una possibile funzione unidirezionale

- moltiplicare due interi a n bit è facile (in $\mathcal{O}(n^2)$ con l'algoritmo usuale)
- trovare un primo a n bit, e verificare che è primo, è facile
- fattorizzare un numero a n bit è difficile ($2^{cn^{1/3}}$)
- si può costruire un crittosistema a chiave pubblica basato su questa osservazione?

crittosistema RSA

- Sia $N = pq$, p, q primi. Sia $\mathcal{P} = \mathcal{C} = \mathbb{Z}_N$.
- Lo spazio delle chiavi è

$$\mathcal{K} = \{(N, p, q, d, e) \mid de \equiv 1 \pmod{\phi(N)}\}.$$

- Se $k = (N, p, q, d, e)$ è una chiave, poniamo
- $e_k(x) = x^e \pmod N$
- N e e sono la chiave pubblica
- $d_k(y) = y^d \pmod N$
- p, q, d sono la chiave privata

Problemi facili

1. dato un intero N, vedere se è primo (test di primalità)
2. dati c e M, trovare (c, M); se è 1, calcolare l'inverso di c modulo M (alg. di Euclide – polinomiale)
3. calcolare la f.ne $x \to x^e \pmod N$ (square and multiply)

Problemi difficili

1. dato un intero N, fattorizzarlo
2. dato un intero N, calcolare $\phi(N)$
3. dati N e e, trovare d tale che $(x^e)^d = x \pmod N$

- Se Eve riesce a fattorizzare $N = pq$, ottiene le info private di Bob: può calcolare sia $\phi(N)$ sia $d = e^{-1} \pmod{\phi(N)}$
- dunque violare l'RSA non può essere più difficile che fattorizzare

equivalenza fra i problemi difficili

- violare l'RSA non può essere più difficile che fattorizzare
- conoscere la fattorizzazione $\iff$ calcolare $\phi(N)$:
  - $\Rightarrow$ ovvio
  - $\Leftarrow$ conoscere N $(= pq)$ e $\phi(N)$ $(= (p-1)(q-1) = N - (p+q) + 1)$ vuol dire conoscere la somma $(N - \phi(N) + 1)$ e il prodotto (N) di p,q – quindi conoscere p e q
- i problemi 1 e 2 sono equivalenti: passo dall'uno all'altro in tempo polinomiale

l'esponente di decifratura

- supponiamo che Eve sappia risolvere il pb. 3: dati N e e, riesce a trovare d tale che $(x^e)^d = x \pmod N$
- allora può fattorizzare N
  - c'è un algoritmo polinomiale che fattorizza N oppure fallisce con probabilità minore di $1/2$ – ripetendolo più volte, la probabilità che fallisca diventa molto bassa
- questo fatto non ha solo un interesse teorico
- se sospettiamo che Eve ha trovato d, non basta cambiare l'esponente e – bisogna scegliere anche un nuovo N

modulo comune

- inoltre, si può pensare a una versione dell'RSA con una "trusted authority" che distribuisce le chiavi ai vari utenti
- in cui, per ogni utente, la chiave pubblica sia (N_u, e_u) e quella privata sia soltanto (N_u, d_u)
- dal momento che si può decifrare anche senza conoscere la fattorizzazione di N
- in questa situazione, non si può assegnare lo stesso N a diversi utenti
- per quanto visto ora, questo consentirebbe a ogni utente di violare i messaggi di ogni altro utente

RSA e fattorizzazione

- risolvere uno dei tre "problemi difficili" è computazionalmente equivalente a fattorizzare N
- questo non basta a dire che per violare bisogna fattorizzare N
- si potrebbe riuscire a decrittare senza calcolare l'esponente d

l'equivalenza si ha con il

RSA problem: dati e e N, calcolare le radici e-sime modulo N

non si sa se l'RSA problem è equivalente alla fattorizzazione

RSA – funzione trapdoor unidirezionale

- una trapdoor (botola) one-way function è una funzione unidirezionale che diventa facile da invertire, se si conosce un'informazione supplementare
- la nostra lo è: abbiamo visto che si può invertire la f.ne $f_e : x \to x^e \pmod N$ usando la funzione $f_d : y \to y^d \pmod N$, se $ed \equiv 1 \pmod{\phi(N)}$
- l' informazione supplementare è l'esponente d – difficile da ricavare dalle informazioni pubbliche

fattorizzazione

- la difficoltà di fattorizzare interi grandi non va comunque sopravvalutata
- nel 77, Rivest, Shamir e Adelman hanno proposto una sfida nella rubrica di Martin Gardner su *Scientific American*
- bisognava decifrare un testo cifrato con l'RSA-129 – con chiave pubblica $e = 9007$ e

```
N = 114381625757888867669235779976146
        612010218296721242362562561842935 7
        069352457338978305971235639587050 5
        898907514759929002687954354 1
```

- Rivest stimava che per fattorizzare N ci sarebbero voluti 40 quadrillioni di anni (1 quadrillione $=10^{15}$)
- il testo è stato decrittato nel 1994, da un team coordinato da Derek Atkins, Michael Graff, Arjen Lenstra, Paul Leyland usando il calcolo distribuito – il calcolo è durato sei mesi, ha coinvolto 1600 macchine
- il testo in chiaro era

  the magic words are squeamish ossifrage

- questo successo è dovuto essenzialmente al miglioramento degli algoritmi di fattorizzazione

algoritmi di fattorizzazione

$n =$ lunghezza dell'intero N

- divisioni successive bisogna fare $< \sqrt{N}$ divisioni - è in $\mathcal{O}(2^{n/2})$; se $N = p \cdot q$ e p e q sono vicini, bisogna farle quasi tutte
- fattorizzazione alla Fermat idea - se ho x e $y \in \mathbb{N}$ tali che $N + y^2 = x^2$ – ho la fattorizzazione $N = (x + y)(x - y)$. L'algoritmo controlla per piccoli valori di y se $N + y^2$ è un quadrato perfetto.
- Esempio: sia $N = 5609$:
  $y = 1 \quad N + y^2 = 5610$ non è un quadrato
  $y = 2 \quad N + y^2 = 5613$ non è un quadrato
  $y = 3 \quad N + y^2 = 5618$ non è un quadrato
  $y = 4 \quad N + y^2 = 5625 = 75^2$
  quindi $5609 = 75^2 - 4^2 = (75 + 4)(75 - 4) = 79 \cdot 71$
  funziona meglio delle divisioni successive se N è il prodotto di due primi della stessa grandezza

algoritmi di fattorizzazione

n = lunghezza dell'intero N

- algoritmo di Lehman combina i due metodi precedenti – è in $\mathcal{O}(2^{n/3})$
- ρ di Pollard è in $\mathcal{O}(2^{n/4})$
- $p-1$ di Pollard funziona bene se N ha un divisore primo p tale che $p-1$ ha tutti i fattori primi piccoli
- crivello quadratico di Pomerance – è il più rapido - e il più usato - sotto le 100 cifre decimali
- general number field sieve (crivello generale) il più rapido in assoluto – molto complesso
- poi c'è l'algoritmo di Shor (1994) – in un computer quantistico, la fattorizzazione ha complessità polinomiale

scelta dei primi

per evitare che la fattorizzazione abbia successo, si scelgono primi forti

- p e q devono avere all'incirca la stessa lunghezza
- $p-1$ e $q-1$ devono avere un fattore primo grande altrimenti si può usare l'algoritmo $p-1$ di Pollard
- la lunghezza minima raccomandata per N è 1024 bit (ca 300 cifre decimali) – RSA-1024 – comincia a sembrare un po' poco
- con un N a 2048 bit (ca 600 cifre decimali) si sta tranquilli (?) fino al 2030
- sempre più spesso si arriva a 4096 bit

scelta di e

- si cerca di scegliere e non troppo grande e tale che nella scrittura binaria di e ci siano pochi 1
- e piccolo = cifratura più veloce
  - nell'alg. square and multiply bisogna calcolare pochi quadrati
- meno 1 = cifratura più veloce
  - il numero di 1 è collegato al numero di moltiplicazioni nell'alg. square and multiply
- valori usati sono $e = 3$ (inizialemente, ma dà problemi), $e = 2^{16} + 1 = 65537$
- per il valore $2^{16} + 1$ sono necessarie 17 moltiplicazioni - per un $b < \phi(N)$ casuale, la stima è di circa 1000 moltiplicazioni
- non vuol dire che anche d sarà piccolo

low exponent attack

- d non deve essere piccolo
- se $3d < N^{1/4}$ e $q < p < 2q$ c'è un attacco dovuto a Wiener
- permette di ricavare rapidamente d
- se N ha 1024 bit, d deve avere almeno 256 bit
- la decifratura è computazionalmente pesante - questo è un problema per esempio per le smartcard
- si può usare il teorema cinese del resto per velocizzare la decifratura

RSA – side channel attack

- sono attacchi all'implementazione dell'algoritmo, non all'algoritmo
- uno è il timing attack
- analogia: uno scassinatore che cerca di capire la combinazione di una cassaforte dal tempo impiegato a "girare le rotelle"
- per l'RSA, si cerca di capire qual è la scrittura binaria dell'esponente di decifratura d osservando il tempo impiegato a decifrare
- nell'algoritmo square and multiply, si fa una moltiplicazione solo quando nella scrittura binaria di d c'è un 1
- posso risalire al numero di 1 in d
- un altro è il power monitoring attack

RSA – attacchi

- la funzione di cifratura è moltiplicativa $e_k(x_1 \cdot x_2) = (x_1 \cdot x_2)^e \pmod{N} = x_1^e \cdot x_2^e \pmod{N} = e_k(x_1)e_k(x_2)$
- questo può essere sfruttato in un attacco di tipo chosen ciphertext (Eve sceglie testi cifrati e riesce a ottenere i corrispondenti testi in chiaro)
- Eve ha y, vuole trovare x tale che $x^e = y \pmod{N}$.
- pone $y' = c^e y$, e chiede a Bob di decifrarlo (vedremo che un modo per fare questo è chiedere a Bob di firmare il messaggio)
- Bob può non accorgersi del problema, perché $y' \neq y$
- $d(y') = cy^d \pmod{N}$. Eve calcola $c^{-1} \pmod{N}$, e trova $x = y^d$
- per evitare questo, si introduce padding
- lo schema più usato è l'OAEP (Optimal Asymmetric Encryption Padding)

problemi della chiave pubblica

- la crittografia a PK è lenta - molto più lenta della crittografia simmetrica
- in genere viene usata all'inizio di una sessione fra due utenti per stabilire una chiave da usare con un algoritmo simmetrico (key exchange)
- e per gli schemi di firma digitale

problemi di sicurezza

- bisogna avere un modo per verificare la corrispondenza fra utenti e chiavi
- Alice scrive a Bob usando quella che crede essere la chiave pubblica di Bob
- se invece fosse quella di Eve?
- inoltre, in un CS simmetrico solo Alice e Bob conoscono la chiave
- se Bob riceve un messaggio di Alice e la decifratura del messaggio ha senso, il messaggio proviene certamente da Alice
- in un PKCS, chiunque può scrivere un messaggio cifrato a Bob affermando di essere Alice
- serve una firma digitale