What is The Continued Fraction Factoring Method

Sohail Farhangi

June 2018

Why is Factoring Important?

The security of some of the most commonly used encryption schemes such as the RSA encryption scheme is proportional to the difficulty of factoring large integers quickly. Consequently, if you can create a "fast" algorithm for factoring integers, then the RSA encryption scheme will no longer be secure. Alternatively, if you can determine a lower bound for the run time of any integer factoring algorithm, then you will have proven a lower bound on the security of the RSA encryption scheme.

Determining the "speed" of an algorithm.

Given an integer factoring algorithm $\mathscr{A}$, we say that $\mathscr{A}$ runs in

(1) **exponential time** if there exists positive real numbers C, D, such that any integer N can be factored by $\mathscr{A}$ in at most CN^D steps. We note that this running time is exponential with respect to the number of digits of N, not N itself.

(2) **sub-exponential time** if $\mathscr{A}$ runs (asymptotically) faster than any exponential time algorithm.

(3) **polynomial time** if there exists positive real numbers C, D, such that any integer N can be factored by $\mathscr{A}$ in at most $C(\log(N))^D$ steps.

We note that the naive factoring method runs in at most $C\sqrt{N}(\log(N))^2$ steps.

Why is the Continued Fraction Factoring Method Important?

The Continued Fraction Factoring Method was used by John Brillhart and Michael Morrison on September 13, 1970, in order to discover that

$$2^{128} + 1 = 59649589127497217 \cdot 5704689200685129054721.$$

In particular, this was the first integer factoring algorithm with a sub-exponential running time. It was heuristically (hand-wavingly) shown in 1979 by Marvin C. Wunderlich that this algorithm factors an integer N in at most

$$C\exp(\sqrt{3\log(N)\log(\log(N))}) = CN^{\sqrt{3\frac{\log(\log(N))}{\log(N)}}}$$

steps, "most" of the time.

A Good Idea For Factoring Part 1

If we want to factor an integer N, and we have 2 distinct integers x and y for which $x^2 \equiv y^2 \pmod{N}$, then N divides $x^2 - y^2 = (x - y)(x + y)$. This tells us that there is a chance that $\gcd(x - y, N) \neq 1$ or $\gcd(x + y, N) \neq 1$. In particular, if we find many pairs of integers $\{(x_n, y_n)\}_{n=1}^{K}$ for which $x_n^2 \equiv y_n^2 \pmod{N}$, then there is a high chance that a factor of N can be obtained by computing $\gcd(x_n - y_n, N)$ and $\gcd(x_n + y_n, N)$ for all $1 \leq n \leq k$. We now need a method of generating our good pairs of integers.

A Good Idea For Factoring Part 2

Defintion: Given $B, N \in \mathbb{N}$, we say that N is B**-smooth**, if every prime factor of N is smaller than B.

Let $\pi(N)$ denote the number of prime numbers smaller than or equal to N, and let $K = \pi(B) + 1$. Now suppose that we have a sequence of integers $\{x_n\}_{n=1}^{K}$, such that $x_n^2 \equiv y_n \pmod{N}$, and y_n is B-smooth. Then through the use of Gaussian elimination over the field $\mathbb{F}_2$, we can find $A \subseteq [1, K]$, such that $\prod_{n \in A} y_n$ is a perfect square denoted by w^2. We have now obtained the good pair $(\prod_{n \in A} x_n, w)$.

We will now see an example of this procedure from [2].

A Good Idea For Factoring Part 3

Suppose that we want to factor 914387. We note that

$$1869^2 \equiv 750000 \pmod{914387} \text{ and } 750000 = 2^4 \cdot 3^1 \cdot 5^6 \cdot 11^0$$

$$1909^2 \equiv 901120 \pmod{914387} \text{ and } 901120 = 2^{14} \cdot 3^0 \cdot 5^1 \cdot 11^1$$

$$3387^2 \equiv 499125 \pmod{914387} \text{ and } 499125 = 2^0 \cdot 3^1 \cdot 5^3 \cdot 11^3$$

We want to know if we can multiply 2 or 3 of the numbers from $\{750000, 901120, 499125\}$ to obtain a perfect square. Since being a perfect square depends only on the parity of the prime factors, we see that this amount to the following matrix equation over $\mathbb{F}_2$.

$$\begin{bmatrix} 4 & 14 & 0 \\ 1 & 0 & 1 \\ 6 & 1 & 3 \\ 0 & 1 & 3 \end{bmatrix} \cdot \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix},$$

which is solved by $a_1 = a_2 = a_3 = 1$, i.e.,

$$750000 \cdot 901120 \cdot 499125 = 2^{4+14+0} \cdot 3^{1+0+1} \cdot 5^{6+1+3} \cdot 11^{0+1+3} = 2^{18} \cdot 3^2 \cdot 5^{10} \cdot 11^4$$

$$= (2^{14} \cdot 3^1 \cdot 5^5 \cdot 11^2)^2 = 580800000^2.$$

Lastly, we note that

$$1869 \cdot 1909 \cdot 3387 \equiv 9835 \pmod{914387} \text{ and } 580800000 \equiv 164255 \pmod{914387}$$

which lets us see that

$$\gcd(914387, 164255 - 9835) = \gcd(914387, 154420) = 1103 \Rightarrow 914387 = 1103 \cdot 829$$

A Good Idea For Factoring Part 4

The Continued Fraction Factoring Method, the Quadratic Sieve (C. Pomerance 1990), and the General Number Field Sieve are 3 different factoring algorithms that do this. Currently (June 2018), the General Number Field Sieve is the fastest known integer factoring algorithm for large integers. It was heuristically shown that the General Number Field Sieve can factor an integer N in at most

$$C\exp((\frac{64}{9})^{\frac{1}{3}}(\log(N))^{\frac{1}{3}}(\log(\log(N)))^{\frac{2}{3}}) = CN^{(\frac{64}{9})^{\frac{1}{3}}(\frac{\log(\log(N))}{\log(N)})^{\frac{2}{3}}}$$

steps. Integers smaller than 10^{100} tend to be factored more quickly by the Quadratic Sieve, and numbers larger than 10^{130} tend to be factored more quickly by the General Number Field Sieve. The Quadratic Sieve has the "same" asymptotic run time as the Continued Fraction Factoring Algorithm, but appears to be faster in practice, as The Continued Fraction Factoring Method seems to only factor numbers of the order 10^{50}.

A Refresher on (simple) Continued Fractions

For any $\theta \in \mathbb{R}^+ \setminus \mathbb{Q}$, there exists a unique sequence of non-negative integers $\{a_n\}_{n=0}^{\infty}$ for which

$$\theta = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cfrac{1}{\ddots}}}}.$$

Furthermore, we will use the notation

$$\frac{P_n}{Q_n} = [a_0, a_1, \cdots, a_n] = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cfrac{1}{\ddots + \frac{1}{a_n}}}}},$$

which allows us to rigorously define the infinite continued fraction expansion as a limit of the partial expansions.

In particular,

$$|\theta - \frac{P_n}{Q_n}| < \frac{1}{Q_n^2} \Rightarrow \theta = \lim_{n\to\infty} \frac{P_n}{Q_n}.$$

To calculate the sequence $\{a_n\}_{n=0}^{\infty}$, we use induction with $\theta_0 = \theta$, $a_0 = \lfloor \theta_0 \rfloor$, and

$$(\theta_{n+1}, a_{n+1}) = (\frac{1}{\theta_n - a_n}, \lfloor \frac{1}{\theta_n - a_n} \rfloor).$$

Furthermore, we may calculate the sequence $\{(P_n, Q_n)\}_{n=-1}^{\infty}$ by $(P_{-1}, Q_{-1}) = (1, 0)$, $(P_0, Q_0) = (a_0, 1)$, and

$$(P_{n+1}, Q_{n+1}) = (a_{n+1}P_n + P_{n-1}, a_nQ_n + Q_{n-1}).$$

The Continued Fraction Expansion of $\sqrt{N}$

Let $\tau = \frac{-U+\sqrt{N}}{2V} \in [0,1]$, with N a non-square, $V > 0$, $4V|N - U^2$, and $|U| < \sqrt{N}$. We note that

$$\frac{1}{\tau} = \frac{2V(U+\sqrt{N})}{N-U^2} = \frac{U+\sqrt{N}}{2V'},$$

where $V' = \frac{N-U^2}{4V} > 0$. We now see that

$$\tau = \frac{-U+\sqrt{N}}{2V} = \frac{1}{\lfloor \frac{1}{\tau} \rfloor + \frac{-U'+\sqrt{N}}{2V'}} = \frac{1}{\lfloor \frac{1}{\tau} \rfloor + \tau'},$$

where $-U' = U - 2\lfloor \frac{1}{\tau} \rfloor V'$, and $\tau' = \frac{-U'+\sqrt{N}}{2V'} \in [0,1)$.

Since $4VV' = N - U^2 \equiv N - U'^2 \pmod{4V'}$, we see that $4V' | N - U'^2$, and $|U'| < \sqrt{N}$. This is significant, because this shows us that the continued fraction expansion of $\sqrt{N}$ can be computed precisely without any worries of the accumulation of rounding errors at each step.

The Continued Fraction Factoring Method

Suppose that we want to factor the non-square integer N. Let us compute the partial continued fractions expansions $\{(P_n, Q_n)\}_{n=1}^{\infty}$ of $\sqrt{N}$ (or $\sqrt{kN}$). We note that

$$|P_n^2 - NQ_n^2| = |P_n + Q_n\sqrt{N}| \cdot |P_n - Q_n\sqrt{N}| = Q_n^2|\frac{P_n}{Q_n} + \sqrt{N}| \cdot |\frac{P_n}{Q_n} - \sqrt{N}|$$

$$< |\frac{P_n}{Q_n} + \sqrt{N}| < 2\sqrt{N} + 1.$$

Since $t_n = P_n^2 - NQ_n^2$ is "small", it has a good chance of being B-smooth, and $t_n \equiv P_n^2 \pmod{N}$, so the continued fraction expansion of $\sqrt{N}$ (and $\sqrt{kN}$) lets us generate our desired pairs of integers.

A slight improvement

We note that if p is a prime that divides one of our t_n, then p cannot divide Q_n since $\gcd(P_n, Q_n) = 1$, so we have that

$$0 \equiv t_n \equiv P_n^2 - NQ_n^2 \pmod{p} \Rightarrow N \equiv (\frac{P_n}{Q_n})^2 \pmod{p},$$

which shows us that the only primes p smaller than or equal to B that we need to consider, are the primes that divide N, and the primes for which N is a square modulo p.

References

(1) M. C. Wunderlich, *A Running Time Analysis of Brillhart's Continued Fraction Factoring Method,* in: M.B. Nathanson, ed., Number Theory Carbondale 1979, Springer Lecture Notes in Math. 751 (1979), 328 - 342.

(2) J. Hoffstein, J. Pipher, J. Silverman, *An Introduction to Mathematical Cryptography,* Springer, New York, N.Y., 2008.

(3) H. Cohen, *A Course in Computational Algebraic Number Theory,* Springer, Verlag Berlin Heidelberg, 1993.

(4) C. Pomerance, *Analysis and comparison of some integer factoring algorithms,* Computational Methods in Number Theory, (1982) 89-139.

(5) C. Pomerance, *A Tale of Two Sieves,* Biscuits of Number Theory, 2008.