

Crittografia a chiave pubblica

One-Time Pad (1917)

Non può essere decifrato senza conoscere la chiave

Assolutamente sicuro, ma...

- richiede una nuova chiave segreta per ogni messaggio
- perfettamente casuale
- e lunga come il messaggio da scambiare!

come si genera e come si scambia la chiave???

Estremamente attraente per chi richieda una sicurezza assoluta e sia disposto a pagarne i costi

AES

Advanced Encryption Standard (AES)

- standard per le comunicazioni riservate ma "non classificate"
- pubblicamente noto e realizzabile in hardware su computer di ogni tipo
- Chiavi brevi (qualche decina di caratteri, 128 o 256 bit)

e le chiavi ?

Ma come si può scambiare una chiave segreta con facilità e sicurezza?

La chiave serve per comunicare in sicurezza , ma deve essere stabilita comunicando "in sicurezza" senza poter ancora usare il cifrario...

Distribuzione delle chiavi

Nel 1976 viene proposta alla comunità scientifica un algoritmo per generare e scambiare una chiave segreta su un canale insicuro



senza la necessità che le due parti si siano scambiate informazioni o incontrate in precedenza

questo algoritmo, detto **protocollo DH**, è ancora largamente usato nei protocolli crittografici su Internet

Merkle

Hellman

Diffie

Distribuzione delle chiavi

Nel 1976 viene proposta alla comunità scientifica un algoritmo per generare e scambiare una chiave segreta su un canale insicuro

Turing Award 2015



Merkle

Hellman

Diffie



Protocollo DH per lo scambio pubblico delle chiavi

Alice e Bob generano una chiave k (un numero intero) in modo *incrementale*

- scambiandosi in chiaro alcuni pezzi di k
- questi pezzi sono sufficienti a ricostruire la chiave k solo se **combinati con informazioni private** in possesso di Alice e Bob, e diverse per entrambi

Non si condividono informazioni 'secrete', ma si COSTRUISCE insieme una chiave segreta

La chiave non viene mai trasmessa!

Informazione
privata di Alice



facile!

facile!



Numero
'pubblico' di
Alice

Numero
'pubblico' di
Bob

difficile!

Informazione
privata di Bob

facile!

facile!



Funzione one-way

calcolare il numero pubblico dall'informazione privata
deve essere facile (Alice e Bob)

ricavare l'informazione privata dal numero pubblico deve
essere difficile (Eve)

Il protocollo DH utilizza una funzione one-way

facile da calcolare ...



Dato x , si conosce un algoritmo efficiente (polinomiale) per calcolare $y = f(x)$

e difficile da invertire ...



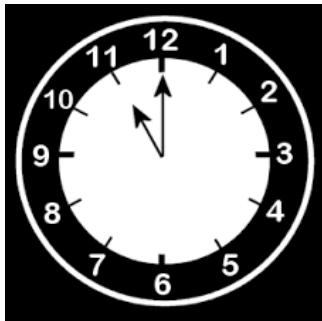
Dato y , si conoscono solo algoritmi inefficienti (esponenziali) per trovare un x t.c $y = f(x)$, ossia per calcolare $x = f^{-1}(y)$

Aritmetica modulare

È un campo della matematica ricco di funzioni one-way.

Sia m un intero positivo.

Si considerano solo i numeri interi compresi tra 0 e $m-1$ come se fossero 'disposti ad anello': ogni volta che si passa dal punto di partenza, si riparte da 0, come accade sul quadrante di un orologio [Aritmetica dell'orologio]

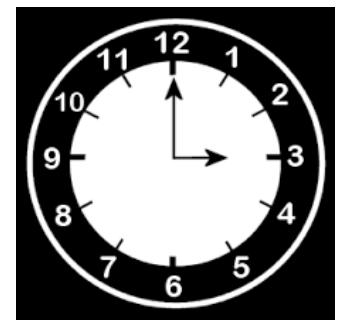


Alice e Bob vanno a una festa alle 11 di sera e ci restano 4 ore, a che ora tornano?

Alle 3 di notte, ma $11 + 4 = 15$

Abbiamo ridotto modulo 12!

$$15 \bmod 12 = 3$$



Aritmetica modulare

Sommare o sottrarre m , o un suo multiplo, è come sommare o sottrarre 0

non interessa il numero dei giri, ma solo quello che avanza alla fine (il **resto** della divisione)

il valore modulare $k = N \bmod m$ coincide con il resto della divisione intera tra N e m

ESEMPI

$$m = 13$$

$$29 \bmod 13 = 3 \quad \text{infatti } 29 = 3 + 2 * 13$$

$$39 \bmod 13 = 0 \quad \text{infatti } 39 = 0 + 3 * 13$$

$$27 \bmod 13 = 1 \quad \text{infatti } 27 = 1 + 2 * 13$$

Proprietà

$$(a + b) \bmod m = (a \bmod m + b \bmod m) \bmod m$$

$$(a - b) \bmod m = (a \bmod m - b \bmod m) \bmod m$$

$$(a \times b) \bmod m = (a \bmod m \times b \bmod m) \bmod m$$

$$a^{r \times s} \bmod m = (a^r \bmod m)^s \bmod m \quad (r \text{ e } s \text{ interi positivi})$$

ESEMPIO

$$(12456 \times 3678) \bmod 10 = 45813168 \bmod 10 = 8$$

$$(12456 \times 3678) \bmod 10 = (6 \times 8) \bmod 10 = 8$$

Aritmetica modulare

Usata in molti algoritmi crittografici per

- ridurre lo spazio dei numeri su cui si opera, e quindi aumentare la velocità di calcolo
- rendere *difficili* problemi computazionali che sono semplici (o anche banali) nell'algebra non modulare

Aritmetica modulare

Nell'aritmetica modulare le funzioni tendono a comportarsi in modo 'imprevedibile'

La funzione 2^x nell'aritmetica ordinaria è una funzione monotona crescente

x	1	2	3	4	5	6	7	8	9	10	11	12
2^x	2	4	8	16	32	64	128	256	512	1024	2048	4096

La funzione $2^x \text{ mod } 13$ diventa

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

Aritmetica modulare

$$2^x = 512, \quad x = ?$$

per errore, supponiamo $x = 8$

$2^8 = 256 \rightarrow$ è stato scelto un numero **troppo basso**
proviamo $x = 9$ ok!

$$2^x \bmod 13 = 5, \quad x = ?$$

per errore, supponiamo $x = 6$

$2^6 \bmod 13 = 12 \rightarrow$ x è **troppo alto...**

proviamo $x = 3$, otteniamo **8 (troppo alto)**

proviamo $x = 2$, otteniamo **4 (troppo basso)**

???

occorre provare tutti i valori di x

la risposta giusta è $x = 9$

Aritmetica modulare

- $n \in \mathbb{N}$

$$Z_n = \{0, 1, 2, \dots, n - 1\}$$

$$Z_n^* \subseteq Z_n$$

è l'insieme degli elementi di Z_n co-primi con n .

- Se n è primo, $Z_n^* = \{1, 2, \dots, n - 1\}$
- Se n non è primo, calcolare Z_n^* è computazionalmente difficile

richiede tempo proporzionale al **valore** di n (per confrontare n con gli elementi di Z_n) quindi esponenziale nella lunghezza della sua rappresentazione.

Funzione di Eulero

Il numero di interi minori di n e co-primi con esso

$$\phi(n) = |\mathbb{Z}_n^*|$$

n primo $\Rightarrow \phi(n) = n - 1$

TEOREMA

$$n \text{ composto} \Rightarrow \phi(n) = n (1 - 1/p_1) \dots (1 - 1/p_k)$$

p_1, \dots, p_k fattori primi di n , presi senza molteplicità

TEOREMA

n prodotto di due primi (semiprimo)

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

Teorema di Eulero

Teorema di Eulero

Per $n > 1$ e per ogni a **primo** con n ,

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

Piccolo Teorema di Fermat

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

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

Generatori

$a \in Z_n^*$ è un generatore di Z_n^* se la funzione

$$a^k \bmod n \quad 1 \leq k \leq \phi(n)$$

genera tutti e soli gli elementi di Z_n^*

produce come risultati tutti gli elementi di Z_p^* , ma in un ordine difficile da prevedere

ESEMPIO: $g = 2$ è un generatore di $Z_{13}^* = \{1, 2, \dots, 12\}$

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

Generatori

Non per tutti i valori di n , Z_n^* ha generatori

- Z_8^* non ammette generatori

TEOREMA

Se p è un numero primo, Z_p^* ha almeno un generatore

Per n primo, non tutti gli elementi di Z_n^* sono suoi generatori

Elevamento a potenza

Dati un numero primo p (e.g., di 600 cifre decimali), un generatore g per \mathbb{Z}_p^* e un intero $k < p$, calcolare

$$y = g^k \bmod p$$

ALGORITMO 1

$$y = g \cdot g \cdot g \cdots g \quad (k \text{ volte})$$

esegue **k moltiplicazioni**, $k \sim 10^{600}$!!!

esponenziale nel numero di cifre di k

ALGORITMO 2

esponenziazione veloce

esegue un **numero di moltiplicazioni** proporzionale al numero di **cifre binarie di k** , e non al suo valore

(600 cifre decimali, circa 2000 cifra binarie)

Elevamento a potenza 'veloce'

Quadrature successive

$$(23)_{10} = (10111)_2$$

$$\begin{aligned}3^{23} \bmod 29 &= 3^{16+4+2+1} \bmod 29 \\&= (3^{16} \times 3^4 \times 3^2 \times 3) \bmod 29\end{aligned}$$

$$3^2 \bmod 29 = 3 \times 3 \bmod 29 = 9$$

$$3^4 \bmod 29 = 9 \times 9 \bmod 29 = 23$$

$$3^8 \bmod 29 = 23 \times 23 \bmod 29 = 7$$

$$3^{16} \bmod 29 = 7 \times 7 \bmod 29 = 20$$

$\log k = \log 23 = 4$ moltiplicazioni

$$3^{23} \bmod 29 = 20 \times 23 \times 9 \times 3 \bmod 29 = 8$$

7 moltiplicazioni in totale

Elevamento a potenza 'veloce'

$$k = (k_t k_{t-1} \dots k_1 k_0)_2$$

$$k = \sum_{i=0}^t k_i 2^i = \sum_{k_i \neq 0} 2^i$$

$$g^k = g^{\sum_{k_i \neq 0} 2^i} = \prod_{k_i \neq 0} g^{(2^i)}$$

$$g^k \bmod n = \left[\prod_{k_i \neq 0} g^{(2^i)} \right] \bmod n = \left[\prod_{k_i \neq 0} (g^{(2^i)} \bmod n) \right] \bmod n$$

Elevamento a potenza 'veloce'

$$f = g^k \bmod n \quad k = (k_t k_{t-1} \dots k_1 k_0)_2$$

```
f = 1;
for (i = t; i >= 0; i--) {
    f = (f * f) % n;
    if (k_i == 1) {
        f = (f * g) % n;
    }
}
return f
```

Logaritmo discreto

Dati un numero primo p , un generatore g per \mathbb{Z}_p^* e un intero $y < p$, trovare k tale che

$$y = g^k \bmod p$$

è difficile

- non è noto a priori in che ordine sono generati gli elementi
- quindi non è noto per quale valore di k si genera y

Algoritmo bruteforce

si provano tutti i valori possibili di k tra 1 e $p-1$, fino a trovare il valore che permette di ottenere y

Richiede un numero di 'tentativi' proporzionale al valore di p

→ Algoritmo esponenziale nel numero di cifre di p

Protocollo DH

Alice e Bob scelgono una coppia p, g (nota a tutti)

sceglie a caso

$$1 < a < p-1$$

(segreto di Alice)

calcola

$$A = g^a \text{ mod } p$$

invia A a Bob



sceglie a caso

$$1 < b < p-1$$

(segreto di Bob)

calcola

$$B = g^b \text{ mod } p$$

invia B ad Alice



A

B

riceve B e calcola

$$\begin{aligned} K &= B^a \text{ mod } p \\ &= g^{b \times a} \text{ mod } p \end{aligned}$$

riceve A e calcola

$$\begin{aligned} K &= A^b \text{ mod } p \\ &= g^{a \times b} \text{ mod } p \end{aligned}$$

Protocollo DH

Alice e Bob hanno costruito la stessa chiave segreta **K**

$$K = g^{b \times a} \text{ mod } p = g^{a \times b} \text{ mod } p$$

Alice e Bob hanno una chiave comune che possono usare per comunicare usando un cifrario simmetrico (AES)



ESEMPIO

Alice e Bob scelgono una coppia $p = 13, g = 2$

sceglie a caso

$$a = 5$$

(segreto di Alice)

calcola

$$A = 2^5 \bmod 13 = 6$$

invia 6 a Bob



sceglie a caso

$$b = 2$$

(segreto di Bob)

calcola

$$B = 2^2 \bmod 13 = 4$$

invia 4 ad Alice



6



4

riceve $B=4$ e calcola

$$K = 4^5 \bmod 13 = 10$$

riceve $A=6$ e calcola

$$K = 6^2 \bmod 13 = 10$$

Alice e Bob hanno una chiave in comune: il numero 10.

Sicurezza del protocollo



Eve intercetta la trasmissione e viene a conoscenza di p, g, A, B

Per ricostruire la chiave deve

- risolvere $A = g^a \text{ mod } p$ rispetto ad a , e calcolare $K = B^a \text{ mod } p$ oppure
- risolvere $B = g^b \text{ mod } p$ rispetto a b , e calcolare $K = A^b \text{ mod } p$

→ deve calcolare il logaritmo discreto di A oppure il logaritmo discreto di B ,
ma queste operazioni richiedono un numero di passi (sub)esponenziale nel numero di cifre di p

Easy challenge #1

$$g = 3$$

$$p = 131$$

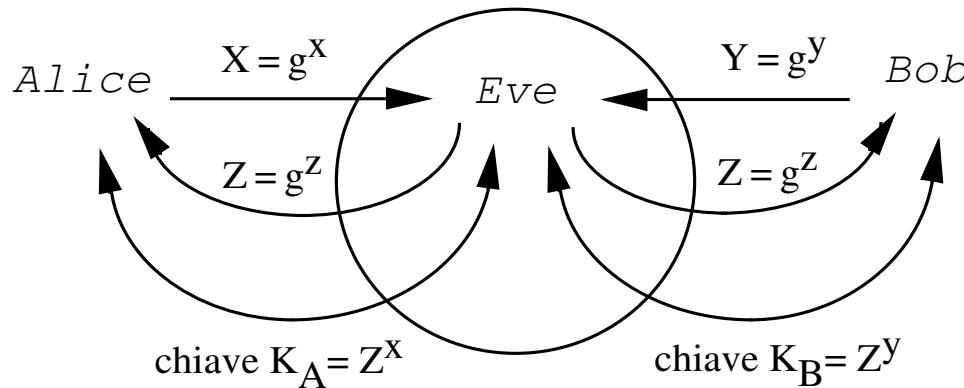
$$g^x \bmod p = 112$$

$$g^y \bmod p = 74$$

compute K[session]

Attacco 'man in the middle'

Eve si finge Bob agli occhi e Alice agli occhi di Bob, sostituendo **A** e **B** con un proprio valore $Z = g^z \text{ mod } p$



Alice e Bob interpretano **Z** come proveniente dall' altro partner e costruiscono le chiavi (diverse)

$$K_A = Z^x \text{ mod } p \quad K_B = Z^y \text{ mod } p$$

Eve conosce entrambe le chiavi

- comunica con **Alice** fingendosi Bob e usando l' AES con K_A
- comunica con **Bob** fingendosi Alice e usando l' AES con K_B

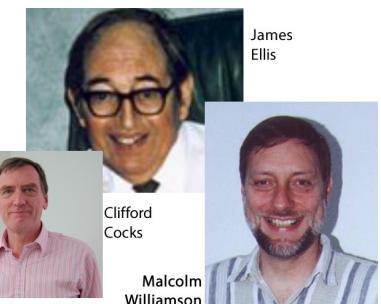
Crittografia a chiave pubblica

Nel 1976 D. e H. propongono alla comunità scientifica anche la definizione di **crittografia a chiave pubblica** (ma senza averne un'implementazione pratica)



rivoluziona il modo di concepire le comunicazioni segrete

Nata ufficialmente nel 1976
ma preceduta dal lavoro,
coperto da segreto, degli
agenti britannici (Ellis, Cocks e
Williamson)



Cifrari simmetrici

Nei cifrari simmetrici, la chiave di cifratura è uguale a quella di decifrazione (o l'una può essere facilmente calcolata dall'altra)

ed è nota solo ai due partner che la scelgono di comune accordo e la mantengono **segreta**

Cifrari a chiave pubblica (asimmetrici)

Obiettivo: permettere a tutti di inviare messaggi cifrati ma abilitare solo il ricevente (BOB) a decifrarli

Le operazioni di cifratura e decifrazione sono pubbliche e utilizzano due chiavi diverse:

k_{pub} per **cifrare**: è **pubblica**, nota a tutti;

k_{priv} per **decifrare**: è **privata**, nota solo a BOB

Cifrari a chiave pubblica

La **cifratura** di un messaggio m da inviare a BOB è eseguita da qualunque mittente come

$$c = C(m, k_{pub})$$

chiave k_{pub} e funzione di cifratura sono note a tutti

La **decifrazione** è eseguita da BOB come

$$m = D(c, k_{priv})$$

la funzione di decifrazione è nota a tutti, ma k_{priv} non è disponibile agli altri

Cifrari a chiave pubblica

1) Correttezza del procedimento di cifratura e decifrazione

BOB deve interpretare qualunque messaggio che gli altri utenti decidano di spedirgli.

Quindi per ogni possibile messaggio m

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

Cifrari a chiave pubblica

2) Efficienza e sicurezza del sistema

- a) la coppia di chiavi è facile da generare, e deve risultare praticamente impossibile che due utenti scelgano la stessa chiave (generazione casuale delle chiavi)
- b) dati m e k_{pub} è facile per ALICE calcolare il crittogramma $c = C(m, k_{\text{pub}})$ (adattabilità del sistema)
- c) dati c e k_{priv} è facile per BOB calcolare il messaggio originale $m = D(c, k_{\text{priv}})$ (adattabilità del sistema)
- d) pur conoscendo il crittogramma c , la chiave pubblica, e le funzioni di cifratura e decifrazione, è difficile per il crittoanalista risalire al messaggio m (sicurezza del cfrario)

Crittografia a chiave pubblica

La funzione di cifratura deve essere una funzione **one-way trap-door**

calcolare $c = C(m, k_{\text{pub}})$ è computazionalmente facile
decifrare c è computazionalmente **difficile**

a meno che non si conosca un meccanismo segreto,
rappresentato da k_{priv} (**trap-door**)

facile da calcolare ...



e difficile da invertire ...



a meno che non si conosca una chiave
segreta !



Adleman

Shamir

Rivest

RSA (1977)

propongono un sistema a chiave pubblica basato su una funzione “facile” da calcolare e “difficile” da invertire

Turing Award 2002



RSA (1977)

RSA si basa sulla moltiplicazione di due numeri primi p, q

Calcolare $n = p \times q$ è facile

Calcolare p, q da n è difficile
a meno che non si conosca uno dei due

Fattorizzare $n = 1841488427$

$$p = ?$$

$$q = ?$$

RSA (1977)

1. Si scelgono due numeri primi a caso p e q
2. Si calcola il prodotto $n = p * q$
3. Si calcola $\phi(n) = (p-1) * (q-1)$
4. Si sceglie a caso un numero e coprimo e minore di $\phi(n)$
5. Si calcola $d = e^{-1} \text{ mod } \phi(n) \rightarrow e d \equiv 1 \pmod{\phi(n)}$
 $\rightarrow e d = 1 + k \phi(n)$

Chiave pubblica $k_{\text{pub}} = (n, e)$

Chiave privata $k_{\text{priv}} = (d)$

Cifratura e decifrazione

- il messaggio m è codificato come una sequenza binaria, trattata come un numero intero
- $m < n$ (altrimenti m e $m \bmod n$ genererebbero lo stesso crittogramma)
- se $m \geq n \rightarrow$ cifratura a blocchi, di $\log_2 n$ bit ciascuno

CIFRATURA

$$c = C(m, k_{\text{pub}}) = m^e \bmod n$$

DECIFRAZIONE

$$m = D(c, k_{\text{priv}}) = c^d \bmod n$$

Correttezza

Ipotesi: m e n co-primi $\gcd(m,n) = 1$

$$\begin{aligned} D(c, k_{\text{priv}}) &= D(C(m, k_{\text{pub}}), k_{\text{priv}}) = (m^e \bmod n)^d \bmod n \\ &= m^{ed} \bmod n \end{aligned}$$

d è l'inverso di e, modulo $\phi(n)$

$$= m^{1 + k\phi(n)} \bmod n = m(m^{\phi(n)})^k \bmod n$$

Teorema di Eulero $m^{\phi(n)} \equiv 1 \bmod n$

$$= m * 1^k \bmod n = m \quad (m < n)$$

ESEMPIO

$$p = 53 \quad q = 61$$

$$n = p * q = 3233$$

$$\phi(n) = (p-1) * (q-1) = 52 * 60 = 3120$$

$$e = 17$$

$$d = e^{-1} \bmod \phi(n) = 17^{-1} \bmod 3120 = 2753$$

$$k_{\text{pub}} = (3233, 17) \quad k_{\text{priv}} = 2753$$

$$m = 65$$

$$c = 65^{17} \bmod 3233 = 2790$$

$$m = 2790^{2753} \bmod 3233 = 65$$

Easy challenge #2

RSA con $k_{\text{pub}} = (235, 11)$

Decifrage $c = 200$

Sicurezza

legata alla difficoltà di fattorizzare un numero arbitrario molto grande

fattorizzare \Rightarrow forzare RSA ✓

fattorizzare \Leftarrow forzare RSA ?

calcolo della radice in modulo

$$m = \sqrt[e]{c} \bmod n$$

difficile almeno quanto fattorizzare (n è composto)

Calcolare $\phi(n)$ direttamente da n è equivalente a fattorizzare n

Ricavare d direttamente da (n, e) sembra costoso quanto fattorizzare n

Easy challenge #3

In un sistema RSA sono noti i valori

$$n = 18830129$$

$$\phi(n) = 18819060$$

Ricavare p e q (senza fattorizzare n)

Come possiamo fattorizzare un intero n ?

La fattorizzazione è un problema difficile, ma non più come un tempo...

- Da un lato la potenza di calcolo aumenta, dall' altro gli algoritmi di fattorizzazione vengono raffinati
- Esistono algoritmi relativamente veloci di complessità subesponenziale
 - General Number Field Sieve (GNFS) richiede

$$O\left(e^{(c+o(1))(\log n)^{\frac{1}{3}}(\log \log n)^{\frac{2}{3}}}\right)$$

operazioni per fattorizzare un intero n , con $[\log_2 n] + 1$ bit

- Un attacco bruteforce ne richiede $O(n)$, i.e., $O(2^{\log n})$
- Con la potenza di calcolo attuale, usando l' algoritmo GNFS è possibile fattorizzare semiprimi fino a circa 768 bit
- Per interi con una struttura particolare, esistono algoritmi di fattorizzazione particolarmente efficienti
- La fattorizzazione e il logaritmo discreto non sono problemi NP-hard, e si possono risolvere in tempo polinomiale su macchine quantistiche

RSA- scelta dei parametri

Scegliere p e q di almeno 1024 bit.

TDEA, AES (bit della chiave)	<i>RSA</i> e <i>DH</i> (bit del modulo)	<i>ECC</i> (bit dell'ordine)
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	512

RSA- vincoli su p e q

- Scegliere p e q molto grandi (di almeno 1024 bit) per resistere agli attacchi bruteforce
- Sia $p - 1$ che $q - 1$ devono contenere un fattore primo grande (altrimenti n si fattorizza velocemente)
- $\gcd(p-1, q-1)$ deve essere piccolo
conviene scegliere p e q tale che $(p-1)$ e $(q-1)$ siano co-primi
- Non riusare uno dei primi per altri moduli!
 $n_1 = p * q_1$
 $n_2 = p * q_2$
allora $p = \gcd(n_1, n_2)$

RSA- vincoli su p e q

Scegliere p e q non troppo vicini tra loro

$n \sim p^2 \sim q^2$ e quindi anche \sqrt{n} sarà vicino ai primi

basterà quindi un attacco bruteforce che cerca i fattori vicino alla \sqrt{n}

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

$$1) \quad \frac{(p+q)}{2} > \sqrt{n}$$

$$2) \quad \frac{(p+q)^2}{4} - n \text{ è un quadrato perfetto}$$

Si scandiscono gli interi maggiori di \sqrt{n} fino a trovare z t.c.

$$z^2 - n = w^2$$

si suppone

$$z = (p+q)/2 \quad w = (p-q)/2$$

da cui

$$p = z + w \quad q = z - w$$

la vicinanza tra p e q assicura che non dobbiamo allontanarci troppo da \sqrt{n} per trovare p e q

Attacchi con esponenti bassi

- esponenti e e d bassi sono attraenti perché accellerano cifratura o decifrazione
- ovviamente d dovrebbe essere scelto sufficientemente grande, per evitare attacchi bruteforce
- ATTENZIONE
 - se m ed e sono così piccoli che $m^e < n$, allora risulta facile trovare la radice e -esima di c , poiché $c = m^e$ non interviene la riduzione in modulo!

Attacchi a tempo (DH, RSA)

- Si basano sul tempo di esecuzione dell'algoritmo di decifrazione
- **IDEA:** determinare d analizzando il tempo impiegato per decifrare
 - Quando viene eseguita l'esponenziazione modulare, si esegue una moltiplicazione ad ogni iterazione, più un'ulteriore moltiplicazione modulare per ciascun bit uguale a 1 in d
 - **Rimedio:** aggiungere ritardo causale per confondere l'attaccante

Crittografia a chiave pubblica

VANTAGGI

- Se gli utenti di un sistema sono n , il numero complessivo di chiavi (pubbliche e private) è $2n$ anziché $n(n-1)/2$
- Non è richiesto alcuno scambio segreto di chiavi

SVANTAGGI

- Questi sistemi sono molto più lenti di quelli basati sui cifrari simmetrici
- Sono esposti ad attacchi di tipo *chosen plain-text*

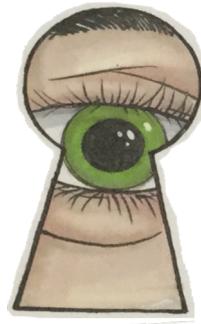
Attacchi *chosen plain-text*

Un crittoanalista può

- scegliere un numero qualsiasi di messaggi in chiaro
 m_1, \dots, m_h
- **cifarli** con la funzione pubblica C e la chiave pubblica k_{pub} del destinatario, ottenendo i crittogrammi
 c_1, \dots, c_h
- quindi può confrontare qualsiasi messaggio cifrato c^* che viaggia verso il destinatario con i crittogrammi in suo possesso

Cifrari ibridi

- Si usa un **cifrario a chiave segreta (AES)** per le **comunicazioni di massa**
- e un **cifrario a chiave pubblica** per **scambiare le chiavi segrete** relative al primo, senza incontri fisici tra gli utenti
- La trasmissione dei **messaggi lunghi** avviene ad **alta velocità**, mentre è **lento lo scambio delle chiavi segrete**
 - le chiavi sono composte al massimo da qualche decina di byte
 - attacco chosen plain-text è risolto se l'informazione cifrata con la chiave pubblica (chiave segreta dell'AES) è scelta in modo da **risultare imprevedibile al crittoanalista**
- La chiave pubblica deve essere estratta da un certificato digitale valido, per evitare attacchi *man-in-the-middle*



domande ?