UNIVERSITÀ
degli STUDI
di CATANIA

RELAZIONE SU ATTACCHI
A
RSA

Mario Toscano

X81000325

Indice

Introduzione

Cosa è la crittografia?

Per crittografia si intende quella tecnica che permette di “cifrare” un messaggio rendendolo incomprensibile a tutti fuorché al suo destinatario.

In genere esistono due processi principali che vengono applicati nella crittografia: essi sono cifratura e codifica.

La cifratura lavora sulle lettere individuali di un alfabeto, mentre una codifica lavora ad un alto livello semantico, come può essere una parola o una frase.

Ogni sistema di crittografia ha due parti essenziali: un algoritmo (per codificare e decodificare) e una chiave, la quale consiste di informazioni che, combinate con il testo “in chiaro” passato attraverso l’algoritmo, vi darà poi il testo codificato.

In ogni moderno sistema di crittografia si assume che l’algoritmo sia conosciuto dai potenziali “nemici”, quindi la sicurezza di un sistema risiede solo ed esclusivamente nella segretezza della chiave.

Crittografia simmetrica e asimmetrica

Come abbiamo visto mantenere la chiave segreta è molto importante. Senza assumere tale assioma non riusciremo a concludere nulla. La chiave però deve essere conosciuta anche dal ricevente del messaggio criptato, in quanto, senza di essa, il messaggio sarebbe privo di senso, illeggibile.
Inizia qui il problema distribuzione delle chiavi.
Innanzi tutto esistono due famiglie di codici crittografici : quelli a crittografia simmetrica e quelli a crittografia asimmetrica.

Crittografia Simmetrica

In questa famiglia crittografica Mittente e Destinatario usano la stessa chiave, o meglio l'algoritmo codifica e decodifica il messaggio con la stessa chiave privata o con chiavi diverse che sono in diretta relazione tra di loro.

Alcuni algoritmi che fanno parte di questa famiglia sono DES, IDEA, 3DES e RC2. Il problema di questi algoritmi e proprio lo scambio della chiave che entrambi, mittente e destinatario, devono conoscere, ed anche l'alto numero di chiavi che deve mantenere e generare per ogni richiesta(si pensi ad un sistema bancario con milioni di utenti). Inoltre è bene generare una chiave lunga, infatti la robustezza degli algoritmi simmetrici è legata alla lunghezza della chiave.

Crittografia asimmetrica

Nella crittografia asimmetrica vengono usate due chiavi: la chiave di cifratura e la chiave di decifratura. La chiave di cifratura viene resa pubblica e prende il nome di chiave pubblica mentre la chiave di decifratura deve essere tenuta privata e prende il nome di chiave privata.
Il mittente (M) cifra il messaggio con la chiave pubblica del destinatario (D) e manda il messaggio cifrato a M. Una vola ricevuto il messaggio, M lo decifra con la sua chiave privata e legge il testo "in chiaro".
Questi algoritmi crittografici consentono e risolvono il problema della distribuzione delle chiavi degli algoritmi crittografici simmetrici.
Inoltre tali algoritmi vengono usati anche per "firmare" un messaggio, ovvero garantire l'integrità dei dati. Il mittente (M) cifra il messaggio con la sua chiave privata e lo invia al destinatario (D). D riceve il messaggio e lo decifra con la chiave pubblica di M. Se il messaggio è leggibile allora è stato inviato da M, altrimenti qualcuno ha modificato il pacchetto.
Esistono anche combinazioni di cifratura che possono garantire sia integrità che segretezza ed è possibile avere anche autenticazione.

RSA

RSA è un algoritmo di crittografia asimmetrica, inventato nel 1977 da Ronald Rivest, Adi Shamir e Leonard Adleman utilizzabile per cifrare informazioni o per firmarle.

Cenni storici

È nel 1978 che tale algoritmo trova la sua prima applicazione reale. L'algoritmo da loro inventato non è sicuro da un punto di vista matematico teorico, in quanto esiste la possibilità che tramite la conoscenza della chiave pubblica si possa decifrare un

messaggio; ma l'enorme mole di calcoli e l'enorme dispendio in termini di tempo necessario per trovare la soluzione fanno di questo algoritmo un sistema di affidabilità pressoché assoluta.

Nel 1983 Ronald Rivest, Adi Shamir e Leonard Adleman brevettano l'algoritmo negli Stati uniti e fondano la RSA Data Security in seguito acquisita dalla Security Dynamics che vendette l'algoritmo a società come Netscape, Microsoft e altri.

L'algoritmo RSA costituisce la base dei sistemi crittografici su cui si fondano i sistemi di sicurezza informatici utilizzati sulla rete Internet per autenticare gli utenti.

Algoritmo RSA

RSA è basato sull'elevata complessità computazionale della fattorizzazione in numeri primi di un numero composto dal prodotto di due numeri primi. Cioè non è impossibile, infatti esistono diversi metodi per fattorizzare un numero, ma diventa computazionalmente impossibile man mano che il numero da fattorizzare diviene sempre più grande

Il funzionamento base dell'algoritmo è il seguente:

1. Si scelgano due numeri primi che siano abbastanza grandi da garantire la sicurezza dell'algoritmo; Chiameremo tali numeri **p** e **q**;
2. Si calcola il loro prodotto di **p** e **q**, che prende il nome di modulo; Tale modulo sarà indicato con **n**;
3. Si calcola **$\varphi(n)=(p-1)(q-1)$**;
4. Si sceglie poi un numero **e** tale che esso sia coprimo con **$\varphi(n)$** più piccolo di esso;
5. Si calcola il numero **d** tale **$e*d\equiv 1 \bmod (\varphi(n))$;**

Il numero **e** prende il nome di esponente pubblico mentre il numero **d** prende il nome di esponente privato;

La **chiave pubblica** è (n,e) e **la chiave privata** è (n,d);

Un messaggio viene cifrato tramite l'operazione **$c = m^e \pmod n$**;

Un messaggio cifrato **c** viene decifrato tramite l'operazione **$m = c^d \pmod n$**;

Attacco a RSA

Come abbiamo già detto, RSA, dal punto di vista matematicamente teorico, può essere attaccato attraverso la sua chiave pubblica. Infatti è possibile fattorizzare N, ovvero il modulo dell'algoritmo, e trovare p e q. E' per questo che si usa N di almeno

1024 bit in quanto la mole di calcoli e le risorse di tempo necessarie rendono la fatorizzazione difficile. In realtà esistono anche altri tipi di attacchi di RSA.

Metodo di Fermat

Come abbiamo detto più volte esistono diversi algoritmi per fattorizzare un numero. In base alla dimensione della numero alcuni algoritmi sono più efficienti di altri. Quello preso in considerazione sarà Fermat.
Il metodo di Fermat si basa sulla rappresentazione di un numero come una differenza di quadrati. Esso è più efficace quando esistono due fattori del numero vicini tra loro.

L'algoritmo per la fattorizzazione è il seguente:

1. Sia **n** un intero dispari.
2. $\boldsymbol{a = \lceil \sqrt{n} \rceil}$
3. Ripeti:
    1. $\boldsymbol{b_2 = a^2 - n}$
    2. Se $\mathbf{b_2}$ non è un quadrato perfetto allora **a=a+1;**
4. fin quando $\mathbf{b_2}$ non è un quadrato perfetto;
5. $\boldsymbol{b = \sqrt{b_2};}$
6. $\boldsymbol{n = (a - b)(a + b);}$

Demo

Tramite il suo metodo mostrerò come attaccare RSA con una chiave di 64 bit. Per far usiamo un programma in python che implementa l'algoritmo RSA per la generazione delle chiavi.

A tale programma passiamo la dimensione che la chiave deve avere.

```
C:\Users\tosca\Documents\Università\RSA>python RSA_generate.py
Insert the key dimension...
64
Making key files...
Generating p prime...
Generating e that is relatively prime to (p-1)*(q-1)...
Calculating d that is mod inverse of e...
Public key: (10463903239541082367, 3145277701)
Private key: (10463903239541082367, 1934955345245464149)

The public key is a 20 and a 10 digit number.
Writing public key to file al_sweigart_pubkey.txt...

The private key is a 20 and a 10 digit number.
Writing private key to file al_sweigart_privkey.txt...
Key files made.
```

Veranno, oltre che visualizzati a schermo, salvati in due file diversi la chiave pubblica (composta da N ed e) e la chiave privata (composta da N e d). Tali file si chiameranno rispettivamente “al_sweigart_pubkey.txt” e “al_sweigart_privkey.txt”

Simuleremo quindi un attacco in cui siamo a conoscenza di N, il numero di 64 bit multiplo di p e q, e dell’esponente pubblico e. Tale conoscenza ha senso in quanto sia N che e compongono la chiave pubblica.

Successivamente usiamo un algoritmo che implementa il metodo di Fermat. Passiamo il numero N che conosciamo al programma:

```
C:\Users\tosca\Documents\Università\RSA>python Fermat.py
Enter the number to factor of form (p*q):
10463903239541082367
```

Lo script impiegherà del tempo per risolvere la fattorizzazione per i motivi spiegati prima.

Una vola finito avremo sia p che q, ed inserendo l’esponente pubblico e otteniamo anche d. Adesso abbiamo tutti gli elementi per decifrare un messaggio.

Algoritmo rho di Pollard

L’algoritmo rho di Pollard è un algoritmo di fattorizzazione di numeri interi, basato sull’aritmetica modulare. E’ stato ideato da John Pollard nel 1975.

L’algoritmo è il seguente:

1. Si scelgono **x** e **c** randomici. Si pone $y = x\ e\ f(x) = x^2 + c$
2. Pongo $p = 1$
3. Ripeti finché $p == 1$
    a. $x = f(x)\ mod\ n$
    b. $y = f(f(y))\ mod\ n$
    c. $p = gcd(|x - y|, n)$
4. Se p==n ripeti dal punto 1 altrimenti p divide n.

Demo

Ho implementato questo algoritmo creando uno script python. Lo script ci chiederà il numero da fattorizzare. Tale numero lo possiamo creare tramite lo script RSA_generate.py che ho illustro nella sezione precedente.

```
C:\Users\tosca\Documents\Università\RSA>python Rho.py
Insert n
8985192004196054461
```

Una volta inserito otteniamo p e q (L'algoritmo Rho di Pollard ci da un solo divisore di N ma l'altro lo otteniamo facilmente dividendo N per il divisore).

```
C:\Users\tosca\Documents\Università\RSA>python Rho.py
Insert n
8985192004196054461
p =  3112936033
q =  2886404317
Time:  0.0628516674041748
```

Abbiamo ottenuto sia p che q, grazie a questi possiamo ottenere d inserendo l'esponente pubblico e.

```
C:\Users\tosca\Documents\Università\RSA>python Rho.py
Insert n
8985192004196054461
p =  3112936033
q =  2886404317
Time:  0.0628516674041748
Insert the public exponent
2809829555
d=  1187251561409827067
```

e con questi elementi possiamo decifrare i messaggi o fingerci anche il mittente o il destinatario.

Conclusioni

Siamo riusciti a rompere RSA che usa una chiave di 64 bit. Notiamo come l'algoritmo Rho di Pollard sia notevolmente più veloce rispetto al metodo di Fermat. Ma comunque non è così veloce con chiavi di dimensione maggiore. E' solo con l'assunzione che la chiave sia di 64 bit che siamo riusciti a fattorizzare N, proprio perché, per arginare il problema della fattorizzazione, si usano chiavi di almeno 1024 bit, che rendono computazionalmente impossibile fattorizzare il numero. Si stanno cercando delle soluzioni computazionali per fattorizzare numeri di queste dimensioni come l'algoritmo di Shor che trova una reale implementazione in computer quantistici che non sono ancora una realtà affermata. Inoltre si cercano altri tipi di attacchi ad RSA come quello messo in pratica da Weiner mediante frazioni continue. La conclusione è che RSA 64 è attaccabile ma il suo utilizzo quasi nullo rende inutile l'attacco.

Glossario

Testo “in chiaro” : le parole, i caratteri o le lettere del messaggio originale in forma comprensibile.

Integrità : l’informazione non sia modificata da entità non autorizzate.

Segretezza : L’informazione non sia rilasciata ad entità che non sono autorizzate a conoscerla

Script : in informatica, designa un tipo particolare di programma, scritto in una particolare classe di linguaggi di programmazione, detti linguaggi di scripting.

Referenze

Script per generare la key:
https://inventwithpython.com/cracking/chapter23.html

Cryptomath Module:

https://inventwithpython.com/cryptomath.py

Metodi di Attacco al crittosistema RSA:

https://www.unisalento.it/documents/20152/851927/Metodi+di+attacco+al+Crittosistema+RSA.pdf/f351ee9e-d8a5-c1b4-6bc1-b2651a415271?version=1.0&download=true

Euclyd.py Module:

https://github.com/ralphleon/Python-Algorithms/blob/master/Cryptology/euclid.py