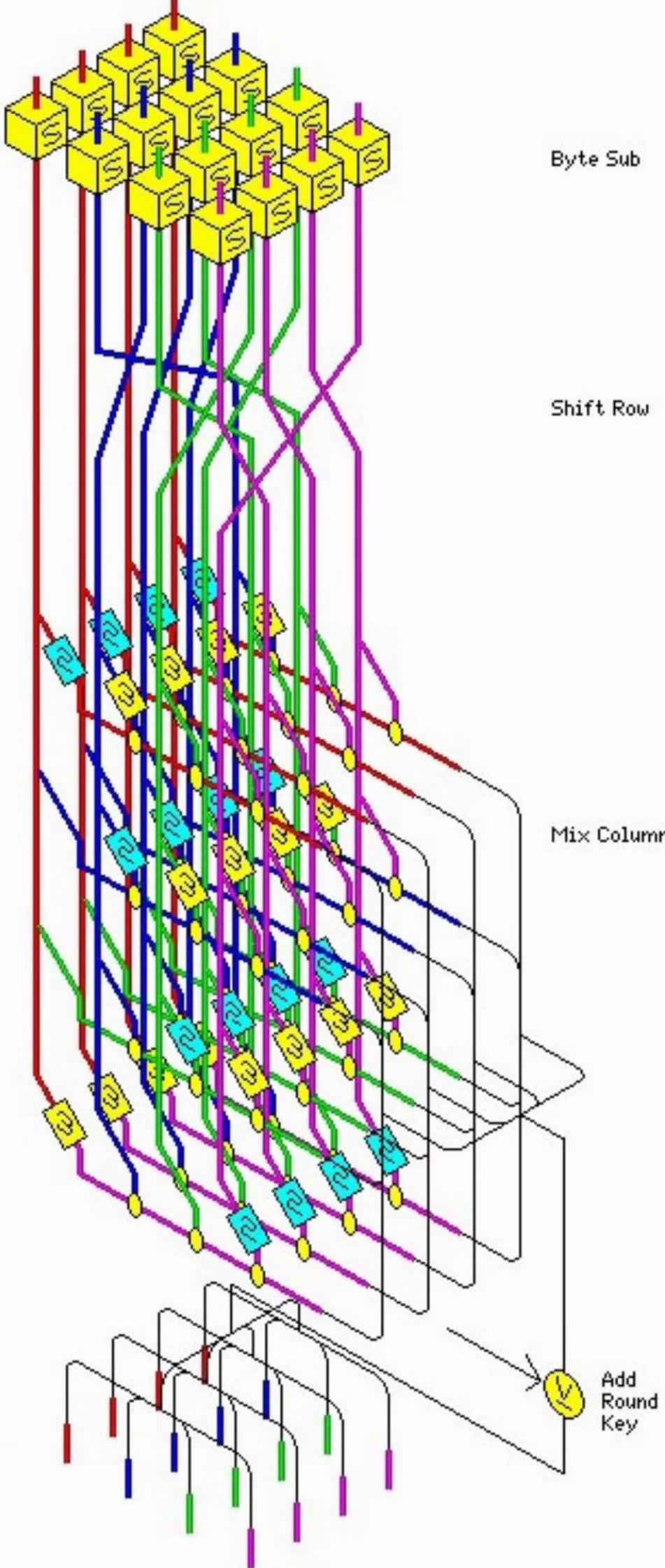


C
R
I
T
T
O
G
R
A
F
I
A



Introduzione

• TERMINOLOGIA

Crittografia = metodi di cifratura } Crittologia = studio di communication su canali
Crittanalisi = metodi di analisi non sicuri + rischi derivanti

↓

messaggio in chiaro : m

messaggio cifrato : C → incomprendibile per crittanalista ma decifrabile
in tempo polinomiale da destinatario

cifratura = funzione : C : MSG → CRITTO decifrare D : CRITTO → MSG

↓ deve essere iniettiva

D(C(m)) = m : il crittogramma deve essere definito
attraverso il cifrario

Il cifrari vengono classificati in base alle informazioni che vengono mantenute segrete: per uno ristretto: funzioni di cifratura e decifrare vengono mantenute segrete → comunicazione diplomatiche e militari

per uno generale: viene mantenuta segreta soltanto la chiave mentre le regole sono pubbliche

⚠ ATT: il nemico conosce il sistema ed il cifrario deve comunque rimanere sicuro

↓

Si usa una chiave segreta: diversa per ogni coppia di utenti ed invertibile come parametro nelle funzioni C e D → la chiave non deve essere grande per permettere di estrarre informazione. La chiave non deve essere grande per impedire di estrarre informazione → facile farlo con la chiave in quanto è un'informazione piccola ed è sufficiente cambiarla se scoperti

La sicurezza dipende dalla chiave: il numero delle chiavi deve essere molto ampio in modo tale che il crittanalista non possa trovarle e la chiave segreta deve essere scelta in modo casuale

↓ implicazioni

1. Un cifrario complicato non è necessariamente sicuro
2. Mai sottovalutare la bravura del crittanalista

• METODI DI ATTACCO

PASSIVO: Il crittanalista si limita ad ascoltare la comunicazione

ATTIVO: Il crittanalista modifica il contenuto dei messaggi

↓ obiettivi:

Fortificare del sistema Tipi attacchi solo testo cifrato: il crittanalista rileva una serie di crittogrammi

testo in chiaro noto: il crittanalista conosce alcune copie crittogramma - messaggio

testo in chiaro nascosto: in questo caso le coppie vengono scelte dal crittanalista

testo cifrato nascosto: in questo caso il crittanalista sceglie i crittogrammi da decifrare

bruteforce attack: vengono provate tutte le chiavi

Il cifrario è più sicuro se non viene violato da metodi di complottà inferiore o quello di forza bruta

divelli di successo dato da infruttuazione ottenuta → anche l'informazione parziale può essere molto (ad esempio nel linguaggio naturale)

↓ Tipi CIFRARI

Cifrari perfetti: usati in condizioni estreme e non adatti a cifratura di messaggi

↓
Cifrari dedicati sicure: Adatti alla cifratura di messaggi in chiaro e necessari algoritmi molto complessi per la loro violazione → impossibile del pdv pratico riservate ma non classificate

↓
Advanced Encryption Standard: standard per comunicazioni e quindi in ambito commerciale → chiavi lunghe 128/256 bit divise a blocchi

↓
Le chiavi vengono decise dal terminali usato per la comunicazione all'inizio di ogni sessione → Scambio chiave tramite crittografia a chiave pubblica: è possibile scambiare le chiavi in chiaro

↓
CIFRARI SIMMETRICI: chiave cifrazione = chiave decifrazione

↓
CIFRARI A CHIAVE PUBBLICA: Tutti possono mandare messaggi a destinatario ma solo lui può decifrarli con la sua chiave privata

↓ 2 chiavi

Kpub: per cibrare, nota a tutti → $c = C(m, K_{pub})$: tutto pubblico

Kpriv: per decifrare, nota solo al destinatario → $m = D(c, K_{priv})$: tempo polinomiale

↓ Sistema assimmetrico

↓ proprietà funzione di cifrazione

↓
One-way trap-door: calcolare c deve essere facile (tempo polinomiale) mentre decifrare deve essere complicato se non si ha la chiave privata

↓

Vantaggi: su adattar bene ad un sistema molti ad uno. Se abbiano m utenti che vogliono inviare messaggi ad uno, le chiavi sono $2n$ anziché $n(n-1)/2$ nel caso simmetrico

↓
Svantaggi: molto lenti a causa della elaborazione

↓
Vengono spesso usati cifrari ibridi

↓
Sono esposti ad attacchi di tipo chosen plain-text: il crittanalista si costruisce il crittogramma del messaggio desiderato, si mette in circolo sul canale e vede quando giunge il messaggio desiderato

↓

Viene superato con cifrari ibridi: con il cifrario a chiave pubblica manda la chiave. Questa è corta (256 bit per AES) e impareggiabile (l'analista dovrebbe cifrare 2^{256} chiavi e rimanere in circolo per tutte). Una volta scambiate le chiavi, queste vengono utilizzate all'interno di un cifrario simmetrico: lo scambio dei messaggi lunghi è adesso veloce

↓ Applicationi

↓
Identificazione: sistema dove accettare identità di chi richiede di accedere ai suoi servizi

↓
Autenticazione: permette di accertarsi che il messaggio ricevuto non sia stato

manomesso

Funza digitale: Garantisce identificazione, certificazione ed infine la funza non può essere falsificata

• RAPPRESENTAZIONE MATEMATICA di OGGETTI

Ricette di associare ad ogni oggetto una stringa che lo identifica univocamente → Sequenze in cui conta l'ordine delle lettere → Alfabeto: insieme di caratteri o simboli: a oggetti diversi corrispondono sequenze diverse e il numero di oggetti rappresentabili deve essere infinito
↓

Alfabeto Γ , $|\Gamma|=s$ con N oggetti da rappresentare: $d(s, N)$ è la lunghezza della sequenza più lunga che rappresenta un oggetto dell'insieme eventuale $\text{dun}(s, N)$ è il valore minimo di $d(s, N)$ su tutte le rappresentazioni possibili.
L'efficienza della rappresentazione aumenta tanto più $d(s, N)$ si avvicina a $d_{\min}(s, N)$

↓ Vantaggi della lunghezza

$s=1$, $\Gamma = \{0\} \rightarrow d_{\min}(1, N) = N$: l'unico modo di differenziare è la lunghezza → sbavorevole

$s=2$, $\Gamma = \{0, 1\} \rightarrow \forall k > 1$, 2^k sequente di lunghezza k . Il numero totale di sequenze lunghe da 1 a k è dato da $\sum_{i=1}^k 2^i = 2^{k+1} - 2$. Avendo N oggetti da rappresentare: $2^{k+1} - 2 \geq N$ da cui $k \geq \log_2(N+2) - 1$
Ho migliorato molto

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

È possibile costruire N sequenze differenti con $\lceil \log_2 N \rceil$ caratteri tutte lunghe $\lceil \log_2 N \rceil$ e questo è vantaggioso per la concatenazione in quanto non bisogna di separatori → Rappresentazione efficiente se ha un numero di caratteri minimi pari al logaritmo della cardinalità N dell'insieme
↓

da rappresentazione posizionale degli interi è efficiente perché permette una riduzione logaritmica tra il valore del numero e la sua rappresentazione (d)

• TEORIA DELLA CALCOLABILITÀ

Si occupa dei problemi che possono risolvere i problemi di calcolo e delle loro limitazioni → esistono problemi non risolvibili per via algoritmica: problemi indecidibili → Oggetto: problemi computazionali che hanno quindi una formulazione matematica e soluzione algoritmica
↓ Tipologie

Non decidibili: non risolti per via algoritmica

Decidibili ↳ Trattabili: risoluzione polinomiale

Redi ↓ ↳ Non trattabili: risoluzione esponenziale

Calcolabilità divide in decidibili / non decidibili, con plessità in facili / difficili
↓ esistenza problemi indecidibili

Numero problemi >> numero algoritmi: conoscenza umana trasmessa su sequenze finite di simboli → tante ma numerabili mentre invece i problemi possono essere infiniti in corrispondenza con il continuo (ω)
↓

Insieme numerabili: possibile corrispondenza con naturali. 2 insiemni hanno lo stesso numero di elementi se possono avere una in corrispondenza biunivoca tra di loro. Gli insiemni numerabili possono avere scatti come lista sfaticando la corrispondenza biunivoca con i naturali disegni → \mathbb{N} , \mathbb{Z} , \mathbb{P} , \mathbb{Q} , stringhe finite di simboli alfabeto finito → Costruzione delle sequenze (ω): devo raggiungere qualche sequenza in un numero finito di passi (distanza finita da inizio

elenco) → Ordinamento canonico: lunghezza crescente e poi ordinamento alfabetico → una sequenza s arbitraria si trova tra quelle di 1s i caratteri, in posizione alfabetica tra queste → Ad una sequenza arbitraria corrisponde il numero che ne indica la posizione nell'elenco. Ad un numero naturale n inoltre corrisponde la sequenza che occupa la posizione n nell'elenco. Tutto ciò è possibile poiché le sequenze sono di lunghezza finita anche se illimitata (Ad esempio a priori, la sequenza di lunghezza > d). Se le sequenze avessero lunghezza ∞ l'insieme non sarebbe numerabile. Per quanto riguarda gli insiemini non numerabili abbiamo ad esempio IR e le funzioni → il problema possono essere visti come funzioni:

Dimostrazione ↓

$$f: \begin{matrix} N^k \\ \text{dati} \end{matrix} \rightarrow \begin{matrix} N^3 \\ \text{risultato} \end{matrix}$$

$F = \{ \text{funzioni } f \mid f: N \rightarrow \{0, 1\}^N \}$. Possiamo rappresentarle con una sequenza ∞ di valori: $x_0 \ 1 \ 2 \ 3 \ 4 \dots n \dots$ oppure $f(x) = 0 \ 1 \ 0 \ 1 \ 0 \dots 0 \dots$

oppure rappresentarle con una sequenza ∞ di $f(x) = \begin{cases} 0 & x \text{ pari} \\ 1 & x \text{ dispari} \end{cases}$

Per comodità, F numerabile: numero ogni funzione (assegno numero progressivo ad ogni f) e costruiamo tabella ∞ di tutte le funzioni → costruiamo $g(j) = \begin{cases} 0 & f_j(x) = 1 \\ 1 & f_j(x) = 0 \end{cases}$ e F numerabile

g non corrisponde ad elenco $f_j \in F$, in particolare sulla diagonale principale. Per esempio, $\exists j / g(j) = f_j(x) = f_j(j)$ allora $g(j) = f_j(j)$, ma per definizione $g(j) = \begin{cases} 0 & \text{se } f_j(j) = 1 \\ 1 & \text{se } f_j(j) = 0 \end{cases}$, ovvero $g(j) \neq f_j(j)$ che è quindi una contraddizione

e quindi F non è numerabile → Basta costruire, \forall linea arbitraria, una funzione $f \in F$ che assume un valore ≠ su quella linea. Dato che F non è numerabile, a maggiore ragione non saranno numerabili gli insiemini $f: N \rightarrow N$, $f: N \rightarrow IR$, $f: IR \rightarrow IR$, $f: N^k \rightarrow N^3$ e quindi l'insieme dei problemi computazionali non è numerabile

↓ Quanti sono gli algoritmi?

da concentra si traccia attraverso sequenze finite di simboli di alfabeti finiti (compresi gli algoritmi) e dunque sono ∞ ma numerabili → ↵ dei problemi: i problemi non decidibili esempio Problema dell'corretto: decidere (sì/no), prendiamo una coppia algoritmo A e input B e dobbiamo decidere se A, ricevendo in input i dati B, termina o no in tempo finito → Vogliamo un algoritmo che indaga le proprietà di un altro algoritmo: Algoritmo perché usiamo lo stesso alfabeto (binario) per codificare dati ed algoritmi → A può operare nella rappresentazione di un altro algoritmo B: possiamo calcolare A(B) ma la non calcolare anche A(A)

1. ↓ Esempi

Primo(p)

```
fattore = 2;
while (p % fattore != 0)
    fattore++;
return (fattore == p);
```

2. Algoritmo di Goldbach

- Scendi in ordine crescente i numeri naturali pari maggiori di 2, fino a trovare un numero che NON sia esprimibile come la somma di due numeri primi
- Se e quando questo accade, stampa il numero e termina

XVIII secolo

"ogni numero intero pari $n \geq 4$ è la somma di due numeri primi"

Congettura falsa → Goldbach() si arresta

Congettura vera → Goldbach() NON si arresta

Turing: per alcuni problemi è impossibile stabilire la terminazione → il problema dell'arresto è INDECIDIBILE.

↓ Dimostrazione

Supponiamo per assurdo che esista un algoritmo Arresto che prenda A e D come input e determini in tempo finito le risposte ARRESTO(A,D) = 1 se A(D) termina ARRESTO(A,D) = 0 se A(D) non termina.

⚠ ATT: Arresto non può simulare A(D) → Se A non termina su D, Arresto non può rispondere Ø in tempo finito

Segliamo D = A, ovvero consideriamo la computazione A(A): ARRESTO(A,A) = 1 \iff A(A) termina. → Se esiste ARRESTO esisterebbe anche PARADOSSO(A)

```
while (ARRESTO(A,A)) {  
    ;  
}
```

Se l'↑ se 1 → se 0 termina ↓

Ma puradomo(A) termina \iff A(A) non termina: se a puradomo puradono puradono: PARADISO(PARADISO) termina \iff X = ARRESTO(PARADISO, PARADISO) = 0 \iff PARADISO(PARADISO) non termina ⚡ Assurdo: d'algoritmo ARRESTO non esiste → Non è possibile deciderlo per una coppia (A,D) arbitraria. Se potessi, potrei dimostrare la congettura di Goldbach. Inoltre, Turing ha dimostrato che non è possibile determinare estremamente proprietà di algoritmi senza simularli → sta decidibilità è una proprietà del problema e non dipende dal modello di calcolo (congettura) de voce nolo l'efficienza

• TEORIA della COMPLESSITÀ

Problemi intrattabili: risoluzione con tempo esponenziale → diventano quasi indecidibili

Problemi facili: algoritmi polinomiali → trattabili

Problemi pseudobilmente intrattabili: il limite superiore è esponenziale mentre quello inferiore non è noto ma si crede essere esponenziale

↓ Ma perché è importante il costo polinomiale?

Supponiamo di avere 2 calcolatori C₁, C₂ (n volte più veloce di C₁) e un tempo di calcolo t. Vogliamo sapere i dati trattabili in t: n₁ su C₁ e n₂ su C₂ → avere C₂ per t eguale ad avere C₁ per n₁t. Se l'algoritmo è polinomial e risolve il problema in C_n^s secondi, C₁: C_n^s = t e quindi n₁ = (t/c)^{1/s} C₂ = C_n^s = kt e quindi n₂ = (kt/c)^{1/s} = K^{1/s} (t/c)^{1/s} $\boxed{? n_2 = K^{1/s}}$ migliora fattore sull'ipotesi

Se l'algoritmo è esponenziale, C₁: C₂^M = t \rightarrow 2^M = t/c C₂: C₂^{N₂} = n₁t \rightarrow 2^{N₂} = kt/c = K₂^M } n₂ = n₁ + log₂K → fattore additivo algoritmo più per calcolo

Problema Π: I = istanze input

↓ S = soluzioni

Decisionali: risposta binaria (indagano sue proprietà) → istanze positive/accettabili: risposta = 1 → istanze negative: risposta = 0

Ricerca: vogliamo cercare una soluzione (distanza tra vertici di grafo)

Ottimizzazione: vogliamo trovare la soluzione migliore

→ Utilizzati da teoria della complessità: il tempo di calcolo è utilizzato solo per il calcolo della soluzione in quanto l'output è O/1 ed inoltre la difficoltà del problema è comunque contenuta nella sua versione decisionale → Possiamo trasformare un problema in decisionale chiedendo l'esistenza di una soluzione che soddisfi una determinata proprietà: Non è più difficile di trovare la soluzione del problema di ottimizzazione e quello di ottimizzazione non è più semplice ma può essere più difficile

↓ classi di complessità

Busto Π ed un algoritmo A, A risolve Π se, data un'istanza di input x,

$A(x) = 1 \iff \pi(x) = 1$. A rende π in tempo $t(n)$ e spazio $s(n)$ se il tempo di esecuzione e l'occupazione di memoria a cui A sono rispettivamente $t(n)$ e $s(n)$

Time ($f(n)$): insieme di problemi decinondi che possono essere risolti in tempo $O(f(n))$

Space ($f(n)$): insieme di problemi decinondi che possono essere risolti in spazio $O(f(n))$

Algoritmo polinomiale: $\exists C, n_0 > 0$ / il numero di passi elementari è al più n^C

\forall input di dimensione n e $\forall n > n_0$

Classe P: classe dei problemi risolvibili in tempo polinomiale

Classe PSPACE: celle di memoria che crescono in modo polinomiale rispetto a x

Classe EXPTIME: tempo cresce esponenzialmente con x

$P \subseteq PSPACE$ (congettura): Costo in tempo limita numero celle e $PSPACE \subseteq EXPTIME$. Non è noto se le inclusioni siano proprie o no

↓ Esempi

CLIQUE: Consideriamo sottoinsiemi di vertici in ordine di cardinalità decrescente e verifichiamo se formano clique di dimensione k . Se n è il numero di vertici, nel caso peggiore si andrà su 2^n insiemi, quindi CLIQUE è EXPTIME

CAMMINO HAMILTONIANO: Verifichiamo se, camminando sugli archi, è possibile percorrere da tutti i nodi visitando gli archi una ed una sola volta $\rightarrow n!$ permutazioni e quindi è EXPTIME

SAT: Esempio $\rightarrow V = \{x, y, z, w\}$, FNC: $(x \vee \bar{y} \vee z) \wedge (\bar{x} \vee w) \wedge y$

Forma normale congiuntiva

clausola

↓

Vogliamo sapere se \exists un assegnamento delle variabili V che rende la formula vera $\rightarrow 2^n$ assegnamenti, quindi è EXPTIME

Non sappiamo se è possibile

Classe NP: Certificati \rightarrow Non sappiamo rendere questi problemi in tempo polinomiale, ma se qualcuno ci fornisce una soluzione possiamo verificarla velocemente. È possibile verificare l'esistenza della proprietà per le istanze positive \rightarrow Dando delle informazioni aggiuntive si risponde molto più velocemente: il certificato è una descrizione breve del problema con tali proprietà \rightarrow Certificato per CLIQUE

sottoinsieme di k vertici, che forma la clique

Certificato per Cammino Hamiltoniano
permutazione degli n vertici che definisce un cammino semplice

Certificato per SAT

Un'assegnazione di verità alle variabili che renda vera l'espressione

Un problema si verifica in tempo polinomiale se ogni istanza accettabile x di π di lunghezza n ammette un certificato y di lunghezza polinomiale in n ed esiste un algoritmo di verifica polinomiale in n e applicabile a ogni coppia $\langle x, y \rangle$, che permette di ottenere che x è accettabile \rightarrow Classe NP: verificabili in tempo polinomiale

Dal teoria della verifica ci serve solo per capire la difficoltà dei problemi in quanto il certificato va comunque calcolato in tempo esponenziale

↓ Rapporto tra P ed NP

$P \subseteq NP$: Ogni problema P avrà certificato verificabile in tempo polinomiale in quanto basta eseguire l'algoritmo che risolve il problema per costruire il certificato.

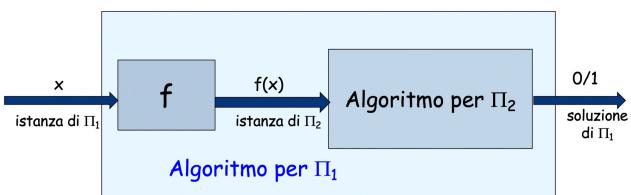
Non è invece valido il contrario. Non sappiamo se $P = NP$ o $\neq NP$.

Poniamo però come congettura che $P \neq NP$. La ricerca si è concentrata ad individuare i problemi più difficili in NP → Riduzione polinomiale
 cui dice che, se esistesse un algoritmo polinomiale che risolve problemi NP-completi, allora tutti i problemi NP potrebbero essere risolti in tempo polinomiale, e dunque $P = NP$. Quindi, o tutti i problemi NP-completi sono risolvibili in tempo polinomiale oppure nessuno lo è.

Π_1, Π_2 problemi decisiando com'è istante di input I_1 e I_2 . Π_1 si riduce in tempo polinomiale a Π_2 ($\Pi_1 \leq_p \Pi_2$) se $\exists f: I_1 \rightarrow I_2$ calcolabile in tempo polinomiale / \forall istanza x di Π_1 , x è un'istanza accettabile di Π_1 se e solo se $f(x)$ è un'istanza accettabile di Π_2 .

↓ importanza

$\Pi_1 \leq_p \Pi_2$ e $\Pi_2 \in P \rightarrow \Pi_1 \in P$ in quanto trovare f richiede tempo polinomiale



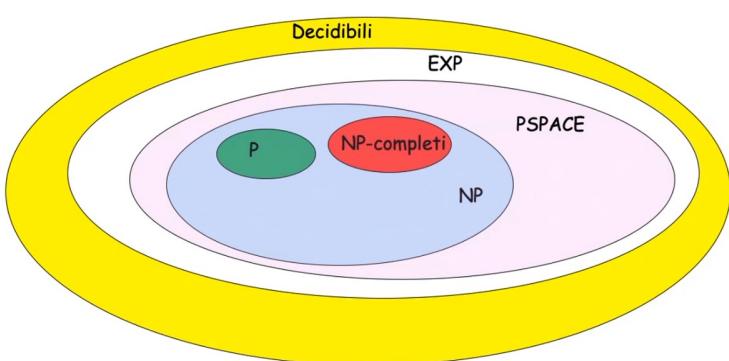
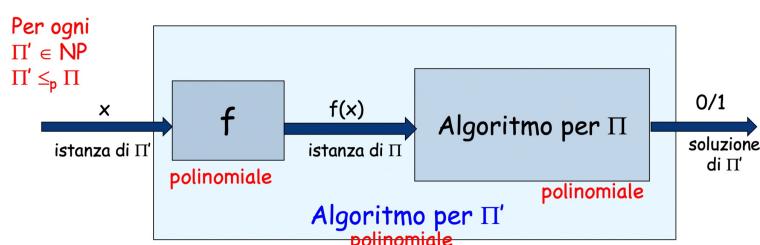
In problema Π si dice NP-hard se $\forall \Pi' \in NP, \Pi' \leq_p \Pi$. È NP-completo se anche $\Pi \in NP$. → La verifica non è facile ma la dimostrazione fatta ci semplifica il lavoro → TEOREMA: SAT è NP-completo

Per vedere se il suo problema è NP-completo verifichi se SAT

Si può ridursi al suo problema

- $SAT \leq_p CLIQUE \Rightarrow CLIQUE$ è NP completo
- SAT è NP completo $\Rightarrow CLIQUE \leq_p SAT$
- SAT e CLIQUE sono NP equivalenti.
- Tutti i problemi NP completi sono tra loro NP equivalenti.
- Sono tutti facili, o tutti difficili

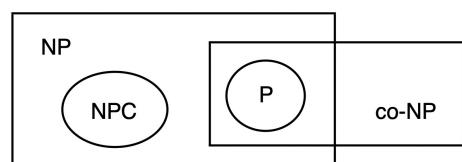
↓ concludendo



↓ Altre classi

co-P e co-NP. In generale, un problema complementare accetta la stessa riduzione del problema iniziale. Se il problema è polinomiale, anche il complementare lo è (se Π rientra in 1, co- Π rientra in 0), quindi $P = co-P$. Non succede invece cosa accade per problemi NP. Si congettura che $NP \neq co-NP$.

→ Se un problema è NP-hard e dobbiamo fare ottimizzazione ci accontentiamo di avvicinarci all'ottimo in tempo polinomiale



da casualità

Permettono di utilizzare algoritmi randomizzati se prob. sovra è piccola e inaccettabile eufemistica e di generare le chiavi segrete dei cifrari in modo da renderle indecifrabili

↓

$$\begin{array}{ll} h_1 = 111 \dots 1 \dots 1 & |h_1| = 20 \longrightarrow \text{Probabilità ottenuta } \left(\frac{1}{2}\right)^{20} \\ h_2 = 011 \dots 0 \dots 1 & |h_2| = 20 \longrightarrow \text{Probabilità ottenuta } \left(\frac{1}{2}\right)^{20} \end{array}$$

↪ Non posso derivere facilmente h_2

↓ Significato algorithmico della casualità

Teoria algorithmica dell'informazione: Kolmogorov 1960 → Idea: una sequenza binaria h è casuale se non ammette un algoritmo di generazione A la cui rappresentazione binaria sia più corta di h .

↓

Supponiamo h = sequenza di n 1 e quindi $A_h = \langle$ genera n 1 \rangle

↓ In C++
for (int i=0; i<n; i++)
print 1;

da cui sappiamo $|h|=n$ e $|A_h| = \text{cost} + \Theta(\log n)$ → NON CASUALE rappresentazione posizionale numeri

Se però h non presenta evidenti regolarità o non ci appare casuale, l'algoritmo di generazione la contiene al suo interno, e la contiene in output → $h_2 = 1001110001\dots01$

↓ P2

↓ print '1001110001\dots01' e quindi $|P_2| = \text{cost} + \Theta(n)$

Averendo algoritmi, dobbiamo avere sistemi di calcolo: devono essere derivabili e quindi numerabili

↓ FORMALIZZAZIONE MATEMATICA

Dobbiamo rendere la definizione di casualità indipendente dal sistema di calcolo adottato → sono numerabili (dobbiamo derivarli): $S_1, S_2, \dots, S_i, \dots$

↓

DEFINIZIONE: La complessità di Kolmogorov di h in S_i , indicata con $K_{S_i}(h)$ è definita come $K_{S_i}(h) = \min \{ |P| \mid S_i(P) = h \}$ ove P è un programma per S_i , che genera h , $S_i(P) = h$

↓

Tra i vari sistemi di calcolo ∃ almeno un sistema di calcolo in grado di simulare gli altri sistemi di calcolo → sistema di calcolo universale

↓

P programma che genera h in S_i . $S_i(P) = h$. Per contenere un programma che genera h in S_i prendo $q = \langle i, P \rangle$ → genera h in S_u :

$S_u(q) = S_u(\langle i, P \rangle) = S_u(i, P) = h$: produce lo stesso output

↓

Sufficiente aggiungere indice i

$q = \langle i, P \rangle$ genera h in S_u e $|q| = |i| + |P| = \Theta(\log i) + |P|$
hit Non dipende da h

Averemo quindi $K_{S_u}(h) \leq K_{S_i}(h) + C_i$ $h_i, \forall h$

non dipende da h

Basta ottenere un programma migliore in S_u simulando un sistema di calcolo diverso: limite inferiore preo come riferimento

DEFINIZIONE: La completa secondo Kolmogorov di una sequenza h è data da $K(h) = K_{\text{SU}}(h)$ [vedi la relazione precedente]

DEFINIZIONE CASUALITÀ: Una sequenza h è casuale se $K(h) \geq |h| - \lceil \log_2 |h| \rceil$

permette di includere più sequenze e rendere applicabile definizione

⚠ Proprietà della sequenza che non dipende dal sistema di calcolo

↓ Dimostrazione: esistono sequenze casuali

Consideriamo sequenze binarie di lunghezza $n \rightarrow S = \# \text{ seq. binarie lunghe } n = 2^n$
 $T = \# \text{ seq. binarie di lunghezza } n \text{ non casuali}$

Per dimostrare \exists , dobbiamo mostrare che $T < S$. Le T seq. non casuali sono generate da programmi di lunghezza $< n - \lceil \log_2 n \rceil$. Indichiamo con N il numero di sequenze binarie di lunghezza $< n - \lceil \log_2 n \rceil$. Tra queste ci saranno anche la sequenza che codificano i programmi che generano le T sequenze non casuali. Avendo $N = \sum_{i=0}^{n-\lceil \log_2 n - 1} 2^i = \frac{2^{n-\lceil \log_2 n \rceil} - 1}{2 - 1} = 2^{n-\lceil \log_2 n \rceil} - 1$

da cui ottieniamo $T \leq N \leq S$ da cui $T < S$ ■

$$\lim_{n \rightarrow \infty} \frac{T}{S} = \lim_{n \rightarrow \infty} \frac{2^{n-\lceil \log_2 n \rceil} - 1}{2^n} = \lim_{n \rightarrow \infty} \left(\frac{1}{2^{\lceil \log_2 n \rceil}} - \frac{1}{2^n} \right) = 0 : \text{La maggioranza sono casuali ed essere della lunghezza}$$

⚠ Non è possibile trovare un algoritmo che verifichi la casualità secondo Kolmogorov → Indice di hit

↓ Dimostrazione

Per avviare, supponiamo che \exists un algoritmo $RANDOM(h) = \begin{cases} 1 & h \text{ casuale} \\ 0 & h \text{ non casuale} \end{cases}$

Consideriamo PARADISO che enumera tutte le sequenze di lunghezza h e per chiama $RANDOM$ — for (binary $h < 1$ to ∞) — → di genere di lunghezza h .

if ($|h| - \lceil \log_2 |h| \rceil > P$) & $RANDOM(h) = 1$ casuale
 return h ;

2 → restituirne la seq. casuale di lunghezza h .

1. Condizione di lunghezza: $|P| = |\text{Paradiso}| + |\text{Random}| \rightarrow$ costante indipendente da h in quanto h compare in paradiso solo come nome di variabile (sequenze non si memorizzano)
2. Casualità sequenza

↓

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

programma che genera h , breve $\iff h$ non casuale

2. h casuale

↓

⚠ h casuale e non casuale e quindi siamo all'ASSURDO: \exists l'algoritmo Random

COME OTTENIRE LE SEQUENZE

Sorgente casuale binaria: 0 e 1 sono equiprobabili ($P(0) = P(1) = 1/2$) → la generazione di un hit è indipendente da quella degli altri. La prima può avere "dallegerita" dicendo che le probabilità di ottenere 0/1 siano immutabili nel processo di generazione: $P(0) > 0, P(1) > 0$ immutabili

↓ Perché?

$P(0) > P(1)$: 0110001001011000001001

01 → 0 10 → 1: equiprobabili. 0110010 (nuova scrittura)

Per quanto riguarda la seconda ipotesi, è più difficile da applicare.

↓ Come si mano le varigenti?

Campioniamo il rumore di fondo di un microfono, casualità im processi software a fatto che i dispositivi di campionamento siano protetti

↓ Nella pratica

Generatori di numeri pseudo-casuali → casualità generata tramite algoritmo obiettata all'interno di processi matematici. Le possibili sequenze generate sono funzione dell'input (seme)

↓

INPUT: sequenza breve (seme) → CASUALE

OUTPUT: flusso di hit obbligatoriamente lungo → Si ripete quando viene riconosciuto il seme (periodo): migliore tanto è più lungo il periodo

S = seme composto da n hit. Se diamo lo stesso seme il generatore dà la stessa sequenza in output. → Al massimo sono 2^n sequenze diverse, che sono molto minori di tutte le sequenze possibili

↓

Generatore lineare: $x_i = (a \cdot x_{i-1} + b) \bmod m$ con $a, b, m \in \mathbb{N}$ parametri generatore

↓

$x_0, x_1 = (ax_0 + b) \bmod m, x_2 = (ax_1 + b) \bmod m, \dots$

Periodo massimo m (caso m). Se i parametri sono scelti bene il generatore produce una permutazione degli interi $[0, m-1]$ diversi, $\gcd(b, m) = 1$, $(a-1)$ divisibile + fattore primo di m

Per sequenze binarie si fa $\frac{x_i}{m}$ e si prende la parte della prima cifra decimale

Generatore polinomiale: $x_i = (a_1 \cdot x_{i-1}^t + a_2 \cdot x_{i-1}^{t-1} + \dots + a_t \cdot x_{i-t+1}) \bmod m$

↓

TEST STATISTICI: 1. **Test di frequenza:** I simboli della sequenza devono apparire con pari frequenza

2. **Poker test:** le otto sequenze con la stessa lunghezza devono avere equiprobabilità

3. **Test di autocorrelazione:** Verifica che ad intervalli finiti non sia presente lo stesso numero

4. **Run test:** Verifica la frequenza di rotture consecutive identiche con lo stesso hit che si ripete e vogliamo che diminuisca esponenzialmente con la lunghezza

Vogliono per entrambi i generatori che però non possono avere utilizzati per generare le chiavi in quanto i parametri (e di conseguenza i hit prodotti) possono essere stimati con facilità

↓ Nuovo test

Test di prossimo hit: Un generatore binario supera questo test se \exists un algoritmo polinomiale in grado di prevedere l' $(i+1)$ -esimo hit generato a partire dalla conoscenza degli i -hit precedentemente generati con probabilità $> 1/2$. Un generatore è critograficamente sicuro se supera il test del prossimo hit

↓ Tipicamente erano

Fermioni one-way: è computazionalmente facile calcolare $y = f(x)$ [$x \rightarrow f(x)$ tempo polinomiale] mentre è difficile calcolare $x = f^{-1}(y)$ [algoritmi di costo esponenziali]

↓ Idea

Se ne x_0 è generato $x_1 = f(x_0)$, $x_2 = f(x_1) = f(f(x_0)) = f^{(2)}(x_0), \dots, x_i = f^{(i)}(x_0)$. Non posso comunque in ordine di generazione era invece che comunque al

comteario perché x_{i-1} può avere sì ottenuto da x_i ma in tempo esponenziale: $x_i = f^{-1}(x_{i-1}) \rightarrow$ sequenza difficile da prevedere

↓ Per ottenere sequenze binarie

Per di soli hard-core funzioni one-way: concentrano la difficoltà computazionale in un hit → $b(x)$ è un predicato hard-core di una funzione one-way $f(x)$ se è facile da calcolare conoscendo x ed è difficile da prevedere con probabilità $> 1/2$ se si conosce $f(x)$ esempio: $f: (\text{elemento})^2 \rightarrow \text{modulo num. composto}$

↓ b: parità

Generatore BBS (1986) → ortograficamente sicuro

↓
 $n = p \cdot q$ dove p, q primi molto grandi tali che
 $p \bmod 4 = 3, q \bmod 4 = 3$ e $2^{p/q} \bmod n$ è $2^{q/p} \bmod n$
primi tra loro. Sceglio q coprimo con n e
calcoliamo $x_0 = q^2 \bmod n$ (true). Generiamo una successione
di $m \leq n$ interi $x_i = (x_{i-1})^2 \bmod n$, $i > 1$. Sceglio
come predicato la parità: $b_i = 1 \iff x_{m-i} \in \text{dispari}$
(leggo grandi la sequenza al contrario)

Droppo lento per sequenze lunghe

↓ Altra tecnica: uso cifrari simmetrici

Generatori di numeri pseudo casuali basati su cifrari simmetrici: blochi di 128/256 bit che
cifrano uno per volta → cifrario simmetrico (AES, 3DES, DES) che produce $\pi = \# \text{ hit}$
parole predette (blochi) [$\pi = 64 \text{ DES}, 128, 256 \text{ AES}$] S è il rene casuale di π hit.
 $m = \# \text{ seguenti di } \pi \text{ hit predette}$. $k = \text{chiave segreta del cifrario}$

Generatore (s, m) , $d = \text{rappresentazione in } \pi$ hit di data e ora

$y = C(d, k)$ [funzione cifratura]

$\tilde{x} = S$

for ($i=1; i \leq m; i++$) {
 $x_i = C(y \oplus \tilde{x}, k);$
 $\tilde{x} = C(y \oplus x_i, k);$
commico x_i all'esterno;
 y }

ove $\oplus = \text{XOR hit a hit}$

↓
Otengo m sequenze lunghe π per ottenere sequenza casuale

• TEST di PRIMALITÀ

```
for (i=2; i <= sqrt(N); i++) {  

    if (N % i == 0) return false; // divisore < della radice  

}
return true;
```

→ Se un numero è composto contiene almeno un
divisore $<$ della radice

↑ più \sqrt{N} operazioni acunno di costo di $O(\log^2 N)$ per quanto riguarda il costo
e quindi $T(I) = O(\sqrt{N} \cdot \log^2 N)$ → $I = O(\log n)$, $N = O(2^{\lfloor I \rfloor})$ ma $\sqrt{N} = N^{1/2} = O(2^{\frac{\lfloor I \rfloor}{2}})$

↓ ottieniamo $O(|I|^2 \cdot 2^{\frac{|I|}{2}})$: costo esponenziale per il numero di cifre di N (I),
quindi tropo complicato

utilizziamo un algoritmo randomizzato

→ abbiamo 2 tipi

quicksort

primarietà

das Vegas: risultato sicuramente corretto in tempo probabilmente breve

MonteCarlo: risultato probabilmente corretto in tempo sicuramente breve → ovviamente vogliamo che la probabilità di avere una piccola A piacere e sostanzialmente trascurabile

TEST di PRIMALITÀ di MILLER-RABIN

N , intero di N cifre dunque assomma

$$N-1 = 2^w \cdot z$$

massimo esponente divisione per 2

Costo: $\log_2 n$ (determinazione w e z)

Supponiamo di negligenza y intero arbitrario:
allora: P1: $\text{GCD}(N, y) = 1$ (def. primalità)
P2: $y^z \mod N$ congruo a 1 OR $\exists i, 0 \leq i \leq w-1 / y^{2^i} \mod N = 1$

Se sono veri il numero è primo e sicuramente è composto: la probabilità che sia composto diminuisce provando tanti y diversi

↓ decisiva di Miller-Rabin

Se N è un numero composto, il numero di interi (y) compresi tra 2 e $N-1$ che soddisfano i prediciati P1 e P2 è minore di $N/4$ → $\# \{2 \leq y \leq N-1 / P_1(y) = \text{TRUE}$ AND $P_2(y) = \text{TRUE}\} < N/4$ e quindi la probabilità di negligenza un testimone y che deve rendere vero P1 e P2 è $< \frac{N/4}{N-2} \approx \frac{1}{4}$

⚠ ATT: Se divenne un predi cato è falso il numero è sicuramente composto
↓ Idee

Sceglio a caso $y \in \{2, N-1\}$

Se uno dei due prediciati è falso allora N è sicuramente composto
Se i prediciati sono entrambi veri, N è composto con probabilità $< 1/4$ e dunque è primo con probabilità $> 1 - 1/4 = \frac{3}{4}$

Hanno la volte, con la volte comuni e indipendenti del testimone y la probabilità di essere dunque $(\frac{1}{4})^k$: si può fare in quanto la verifica è polinomiale

VERIFICA (x, y) : controlla la validità del certificato y ove y è certificato che N sia composto
if $(P_1 = \text{falso} \text{ OR } P_2 = \text{falso})$ return 1; (composto)
else return 0;

↳ P1: cubico in funz. del numero di cifre
P2: $N-1 = 2^w \cdot z$. Se $w=1 \rightarrow z = \frac{N-1}{2}$ e voglio calcolare $y^z \mod N$

Algoritmo di ESPOENZIAZIONE VELOCE (quadrature successive)
obiettivo → # operazioni $O(\log z)$

↓ esempio

$$x = 9^{45} \mod 11 = 9^{\frac{32+2+4+1}{2}} \mod 11$$

Calcolano le potenze 9^{2^i} fino ad arrivare a 9^{32}
ciascuna come quadrato della precedente

$$\begin{aligned}
 q^2 \bmod M &= 4 \\
 q^4 \bmod M &= (q^2)^2 \bmod M = 5 \\
 q^8 \bmod M &= (q^4)^2 \bmod M = 3 \\
 q^{16} \bmod M &= (q^8)^2 \bmod M = 9 \\
 q^{32} \bmod M &= (q^{16})^2 \bmod M = 4
 \end{aligned}
 \left. \begin{array}{l} 5 \text{ moltiplicazioni} \\ \downarrow \text{In generale} \\ t \text{ moltiplicazioni} \end{array} \right\} \text{ove } t = \lfloor \log_2 k \rfloor \text{ (massima pot. 2 composta in } k \text{)}$$

$$\begin{aligned}
 q^{64} &= q^{32} \bmod M \cdot q^8 \bmod M \cdot q^4 \bmod M \cdot q^2 \bmod M = \\
 &= (4 \cdot 3 \cdot 5 \cdot 9) \bmod M = 1 \rightarrow O(t) \text{ moltiplicazioni}
 \end{aligned}$$

Formalizzando: $x = q^k \bmod M$ ove y, z, s dello stesso odg (al max $O(n)$).

1. Si ricompono z in somma di potenze di 2: $z = \sum_{i=0}^t k_i \cdot 2^i$ ove $k_i \in \{0, 1\}$

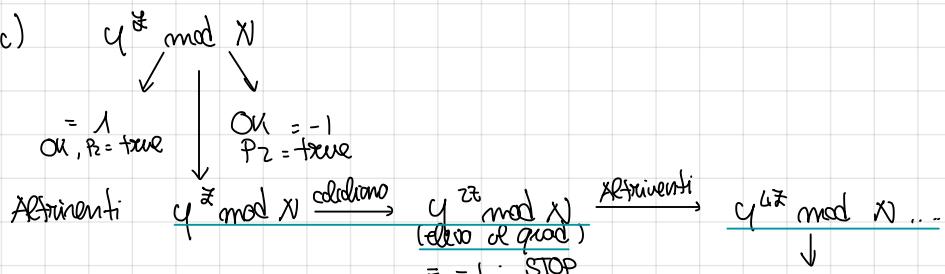
2. Si calcolano tutte le potenze $y^{2^i} \bmod M = (y^{2^{i-1}})^2 \bmod M$

3. Si calcola $x = y^z \bmod M$ come $x = \prod_{i: k_i \neq 0} y^{2^i} \bmod M$

↓ $O(\log z)$ moltiplicazioni

Vediamo adesso la seconda condizione (in caso la prima sia falsa)

Si calcola (Quad. succ)



Al massimo w volte ($O(\log n)$)
Ove $w = O(\log n)$

Test MR (N, k)

for $i = 1, i \leq k, i++$

- scegli a cono $a \in [2, N-1]$;
- if (verifica(N, a) = -1) return 0; $\rightarrow N$ certamente composto

return 1; $\rightarrow N$ è primo con prob. eraria $< (\frac{1}{4})^k$

Generazione di numeri primi

Si genera numero casuale disponibili seguito da test di primalità \rightarrow si ripete fino a quando

si trova un numero dichiarato primo (predicati veri)

↓ TEOREMA

Il numero dei numeri intorni primi e minori di N tende a $\frac{N}{\ln N}$ per $N \rightarrow \infty$

e quindi per N sufficientemente grande, in un suo intorno di ampiezza $\ln N$ cade mediamente un numero primo $\rightarrow \ln N$ è proporzionale alla dimensione dell'input ovvero al # di cifre \rightarrow # tentativi è polinomiale in $\log N$

↓ Algoritmo

Primo (n) // n : # hit significativi num. primo da generare

// genera numero primo di almeno n hit (prob. eraria $< \frac{1}{4^n}$)

$S =$ sequenza cumuli $n-1$ hit (gen. pseudo-cumuli)

$N = 1\$1$ (aggiungo 1 a inizio e fine per ottenere nr. dispon)

while (TestMR(N, n) = 0) $N \cdot N+2$; (puoi dare disponibile a dispon)

return N ; // Al massimo dei $n+1$ hit

↓

Costo = $O(n^4)$ ove $n = \text{nr hit}$ $N \rightarrow O(n)$ volte test MR che è $O(\ln^3)$

Classe RP (Random Polynomial)

Classe dei problemi decisionali verificabili in tempo polinomiale randomizzato



Π: problema decisionale

X: insieme di input di Π

Y è un certificato probabilistico di X se: 1. è di lunghezza polinomiale nella dim. di X

2. è estratto perfettamente a caso da un insieme accettabile a X

A: Algoritmo di verifica polinomiale

X non possiede la proprietà ($\Pi(x) = 0$)

possiede la proprietà ($\Pi(x) = 1$) con probabilità

di primalità

↓
congettura

P ⊈ RP ⊈ NP

$A(x, y) \rightarrow$ Attesta in tempo polinomiale che

con CERTA

oppure attesta che X

probabilità $> 1/2$

($^{3/4}$) nel caso del test

Cifrari storici

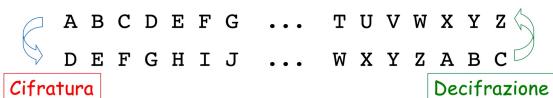
Considerati storici fino alla macchina Enigma. Sono stati tutti forzati e riguardano la crittografia manuale che cifra messaggi in linguaggio naturale che prendono come riferimento l'alfabeto inglese.

↓ principi:

1. Funzioni C e D facili da calcolare
2. impossibile ricavare D se C non è nota
3. C = C(m) deve apparire "innocente" (devono sembrare testi in chiaro)

o Cifrario di CESARE

Il crittogramma si ottiene traslando le lettere dell'alfabeto di 3 posizioni verso destra (circularmente) →



Non c'è una chiave e quindi il metodo deve rimanere segreto. Funzionava perché solo pochi sapevano leggere e scrivere

> Sicurezza: Rotazione di K posizioni con K nello tra chi comincia ($1 \leq k \leq 25$). Si introduce quindi una chiave

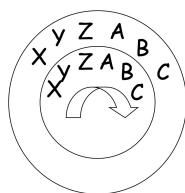
↓ Formulazione matematica

$\text{pos}(x)$: posizione nell'alfabeto della lettera x, chiave $K: 1 \leq k \leq 25$.

Cifratura di x: lettera y / $\text{pos}(y) = (\text{pos}(x) + k) \bmod 26$

Decifrazione di y: lettera x / $\text{pos}(x) = (\text{pos}(y) - k) \bmod 26$

realizzazione →



disco interno: lettere alfabeto in chiaro

disco esterno: lettere cifrate

attaccando: Provo tutte le chiavi o uso statistica: lettera cifrata che appare più spesso = lettera più frequente in linguaggio naturale. È sufficiente trovare una corrispondenza per trovare tutte

proprietà: commutativa: l'ordine delle operazioni può essere cambiato senza alterare il crittogramma finale

date due chiavi k_1 e k_2 è una regola s: $C(C(s, k_2), k_1) = C(s, k_1 + k_2)$: equivalente ad applicarle una sola volta nonando le chiavi.

o CATEGORIE

cifrari a sostituzione: sostituiscono ogni lettera del messaggio in chiaro con una o più lettere dell'alfabeto con una regola prefissa

cifrari a transposizione: permettono le lettere del messaggio in chiaro con una regola prefissa

→ sostituzione monofabeticca: ad una lettera del messaggio corrisponde sempre una stessa lettera nel crittogramma

sostituzione polialfabeticca: alla stessa lettera del messaggio corrisponde una lettera scelta in un insieme di lettere possibili

→ Complicidiamo il cifrario: $\text{pos}(y) = (ax + \text{pos}(x) + b) \bmod 26$, $a = \{a, b\}$.



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

\downarrow inverso di a modulo 26 ($a^{-1} \cdot a \equiv 1 \pmod{26}$)

Condizione: $\text{mcd}(a, 26) = 1$ altrimenti la funzione di cifratura non è inieettiva e quindi non posso decifrare

↓ quale chiave?

a e 26 devono essere coprimenti. I fattori primi di 26 sono 2 e 13.

a può quindi avere qualunque di 13

$$\# a = \phi(26)$$

↓

$$\phi(26) = (13-1)(2-1) = 12$$

↓

$$\text{Chiavi possibili} = 12 \cdot 26 = 312 - 1 = 311$$

b è vero

(0,0)

Avremo chiavi:

I fattori primi di 26 sono 2 e 13, ad eccezione del 1.

$\phi(n) = \# \text{ interi } < n \text{ coprimi con } n$ (funz. di Euler)

$n = p \cdot q$ dove p, q primi, allora $\phi(n) = (p-1)(q-1)$

Prendiamo una permutazione arbitraria dell'alfabeto come chiave: la lettera che sta in chiave nella posizione i viene cifrata nella posizione i della permutazione → le chiavi sono $26! - 1 \sim 6 \cdot 10^{26}$ chiavi.

identica

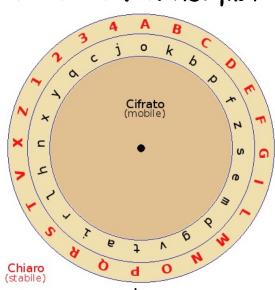
Non è sicuro perché si può forzare sfruttando la struttura logica dei messaggi in chiave e l'occorrenza statistica delle lettere.

→ Esempio più antico: cifrario di Augusto. Augusto riceveva i documenti in greco e la sequenza di lettere del documento con la sequenza di lettere del primo libro dell'Iliade. Sostituiva poi ogni lettera con il numero che indica la distanza di tale lettera con pari posizione nell'Iliade

→ critogrammi erano scritti in numeri e poi metteva in corrispondenza la sequenza di lettere del primo

Difficile da forzare se la chiave è lunga ma sbaglioso perché la chiave deve essere registrata in forma scritta

↓ Esempio: cifrario di Alcibiade



Alfabeto esterno

lettere (alcune) e numeri, per formulare il messaggio

Alfabeto interno

più ricco, disposto in modo arbitrario (e diverso per ogni coppia di utenti), per costruire il critogramma

Molto sicuro se ci sono molti cambi di chiave inseriti in modo irregolare

→ i cambi di chiave rendono invincibili gli attacchi basati sulla frequenza

→ Inizio: gli strumenti vanno allineati allo stesso modo e si inserisce davanti un numero. Quando ci cambia cambio chiave

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 G Q I L M	

Chiave: A-S

Messaggio: NON FIDARTI DI EVE

m = NON FIDA 2 RT ID IE VE

c = UNU JR KS Q

qui la chiave diventa A-Q

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 F W A G	

Chiave: A-Q

Messaggio: NON FIDARTI DI EVE

m = NON FIDA 2 RT ID IE VE

c = UNU JR KS Q U Y B M B S P S

qui la chiave diventa A-Q

Metodo 2: indice modulare

⚠ Si parte sempre con lo stesso allineamento

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 F

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 V

Semplificato da de Vigenère: composizione di cifrari di Cesar con shift tutti diversi. La chiave è corta e viene ripetuta ciclicamente. Ogni lettera della chiave indica una traslazione della corrispondente lettera del testo

chiave: C H I A V E

traslazione: 2 7 8 0 24 4

N	O	N	F	I	D	A	R	T	I	D	I	E	V	E
2	7	8	0	24	4	2	7	8	0	24	4	2	7	8
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
P	V	V	F	G	H	C	Y	B	I	B	M	G	C	M

↓ metodo alternativo

Tabella T, 26 × 26, nota a tutti

Riga i-esima

alfabeto di 26 lettere rotato verso sinistra di i-1 posizioni

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

La ricetta è influenzata dalla lunghezza della chiave, le appaltazioni dello stesso lettera distanti un multiplo di la lunghezza si sovrappongono alla stessa lettera della chiave e quindi non trasformate nella stessa lettera cifrata → Ripetizione di cifrature nuove alfabetiche

Se u una una chiave lunga come il testo, ovule e non riutilizzabile il

→ Per cifrare: si procede carattere per carattere, si dispongono in ciascuna riga due righe adiacenti, allineando le lettere in verticale. Ogni lettera del messaggio in chiave risulta allineata ad una lettera Y della chiave. La X viene sostituita nel ciphertext con la lettera che si trova nella cella di T all'incirca tra la riga che inizia con X e la colonna che inizia con Y.

↓ decifrare

Si cerca nella riga che corrisponde al carattere della chiave, il corrispondente carattere del ciphertext. Il primo carattere della colonna corrisponde al carattere in chiave: se la chiave contiene la lettera che corrisponde alla stessa lettera distanti un multiplo di la lunghezza della chiave, il primo carattere della colonna corrisponde al carattere in chiave.

cifrario diventa inaffacciabile → non può essere decifrato se non si conosce la chiave → problema: scambio della chiave

↓

One-Time Pad: applicato a binario

→ Si fanno permutazioni aggiungendo ad esempio delle lettere per confondere il messaggio

↓

Chiave: intero h ; permutazione π interi $\in \{1, 2, \dots, h\}$. Per cifrare si suddividono il messaggio m in blocchi di h lettere e si permutano le lettere di ciascun blocco secondo π . Se la lunghezza di m non è divisibile per h si aggiungono alla fine del messaggio lettere qualsiasi. Queste partecipano alla transposizione ma sono ignote del destinatario perché poste alla fine del messaggio

↓

$$h = 9$$

$$\pi = \{1 \ 2 \ 5 \ 3 \ 7 \ 6 \ 4 \ 9 \ 8\}$$

1	2	3	4	5	6	7	8	9
C	I	V	E	D	I	A	M	O
↓	↓	↓	↓	↓	↓	↓	↓	↓
C	I	D	V	A	I	E	O	M
D	O	N	M	A	I	A	C	B

↳ Annuo $h! - 1$ chiavi e la foratura diventa più difficile cercando la.

Altro esempio: cifrario a permutazione di colonne

↓

$K = \langle C, r, \pi \rangle$ ove C ed r denotano il numero di colonne e righe di una tabella di lavoro T e π è una permutazione degli interi $\in \{1, 2, \dots, C\}$. m viene decomposto in blocchi di $C \times r$ caratteri che vengono di volta in volta inseriti nella tabella ↓ Numero di chiavi esponenti di

$$c = 6, r = 3, \pi = \{2, 1, 5, 3, 4, 6\}$$

$m = \text{NON SONO IL COLPEVOLE}$

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



1	2	3	4	5	6
N	O	N	S	O	N
O	I	L	C	O	L
P	E	V	O	L	E

2	1	5	3	4	6
O	N	O	S	N	
I	O	O	L	C	L
E	P	L	V	O	E

$$C = \boxed{\begin{array}{cccccc} O & I & E & N & O & P \\ I & O & E & N & P & O \\ E & P & L & V & O & N \end{array}}$$

T ottenuto leggendo per colonna
T permutata

Altro esempio: cifrario a griglia (Richelieu) → Non danno testo in chiaro in un libro. La chiave è data da una scheda perforata e dalla pagina del libro

Variante: costruiamo griglia quadrata $q \times q$ con q fori. $S = q^2/4$ celle della griglia trasparenti e le altre occultate. Sulla tabella vista sopra mettiamo la mia griglia che ha 1/4 di fori e nello in ordine il menaggio. Una volta finiti i fori ruota la griglia di 90° e vedi avanti ruotando 3 volte.

Esempio: $q = 6, S = 9$,

$m = \text{L'ASSASSINO È ARCHIMEDES TARRINGTON}$

→ Decifrazione: degrado
la griglia e poi
ruota

Rot. 1		Rot. 2		*: caratteri a caso
L	A	O	E	
S	S	R	A	
		C	H	
S	I			
	N			
D		G	T	O
E		N	*	S
	T	*	*	S
A		*		S
	R	*		C
I	N	*		H

↓ Rot. 3

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

↓ Rot. 4

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

• CRITTOANALISI STATISTICA

La sicurezza del cifrario è legata alla dimensione dello spazio delle chiavi ma spesso la vulnerabilità non sono nell'obiettivo ma nelle applicazioni: chiavi male mala, si conosce il formato del menaggio. Nella crittoanalisi statistica il cifrario è fortato tramite analisi delle frequenze ed i cifrari storia sono stati violati con attacchi di tipo cipher text. La crittoanalisi statistica è nota nel XIX secolo.

Si suppone che il crittoanalista conosca il meccanismo di cifratura/decifratura, il linguaggio notiziale in cui è scritto il menaggio e che questo sia sufficientemente lungo per rilevare dati statistici (digrammi, trigrammi, ...)

↓ Attacchi

Per sostituzione monoalfabetica: se y nel crittoagramma corrisponde a x nel menaggio, frequenza (y) = frequenza (x) . Si confrontano quindi la frequenza delle lettere e si fa uno confronto. Per il cifrario di Cesare basta trovare tutte. Se abbiamo un cifrario offre ci basta ricavare il sistema di 2 equazioni per trovare a e b .

Per sostituzione polialfabetica: l'intogramma delle frequenze appare appiattito. Per Vigenère, ogni lettera y del crittoagramma corrisponde ad una coppia $\langle x, k \rangle$ proveniente rispettivamente dal menaggio e dalla chiave. La debolezza è la ripetizione della chiave → Se k ha la stessa lettera di k → Monoalfabetico. L'intero $i \leq h$, accompagnano il menaggio tutti in sovrapposizione $m[i:j]$ formati dalle lettere $i, i+1, \dots, i+h-1$ → lettere allineate con stessa lettera della chiave: può applicare la cifratura monoalfabetica → Per la sicurezza, k deve essere lunga. Per scoprire k è possibile avere la frequenza di q -grammi ripetuti. Si vedono

quindi gruppi di caratteri ripetuti più volte → corrispondono a q-gruppi frequenti cifrati con la stessa posizione della chiave. cerchiamo quindi le rotelle che sono identiche che saranno nelle posizioni pi e pi e ri come d = pi - pi

come lunghezza della chiave o un suo multiplo

Ric ~~transcrivere~~: Studiamo i q-gruppi

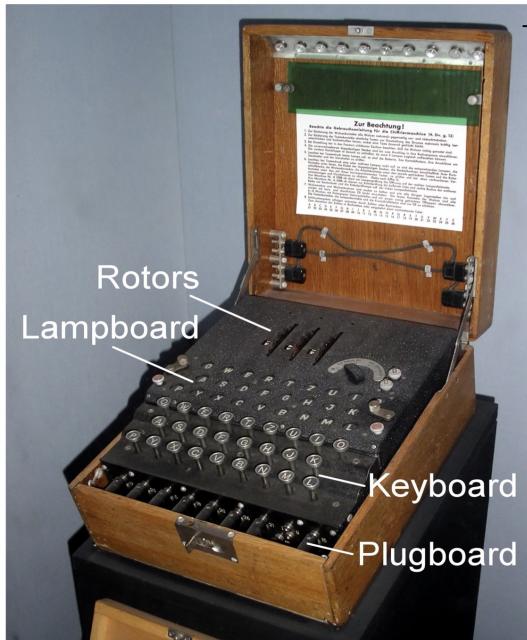
permutazione semplice:

conoscendo h, si divide cattogramma in posizioni di lunghezza h e si cercano i gruppi di q-lettere che formano i q-gruppi frequenti

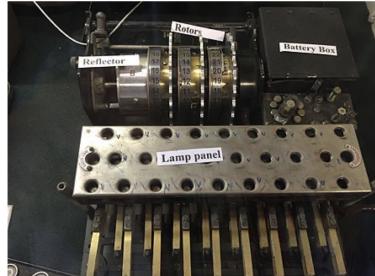
Tramite lo studio degli integratori possono studiare gli integratori delle frequenze

• MACCHINA ENIGMA

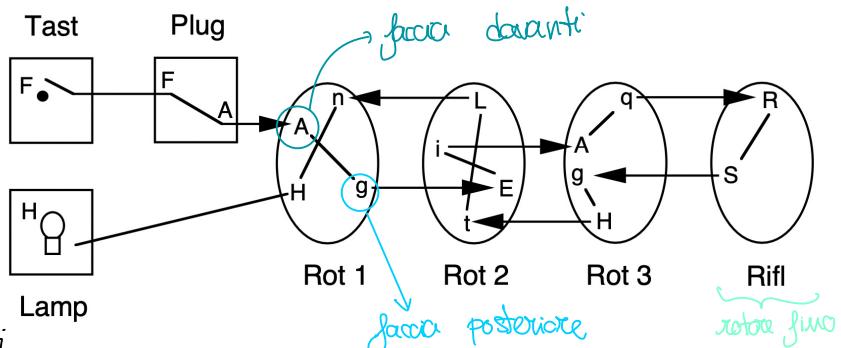
L'evoluzione verso sistemi automatizzati → nata nella II guerra mondiale e molti tentativi di violare → estensione di cattogramma Alberghi



→ Scavavano il menaggio e il cattogramma appare attraverso le luci sopra. Si fa lo stesso per la decifrazione



I rotori si muovevano durante la cifratura. Quando il primo aveva fatto 26 passi si muoveva il secondo e così via. La chiave cominciava quindi ad ogni passo numero 26 per 1° rotore, 26 per 2° rotore e per 1° e anche per il terzo: 26³ permutazioni note a tutti i proprietari di Enigma (rotori eguali)



Aumento chiavi: possibilità di permutare rotori → $(3!) \cdot 26^3 > 10^5$
aggiunta plugboard → Scambia caratteri di 6 coppie contemporaneamente in ogni trasmissione

Ogni collegamento corrisponde a 12 caratteri, quindi $\binom{26}{12} \sim 10^7$. Ogni gruppo di 12

caratteri può presentarsi in 12! permutazioni diverse ma non tutte producono effetti

diversi ad inoltre ad esempio ricevere AB
e BA è la stessa cosa
↓ Totale

$$\binom{26}{12} \cdot \frac{12!}{6! \cdot 6!} \text{ permutations} > 10^{14}$$

stesso scambio fra AB e BA

Altro modo: > 10^5 diversi simboli rotori

$$\underbrace{\binom{26}{2}}_{\text{1^a coppia}} \underbrace{\binom{24}{2}}_{\text{2^a coppia}} \binom{22}{2} \binom{20}{2} \binom{18}{2} \binom{16}{2} = \frac{26!}{2^6 \cdot 16!} = \binom{26}{12}^{12!}$$

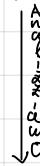
✓ e poi si divide per 6!

Nella II guerra mondiale i rotori erano diventati 3 tra cui scegliere 3 e
avvertano da 6 a 10 le coppie accoppiabili con la plugboard. I tedeschi tenevano
l'avetto iniziale in un registro con il quale trascrivono la chiave utilizzata.

Cifrai perfetti

INTRODUZIONE

2 tipi cifrai



Sicurezza incondizionata:

non condono informazione con certezza assoluta

Sicurezza computazionale:

non condono informazione se certezza ha limiti (P ≠ NP)

Cifrai perfetti: sicurezza mantenuta indipendentemente da informazioni prese col canale → crittografia non fornisce informazioni: concetto formalizzato da Shannon

2 spazi: MSS = spazio dei messaggi
 $CRITO$ = spazio dei crittogrammi

+

U = variabile aleatoria che descrive il comportamento del mittente, e assume valori in MSS

C = variabile aleatoria che descrive la comunicazione sul canale, e assume valori in $CRITO$

$P(U=u)$: probabilità che il mittente voglia inviare $m \in MSS$

$P(U=u | C=c)$: probabilità condizionata che il messaggio sia m , posto che sul canale tratta $c \in CRITO$

Un cifrario è perfetto se $P(U=u | C=c) = P(U=u)$.

Il crittoanalista convoca, nel worst case: la distribuzione di probabilità con cui il mittente invia il messaggio, il cifrario utilizzato e lo spazio delle chiavi

↓ Esempi

1. $\bar{m} \in MSS$: $P(U=\bar{u}) = p > 0$

$0 < p < 1$, $\exists \bar{c} / P(U=\bar{u} | C=\bar{c}) = 1 \rightarrow$ se c è questo

\bar{m}

2. $\bar{m} \in MSS$: $P(U=\bar{u}) = p > 0$

$0 < p < 1$, $\exists \bar{c} / P(U=\bar{u} | C=\bar{c}) = 0 \rightarrow$ se c non è mai questo

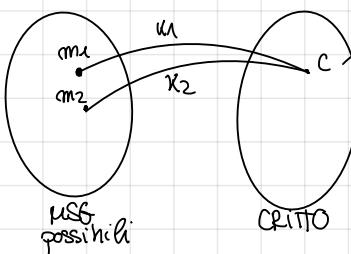
\bar{m}

TEOREMA di SHANNON:

In un cifrario perfetto il numero delle chiavi deve essere almeno quanto il numero dei messaggi possibili

$m \in MSS$ è un messaggio possibile se $P(U=m) > 0$

$N_K = \# \text{chiavi}$, $N_M = \# \text{messaggi possibili}$. Per assicurare sia $N_M > N_K$.



$$P(C=c) > 0$$

Provo a decifrarlo con più chiavi. Può succedere che decifrandolo con K_1 e K_2 diverse si ottengono allo stesso messaggio

$S = \# \text{messaggi che possono corrispondere al crittogramma } c \rightarrow S \leq N_K < N_M$

$$P(U=m') > 0$$

$\exists m'$ è MSG possibile, che non può corrispondere a c

$$P(U=m' | C=c) = 0$$

◦ ONE - TIME PAD

Blocco pieno di hit che uniamo come chiave una volta \rightarrow MSG, CRITTO, KEY = $m_1 \oplus m_2 \oplus \dots \oplus m_n$
Avremo che $c = m \oplus k$ (XOR hit a hit). $m = m_1 m_2 \dots m_i \dots m_n$
 $k = k_1 k_2 \dots k_i \dots k_n$
 $c = c_1 c_2 \dots c_i \dots c_n$ ove $c_i = m_i \oplus k_i$ legge cifratura

$$\begin{array}{l}
 \text{EXEMPLO:} \\
 \begin{array}{ccccccccccccc}
 m & = & 10 & 10 & 10 & 10 & 10 & 10 & 10 & 10 & 10 & 10 & 10 \\
 h & = & \underline{1001011100010110111} \\
 c & = & 001111011011110001 \\
 \hline
 & & \underline{1001011100010110111}
 \end{array} \\
 m = 10 \underset{\downarrow}{10} 10 10 10 10 10 10 10 10
 \end{array}$$

NON posso chiudere la stessa chiave: $C_1 = m_1 \oplus K$
 $C_2 = m_2 \oplus K$

\downarrow

$C_1 \oplus C_2 = (m_1 \oplus K) \oplus (m_2 \oplus K) = (m_1 \oplus m_2) \oplus \underbrace{(K \oplus K)}_{\emptyset} = m_1 \oplus m_2$: traffici di
 \emptyset : traffici di msg =

DIMOSTRÀMO che ONE-TIME PAD è PERFETTO

Ipotesi:

1. Tutti i menaggi hanno lunghezza n (padding se $|m| > n$, cifratura a blocchi lunghezza n se $|m| > n$)
2. Tutte le sequenze di n bit sono menaggi possibili (ci corrisponde una probabilità molto bassa, ma comunque > 0 , alle sequenze hanno le stesse probabilità)
3. La chiave deve avere costante per ogni menaggio \rightarrow usiamo approssimazione

Sotto questo ipotesi ore - tive per è perfetto è impiegata un numero minimo di chiacchiere ($Nu = Nu_m$)

minimicità: segue immediatamente dal fatto che $N_m = N_h = 2^n$

cifrario regolare \rightarrow tesi : $\forall m \in MSG, \forall c \in CRYPTO, P(M=m | C=c) = P(M=m)$

$$\hat{P}(M=m \mid C=c) = \frac{\hat{P}(M=m \cap C=c)}{\hat{P}(C=c)} \rightarrow \text{finato } m, \text{ chiavi + producono } \text{ ottogrammi +} \\ \text{def. probabilità condizionata}$$

$P(C=C)$ = probabilità di neglire a caso l'unica chiave che porta in casa C e quindi

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

$$\frac{P(M=m \cap C=c)}{P(C=c)} \downarrow \text{event indip.}$$

$$\frac{P(\mu = \text{cm}) \cdot P(c = c)}{P(c = c)} = \underline{P(\mu = \text{cm})}$$

SCAMBIO della CHIAVE

potrebbe essere usato un generatore uguale (crittograficamente sicuro) con lo stesso nome per crittografare prima e decrittografare dopo. → Il generatore può diventare noto e quindi la lunghezza dei numeri lunghi per proteggerlo da attacchi sul nome. Inoltre i generatori hanno periodi e quindi la chiave dopo un po' sarebbe ripetuta (dovrei avere periodi molto lunghi). Se ci tentasse di mantenere segreto il generatore, il crittoanalista potrebbe effettuare un attacco sui generatori noti e quindi sarebbe meglio inventarne uno nuovo (troppo difficile). Si possono invece usare file

prei in rete come sequenze di hit carudi.

Se proviamo a togliere i poteri di tutti i messaggi possibili e cominciamo solo messaggi suggeritivi (nella lingua inglese $\sim \alpha^n$ con $\alpha: 1.2$) $\rightarrow N_K \geq N_m = \alpha^n < 2^n \rightarrow$ non devono essere le chiavi con t hit con $t / 2^t > \alpha^n : t \geq n \log_2 \alpha \approx 0.12 n$ e quindi non ridurre hit carudi a t (poi non sono una regola deterministica per estenderli ad m) $\rightarrow \Delta \alpha^t$ attacco forta brutta riconosciuto però ad essere significativo. Per confondere il crittoanalista è opportuno fare in modo che molte coppie (messaggio, chiave) producano lo stesso ottogramma (sconfigge forza brutta) \downarrow per farlo

chiavi di t hit carudi: $\alpha^n \cdot 2^t \gg 2^n$ e quindi $n \log_2 \alpha + t \gg n$ da cui

$t \gg n - n \log_2 \alpha = n(1 - 0.12)$ $\rightarrow t \gg 0.88 n$: non possono rispondere molti hit se vogliono confondere crittoanalista

Cifrari Simmetrici

INTRODUZIONE

Cifrari per la comunicazione di chiavi → sicurezza computazionale: $T \neq NP$

Usiamo AES: preceduto da DES → standard per comunicazioni riservate, nota, chiavi fissate (128 o 256 bit) cambiate al minimo. Cifrat area a blocchi che ha cifratura blocchi indipendente

principi di Shannon: **Diffusione** → il testo in chiaro si deve spargere in tutto il crittogramma: ogni carattere deve dipendere da tutti i bit
Confusione → chiave e messaggio sono combinati in modo complesso in modo da non essere semplici con ortogonali

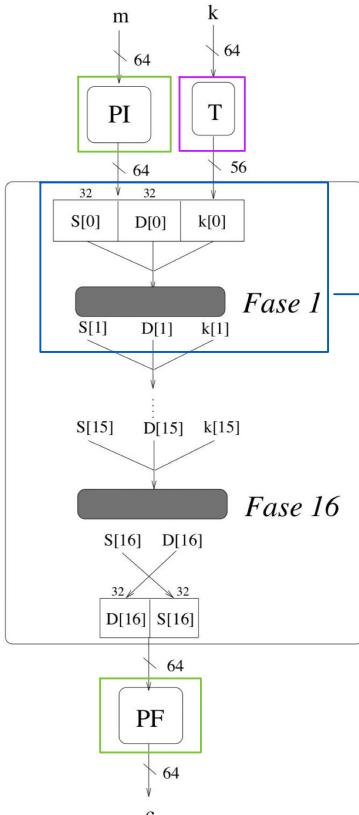
DES

Inventato da IBM negli anni '70 in un contesto di necessità crescente di crittografia → Standard FATO del NIST: richiede che cifratura e decifratura siano pubblici → Anni: 1972 = primo bando, nessun cifrario
 1973 = secondo bando, IBM propone Lucifer → Variato dal NSA: riduce chiave (128 bit iniziale) e cambia S-box (area). Chiave diventa di 64 bit (56 carri) e varia S-box.

Ogni 5 anni il DES è stato ristudiato e nel 1999 è stato riconosciuto l'uso → Disueno nel 2005

Struttura

Messaggio diviso in blocchi cifrati e decifrati indipendentemente. I blocchi sono di 64 bit e subiscono 16 manipolazioni uguali (diffusione e confusione). La chiave è fatta da 2 liste in cui, in ciascuna lista, i bit sono carri e l'ottavo è quello di parità (= se è pari, 0 altrimenti) → Sicurezza di correttezza della chiave. Dai 56 bit si estraggono 16 rotocalchi di fare: ogni bit della chiave viene usato in 14 delle 16 fasi



Permutazione: solo nella realizzazione hardware. Non importanti

Trasposizione: prende 64 bit chiave, toglie via quelli di parità e li permuta

Blochi messaggio divisi in parte dx e parte sx e partendo da dx a 32 bit e la rotocalchi di fare ed otteniamo $S[i], D[i], K[i]$

Con viene fatto

$$S[i] = D[i-1]$$

$$D[i] = S[i-1] \oplus f(D[i-1], K[i-1])$$

non lineare: $f(a \oplus b) \neq f(a) \oplus f(b)$

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	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

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

Permutazione PF

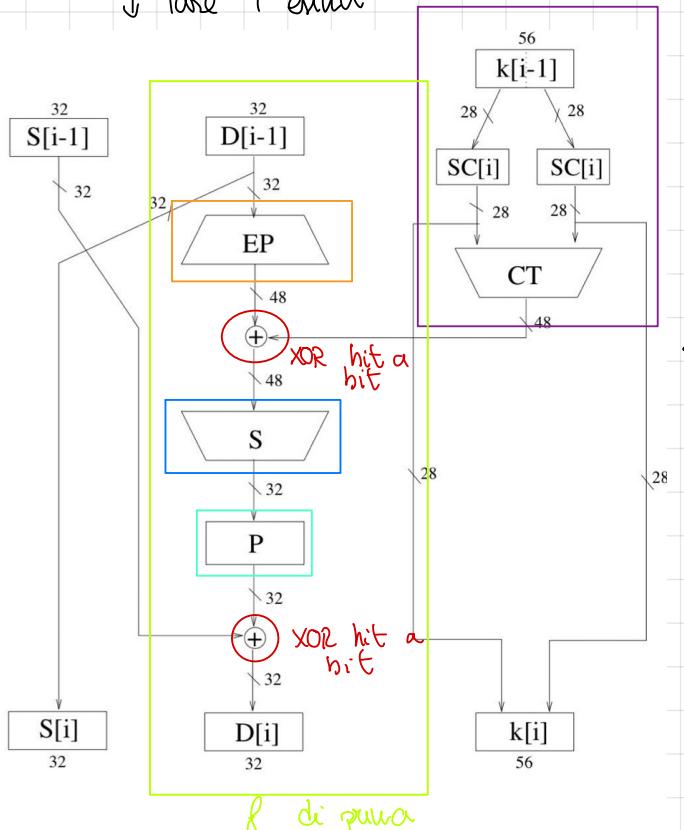
57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	52	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

Trasposizione T

$$m = m_1 m_2 \dots m_n \rightarrow m_{n_1} m_{n_2} \dots m_{n_n}$$

Toglie mult. 8 (parità) + permutazione chiave

↓ Funzione mixing



Entrata rottochiare de fune: K in input diviso in 28 bit ciascuno ed in ciascuno viene fatto uno shift cedendo a dx di 1 per $i = 1, 2, 9, 16$ e di 2 per le altre fasi. Questo permette di selezionare tutti i bit. Le requeste vengono poi concatenate e vengono usate per la fune successiva selezionando 48 bit con CT.

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

La funzione CT

otto bit dell'ingresso (e.g., il bit 09) non sono presenti in uscita

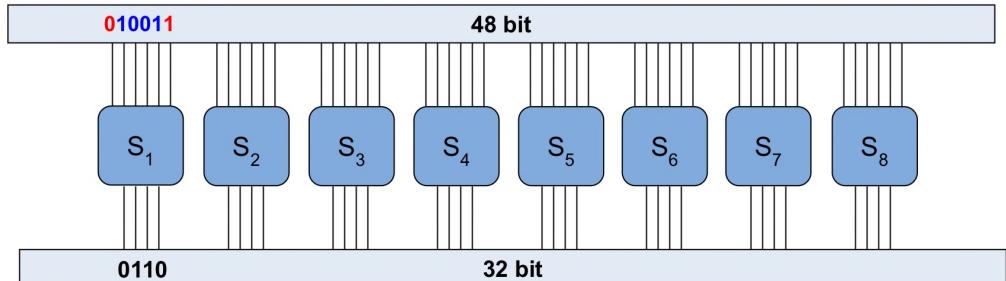
La funzione EP

sedici bit di ingresso sono duplicati (e.g., il bit 32 è copiato nelle posizioni 1 e 47 dell'uscita)

→ Permutation che espande i hit da 32 a 48 (duplicati 16 hit)

→ S-box: funzione non lineare cifrario che trasforma i hit riducendone il numero

→ Permutazione S-box



8 funzioni ciascuna delle quali prende 6 hit in input e ne restituisce 6. Sono tutte nate

x	y	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	14	04	13	01	02	15	11	08	03	10	06	12	05	09	00	07
1	0	00	15	07	04	14	02	13	01	10	06	12	11	09	05	03	08
2	0	04	01	14	08	13	06	02	11	15	12	09	07	03	10	05	00
3	0	15	12	08	02	04	09	01	07	05	11	03	14	10	00	06	13

Si prende il primo e l'ultimo hit dentre i 16 hit interni corrispondenti in uscita sufficienti. Su ogni riga

e si seleziona la riga corrispondente all'indice di colonna. Si prende la 0 a sinistra se hit non 0 a 15

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

ESERCIZIO

- Siano

$$\begin{aligned} c &= C_{\text{DES}}(m, k) \\ c^* &= C_{\text{DES}}(m', k') \\ c^{\wedge} &= C_{\text{DES}}(m, k') \end{aligned}$$

dove, m' e k' sono ottenute complementando bit a bit m e k .

- Spiegare se vi è una semplice relazione tra $c \oplus c^*$ e tra $c \oplus c^{\wedge}$.

$$\begin{aligned} \text{con } c \oplus c^* : \quad bi' \oplus ki' &= (hi \oplus i) \oplus (ki \oplus i) = \\ &= hi \oplus ki \oplus \cancel{(i \oplus i)} = hi \oplus ki \end{aligned}$$

quindi l'input di $s\text{-box}$ è identico. c cambia perché $s(i)$ è complementato $\rightarrow c^* = \overline{c}$

Com $c \oplus c^{\wedge}$: la $s\text{-box}$ riceve un input i e quindi cambia tutto perché la $s\text{-box}$ non è lineare e quindi i non ha relazioni con c

attacchi

Shide del DES fette alla RSA nel 1997: 10.000\$ a chi decifra crittogramma. Vinti da informatico che ha distribuito algoritmo distribuito eseguito su più macchine. Il crittogramma era "strong cryptography makes the world a safer place". Nel 1993 c'è stata un'altra shide che ha trovato in 39 giorni la chiave esplorando l'8% della chiave (2⁵⁶ preceduenti). Sono state poi sviluppate architetture ad-hoc per fortificare il DES.

↓ Attacchi accademici

Un cifrario ha b hit di sicurezza se il costo del miglior attacco noto è $O(2^{b/2})$ operazioni di derivație \rightarrow i hit del DES sono 55 in quanto $C_{\text{DES}}(m, k) \oplus c$ implica $C_{\text{DES}}(m', k) = c'$ e m controllano simultaneamente. Quando si prova una chiave k ne posso battere sia z . Si fa un attacco chosen plain text: $(m, c) \rightarrow (\bar{m}, c')$. $\forall k$, provo a fare $C(m, k)$ e se sono fortunato ottengo C_1 . In tal caso probabilmente la chiave è K . La chiave è probabilmente k perché ci potrebbe essere altre coppie che ragionano il messaggio in C_1 . Se invece non ottengo C_1 vedo se $C(m, k) = \bar{C}_2$. Se è così, k è la chiave perché $C(m, k) = \bar{C}_2$ e, per la proprietà precedente $C(\bar{m}, \bar{k}) = \bar{C}_2 = C_2$. Se invece $C(m, k) \neq C_1$ e C_2 , k è \bar{k} non sono chiavi e quindi ne butto via due. Negli anni '90 è stata diffusa una nuova tecnica che è la crittandini diffaenitiche che prevede che il crittandista abbia a disposizione $2^{k/2}$ copie (m, c) . E una tecnica che il crittandista abbia a disposizione $2^{k/2}$ copie (m, c) . E una crittandini differenti nel messaggio ma differente nel messaggio in probabilità delle chiavi. Nel 1993 è stata introdotta la crittandini lineare che richiede $2^{k/2}$ copie (m, c) ed è un attacco plain text (non chosen). Questa tecnica ad appena la $s\text{-box}$ con una funzione lineare che permette di trovare alcuni hit della chiave \rightarrow vulnerabile.

Alternative

- Selezione rotabili di fare sempre diverse: $16 \cdot 63 = 768$. Non sono tutti hit di sicurezza in quanto per la crittandini diffaenitiche i hit sono 61
- Cifratura multipla: Date k_1 e k_2 arbitrarie, $C_{\text{DES}}(C_{\text{DES}}(m, k_1), k_2) \neq C_{\text{DES}}(m, k_2)$. H_{k_1, k_2} due chiavi di maniera di 56 hit producono 57 hit di sicurezza

↓

Attacchi "meet in the middle": $c = C_{\text{DES}}(C_{\text{DES}}(m, k_1), k_2) \rightarrow D_{\text{DES}}(c, k_2) = C_{\text{DES}}(m, k_1)$. Il crittandino in pratica una coppia (m, c) . H_{k_1} calcolo e salvo $C_{\text{DES}}(m, k_1)$ [2^{56}]. H_{k_2} , calcolo $D_{\text{DES}}(c, k_2)$ e lo cerco nella prima lista. Se è presente, la coppia k_1, k_2 è quella cercata. Il costo è $O(2^b + 2^b) = O(2^{b+1})$

3. Composizione tripla del DES a 2 o 3 chiavi

3TDES: $C = \text{DES}(\text{DES}(m, k_1), k_2, k_3)$ $\quad \quad \quad C, D \text{ per retrocompatibilità con DES}$

2TDES: $C = \text{DES}(\text{DES}(m, k_1), k_2, k_1)$

↓

Entrambe le alternative hanno una sicurezza di 112 bit e sono state certificate dal NIST fino al 2005

↓ Meet in the middle

$\text{DES}(C, k_2) = \text{DES}(\text{DES}(m, k_1), k_2) \rightarrow \text{DES}(\text{DES}(C, k_2), k_2) = \text{DES}(m, k_1)$.

data una coppia (m, C) :

- k_1 , si calcola e si risolve $\text{DES}(m, k_1) : O(2^{56})$
- + connica k_2, k_3 si calcola $\text{DES}(\text{DES}(C, k_2), k_2)$ e si cerca nella tabella. Se c'è, le 3 chiavi sono quelle suate. $O(2^{56} \cdot 2^{56})$

$O(2^{56} + 2^{56} \cdot 2^{56}) = O(2^{112})$: costa come il forza bruta. Non è stato più usato in quanto le implementazioni su sono pesanti

o AES

Giugno 1990: NIST chiede nuovo sistema di cifratura

↓ requisiti

- SICUREZZA**: resistenza agli attacchi noti, correttezza del protocollo di cifratura, "comodità" dell'usata
- COSTO DI REALIZZAZIONE**: costo realizzazione basso, velocità cifratura/decifratura, occupazione memoria bassa
- CARATTERISTICHE ALGORITMICHE**: funzionalità, portabilità, applicabilità a diversi dispositivi, 128, 192, 256 bit

21 cifrari proposti, 15 selezionati nell'agosto del 97

Aprile 1999: 5 cifrari:

- MARS (IBM)
- RGS (RSA)

Rijndael (Proton World Int + Università Louvain, Belgio)

TWOFISH (Berkeley, Princeton)

Serpent (univ. Israele, M., USA)

2001: Scelto come nuovo cifrario AES

↓ Più versatili

	chiave (bit)	dim. blocco (bit)	Numero di fasi
AES (128)	128	128	10
AES (192)	192	128	12
AES (256)	256	128	14

$\frac{1}{2}$
 b
 t

128 bit: 16 byte → matrice di 16 byte:

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

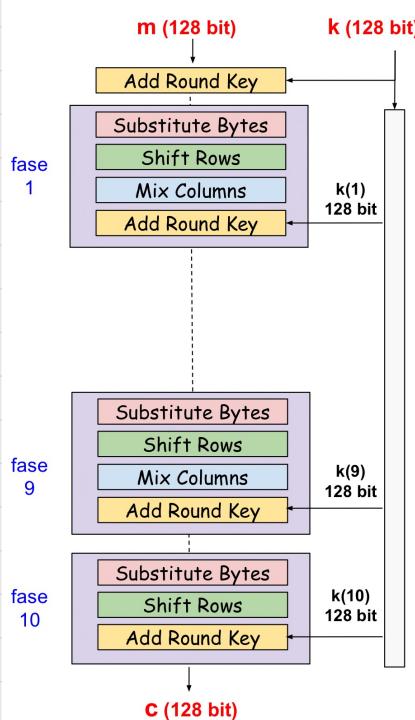
W[0]	W[1]	W[2]	W[3]
$k_{0,0}$	$k_{0,1}$	$k_{0,2}$	$k_{0,3}$
$k_{1,0}$	$k_{1,1}$	$k_{1,2}$	$k_{1,3}$
$k_{2,0}$	$k_{2,1}$	$k_{2,2}$	$k_{2,3}$
$k_{3,0}$	$k_{3,1}$	$k_{3,2}$	$k_{3,3}$

Si estraggono da questo 10 sottocinque di fine tutte lunghe 128 bit. Espandiamo la matrice da 16 a 48 colonne.

W[0]	W[1]	W[2]	...	W[42]	W[43]

Gestore delle chiavi: $W[i,j] = W[i-4] \oplus W[i-1]$, se i non è multiplo di 4
 $W[i,j] = W[i-4] \oplus T(W[i-1])$, se i è multiplo di 4 e T è
 genera confusione una trasformazione non lineare (S-box)
 La chiave per l' i -esima fase sarà data dalle 4 colonne
 $W[Li], W[Li+1], W[Li+2], W[Li+3]$

↓ Struttura generale



Substitute bytes: Applicazione della S-box (senza fare commutazione)

della forma tabellare

byte

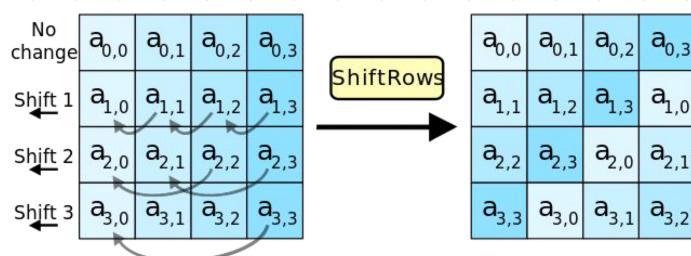
b _{0,0}	b _{0,1}	b _{0,2}	b _{0,3}
b _{1,0}	b _{1,1}	b _{1,2}	b _{1,3}
b _{2,0}	b _{2,1}	b _{2,2}	b _{2,3}
b _{3,0}	b _{3,1}	b _{3,2}	b _{3,3}

a _{0,0}	a _{0,1}	a _{0,2}	a _{0,3}
a _{1,0}	a _{1,1}	a _{1,2}	a _{1,3}
a _{2,0}	a _{2,1}	a _{2,2}	a _{2,3}
a _{3,0}	a _{3,1}	a _{3,2}	a _{3,3}

$$a_{i,j} = S\text{-box}(b_{i,j})$$

16 · 16 byte de contiene una combinazione degli interi da 0 a 255. Per trasformare il byte in ingresso in quello di uscita si prendono i primi 4 bit per selezionare la riga dello tavola di uscita della S-box e gli ultimi 4 bit per selezionare la colonna. All'intervalle troviamo un intervallo da 0 a 255. La S-box calcola l'intervallo moltiplicativo di ogni byte visto come elemento del campo finito di Galois di ordine 2^8 . Si pensi infatti al byte come polinomio ore i coefficienti sono i bit che lo costituiscono. Per la somma si sommano bit a bit modulo 2 che equivale alla somma di polinomi. Per il prodotto si muove invece a causa dell'aritmetica di grado

Shift rows: Shift calcolo delle righe



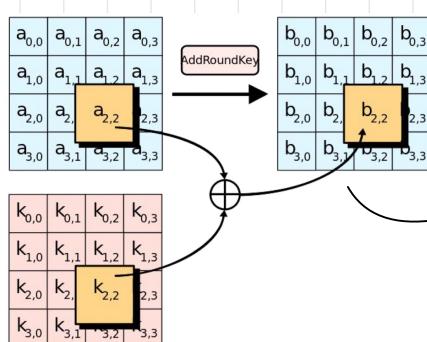
Ogni colonna si diffonde sulle altre

Mix columns: Fa in modo che ogni byte del blocco renviato dipenda da tutti i byte della colonna di partenza. La moltiplicazione è mod 2^8 e la somma mod 2.

Si parla di ogni blocco < i multiplichi per una matrice: il risultato dipende da tutti i byte della colonna

Typo 2 round si dice diffusione totale: ogni bit dell'output dipende da tutti i bit di input

Add Round Key:



Ogni byte della matrice è posto in XOR hit a hit con la chiave locale di fase

Input a fune sconsigliata

Un attacco ha compreso AES numero volte veriore più semplice a 128 bit. Ci sono degli attacchi più efficienti del forte bruta se le chiavi sono 6. Per attaccare AES si usano attacchi side-channel che sfruttano le vulnerabilità della piattaforma nella quale il cifrario è implementato.

↓ osservazioni

1. Se il messaggio non è multiplo della lunghezza del blocco si fa il padding.
2. La cifratura a blocchi esige ad attacchi in quanto blocchi uguali del messaggio producono blocchi uguali nel ciphogramma in quanto non c'è diffusione tra un blocco e l'altro. Questo è utile per la determinazione.

Si fa in modo che la cifratura di un blocco dipenda da quelli precedenti: CBC (Cipher Block Chaining) → Si compongono blocchi tra loro in modo che blocchi di messaggio uguali producano ciphogrammi diversi.

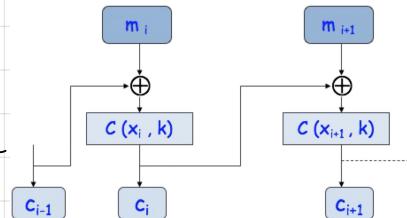
↓ ciphogramma blocco precedente

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

blocco msg

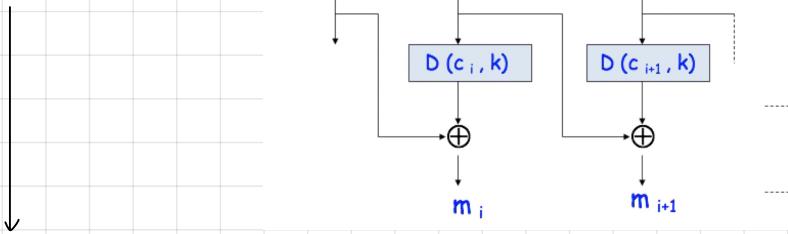
→ Procedimento sequenziale: ricevuto
cifratura precedente

Si dice CFB: concordante con utente



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

→ Può essere parallelizzato



Se si ha qualche bit errato nel blocco successivo e non si pregeva su tutto il ciphogramma l'errore si limita al blocco e

• ALTRI CIFRARI

RGS

- simile al DES, ne adotta la struttura migliorandone alcune parti
- lascia più libertà all'utente
- blocchi di 64 bit
- chiave di $c \times 32$ bit
- r fasi (valore consigliato: $r = 16$)
- r e c possono essere scelti a piacere
- usa shift ciclico, XOR, addizione mod 2^{32}
- molto veloce, resiste con successo agli attacchi standard se c e r sono scelti bene
- sicuro e impiegato con una certa frequenza
- SEMPLICITÀ DI REALIZZAZIONE E GRANDE SICUREZZA

IDEA

- 1992
- chiave da 128 bit
- usa shift ciclico, XOR, moltiplicazione mod $(2^{16}+1)$ e addizione mod 2^{16} ,
- più semplice e più sicuro del DES
- la sicurezza poggia su basi teoriche molto forti
- è rimasto assolutamente inviolato, ma non è diventato il nuovo standard

Esercizi:

Esercizio 6

La proprietà di non linearità di qualsiasi cifratura a blocchi è fondamentale per la sua sicurezza. Infatti, si supponga di avere una cifratura lineare a blocchi C che cifra blocchi di 128 bit di testo in chiaro in 128 bit di testo cifrato, usando una chiave k (la lunghezza di k è irrilevante). Dunque, per ogni coppia di messaggi m_1 e m_2 risulta

$$C(m_1 \oplus m_2, k) = C(m_1, k) \oplus C(m_2, k).$$

Descrivere come un avversario che abbia 128 testi cifrati scelti possa decifrare qualsiasi testo cifrato senza conoscere la chiave segreta k .

NOTA: "testo cifrato scelto" significa che l'avversario ha la possibilità di scegliere un testo cifrato e ottenerne la decifrazione. In questo caso si hanno 128 coppie di testo in chiaro/testo cifrato e si ha la possibilità di scegliere i testi cifrati.

$$C = C(m, k) \quad \text{con} \quad C \quad \text{lineare}$$

$$C = c_1 c_2 \dots c_{128} = \bigoplus_{i=1}^{128} e^{(i)} \rightarrow e^{(i)} = 00 \dots 010 \dots 0 \quad 128 \text{ bit}$$

$$c = 1011 = 1 \cdot (1000) \oplus 1 \cdot (0010) \oplus 1 \cdot (0001) = 1011$$

Chiedo la decifrazione degli $e^{(i)}$
Vi, $1 \leq i \leq 128$, $f^{(i)} = D(e^{(i)}, k)$

$$\begin{aligned} \text{Avendo } C, \text{ voglio trovare } M \longrightarrow m &= DC, k) = D\left(\bigoplus_{i=1}^{128} e^{(i)}, k\right) = \bigoplus_{i=1}^{128} D(e^{(i)}, k) = \\ &= \bigoplus_{i=1}^{128} f^{(i)} \end{aligned}$$

Esercizio 4

La S-box del cifrario AES è composta da 16 blocchi a 8 ingressi e 8 uscite ciascuno. Qual è il numero totale di possibili funzioni che si potrebbero scegliere per costruire ciascun blocco?

$$2^8 \text{ input. Input posso scegliere uno fra } 2^8 \text{ output} \longrightarrow (2^8)^{2^8} = 2^{8 \cdot 2^8} = 2^{256}$$

Esercizio 8

Si consideri un cifrario simmetrico a blocchi. Nel metodo FSM (Fischer Spiffy Mixer) ogni blocco m_i del messaggio in chiaro viene cifrato come

$$c_i = m_{i-1} \oplus C(m_i \oplus c_{i-1}, k), \quad i \geq 1$$

sando due sequenze di inizializzazione fissate (e pubbliche) m_0 e c_0 .

1. Descrivere come eseguire la decifrazione di un blocco.
2. Nel caso in cui il blocco di crittogramma c_i si danneggi nel corso della trasmissione, quali blocchi di testo in chiaro diventano indecifrabiili?

$$1. \quad c_i \oplus m_{i-1} = C(m_i \oplus c_{i-1}, k)$$

$D(c_i \oplus m_{i-1}, k) = D(C(m_i \oplus c_{i-1}, k), k) \longrightarrow D(C \oplus m_{i-1}, k) = m_i \oplus c_{i-1}$ da cui
si ottiene $m_i = c_{i-1} \oplus D(c_i \oplus m_{i-1}, k)$

2. Se c_i è danneggiato l'errore si propaga anche ai blocchi successivi

Esercizio 7

DESX è un cifrario proposto da Rivest per proteggere il DES dagli attacchi esaurienti. DESX usa una chiave segreta w di 64 bit oltre alla chiave DES k di 56 bit, e opera nel modo seguente:

$$C_{\text{DESX}}(m, k, w) = w \oplus C_{\text{DES}}(m \oplus w, k).$$

Mostrare come eseguire la decifrazione.

$$C = C_{\text{DESX}}(m, k, w)$$

↓

$$C = w \oplus C_{\text{DES}}(m \oplus w, k) \longrightarrow w \oplus C = w \oplus w \oplus C_{\text{DES}}(m \oplus w, k)$$

↓ funz. decifrazione

$$D_{\text{DESX}}(w \oplus C, k) = D_{\text{DESX}}(C_{\text{DES}}(m \oplus w, k), k) = m \oplus w$$

↓

$$m = w \oplus D_{\text{DESX}}(w \oplus C, k)$$

Cifografia a chiave pubblica

Ricorre il problema dello scambio della chiave. Prima di questo le soluzioni possibili sono quelle di scambiarsi $N(N-1)/2$ chiavi per N copie di utenti o rendersi ad una trusted 3rd party (TP). Quest'ultimo modo prevede da parte degli utenti di avere una chiave per comunicare iutamente con la TP. Non solo comunicare la TP genera una chiave che mette in grado gli utenti di comunicare. La TP cifra la chiave e la invia agli utenti che le hanno per comunicare. La TP deve essere sempre online e è chiuso perché conosce tutte le chiavi.

↓ Soluzioni

Protocollo di Diffie-Hellman e cifratura a chiave pubblica: la chiave per cifrare è pubblica, quella per decifrare è privata. La coppia di chiavi in cui in una comunicazione è quella creata dal destinatario del messaggio. Se abbiano n utenti avranno quindi $2n$ chiavi $\langle K_{ph}, K_{priv} \rangle$. La cifratura viene fatta ponendo $C = C(m, K_{ph})$ entrambe note mentre la decifratura si ottiene ponendo $m = D(C, K_{priv})$ dove D è pubblica e K_{priv} privata. La cifratura è quindi simmetrica dato che mittente e destinatario hanno rechi diversi. Non sostituisce la simmetria ed una valida verificabilità.

↓ Regolamenti

1. correttezza procedimento cifratura e decifratura: $H(m, D(C(m, K_{priv}), K_{priv})) = m$
2. Efficienza e sicurezza sistema
 - a. $\langle K_{ph}, K_{priv} \rangle$ facile da generare e caricate
 - b. dati m e K_{ph} è facile calcolare $c = C(m, K_{ph})$.
 - c. dati c e K_{priv} è facile calcolare $m = D(c, K_{priv})$.
 - d. Bere avere difficoltà per il catturando, conoscendo c , K_{priv} , C e D tenere c e m

↓

Funzione one-way trap-door → RSA: trovano la funzione (fattorizzazione) e invertitano cifratura

• CIFRARIO RSA

Si ha nella moltiplicazione di 2 numeri primi p e q . Calcolare $N = p \cdot q$ è facile ma è difficile da invertire (fattorizzazione, esponenziali). Trap door: se si conosce uno dei fattori, trovare l'altro è facile. Utilizza l'algebra modulare del Galois: problema del logaritmo discreto

Algebra modulare Richiami

La crittografia ne fa grande uso perché per i protocolli usa la divisione per il modulo. Questo riduce lo spazio dei numeri e quindi aumenta la velocità di calcolo e fa diventare problemi difficili: le funzioni si comportano in modo imprevedibile perdendo alcune proprietà.

Proprietà: 1. $a \equiv b \pmod n \iff J_K: a \equiv b \pmod n$

intesa relazione

zaccardo

$$\begin{aligned} 2. (a+b) \pmod m &= (a \pmod m + b \pmod m) \pmod m \\ (a-b) \pmod m &= (a \pmod m - b \pmod m) \pmod m \\ (a \cdot b) \pmod m &= (a \pmod m \cdot b \pmod m) \pmod m \\ a^r \pmod m &= (a^r \pmod m)^m \pmod m \end{aligned}$$

Notazione: $Z_n = \{0, 1, \dots, n-1\} \rightarrow Z_n^* \subseteq Z_n$ è l'inverso degli Z_n co-piani comuni
↳ Se n è primo, $Z_n^* = \{1, 2, \dots, n-1\}$
se n non è primo si esegue MCD con ordine di n volte: esponentiale nella dimensione dell'input

Funzione di Euler $\phi(n)$: cardinalità di \mathbb{Z}_n^* \rightarrow se n è primo, $\phi(n) = n-1$

\downarrow Teorema

Se n è composto allora $\phi(n) = n(1 - 1/p_1) \dots (1 - 1/p_k)$ dove p_1, p_k sono i fattori primi di n privi senza moltePLICITA'.

Se n è prodotto di 2 primi (primiprioi) allora $\phi(n) = (p-1)(q-1)$

Teorema di Euler: $f_{n>1}$, a primo con $n \rightarrow a^{\phi(n)} \equiv 1 \pmod{n}$

Piccolo Teorema di Fermat: Per n primo e $a \in \mathbb{Z}_n^*$, $a^{n-1} \equiv 1 \pmod{n}$

Inverso im modulo: $a \cdot a^{-1} \equiv 1 \pmod{n}$

\downarrow Alternativa \downarrow Teo. euler: $a \cdot a^{\phi(n)-1} \equiv 1 \pmod{n}$

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

Teorema: $a \equiv b \pmod{n}$ ammette soluzione se e solo se $\text{mcd}(a, n)$ divide b ed in particolare si trovano $\text{mcd}(a, n)$ sol. distinte. Se a ed n sono coprimi si ha una unica soluzione \rightarrow CNS esistente invendo moltiplicativo.

CALCOLO: \downarrow **Algoritmo di Euclide esteso** (polinomiale) \rightarrow Rischio $ax+by \equiv \text{mod}(a, b)$

```
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>
```

\downarrow

$ax \equiv 1 \pmod{b}$ passo scivella come $ax \equiv b\bar{x} + 1$ per \exists opportuno e ponendo $y = -\bar{x}$ avremo $ax + by \equiv \underbrace{\text{mcd}(a, b)}_{1: \text{coprimi per } \exists \text{ invendo}} \pmod{b}$

ESEMPIO:

$$x = 5^{-1} \pmod{132} \rightarrow 5x + 132y = 1$$

$$<d, x, y> = <d', y', x' - ⌊a/b⌋ y'>$$

$$\text{E_E}(5, 132) \rightarrow <1, 53, \dots> //$$

$$\text{E_E}(132, 5) \rightarrow <1, -2, 1 + 2*26> = <1, -2, 53> //$$

$$\text{E_E}(5, 2) \rightarrow <1, 1, 0-1*2> = <1, 1, -2> //$$

$$\text{E_E}(2, 1) \rightarrow <1, 0, 1-0*2> = <1, 0, 1> //$$

$$\text{E_E}(1, 0) \rightarrow <1, 1, 0> //$$

Teorema cinese del resto: Siano n_1, \dots, n_k interi a due a due coprimi ($\text{MCD}(n_i, n_j) = 1$ quando $i \neq j$). Allora, comunque si scelgano degli interi a_1, \dots, a_k , esiste un'unica soluzione intera x del sistema di congruenze

$$x \equiv a_1 \pmod{n_1}$$

$$x \equiv a_2 \pmod{n_2}$$

...

$$x \equiv a_k \pmod{n_k}$$

modulo il prodotto $n = n_1 \dots n_k$.

e vedi Euclide per esempi

Generatori: se $\alpha \in \mathbb{Z}_n^*$ è un generatore di \mathbb{Z}_n^* se la funzione $a^n \mod n$ per $n \in \mathbb{N}$ genera tutti e solo gli elementi di \mathbb{Z}_n^* . Produce tutti gli elementi in un ordine difficile da prevedere
 ↓ Teoria di Euler
 se $\alpha \in \mathbb{Z}_n^*$ è generato per $k = \phi(n)$ è il generatore quando $a^k \mod n$ per $a \in \mathbb{Z} < \phi(n)$.

↓ Teoria

Se n è primo, \mathbb{Z}_n^* ha almeno un generatore

Per n primo, non tutti gli elementi di \mathbb{Z}_n^* sono suoi generatori.
Problemi rilevanti: 1. Determinare un generatore di \mathbb{Z}_n^* , n numero primo: algoritmo esponentiale con algoritmi randomizzati utilizzando coppie (primo, generatore) campionate

2. Algoritmo discreto: $a^x = b \mod n$, n primo. L'equazione ha una soluzione $\forall b$ se e solo se a è un generatore di \mathbb{Z}_n^* . Si usa il metodo forza bruta (si prova con molti valori di x finché si ottiene b). Algoritmo esponenziale.

Funzioni trap-door: Fattorizzazione

Calcolo radici modulo numero composto \rightarrow calcolare $y = x^{\frac{k}{d}} \mod n$ è facile
 ↓

Calcolare $x = y^{\frac{1}{d}} \mod s$ richiede tempo esponenziale
 Oppure si può invertire ruotato a d se è difficile
 comunque

△ Svantaggi: La crittografia a chiave pubblica è più lenta e vulnerabile agli attacchi chosen plain text: il crittanalista si sceglie x, \dots, m_k di suo interesse, li cifra con $C \in \mathcal{U}_{\text{pub}}$ ottenendo C, \dots, c_k che poi sta in accordo nel codice
 ↓

La crittografia a chiave pubblica si usa per securizzare le chiavi: sono corrette e quindi l'utilizzo dei cifrari a chiave pubblica non è pericoloso compito rischiarante e le chiavi sono corrette per costruzione. La chiave pubblica deve essere estratta da certificati digitali per evitare attacchi man in the middle

• RSA

creazione delle chiavi: Bob (destinatario)

Sceglie p e q primi molto grandi (tali che $n = p \cdot q$ abbia 2048 cifre binarie per protezione fino al 2030 oppure 3072 per protezione oltre il 2030). Utilizzando Miller-Rabin si richiede tempo polinomiale. Dopo di che calcola $n = p \cdot q$ e $\phi(n) = (p-1) \cdot (q-1)$ in tempo polinomiale. Calcola poi $d = e^{-1} \mod \phi(n)$ dove esiste perché $\gcd(e, \phi(n)) = 1 \rightarrow \text{kgcd}(e, \phi(n)) = 1 \rightarrow \text{kgcd}(n, e) = 1$, $\text{kgcd}(n, d) = 1$

Cifratura e decifrazione

m : messaggio = sequenza binaria come numero intero \rightarrow dove avere $m < n$ elementi divisi in blocchi. Questo perché, se $m > n$, $m \equiv m \mod n$ e $m \equiv m \mod n$ otteniamo lo stesso crittogramma. I blocchi hanno dimensione $b = \lceil \log_2 n \rceil$ bit. Nelle applicazioni b è fisico e quindi $m < 2^b < n$: finalmente

CIFRATURA: $c = m^e \mod n$

DECODIFICA: $m = c^d \mod n$

Il tempo polinomiale:

quadrature successive

ESEMPIO: $p=5, q=11$

$$n = pq = p \cdot q$$

$$\phi(n) = (p-1) \cdot (q-1) = 40$$

$$e = 7 \rightarrow \text{MCD}(7, 40) = 1$$

$$d = 7^{-1} \pmod{40} \longrightarrow$$

$$\text{EE}(7, 40) = \langle 1, -17, \dots \rangle$$

$$\text{EE}(40, 7) = \langle 1, 3, -17 \rangle$$

$$\text{EE}(7, 5) = \langle 1, -2, 3 \rangle$$

$$\text{EE}(5, 2) = \langle 1, 1, -2 \rangle$$

$$\text{EE}(1, 1) = \langle 1, 0, 1 \rangle$$

$$\text{EE}(1, 0) = \langle 1, 1, 0 \rangle$$

$$d = -17 = -17 \pmod{40} = 23$$

↓

$$K_{\text{pub}} = \langle 7, 55 \rangle, K_{\text{priv}} = \langle 23 \rangle$$

$$m = 28 \longrightarrow c = 28 \pmod{55}$$

$$m = c^{23} \pmod{55}$$

Dimostrazione correttezza: $D(C(m, K_{\text{pub}}), K_{\text{priv}}) = m$

↓

$$(m^e \pmod{n})^d \pmod{n} = m \pmod{n}, \text{ sviluppando davanti}$$

Caso 1: m ed n coprimi $\lceil \text{MCD}(m, n) = 1 \rceil$

↓

a. Teorema di Euler: $m^{\phi(n)} = 1 \pmod{n}$

b. Def. inverso: $e \cdot d = r\phi(n) + 1, r \in \mathbb{N}, ed = 1 \pmod{\phi(n)}$

$$m^{ed} \pmod{n} = m^{1+r\phi(n)} \pmod{n} = m \cdot m^{r\phi(n)} \pmod{n} = m(m^{\phi(n)})^r \pmod{n} \stackrel{a.}{=} m$$

$$= m \pmod{n} = m$$

Caso 2: $\text{MCD}(m, n) + 1 \rightarrow \frac{p | m}{a.} \text{ o } \frac{q | m}{a.} \text{ ma non entrambi divisibili da } m, n$

a. $p | m \rightarrow m \equiv 0 \pmod{p}$ ma in realtà $\forall k \in \mathbb{N} \quad m^k \equiv 0 \pmod{p}$.
Scegliano $K = e \cdot d \Rightarrow m^{ed} - m \equiv 0 \pmod{p}$

$$\begin{aligned} m^{ed} \pmod{q} &= m^{1+r\phi(n)} \pmod{q} = m \cdot m^{r(\phi(n))(q-1)} \pmod{q} = m(m^{(q-1)})^{r(\phi(n))} \pmod{q} \\ &= m(m^{\phi(q)})^{r(\phi(n))} \pmod{q} = m \underbrace{(1)^{r(\phi(n))}}_{\text{MCD}(m, q) = 1 \text{ perche' } q \nmid m} \pmod{q} \end{aligned}$$

Avevamo quindi $m^{ed} \pmod{q} = m \pmod{q}$ da cui $m^{ed} - m \equiv 0 \pmod{q}$

$$\text{Quindi } \begin{cases} m^{ed} - m \equiv 0 \pmod{p} \\ m^{ed} - m \equiv 0 \pmod{q} \end{cases} \quad \begin{cases} m^{ed} - m \equiv 0 \pmod{p} \\ m^{ed} - m \equiv 0 \pmod{q} \end{cases} \quad \begin{cases} m^{ed} - m \equiv 0 \pmod{p} \\ m^{ed} - m \equiv 0 \pmod{q} \end{cases}$$

$$m^{ed} \pmod{n} = m \pmod{n} = m$$

Sicurezza

Se fattori sovraccarico fare facile potrei calcolare la chiave privata perché no per farla la fattoriizza. Per farla la fattoriizza è difficile ed è difficile trovare il costo per calcolare un crittogramma corrispondente a quello di trovare la chiave privata. Calcolare $\phi(n)$ da n equivale a fattoriizzare n .
↓ Dici. relazione tra fatt. n e calcolo $\phi(n)$

Se puoi fattoriizzare trovi p e q e di conseguenza ho $\phi(n) = (p-1)(q-1) = pq - (p+q) + 1$ da

qui $(p+q) = n - \phi(n) + 1$. In più, $(p+q)^2 = (p-q)^2 + 4n$. Chiamiamo
 $X_1 = p+q$ e $X_2 = p-q$. Dalle relazioni, $X_1^2 = X_2^2 + 4n$ da cui
 $X_2 = \sqrt{X_1^2 - 4n}$. A questo punto $p = \frac{X_1 + X_2}{2}$ e $q = \frac{X_1 - X_2}{2}$. I conti

Sono tutti polinomici e quindi i problemi sono equivalenti.
Un'altra strada che si può seguire è ricavare d da (n, e) ma anche
questo è costoso come fattorizzare.

La fattorizzazione non è più difficile come prima grazie all'avvento delle
potenti di calcolo e all'introduzione di alcuni algoritmi sub-esponenziali (2^{15} hit
di ricetta). Siamo in grado di fattorizzare semi-primi con 703 hit (cc AES)
ed esistono numeri che hanno proprietà che rendono facile la fattorizzazione.
 p e q devono avere molto grandi, $p-1$ e $q-1$ devono avere un fattore
primo molto grande e $\text{MCD}(p-1, q-1)$ deve essere piccolo (ideale è 2) quindi
conviene scegliere p e q / $(p-1)$ e $(q-1)$ siano coprimi. Non si devono
mai avere gli stessi numeri primi. p e q devono inoltre avere distanti
per cui $p-q \sim n^e$ ove e è 1 e non $p-q \sim \log n$. Se fesse come
nel secondo caso $p^2 \sim q^2 \sim n$ e quindi il crittanalista cerca i
fattori primi da partire da \sqrt{n} con un numero $\log n$ di tentativi (polinomiale)
↓ Div. . p e q simili

$$\frac{p+q}{2} > \sqrt{n} \quad \left(\frac{p+q}{2}\right)^2 = \left(\frac{p-q}{2}\right)^2 + n \quad \text{da cui } \left(\frac{p+q}{2}\right)^2 - n = \left(\frac{p-q}{2}\right)^2 \quad \text{da} \\ \text{cui vale la relazione 1.}$$

Inoltre no che $\left(\frac{p+q}{2}\right)^2 - n$ è un quadrato perfetto \rightarrow l'attacco si fa

prendendo il primo intorno \sqrt{n} e vedo se togliendo n
 \exists un quadrato perfetto, chiamiamolo $\zeta^2 - n = w^2$. A questo
punto $\zeta = \frac{p+q}{2}$, $w = \frac{p-q}{2}$ da cui $p = \zeta + w$, $q = \zeta - w$

Inoltre, il piccolo è un problema perché facilita la forza bruta sul crittogramma.
Anche il piccolo è un problema perché può accadere che $m^e \equiv n$ e
quindi è facile trovare \sqrt{c} perché $c = m^e$ in quanto non interviene la
riduzione modulare.

Per quanto riguarda la metà di e , $e \neq \frac{\phi(n)+1}{2}$ e $e \neq \frac{\phi(n)-1}{2}$

se k divide $p-1$ e $q-1$ offriunti $m^e \pmod{n=m}$ (quando $\text{MCD}(m, n)=1$)
un problema davante della scelta di e piccolo è la probabilità più
grande che gli utenti scelgano gli stessi valori di e
↓ Attacco

Hip: Almeno e utenti hanno scelto lo stesso valore di e
Gli utenti scelgono lo stesso messaggio c

$$U_1: c_1 = m^e \pmod{n_1}$$

$$U_2: c_2 = m^e \pmod{n_2}$$

⋮

$$U_e: c_e = m^e \pmod{n_e}$$

$$H_p: \forall i, j \quad 1 \leq i < j \leq e, \quad \text{MCD}(n_i, n_j)=1$$

$$H_i \quad 1 \leq i \leq e \quad m \in n_i$$

Sia $n = \prod_{i=1}^e n_i$. Per il teorema circa del resto, $\exists! m \in n$ tale che
 $m^e \equiv c_i \pmod{n}$ e si può calcolare in tempo polinomiale.

$$m' \bmod n = m^e \bmod n$$

↓
diventa

$$m' = m^e \text{ da cui } m = \sqrt[m]{m'}: \text{ se una lo stesso e si fa un po' di padding per camuffare i messaggi}$$

$$\text{ma } m' \bmod n = m^e \bmod n, \text{ essendo che } \forall i, m < n, m^e = m \cdot m \dots m < n \dots n < n$$

$$\text{e quindi } m^e \bmod n = m^e$$

Attacco Common Modulus

Due utenti U_1, U_2 con chiavi pubbliche $\langle e_1, n \rangle$ ed $\langle e_2, n \rangle$ ed $\text{MCD}(e_1, e_2) = 1 \rightarrow \exists r, s$ tali che $r e_1 + s e_2 = 1$ e si calcolano in tempo polinomiale.

Dato che $\text{MCD}(e_1, e_2) = 1$, $\exists r, s$ tali che $r e_1 + s e_2 = 1$ e si calcolano in tempo polinomiale. Sembra possibile generare m considerando $r e_1 + s e_2 = 1$ e n .

$$\begin{aligned} m &= m^1 = m^{r e_1 + s e_2} = m^{r e_1} \bmod n = (m^{r e_1} \bmod n)(m^{s e_2} \bmod n) \bmod n \\ &= (m^{e_1} \bmod n)^r (m^{e_2} \bmod n)^s \bmod n = \text{Cer. } C_1^r C_2^s \bmod n \end{aligned}$$

✓ $\rightarrow C_1^r, C_2^s$ si calcola con quad. successivo in t. polinomiale

Se $\text{MCD}(C_1, n) = 1$ allora:
 1. deve calcolare $C_1^{-1} \bmod n$ in t. polinomiale con Euclidea.
 2. calcola $(C_1^{-1})^r$ con quad. successivo in t. polinomiale
 3. Ottenere m come $m = (C_1^{-1})^r (C_2^s) \bmod n$

Se $\text{MCD}(C_1, n) \neq 1$ allora dato che n è semiprimo allora ha fattorizzato n
 e quindi cerca meglio

Attacchi a tempo: informazioni sul chiave privata date dal tempo impiegato per la decifrazione. Questa via le quadrature successive che richiede una moltiplicazione in più per ogni bit ad 1 in d \rightarrow Si aggiunge costante.

Esercizio 11

Si supponga che Eve intercetti un crittogramma $c = m^e \bmod n$ diretto ad Alice. Si supponga inoltre che Alice sia disposta a decifrare per Eve qualsiasi crittogramma c' , a patto che c' sia diverso da c .

Descrivere come Eve possa decifrare m in tempo polinomiale, richiedendo ad Alice la decifrazione del crittogramma $c' = c x^e$, dove $x < n$ è un intero casuale, co-primo con n .

$$\begin{aligned} m^e &= c^d \bmod n = (Cx^e)^d \bmod n = C^d x^{ed} \bmod n = (C^d \bmod n)(x^{ed} \bmod n) \bmod n \\ &= m x^{ed \bmod \phi(n)} \bmod n = m x^{\underbrace{(x^{\phi(n)})^k}_{\text{per ridursi}}} \bmod n = mx \bmod n \end{aligned}$$

Per trovare m Eve calcola $m = x^{-1} \bmod n \rightarrow x^{-1} m^e \bmod n = M x x^e \bmod n = m \bmod n$

CIFRATURA IBRIDA

Alice

m

genera sua chiave $K[\text{55 bit}]$

$\langle \text{Ces}_K(k, K_{\text{pub}}, \text{Ces}_K(m, k)) \rangle$

Cifra K con RSA cifra m con AES

Bob

decifra il 1° c con la chiave privata (K)
 decifra il 2° c con la chiave K (m)

⚠ Devi avere sicurezza che la chiave pubblica sia quella dell'utente giusto
 altrimenti attacco attivo man in the middle: se viene sostituita da Eve, Alice cifra K con K [pub] e la manda indietro. Eve rimuove la coppia

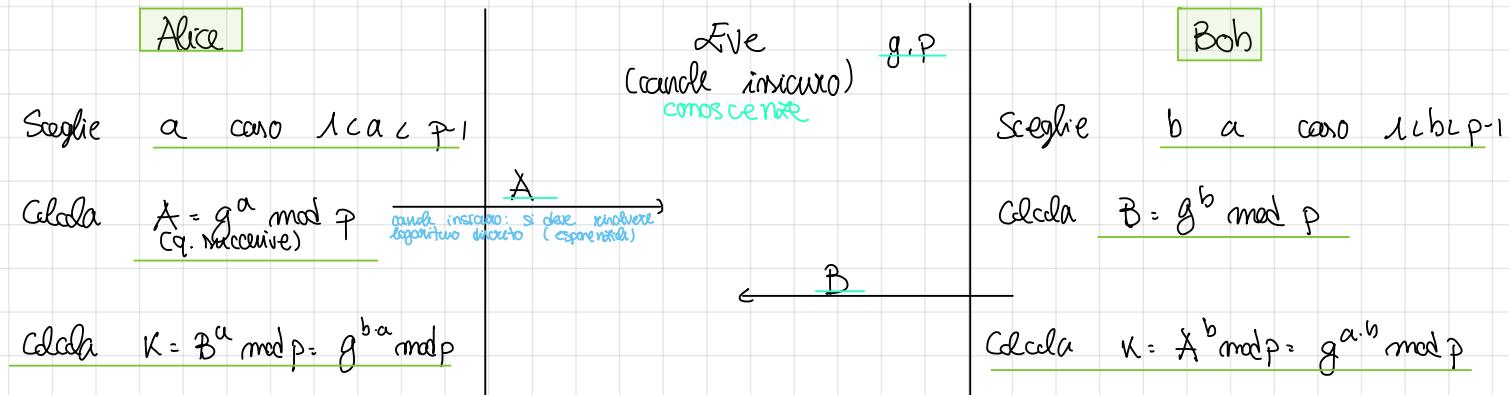
di ottagrammi del canale e prende il 1° c lo decifra. Allo stesso punto
 cifra K con $K_{\text{pub}}^{(p)}$ e gliela invia per non intercettare la comunicazione.

- L'attacco a questo punto deve cominciare K e può decifrare i ottagrammi

Le chiavi pubbliche devono essere prese dai certificati digitali

• PROTOCOLLO Diffie-Hellman (DH)

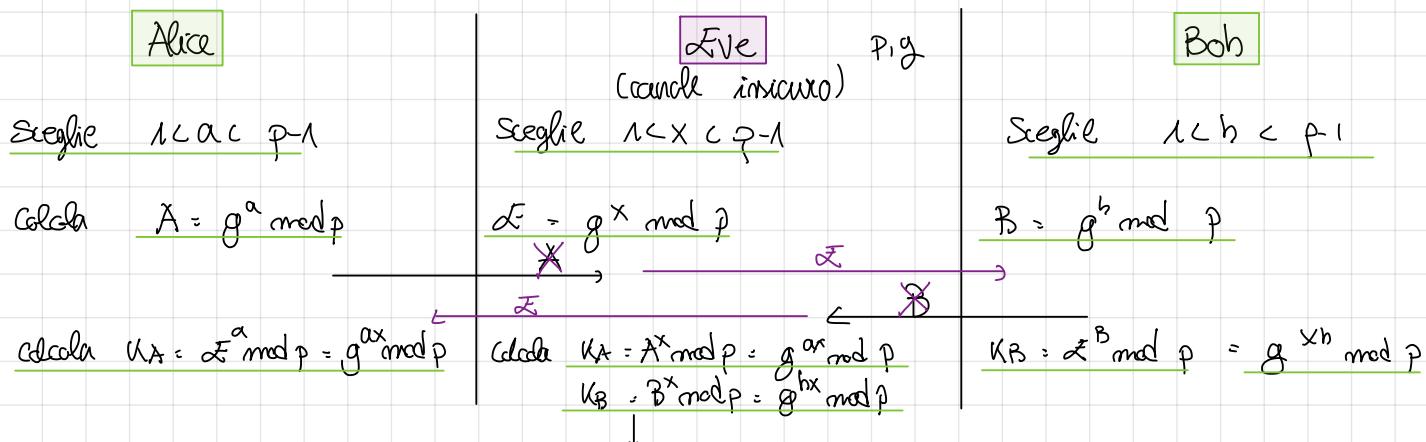
C'è una fase preliminare ove Alice e Bob scelgono un primo p molto grande (migliaia di bit) ed un generatore g per \mathbb{Z}_p^* . p e g sono pubblici e g < p. Con g e \mathbb{Z}_p^*



Attacchi

passivi: Eve conosce g, p, A, B . Per calcolare K deve trovare a o b che sono dati da $a = \log_g A$ e $b = \log_g B$ che sono problemi difficili (log. discreto) → Non è vulnerabile

attivi: È vulnerabile agli attacchi man in the middle



L'attacco diventa pericoloso se Eve non fa corrispondere Alice e Bob che è nel messaggio con K_B .

• CRIPTAZIONE di ElGamal (logaritmo discreto)

- Si sceglie p primo e g generatore di \mathbb{Z}_p^*
- Si sceglie a caso un intero $x \in [2, p-2]$
- Si calcola $y = g^x \text{ mod } p$

$$K_{\text{pub}} = (p, g, y) \quad e \quad K_{\text{priv}} = (x)$$

Cifratura: m è interpretato come intorno e deve avere $m < p$ differenti cifrature a blocchi. Il mittente sceglie x intorno $\in [2, p-2]$ e calcola la coppia di ottagrammi $C = g^x \text{ mod } p$ e $d = m \cdot y^x \text{ mod } p$. C viene usata per rimuovere la "manichera" canale

Decifratura: $m = d \cdot C^{-x} \text{ mod } p$ dove x è ripetutamente calcolato. (C^{-x}) : E perché mod. nr. primo

$$d \cdot c^{-x} \bmod p = m \cdot y^r \cdot c^x \bmod p = m \cdot y^r \cdot (g^r)^{-x} \bmod p = m \cdot (g^x)^r \cdot g^{-rx} \bmod p = m \bmod p$$

attacchi

↓ Passivi

Eve conosce p, y, g e i cattogrammi c e d ove $c = g^r \bmod p$ e $d = g^r m \bmod p$. Può trarre x ricavando il logaritmo discreto ma questo è un problema difficile oppure se Alice non protegge x , Eve decifra il cattogramma facendo $m = d \cdot y^{-r}$ dato che l'inverso moltiplicativo di y esiste ed è unico.

↓ Che succede se Alice utilizza lo stesso r ?

m_1 ed m_2 inviati a Bob con lo stesso r : $c_1 = g^r \bmod p, d_1 = y^r \cdot m_1 \bmod p$
 $c_2 = g^r \bmod p, d_2 = y^r \cdot m_2 \bmod p$.

Eve viene a conoscenza di m_1 : conosce $p, g, y, c, d_1, d_2, m_1$. Può fare
 $g^r = d_1 m_1^{-1}$ e poi $m_2 = d_2 (g^r)^{-1} \rightarrow !r$ è una e getta

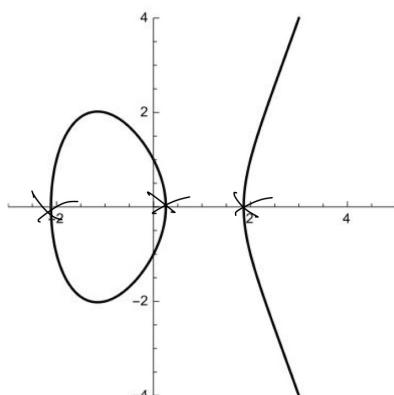
Crittografia su curve ellittiche

Anche per rimettere carico computazionale della crittografia ce chiave pubblica con chiave modulare ed in più fattorizzazione e logaritmo discreto richiedono chiavi sempre più lunghe a causa dei nuovi algoritmi di riduzione. Nel 1990 è stato sviluppato un nuovo sistema di crittografia a chiave pubblica che usa gli stessi algoritmi in un contesto matematico diverso (curve ellittiche). Il vantaggio deriva dal costo degli attacchi che è esponenziale e non sub-esponenziale in quanto difficile solo per una costante e quindi è possibile usare chiavi più corte.

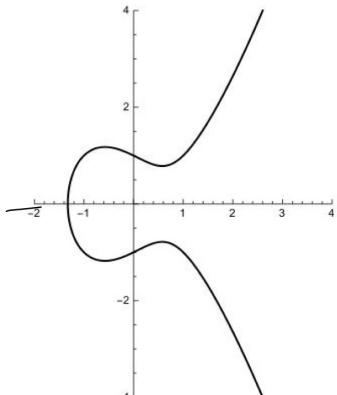
• CURVE ELLITTICHE

Curve algebriche definite da equazioni cubiche: $\mathcal{E}(a,b) = \{(x,y) \in \mathbb{R}^2 | y^2 = x^3 + ax + b\}$ (e.g. genere: campo K , $\mathcal{E}(a,b,c,d,e) = \{(x,y) \in K^2 | y^2 + cxy + by = x^3 + cx^2 + dx + e\}$ ove $a, b, c, d, e \in K$). Se $\text{char } K \neq 2, 3$ si può usare la forma normale di Weierstrass (quella corta). Si aggiunge sempre $\{O\}$ che viene detto punto all'infinito per avere un elemento neutro additivo. Questo non importa nelle curve ellittiche sui reali.

↓ Esempi:



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



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

Cubica con tre radici reali

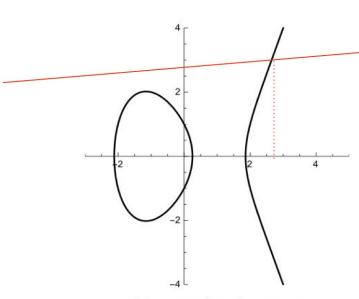
Cubica con una radice reale e due complesse coniugate

In ogni punto è possibile tracciare una tangente: si escludono quelle curve che hanno cuspidi e nodi → $4a^3 + 27b^2 \neq 0$: no radici multiple e no cuspidi e nodi. Le curve ammesso hanno simmetria rispetto a x: il punto possiamo definire l'opposto di $P(x,y)$ come $-P(x,-y)$. Anche il punto O ha l'opposto definito come O stesso.

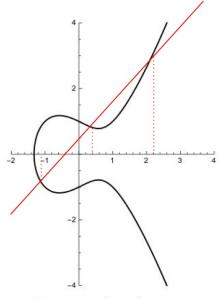
Somma

Ogni retta interseca una curva ellittica al più in 3 punti. Questo perché se mettiamo in sistema le equazioni della retta e della curva e si ottiene una cubica che può avere al massimo 3 soluzioni reali.

⚠ ATT: solo 2 casi (1 reale, 2 cc ✓ 3 IR). se ho 2 soluzioni IR ne ricaveranno una terza. Ma questo per costruire la somma.



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



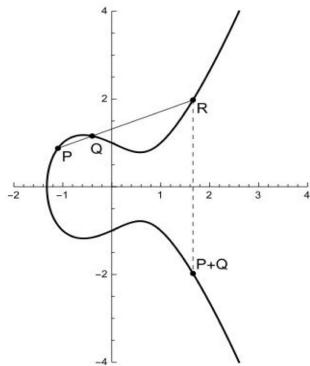
(b) Curva $y^2 = x^3 - x + 1$

una soluzione reale e due soluzioni complesse e coniugate, quindi un punto (reale) di intersezione

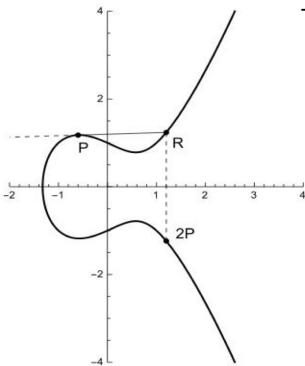
tre soluzioni reali, quindi tre punti di intersezione

hanno 3 punti $P, Q, R \in E(a, b)$, se $P + Q + R = O$.

→ Ricaviamo la regola della somma



(a) Somma di due punti P e Q .



(b) Raddoppio di un punto P

→ Faccio la retta $P+Q$ e poi, dato che sono sicuro che esiste, prendo R come punto somma. Se voglio sommare un punto con se stesso mondo inverso la tangente e prendo sempre $-R$

Siano $P = (x_P, y_P)$ e $Q = (x_Q, y_Q)$ due punti della curva $E(a, b)$

1. $P \neq \pm Q$

$$S = P + Q$$

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

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

$\lambda = \frac{(y_Q - y_P)}{(x_Q - x_P)}$ ~ coefficiente angolare della retta per P e Q

2. $P = Q$

$$S = P + Q = 2P$$

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

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

$\lambda = \frac{(3x_P^2 + a)}{2y_P}$ ~ coefficiente angolare della tangente alla curva in P ($y_P \neq 0$)

Se $y_P = 0$, $S = 2P = O$

3. $Q = -P$

$$S = P + Q = P + (-P) = O$$

Se sceglioamo curve che hanno $4a^3 + 27b^2 \neq 0$ la legge dell'addizione definisce un gruppo chiamato additivo

↓ Regole

Chiusura: $\forall P, Q \in E(a, b), P + Q \in E(a, b)$;

Elemento neutro: $\forall P \in E(a, b), P + O = O + P = P$ (infatti le rette passanti per O sono verticali, dunque la retta per P e O interseca la curva in $-P$, il cui simmetrico è P);

Inverso: $\forall P \in E(a, b)$, esiste un unico $Q = -P \in E(a, b)$ tale che $P + Q = Q + P = O$;

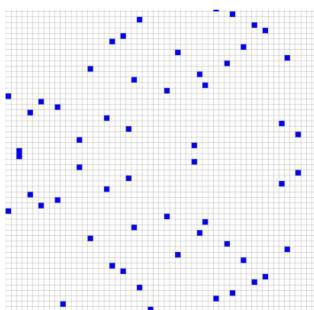
Associatività: $\forall P, Q, R \in E(a, b), P + (Q + R) = (P + Q) + R$;

Commutatività: $\forall P, Q \in E(a, b), P + Q = Q + P$.

Curve ellittiche su campi finiti

2 tipi di campi finiti: # numero primo, # pot. numero primo.

curve ellittiche prime: verificabili e coefficienti $\in \mathbb{Z}_p$, p primo, $a, b \in \mathbb{Z}_p, p > 3$ (per forma Weierstrass)



→ Numero finito di punti simmetrici rispetto a $y = p/2$ perché lavorano sempre tra $0 \leq y \leq p-1$. Possiamo definire l'opposto di $P(x, y)$ come $-P = (x, -y) = x(1, p-y)$
 $-y \bmod p = p-y \bmod p$

La curva si ottiene con le formule di prima
c'è qualcosa al denominatore si fa la moltiplicazione
per avere il denominatore si deve sempre
scrivere curve ellittiche hincorbie: $K = GF(2^m)$, chiedi $k=2 \rightarrow$ l'equazione non si può ridurre

applicando il modulo. se per l'inverso moltiplicativo
potere $4a^3 + 27b^2 \bmod p \neq 0$
l'equazione non si può ridurre
alla forma normale di Weierstrass e formule somma diverse.

Esempio: $\mathbb{F}_5(4, 4)$

$$4a^3 + 27b^2 \bmod 5 \neq 0 \rightarrow \frac{4 \cdot 4^3}{27} + 27 \cdot 4^2 \bmod 5 = 3 \bmod 5 \neq 0 \quad \checkmark$$

$$\text{eq: } y^2 = x^3 + 4x + 4 \bmod 5$$

troviamo i residui quadratici di \mathbb{F}_5

4	0	1	2	3	4
y^2	0	1 (4)	4	1	

x	y^2	y	
0	4	2, -2	$\rightarrow (0, 2) \quad (0, -2)$
1	4	1, -1	$\rightarrow (1, 1) \quad (1, -1)$
2	0	0	$\rightarrow (2, 0)$
3	3		Non ci sono pt. della curva di curva 3
4	4	2, -2	$\rightarrow (4, 2) \quad (4, -2)$

↓
Ordine = 8 ($7pt + O(\infty)$)
 $(\#pt)$

Teorema di Hasse: $N = \text{ordine curva prima g.P p}(a,b) \rightarrow |N - (p+1)| \leq 2\sqrt{p}$ (ordine)

funzione one-way trap-door

Moltiplicazione di punti curve ellittiche \sim molti pli costare interi modulo p

Moltiplicazione radice di un punto $P \in E_p(a,b)$ per intero $k \sim$ elevamento a potenza reale

↓ definizione

$KP = \underbrace{P+P+\dots+P}_{k \text{ volte}}$ - si calcola in tempo polinomiale, con $\Theta(\log k)$ addizioni

↓ Algoritmo dei raddoppi ripetuti

$$13P = (8+4+1)P = 8P + 4P + P =$$

= 3 raddoppi e 2 somme

↓ In generale

L log₂ k ↓ raddoppi e $O(\log k)$ somme

P

↓ 1 raddoppio

~~non ci serve~~

↓ 1 raddoppio

$2(2P) = 4P$

↓ 1 raddoppio

$2(2P) = 8P$

Generale: $P \in E_p(a,b)$ e $K = \sum_{i=0}^t K_i \cdot 2^i$ ove $K_i \in \{0, 1\}$

$$K_0 = \underbrace{(K_t K_{t-1} \dots K_1 K_0)_2}_{\text{cod. binaria di } K}$$

$$t = \lfloor \log_2 K \rfloor$$

1. Si calcolano $2, 2^2P, 2^3P, \dots, 2^tP$ con t raddoppi ciascuno come raddoppio precedente

2. Si ottiene $Q = KP$ come $\sum_{i=0}^{t-1} (2^i P)$ con $\leq t$ somme

Algoritmo che esegue $\Theta(t) = \Theta(\log_2 K)$ somme (raddoppi)

↓ Inversione funzione

dati Q e P , trovare, se esiste, il più piccolo intero $K / Q = KP$ è un problema difficile → si dice $K = \log_p Q$: logaritmo discreto su curve ellittiche. Questo si risolve con metodi brutte force ($P, 2P, 3P, \dots, KP$ quindi esponenziali).

• PROTOCOLLI CRIPTOGRAFICI

protocollo Diffie-Hellman su curve ellittiche

Alice e Bob scelgono una curva ellittica prima $E_p(a,b)$ ed un punto $B \in E_p(a,b)$ di ordine elevato ove l'ordine di un punto è il più piccolo intero n tale che $nB = 0$. Questo accordo è pubblico.

Alice

Sceglie a con in \mathbb{Z}_n

$a \in \mathbb{Z}_n$

Calcola $P_A = aB$

Eve

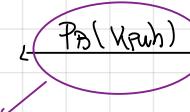
$P_A (K_{AB})$

Bob

Segliere un punto della curva

$n_B \geq n$

Calcola $P_B = n_B B$



Calcola $K = n_A P_A (= n_A n_B B)$

Calcola $K = n_A P_A (= n_A n_B B)$



Estrazione n_A, n_B richiede il deposito dei corrispondenti chiavi ellittiche che è esponentiale

K è un punto della curva (coppia di interi): $K = (x_K, y_K)$. Facendo $x_K \bmod 2^{256}$ si ottiene una chiave per AES (256)

attacchi

pari: deve conoscere $\mathbb{Z}_p(a,b)$, B , P_A e P_B . Per calcolare K bisogna calcolare $n_A = \log_p P_A$ e $n_B = \log_p P_B$ usando il bruto force che è esponenziale: richiesto n_A cal attacchi pari

attivi: se deve sostituire P_A con P_E può ricavare informazioni: le chiavi devono essere estratte da certificati digitali

protocollo di ElGamal su curve ellittiche (cifratura a chiave pubblica)

Il protocollo richiede di "incapsulare" il messaggio su un punto della curva ellittica: $m \rightarrow P_m \in \mathbb{Z}_p(a,b)$

↳ idea

$q^2 = x^3 + ax + b$. Vorrei forse $x = m$ ma la cubica $x^3 + ax + b$ corrisponde ad un residuo quadratico con probabilità $\approx \frac{1}{2}$ che non va bene

Algoritmo di Kohlitz:

for loop

si sceglie un intero $h / (m+1)h < p$ e si sostituisce $X = mh + i$ con $0 \leq h < p$. Fatto fare h tentativi e la prob. di fallimento è $\sim (\frac{1}{2})^h$. La prob. di successo è

quindi $1 - (\frac{1}{2})^h$ che è ~ 1 se h è grande.

→ p fallimento: $(\frac{1}{2})^h$ (per la volte che non è residuo quadratico)

→ succede: $1 - (\frac{1}{2})^h$ (~ 1 se grandi)

```

KOHLITZ (m, h, a, b, p)
for (i=0; i<h; i++) {
    X = mh+i;
    Z = (X^3 + ax + b) mod p;
    if (Z residuo quadratico) {
        y = sqrt(Z) mod p; → polinomiale: mod numero primo
        return P_m = (X, y)
    }
}

```

else: algoritmo ha fallito
return failure

↓ Estrazione messaggio

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

Cifratura: $\mathbb{Z}_p(a,b)$ è B pt. base di ordine n elevato: $B \in \mathbb{Z}_p(a,b)$. Ogni utente costituisce la sua coppia (K_{pub}, K_{priv}) .

Utente U : $K_{priv} = n_U \in \mathbb{Z}_n$ intero scelto a caso

$$K_{pub} = n_U B = P_U$$

Alice

$m \rightarrow P_{\text{in}} \text{ con } K_{\text{Koblitz}}$

Sceglie $r \in \mathbb{Z}$ casuale del t. B

Calcola $V = rB$ (raddoppi)

Ripetuti tempo polinomiale

Calcola $W = P_{\text{in}} + rP_{\text{Bob}}$ caso dest
v, w \in \mathbb{Z}_p[t, b]

Invia la coppia $\langle V, W \rangle$ a Bob

Eve

Bob

Riceve $\langle V, W \rangle$ Decifra calcolando $W - n_{\text{Bob}} \cdot V \cdot P_{\text{public}}$ caso BobRicerca $m = \lfloor \frac{x}{q} \rfloor$ Dimostrazione correttezza: $W - n_{\text{Bob}} \cdot V = P_{\text{in}} + rP_{\text{Bob}} - n_{\text{Bob}}V = P_{\text{in}} + r(n_{\text{Bob}}B) - n_{\text{Bob}}(rB) = P_{\text{in}}$

Attacli puini: 1. Trovare n_{Bob} da $P_{\text{Bob}} = n_{\text{Bob}}B \rightarrow$ log discreto su C.E.

2. Compondo x, m puoi decifrare: $W = P_{\text{in}} + rP_{\text{Bob}} \rightarrow P_{\text{in}} = W - rP_{\text{Bob}}$

Sicurezza curva ellittica: Il logaritmo discreto in algebra modulare ha costo $O(2^{b \log b})$ ove $b = \text{bit del modulo}$. Lavoro modulo \neq ove \neq è un numero di q bit. L'algoritmo in algebra modulare sfrutta il fatto che i numeri costituiscono un campo. Questo non è vero per le curve ellittiche (è moltiplicazione, è index calculus). Il logaritmo discreto in C.E. può essere fatto con l'algoritmo Pollard \neq che ha un costo $O(2^{b^2})$ ove $b = \text{bit ordine di } B \rightarrow$ possiede chiavi più corte.

Esercizi: Esercizio 4Determinare i punti appartenenti alla curva ellittica $E_{11}(1, 6)$.

$$y^2 = x^3 + x + 6 \pmod{11}$$

zerni quadratci \mathbb{Z}_M :

q	0	1	2	3	4	5	6	7	8	9	10
y^2	0	1	4	9	5	3	3	5	9	4	1

 $\xrightarrow{\text{ZG}} 0, 1, 3, 4, 5, 9$

$$x = 0 \quad y^2 = 6 \rightarrow \text{NO PQ, Nem un pt.} \quad x = 0$$

$$x = 1 \quad y^2 = 3 \rightarrow \text{"}$$

$$x = 2 \quad y^2 = 5 \rightarrow y = 4, 7 \rightarrow (2, 4), (2, 7) \in \mathbb{Z}_M(1, 6)$$

$$x = 3 \quad y^2 = 3 \rightarrow y = 5, 6 \rightarrow (3, 5), (3, 6) \in \mathbb{Z}_M(1, 6)$$

$$x = 4 \quad y^2 = 8 \rightarrow \text{NO PQ}$$

:

$$x = 10 \quad y^2 = 4 \rightarrow y = 2, 9 \rightarrow (10, 2), (10, 9) \in \mathbb{Z}_M(1, 6)$$

$$\downarrow \\ \text{Ordine} = 13$$

Protocolli

identificazione

Sistema di elaborazione, locale o in rete, che permette l'accertamento dell'identità dell'intervento che richiede l'accesso a un servizio.

autenticazione

Il destinatario deve accertarsi dell'identità del mittente e dell'integrità del messaggio.

firmature digitali

a. MITT non può negare di aver inviato un

b. DEST deve poter autenticare il messaggio

c. DEST non può sostenere che MITT ha inviato un antico messaggio

} Verificabile da terze parti

Permettono di contrastare attacchi attivi, utilizzano cifrari simmetrici e assimmetrici.

funzioni hash

Op. $X \rightarrow Y$ è una funzione / $n = |X| \gg m = |Y|$, $\exists X_1 \dots X_m \subseteq X$ disgiunti / $X = X_1 \cup X_2 \cup \dots \cup X_m$
 $\forall i, \forall x \in X_i, f(x) = y$. Una buona funzione hash si ha quando gli x_i hanno circa la stessa cardinalità. Elevanti uanto sia più sicuri appartenino a sottimi errori diversi. Ci vogliono algoritmi per gestire le collisioni

Criptografiche: one-way

1. $\forall x \in X$ è computazionalmente facile calcolare $y = f(x)$

2. per la maggior parte degli $y \in Y$ è difficile determinare $x \in X / f(x) = y$

3. Proprietà claw-free: è difficile determinare $x_1, x_2 \in X / f(x_1) = f(x_2)$

↓ esempi funzioni:

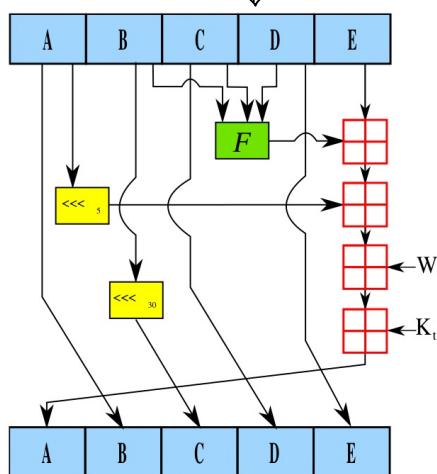
MD5 (Message digest): il valore hash ha lunghezza fisica. Non resistente a collisioni. Prende un input S di 512 bit e produce un'immagine di 128 bit

RIPEND-160: Versione ridotta MD → lunghezza di 160 bit, resistente a collisioni.

SMA (Secure Hash Algorithm): 6 versioni (0, 1, 2, 3) con diverse dimensioni immagini

↓ SHA-1

lunghetti di 160 bit. Il messaggio deve avere lunghezza multipla di 512 bit



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; $\ll_{n,l}$ 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

Candidi segnali

Esempio: Acceso di U alla sua casella di posta / file su calcolatore con account riservato a membro organizzatore

↓

1. Invio login e password

2. Se il candido è protetto il candido può essere bloccato da un utente
la cle → le password non devono mai essere scritte in chiaro

↓

3. La password viene crittata con una funzione hash

↓

U fornisce per la 1^a volta la sua pw. Il sistema associa ad U due sequenze binarie che salva nel file delle pw: un numero S prodotto da un generatore pseudocasuale e Q = h(PW) ore in cui user. Usiamo S per avere immagini hash ≠ per pw =. Per le connessioni successive il sistema recupera S dal file delle pw, calcola l'immagine one-way h(PW) e se h(PW) = Q allora l'identificazione ha successo. → Un account ha file delle pw non fornire informazioni (computazionalmente difficile)

Candidi incisurati

Ch. n. 1 = chiavi pubbliche e private di U che vuole accedere ai suoi dati offerti da S

↓ Passaggi

1. S genera numero casuale r/n e lo invia ad U in chiaro

2. U calcola f = r^e mod n (decifrazione, firma di U su r, solo U conosce d) e lo invia ad S

3. S verifica la correttezza verificando f^e mod n = r (cifratura). Se sì, l'identificazione ha successo.

↓

Va bene perché solo U conosce d e quindi solo U può generare f. Il protocollo funziona solo se il sistema è gestito: il sistema può generare un r per scoprire la chiave privata C più facilmente

• AUTENTICAZIONE

DEST deve identificare MIT ed ammesserne l'integrità di m

↓

MIT e DEST concordano su chiave segreta K che viene usata da MIT per calcolare il MAC (Message Authentication Code) A(m, K) per garantire la protezione e l'integrità. Solo chi conosce K può inviare. Spedisce poi la coppia < m, A(m, K) > in chiaro oppure cifra m e < C(m, K), A(m, K) > ove C è la funzione di cifratura pubblica o segreta del cifrario scelto. DEST calcola il MAC e lo confronta con quello ricevuto e lo accetta se sono uguali. Il MAC può avere qualsiasi funzione hash one way. Il crittoanalista infatti per sostituire il messaggio con il suo doppio collegare un MAC comune e poi fare ciò doppio cancellare K (se h è resistente alle collisioni). L'h è nota e deve essere difficile invertirla altrimenti si potrebbe recuperare K. Un modo alternativo è quello di utilizzare CBC utilizzando l'ultimo blocco come MAC

FIRMA DIGITALE

Forma "clamorosa":

risparmio a
digitale

- a. Autentica e non falsificabile
 - b. Non riutilizzabile
 - c. Il documento firmato non è alterabile
 - d. Non può essere ripudiata da chi l'ha apposta
- Non può avere la digitale scrittura della firma grafica (è facile fare cattura and parte), deve avere incindibile del documento firmato e può avere realizzata con cifrari simmetrici o asimmetrici → funzione hashometrica del contenuto

Protocollo di Diffie e Hellman

U: ente, V: $K_U[\text{priv}]$ e $K_U[\text{pub}]$ chiavi di U, C e D funzioni di cifratura e decifratura di un cifrario asimmetrico

U genera la firma $f = D(m, K_U[\text{priv}])$ per m e spedisce a V la tupla $\langle U, m, f \rangle$. V verifica la firma facendo $C(f, K_U[\text{pub}]) = m$ e se l'accoppiamento è valido allora la firma è verificata e impossibile proteggere il messaggio in lettura in quanto la verifica della firma è fatta con la chiave pubblica. Inoltre, V controlla tutti i requisiti della firma. Infatti è autentica e non falsificabile perché $K_U[\text{priv}]$ è nota solo ad U e per otenerla si dovrebbe invertire D che è one-way. Il documento non può essere alterato altrimenti la verifica non andrebbe a buon fine e la firma non può essere ripudiata in quanto può essere prodotta solo da U con $K_U[\text{priv}]$. Infine, la firma non può avere utilizzata perché si prende da m.

Protocollo 2: un cfrato e firmato

FIRMA e CIFRATURA: a. U genera $f = D(m, K_U[\text{priv}])$

b. calcola $C = C(f, K_V[\text{pub}])$ con la chiave pubblica del destinatario → la firma è incapsulata nel crittogramma

c. Si spedisce $\langle V, C \rangle$ a V

VERIFICA: a. V riceve $\langle V, C \rangle$ e lo decifra ponendo $f = D(C, K_V[\text{priv}])$

b. Cifra f con la chiave pubblica di U: $C(f, K_U[\text{pub}]) = C(D(m, K_U[\text{priv}]), K_U[\text{pub}]) = m$

c. V ricontrolla m e se m è significativo la firma è verificata

è molto poco probabile che la firma venga falsificata grazie all'utilizzo di $K_U[\text{pub}]$

↓ IMPLEMENTAZIONE: RSA

$\langle d_u \rangle, \langle e_v, n_v \rangle$ chiavi di U
 $\langle d_v \rangle, \langle e_v, n_v \rangle$ chiavi di V

U genera la firma di un. f: $m \xrightarrow{\text{mod } n_v}$, cifra f con la chiave pubblica di V, $C = f \xrightarrow{\text{mod } n_v}$ e spedisce a V $\langle V, C \rangle$

V riceve $\langle V, C \rangle$ e decifra C, $C \xrightarrow{\text{mod } n_v} f \xrightarrow{\text{mod } n_v} m$
decifra f con la chiave pubblica di U, f $\xrightarrow{\text{mod } n_v} m$
se m è significativo allora l'autenticità.

↙

Per cifrare la firma deve accadere $f \neq n_v$ per cifrare la firma e quindi $n_v \neq n_u$ (relazioni sopra). Questo però impedisce a V di inviare numerose firme e cifrati a U → d'entente tra 2 copie di chiavi RSA: una coppia viene usata per la firma ed una per la cifratura. Si neglige il grande e si sommano le chiavi di firma + M e quelle di cifratura → M da scelta di M elevato costituisce gli intenti a scegliere chiavi sufficientemente grandi.

↓ Attacchi derivanti da "gradi" algheci
 Il catturando si provoca la firma di un utente su messaggi apparentemente privi di senso. Supponendo che V invii una risposta automatica a U ogni volta che riceve un messaggio m. L'ack è il critogramma della firma di U su m. Eve intercetta c e lo ritrae al canale e lo riduce a V spacciandosi per il mittente. V produce un ack ad Eve. Eve a questo punto può rintracciare il messaggio su mondo le chiavi pubbliche (tempo polinomiale)

1. U invia c a V: $c = C(f, k_U [pub]) \rightarrow \langle U, c \rangle$
 $f = D(m, k_U [priv])$
 2. Eve intercetta c, lo rimuove e lo sostituisce con $\langle Eve, c \rangle$
 3. V decifra c, $f = D(c, k_V [priv])$ e verifica la firma con la chiave pubblica di Eve ottenendo $m' \neq m = C(f, k_{Eve} [pub])$
 4. $m' \neq m$ è prova di senso ma V manda l'ack c' a X
 ↓ ack
 $f' = D(m', k_{Eve} [priv])$
 $c' = C(f', k_{Eve} [pub])$
 5. Eve manda c' per richiedere ad m
 - a. decifra c' e trova f': $D(c', k_{Eve} [priv]) = D(C(f', k_{Eve} [priv]), k_{Eve} [priv]) = f'$
 - b. Verifica f' e trova m': $C(f', k_V [pub]) = C(D(m', k_V [priv]), k_V [priv]) = m'$
 - c. Su m' ricava f: $D(m', k_{Eve} [priv]) = D(C(f, k_{Eve} [pub]), k_{Eve} [priv]) = f$
- ↓ evitare l'attacco

L'ack viene mandato solo dopo la verifica di m significativo → Nelle applicazioni reale non si firma il messaggio ma l'immagine di una funzione hash one-way.

FIRMA e CRIPTATURA: a. U calcola h(u) e genera la firma $f = D(h(u), k_U [priv])$
 b. Calcola $c = C(m, k_U [pub])$
 c. Spedisce a V $\langle U, c, f \rangle$

VERIFICA: a. U riceve $\langle U, c, f \rangle$
 b. Decifra c: $m = D(c, k_V [priv])$
 c. Calcola h(u) e $C(f, k_U [pub])$. Se questi valori coincidono la firma viene accettata

h(u) si calcola velocemente ed il suo utilizzo produce firme certe.
 Il cifrario è vulnerabile agli attacchi man-in-the-middle alterando lo scambio della chiave pubblica.

↓ Scavetta rispetto a questi attacchi

Certification Authority: enti che garantiscono sicurezza rispetto a questi attacchi garantendo la scavetta delle chiavi pubbliche
 ↓ Come?

- a. La CA certifica l'associazione $\langle \text{utente}, \text{chiave pubblica} \rangle$ con un certificato digitale
- b. Il certificato contiene la chiave pubblica e le informazioni dell'utente ed è firmato dalla CA
- c. La CA ha un archivio con tutte le chiavi pubbliche accessibili a tutti in crittografia ma protetto da attacchi in rotellatura
- d. La chiave pubblica della CA è nota agli utenti che la proteggono da attacchi esterni e che verificano la firma della CA sui certificati. → Le chiavi pubbliche delle CA hanno le proprie chiavi nei browser.

- 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.

Se U vuole comunicare con V può richiedere K_U [pub] alla CX che gli invia il certificato digitale di V. L'autenticità di questo viene verificata da U poiché conosce K_C [pub]. Se i controlli vanno a buon fine prende K_U [pub] ed invia la comunicazione → Attacchi siano in the middle difficili perché possibili solo falsificando le certificazioni. tuttavia bisogna CA organizzate ad interno e quindi sono richieste più verifiche per i certificati fino ad arrivare alla CA comune da U a V. Ogni ente deve mantenere localmente una copia di certi e una copia di K_C [pub] per avere un solo accesso alla CA.

Protocollo 4: m cifrato, firmato in hash e certificato

FIRMA, CIFRATURA e CERTIFICAZIONE:

- U si procuro Cert_U di V e verifica autenticità Cert_U
- Calcola $H(m)$ e genera la firma $f = D(H(m), K_U)$
- Calcola $c = C(m, K_U)$
- Spedisce a V la tripla $\langle \text{Cert}_U, c, f \rangle$

contiene K_U [pub] e si utilizza

VERIFICA DEL CERTIFICATO: a. Riceve $\langle \text{Cert}_U, c, f \rangle$ e verifica autenticità Cert_U

DECIFRA FIRMA e VER.FIRMA: a. V decifra c e ottiene $m = D(c, K_V)$
 b. V verifica autenticità firma apposta da U su m controllando $C(f, K_U)$. Inoltre

punto debole: certificati scaduti, attacchi siano in the middle rendono comunque un incontro fisico con CA

Esercizio 8

Si presenta un primo tentativo di firma elettronica basato su curve ellittiche. Si ha una curva ellittica globale, un numero primo p , e un "generatore" B . Alice sceglie una chiave di firma privata x_A e crea la chiave pubblica di verifica $Y_A = x_A B$. Per firmare un messaggio M :

- Alice sceglie un valore k
- Alice invia a Bob M, k e la firma $F = M - k x_A B$
- Bob verifica che $M = F + k Y_A$

1. Dimostrare che questo schema funziona correttamente. Ovvero che il processo di verifica produce un'uguaglianza quando la firma è valida.

2. Dimostrare che lo schema è inaccettabile descrivendo una semplice tecnica per creare la firma falsa di un utente su un qualsiasi messaggio.

$$1. F = M - k x_A B \rightarrow F + k Y_A = M - k x_A B + k x_A B$$

2. È facile falsificare la firma poiché Alice sa la sua chiave pubblica per cifrare il messaggio → Δ inaccettabile: dovrebbe sapere la sua chiave privata.

• PROTOCOLLO ZERO-KNOWLEDGE

Peggy P
prover

Victor V
verifier

↓ Esempio

Il prover promette di sapere forse qualcosa e deve convincere il verifier. Supponiamo che Peggy dica di sapere stabilire se i granelli di sabbia in una scatola sono pari o dispari e vuole convincere Victor senza svelargli il metodo. Victor mette i granelli nella scatola e la mostra a Peggy che risponde O no perciò, i se dispari. Victor lancia una moneta chiedendo a Peggy di girarsi estraendone un bit a caso. Se genera 0 toglie un granello, se genera 1 lancia tutto inviato. Mostra poi di nuovo la scatola a Peggy. Se V ha estratto 0 e la risposta di P è cambiata allora ha vinto la sfida. Se invece la risposta non cambia allora P ha perso la sfida. V continua a ripetere le sfide ed il protocollo si interrompe quando P conclude che P è onesto. Viceversa, se P risponde correttamente si interrompe quando V è soddisfatto $\rightarrow P(\text{imbarazzo}) \approx P = \left(\frac{1}{2}\right)^K$. P ha questa

facoltà con $P = 1 - \left(\frac{1}{2}\right)^K$

completezza: Se P è onesto, e la sua affermazione è vera, V accetta sempre la dimostrazione

correttezza: Se P è onesto, e la sua affermazione è falsa, V può avere un'inganno con probabilità $\leq \left(\frac{1}{2}\right)^K$ con K scelto da V.

La completezza è fondamentale

concentrazione zero: Se l'affermazione di P è vera, un V anche duocento non può acquisire nessuna informazione se non la veridicità dell'affermazione

Protocollo di FIAT - SHAMIR (Identificazione)

Basato sul calcolo della radice modulo un numero composto

PREPARAZIONE: P sceglie p e q primi molto grandi, calcola $n = p \cdot q$, sceglie s e n segreto (chiave privata di P), calcola $t = s^e \bmod n$. Pnde nota $\langle t, n \rangle$ e mantiene privata $\langle p, q, s \rangle$

PROTOCOLLO: Ripeti K volte // K scelto da V
1) V chiede a P di iniziare un'iterazione (sfida)

2) P sceglie r e n random

calcola $m = r^2 \bmod n$

comunica V a V

3) V genera un bit e ∈ {0, 1} random // gen. crittograficamente sicuro
e lo comunica a P

4) P calcola $z = r \cdot s^e \bmod n$ e invia z a V

// e = 0 → z = r mod n

// e = 1 → $z = r \cdot s \bmod n$
casuale sicuro!

5) V calcola $x = z^2 \bmod n$

if ($x = m \cdot t^e \bmod n$) torna al punto 1;

else STOP // P non identificato

Giustificazione: $e = 0 \rightarrow z = r \bmod n$ e $x = z^2 = r^2 = V$

$e = 1 \rightarrow z = s \bmod n$ e $x = z^2 = s^2 = t \cdot r^2 = V \rightarrow x = t \cdot V$

VERIFICHE:

COMPROVATE Z

$$e=0 \rightarrow x = u t^e \bmod n = u t \bmod n$$

$$e=1 \rightarrow x = (z^1) \bmod n = (r s^e)^2 \bmod n = u t \bmod n$$

↓

P supera tutte le sfide se conosce s

CORRETTEZZA

P dunque (non conosce s)

1° caso: P prevede di ricevere $e=0 \rightarrow$ esegue protocollo norma modi-
-ficarlo e se $e=0$ vince lo sfido

2° caso: P prevede di ricevere $e=1 \rightarrow P$ cambia pauro 2 protocollo
in 2': P invia $u = r^2 \cdot t^1 \bmod n$ (t, r noti, molto probabile
che t, r coprimi perché grandi quindi $\exists t^{-1}$). P al pauro 2
manda $z = r \bmod n$. Al pauro 5, se la previsione di
 P è corretta, P supera la sfida

↓ Infatti

$$\checkmark \text{ controlla se } x = z^2 \cdot u t^e = u t$$

$$\downarrow \text{Quindi } z^2 = (r t^1)^2 = r^2 \quad (r^2 t^1) \cdot t = r^2 \rightarrow \text{valori uguali}$$

P dunque sa che ad ingannare V se è in grado di prevedere
il bit "e" se i bit "e" sono generati casualmente, P inganna V con
probabilità $(\frac{1}{2})^n$

• PROTOCOLLO SSL

garantisce confidenzialità e affidabilità. Sviluppato da Netscape per comunicazioni offi-
-cionali HTTP concordando il meccanismo crittografico fra client e server

↓ scenario

U vuole accedere a servizio di S tramite internet: garantisce confidenzialità da comunicazione
cifrata → Cifrario simmetrico: asimmetrico per creazione e scambio chiavi di renuncio e simmetrico
per comunicazioni successive

Garantisce l'autenticazione dei messaggi attraverso un cifrario asimmetrico o usando
certificati digitali inserendo il MAC nei messaggi

SSL sta tra il protocollo di trasporto (TCP/IP) ed il livello applicativo ed è
indipendente da esso. → HTTPS = HTTP+SSL (oggi TLS).

↓ 2 livelli

SSL record e SSL handshake. Il primo è il più basso collegato al livello di trasporto
che usa i meccanismi di cifratura scelti nell'handshake per imballare ed inviare i
messaggi. L'handshake crea il completo sicuro per la comunicazione e durante questo
gli utenti ed il sistema si autentizzano, decidono gli algoritmi da usare e
creano le chiavi. → Meccanismo crittografico: comune sicuro, affidabile e autenticato.

↓ Come? Scambio di messaggi

1. U manda a S il messaggio client hello: richiede creazione comunicazione sicura tramite
SSL, specifica le prestazioni di sicurezza da utilizzare (cifrari) e invia
una sequenza di byte casuali

↓ esempio

SSL_RSA_WITH_AES_CBC_SHA1

RSA per scambio chiavi di sessione

AES per cifratura simmetrica

CBC indica l'impiego di un cifrario a composizione di blocchi

SHA1 funzione hash one-way per il MAC

2. S riceve il client hello, seleziona una cipher suite compatibile con U e invia un messaggio di server hello con con una richiesta di byte casuali. Se U non riceve risposta interrompe la comunicazione.
 3. S si autentica con U inviando il suo certificato digitale ed eventualmente la CA fino alla radice CA di cui U ha fiducia. Se anche il server vuole proteggere gli accessi richiede ad U un certificato digitale.
 4. S invia server hello done: gare accordi cipher suite e parametri crittografici.
 5. U controlla validità del certificato con data validità, affidabilità CA e autenticità firma CA.
 6. U costruisce un pre-master secret: sequenza di byte casuali che poi si fa con la chiave pubblica del server prima del certificato digitale e lo spedisce a S. Il pre-master secret viene combinato con i byte casuali del client e server hello costituendo il master secret. Per costruire il master secret U applica anche funzioni hash one-way.
 7. S decifra il pre-master secret e costruisce il master secret con la stessa procedura di U.
 8. U deve inviare ad S il certificato digitale (se richiesto). Al certificato vengono aggiunti il master secret ed i messaggi combinati fino al quel momento (SSL history). S controlla il certificato, la SSL history ed in caso di anomalie interrompe la comunicazione.
 9. messaggio finished: prima costruito da U e inviato a S e poi vice-versa. Protetto con master secret e cipher suite scelta.
- ↓ costruzione
- Concatenati master secret, SSL-history e l'identità. A questa stringa vengono applicate SHA-1 e RSA e si ottiene una coppia di valori. Lo stesso fa il server anche se il messaggio finished è ≠ perché aggiunge anche quello di U. Non è possibile invertire la funzione hash e quindi il confronto viene fatto sul 'immagine'

10. Il master secret è usato per costruire una triple contenente la chiave negata per il cifrario simmetrico, la chiave per la costruzione del MAC e la chiave per l'initializzazione del CBC → 2 triple ≠ per client < server ma note ad entrambi.

11. Comunicazione tramite SSL record: dati frammentati in blocchi

- 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: Master secret creato con sequenze casuali di server hello e client hello → ≠ + reinvio e messaggi di hs non utilizzabili dal crittoanalista per sostituirsi in comunicazioni successive. Ogni blocco inoltre è cifrato con il MAC e quindi difficile forzabile dal crittoanalista in quanto viene anche cifrato. Un attacco volto a modificare la comunicazione è quindi difficile divenendo quanto la decentralizzazione della chiave simmetrica. Il canale è immune agli attacchi man-in-the-middle grazie all'uso di certificati e solo S può decifrare il pre-master secret in quanto solo lui conosce la chiave privata.

⚠ La sequente vista devono avere veramente casuali altrimenti il protocollo diventa molto debole.

Hmacropic: finito servizio da controllo

Sicurezza: chiavi quanto le più delicate cipher-suite selezionata. È buona norma disabilitare cifrari innervosi

• PROTOCOLLO TLS

Funziona sul SSL. Abbiamo sempre un client (browser web) ed un server (sito web) che stabiliscono un canale sicuro con chiavi simmetriche condivise utilizzate per identificazione e autenticazione. Anche in questo caso abbiamo le 2 fasi di hand shake, in cui si scambiano le chiavi, e il record layer protocol, in cui si usano le chiavi per cifrare e autenticare la comunicazione → TLS è tipicamente usato per autenticare il server con certificato digitale e non lo richiede all'utente
↓ handshakes

Pro: Client C possiede un insieme di chiavi pubbliche pk_1, pk_2, \dots di CA (autenticazione delle CA)

Server S: ha proprie chiavi pubbliche / private (pk_s, sk_s) per la firma. pk_s è contenuta in Certs rilasciato da una CA riconosciuta da C

1. C invia ad S messaggio iniziale protocollo DH: specifica gruppo G inviato dal client, che può essere \mathbb{Z}_p o una curva ellittica ed un generatore g (generatore di \mathbb{Z}_p o punto base g). Si calcola g^x con x nato a caso da C (oppure X_B su C_E). Calcola inoltre un nonce Na → tutto inviato al server insieme a cipher suite segnalate

2. S riceve le informazioni e completa DH: genera y e calcola g^y . Invia ad C g^y , un nonce Ns e calcola $K = g^{xy}$ ed applica una key derivation function per entrare le chiavi ks', kc', ks, kc . Dopo che invia pk_s , certs, la firma di calcolata usando sk_s su tutti i messaggi inviati. Tutti i dati inviati sono critti con ks' .

3. C calcola K e deriva ks', kc', ks, kc ed usa ks' per recuperare pk_s , certs e s . Controlla se possiede la pk_{ca} della CA che ha rilasciato Certs. Se non è nata la comunicazione si interrompe. Altrimenti, verifica s sui messaggi inviati pk_s . Calcola per il MAC usando kc' e lo invia ad S.

4. Se tutti i punti vanno a buon fine si passa a Record layer
↓ punti

1. C usa kc' per cifrare i messaggi che invia a S

2. S usa ks' per cifrare i messaggi per C.

↓ Termino

C ed S condividono kc e ks' per cifrare e autenticare la comunicazione. C verifica il certificato e quindi sa che pk_s è corretta. Se s è valido, C è in grado di comunicare con S (è generata con sk_s conosciuta solo da S). Dato che S firma tutti i messaggi per il protocollo di RT siamo di sicuro dai catturati man-in-the-middle e con l'utilizzo di we siamo di sicure anche da attacchi passivi.
↓ Differenza con SSL

Non più ormai avere RSA ma solo DH: Forward Security (sicurezza futura) → il numero y è mai gestito e viene buttato via mentre le chiavi private viene conservata e ricordandola si possono decifrare i dati sensibili scambiati precedente- mente

Authenticated Encryption: garantisce confidenzialità, autenticità ed integrità.

↓ 3 modi

Encrypt-then-MAC: $m' = C(m, k) || MAC(C(m, k))$

Encrypt-and-MAC: $m' = C(m, k) || MAC(m)$

MAC-then-encrypt: $m' = C(m || MAC(m), k)$
(TLS)

→ usa $u' || k$

→ usa k

→ usa k

Criptovalute

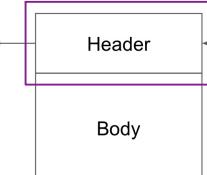
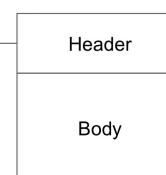
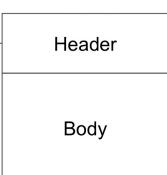
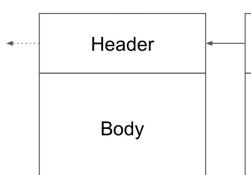
Si è iniziato nel 2009 come alternativa alle monete gestite dalle banche → Utilizzano tecniche di crittografia. Sono quindi utilizzabili solo in modo digitale. Il concetto nasce nel 1983 ma il Bitcoin è il primo esempio funzionante nel 2009 → Funziona in quanto non replicabili e differente dalle transazioni digitali ordinarie in quanto queste sono la digitalizzazione del movimento di soldi fisici.

• FUNZIONAMENTO

Bitcoin è un'architettura peer-to-peer collegati tramite Bitcoin core che permette di eseguire le transazioni. Ogni nodo Bitcoin salva il libro contabile di tutti i partecipanti. Per l'accorciamento viene utilizzato il consenso, pratico con cui quale dei modi si mettono d'accordo inserendo una nuova pagina.

↓ Fasi

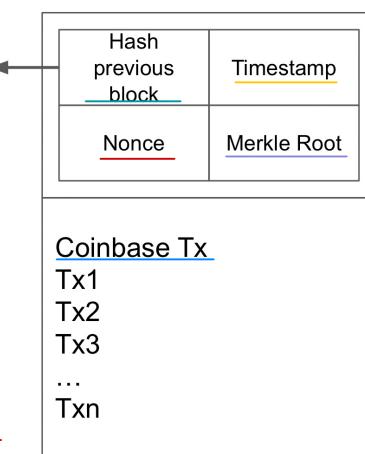
1. Scelta di un leader: competizione = rendere problema crittograficamente complesso (Sudoku). Questo dove accade ogni 10 secondi. Gli utenti i computer sempre più potenti si deve aumentare la complessità. Se fissa troppo difficile nessuno parteciperà e quindi escludono questi canali.
2. Creazione di una nuova pagina del libro contabile: Nel mentre che c'è la competizione i modi si inviano transazioni e quindi il leader rimane in ascolto per creare la pagina.
3. Da pagina viene inviata ai modi: quando viene inviata viene controllata la correttezza e nel caso di essere la pagina viene rifiutata.
4. Se la pagina viene accettata la pagina viene inserita e viene fatta una transazione speciale in cui il leader riceve una ricompensa in Bitcoin ed insieme ogni transazione riceve una piccola commissione per il leader.



Blockchain (struttura a catena)

Body = transazioni

Header = metadata e linking information



→ transazione di ricompensa del leader

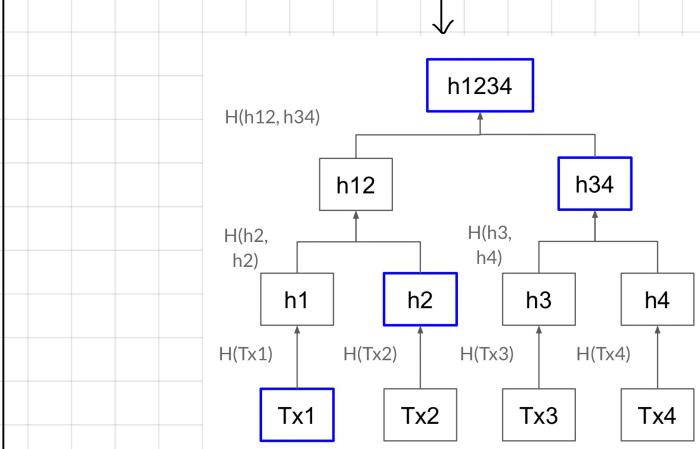
→ ts creazione transazione: stabilito da protocollo

→ primo risultato delle competizioni di prima (vedi f Hash)

Viene fatto lo Sha256(block_header, nonce) < value. Il value è dato dal difficulty-target e più è piccolo più è difficile → L'unico metodo è il brutto forza bruta sul nonce. Quando viene trovato si è vinta la competizione (proof of work).

È facile verificare lo Sha256. La difficoltà aumenta incrementando il numero di 0.

Alberi di Merkle: permettono di verificare l'integrità dati su cui viene applicata → hash ad albero → Nei Bitcoin chiamiamo transazioni: ne calcolo l'hash e poi prosegui a calcolare gli hash delle concatenazioni a due a due di elementi adiacenti fino ad arrivare a radice intera. Se ho fatto Tx1 e compongo hash (public) mi faccio dare l'hash e l'albero e calcolo le l'hash dei sottostanti fino ad arrivare alla radice → complessità logaritmica: altezza dell'albero



- Creato del leader quando crea la pagina ed inserisce la radice nell'header e viene messo per la verifica della correttezza delle transazioni
 - Minha blocco precedente: Blockchain immutabile → Se modifica una transazione precedente il Merkle tree è inconsistente e se ricalcola la root dovrà rifare anche un nuovo nonce. Lo trova e cambia a questo punto l'header e quindi cerca il blocco bisognoso di blocco precedente e dovrà ricalcolare il nonce anche del successivo fino all'attuale. Se riesce a fare tutto ciò c'è modo il blocco gli altri vedranno l'inconsistenza dell'hash del blocco precedente e non lo accettano quindi l'attacco fallisce
- ↓ Atacco 51%

Se alcuni minatori hanno una potenza di calcolo > 51% di tutta la rete abbra足e i propri Bitcoin.

Transazioni

Costituita dal invio di Bitcoin da A a B: $Tx(A, B, BTC)$ → firmate digitali: autenticate, integre, non ripudibili. → Ogni utente ha una chiave (generata da software) privata ed una pubblica ed i Bitcoin hanno indirizzi generati dalle chiavi con le curve ellittiche. (chiavi pubbliche più comuni). Per spendere si deve generare una firma con la chiave privata autenticabile con la chiave pubblica. Siano A e B gli utenti. Sia A_{pub} e B_{pub} le loro chiavi pubbliche e A_{priv} e B_{priv} i loro indirizzi

Transazione

A: $Sign(Tx(A, B, 50), S_{priv}) : -Msy$
A: $Send(Msy, AddrB)$
B: $Verify(Msy, P_{pub}) : -True$

} Generata dai programmi in Script (linguaggio a stack)

Indirizzo di Escrow: multi-signature → Anche se inviare da A a B invio ad indirizzo multi-signature: per autenticare serve la firma di una terza parte fidata

→ Preghiamo per spendere tutto quello che ho: differenza tra input ed output dividendo input → posso rimanere a me come resto: UTXO

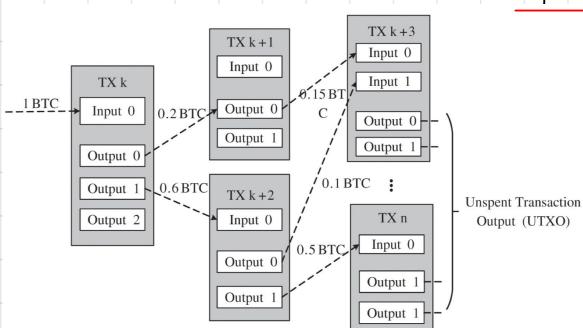
Nel mondo reale Bitcoin aveva un corrispondente che doveva avere la chiave privata per spendere i soldi inviati ad un certo indirizzo.

Esempi contratti

Ethereum: Nota nel 2014 per creare app decentralizzate in cui i peer eseguono transazioni più complesse → Smart contract: simil-Java → ETH viene utilizzato per ricompensare chi esegue smart-contract

↓ smart contract più popolari

Token: intercambiabili → tutti uguali, valore è la quantità non fungibili (NFT) → tutti diversi, hash: parametri che vengono inseriti davvero



l'bank → il valore è dato dall'emisfero

↓
da video giochi:

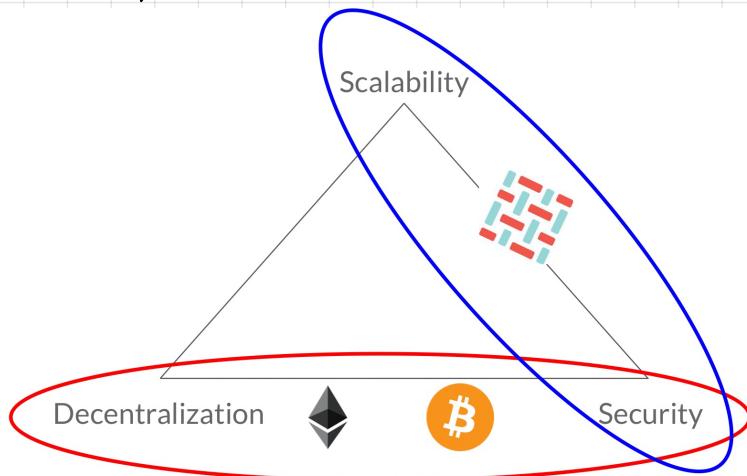
Nel 2017 comincia rollamento di Ethereum per traffico interno
non replicabili, non sono alterabili e tutti possono vedere le transazioni

↓ caso d'uso

Filiera produttiva di un prodotto

Svantaggi: costi elevati transazione, lentezza, ~~privacy~~

Al massimo si possono avere 2 caratteristiche su 3



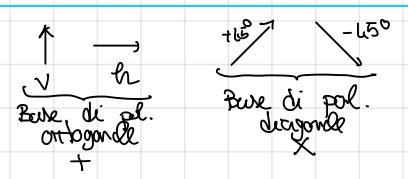
Mecanica Quantistica

• PRINCIPI

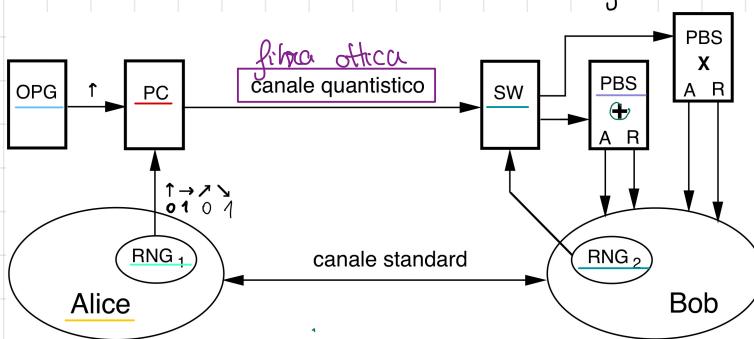
- SOPPOSIZIONE:** capacità per un sistema di avere più valori contemporaneamente (bit 0 o 1 sostituito da qbit = $\alpha|0\rangle + \beta|1\rangle$ dove α, β complessi / $|\alpha|^2 + |\beta|^2 = 1$) → evoluzione e cui risultato
- DECERPERIA:** L'operazione fu perdere la nonopposizione e si avrà solo lo stato singolo
- XO-CLOVING:** Non si può copiare uno stato quantistico non nato
- ENTANGLEMENT:** Possibilità di creare due elementi in stato quantistico correlato. Queste mantengono correlazione anche a grandi distanze

• PROTOCOLLO BB84

Due bitoni polarizzati: 4 stati di polarizzazione:



È possibile distinguere le polarizzazioni solo con misure relative a 2 stati ortogonali dello stesso base: si invia bit 0 o 1 ad uno stato e il bit inviano all'ortogonale (0 e ↑ ed 1 ad ↓). Se il catturatore aveva il fotone lancia un'impresa e quindi si vede se ha operato sulla comunicazione. È però possibile vedere solo alla fine e quindi si ha solo per lo scambio delle chiavi.



- Prepara tutti i fotoni con polarizzazioni.
- Deve mandare bit 0 o 1 e scegliere a caso base di polarizzazione.
- Invia polarizzazione a fotone.
- Sceglie a caso una delle 4 possibilità.
- Sceglie a caso base in cui misurare.
- Misuratore: se viene notata la stessa base di invio deve misurare correttamente il bit inviato. Se sbaglia, il risultato è casuale e ha probabilità di ottenere 0 o 1.
- F: polarizzazione fotone

PBS: Beam splitter polarizzante → il fotone può essere inviato ad A ed R (Assorbimento e Riflessione) →

A: il fotone viene inviato ad A con $P = \cos^2 \theta$ ed come θ
B: il fotone viene inviato ad R con $P = \sin^2 \theta$ ed come θ

→ S: polarizzazione PBS

polarizzazione S perpendicolare a S (S^{perp})

$\theta = 0^\circ$: il fotone esce da A con $P=1$ e polarizzazione $F=S$
 $\theta = 90^\circ$: il fotone esce da R con $P=1$ e con polarizzazione S^2
 $\theta = \pm 45^\circ$: $\cos^2 \theta = \sin^2 \theta = \frac{1}{2}$ → metà delle volte esce da A con R con polarizzazione 90° → Misura quantistica: la lettura con il PBS ha dunque lo stato quantistico precedente

? polarizzazioni combinate: misura correttamente (stessa base) polarizzazione 0° e metà delle non ortogonali → fenomeno

bit e fotone inviato da A

baso di B	0	↑	0	↑	1	→	1	↓
	+	\uparrow 0	$\uparrow \rightarrow$	\rightarrow	\downarrow	$\uparrow \rightarrow$		
X	$\nearrow \downarrow$ 50% 50%	\nearrow	$\nearrow \downarrow$ 50% 50%	\downarrow				

CANALE QUANTISTICO: Alice ha seq. iniziali composta di n hit e la vuole inviare a Bob codificandole come stati di un fotone (utilizzate tecniche di correzione di errori)

↓

for $i=1$ to n

Alice: sceglie uno hbar a caso, codifica $S_A[i]$ e invia il fotone a Bob

Eve: intercetta il fotone, lo misura con una sua hbar, lo invia a Bob, costituisce $S_E[i]$ (non vi) : se misura con hbar diversa plausibile originale per Alice: sceglie una hbar a caso, interpreta il fotone ricevuto, costituisce $S_B[i]$

CANALE STANDARD: (in questo ci sono certificazione)

Bob: comunica ad Alice la sequenza di hbar scelte

Alice: comunica a Bob le hbar scelti

Alice e Bob: 1. estraggono S_A e S_B corrispondenti alle hbar comuni (lunghe circa la metà) che possono essere identificate hbar. Siccome devono fare le conclusioni contro Eve, scartano alcuni hit e ignorano le seguenti S_A'' e S_B'' .

2. Si confrontano S_A'' e S_B'' e se la % di hit diversi è \rightarrow QBER (hit ormai rafra) allora STOP (Eve presente)
3. Altrimenti calcolano $S_A \oplus_{\text{diff. insensistica}} S_A''$ e $S_B \oplus_{\text{diff. insensistica}} S_B''$, le decodificano applicando la correzione degli errori ed ottengono una sequenza S_C
4. Si prende $K = \ln(S_C)$: d'isio del codificante

ESEMPIO: Alice

$S_A = 1011100\dots$	hit di Alice
hbar: $+ x + x x + x \dots$	hbar di Alice
$\rightarrow \uparrow \rightarrow \downarrow \uparrow \uparrow \dots$	fotoni di Alice

Eve

hbar: $+ + + x + x +$	hbar di Eve
$\rightarrow \rightarrow \rightarrow \uparrow \rightarrow \rightarrow$	fotoni di Eve
$S_E = 1111011$ <small>(corrisponde)</small>	hit di Eve

Bob senza Eve

hbar: $+ x + + + x x$	hbar di Bob
$\rightarrow \uparrow \rightarrow \uparrow \rightarrow \uparrow \uparrow$	fotoni di Bob
$\oplus \circ$ <small>controllo ON</small>	bit di Bob
	conservati

Bob con Eve

hbar: $+ x + + + x x$	hbar di Bob
$\rightarrow \downarrow \rightarrow \rightarrow \uparrow \rightarrow \uparrow$	fotoni di Bob
1111010 <small>controllo NO</small>	hit di Bob

Eve deve cercare di fare meno errori possibili: attacca pochi fotoni per non fornire sequenze → lunghe hash per rendere inutile conoscere hit chiave

• CRIPTOGRAFIA POST-QUANTISTICA

Computer quantistici renderebbero inutili cifrari a chiave pubblica. Per bucare una RSA esiste un algoritmo cubico per fattorizzare. La macchina quantistica necessita di una porta emittente un cubico di gate che vanno fatti lavorare mantenendo la sovrapposizione degli stati (estendendo interferenze). I problemi NP-completi rimangono tali e non cambia la calcolabilità. One-time pad non risulta nemmeno crittograficamente significativo della computazione quantistica mentre i cifrari simmetrici di uso comune (ad esempio AES) sono influenzati

attraverso l'algoritmo di Grover: riduce la ricerca sequenziale in modo più semplice: riduce $O(n)$ in $O(\sqrt{n})$ con algoritmi probabilistici → da ricerca della chiave in uno spazio di 2^n pura da $O(2^n)$ in $O(2^{n/2})$. Per mantenere la ricerca non aumentare la lunghezza della chiave