Quadratic Sieve: introduzione e primo impatto sugli standard crittografici

Leonardo Errati

19 ottobre 2021

Indice

1 Il crittosistema RSA **2**

2 Fattorizzare numeri interi **5**

3 Il crivello quadratico: introduzione **6**

4 Il crivello quadratico **8**

5 Applicazioni di QS **11**

Bibliografia **13**

Contenuti

Discuteremo dell'alrogitmo di fattorizzazione QS (Quadratic Sieve) e del suo ruolo nel contesto storico crittografico di fine secolo, utilizzando il crittosistema RSA come base didattica ed intuitiva per tale utilizzo di QS. Dopo un approccio intuitivo ad RSA e QS, potremo procedere da una prospettiva teorica ed in seguito analizzare le conseguenze di quanto visto.

1 Il crittosistema RSA

Il sistema crittografico RSA è stato brevettato nel 1977; è basato su proprietà elementari di algebra, quali l'aritmetica modulare, e soprattutto sulla difficoltà nella fattorizzazione di numeri interi di grandi dimensioni.
Essendo un sistema a chiave asimmetrica, è composto da due chiavi: la *chiave pubblica* può essere divulgata ed è necessaria per criptare, la *chiave privata* deve essere tenuta segreta ed è utilizzata per decriptare. Più in particolare, se Alice (mittente) vuole inviare un messaggio a Bob (destinatario), i passi necessari che devono seguire sono:

(i) Alice e Bob scelgono due numeri primi distinti molto grandi p e q, come vedremo la violabilità del sistema dipende dalla loro lunghezza

(ii) Alice e Bob calcolano $n = pq$ e $\varphi(n)$, dove φ è la funzione di Eulero, quindi

$$\varphi(n) = (p-1)(q-1)$$

(iii) Alice e Bob scelgono un intero $e \leq \varphi(n)$ tale che $\gcd(e, \varphi(n)) = 1$, proseguono poi calcolando un altro intero d tale che

$$ed \equiv 1 \mod \varphi(n) \tag{1}$$

(iv) la chiave pubblica sarà (n, e), per criptare un messaggio numerico (plaintext) m è sufficiente calcolare

$$c \coloneqq E(m) = m^e \mod n$$

ottenendo il ciphertext c

(v) la chiave privata sarà invece (n, d), per decriptare il ciphertext c basta calcolare

$$m \coloneqq D(c) = c^d \mod n$$

E è detta essere la *encryption function*, D la *decryption function*. Ovviamente, poter criptare solo un messaggio numerico non è restrittivo. Notiamo che l'algoritmo funziona correttamente, perché

$$D\left(E(m)\right) = m^{ed} = m \mod n$$

dove l'ultimo passaggio si basa sui seguenti teoremi.

Teorema 1.1 (Teorema di Fermat)**.** *Se p è un primo, allora per ogni a intero coprimo con p si ha*

$$a^{p-1} \equiv 1 \mod p \tag{2}$$

Teorema 1.2 (Teorema cinese del resto, forma modulare)**.** *Siano p e q primi distinti, detto n il loro prodotto, allora per ogni coppia di interi α, β il seguente sistema di congruenze*

$$\begin{cases} x \equiv \alpha \mod p \\ x \equiv \beta \mod q \end{cases} \tag{3}$$

ha una soluzione ed ogni coppia di soluzioni x_1, x_2 sono congrue in modulo n.
Il teorema è generalizzabile ad un insieme finito di primi, ma per noi non sarà necessario.

Possiamo proseguire: l'ipotesi dell'Equazione 1 permette di ottenere

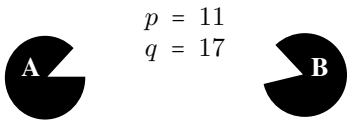
$$ed \equiv 1 \mod p-1$$
$$ed \equiv 1 \mod q-1$$

quindi, sia in modulo p che in modulo q, per il piccolo teorema di Fermat $m^{ed} \equiv m$. Infine otteniamo il risultato per il teorema cinese del resto.
Si noti che per ottenere un procedimento corretto è richiesto $m < n$, ma nemmeno questa ipotesi è restrittiva data la grandezza di n nelle implementazioni pratiche.

Un esempio è d'obbligo. Supponiamo che Alice e Bob vogliano stabilire un canale di comunicazione a distanza; si incontrano per decidere i due primi $p = 11$, $q = 17$ il cui prodotto è $n = 187$ con $\varphi(n) = 160$. L'esponente e può essere scelto come $e = 7$, mentre d deve essere preso tale che $ed-1$ sia multiplo di $\varphi(n)$, ad esempio $ed-1 = 160 = 7 \times 23 - 1$ ci suggerisce $d = 23$. Ora Alice e Bob possono costruire le due chiavi:

$$\begin{aligned} \text{chiave pubblica} \quad & (n, e) = (187, 7) \\ \text{chiave privata} \quad & (n, d) = (187, 23) \end{aligned}$$

Possono scrivere dovunque la chiave pubblica, non è necessario proteggerla; è invece fondamentale che proteggano la chiave privata. Una terza persona che fungerà da attaccante, Eve, non deve poterla ottenere.

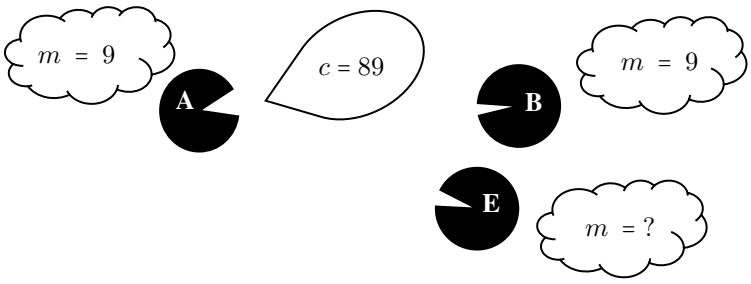


Alice e Bob possono ora separarsi, e tramite un qualsiasi canale di comunicazione Alice può inviare a Bob il messaggio $m = 9$ criptato in c sfruttando la chiave pubblica:

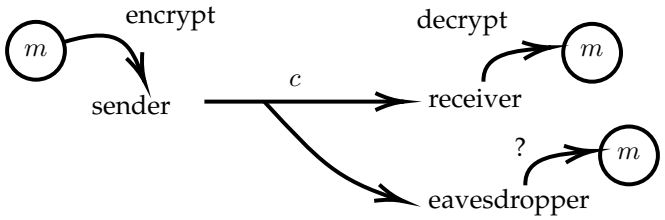
$$c = E(9) = 9^e \equiv 89 \mod 160$$

Starà a Bob recuperare la chiave criptata e decriptare il ciphertext $c = 89$:

$$m = D(89) = 89^d \equiv 9 \mod 160$$



Se invece Alice avesse provato ad inviare 170, maggiore di 160, i calcoli sarebbero risultati in $E(175) = 15$ e $D(15) = 15$.
Di seguito una schematica riassuntiva.



Ormai è chiaro che la sicurezza dell'algoritmo risieda in parte nella difficoltà del calcolo di p e q: difatti se Eve (eavesdropper) possiede solo la chiave pubblica (n, e) non può risalire alla chiave pubblica d se non fattorizzando $n = pq$ per poi calcolare $\varphi(n) = (p-1)(q-1)$ ed utilizzare l'Equazione 1 per determinare d. Per questa ragione, il crittosistema RSA utilizza interi con un numero molto elevato di cifre - congiuntamente ad altre strategie di padding/salting per ovviare ad altri tipi di attacchi, che non tratteremo. L'attacco a cui siamo interessati è l'approccio tramite *brute-force*, ovvero la ricerca della fattorizzazione di n.

Più in generale, la difficoltà nel decriptare RSA si basa su due problemi:

(i) **fattorizzazione di un intero dato**, nel nostro caso $n = pq$

(ii) **il "problema RSA"**[1], ovvero la ricerca di e possedendo solo la chiave pubblica ed il ciphertext c

Per quanto non esistano ancora algoritmi efficienti per risolvere nessuno dei due, i laboratori RSA hanno offerto un premio in denaro per la fattorizzazione di alcuni interi (prodotto di due primi) con un numero di cifre decimali compreso tra 100 e 617. Alcuni di questi sono stati fattorizzati, e risulteranno interessanti in seguito:

nome	anno	numero cifre	metodo fattorizzazione
RSA-100	1991	100	multiple-polynomial quadratic sieve
RSA-129	1994	129	quadratic sieve
RSA-130	1996	130	number field sieve
RSA-150	2004	150	general number field sieve
RSA-140	2020	140	quadratic sieve

[1] Vale la pena notare che è possibile violare RSA senza incorrere nel problema RSA, quindi senza doverlo risolvere; sono nuovamente necessarie strategie di padding per evitare questo.

2 Fattorizzare numeri interi

Come è ben noto grazie al teorema fondamentale dell'algebra, ogni elemento di $\mathbb{Z}$ ha una decomposizione unica (a meno di elementi invertibili) in numeri primi. Trovare questa fattorizzazione, tuttavia, è un lavoro decisamente complesso. Il primo metodo a cui possiamo pensare è cercare manualmente un divisore: questo è sicuramente possibile, e può addirittura essere reso un metodo più efficiente con alcune accortezze, ma ancora non basta per fattorizzare interi di grandi dimensioni, arrivando a diventare estremamente dispendioso. Infatti se un intero ha forma $n = pxqxr$, dove p, q, r sono primi con $p < q < r$, il numero di tentativi alla cieca richiesto per trovare solo il primo fattore è p. Questo metodo relativamente "elementare" viene detto *trial division*.

Chiaramente il problema della fattorizzazione si trova nella classe NP; nessun algoritmo proposto fino ad ora lo rende di classe P, se non l'algoritmo di Shor per i calcolatori quantistici. Altri algoritmi, intesi per calcolatori tradizionali, sono comunque in grado di affrontare il problema con maggiore facilità rispetto alla trial division.

Il *metodo di Fermat* si basa sulla seguente osservazione: se per un intero n dispari dato è possibile trovare a, b tali che

$$N = a^2 - b^2 \tag{4}$$

allora abbiamo ottenuto una sua fattorizzazione. In effetti i suoi fattori si esibiscono facilmente scrivendo $(a - b)(a + b) = N$. Ogni intero dispari (ma anche pari) ha una fattorizzazione simile perché se $N = \alpha\beta$

$$N = \left(\frac{\alpha + \beta}{2}\right)^2 - \left(\frac{\alpha - \beta}{2}\right)^2$$

ed ovviamente questa richiesta non è restrittiva, un intero pari sarebbe facilmente fattorizzabile. Da adesso discuteremo solo di N interi dispari.
Supponiamo ad esempio di voler fattorizzare $N = 119$, dobbiamo testare diversi valori di $a^2 - N$ fino a quando non otteniamo un quadrato, partendo da $\lceil\sqrt{N}\rceil$.

$$\begin{aligned} \lceil\sqrt{N}\rceil &= 11 & \sqrt{a^2 - N} &= 1.4142\ldots \\ 12 & & \sqrt{a^2 - N} &= 5 \end{aligned}$$

quindi $N = 12^2 - 5^2 = (12 + 5)(12 - 5) = 17 \times 7$. Chiaramente questo metodo è tanto più efficiente quanto vicino a $\sqrt{N}$ si trova uno dei fattori. Nonostante segua una procedura forse più interessante della trial division, questo algoritmo è comunque di complessità paragonabile: $O(N)$.
Curiosamente, i due possono essere combinati: non è di nostro interesse studiare il metodo in modo approfondito, ma basti notare che scegliendo un'adeguata soglia $k < \sqrt{N}$ e procedendo con la trial division fino a k per poi passare al metodo di Fermat è possibile combinare il meglio dei due algoritmi.

Tuttavia, è possibile procedere con un approccio ben più misurato.

3 Il crivello quadratico: introduzione

Nel 1982, il matematico statunitense Carl Pomerance pubblica l'articolo "*Analysis and Comparison of Some Integer Factoring Algorithms*" ([3]) in cui descrive il *quadratic sieve* (da qui in poi QS), "crivello quadratico". Si tratta di un algoritmo di fattorizzazione per numeri interi dalla notevole efficienza, che parte dall'idea di Fermat appena descritta e ne migliora il funzionamento.

> *"It occurred to me early in 1981 that one might use something akin to the sieve of Eratosthenes to quickly recognize the smooth values of Kraitchik's quadratic polynomial $Q(x) = x^2 - n$."*
>
> (Carl Pomerance, *A Tale of Two Sieves* [4])

Necessiteremo di diverse definizioni per poterlo costruire propriamente, ma iniziamo con un esempio. Supponiamo di voler fattorizzare $N = 1771$, in generale il metodo di Fermat richiederebbe di testare elementi compresi tra $\lceil \sqrt{N} \rceil$ ed $N - 1$, ovvero 42 e 1770. In particolare, in questo caso il metodo di Fermat richiederebbe 7 passaggi: partendo da 43, al passaggio con $a = 50$ individuerebbe $b = 27$ e quindi 23 come fattore primo. Il metodo di Fermat è notevolmente rapido per interi N con fattori vicini alla propria radice, ma non sempre è questo il caso. Tuttavia, invece di cercare a e b tali che $a^2 - b^2 = N$, è possibile cercarli tali che

$$a^2 - b^2 \equiv 0 \mod N \tag{5}$$

ovvero $(a - b)(a + b) = kN$ per k intero, e poi computare

$$\gcd(a - b, N), \quad \gcd(a + b, N)$$

per trovare un fattore. [2]
Ancora meglio, potremmo provare l'approccio proposto da Dixon e perfezionato da Pomerance: per arrivare allo stesso risultato più rapidamente, basta procedere esattamente come Fermat ma cercando una sequenza di a_i tali che

$$\prod a_i \equiv \gamma^2 \mod N \tag{6}$$

Tornando ad $N = 1771$, proviamo ad applicare questo metodo. Partendo da $\lceil \sqrt{N} \rceil = 43$ verifichiamo la decomposizione del suo quadrato in modulo N:

$$43^2 \equiv 78 = 2^1 \times 3^1 \times 13^1 \mod N$$

ora dobbiamo cercare degli interi tali che le potenze a destra siano pari nel prodotto. Notiamo che non è necessario scegliere $a_1 = 43$. Procedendo con un calcolatore, notiamo che

$$\begin{aligned} 43^2 &\equiv 78 = 2^1 \times 3^1 \times 13^1 \mod N \\ 44^2 &\equiv 165 = 3^1 \times 5^1 \times 11^1 \mod N \\ &\vdots \\ 75^2 &\equiv 312 = 2^3 \times 3^1 \times 13^1 \mod N \end{aligned}$$

[2] Non sempre questo risulta in fattori propri, è possibile ottenere 1 o N, i fattori banali.

Finalmente abbiamo $a_2 = 75$ tale che

$$(43 \times 75)^2 \equiv (2^2 \times 3 \times 13)^2 \mod N$$

ovvero $a = 3225$ e $b = 156$ soddisfano la relazione dell'Equazione 5, quindi i fattori di $N = 1771$ sono

$$\gcd(1771, a - b) = 11$$
$$\gcd(1771, a + b) = 161$$

Chiaramente il procedimento andrebbe ripetuto per 161, che non è primo, ma 11 è uno dei fattori primi di $1771 = 7 \times 11 \times 23$.

L'algoritmo dovrebbe essere chiaro: piuttosto che cercare direttamente a e b, possiamo provare a ricavarli con questa osservazione di natura algebrica.
Procediamo a formalizzare il metodo di Pomerance.

Definizione 3.1. *Un intero positivo si dice B-regolare se tutti i suoi fattori primi sono maggiorati da B. Si noti che B non è necessariamente un primo.*

Lemma 3.2. *Sia B un intero positivo composto, un intero N è B-regolare se e solo se è P-regolare, dove P è il maggiore primo minore di B.*

Dimostrazione. Abbastanza immediato, se è B-regolare i suoi fattori primi non possono essere più grandi di tale P; l'altra implicazione è banale. □

Possiamo quindi assumere B primo senza perdita di generalità
La B-regolarità e fondamentale per porre un limite "arbitrario" ai calcoli svolti durante l'esecuzione dell'algoritmo: infatti nello studio delle congruenze in modulo N viene solitamente fissata una soglia B per i fattori primi, ovvero scartiamo m se m^2 non è B-regolare in modulo N. Siccome stiamo limitando l'insieme dei fattori da considerare, possiamo permetterci di introdurre la seguente notazione.

Notazione 3.3. *Sia $m = p_1^{e_1} \dots p_k^{e_k}$, diciamo **exponent vector di** m il vettore*

$$m = \left(p_1^{e_1}, \dots, p_k^{e_k}\right) \tag{7}$$

per indicare in modo compatto i fattori primi di m, indicando con esponente nullo anche quelli che non appaiono. Questo tornerà utile in seguito.

Tale notazione permette di riscrivere più chiaramente il nostro esempio iniziale, $N = 1771$. In quel caso possiamo restringerci a $B = 13$, e ripercorrendo le fattorizzazioni viste

$$\begin{aligned}
43^2 &\equiv 78 = \left(2^1, 3^1, 5^0, 7^0, 11^0, 13^1\right) \mod N \\
44^2 &\equiv 165 = \left(2^0, 3^1, 5^1, 7^0, 11^1, 13^0\right) \mod N \\
&\vdots \\
75^2 &\equiv 312 = \left(2^3, 3^1, 5^0, 7^0, 11^0, 13^1\right) \mod N
\end{aligned}$$

Ora, siccome cerchiamo di ottenere un quadrato sulla destra, chiaramente tutti gli esponenti degli a_i scelti devono avere somma pari. Lavorando in $\mathbb{Z}_2$, questo significa ottenere

$$\left(2^0, 3^0, 5^0, 7^0, 11^0, 13^0\right) \tag{8}$$

oppure, considerando solo gli esponenti, $(0,0,0,0,0,0)$. In altre parole, dopo aver costruito un numero sufficiente di elementi $\lceil\sqrt{N}\rceil, \lceil\sqrt{N}\rceil+1, \ldots, \lceil\sqrt{N}\rceil+k$ è possibile costruire i relativi vettori degli esponenti

$$\left(e_1^{(0)}, \ldots, e_n^{(0)}\right), \ldots, \left(e_1^{(k)}, \ldots, e_n^{(k)}\right)$$

e rendere la ricerca degli a_i un problema di algebra lineare. Per quanto riguarda la lunghezza di tali vettori, nel nostro caso $B = 13$ questi hanno dimensione 6. In generale è possibile utilizzare la seguente funzione.

Definizione 3.4. *Si definisce prime-counting function $\pi(n)$ una funzione di forma complessa che restituisce la cardinalità dell'insieme dei numeri primi minori o uguali ad n.*

Teorema 3.5 (Teorema dei numeri primi)**.** *La funzione $p(n)$ definita come segue*

$$p(n) \coloneqq \frac{n}{\ln(n)} \tag{9}$$

è una buona approssimazione per $\pi(n)$, nel senso che

$$\lim_{n\to+\infty} \frac{\pi(n)}{p(n)} = 1 \tag{10}$$

In altre parole, la lunghezza del vettore (per B sufficientemente grande) è approssimabile con $p(B)$. Nel nostro caso $p(13) = 5.0683\ldots$, ma un calcolatore può facilmente mostrare come $\pi(13) = 6$. Non stiamo trattando numeri sufficientemente grandi per sfruttare la proprietà asintotica.

Questo è solo uno dei metodi possibili per formalizzare la ricerca di tali a_i, altri ad esempio sfruttano le frazioni continue, ma non entreremo in merito.
Ora che siamo più familiari con la struttura intuitiva dell'algoritmo, possiamo scendere nei dettagli.

4 Il crivello quadratico

L'idea è esattamente quella presentata, ma deve essere adeguatamente formalizzata. Ad esempio, come possiamo capire quando fermarci nella ricerca degli a_i? L'esserci affidati all'algebra lineare risolve solo il problema successivo, quello della giusta combinazione di certi a_i...

Seguiremo la strategia illustrata da Brillhart e Morrison in [1]. Innanzitutto, fissata una soglia B, abbiamo visto come sia opportuno tralasciare tutti i candidati non B-regolari. Questo semplifica in parte la ricerca. Per quanto concerne la quantità degli a_i da calcolare, vale il seguente risultato.

Lemma 4.1. *Fissato B e dati k interi B-regolari $m_1, \ldots, m_k$ distinti, dove $k > \pi(B)$, il prodotto di una loro sottosequenza è necessariamente un quadrato, ovvero scrivendo*

$$m_i = \left(e_1^{(i)}, \ldots, e_n^{(i)}\right) \tag{11}$$

otteniamo una sottosequenza tale che il sistema lineare

$$\begin{cases} x_1 e_1^{(1)} + \cdots + x_k e_1^{(k)} \equiv 0 \mod 2 \\ x_1 e_2^{(1)} + \cdots + x_k e_2^{(k)} \equiv 0 \mod 2 \\ \quad \vdots \\ x_1 e_{\pi(B)}^{(1)} + \cdots + x_k e_{\pi(B)}^{(k)} \equiv 0 \mod 2 \end{cases} \tag{12}$$

ammette una soluzione.

Dimostrazione. Si noti che data l'Equazione 12 stiamo lavorando su $\mathbb{Z}_2^{\pi(B)}$, uno $\mathbb{Z}_2$- spazio vettoriale di dimensione $\pi(B)$. L'equivalenza tra questa equazione e l'ottenere un quadrato come prodotto è evidente.
Data la dimensione del campo, possiamo notare di aver richiesto l'esistenza di una dipendenza lineare tra i vettori nella forma dell'Equazione 11 - e questo conclude subito la dimostrazione. □

Tornando nuovamente all'esempio $N = 1771$ con $B = 13$, $\pi(B) = 6$, quindi necessitiamo di soli 7 candidati per trovare una sottosequenza di a_i tale che $\prod_i a_i$ sia un quadrato. Possiamo concludere che la scelta svolta in precedenza, per quanto semplice, potrebbe non essere quella ottimale. Riprovando un'ultima volta:

$$N = 1771$$

$$\begin{aligned} a_1 = 43^2 &\equiv 78 = \left(2^1, 3^1, 5^0, 7^0, 11^0, 13^1\right) \mod N \\ a_2 = 44^2 &\equiv 165 = \left(2^0, 3^1, 5^1, 7^0, 11^1, 13^0\right) \mod N \\ a_3 = 49^2 &\equiv 630 = \left(2^1, 3^2, 5^1, 7^1, 11^0, 13^0\right) \mod N \\ a_4 = 50^2 &\equiv 729 = \left(2^0, 3^6, 5^0, 7^0, 11^0, 13^0\right) \mod N \\ a_5 = 56^2 &\equiv 1365 = \left(2^0, 3^1, 5^1, 7^1, 11^0, 13^1\right) \mod N \\ a_6 = 73^2 &\equiv 16 = \left(2^4, 3^0, 5^0, 7^0, 11^0, 13^0\right) \mod N \\ a_7 = 75^2 &\equiv 312 = \left(2^3, 3^1, 5^0, 7^0, 11^0, 13^1\right) \mod N \end{aligned}$$

questo porta al sistema lineare in $\mathbb{Z}_2^6$

$$\begin{cases} x_1 e_1^{(1)} + x_2 e_1^{(2)} + x_3 e_1^{(3)} + x_4 e_1^{(4)} + x_5 e_1^{(5)} + x_6 e_1^{(6)} + x_7 e_1^{(7)} \equiv 0 \mod 2 \\ x_1 e_2^{(1)} + x_2 e_2^{(2)} + x_3 e_2^{(3)} + x_4 e_2^{(4)} + x_5 e_2^{(5)} + x_6 e_2^{(6)} + x_7 e_2^{(7)} \equiv 0 \mod 2 \\ \quad \vdots \\ x_1 e_6^{(1)} + x_2 e_6^{(2)} + x_3 e_6^{(3)} + x_4 e_6^{(4)} + x_5 e_6^{(5)} + x_6 e_6^{(6)} + x_7 e_6^{(7)} \equiv 0 \mod 2 \end{cases}$$

dove l'apice (i), come prima, indica l'appartenenza alla decomposizione del numero a_i. Nel nostro caso il sistema lineare diventa

$$\begin{cases} x_1 + x_3 + x_7 \equiv 0 \mod 2 \\ x_1 + x_2 + x_5 + x_7 \equiv 0 \mod 2 \\ x_2 + x_3 + x_5 \equiv 0 \mod 2 \\ x_3 + x_5 \equiv 0 \mod 2 \\ x_2 \equiv 0 \mod 2 \\ x_1 + x_5 + x_7 \equiv 0 \mod 2 \end{cases}$$

Chiaramente otteniamo $x_2 \equiv 0$, e semplificando

$$\begin{cases} x_1 + x_3 + x_7 \equiv 0 \mod 2 \\ x_1 + x_5 + x_7 \equiv 0 \mod 2 \\ x_3 + x_5 \equiv 0 \mod 2 \\ x_2 \equiv 0 \mod 2 \end{cases}$$

da cui possiamo ottenere diverse soluzioni. Ad esempio

$$(1, 0, 0, 0, 0, 0, 1)$$

rispetta evidentemente tutti i vincoli del sistema ed è quella trovata in precedenza ($a_1 = 43$, $a_7 = 75$); potremmo trovarne anche altre, come

$$(1, 0, 1, 0, 1, 0)$$

che rappresenta $a_1 \cdot a_3 \cdot a_5 = (117992)^2$, con fattorizzazione in modulo N pari a $\left(2^2, 3^4, 5^2, 7^2, 11^0, 13^2\right)$. Concludiamo con una scrittura in pseudocodice dell'algoritmo.

Codice 1: Quadratic Sieve

```
algorithm Quadratic Sieve (integer n):
    \\ starting phase
    Choose bound B
    Calculate p prime counting function of B
    \\ finding (a_i)_i
    a = floor of sqrt(n)
    for i in [0...p]
        Choose randomly x in {0,1,–1,2,–2,...}
        a_i = a + x
        end for
    Decompose them in mod n
    Solve the linear system
    x is the product of the resulting subsequence
    y is the product of their decompositions mod n
    \\ output
    if x − y and x + y are multiples of n
        return Gcd(x−y,n) and Gcd(x+y,n)
    else
        go back and choose differently (a_i)_i
```

5 Applicazioni di QS

QS è in grado di operare ad una velocità estremamente notevole: in "*A Tale of Two Sieves*" Pomerance spiega che la complessità computazionale dell'algoritmo è $exp\sqrt{(p \log n \log\log n)}$. Curiosamente, QS è nato come una pura "discussione sulla complessità" sul *linear sieve* di Schroeppel ed alcuni algoritmi precedenti. In effetti, l'intuizione di sfruttare una sottosequenza che abbia come prodotto un quadrato nasce dal matematico inglese Dixon (si veda il suo articolo del 1982 "*Asymptotically fast factorization of integers*" [2]): Pomerance cerca di migliorare la ricerca di tale sottosequenza, rendendo quindi l'algoritmo più efficiente. La complessità computazionale in situazione ottimale del suo algoritmo è nell'ordine di $exp\left(2\sqrt{2}\sqrt{(p \log n \log\log n)}\right)$. Riferendosi ad una versione dell'algoritmo di Dixon implementata con l'utilizzo delle frazioni continue, l'algoritmo di Lehmer-Powers, Pomerance dice:

"The time to factor n is now about $exp\sqrt{(p \log n \log\log n)}$; namely, the factor $\sqrt{2}$ in the exponent is missing. Is this a big deal? You bet. This lower complexity and other friendly features of the method allowed a twofold increase in the length of the numbers that could be factored (compared with the continued-fraction method discussed above). And so was born the quadratic sieve method as a complexity argument and with no numerical experiments."

(Carl Pomerance, *A Tale of Two Sieves* [4])

Come detto, la creazione dell'algoritmo da parte di Pomerance è stata preponderantemente teorica: dopo le difficoltà nel convincere la comunità a testarlo, Joeseph Gerver prima e Gus Simmons e Tony Warnock dopo l'hanno usato su grandi numeri e perfezionato: questo ha causato la necessità di scartare diverse implementazioni hardware di sistemi RSA basate su interi di 100 cifre, tra cui una dello stesso laboratorio di Gerver e Simmons, e modificare le implementazioni software. Come abbiamo visto in precedenza, infatti, già nel 1994 RSA-129 era stato fattorizzato con successo da QS.

Il punto di forza di QS rispetto ad altri metodi è la possibilità di distribuire facilmente il calcolo su diversi calcolatori in parallelo: tale versione di QS viene detta *multiple-polynomial quadratic sieve* (MPQS), e si basa sulla seguente osservazione. Un singolo polinomio $x^2 - N$ non fornisce facilmente sufficienti interi B-regolari, ma è possibile utilizzare diversi polinomi nella forma

$$(ax + b)^2 - N \tag{13}$$

per a, b interi. In generale si isola un intervallo $[-M, M]$ scegliendo valoti tali che $b^2 - N = ac$ per un certo c, quindi il polinomio assume forma

$$a(ax^2 + bx + c)$$

Se conosciamo i fattori di a possiamo considerare solo $(ax^2 + bx + c)$, relativamente piccoli rispetto ad N, ed aumentare la probabilità che siano B-regolari. In sintesi, si ottiene

$$t^2 - u \equiv 0 \mod N$$

dove $t = ax + b$ ed $u = (ax + b)^2 - N$.

La ricerca delle congruenze ed il metodo di calcolo del polinomio $x^2 - N$ è cruciale, e diversi sforzi nel corso del tempo hanno perfezionato l'implementazione di questi calcoli per rendere più efficiente l'algoritmo. Diverse ottimizzazioni sono infatti possibili, ma non entreremo in dettaglio. Per approfondire si veda [5], dove Pomerance espande su argomenti come la verifica della B-regolarità (*processo di sieving*) e le fasi preliminari di raccolta dati. L'ultimo punto è particolarmente interessante, in quanto prima di procedere con calcoli brutali è ovviamente opportuno studiare la situazione: ad esempio la scelta di una soglia B ottimale.

Può sembrare che ci siamo allontanati dal nostro punto di inizio - il crittosistema RSA - ma è interessante notare che ci ha sempre seguiti nell'ombra, e questo può essere un momento interessante per svelarlo. Con la diffusione di crittosistemi basati sul problema algebrico della fattorizzazione di interi, domande come "quante cifre deve avere questa chiave per essere sicura?" e "quanto impiegherebbe un calcolatore a fattorizzare questo intero?" sono fondamentali. In effetti abbiamo visto come le prime implementazioni di QS, seppur ancora in uno stadio iniziale e non ottimizzato a dovere, abbiano portato ad un cambio negli standard crittografici di allora (in particolare, facendo scartare sistemi con chiavi di 100 cifre o meno). QS è in grado di fattorizzare in modo relativamente agevole interi con un numero di cifre decimali fino a 150.
Vale la pena notare che questo non significa aver individuato necessariamente una vulnerabilità negli algoritmi crittografici, in quanto un incremento della lunghezza degli interi utilizzati è sufficiente a portare nuova sicurezza: tuttavia, è sempre corretto chiedersi quando un algoritmo inizia a necessitare di chiavi troppo lunghe per restare al passo con i tempi. RSA è attualmente considerato "sicuro" con chiavi di lunghezza tra 1,024 e 4,096 bit - ovvero tali chiavi sono sufficienti per rendere la forzatura del crittosistema relativamente difficile.

Discorsi simili possono essere fatti sugli algoritmi eredi di QS, come il *General Number Field Sieve* (GNFS, *crivello dei campi di numeri generali)*, l'algoritmo di fattorizzazione più efficiente a noi noto per interi di dimensioni molto grandi - ovvero con un numero di cifre superiore a 10^{100}. Creato a metà degli anni '90, anch'esso ha portato a cambiamenti negli standard dei crittosistemi ed a diversi attacchi, come ad esempio la vulnerabilità Logjam scoperta nel 2015 per chiavi di addirittura 1024 bit.
Non dovrebbe stupirci molto, d'altronde l'avvento dei calcolatori ha portato alla risoluzione sempre più rapida di problemi matematici: si vedano le vicende dei numeri di Mersenne, matematici come Fermat ed Eulero hanno impiegato anni per fattorizzarne alcuni con esponente "solo" minore di 100, mentre dalla fine del diciannovesimo secolo fino ad ora siamo passati dalla verifica dei numeri nella congettura originale di Mersenne (esponenti $n =$) allo studio di numeri con esponenti nell'ordine di grandezza dei milioni.
Sta alla ricerca nel campo crittografico ad aggiornare continuamente gli algoritmi che usiamo tutti i giorni, e fare in modo che siano sempre in grado di tenere testa ai migliori cervelli elettronici.

22 Ottobre 2021, Trento

Riferimenti bibliografici

[1] Michael A. Morrison e John Brillhart. "A Method of Factoring and the Factorization of F_7". In: *Mathematics of Computation* 129 (1975).

[2] J. D. Dixon. "Asymptotically fast factorization of integers". In: *Math. Comp.* 36 (1981).

[3] Carl Pomerance. "Analysis and Comparison of Some Integer Factoring Algorithms". In: *Math. Centre Tract* 154 (1982).

[4] Carl Pomerance. "A Tale of Two Sieves". In: *Notices of the AMS* 43 (1996).

[5] Carl Pomerance. *Prime Numbers: A Computational Perspective*. 2005.