = P+(Q+R)$

Formulazione algebrica

Sia $P = (x_P, y_P)$ e $Q = (x_Q, y_Q)$ e $P+Q=S=(x_S, y_S)$

1) Se $Q \neq \pm P$:

$$x_S = \lambda^2 - x_P - x_Q$$

$$y_S = -y_P + \lambda(x_P - x_S)$$

$$\text{con } \lambda = \frac{y_Q - y_P}{x_Q - x_P}$$

2) Se $Q = P$ cioè $S = P + P$

$$\begin{aligned}x_S &= \lambda^2 - x_P - x_Q \\y_S &= -y_P + \lambda(x_P - x_S)\end{aligned}$$

$$\text{con } \lambda = \frac{3x_P^2 + a}{2y_P}$$

$$(\text{Se } y_P = 0 \Rightarrow S = O)$$

(origine)

3) Se $Q = -P$

$$S = P - P = O$$

Tipi di curve:

1) **Curva binaria**: Il campo è $K = GF(2^m)$ con $m \in \mathbb{N}$

2) **Curve prime**: Sia il campo $K = \mathbb{Z}_p$ con p primo

$$E_p(a,b) = \{(x,y) \in \mathbb{Z}_p^2 \mid y^2 \bmod p = x^3 + ax + b \bmod p\} \cup \{O\}$$

Caratteristiche

1) Se $P(x,y) \in E_p(a,b) \Rightarrow -P = (x, p-y) \in E_p(a,b)$

2) Per definire un gruppo abeliano: $(4a^3 + 27b^2) \bmod p \neq 0$

Ordine:

L'ordine è il numero di punti della curva.

Dato $y^2 = x^3 + ax + b$, dato \mathbb{Z}_p , il numero di residui quadratici

$$(q \mid \exists x. x^2 = q \bmod p)$$

Sono: $\frac{p-1}{2}$ residui quadratici

Le x che non danno un residuo quadratico non sono punti sulla curva

Teorema di Hasse

L'ordine N di una curva ellittica $E_p(a,b)$ verifica la diseguaglianza

$$|N - (P+1)| \leq 2\sqrt{P}$$

quindi N va \pm come P

Esempio:

Sia $y^2 = x^3 + 4x + 4 \pmod{5}$ la curva ellittica, calcolare il suo ordine

1) Calcolo i residui quadratici:

y	y^2
0	0
1	1
2	4
3	4
4	1

$\Rightarrow 0, 1 \text{ e } 4$ sono residui quadratici

2) Considero tutti i possibili valori di x

$$x=0 \quad y^2 = 4 \quad \text{che è un residuo quad} \Rightarrow (0,2) \text{ e } (0,3) \in E_5(4,4)$$

$$x=1 \quad y^2 = 4 \quad // \Rightarrow (1,2) \text{ e } (1,3) \in E_5(4,4)$$

$$x=2 \quad y^2 = 0 \quad // \Rightarrow (2,0) \in E_5(4,4)$$

$$x=3 \quad y^2 = 3 \quad \text{Non è un res quad} \Rightarrow \text{Non ci sono punti di ascissa } x=3$$

$$x=4 \quad y^2 = 4 \quad // \Rightarrow (4,2) \text{ e } (4,3) \in E_5(4,4)$$

$$\text{Ordine} = 7 + 1 = 8$$

punti
sulla curva

punto all'infinito

Funzione one-way delle ECC - "Moltiplicazione Scalare"

$Q = k \cdot P$ si può fare in tempo pol.
con nodi doppi ripetuti

Calcolare $Q = k \cdot P$ dato k e P si può fare in $\Theta(\log k)$ operazioni

con i redolloppi ripetuti:

Se voglio calcolare $Q = 13P$

Considero $13P = (8+4+1)P = \underbrace{8P + 4P + P}_{}$

Per trovare $8P$ e $4P$ mi basterà fare 3 redolloppi

$$P \rightarrow 2P \rightarrow 4P \rightarrow 8P$$

Calcolo Q come:

$$8P + 4P + P \quad \text{cioè} \quad 2 \text{ somme}$$

Algoritmo generale:

$$k = \sum_{i=0}^t k_i 2^i \quad \text{perché avrò } t+1 = \lfloor \log_2 k \rfloor + 1 \text{ bit}$$

1) Si calcolano $2P, 4P, \dots, 2^t P$

2) Si calcola $Q = \sum_{i=1}^t 2^i P \quad \} \quad O(t) \text{ somme cioè } O(\log k)$

Calcolare k da P e Q risulta difficile (Inverso Molt. scalare)

Cioè trovare $\min(k)$ t.c. $Q = kP$

log. discreto su Curve ellittiche

Perché per trovare k semplicemente faccio:

1) Prendo P e calcolo $2P$

2) Se $2P = Q$ bene se no calcolo $3P$

⋮

Costo esponenziale

Protocollo DH su curve ellittiche

Alice e Bob scelgono una curva ellittica e un punto B della curva con ordine molto grande.

ORDine (n) di B = più piccolo intero n t.c. $nB = 0$



Svolgo $B + B + \dots + B$ molte volte

le curve e i punti sono condivisi: in delle liste pubbliche

Alice:

\uparrow intero

1) Estrae $n_A < n$ casuale

Bob:

\uparrow intero

1) Estrae $n_b < n$ casuale

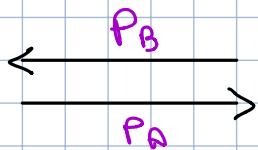
2) Calcola la $K[\text{pub}] = n_A B$

2) Calcola la $K[\text{pub}] = n_b B$

3) Invia $K[\text{pub}]$ a Bob

3) Invia $K[\text{pub}]$ a Alice

P_A



P_B

1) Riceve P_B

1) Riceve P_A

2) Calcola $S = n_A \cdot P_B = n_A \cdot n_B \cdot B$

2) Calcola $S = n_B \cdot P_A = n_B \cdot n_A \cdot B$



Punto comune
sulla curva



Punto comune
sulla curva

$$K[\text{sessione}] = S \mod 2^{256}$$

CRITTO ANALISTA

1) Conosce $E_p(a, b)$, B , P_A, P_B

2) Per trovare n_A o n_B deve fare:

- Per $n_A \Rightarrow n_A B = P_A$

- Per $n_B \Rightarrow n_B B = P_B$

Ma deve risolvere il log. discreto N.C.E che ha costo esponenziale.

Scambio di messaggi cifrati:

Dovrò trasformare il messaggio m in punto della curva $E_p(a, b)$

$$m \longrightarrow p_m$$

Prob. che sostituendo x all'eq., x non sia residuo quadratico $\approx 1/2$

Algoritmo di Koblitz (Algoritmo polinomiale randomizzato)

Prende $m < p$ e restituisce $p_m \in E_p(a, b)$

Si pima un intero h t.c. $(m+1) \cdot h < p$

$$x = m \cdot h + i \quad \text{con} \quad 0 \leq i \leq h-1$$

Koblitz (m, h, b, p)

for ($i=0$; $i < h$; $i++$) {

$$x = mh + i$$

$$z = (x^3 + ax + b) \bmod p$$

if ($z \neq -e$ un residuo quadratico) {

$$y = \sqrt{z}$$

$$\text{return } p_m = (x, y)$$

} costo polinomiale

perché p primo

}

}

return failure;

Prob. di fallimento $\approx (1/2)^h$

Prob. di successo $\approx 1 - (1/2)^h$

Per risalire ad m da x :

$$\left\lfloor \frac{x}{h} \right\rfloor = \left\lfloor \frac{mh+i}{h} \right\rfloor = \left\lfloor m + \frac{i}{h} \right\rfloor = m$$

Scambio dei messaggi

Ogni utente ha:

- 1) $E_p(a, b)$
- 2) B di ordine elevato n con $B \in E_p(a, b)$
- 3) h per alg. di Koblitz

Ogni utente genera $u[\text{pub}]$ e $u[\text{priv}]$

chiave privata: $\boxed{n_u} < n$

chiave pubblica: $\boxed{P_u} = n_u B$

Alice (mittente)

- 1) Mappa m su $P_m \in E_p(a, b)$ (Alg. Koblitz)
- 2) Sceglie un intero casuale n e calcola $V = nB$
- 3) $w = P_m + n \underbrace{P_{\text{Bob}}}_{\text{chiave pubb. di bob}}$

$n P_{\text{Bob}}$ → punto casuale su $E_p(a, b)$

- 4) Invia a Bob la coppia $\langle V, w \rangle$

Bob (Destinatario)

- 1) riceve $\langle V, w \rangle$
- 2) Decifra: $w - n \underbrace{P_{\text{Bob}}}_{\text{chiave pubb. di bob}} V = (P_m + n P_{\text{Bob}}) - nV = P_m + n n \cancel{\text{Bob}} B - n \cancel{n \text{Bob}} B = P_m$
- 3) Transforma P_m in $m \rightarrow$ calcola $m = \lfloor \frac{x}{h} \rfloor$

Criptanalisi: la difficoltà sta sul log. discreto su curve ellittiche

- Se trova n , decifra: $w - n P_{\text{Bob}} = P_m$

Pero per trovare n da V e B deve fare

$$V = nB \rightarrow \text{log Discreto (Esponenziale)}$$

- Deve trovare n_{Bob} da P_{Bob}, B

$$P_{\text{Bob}} = n_{\text{Bob}} B \rightarrow \text{log Discreto (Esponenziale)}$$

Sicurezza:

Per attaccare RSA, DH e El gamal

$$\text{Algoritmi di costo} = O(2^{\sqrt{b \log b}})$$

con $b = \# \text{ bit del modulo}$

Per attaccare protocolli Au CE

$$\text{Algoritmi di costo} = O(2^{b/2})$$

con $b = \# \text{ bit dell'ordine di } B$

IDENTIFICAZIONE, AUTENTICAZIONE E FIRMA DIGITALE

Identificazione:

Un sistema di elaborazione, isolato o in rete, deve essere in grado di accettare l'identità di un utente che richiede di accedere ai suoi servizi.

Autenticazione:

Il destinatario di un messaggio deve essere in grado di accettare

- l'identità del mittente
- l'integrità del crittogramma ricevuto.

Firma digitale

1. MITT non deve poter negare di aver inviato un messaggio m .

2. DEST deve essere in grado di autenticare il messaggio

3. DEST non deve poter sostenere che $m' \neq m$ è il messaggio inviato da MITT.

Tutto deve essere verificabile da una terza parte.

OSS: FIRMA digitale \Rightarrow Autenticazione \rightarrow Identificazione

Funzioni hash one-way

Una funzione può essere applicata in crittografia \Leftrightarrow

1. per ogni $x \in X$ è computazionalmente facile calcolare

$$y = f(x)$$

2. Proprietà one-way: per la maggior parte degli $y \in Y$ è computazionalmente difficile determinare $x \in X$ tale che

$$f(x) = y, \text{ i.e., } x = f^{-1}(y)$$

3. Proprietà claw-free: è computazionalmente difficile determinare una coppia di elementi x_1, x_2 in X tali che

$$f(x_1) = f(x_2)$$

Funzione hash MD5

Riceve in input una sequenza S di 512 bit e produce un'immagine di 128 bit.

È stato dim. che MD5 non resiste alle collisioni

Non più sicura!

Funzione hash RIPEMD-160

Variante "matura" delle MD

Produce immagini da 160 bit ed è esente dai difetti di MD5

Funzione hash SHA (Secure Hash Algorithm)

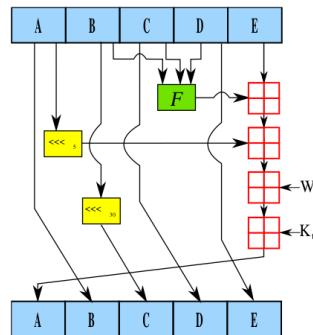
Si usa quando la proprietà claw-free è cruciale per la sicurezza del sistema.

Opera su sequenze fino a 2^{64} bit e produce immagini di 160 bit

Funzione crittograficamente sicura

SHA-1

- Opera su sequenze lunghe fino a $2^{64}-1$ bit, e produce **immagini di 160 bit**.
- E' molto usata nei protocolli crittografici anche se non è più certificata come standard.
- Tutte le altre funzioni hanno una struttura molto simile a SHA-1.
- Opera su blocchi di 160 bit, contenuti in un buffer di 5 registri di 32 bit ciascuno, in cui sono caricati inizialmente dei valori pubblici.
- Il messaggio m viene concatenato con una sequenza di padding che ne rende la lunghezza multipla di 512 bit.
- Il contenuto dei registri varia nel corso dei cicli successivi in cui questi valori si combinano tra loro e con blocchi di 32 bit provenienti da m .
- Alla fine del procedimento, i registri contengono SHA-1(m).



Un'iterazione all'interno della funzione di compressione di SHA-1.
A, B, C, D ed E sono parole di stato a 32 bit; F è una funzione non lineare che varia; $\ll\ll_n$ denota una rotazione del bit di sinistra di n posti; n varia per ogni operazione. \oplus denota l'addizione modulo 2^{32} . K_t è una costante.

W_t blocco di 32 bit ottenuto tagliando e rimescolando i blocchi di messaggio

Il contenuto dei registri varia nel corso dei cicli (all'inizio sono caricati valori fissi e pubblici) in cui questi valori si combinano tra loro e con blocchi di 32 bit provenienti dal messaggio W , nonché con alcuni parametri relativi al ciclo. Alla fine del procedimento (quando è stato letto l'intero messaggio) i registri contengono l'hash SHA1(W)

IDENTIFICAZIONE

1) Se il canale è sicuro, cioè non è possibile intercettare la password la prima volta che un utente U fornisce la password, il sistema associa a U due seq. binarie che memorizza al posto delle password:

- Seme (S) = Prodotto del generatore pseudocanale
- $Q = h(P+S)$ = Risultato ottenuto applicando h funzione hash one-way

Ad ogni successiva connessione di U, il sistema:

- recupera S dal file delle password,
- concatena S con la password fornita da U
- calcola l'immagine one-way della nuova sequenza: $h(PS)$
- se $h(PS) = Q$ l'identificazione ha successo.

Un accesso illegittimo al file delle password non fornisce informazioni interessanti:

è computazionalmente difficile ricavare la password originale dalla sua immagine one-way

alla Password (P) + Seme (S)

e, n = chiave pubblica
 d = chiave privata

2) Se il canale non è sicuro e un utente U richiede l'accesso al sistema S, protocollo di id. basato su RSA:

- S genera un numero n Canale con $n < n$ e lo invia a U
- U calcola $f = n^d \text{ mod } n$ e lo invia a S
- S verifica la correttezza del valore ricevuto calcolando e verificando se $f^e \text{ mod } n = n$ → Se ciò avviene, identificato!

Cio' possibile perche' f puo' essere generata solo da U che fornisce $\langle d \rangle$

Problema: Si potrebbe scegliere di proprio n per ricavare info da U

→ Soluzione: **Protocollo a conoscenza zero**

Autenticazione:

Destinatario deve autenticare il messaggio accertandosi l'identità del mittente e l'integrità del msg \Rightarrow Mitt e Dest condividono una chiave segreta.

L'autenticazione avviene attraverso una funzione $A(m, k)$, con A e k condivisi precedentemente, la quale produce un'informazione MAC (Message Authentication Code)

- 1) Il MAC viene allegato al messaggio o al suo ciphogramma da parte del mittente
 $\langle m, A(m, k) \rangle$ $\langle C(m, k), A(m, k) \rangle$
- 2) Il destinatario calcola $A(m, k)$ e lo confronta per la verifica.

- OSS:
- a) $A(m, k)$ è una funzione hash one-way t.c. $A(m, k) = h(m, k)$
quindi per recuperare k si dovrà invertire h
 - b) Il crittanalista per sostituire il msg dovrà conoscere k
per calcolare il MAC
 - c) Spazio dei MAC \subset Spazio Mex \subset Spazio ciphogrammi
 - d) Usando un cifrario a blocchi si può usare il blocco finale
del ciphogramma come MAC il quale dipende dall'intero msg.

Firma Digitale

Deve rispettare i requisiti di una firma manuale

1. è autentica e non falsificabile

prova che chi l'ha prodotta è chi ha sottoscritto il documento;

2. non è riutilizzabile

è legata strettamente al documento su cui è stata apposta;

3. il documento firmato non è alterabile

chi ha prodotto la firma è sicuro che questa si riferirà solo al documento sottoscritto nella sua forma originale;

4. non può essere ripudiata da chi l'ha apposta

costituisce prova legale di un accordo o dichiarazione.

Protocollo 1: Mensaggio m in chiaro e firmato

Funzione di decifrazione



1) Firma: Mittente U genera la firma $f = D(m, k_U[\text{priv}])$

e invia la tripla $\langle U, m, f \rangle$

2) Verifica: Destinatario riceve la tripla e verifica f controllando

$$m = C(f, k_U[\text{pub}])$$

↳ funzione di cifratura

Problema: Quando un messaggio è doppio più lungo poiché f ha dim.

paragonabile a m. Inoltre non sono cifrare i p. mex.

Soddisfa i requisiti:

Il protocollo soddisfa i requisiti della firma manuale

> f è autentica e non falsificabile (1)

• $k_U[\text{priv}]$ è nota solo a U

• per falsificare la firma occorre conoscere $k_U[\text{priv}]$, ma D è one-way

> il documento firmato $\langle U, m, f \rangle$ non può essere alterato se non da U, pena la non consistenza tra m e f (3)

> Poiché solo U può aver prodotto f, U non può ripudiare la firma (4)

> La firma f non è riutilizzabile su un altro documento $m' \neq m$ poiché è immagine di m (2)

Protocollo 2: Mensaggio m cifrato e firmato

1) Firma e cifratura: Mmittente U genera la firma $f = D(m, k_U[\text{priv}])$

$$\text{calcola } c = C(f, k_V[\text{pub}])$$

Invia la coppia $\langle U, c \rangle$ Destinatario

2) Decifrazione e verifica:

a) V riceve la coppia $\langle U, c \rangle$ e decifra il crittogramma:

$$D(c, k_v[\text{priv}]) = f$$

b) Successivamente cifra tale valore con la chiave pubblica di U:

$$c(f, k_u[\text{pub}]) = c(D(m, k_v[\text{priv}]), k_u[\text{pub}]) = m$$

c) Se m è significativo attesta l'identità di U

Algoritmo RSA per il protocollo 2

Cifrario RSA, con

- $\langle d_U, e_U, n_U \rangle$ chiavi di U
- $\langle d_V, e_V, n_V \rangle$ chiavi di V

utente U

- genera la firma del messaggio m: $f = m^{d_U} \pmod{n_U}$
- cifra f con la chiave pubblica di V: $c = f^{e_V} \pmod{n_V}$
- spedisce a V la coppia $\langle U, c \rangle$

utente V

- riceve la coppia $\langle U, c \rangle$ e decifra c: $c^{d_V} \pmod{n_V} = f$
- "decifra" f con la chiave pubblica di U: $f^{e_U} \pmod{n_U} = m$
- Se m è significativo, conclude che è autentico

Per la correttezza del procedimento è necessario che:

- $n_U < n_V$ perché risulti $f < n_V$ e f possa essere cifrata correttamente e spedita a V
- questo impedirebbe a V di inviare messaggi firmati e cifrati a U
- ogni utente stabilisce chiavi distinte per la firma e per la cifratura:
 - si fissa pubblicamente H molto grande, ad esempio $H = 2^{1024}$
 - chiavi di firma $< H$
 - chiavi di cifratura $< H$
- Il valore elevato di H assicura che tutte le chiavi possano essere scelte sufficientemente grandi, e quindi inattaccabili in modo esauriente.
- Il meccanismo di firma si presta tuttavia a diversi attacchi

basati sulla possibilità che un crittoanalista si procuri la firma di un utente su messaggi apparentemente privi di senso.

Attacco al protocollo 2

Vulnerabile se U invia un ack automatico quando riceve un messaggio cifrato e firmato.

Un crittoanalista attivo X può decifrare i messaggi inviati a U:

- X intercetta il crittogramma c firmato inviato da V a U, lo rimuove dal canale e lo rispedisce a U, facendogli credere che c sia stato inviato da lui.
- U spedisce automaticamente a X un ack
- X usa l'ack ricevuto per risalire al messaggio originale m applicando le funzioni del cifrario con le chiavi pubbliche di V e di U

Protocollo 3: Menaggio in cifrato e firmato in hash

1) Firma e cifratura:

a) U mittente U calcola $h(m)$ e genera la firma

$$f = D(h(m), k_u[\text{priv}])$$

b) Calcola separatamente $c = C(m, k_v[\text{pub}])$

c) Spedisce a V la tripla $\langle U, c, f \rangle$

2) Decifratura e verifica

a) V riceve $\langle U, c, f \rangle$

b) Calcola $m = D(c, k_v[\text{priv}])$

c) Calcola separatamente $h(m)$ e $c(f, k_u[\text{pub}])$

d) Se $h(m) = c(f, k_u[\text{pub}])$ conclude che il messaggio è autentico

OSS:

- richiede lo scambio di una maggiore quantità di dati, ma l'incremento è trascurabile
- la firma può essere calcolata più velocemente

Attacchi Man-in-the-Middle

• Debolezza dei protocolli:

- le chiavi di cifratura sono pubbliche e non richiedono un incontro diretto tra gli utenti per il loro scambio.
- Un crittoanalista attivo può intromettersi proprio in questa fase iniziale del protocollo, pregiudicando il suo corretto svolgimento.
- Attacco attivo chiamato **man in-the-middle**:

Il crittoanalista si intromette nella comunicazione tra U e V, si comporta agli occhi di U come se fosse V, si comporta agli occhi di V come se fosse U, intercettando e boccando le comunicazioni di U e V

Il crittoanalista X devia le comunicazioni che provengono da U e V e le dirige verso se stesso:

- U richiede a V la sua chiave pubblica (attraverso e-mail, pagina web, ...)
- X intercetta la risposta contenente $k_v[\text{pub}]$ e la sostituisce con la sua chiave pubblica $k_x[\text{pub}]$.
- X si pone in ascolto in attesa dei crittogrammi spediti da U a V, cifrati mediante $k_x[\text{pub}]$.
- X rimuove dal canale ciascuno di questi crittogrammi, lo decifra, lo cifra mediante $k_v[\text{pub}]$ e lo rispedisce a V.
- U e V non si accorgono della presenza di X se il processo di intercettazione e rispedizione è sufficientemente veloce da rendere il relativo ritardo apparentemente attribuibile alla rete.

Certification Authority

È un ente per la certificazione di validità delle chiavi.

Autentica l'associazione $\langle \text{utente}, k_{\text{pub}} \rangle$ emettendo un certificato digitale che contiene:

- una indicazione del suo formato (numero di versione)
- il nome della CA che lo ha rilasciato
- un numero seriale che lo individua univocamente all'interno della CA emittente
- la specifica dell'algoritmo usato dalla CA per creare la firma elettronica
- il periodo di validità del certificato (data di inizio e data di fine)
- il nome dell'utente a cui questo certificato si riferisce e una serie di informazioni a lui legate
- un'indicazione del protocollo a chiave pubblica adottato da questo utente per la cifratura e la firma: nome dell'algoritmo, suoi parametri, e chiave pubblica dell'utente
- firma della CA eseguita su tutte le informazioni precedenti.

Un utente U richiede $k_v[\text{pub}]$ di un utente V alla CA, che risponde inviando a U il certificato digitale cert_v di V.

Poiché U conosce $k_{\text{ca}}[\text{pub}]$, può controllare l'autenticità del certificato cert_v verificandone il periodo di validità e la firma prodotta dalla CA stesso.

Per evitare un colpo di bottiglia nelle comunicazioni con il CA, ogni utente mantiene localmente al sistema una copia del proprio certificato cert_v e una copia di $k_{\text{ca}}[\text{pub}]$.

Protocollo 4 : Menaggio cifrato, firmato in hash e certificato

1) Firma, cifratura e certificazione:

- a) Il mittente si procura il certificato cert_v di V
- b) Calcola $h(m)$ e genera la firma $f = D(h(m), k_v[\text{priv}])$
- c) Calcola il cattogramma $c = C(m, k_v[\text{pubb}])$
- d) spedisce la tripla $\langle \text{cert}_v, c, f \rangle$

⁶ contiene $k_u[\text{pub}]$ e h utilizzata

2) Verifica del certificato:

Il destinatario V riceve la tripla $\langle \text{cert}_U, c, f \rangle$ e verifica l'autenticità di cert_U utilizzando $K_{CA}[\text{pub}]$

3) Decifrazione e verifica della firma:

a) V decifra il citoogramma c mediante la sua chiave privata e ottiene

$$m = D(c, K_V[\text{priv}])$$

b) V verifica l'autenticità della firma apposta da U su m controllando che risulti

$$c(f, K_U[\text{pub}]) = h(m)$$

Conclusioni:

- Il punto debole di questo meccanismo è rappresentato dai certificati revocati, i.e., non più validi
- Le CA mettono a disposizione degli utenti un archivio pubblico contenente i certificati non più validi
- La frequenza di controllo di questi certificati e le modalità della loro comunicazione agli utenti sono cruciali per la sicurezza
- Attacchi man-in-the-middle possono essere evitati se si stabilisce un incontro diretto tra utente e CA nel momento in cui si istanzia il sistema asimmetrico dell'utente

Protocollo SSL (Secure Socket Layer)

È stato progettato in modo da permettere la comunicazione tra computer che non conoscono le rispettive funzionalità.

Garantisce:

- 1) Confidentialità: Trasmissione crittografata tramite un sistema ibrido

Cifrario asimmetrico per scambio delle chiavi;
Cifrario simmetrico per criptare i messaggi

2) Autenticazione dei messaggi:

- a) Attraverso un cifrario asimmetrico
- b) Attraverso uno di certificati e MAC

ORGANizzato in 2 livelli:

1) SSL Record: Comune al lvl di trasporto e ha l'obiettivo di
incapsulare i dati spediti dai protocolli di lvl superiori,
assicurando confidantialità e integrità

realizza fisicamente questo canale (meccanismo di rete):

utilizza la cipher suite stabilita da SSL Handshake per cifrare e autenticare i blocchi di dati, prima di spedirli attraverso il protocollo di trasporto sottostante

2) SSL Handshake: Responsabile dell'instaurazione e del mantenimento dei parametri usati da SSL record.

Permette l'autenticazione, Negoziazione algoritmi di cifratura e firma, stabilire le chiavi.

crea un canale **sicuro, affidabile e autenticato** tra utente e sistema, entro il quale SSL Record fa viaggiare i messaggi divisi in blocchi opportunamente cifrati e autenticati.

L'handshake è diviso in passi:

1. Utente: client hello

U manda a S un messaggio di *client hello*, con cui

- richiede la creazione di una connessione SSL,
- specifica le prestazioni di "sicurezza" che desidera siano garantite durante la connessione
 - cifrari e meccanismi di autenticazione che U può supportare
 - e invia una sequenza di byte casuali.

2. Sistema: server hello

- Il sistema S
 - riceve il messaggio di *client hello*
 - seleziona una *cipher suite* che anch'esso è in grado di supportare
 - invia a U un messaggio di *server hello* che specifica la sua scelta e contiene una nuova sequenza di byte causali.
 - Se U non riceve il messaggio di *server hello* interrompe la comunicazione.

3. Sistema: invio del certificato

- S si autentica con U inviandogli il proprio certificato digitale
 - e gli eventuali altri certificati della catena di certificazione dalla sua CA fino alla CA radice
- Se i servizi offerti da S devono essere protetti negli accessi, S può richiedere a U di autenticarsi inviando il suo certificato digitale.
 - avviene raramente: la maggior parte degli utenti non ha un certificato personale,
 - il sistema dovrà accertarsi dell'identità dell'utente in un secondo tempo.

5. Utente: autenticazione del sistema

Per accettare l'autenticità del certificato ricevuto da S, l'utente U

- controlla che la data corrente sia inclusa nel periodo di validità del certificato
- che la CA che ha firmato il certificato sia tra quelle "fideate"
- e che la firma apposta dalla CA sia autentica

6. Utente: invio del pre-master secret e costruzione del master secret

- Il *pre-master secret* viene combinato da U con alcune stringhe note e con i byte casuali presenti sia nel messaggio di *client hello* che in quello di *server hello*.
- A tutte queste sequenze U applica delle funzioni hash one-way (SHA-1 e MD5) secondo una combinazione opportuna.
- Ottiene così il *master secret*

7. Sistema: ricezione del pre-master secret e costruzione del master secret

- Sia U che S conoscono:
 - Il numero casuale che U ha mandato a S (invia in chiaro)
 - Il numero casuale che S ha mandato a U (invia in chiaro)
 - Il *premaster secret* (invia cifrato)
- U e S calcolano il master secret.

- ↓
- Il *master secret* è utilizzato da U e da S per costruire una propria tripla contenente
 - la chiave segreta da adottare nel cifrario simmetrico
 - la chiave per l'autenticazione del messaggio (costruzione del MAC)
 - la sequenza di inizializzazione per cifrare in modo aperiodico messaggi molto lunghi (usata e.g., come valore iniziale nel CBC).

4. Sistema: server hello done

S invia il messaggio **server hello done** con cui sancisce la fine degli accordi sulla *cipher suite* e sui parametri crittografici a essa associati.

6. Utente: invio del pre-master secret e costruzione del master secret

L'utente U

- costruisce un *pre-master secret* costituito da una nuova sequenza di byte casuali (i.e., genera un numero segreto)
- lo cifra con il cifrario a chiave pubblica su cui si è accordato con S
- spedisce il relativo crittogramma a S

(e.g., U cifra il premaster secret con RSA, usando la chiave pubblica presente nel certificato di S)

7. Sistema: ricezione del pre-master secret e costruzione del master secret

- S decifra il crittogramma contenente il *pre-master secret* ricevuto da U
- S calcola il *master secret* mediante le stesse operazioni eseguite da U al passo 6 (dispone delle stesse informazioni).

8. Utente: invio del certificato (opzionale)

- Se all'utente U viene richiesto un certificato (passo 3) ed egli non lo possiede il sistema interrompe l'esecuzione del protocollo.
- Altrimenti U invia il proprio certificato con allegate una serie di informazioni firmate con la sua chiave privata, tra cui
 - il *master secret*
 - tutti i messaggi scambiati fino a quel momento (*SSL-history*)

8. Utente: invio del certificato (opzionale)

- S controlla il certificato di U e verifica autenticità e correttezza della SSL-history.
- In presenza di anomalie, la comunicazione con U viene interrotta.

9. Utente/Sistema: messaggio *finished*

- La costruzione avviene in due passi:
 - all'inizio si concatenano il *master secret*, tutti i messaggi di *handshake* scambiati fino a quel momento e l'identità del mittente (U o S)
 - la stringa ottenuta viene trasformata applicando le funzioni SHA-1 e MD5: si ottiene una coppia di valori che costituisce il messaggio *finished*.

OSSERVAZIONI:

Il canale sicuro approntato dal protocollo *SSL handshake* viene realizzato dal protocollo *SSL record*.

- I dati sono frammentati in blocchi.
- Ciascun blocco viene
 - numerato, compresso e autenticato mediante l'aggiunta di un MAC
 - cifrato mediante il cifrario simmetrico su cui U e S si sono accordati
 - trasmesso dall'*SSL record* utilizzando il protocollo di trasporto sottostante.

Sicurezza

Nei passi di *hello* i due partner creano e si inviano due sequenze casuali per la costruzione del *master secret*, che risulta così diverso in ogni sessione di SSL.

Un crittoanalista non può riutilizzare i messaggi di *handshake* catturati sul canale per sostituirsi a S in una successiva comunicazione con U.

9. Utente/Sistema: messaggio *finished*

- È il primo messaggio protetto mediante il *master secret* e la *cipher suite* su cui i due partner si sono accordati.
- Il messaggio viene prima costruito da U e spedito a S, poi costruito da S e spedito a U:
 - nei due invii la struttura del messaggio è la stessa, ma cambiano le informazioni in esso contenute.

9. Utente/Sistema: messaggio *finished*

- Il messaggio è diverso nelle due comunicazioni perché S aggiunge ai messaggi di *handshake* anche il messaggio *finished* ricevuto da U.
- Il destinatario della coppia (S o U) non può invertire la computazione precedente in quanto generata da funzioni one-way:
 - ricostruisce l'ingresso delle due funzioni SHA-1 e MD5, le ricalcola e controlla che la coppia generata coincida con quella ricevuta, come dimostrazione che la comunicazione è avvenuta correttamente.

Il destinatario esegue un procedimento inverso sui blocchi ricevuti:

- decifra e verifica la loro integrità attraverso il MAC
- decomprime e riassembra i blocchi in chiaro
- li consegna all'applicazione sovrastante.

MAC dei Blocchi di dati

- *SSL record* numera in modo incrementale ogni blocco di dati e autentica il blocco attraverso un MAC.
- Per prevenire la modifica del blocco da parte di un crittoanalista attivo, il MAC viene calcolato come immagine hash one-way di una stringa costruita concatenando
 - il contenuto del blocco,
 - il numero del blocco nella sequenza,
 - la chiave del MAC
 - alcune stringhe note e fissate a priori.

Dato che i MAC sono cifrati insieme al messaggio, un crittoanalista non può alterarli senza avere forzato prima la chiave simmetrica di cifratura:

un attacco volto a modificare la comunicazione tra i due partner è difficile almeno quanto quello volto alla sua decriptazione.

Il sistema è autenticato

- Il canale definito da SSL **handshake** è immune da attacchi attivi **man-in-the-middle** poiché il sistema S viene autenticato con un certificato digitale.
- L'utente U può comunicare il **pre-master secret** al sistema S in modo sicuro attraverso la chiave pubblica presente nel certificato di S.
- Solo S può decifrare quel crittogramma e costruire il **master secret**, su cui si fonda la costruzione di tutte le chiavi segrete adottate nelle comunicazioni successive.
- Solo il sistema S, quello cui si riferisce il certificato, potrà quindi entrare nella comunicazione con l'utente U.

Generazione delle seq. casuali

- Le tre sequenze casuali generate da U e da S e comunicate nei messaggi di *client hello*, *server hello*, *pre-master secret* sono usate per creare il **master secret**, quindi per generare le chiavi segrete di sessione.
- La sequenza corrispondente al *pre-master secret* viene generata da U e comunicata per via cifrata a S.
- La non predicitività di questa sequenza è cruciale per la sicurezza del canale SSL:
 - una sua cattiva generazione renderebbe il protocollo molto debole.

Messaggio finished

- Contiene tutte le informazioni scambiate nel corso dell'**handshake**
- Scopo: consente un ulteriore controllo sulle comunicazioni precedenti per garantire che
 - queste siano avvenute correttamente
 - U e S dispongano dello stesso *Master Secret*
 - che la comunicazione non sia stata oggetto di un attacco attivo

Protocollo a conoscenza - zero (Zero Knowledge)

Gli utenti sono il

1) Prover = Dimostratore = P

2) Verifier = Verificatore = V

Il protocollo consiste in delle sfide che P deve superare per convincere V.

P deve convincere V senza fornire altre info tranne la conoscenza di una capacità.

La sfida viene ripetuta per N volte scelto da V. Per convincere V, P deve superare tutte le sfide.

L'utente può essere autenticato

Il certificato di U (se richiesto) e la sua firma apposta sui messaggi scambiati nel protocollo (*SSL-history*) consentono a S di verificare che U sia effettivamente quello che dichiara di essere e che i messaggi siano stati effettivamente spediti da esso.

Se ciò non si verifica, S deduce che il protocollo è stato alterato (casualmente o maliziosamente con un attacco *man-in-the-middle*) e interrompe la comunicazione.

- L'opzionalità dell'autenticazione dell'utente ha reso l'SSL molto diffuso nelle transazioni commerciali via Internet:
 - per gli utenti la necessità di certificazione può costituire un ostacolo pratico ed economico
- L'utente può essere autenticato con altri metodi (*login e password*, # carta di credito)

OSS: Nel caso P fosse un "disonesto", la prob. di ingannare V è la probabilità di prevedere il bit generato da V per verificare la correttezza, quindi di $(\frac{1}{2})^n$ - prob. di generare il bit corretto per le volte

Il protocollo deve avere tre proprietà:

- 1) Completezza: Se l'affermazione è vera, quindi P è onesto, riesce sempre a convincere V.
- 2) Correttezza: Se l'affermazione è falsa, quindi P è disonesto, la probabilità di convincere V è molto bassa.
 $\text{prob} \leq (\frac{1}{2})^n$ con n scelto da V
- 3) Conoscenza-zero: Se l'affermazione è vera, non esiste V (onesto/disonesto) che può acquisire alcuna informazione salvo la sua veridicità.cioè non ottiene informazioni in più su P.

Protocollo di identificazione di Fiat-Shamir:

È basato sulla difficoltà del calcolo della radice quadrata in modulo n, con n composto

$$t = s^2 \pmod{n} \quad n \text{ composto}$$

V: Conosce t e n

P: Convince V di conoscere la radice quad di t (\sqrt{t})

OSS: P deve convincere V in tempo polinomiale!

Preparazione:

- 1) P sceglie p e q primi
- 2) Calcola $n = p \cdot q$, e sceglie $s < n$
- 3) Calcola $t = s^2 \bmod n$
- 4) P rende nota la coppia $\langle t, n \rangle$ (chiave pubblica)
- 5) P mantiene segreta la triple $\langle p, q, s \rangle$ (chiave segreta)

Protocollo:

Ripeti k volte

- 1) V chiede a P di iniziare un iterazione
- 2) P genera un intero $u < n$ casuale
 - a) Calcola $u = u^2 \bmod n$
 - b) Comunica u a V
- 3) V genera $e \in \{0, 1\}$ casuale
Comunica e a P

- 4) P calcola $z = n \cdot s^e \bmod n$
Comunica z a V
 - 5) V calcola $x = z^2 \bmod n$
 $\text{if } (x == u^{t^e} \bmod n)$ Ripete da 1
 else STOP
 - 1) Se $e = 0$
 $z = n$
 - 2) Se $e = 1$
 $z = n \cdot s \bmod n$
- P non identificato
- P identificato
- 3) $(n s^e)^2 = n^2 (s^e)^2 = u \cdot (s^2)^e = u^e$

Completenza:

$$e=0 \Rightarrow x = u^{t^e} \bmod n = u \bmod n$$

Se
 $e=1 \Rightarrow x = z^2 \bmod n = (n s^e)^2 \bmod n = u^t \bmod n$

P rispetta tutte le sfide, V accetta la dimostrazione

Connessione:

Poiché è generato casualmente, le previsioni di P sono connette con prob. 1/2 ad ogni iterazione.

Criptografia Quantistica

Caratteristiche Meccanica quantistica:

- 1) **Sovraposizione:** Proprietà di un sistema quantistico di trovarsi in diversi stati contemporaneamente
- 2) **Decerenza:** La misurazione di un sistema quantistico disturba il sistema e il sistema disturbato perde la sovrapposizione degli stati e collassa in uno singolo
- 3) **No-cloning:** Impossibilità di duplicare un sistema conservando nello copy lo stato quantistico dell'originale (senza misurarlo).
È impossibile copiare uno stato quantistico non noto.
- 4) **Entanglement:** Possibilità che due o più elementi si trovino in stati quantistici correlati tra loro in modo che, pur se portati a grande distanza, mantengano la correlazione

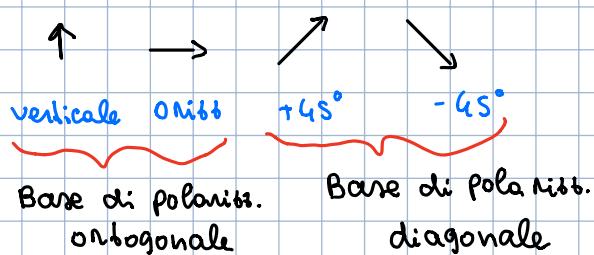
Fotone

Il fotone è una particella elementare che costituisce la luce corpuscolare.

Ogni fotone ha uno stato di **POLARIZZAZIONE**.

La polarizzazione è il piano di oscillazione del campo elettrico

4 stati di polarizzazione: (servono ad Alice per codificare il bit 0,1)



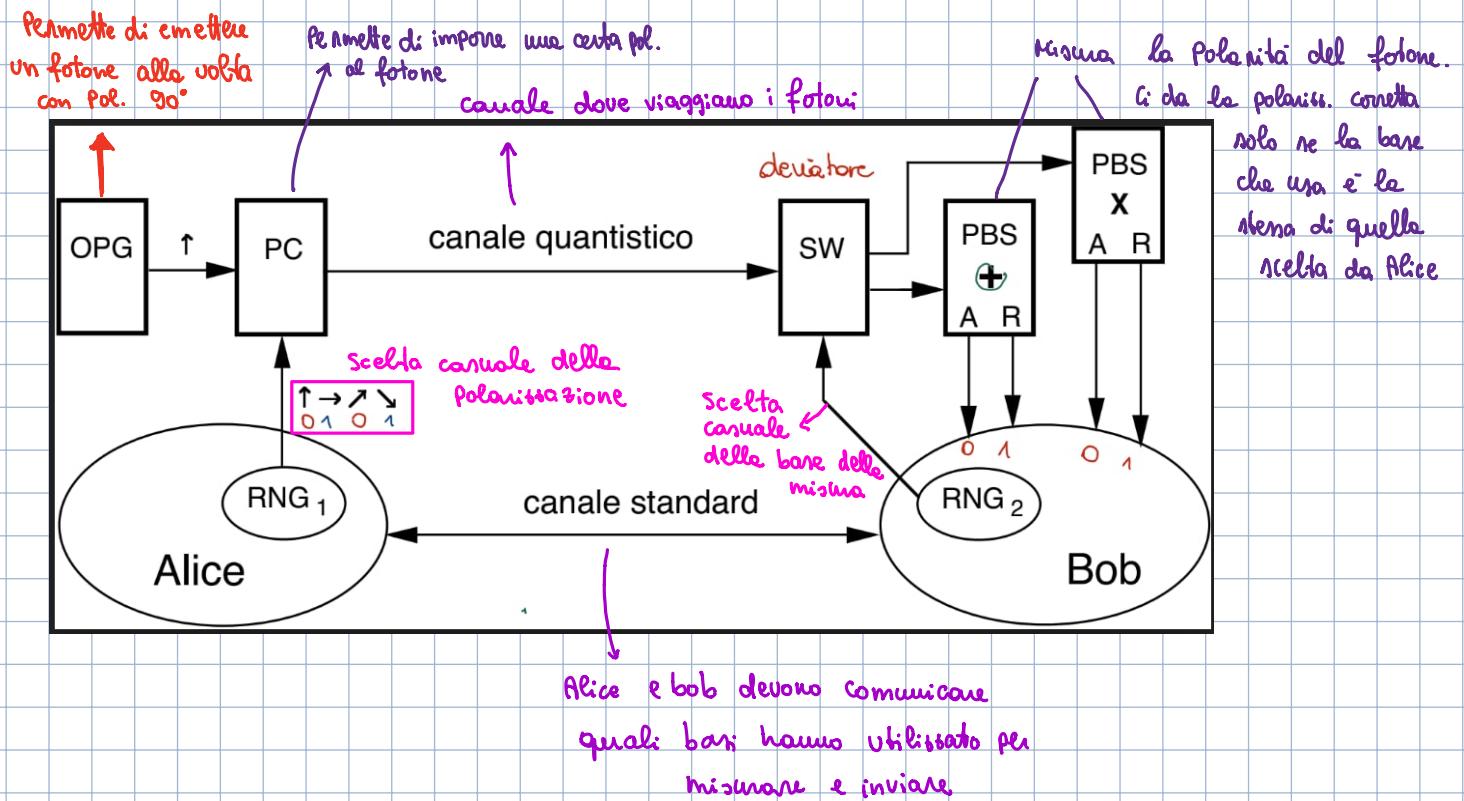
+

X

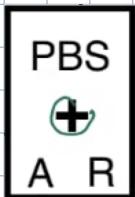
O | O | = bit associati ad ogni polarizzazione
 (Scelta arbitraria)

Se riceghiamo una base solo saremo in grado di misurare con precisione lo stato di polarizzazione del fotone.

Se invece uso le due basi, non posso distinguere i 4 casi, posso solo misurare tra i 2 stati ortogonali nella stessa base.

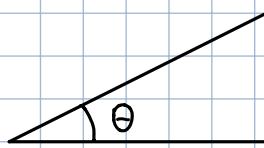


PBS: Beam Splitter. Devia il fotone verso una tra le due uscite A e R.



A: Assorbimento

R: Riflessione



F: pol. del fotone

S: Azi di pol. del PBS

fornibili uscite, con in input un fotone, sono A e R:

1) Caso A: Il fotone viene inviato all'uscita A con prob. $\cos^2 \theta$.

Ero prense la polarizzazione S dello strumento PBS

2) Caso R: Il fotone viene inviato all'uscita R con prob. $\sin^2 \theta$.

Ero prense la polarizzazione ortogonale alla pol. S dello strumento PBS

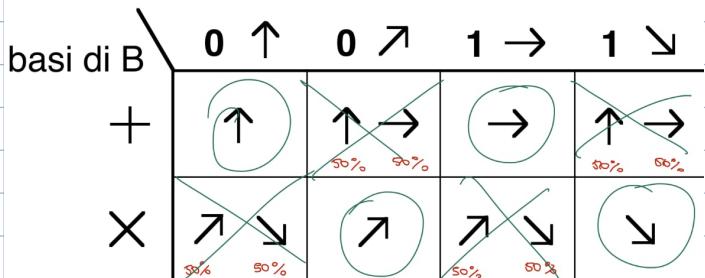
$\Theta = 0^\circ$ ($F = S$) Il fotone conserva la polarizzazione ed esce da A

$\Theta = 90^\circ$ ($F \perp S$) Il fotone e' conserva la polarizzazione ed esce da R

$\Theta = \pm 45^\circ$ Il fotone esce con pari prob. da A o R e la sua pol. cambia (S, S^\perp)

la lettura attraverso il PBS ha distrutto lo stato quantistico precedente

bit e fotone inviato da A



Quando il flusso di fotoni di Alice verra' bob fermata, Alice ha preso nota delle basi che ha utilizzato e dei fotoni che ha inviato.

Bob stesso: fotoni, fa delle scelte casuali delle basi e avra' un po' di bit.

Prende nota delle basi utilizzate e le misure.

La comunicazione sul canale standard bob dice ad Alice le basi che ha utilizzato per le misure.

Alice risponde dicendo quali basi sono corrette.

Dopo cio', bob e Alice sanno dove la comunicazione e' andata correttamente (base Bob - base Alice) e quali no (Base bob * base Alice).

Nei casi di base * i bit vengono scartati.

Prob. che bob abbia scelto la stessa base di alice e' $\frac{1}{2}$

$$P(b_{\text{bob}} = b_{\text{Alice}}) = \frac{1}{2}$$

Quindi circa la metà vengono scartate e l'altra metà ha la chiave.

Protocollo BB 84

Si basa lo scambio della chiave sull' invio di fotoni polarizzati

- 4) Alice: Invia una sequenza S_A di bit (codificati nello polarizz. dei fotoni) sul canale quantistico.
 - 2) Bob: Interpreta S_A con le basi che ha ricevuto casualmente e ottiene S_B .
 - 3) Bob: Comunica sul canale standard ad Alice le basi scelte.
 - 4) Alice: Comunica quali sono quelle comuni con le sue.

Al termine di questa comunicazione, Alice e Bob forniscono una sottosig.

$S_A^1 = S_B^1$ formata dai bit codificati e decodificati con bari comuni.

$$|S_A'| = |S_B'| \approx \frac{|S_A|}{2}$$

- 5) Alice e Bob : Sacrificano una porzione s_A'', s_B'' di s_A' e s_B' ,
 in quantita' predefinita, comunicandole sul canale standard.
 Se $s_A'' \neq s_B'' \Rightarrow$ A e B interrompono la comunicazione
 Perche ci sono state interruzioni / Malfunzionamento

Esempio:

A 1 0 1 1 1 1 0 0 - - - - → bit che alice vuole inviare = S_A
 A + X + X X + X - - - - → Bari scelte
 → pol. fotoni
 → Bari di Eve
 → lettura di Eve
 → Bit letti = S_E
 + + + X + X + - - -
 → 1 1 1 1 0 0 1

Scarto : bit con bani diverse

$$\begin{array}{ccccccccc}
 B & + & X & + & + & + & \times & \times & - \\
 \downarrow & \nearrow & \downarrow & \rightarrow & 1 & \rightarrow & \rightarrow & \uparrow & \\
 1 & 0 & 1 & 0 & 1 & 1 & 0 & & \\
 \downarrow & \nearrow & \downarrow & \uparrow & \uparrow & \nearrow & \rightarrow & & \\
 1 & 1 & 1 & 0 & 0 & 0 & 1 & & \\
 \end{array}$$

senza Eve

con Eve

$S_A = S_B = 1010$

→ Bari scelte
 → lettura di bob
 → Bit lett: = SB
 → Come cambia la misura di Bob
 con Eve nel mezzo delle com.

CRIPTOANALISTA: Ye crittoanalista mi puoi mettere nel mezzo delle comunicazione ma se sbaglia la base con cui misurare il fotone, il fotone perde la polarizzazione originale e Alice e Bob sono in grado di verificare questa anomalia.

Questo deriva dalla proprietà della no-duplication del fotone, cioè non si può duplicare uno stato non conosciuto. Alice e Bob se ne accorgono andando a confrontare s'_A e s'_B .

Attenzione: Se il crittoanalista interviene dovrebbero discordare circa

$$\frac{1}{4} |S_A| \text{ bit}$$

QBER (quantum bit error Rate)

è la % prevedibile di bit errati dovuti per esempio dell'apparato sperimentale

Se nel confronto s''_A e s''_B , se la % di errore è:

> QBER \rightarrow Indica che c'è stata un'intuizione di Eve quindi la chiave viene scartata

\leq QBER \rightarrow Usano un codice a correzione di errori per ricostruire una chiave corretta

Eseguo che Eve potrebbe ottenere delle info su alcuni bit, abbiamo che Alice e Bob usano delle funzioni hash crittografiche con input $s'_1 - s''_1$ e $s'_2 - s''_2$ per calcolare la chiave.

Protocollo BB84 (Soggetto a interferenza)

Canale quantistico

$S_A[1:n] \rightarrow$ Sequenza iniziale di bit da cui viene estratta la chiave
(Rappresentata con un codice a correzione di errori)

for $i=1$ to n

Alice: Sceglie una base a caso, Codifica $S_A[:]$, invia il fotone a bob

Eve (Se presente): Intercastra il fotone, lo mima con una base, lo invia a bob e calcola $S_E[:]$

bob: Sceglie una base a caso, interpreta il fotone ricevuto,
costruisce $S_B[:]$

Canale Standard

QBER: % di errori dovuti al rumore (errori sperimentali)

h : funzione hash crittografica

Bob: Comunica ad Alice la sequenza di basi scelte

Alice: Comunica a Bob le basi comuni

Alice e bob:

1) Estraggono S_A' e S_B' secondo le basi comuni e 2 sottoseq. S_A'' e S_B''

2) Si scambiano S_A'' e S_B'' .

Se % di bit + in S_A'' e $S_B'' > QBER \Rightarrow$ STOP

3) Else: Si calcolano $S_A' \setminus S_A''$ e $S_B' \setminus S_B''$ e li Codificano con il codice di correzione ottenendo S_C (eve potrebbe conoscere dei bit)

4) Alice e Bob calcolano $h = h(S_C)$ e usano h come chiave