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 (polinomiali)

1. dato un intero N, vedere se è primo
2. dati e e M, trovare (e, M); se è 1, calcolare l'inverso di e modulo M
3. calcolare la f.ne $x \to x^e \pmod N$

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 $(x^e)^d = x \pmod N$

Abbiamo mostrato che i problemi difficili sono equivalenti

i problemi 4,5 e 6 sono quindi equivalenti: ma questo non prova che la sicurezza di RSA è equivalente alla fattorizzazione – c'è la possibilità di trovare la radice e-sima in qualche altro modo...

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 problem e fattorizzazione

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

- se si sa fattorizzare N, si può risolvere l'RSA problem
- c'è un algoritmo polinomiale che calcola radici e-sime modulo p se $(e, p-1) = 1$
- è facile vedere (teorema cinese dei resti) che calcolare la radice e-sima di x mod p e mod q $\Rightarrow$ calcolare la radice e-sima di x mod $N = pq$
- per provare l'equivalenza basta provare una riduzione del tipo
  - algoritmo efficiente per il calcolo relle radici e-sime mod N $\Rightarrow$ algoritmo efficiente per fattorizzare N
- trovare questa riduzione è probabilmente il più importante problema aperto della PKC
- una tale riduzione esiste per $e = 2$ - radici quadrate. Questo si usa nel CS di Rabin

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

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 $e < \phi(N)$ casuale, la stima è di circa 1000 moltiplicazioni
- non vuol dire che anche d sarà piccolo

RSA – broadcast attack

- se l'esponente e è piccolo e usato da molti utenti ho un possibile attacco (broadcast attack)
- se Alice manda lo stesso messaggio m a B_1, B_2, B_3 che usano lo stesso esponente di cifratura $e = 3$ e chiavi N_1, N_2, N_3 prime fra loro
- Eve conosce m^3 mod N_1, mod N_2, mod N_3
- supponiamo che $m < N_1, N_2, N_3$ – allora $m^3 < N_1 \cdot N_2 \cdot N_3$
- applicando il Teorema cinese del resto
- Eve riesce a trovare $x \equiv m^3$ mod $N_1 \cdot N_2 \cdot N_3$
- quindi $x = m^3$ come interi, non modulo qualcosa!
- può fare la normale radice cubica e decifrare

Wiener low exponent attack

- d non deve essere piccolo
- se $\sqrt{6}d < N^{1/4}$ 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
- e alla loro posizione
- un altro è il power monitoring attack