A review of the general number field sieve

Rui Shen Jingqin Yang

IIIS, Tsinghua University, Beijing, China

January 10, 2020

Abstract

It is widely known that one of the most important cryptographic algorithms, RSA, is based on the fact that factoring a large number is computationally hard. This article makes a review about general number field sieve (GNFS), which is known as one of the most efficient algorithm to factor large numbers. We will show the intuition behind it, and give detailed implementation steps. At last we will analyse the time complexity, and introduce recent research results about GNFS.

1 Introduction

During security analysis of RSA, one required fact that factoring a large number is computationally hard is very important. Actually if in future we find out it is easy to factor large number, then many encryption algorithm will become insecure. Up to now, we do not know whether there exists one polynomial factoring algorithm, but we did found some subexponential factoring algorithms, and one most efficient algorithm among them is general number field sieve (GNFS) algorithm. In this article, we focus on detailed steps of GNFS and its time complexity.

In section 3, we will introduce two basic ideas, the difference of squares method and the quadratic sieve, which GNFS bases on. Then in section 4, we will show how to get GNFS from these two basic ideas. Next, in section 5, we will give formal steps of GNFS and analyse its time complexity. At last, we will introduce the recent researching result, and make a conclusion.

2 Notations

In following sections, let $f \in \mathbb{Z}[x]$ be a monic and irreducible polynomial of degree $d > 1$. Then we will use $\mathbb{Z}[\alpha]$ to represent the ring generated by a root α of f, i.e,

$$\mathbb{Z}[\alpha] = \{x : x = \sum_{i=0}^{d-1} a_i \alpha^i\}$$

And for better representing subexponential time complexities, we use L-notation which is defined as

$$L_n(\alpha, c) = e^{(c+o(1))(\ln n)^{\alpha}(\ln \ln n)^{1-\alpha}}.$$

Note that this L-notation is widely used in previous related works [2, 3, 6]

3 Basic ideas

3.1 The Difference of Squares Method

Suppose we now have a large number n, and we have found two integers x and y, s.t.

$$x^2 \equiv y^2 \pmod{n}$$

Thus we will have

$$(x+y)(x-y) \equiv 0 \pmod{n}$$

Then with high probability, $(x-y, n)$ or $(x+y, n)$ give a divisor of n. Actually let $n = pq$, we will get following table:

p|(x+y)	p|(x-y)	q|(x+y)	q|(x-y)	gcd(x+y, n)	gcd(x-y, n)	Give Factors
Yes	Yes	Yes	Yes	n	n	
Yes	Yes	Yes	No	n	p	√
Yes	Yes	No	Yes	p	n	√
Yes	No	Yes	Yes	n	q	√
Yes	No	Yes	No	n	1	
Yes	No	No	Yes	p	q	√
No	Yes	Yes	Yes	q	n	√
No	Yes	Yes	No	q	p	√
No	Yes	No	Yes	1	n	

Figure 1: Table of square method

And this idea is called the difference of squares method.

3.2 The quadratic sieve

According to the difference of squares method, we can factor a large number n by finding two distinct numbers x and y, so that their squares are equivalent module n. The quadratic sieve idea will give one efficient way to find such x and y.

Consider the quadratic polynomial

$$f(x) = x^2 - n$$

We have

$$\forall x, x^2 \equiv f(x) \pmod{n}$$

Base on this, if we can find a set of integers $\{x_k\}$, s.t, the product of their quadratic polynomial is a perfect square in $\mathbb{Z}$, i.e

$$\exists y \in \mathbb{Z}, y^2 = \prod_i f(x_i)$$

Then we can obtain that

$$\left(\prod_i x_i\right)^2 \equiv \prod_i f(x_i) \equiv y^2 \pmod{n}$$

Then we can apply difference square method on this $x = \prod_i x_i$ and y pair.

To find valid set $\{x_k\}$ which satisfy above condition, we first need to introduce *factor base* and *smoothness*:

Definition 3.1. *A nonempty set $F = \{p_1, p_2, \ldots, p_m\}$ of prime numbers is called a* factor base. *An integer n is* smooth *over a factor base F if all prime factors of n are elements of F.*

Here m is a parameter, and in practice we usually take all primes below a parameter bound y and naturally let $m = \pi(y)$, where $\pi(y)$ denotes the number of primes below y. In this case we define y-smoothness:

Definition 3.2. *An integer n is y-*smooth *if all prime factors of n are below y.*

Notice that if we can find a large set $X = \{x_k\}$ with size $|X| \geq m$ (here $m = \pi(y)$, same to above definition), s.t., $\forall x_i \in X$, $f(x_i)$ is *y-smooth*, then by linear algebra, we can easily select a subset $S \subseteq X$ such that $\prod_{x \in S} f(x)$ is a perfect square.

Then we can focus on finding the set X. A brute force solution is to select a parameter u (which is the bound of x), enumerate every $x \in [-u, u]$ and do trial divisions by all primes in F, but it is too slow because of too many divisions. To optimize this, for each prime p_i we solve the quadratic congruence equation

$$f(x) = x^2 - n \equiv 0 \pmod{p_i}.$$

This can be solved by Shanks-Tonelli algorithm and the solution looks like $x \equiv \alpha, \beta \pmod{p_i}$ (if the solution exists, i.e., $n^{(p_i-1)/2} \equiv 1 \pmod{p_i}$). Then we can mark all $f(kp_i + \alpha), f(kp_i + \beta)$ as 'divisible by p_i' like the sieve of Eratosthenes. We also can do this for each power of p_i. Then for every $x \in [-u, u]$, we just need to divide it by largest tag for each p_i and its power. This trick avoids useless divisions and leads to a dramatic speed-up.

4 The General Number Field Sieve

The GNFS algorithm still uses the difference of squares, factor base, smoothness and finding dependencies modulo 2. The key breakthrough is to use polynomial rings and look for proper factor base and smoothness in these rings.

The idea of GNFS is as follows. Let $f \in \mathbb{Z}[x]$ be a monic and irreducible polynomial of degree $d > 1$. The algorithm will work on the ring $\mathbb{Z}[\alpha]$ generated by a root α of f, i.e,

$$\mathbb{Z}[\alpha] = \{x : x = \sum_{i=0}^{d-1} a_i \alpha^i\}$$

Now assume that m is an integer which satisfies $f(m) \equiv 0 \pmod{n}$. Then there is a natural ring homomorphism $\phi : \mathbb{Z}[\alpha] \to \mathbb{Z}/n\mathbb{Z}$ defined by $\phi(\alpha) = m \pmod{n}$. Similar to the quadratic sieve, once we find a non-empty set S of relatively prime integer pairs (a, b) with properties

$$\prod_{(a,b) \in S} (a + bm) = x^2 \quad \text{is a perfect square in } \mathbb{Z}, \tag{1}$$

$$\prod_{(a,b) \in S} (a + b\alpha) = \beta^2 \quad \text{is a perfect square in } \mathbb{Z}[\alpha], \tag{2}$$

then let $y = \phi(\beta)$ and we find

$$y^2 = \phi(\beta)^2 = \phi(\beta^2) = x^2 \pmod{n},$$

which is a pair of congruent squares.

The rest of this section explains the details in the outline above.

4.1 Determine the polynomial f

Given a number n to factor, the first step is to find a polynomial f with integer coefficients and an integer m such that $f(m) \equiv 0 \pmod{n}$. To do this we present the *base m* method as follows.

Let d be a integer with $d > 1$ and $n > 2^{d^2}$. Let $m = [n^{1/d}]$, and write n to the base m:

$$n = c_d m^d + c_{d-1} m^{d-1} + \cdots + c_0.$$

Then we define the polynomial f:

$$f(x) = c_d x^d + c_{d-1} x^{d-1} + \cdots + c_0.$$

Proposition 4.1. *f is monic (i.e., $c_d = 1$), $c_{d-1} < m$, and $f(m) = n$.*

We omit the proof because it is straight-forward.

The polynomial f may be reducible. However, since our interest lies in factoring n, this would be fortunate, cause it is widely known that there exists a polynomial time algorithm [5] to factor f. At the same time, if $f(m) = g(m)h(m)$, then $p = g(m)$ is a non-trivial factor of n.

4.2 The Rational Sieve

Recall that we are to construct a set S such that

$$\prod_{(a,b)\in S} (a + bm) = x^2 \quad \text{is a square in } \mathbb{Z}, \tag{3}$$

$$\prod_{(a,b)\in S} (a + b\alpha) = \beta^2 \quad \text{is a square in } \mathbb{Z}[\alpha]. \tag{4}$$

We first focus on first line. Similar to quadratic sieve, we select a parameter u which is sufficiently large, and determine the universe of possible pairs

$$U = \{(a, b) : a, b \in \mathbb{Z}, \gcd(a, b) = 1, |a| \le u, 0 < b \le u\},$$

We try to find enough pairs (a, b) such that $a + bm \in \mathbb{Z}$ is y-smooth (y is again a parameter).

And like what we did in quadratic sieve, for each fixed integer b, note that for every prime $p \le y$, $a + bm \equiv 0 \pmod{p}$ iff $a \equiv -bm \pmod{p}$. So we can mark all such $a + kp$ as '$a + bm$ divisible by p'. We can do this for all prime $p \le y$ and their powers and obtain the set with $a + bm$ y-smooth. With the same trick in the quadratic sieve we can find a subset S with $\prod_{(a,b)}(a + bm)$ is a square.

4.3 The Algebraic Sieve

Then we focus on the second line. To apply the tricks we used before, we should define *smoothness* in $\mathbb{Z}[\alpha]$.

Definition 4.2. *An element $\beta \in \mathbb{Z}[\alpha]$ is* y-smooth *if its norm $N(\beta) \in \mathbb{Z}$ is y-smooth.*

Recall that in abstract algebra, the norm of an element $\beta \in \mathbb{Z}[\alpha]$ can be defined as

$$N(\beta) = \prod_{\sigma} \sigma(\beta),$$

where σ are the distinct embeddings of $\mathbb{Z}[\alpha]$ into $\mathbb{C}$. Note that σ just permutes α to other roots of $f(x)$, so we suppose

$$f(x) = (x - \alpha_1)(x - \alpha_2)\dots(x - \alpha_d)$$

and the norm of $a + b\alpha$ can be written as

$$\begin{aligned} N(a + b\alpha) &= \sigma_1(a + b\alpha)\sigma_2(a + b\alpha)\dots\sigma_d(a + b\alpha) \\ &= (a + b\alpha_1)(a + b\alpha_2)\dots(a + b\alpha_d) \\ &= b^d \left(\frac{a}{b} + \alpha_1\right)\left(\frac{a}{b} + \alpha_2\right)\dots\left(\frac{a}{b} + \alpha_d\right) \\ &= (-b)^d \left(-\frac{a}{b} - \alpha_1\right)\left(-\frac{a}{b} - \alpha_2\right)\dots\left(-\frac{a}{b} - \alpha_d\right) \\ &= (-b)^d f\left(-\frac{a}{b}\right) \end{aligned}$$

Note that if one element β's norm $N(\beta)$ is a perfect square, then with high probability β is a perfect square, that is why we define smoothness in this way.

Under this definition, with a small modification on the earlier sieving idea, we can find the y-smooth pairs as follows. Firstly define $R(r) = \{r \in \{0, 1, \dots, p-1\} : f(r) \equiv 0 \pmod{p}\}$ be the roots of f modulo p. For any fixed integer $0 < b \le u$, the integers a which satisfies $N(a + b\alpha) \equiv 0 \pmod{p}$ are those with $a \equiv -br \pmod{p}$ for some $r \in R(p)$. As we find the y-smooth pairs, we can easily find a subset S of pairs such that $\prod_{(a,b)\in S} N(a + b\alpha)$ is a square, and with high probability $\prod_{(a,b)\in S} a + b\alpha$ is a square.

However, even if we find out $\prod_{(a,b)\in S} a + b\alpha = y^2$ is a square, but sometimes $y \notin \mathbb{Z}[\alpha]$, which leads to failure. Luckily, by abstract algebra, we know $f'(\alpha)y \in \mathbb{Z}[\alpha]$ always holds. So in practice, we usually apply the difference square method on square roots of $f'(\alpha)^2 \prod_{(a,b)\in S}(a + b\alpha)$ and $f'(m)^2 \prod_{(a,b)\in S}(a + bm)$ at the last step of GNFS.

5 Run Time Analysis

To analyse GNFS's time complexity, we first give its whole steps.

5.1 Complete steps of GNFS

Step 1. Input a number n.

Step 2. Test whether n is a power of a prime or divisible by a prime under y. In either case, output the prime and stop.

Step 3. Find a monic polynomial $f(x)$ of degree d such that $f(m) \equiv 0 \pmod{n}$. If $f(x)$ is found to be reducible with a non-trivial factor $g(x)$, output the non-trivial factor $g(m)$ of n and stop. Assume now that $f(x)$ is irreducible and let α be a root. Check that if $\gcd(f'(m), n)$ is a non-trivial factor of n then output this factor and stop.

Step 4. Sieve all (a, b) pairs such that $\gcd(a, b) = 1, |a| \leq u, 0 < b \leq u$, $(a+bm)N(a+b\alpha)$ is y-smooth.

Step 5. Find a subset S of the smooth (a, b) pairs such that $\prod_{(a,b)\in S}(a + bm)$ is a perfect square in $\mathbb{Z}$ and $\prod_{(a,b)\in S}(a + b\alpha)$ is a perfect square in $\mathbb{Z}[\alpha]$. If this is unsuccessful, stop.

Step 6. Express the algebraic integer $\gamma = f'(\alpha)^2 \prod_{(a,b)\in S}(a + b\alpha)$ as a polynomial in α of degree less than d. Attempt to find a square root β of $\gamma(\alpha)$. If this is unsuccessful, stop.

Step 7. For an integer c with $c^2 = f'(m)^2 \prod_{(a,b)\in S}(a + bm)$, find the residue $c \bmod n$.

Step 8. Compute $\gcd(c - \beta, n)$ and $\gcd(c + \beta, n)$. If there exists a non-trivial factor of n, output the result and stop. Otherwise, remove an element of S from T and start again at Step 5.

It is clear that Steps 2, 3, 7 and 8 are negligible compared with the pivot Step 3. To analyze this pivot step, we first need to estimate how frequently the smooth numbers are distributed.

5.2 Estimating Smooth Numbers

Let $\psi(x, y)$ denote the number of y-smooth positive integers up to x. Suppose we choose random integers with the uniform distribution from $[1, x]$ and stop when we have chosen y numbers that are y-smooth. The probability that we choose a y-smooth number on one draw is $\psi(x, y)/x$, and thus the expected number of draws to choose y numbers that are y-smooth is $xy/\psi(x, y)$. For a slightly more general purpose, they propose the following theorem.

Theorem 1. *Suppose $g(y) = y^{1+o(1)}$. Then as $x \to \infty$,*

$$\frac{xg(y)}{\psi(x, y)} \geq L_x\left[\frac{1}{2}, \sqrt{2} + o(1)\right]$$

uniformly for all $y \geq 2$. In addition, the equality holds if and only if

$$y = L_x\left[\frac{1}{2}, \frac{\sqrt{2}}{2} + o(1)\right].$$

This theorem directly shows that if x is a bound on the numbers that would be smooth in a factoring algorithm, then the running time of the algorithm is at least $L_x[1/2, \sqrt{2} + o(1)]$. More importantly, this theorem implies that the sieving bound u should be at least

$$\exp\left(\left(\frac{1}{2} + o(1)\right)\left(d\log d + \sqrt{(d\log d)^2 + 4\log(n^{\frac{1}{d}})\log\log(n^{\frac{1}{d}})}\right)\right).$$

5.3 Optimal Parameters

Consider the optimal parameter choice as a function of the degree. Inspecting the case in which equality is achieved in Theorem 1, we find in a straightforward way that

$$u = y = \exp\left(\frac{1}{2}\left(d\log d + \sqrt{(d\log d)^2 + 4\log(n^{\frac{1}{d}})\log\log(n^{\frac{1}{d}})}\right)\right).$$

We shall choose u and y to be a little larger.

To minimize the above term, it is easy to see that we have to make $(d \log d)^2$ and $\log(n^{1/d}) \log\log(n^{1/d})$ of the same order, which occurs when d has the same order as $(\log n / \log\log n)^{1/3}$. Putting $d = \delta(\log n / \log\log n)^{1/3}$ and optimizing δ we find that the optimal choice of d satisfies $\delta = 3^{1/3} + o(1)$ for $n \to \infty$, i.e.,

$$d = \left(\frac{3 \log n}{\log\log n}\right)^{\frac{1}{3}}.$$

5.4 Run Time over Steps

It remains to estimate the running time of the algorithm with this choice of parameters. It is easy to see that the time taken by Step 3 equals $u^{2+o(1)}$, which is the length of the sieve multiplied by a lower order factor. We mentioned that Steps 2, 3, 7 and 8 are negligible compared with Step 3. To estimate the running time of Step 5, note that the matrix formed in this step has $y^{1+o(1)}$ columns and about as many rows. In addition, the number of non-zero entries in each row is $O(\log n) = y^{o(1)}$. Thus the number of non-zero entries in the matrix is $y^{1+o(1)}$ and the running time is $y^{2+o(1)}$. For Step 6, they claim that the running time is $y^{2+o(1)}$ with naive arithmetic and $y^{1+o(1)}$ with fast arithmetic subroutines. Thus either way this step too is dominated by Step 4. Finally, the number of times that we cycle through Step 5 to 8 is likely to be $y^{o(1)}$.

In conclusion, heuristically the total run time is dominated by $y^{2+o(1)}$, namely

$$L_n\left[\frac{1}{3}, \left(\frac{64}{9}\right)^{\frac{1}{3}}\right].$$

6 Recent Results

Actually NFS algorithms (include GNFS, TNFS, SNFS, and so on) has three main phases, namely, relation collecting (sieving), linear algebra and descent. Prior to these, is the set-up phase. In the set-up phase, two number fields are constructed and the sieving parameters are determined. The two number fields are set up by choosing two irreducible polynomials $f(x)$ and $g(x)$ over the integers such that their reductions module p have a common irreducible factor $\phi(x)$ of degree n over a finite field $\mathbb{F}_p$ (in GNFS we mentioned above, $g(x) = x - m$ is fixed). Its work-flow can be shown as follow figure:

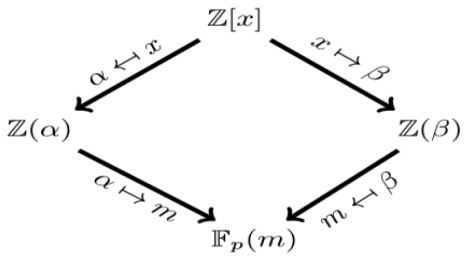

Figure 2: work-flow of NFS

The polynomial selection determines the efficiency of NFS algorithms for solving factoring or discrete logarithm in finite fields.

Palash Sarkar and Shashank Singh [8] proposed one new method of polynomial selection recently as follow:

Algorithm: $\mathcal{A}$: A new method of polynomial selection.

Input: p, n, d (a factor of n) and $r \geq n/d$.
Output: $f(x)$, $g(x)$ and $\varphi(x)$.

Let $k = n/d$;
repeat
Randomly choose a monic irreducible polynomial $A_1(x)$ having the following properties: $\deg A_1(x) = r + 1$; $A_1(x)$ is irreducible over the integers; $A_1(x)$ has coefficients of size $O(\ln(p))$; modulo p, $A_1(x)$ has an irreducible factor $A_2(x)$ of degree k.
Randomly choose monic polynomials $C_0(x)$ and $C_1(x)$ with small coefficients such that $\deg C_0(x) = d$ and $\deg C_1(x) < d$.
Define

$$\begin{aligned} f(x) &= \mathrm{Res}_y\left(A_1(y), C_0(x) + y\, C_1(x)\right); \\ \varphi(x) &= \mathrm{Res}_y\left(A_2(y), C_0(x) + y\, C_1(x)\right) \bmod p; \\ \psi(x) &= \mathrm{LLL}(M_{A_2,r}); \\ g(x) &= \mathrm{Res}_y\left(\psi(y), C_0(x) + y\, C_1(x)\right). \end{aligned}$$

until *$f(x)$ and $g(x)$ are irreducible over $\mathbb{Z}$ and $\varphi(x)$ is irreducible over $\mathbb{F}_p$.*
return $f(x)$, $g(x)$ and $\varphi(x)$.

Figure 3: new method of polynomial selection

They did many experiments and showed their method improved existing related work.

7 Conclusion

This article make a review of GNFS, one of the most important factoring algorithm for large number. We showed the intuition behind GNFS, and gave the detailed implementation steps. We also analysed its time complexity and make a research about its recent results. We hope this article can help with others interested in GNFS, and lead to more good works.

References

[1] M. E. Briggs, *An introduction to the general number field sieve*, PhD thesis, Virginia Tech, 1998.

[2] J. P. Buhler, H. W. Lenstra, and C. Pomerance, *Factoring integers with the number field sieve*, in The development of the number field sieve, Springer, 1993, pp. 50–94.

[3] D. Coppersmith, *Modifications to the number field sieve*, Journal of Cryptology, 6 (1993), pp. 169–180.

[4] T. Kim and R. Barbulescu, *Extended tower number field sieve: A new complexity for the medium prime case*, in Annual Cryptology Conference, Springer, 2016, pp. 543–571.

[5] A. K. Lenstra, H. W. Lenstra, and L. Lovász, *Factoring polynomials with rational coefficients*, Mathematische Annalen, 261 (1982), pp. 515–534.

[6] A. K. Lenstra, H. W. Lenstra, M. S. Manasse, and J. M. Pollard, *The number field sieve*, in The development of the number field sieve, Springer, 1993, pp. 11–42.

[7] C. Pomerance, *The quadratic sieve factoring algorithm*, in Workshop on the Theory and Application of of Cryptographic Techniques, Springer, 1984, pp. 169–182.

[8] P. Sarkar and S. Singh, *New complexity trade-offs for the (multiple) number field sieve algorithm in non-prime fields*, in Annual International Conference on the Theory and Applications of Cryptographic Techniques, 2016.

[9] R. D. Silverman, *Quadratic Sieve*, 2005.