

## FUNZIONI ONE-WAY TRAPDOOR

Sono funzioni semplici da calcolare, facili da invertire solo se si dispone di una informazione aggiuntiva sui dati.

Esempio: Calcolare  $y = x^2 \text{ mod } p$ , richiede tempo polinomiale

$x = y^{1/2} \text{ mod } p$ , e' difficile se s

e' composto

trovare  $x \mid y = x^2 \text{ mod } p$ , e' difficile

(logaritmo discreto)

## Crittografia a chiavi pubbliche

Vantaggi: - niente scambio segreto di chiavi

- sono sufficienti 2n chiavi

Svantaggi: - molto lente

- esposti ad attacchi chosen plain text

E puo':

scegliere  $m_1, \dots, m_k$

calcolare  $c_1, \dots, c_k$

confrontare con ogni  $c^*$  messaggio sul canale.

- se non esiste  $c_i = c^*$  allora E

scopre che nessun  $c_j \in \{c_1, \dots, c_k\}$  e' passato sul canale

- se esiste  $c^* = c$ : allora E può decifrarlo

Al giorno d'oggi si usano cifrari ibridi:

- Si usa AES per le comunicazioni di massa
- Si usa un cifrario a chiave pubblica per lo scambio delle chiavi

E' lento lo scambio delle chiavi, ma sono tipicamente lunghe alcune decine di byte;  
chosen plain text non e' un problema, le chiavi devono essere casuali.

Tipicamente la chiave deve essere estraetta da un certificato digitale per evitare attacchi man-in-the-middle

## RSA

### Creazione della chiave

B := destinatario

B sceglie (p, q) numeri primi molto grandi

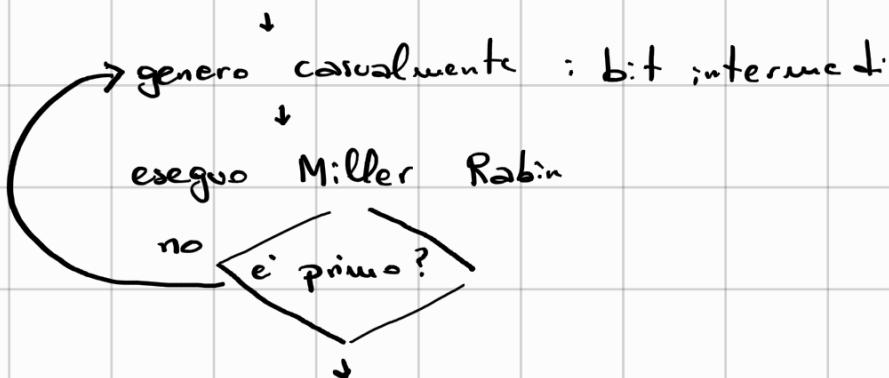
$n = p \cdot q$  deve avere

• 2048 cifre binarie per proteggere i dati fino al 2030

• 3072 cifre binarie per andare oltre

esempio:  $|p| = 15$  cifre binarie

1 00 ... 00 1      15 bit



B calcola  $n = p \cdot q$  tempo polinomiale

$$\phi(n) = (p-1)(q-1) \quad \text{tempo polinomiale}$$

B sceglie  $e < \phi(n)$  t.c.  $\text{MCD}(e, \phi(n)) = 1$

tempo polinomiale

B calcola  $d := e^{-1} \bmod \phi(n)$  tempo polinomiale,

algoritmo di Euclide esteso

B: chiave pubblica:  $n, e$

privata:  $d$

Ovviamente, anche  $p$  e  $q$  non devono essere noti

Cifratura:  $m$ : sequenza binaria vista come un intero  
con  $m < n$  altrimenti si divide

$m$  in blocchi di  $b = \lceil \log_2 n \rceil$  bit.

$m$  non è necessario per evitare che  $m$  e

$m \bmod n$  producano lo stesso cifrogramma

Sposto  $b$  viene fissato a priori

Vale che  $m < 2^b < n$

Fissare  $b$  significa dare un lower bound  
alla lunghezza della chiave pubblica

Definisco:  $c = m^e \pmod{n}$  tempo polinomiale

Decifrare:  $m = c^d \pmod{n}$  tempo polinomiale

Si usa l'algoritmo di quadrate successive

Esempio:  $p=5, q=11$   $n=55$

$$\phi(n) = 40$$

Se dico  $e=7$

$$\text{MCD}(e, \phi(n)) = 1$$

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

$$\text{EE}(7, 40)$$

$$(1, -1, \dots)$$

$$\text{EE}(40, 7 \pmod{40})$$

$$(1, 3, -2, 3, 40, 7)$$

$$\text{EE}(7, 40 \pmod{7})$$

$$(1, 2, 1+2, 1^2, 5, 7)$$

$$\text{EE}(5, 7 \pmod{5})$$

$$(1, 1, 0-1, 5, 2)$$

$$\text{EE}(2, 1)$$

$$(1, 0, 1-0, 2, 1)$$

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

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

$$K_{\text{pub}} = (7, 55)$$

$$K_{\text{priv}} = 23$$

$$m = 28$$

$$C = 28^7 \pmod{55}$$

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

Si dimostra la correttezza del cifrario RSA:

$$\text{Th: } D(C(m, k_{\text{pub}}), k_{\text{priv}}) = m$$

$$\text{Dim: } (m^e \pmod{n})^d \pmod{n} = m$$

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

Si distinguono due casi:

-  $m, n$  coprimi  $\text{MCD}(m, n) = 1$

Per il teorema di Eulero:  $m^{\phi(n)} \pmod{n} = 1$  (1)

Per definizione di inverso:  $ed = 1 \pmod{\phi(n)}$

$$\Rightarrow ed = r \cdot \phi(n) + 1 \text{ con } r \in \mathbb{N} \quad (2)$$

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

$$(2) \\ \Leftrightarrow$$

$$m^{1+r\phi(n)} \pmod{n}$$

$$(3) \\ \Leftrightarrow$$

$$m \cdot m^{r\phi(n)} \pmod{n}$$

$$(4) \\ \Leftrightarrow$$

$$m \cdot (m^{\phi(n)})^r \pmod{n}$$

$$(5) \\ \Leftrightarrow m \cdot 1^r \pmod{n} = \begin{cases} \text{HP: } m \neq n \\ m \end{cases}$$

Th

-  $m, n$  non sono coprimi  $\text{MCD}(n, m) \neq 1$

$$p \mid m \text{ e } q \nmid m \text{ oppure } p \nmid m \text{ e } q \mid m$$

Supponiamo di essere nel primo caso:

$$p \mid m \Rightarrow m = 0 \pmod{p}$$

$$\forall x. m^r \pmod{p} = 0$$

$$\forall x. \quad m^r - m \bmod p = 0$$

Scelgo  $x = e \cdot d$

$$m^{ed} - m \bmod p = 0$$

$$\begin{aligned} \text{Si calcola } m^{ed} \bmod q &= m^{1+r\phi(n)} \bmod q = \\ &= m \cdot m^{r\phi(n)} \bmod q = \\ &= m \cdot m^{r(p-1)(q-1)} \bmod q = \\ &= m \cdot (m^{q-1})^{r(p-1)} \bmod q = m \cdot (m^{t(q)})^{r(p-1)} \bmod q = \\ &= m \cdot 1^{r(p-1)} \bmod q = m \bmod q \end{aligned}$$

$$\text{Dunque } m^{ed} - m \bmod q = m^{ed} - m \bmod p$$

$$\Rightarrow m^{ed} - m \bmod p \cdot q = 0$$

$$\Leftrightarrow m^{ed} - m \bmod n = 0$$

$$\Rightarrow m^{ed} \bmod n = m \bmod n$$

$$\left\{ \begin{array}{l} m \in \mathbb{N} \\ \Rightarrow m^{ed} \bmod n = m. \end{array} \right.$$

oh. Non puo' valere  $m \mid p \wedge m \mid q$  perché altrimenti  $m \mid p \cdot q = n$

La sicurezza di RSA dipende dalla fattorizzazione

fattorizzare  $\Rightarrow$  forzare RSA ✓

fattorizzare  $\Leftarrow$  forzare RSA ?

Si congettura di sì

Si potrebbe pensare di usare l'operatore radice

$m = \sqrt[n]{c} \bmod n$  ma è difficile quanto fattorizzare

oppo. affacciare un solo crittogramma è difficile  
quanto affacciare il sistema

Si potrebbe cercare di calcolare  $\phi(n)$ :

calcolare  $\phi(n)$  senza conoscere  $p, q$  e' difficile  
quanto la fattorizzazione

esempio:

Se fattorizzo allora trovo  $p, q$

quindi calcolo velocemente  $\phi(n) = (p-1)(q-1)$

Si dimostra che dati  $n, \phi(n)$  calcolo  $p, q$  in tempo

polinomiale:

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

$$= pq - (p+q) + 1 =$$

$$= n - (p+q) + 1$$

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

Si ricorda che  $(p+q)^2 = (p-q)^2 + 4n$

$$x_1 := p+q$$

$$x_2 := p-q$$

$$\Rightarrow x_1^2 = x_2^2 + 4n$$

$x_1$  e' nota

$$\text{Ricavo } x_2 : x_2 = \sqrt{x_1^2 - 4n}$$

A questo punto:

$$p = (x_1 + x_2) / 2$$

$$q = (x_1 - x_2) / 2$$

Quindi calcolare  $\phi(n)$  e' difficile quanto fattorizzare

Fattorizzare c'è ancora un problema difficile?

Sí, ma non quanto lo era negli anni '70

+ potenza di calcolo

+ GNFS fattorizza  $n$  in  $O(2^{\sqrt{\log b}})$

operazioni con  $b := \lceil \log_2 n + 1 \rceil$  bit.

Forza bruta ne richiede  $\mathcal{O}(2^n)$

Con la potenza di calcolo attuale si fattorizzano semiprimi

Lunghi 768 bit.

Esistono interi con una struttura particolare che possono essere fattorizzati velocemente.

I bit di sicurezza della chiave  
di RSA sono  $\sqrt{b}$ , non  
 $b$  come in AES.

Criteri:

$$|p| > 1024, \quad |q| = 1024$$

MCD( $p-1, q-1$ ) piccolo

conviene scegliere  $\frac{p-1}{2}, \frac{q-1}{2}$  coprimi

$p-1, q-1$  devono contenere un fattore primo grande

Non riusare  $p \circ q$

$p-q$  deve essere polinomiale in  $n$ , non logaritmico

$p-q \sim n^\epsilon$  perché  $n \sim p^2 - q^2$ , quindi  $\sqrt{n}$  sarà

vicina a entrambi

$$\frac{p+q}{2} > \sqrt{n}$$

$$\left(\frac{p+q}{2}\right)^2 = \left(\frac{p-q}{2}\right)^2 + n$$

$$\left(\frac{p+q}{2}\right)^2 - n = \left(\frac{p-q}{2}\right)^2 \Rightarrow \left(\frac{p+q}{2}\right)^2 > n$$

Sì scandiscono gli interi maggiori di  $\sqrt{n}$  fino a 2 tc.

$Z^2 - n = w^2$  è un quadrato perfetto

$$Z = \frac{p+q}{2}, \quad w = \frac{p-q}{2}$$

$$p = Z + w$$

$$q = Z - w$$

Sostanzialmente, c'è un forte buco in un intorno di  $\sqrt{n}$ .

Usare e piccolo permette di velocizzare la cifratura

↓

decifratura

oss. se vale che  $m^e < n$  allora diventa facile

calcolare  $c = m^e \bmod n = m^e$  (non c'è una radice  
in modulo, c'è facile)