LA MATEMATICA DELLA CRIPTOGRAFIA: COPPIE DI NUMERI PRIMI PER L'RSA

di Mario Marobin

PREMESSA

Nel 1978 un matematico e un informatico del MIT, insieme ad un matematico israeliano ospite dell'istituto, pubblicarono un nuovo sistema di crittografia denominato RSA dalle iniziali dei loro nomi ([1]). La novità era legata all'uso di chiavi separate per la codifica e la decodifica, alla cifratura con uso esclusivo di numeri interi e alla sicurezza intrinseca basata sulla complessità computazionale della fattorizzazione di numeri interi molto grandi.

Il tutto, unito alla semplicità strutturale del metodo, condusse in breve tempo al successo commerciale del sistema ideato. In particolare l'uso di chiavi separate divenne successivamente il metodo più usato per crittografare i numeri delle carte di credito nei commerci su internet. L'implementazione di una applicazione del metodo RSA a scopo didattico su un normale PC mostra da subito notevoli limiti che illustreremo con lo scopo di evidenziare le barriere da superare per ottenere in prima battuta un risultato accettabile.

IL SISTEMA RSA

Il destinatario di un messaggio «m» crea una chiave pubblica PUB e una chiave privata PRI con quattro numeri interi p, q, e, d. I numeri p e q devono essere due numeri primi distinti. I numeri e (encrypt) e d (decrypt) completano rispettivamente la chiave pubblica per la codifica e quella privata per la decodifica.

$$\begin{aligned} \mathrm{PUB} &= (n,e) \\ \mathrm{PRI} &= (n,d) \\ n &= pq \end{aligned}$$

Il messaggio «m» è un numero intero come ad esempio la codifica ASCII di un carattere stampabile. Il destinatario invia la chiave pubblica al mittente e tiene la chiave privata segreta per se stesso. Se «m» è il messaggio da scambiare allora il mittente spedisce il codice «z» decodificabile solo con la chiave privata del destinatario. Il sistema crittografico è definito dalle relazioni che seguono.

([1]) Ronald Rivest, Adi Shamir, Leonard Adleman.

$$z = m^e \text{ Mod } n \qquad (1)$$
$$m = z^d \text{ Mod } n \qquad (2)$$

con

e: coprimo di $\varphi(n)$ and $< \varphi(n)$ ovvero $\mathrm{MCD}[e,\varphi(n)] = 1$ (3)

d: $ed \equiv 1$ Mod $\varphi(n)$ ovvero $\mathrm{Resto}[ed/\varphi(n)] = 1$ (4)

$\varphi(n)=(p-1)(q-1)$ funzione di Eulero (5)

Ampia letteratura descrive in dettaglio le giustificazioni delle relazioni esposte basate sul piccolo teorema di Fermat e le modifiche aggiunte successivamente da Eulero. Si sottolinea che nonostante le due chiavi siano fra loro dipendenti risulta praticamente impossibile risalire alla chiave privata di decodifica dalla conoscenza di quella pubblica. Questo perché un estraneo può risalire alla chiave privata solo se riesce a fattorizzare il numero n che è un problema di complessità computazionale elevata se n è sufficientemente grande.

In pratica oggi con l'uso dei computer è molto facile e veloce generare coppie di numeri primi p q molto grandi mentre la fattorizzazione del loro prodotto pq è un problema di complessità computazionale di classe NP, ovvero risolvibile solo in tempi esponenziali rispetto la grandezza del dato. La sicurezza del metodo RSA si basa pertanto su questo fatto e a questo proposito citiamo un passo dal discorso di H. W. Lenstra jr. al Congresso Internazionale di Matematica, tenutosi a Berkeley (USA) nel 1986: «*Supponiamo di aver dimostrato che due numeri p e q, di circa 100 cifre, sono primi; con gli odierni test di primalità la cosa è molto semplice. Supponiamo inoltre che per sbaglio i numeri p e q vadano perduti nella spazzatura e che ci rimanga invece il loro prodotto p·q. Come fare per risalire a p e q? Deve essere sentito come una sconfitta della matematica il fatto che la cosa più promettente che possiamo fare sia di andare a cercare nell'immondizia o di provare con tecniche mnemo-ipnotiche …*»

La fattorizzazione di interi grandi è progredita rapidamente mediante l'utilizzo di hardware dedicati. Le chiavi a 512 bit sono ormai fattorizzabili in poche ore mentre quelle a 1024 bit, attualmente in uso, potrebbero essere fattorizzate in un solo anno di tempo al costo di un milione di dollari, costo sostenibile per qualunque grande organizzazione interessata a trafugare segreti strategici [1] [2]

LA SFIDA RSA

La «RSA Factoring Challenge» è stata una sfida proposta dai laboratori RSA (2) dal 1991 al 2007 per incoraggiare la ricerca nel campo della teoria computazionale dei numeri e, in particolare, nella fattorizzazione di grandi numeri naturali. Venne pubblicata una lista di semiprimi, numeri che hanno esattamente due fattori primi, co-

(2) I laboratori RSA fanno parte della società fondata nel 1982 dagli ideatori del metodo RSA. La società si occupa si sicurezza informatica con oltre 1500 dipendenti e dal 2016 fa parte del gruppo DELL EMC.

nosciuti come numeri RSA. Il più piccolo di questi, un numero con 100 cifre decimali chiamato RSA-100, fu fattorizzato in pochi giorni, ma molti dei numeri più grossi non sono stati ancora fattorizzati. Vediamo in dettaglio alcune di queste sfide.

RSA-110 fattorizzato nell'aprile 1992 in circa un mese, RSA-120 fattorizzato nel giugno 1993 in circa tre mesi.

RSA-129 fattorizzato nell'aprile 1994 in otto mesi con centinaia di computer in rete. A ciascun computer vennero assegnati moduli differenti con cui setacciare i numeri primi. Si chiedeva così a internet, che in teoria avrebbe dovuto proteggere quei codici, di contribuire a vincere la sfida posta [3]. La setacciatura dei dati richiese otto mesi di computazione con 600 volontari di 20 paesi. Il progetto includeva il MIT, la Oxford University, la Iowa State University e altri. L'Energia di calcolo messa in gioco è stata valutata in 5000 MIPS-anno [6]. Un MIPS-anno è la capacità di calcolo di una macchina che esegue 1 milione di istruzioni al secondo in funzione per un anno.

RSA-180 fattorizzato nel maggio 2010. Sono stati usati strumenti open source per la fattorizzazione su 3 PC Intel Core i7 e su supercomputer SKIF MSU «Chebyshov». L'intero processo ha impiegato tempi comparabili su entrambe le piattaforme sebbene la fase di setacciatura fosse più efficace sul supercomputer a causa di un maggior numero di processori mentre le altre fasi erano più efficienti sul PC Intel Core i7. Questo significa che oggi per circa $3000 si può acquistare una potenza computazionale paragonabile al piccolo Supercomputer e fattorizzare numeri di 180 cifre decimali in 3 mesi [4].

RSA-200 fattorizzato nel maggio 2005 con uno sforzo pari a quello di 75 anni di lavoro di un PC 2.2 GHz.

RSA-768 fattorizzato il 12 dicembre 2009. Consta di 232 cifre decimali, 768 bits. Il calcolo ha richiesto più di 10^{20} operazioni equivalenti a circa 2000 anni di elaborazione su un singolo core 2.2 GHz AMD Opteron. Lo sforzo complessivo è stato comunque sufficientemente basso e moduli di 768 bit RSA non possono essere utilizzati per la protezione di dati anche di corto termine [5].

RSA-1024 ha una lunghezza di 309 cifre decimali e non è ancora stato fattorizzato. L'eventuale fattorizzazione avrebbe importanti conseguenze sulla sicurezza di molti messaggi criptati con l'algoritmo di crittografia a chiave pubblica RSA che attualmente usa chiavi a 1024 bit.

RSA-2048 ha una lunghezza di 617 cifre decimali e non è ancora stato fattorizzato. È il più grande numero RSA e probabilmente non sarà fattorizzato per ancora molti anni, a meno di considerevoli progressi nella velocità di fattorizzazione dei numeri interi o nella potenza computazionale. A titolo di esempio riportiamo i numeri delle due sfide RSA-129 e RSA-2048.

```
RSA-129= 114381625757888867669235779976146612010218296721
242362562561842935706935245733897830597123563958705058989
075147599290026879543541

RSA-2048= 25195908475657893494027183240048398571429282126
204032027777137836043662020707595556264018525880784406918
290641249515082189298559149176184502808489120072844992687
392807287776735971418347270261896375014971824691165077613
379859095700097330459748808428401797429100642458691817195
118746121515172654632282216869987549182422433637259085141
865462043576798423387184774447920739934236584823824281198
163815010674810451660377306056201619676256133844143603833
904414952634432190114657544454178424020924616515723350778
707749817125772467962926386356373289912154831438167899885
040445364023527381951
378636564391212010397122822120720357
```

PROGRAMMAZIONE DEL METODO CRITTOGRAFICO RSA SU PC

La costruzione delle chiavi RSA e la codifica/decodifica di messaggi secondo le relazioni da (1) a (5) fa uso di numeri interi positivi. Sui normali PC non dotati di software specifici i numeri interi sono rappresentati da variabili a 32 bit che denominiamo «Long». Poiché 1 bit è riservato al segno il valore massimo di un «Long» è 2^{31}=2.147.483.648 che non è un numero tanto grande.

Come esempio la semplice codifica secondo la relazione (1) di un messaggio m=90 (codice ASCII della lettera «Z») con un esponente piccolissimo come 5 porta a eccedere ampiamente il limite di 2^{31}. Fortunatamente la crittografia RSA fa riferimento all'algebra modulare che, definito un modulo n, confina i numeri risultanti di operazioni algebriche nell'intervallo limitato $(-n,n)$.

Per le relazioni (1) e (2) non si deve fare l'errore di considerare la legittima operazione «Mod» come il resto della divisione fra una potenza e un numero «n» ma di usare l'equivalente metodo dell'elevamento a potenza modulare realizzabile con uno dei diversi algoritmi ideati allo scopo. A chiarimento prendiamo l'algoritmo «Right to Left» che illustriamo con i passi che seguono. Sia R = B^E Mod Z. Scelti ad esempio B=3, E=15, Z=19 abbiamo:

3^{15} Mod 19 = 12 perché $3^{15}/19$=755.205,6315789474 e 0,6315789474·19=12
$3^{15}=3^{1+2+4+8}=3^1 3^2 3^4 3^8$= 3·9·81·6561=14.348.907

L'algoritmo «Right To Left» è mostrato con il pseudo codice riportato nelle colonne A e B della tabella riportata. L'esponente E deve essere convertito in un numero binario con E(i) i singoli bit 0/1 e t+1 il numero di bits. L'algoritmo scandisce l'esponente binario da destra verso sinistra secondo i pesi crescenti dei singoli bit con E(0)=1, E(1)=2, E(2)=4, E(3)=8 e così via. Nella colonna di **sinistra A** vediamo il risultato che si otterrebbe moltiplicando per se stessa E volte la base B; ad ogni passo i valori di B e R giustamente crescono fino al risultato finale. Nella colonna di B di **destra** l'algoritmo sostituisce le moltiplicazioni con moltiplicazioni modulari; ad ogni passo i valori di B e R cambiano senza mai superare il valore del modulo Z. L'algoritmo fa uso della proprietà distributiva dell'algebra modulare

$$(a\cdot b) \text{ Mod } z = [(a \text{ Mod } z)\cdot(b \text{ Mod } z)] \text{ Mod } z.$$

<table>
<tr><th>A</th><th>B</th></tr>
<tr><td><pre>R=1
If E > 0 Then
   If E(0) = 1 Then R=B
   For i = 1 To t
      B= B*B
      If E(i) = 1 Then R = R*B
   Next i
End If
Return R</pre></td><td><pre>R=1
If E > 0 Then
   If E(0) = 1 Then R=B
   For i = 1 To t
      B= B*B Mod Z
      If E(i) = 1 Then R = R*B Mod Z
   Next i
End If
Return R</pre></td></tr>
<tr><td><pre>   Potenza di 2   Accumulo potenze

i=0 B=3   R=3
i=1 B=B·B=B²=9   R=3·9=27
i=2 B=B·B=B²·B²=B⁴=81   R=27·81=2187
i=3 B=B·B=B⁴·B⁴=B⁸=6561 R=2187·6561=14348907</pre></td><td><pre>   Potenza di 2   Accumulo potenze

i=0 B=3   R=3
i=1 B=3·3 mod 19=9   R=3·9 mod 19=8
i=2 B=9·9 mod 19=5   R=8·5 mod 19=2
i=3 B=5·5 mod 19=6   R=2·6 mod 19=12</pre></td></tr>
</table>

La realizzazione di una APP, secondo le descrizioni fatte, con l'utilizzo di variabili «Long» consente di operare con numeri primi (*p*,*q*) tali che il loro prodotto $n=pq$ non superi valori compresi fra $45\cdot10^3$ e $55\cdot10^3$. Con numeri più grandi il prodotto B*B o R*B, in una delle fasi dell'algoritmo «Right To Left», va in «overflow» perché si supera il limite 2^{31}. È indicato un intervallo di valori perché il risultato dipende dalla scelta dei numeri primi *(p,q)* che possono essere uno piccolo e uno grande oppure entrambi dello stesso ordine di grandezza. Coppie di numeri primi (*p*,*q*) come (*211*,*223*) e (*599*,*73*) forniscono chiavi e codifiche/decodifiche corrette mentre coppie come (*239*,*241*) e (*661*,*73*) segnalano il messaggio di «overflow» di calcolo. Nelle condizioni limite può intervenire anche la scelta delle chiavi perché la relazione (3) fornisce un lungo elenco di valori «*e*»([3]). Scelto casualmente uno di questi numeri si trovano molteplici valori «*d*» in base alla relazione (4). In altre parole per ogni singola chiave pubblica si hanno molteplici chiavi private, tutte intercambiabili fra loro.

Per operare con numeri primi (*p*,*q*) più grandi si può passare dai «Long» alle variabili in virgola mobile (double-precision floating-point IEEE 754) che operano

([3]) La funzione di Eulero (3) rappresenta il numero di interi fra *1* e *n* che non hanno divisori in comune con *n* e vale *(p-1)(q-1)*. Come esempio con φ(n)=(211-1)(223-1) si trovano 10.368 coprimi di φ(n); scelto *e*=551 si trovano 409 chiavi *d*.

da $-1.79769313486231 \cdot 10^{308}$ a $1.79769313486232 \cdot 10^{308}$ con 15 cifre significative. Queste variabili sono comunemente denominate «Double» e vanno utilizzate nel nostro caso fintanto che la parte decimale della variabile rimane nulla e il numero delle cifre significative utilizzate rimane minore di 16. In altre parole i limiti superiori delle variabili riferite sono:

2^{31}	= 2.147.483.648	per un Long inteso come numero naturale
$10^{16}-1$	= 999.999.999.999.999	per un Double inteso come numero naturale

La APP con variabili «Double» è in grado di operare correttamente con numeri primi (p,q) tali che il loro prodotto $n=pq$ non superi valori compresi fra $70 \cdot 10^6$ e $90 \cdot 10^6$. Con numeri primi più grandi il prodotto B*B o R*B, in una delle fasi dell'algoritmo «Right To Left», supera il valore $10^{16}-1$ con trascinamento di zeri. A Chiarimento vale il seguente esempio facilmente verificabile su PC:

38^{10}=6.278.211.847.988.224	Excel fornisce 6.278.211.847.988.220
38^{11}=238.572.050.223.552.512	Excel fornisce 238.572.050.223.553.000

È chiaro che quando il prodotto B*B o R*B nel calcolo di una potenza modulare supera le quindici cifre significative si perviene a risultati sbagliati senza che il sistema segnali errori perché siamo sempre nel range di definizione dei «Double» che hanno il limite superiore pari a $1.79769313486232 \cdot 10^{308}$. Coppie di numeri primi (p,q) come (*8.317,8.423*) e (*50.047,1.409*) forniscono chiavi e codifiche/decodifiche corrette mentre coppie come (*9967,9973*) e (*50.047,1811*) forniscono risultati incongruenti. Anche se il sistema non segnala errore, rispetto l'avviso di overflow dei «Long», il destinatario che genera le chiavi può verificare la congruenza dei dati prima di pubblicare la chiave pubblica.

L'operatore «Mod» funziona solo con operandi di tipo «Long». Con variabili di tipo «Double» si deve pertanto costruire una opportuna funzione che chiamiamo convenzionalmente «RestoFun» e che opera secondo il pseudocodice che segue.

```
Sub RestoFun(Dividendo, Divisore, Resto As Double)
      Dim u, rapporto, quoziente As Double
      rapporto = Dividendo / Divisore
      quoziente = Fix(rapporto)
      u = quoziente * Divisore
      Resto = Dividendo - u
End Sub
```

L'operatore «Fix» che appare rimuove la parte frazionaria e restituisce la parte intera del numero reale. Si rammenta che le variabili «Double» operano con 15 cifre significative. Nel nostro caso il superamento delle 15 cifre significative conduce ad approssimazioni non accettabili. Valgono i seguenti esempi:

Resto $(7+10^{15})/3 = 2$ la funzione RestoFun ritorna il valore corretto Resto=2
Resto $(7+10^{16})/3 = 2$ la funzione RestoFun ritorna il valore sbagliato Resto=0

Il nuovo operatore «RestoFun» sostituisce l'operatore «Mod» nella generazione della chiave privata (4) e nelle codifiche/decodifiche via «Right To Left».

CONCLUSIONI

Con uno qualsiasi dei linguaggi di programmazione disponibili per PC, come Microsoft Visual Basic for Application, C/C++, Pascal e simili, risulta abbastanza semplice generare le chiavi (3) e (4) e trasformare un messaggio composto da una stringa alfanumerica in una tabella di numeri. A ciascun elemento di questa tabella si applicano poi le trasformazioni di codifica (1) e di decodifica (2). Si termina con la trasformazione della tabella risultante in una stringa alfanumerica che dovrebbe corrispondere al messaggio originale. Ovviamente si deve far uso delle descritte funzioni «RestoFun» e «Right To Left» per arrivare ad un risultato accettabile. Tutto questo potrebbe sicuramente essere un interessante esercizio didattico per giovani studenti di informatica.

Per una applicazione del metodo RSA completa, che supera quindi il limite imposto dalle 15 cifre significative delle variabili «Double», si devono costruire degli opportuni operatori di Moltiplicazione e Divisione dove gli operandi non sono più variabili tipo «Long» o «Double» ma stringhe di lunghezza variabile. I nuovi operatori vanno inseriti nelle già descritte funzioni «RestoFun» e «Right To Left». Il problema così posto è molto più complesso ma risolvibile: esercizio didattico per i più tenaci. Quanto qui descritto è stato realizzato su notebook Dell con processore 64 bit Intel Core i-7 2.4 GHz e linguaggio Visual Basic for Application. Software specifici come Mathematica della Wolfram Research o linguaggi di programmazione più evoluti come Python della Python Software Foundation consentono lo sviluppo di programmi per la crittografia RSA senza nessuno dei limiti descritti in questa memoria.

Mario Marobin
Federmanager Vicenza - socio
mmarobin@alice.it

Bibliografia

[1] *On the Cost of Factoring RSA-1024* - A. Shamir, E. Tromer 2003.
[2] *A realizable hardware sieving device for factoring 1024-bit* - Franke et alt., SHARK 2005.
[3] *L'enigma dei numeri primi* - cap.10 - Marcus du Sautoy - Ed. Rizzoli 2004.
[4] *Factorization of RSA-180 S.* - A. Danilov, I. A. Popovyan - Moscow State University, Russia May 9, 2010.
[5] *Factorization of a 768-bit RSA modulus version 1.4, February 18, 2010* - T. Kleinjung, K. Aoki, J. Franke, A. K. Lenstra, E. Thomé, J. W. Bos, P. Gaudry, A. Kruppa, P. L. Montgomery, D. Arne Osvik, H. Riele, A. Timofeev, P. Zimmermann - EPFL IC LACAL Laboratory for Cryptologic Algorithms, Lausanne, Switzerland - NTT Nippon Telegrapf & Telephone Tokyo, Japan - University of Bonn, Department of Mathematics, Germany - INRIA CNRS LORIA Institut National Recherche Informatique Automatique, Nancy Cedex, France - MICROSOFT Research, Redmond, USA - CWI Centrum Weskunde & Informatica, Amsterdam, The Netherlands.
[6] http://www.mit.edu/people/warlord/RSA129-announce.txt