RSA

Di

Federica Crespi

E' il più celebre tra gli algoritmi a chiave pubblica.
RSA sta per Rivest, Shamir, Adlemann, gli studiosi chelo scoprirono nel 1976.
Questi algoritmi a chiave pubblica vengono detti anche *asimmetrici*, poiché a differenza di quelli a chiave privata (o simmetrici) ,vengono usate due tipi di *chiavi:pubblica e privata.*
Le chiavi pubbliche sono accessibili a tutti gli utenti, mentre quella segreta è nota solo all'utente che l'ha scelta.
Questo algoritmo opera a blocchi e ogni messaggio viene interpretato come numero intero; la sua sicurezza si fonda sulla difficoltà di fattorizzare numeri interi molto grandi.

Come vengono generate le chiavi

1. Un generico utente A sceglie due numeri primi molto grandi **p** e **q**(ca. 150 cifre decimali);
2. A calcola il numero **n = p*q** , e la relativa *funzione di Eulero* **ψ(n) = (p-1)*(q-1);**

3. A sceglie un numero naturale **e** tale che *mcd(e,ψ(n)) = 1;*
4. A calcola il numero **d = e^-1 mod ψ(n),** cioè **d*e ≡ 1 mod ψ(n);**
5. A pubblica la coppia **(n,e)** come *chiave pubblica*;
6. A conserva **d** come *chiave privata.*

Come funziona l'RSA

Se un utente A vuole mandare un messaggio **m** a B o come sequenza di numeri(in questo caso m<n) o come codice ASCII.
1) A cifra **m** con la chiave pubblica dell'utente destinatario B:

c = m^e(B) mod n(B)

2) B riceve il messaggio e lo decifra con la sua chiave privata :

m' = c^d(B) mod n(B)

A questo punto ci si chiede: **m=m' ?**
Per verificarlo applichiamo il *Teorema di Eulero:*
∀ m <n risulta che
m^(e*d) mod n = m' **se e solo se** ***e*d = 1 mod ψ(n)***

Dim.

m' = c^d mod n = m^(e*d) mod n

Questo significa che l'algoritmo per decifrare lavora correttamente se e solo se

m^(e*d) mod n = m ∀ m positivo

Poiché i numeri **e** e **d** sono stati scelti in modo che

e*d = 1 mod ψ(n)

∃ un intero non negativo **k** tale che

e*d = 1+k*ψ(n) = 1+k*(p-1)*(q-1)

Applicando Eulero :

m^(e*d) = m^(1+k*(p-1)*(q-1) ≡ m mod n

cioè **m= m'.**

Forza dell'algoritmo RSA

La forza di questo algoritmo sta nella risposta a queste due domande :

1) si può dedurre la chiave privata dalla chiave pubblica ?
2) è possibile decifrare un messaggio senza la chiave privata ?

Se conosciamo **e** ed n (la mia chiave pubblica), il nostro problema si ricondurrebbe al ricavare **ψ(n)**. Ma anche per calcolare ψ(n) dovremmo conoscere la scomposizione di **n** in fattori primi. Ed è proprio questo il problema che rende difficile forzare l'**RSA:** la **fattorizzazione del numero n.**

Il numero **n** nell' RSA ha una lunghezza di circa 300 cifre decimali.

Attacchi all' RSA

Abbiamo fin qui detto che per forzare l' RSA basterebbe conoscere **n** e **ψ(n)** per poi trovare facilmente i numeri **p** e **q**; infatti conoscendo n e ψ(n) avrò:

n = p*q ψ(n) = (p-1)*(q-1)

Sostituendo **q = n/p** all'equazione di ψ(n) avrò una semplice equazione di secono grado:

ψ(n) = (p-1)*(n/p -1)
(p-1)*(n/p –1) - ψ(n) = 0
n – p - n/p + 1 - ψ(n) = 0

Moltiplico per **(-p):**

np + p^2 + n – p + p*ψ(n) = 0
p^2 – p(n - ψ(n) + 1) + n = 0

Attacco “Common Modulus”

Siano A e B due utenti con **n(A) = n(B)** e con **e(A) ≠ e(B)** tale che **MCD (e(A), e(B)) =1.**
Se viene inviato lo stesso messaggio m ad entrambi gli utenti, allora i corrispondenti testi cifrati saranno:
c1 = m^e(a) mod n(A)
c2 = m^e(B) mod n(B)

Se qualcuno intercetta i messaggi, l’intruso con l’algoritmo di Euclide calcola due numeri **r** e **s** tali che:
r*e(A) + s*e(B) = 1
e quindi si ottiene:
c1r = (m^e(A))^r mod n(A)
c2s = (m^e(B))^s mod n(B)
M = c1r*c2s mod n= (m^(r*e(A) + s*e(B)) mod n

Morale: non distribuire mai lo stesso n in un gruppo di utenti

Attacco “Low Ecryption Exponent”

Questo attacco dipende dalla scelya dell’esponente **e**. Se si sceglie un basso valore di e, la fase di cifratura e firma è molto veloce.
E’ impossibile scegliere **e = 2** poiché non sarebbe mai coprimo con **ψ(n).**
Scegliendo **e = 3** ho vantaggi nella codifica e decodifica.
Infatti avremmo **c = m^3 mod n** (con m^3 >>n) e nel caso in cui **m^3<n,** m sarà una semplice radice cubica compresa nell’intervallo[0; n).
Però questo valore di “e” rende l’RSA vulnerabile ad alcuni attacchi. **Hastad** ha dimostrato che la scelta di un basso valore di **e** rende comunque l’RSA vulnerabile nel caso in cui ci siano dipendenze lineari fra le parti del testo in chiaro.
Se si cifrano **(e*(e+1))/2** messaggi linearmente dipendenti con diverse chiavi pubbliche con lo stesso valore di **e** può esserci un attacco contro il sistema.
Per ovviare a queste debolezze si imbottisce il messaggio di valori random.