

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**

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 $e = 3$ (inizialmente, 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 $m^3 \bmod N_1, \bmod N_2, \bmod N_3$
- 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 $x \equiv m^3 \bmod N_1 \cdot N_2 \cdot N_3$
- 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**

come utilizzare un PKCS

- affinché un CS sia sicuro, la cifratura deve essere **randomizzata**
- in un cifrario simmetrico (per esempio AES) si sceglia un’opportuna **modalità di funzionamento** (per esempio CBC)
- e in un PKCS?
- sia $f : A \rightarrow B$ una funzione trapdoor one-way (come quella RSA)
- f si utilizza per generare la chiave

come utilizzare un PKCS

- per cifrare si usa $f : A \rightarrow B$ funzione trapdoor one-way
- insieme a un CS simmetrico in modalità sicura (CBC-AES)
- Alice sceglie in modo casuale $a \in A$ e calcola $f(a) = b$ (facile - pubblico)
- Alice cifra il messaggio x con la chiave a e ottiene il CT y (usando AES: $y = e_a(x)$)
- trasmette a Bob la coppia (b, y)

come utilizzare un PKCS

- Bob riceve la coppia (b, y)
- per decifrare deve conoscere $a = f^{-1}(b)$
- è il solo a possedere l'informazione segreta che gli permette di invertire la f
- può ottenere a e decifrare con AES: $x = d_a(y)$
- la randomizzazione "viene" dall'AES
- per maggiore sicurezza, in genere la chiave non è a ma è $h(a)$ dove h è una funzione hash