General Number Field Sieve: Advanced Algorithm for Factoring Large Numbers

Qian Xie[1], Gaobo Hui[2]

1. Yao Class 50, Institute for Interdisciplinary Information Sciences, 2015011511;
2. Physics 42, Department of Physics, 2014012231

Abstract

RSA is a very popular public key cryptosystem and its security relies upon the fact that factoring a large number is computationally difficult. However, if an efficient algorithm that can factor any arbitrarily large number within the reasonable time is discovered, the RSA system could be attacked and its security value would be nullified. The General Number Field Sieve (GNFS) algorithm is the asymptotically known fastest factoring algorithm for large numbers. We first review several modern popular large number factorization algorithms including GNFS which apply the Difference of Squares Method. Their basic principles and core concepts are also introduced. In addition, we present the implementation steps of GNFS.

1 Introduction

The RSA cryptosystem invented by Rivest, Shamir and Adleman is a scheme for encrypting and decrypting secret messages, which plays a vital role in today's pervasive electronic communications and transactions. Most cryptography systems make use of *one-way functions* which can be thought of as mapping that are easy to compute but hard to invert. The one-way function in the RSA system is multiplication of large prime integers which have over 110 digits. The key is that it is almost instant to multiply such integers, whereas the inversion operation of factoring back into primes is virtually impossible. The integer factorization problem is still not known to be in P or NP-complete, so study on factorization algorithm is of great value for theory and practice.

The most straightforward way to factor a large number is through *trial division*, which can be described as simply try to divide the large number by each prime up to its square root. Although this method is guaranteed to find a factor of any composite number, it is computationally infeasible for large enough numbers.

However, the size of the numbers that we are able to factor is increasing rapidly year by year. This progress is partly attribute to the advancements in computing software, but largely credit to improvements in factoring algorithms. Many successful factor methods in the past twenty years used the same basic technique called *Difference of Squares Method*, which can be dated back to the time of Fermat. The *Dixon Random Squares Algorithm* is one that apply such method. The *Quadratic Sieve (QS) factoring*

algorithm of Carl Pomerance was the most effective factoring algorithm in the 1980's and early 1990's and still the choice for integers between 50 and 100 digits. QS has a slight modification over Dixon's method, but proves a dramatic speed-up. The General Number Field Sieve algorithm is currently the best known approach for factoring large numbers, the heart of which is Difference of Squares Method as with Dixon's method and QS. By drawing ideas and results from mathematics and computer science such as number theory, abstract algebra, linear algebra and even real and complex analysis, GNFS has great enhancements compared to the previous algorithms.

2 The Difference of Squares Factorization Method

The basic idea of Squares Factorization Method is to find integers x and y such that $x \not\equiv y \pmod{n}$ and $x^2 \equiv y^2 \pmod{n}$, that is

$$(x+y)(x-y) = x^2 - y^2 \equiv 0 \pmod{n},$$

then $\gcd(x+y, n)$ and $\gcd(x-y, n)$ are factors of n. In the case where n is the product of two distinct primes p and q, the probability that $\gcd(x+y, n)$ or $\gcd(x-y, n)$ is a non-trivial factor of n is $2/3$. The reason is that if $n = pq$, then

$$pq|(x+y)(x-y)$$
$$\Rightarrow \quad p|(x+y)(x-y) \text{ and } q|(x+y)(x-y)$$
$$\Rightarrow \quad \begin{cases} p|(x+y) \text{ or } p|(x-y) \\ q|(x+y) \text{ or } q|(x-y) \end{cases}$$

which implies that it is impossible that $p \nmid (x+y)$ and $p \nmid (x-y)$. Similarly, it is impossible that $q \nmid (x+y)$ and $q \nmid (x-y)$. The following table gives all possibilities for p and q dividing $x+y$ and $x-y$.

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	

2.1 Dixon Random Squares Algorithm

The "random squares" algorithm is one of the algorithms that apply such method. It not only devises a mean for producing integers x and y that satisfy the aforementioned conditions, but also introduces notions such as *factor base* and *smooth* employed in GNFS.

Definition 2.1.1. A nonempty set $F = \{p_1, p_2, \cdots p_m\}$ of positive prime integers is called a *factor base*. An integer n is said to be *smooth* over the factor base F if all prime factors of n are members of F.

The core idea of Dixon's method is: fix a factor base $F = \{p_1, p_2, \cdots p_m\}$ and then compute a set of random integers $r_i > \sqrt{n}$ with the property that $f(r_i) \equiv {r_i}^2 \pmod{n}$ is smooth over F. When more than m such integers are found, we can obtain a subset U such that

$$\prod_{r_i \in U} f(r_i) = {p_1}^{2e_1} {p_2}^{2e_2} \cdots {p_m}^{2e_m} = ({p_1}^{e_1} {p_2}^{e_2} \cdots {p_m}^{e_m})^2$$

where $e_i \geq 0$. Let

$$x = \prod_{r_i \in U} r_i \text{ and } y = {p_1}^{e_1} {p_2}^{e_2} \cdots {p_m}^{e_m}$$

then we have

$$x^2 \equiv \prod_{r_i \in U} {r_i}^2 \equiv \prod_{r_i \in U} f(r_i) \equiv y^2 \pmod{n}.$$

Finding the subset U is not a complicated task. For each r_i, we can associate a vector $(e_{1i}, e_{2i}, \cdots, e_{mi})$ such that
$f(r_i) \equiv {p_1}^{e_{1i}} {p_2}^{e_{2i}} \cdots {p_m}^{e_{mi}} \pmod{n}$.
It is a proven result from linear algebra that if more than m such vectors are collected, there is a non-trivial linear dependence among them. One efficient technique for finding dependencies among vectors is Gaussian elimination. There exists a vector $(a_1, a_2, \cdots, a_{m+1})$ where $a_i \in \{0, 1\}$ such that

$$\begin{bmatrix} e_{11} & \cdots & e_{1(m+1)} \\ \vdots & \ddots & \vdots \\ e_{m1} & \cdots & e_{m(m+1)} \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_m \end{bmatrix} \equiv \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} \pmod{2}$$

Let $U = \{(e_{1i}, e_{2i}, \cdots, e_{mi}) | a_i = 1, 1 \leq i \leq m+1\}$, then $\prod_{r_i \in U} f(r_i)$ must be a perfect square.

However, finding enough r_i with $f(r_i)$ smooth over F is costly. The Quadratic Sieve (QS) factoring algorithm is a further step of Dixon's method by finding smooth values in a remarkably fast manner.

2.2 Quadratic Sieve Factoring Algorithm

Instead of searching for integers r_i such that $f(r_i) \equiv {r_i}^2 \pmod{n}$ is smooth over F, QS chooses the polynomial to be $f(r_i) = {r_i}^2 - n$, then

$$x^2 \equiv \prod_{r_i \in U} {r_i}^2 \equiv \prod_{r_i \in U} ({r_i}^2 - n) \equiv \prod_{r_i \in U} f(r_i) \equiv y^2 \pmod{n}.$$

The big improvement compared with Dixon's method is how different r_i are chosen when considering whether $f(r_i)$ is smooth over F. In Dixon's method, the approach is rather straightforward: pick a random integer r_i and then trial-divide $f(r_i)$ by primes in F. If $f(r_i)$ factors completely over F, then r_i is regarded to be useful and added to the "candidate pool" of set U, otherwise it is discarded. Whereas in QS, a prime $p \in F$ is fixed and then determine which value of r_i have $f(r_i)$ divisible by that p. Specifically, the implementation is as follows: suppose p divides $f(r_i)$, then ${r_i}^2 - n = f(r_i) \equiv 0 \pmod{p}$. Hence for any integer k, it follows that

$$f(r_i + kp) = (r_i + kp)^2 - n \equiv {r_i}^2 - n \equiv 0 \pmod{p}$$

which means p divides $f(r_i + kp)$ as well. Thus the question now becomes how to initially solve the quadratic congruence ${r_i}^2 \equiv n \pmod{p}$ for once. Since division is one of the most time-consuming operations in a computer, this trick leads to a dramatic speed-up.

2.3 Generalizing the Quadratic Sieve

Almost all factoring algorithms applying difference of squares methods use the same concepts of factor base and smoothness as well as the approach of finding dependencies among vectors over $\mathbb{Z}/2\mathbb{Z}$. The key breakthroughs are to produce smooth values over a factor base in a dramatic time decrease. The breakthrough of General Number Field Sieve (GNFS) comes by first realizing that the polynomials used in Dixon's method and QS are not necessarily be quadratic. Even higher degree polynomials could produce smooth values faster. Another improvement is allowing other rings besides $\mathbb{Z}$ and $\mathbb{Z}/\mathrm{n}\mathbb{Z}$ to play a role in the algorithm. The idea is that other rings could have the similar notion of smoothness over them. Furthermore, if there exists a natural mapping between such rings and $\mathbb{Z}/\mathrm{n}\mathbb{Z}$, then we could find a different way of producing a difference of squares.

Before talking about the ring and the mapping employed in GNFS, we make a detour to introduce the concept of ring homomorphism.

Definition 2.3.1 A *ring homomorphism* is a function between two rings $\varphi: R \to S$ such that

1) $\varphi(a + b) = \varphi(a) + \varphi(b)$ for all a, b in R
2) $\varphi(ab) = \varphi(a)\varphi(b)$ for all a, b in R
3) $\varphi(1_R) = 1_S$.

Recall from subsection 2.2 that the polynomial $f(r_i) = {r_i}^2 - n$ used in the QS can be thought of as a ring homomorphism $\varphi: \mathbb{Z} \to \mathbb{Z}/n\mathbb{Z}$. Particularly, φ maps the product of all $f(r_i)$ which is smooth over F for $r_i \in U$ and a perfect square in $\mathbb{Z}$, to a perfect square in $\mathbb{Z}/n\mathbb{Z}$.

Differently, GNFS tries to find a ring R and a ring homomorphism $\varphi: R \to \mathbb{Z}/n\mathbb{Z}$ such that if $\beta \in R$ with $x \equiv \varphi(\beta) \pmod{n}$ and $\varphi(\beta^2) \equiv y^2 \pmod{n}$ then

$$x^2 \equiv \varphi(\beta)^2 \equiv \varphi(\beta^2) \equiv y^2 \pmod{n}.$$

Thus, if an element in R can be found that is a perfect square in R and maps to a perfect square in $\mathbb{Z}/n\mathbb{Z}$, then applying φ will yield a difference of squares. We will see in the next section the way to construct such rings and the corresponding homomorphism to $\mathbb{Z}/n\mathbb{Z}$.

3 General Number Field Sieve Algorithm

3.1 Selecting A Irreducible Polynomial

As mentioned, GNFS explores the important role of higher degree polynomial. Let $m \approx \lfloor n^{1/d} \rfloor$, then consider the base-*m* expansion of n:

$$n = m^d + a_{d-1}m^{d-1} + \cdots + a_0.$$

Define the monic polynomial f as

$$f(x) = x^d + a_{d-1}x^{d-1} + \cdots + a_0,$$

we can see that $f(m) = n$. Therefore, $f(m) \equiv 0 \pmod{n}$. An important criteria for f is irreducibility of $f(x)$.

Definition 3.1.1 An *irreducible polynomial* is a non-constant polynomial that cannot be factored into the product of two non-constant polynomials. The property of irreducibility depends on the nature of coefficients that are accepted for the possible factors, that is the *field* or *ring* to which the coefficients of the polynomial and its possible factors are supposed to belong.

If $f(x)$ does not meet this criteria, say it can be decomposed to $g(x)h(x)$, then either we are done (consider $n = f(m) = g(m)h(m)$, $\gcd(g(m), n)$ and $\gcd(h(m), n)$ may give factors) or we could take one of $g(x)$ and $h(x)$ to continue our algorithm.

GNFS works because of the properties of ring $\mathbb{Z}[\theta]$. Let $\theta \in \mathbb{C}$ be a root of the polynomial f, the space $\mathbb{Z}[\theta]$ is defined as follows:

$$\mathbb{Z}[\theta] = \{x : x = a_{d-1}\theta^{d-1} + a_{d-2}\theta^{d-2} + \cdots + a_0 \text{ for } \{a_j\} \subset \mathbb{Z}\}.$$

Theorem 3.1.2 *With multiplication defined as the variation of normal polynomial multiplication*

$$a(x)|_{x=\theta} b(x)|_{x=\theta} = [a(x)b(x) \pmod{f(x)}]|_{x=\theta},$$

$\mathbb{Z}[\theta]$ *forms a ring.*

Theorem 3.1.3 *Given a polynomial $f(x)$ with integer coefficients, a root $\theta \in \mathbb{C}$ and $m \in \mathbb{Z}/n\mathbb{Z}$ such that $f(m) \equiv 0 \pmod{n}$, there exists a unique mapping $\varphi : \mathbb{Z}[\theta] \to \mathbb{Z}/n\mathbb{Z}$ satisfying*

1) $\varphi(a + b) = \varphi(a) + \varphi(b) \quad \forall a, b \in \mathbb{Z}[\theta]$
2) $\varphi(ab) = \varphi(a)\varphi(b) \quad \forall a, b \in \mathbb{Z}[\theta]$

3) $\varphi(1) \equiv 1 \pmod{n}$
4) $\varphi(\theta) \equiv m \pmod{n}$
The above conditions also imply that $\varphi(za) = z\varphi(a) \ \forall a \in \mathbb{Z}[\theta], z \in \mathbb{Z}$.

We can apply this theorem to obtain a difference of squares in the following way: suppose there exists a finite set U of integer pairs (a, b) such that

$$\prod_{(a,b)\in U} (a + b\theta) = \beta^2 \quad \text{and} \quad \prod_{(a,b)\in U} (a + bm) = y^2$$

for $\beta \in \mathbb{Z}[\theta]$ and $y \in \mathbb{Z}$, then let $x = \varphi(\beta)$ we have

$$x^2 = \varphi(\beta)^2 = \varphi(\beta^2) = \varphi\left(\prod_{(a,b)\in U} (a + b\theta)\right) = \prod_{(a,b)\in U} \left(\varphi(a + b\theta)\right)$$

$$\equiv \prod_{(a,b)\in U} (a + bm) = y^2 \pmod{n}.$$

3.2 Sieving (a, b) pairs

Now we need to find a set of (a, b) pairs such that both $\prod_{(a,b)\in U}(a + b\theta)$ and $\prod_{(a,b)\in U}(a + bm)$ are perfect squares. To achieve that, first we will define a *rational factor base* (RFB) $\mathcal{R}$, and an *algebraic factor base* (AFB) $\mathcal{A}$. If we can find a set $S = \{(a, b)\}$ s. t. $a + bm \in \mathbb{Z}$ are smooth over RFB and meanwhile $a + b\theta \in \mathbb{Z}[\theta]$ are smooth over AFB, then we could use a Gaussian elimination as in section 2.1 to find a subset of S to get perfect squares.

Defining $\mathcal{R}$ is pretty easy, just set an upper bound B and list all prime numbers smaller than B. To give a definition of $\mathcal{A}$, we need to define what does "prime" mean in $\mathbb{Z}[\theta]$.

Definition 3.2.1 An *algebraic factor base* is a finite set $\{a + b\theta\} \subset \mathbb{Z}[\theta]$ where for each $a + b\theta$ in this set, $\nexists c, d \notin \mathbb{Z}[\theta]$ s. t. $c \cdot d = a + b\theta$.

Constructing and representing AFB using the form $\{a + b\theta\}$ are difficult on a computer, but fortunately we have an analog that gives a way to simplify the calculation.

Theorem 3.2.2 *The set of pairs* $\{(r, p)\}$ *where* p *is a prime and* $r \in \mathbb{Z}/n\mathbb{Z}$ *satisfies* $f(r) \equiv 0 \pmod{p}$ *is in bijective correspondence with the set of* $a + b\theta \in \mathbb{Z}[\theta]$ *that satisfy the criteria for being in an AFB.*

Now defining $\mathcal{A}$ is straightforward. Set another upper bound B' and list all the prime numbers smaller than B', for each prime p, find all $r \in \mathbb{Z}/n\mathbb{Z}$ s. t. $f(r) \equiv 0 \pmod{p}$. AFB is simply all the pairs $\{(r, p)\}$ we found.

Now we need to check whether a given pair (a, b) is smooth over both RFB and AFB. For RFB case, theories are quite simple:

Theorem 3.2.3 *A prime number* q *divides* $a + bm \in \mathbb{Z}$ *if and only if* $a + bm \equiv 0 \pmod{q}$.

Theorem 3.2.4 *A finite set* V *of primes represents a complete factorization of* $a + bm \in \mathbb{Z}$ *if and only if* $a + bm$ *could be factored as* $\prod_{q_i \in V} q_i^{e_i}$ *for some* $e_i \in \mathbb{N}$.

By theorem 3.2.3 we can select all the prime numbers that is a divisor of $a + bm$, then by theorem 3.2.4 we can tell if $a + bm$ is smooth over these primes, and get all the exponents from factoring.

For AFB case, we have similar theorems:

Theorem 3.2.5 *An element* (r, p) *in AFB divides* $a + b\theta \in \mathbb{Z}[\theta]$ *if and only if* $a + br \equiv 0 \pmod{p}$.

Theorem 3.2.6 *A finite AFB* $\mathcal{A}$ *represents a complete factorization of* $a + b\theta \in \mathbb{Z}[\theta]$ *if and only if* $(-b)^d f\left(-\frac{a}{b}\right)$ *could be factored as* $\prod_{(r_i, p_i) \in \mathcal{A}} p_i^{e_i}$ *for some* $e_i \in \mathbb{N}$.

Now we can check the smoothness of $a + b\theta$ as well as $a + bm$. Note that for some prime p, there may be more than one corresponding (r, p) pairs satisfies the condition described in theorem 3.2.2. If our AFB included one of them, then all of them should be in AFB to make theorem 3.2.6 correct.

To get enough pairs of (a, b), one can fix b and make a vary form $-M$ to M for a predefined upper limit M, and list all the integers in the range $[-M + bm, M + bm]$. Then perform a Sieve of Eratosthenes algorithm on this list using primes in RFB. Instead of removing all composite numbers in standard sieve algorithm, we divide the composite number by the current using prime until it's not divisible. After sieving, smooth numbers are divided into 1. For each 1 in the resulting list, try to factor corresponding $(-b)^d f\left(-\frac{a}{b}\right)$ using primes p_i in AFB $\mathcal{A} = \{(r, p)\}$ to tell if (a, b) is also smooth over AFB. After that one can collect all the smooth (a, b) pairs, if more pairs are needed, choose another fixed b value and sieve again.

3.3 From Smooth Numbers to Square Numbers

Up to this point, we are able to find a set of numbers $S = \{(a, b)\}$ such that $a + bm$ is smooth over rational factor base $\mathcal{R}$ and $a + b\theta$ is smooth over algebraic factor base $\mathcal{A}$. This section we will describe how to use this information to find a square in $\mathbb{Z}$ and in $\mathbb{Z}[\theta]$.

The idea is similar to the technique for finding dependencies in subsection 2.1. But unfortunately, $\prod_{(a,b)\in U}(a+b\theta)$ may not be a perfect square even if all the exponents of its factors are even.

Theorem 3.3.1 *If* $\prod_{(a,b)\in U}(a+b\theta)$ *is a perfect square in* $\mathbb{Z}[\theta]$*, then for any* (s,q) *satisfies the criteria for being in an AFB, if* $(s,q) \nmid a+b\theta$ *for any* $(a,b)\in U$*, then*

$$\prod_{(a,b)\in U}\left(\frac{a+bs}{q}\right)=1,$$

where the Legendre symbol $\left(\frac{a}{p}\right)$ *for* $a\in\mathbb{Z}$ *and* p *a prime is defined as:*

$$\left(\frac{a}{p}\right)=\begin{cases}1, & if\ x^2\equiv a\pmod{p}\ has\ a\ solution\\ -1, & if\ x^2\equiv a\pmod{p}\ has\ no\ solution\\ 0, & if\ p|a\end{cases}$$

So we can define an *quadratic character base* (QCB) $\mathcal{Q}$ using the same method as defining AFB, but with prime numbers which are not used in AFB. Since $a+b\theta$ is smooth over AFB, all of the pairs $(s,q)\in\mathcal{Q}$ will not divide $a+b\theta$ for any $(a,b)\in S$, then we can check the value of $\prod_{(a,b)\in U}\left(\frac{a+bs}{q}\right)$ to rule out these $\prod_{(a,b)\in U}(a+b\theta)$ which are not perfect squares.

Suppose the RFB $\mathcal{R}$ has k elements, the AFB $\mathcal{A}$ has l elements, and QCB $\mathcal{Q}$ has u elements. $\mathcal{R}$ and $\mathcal{A}$ will be used in finding a square in $\mathbb{Z}$ and $\mathbb{Z}[\theta]$, and $\mathcal{Q}$ will be used in verifying that the result is a perfect square.

Each $(a,b)\in S$ can be represented as a row vector with $1+k+l+u$ entries. The first entry is to indicate the sign of $a+bm$, that is 0 if $a+bm$ is positive, and 1 if $a+bm$ is negative. The next k entries are left to the exponent vector as described in subsection 2.2. The following l entries are used for indicating whether a particular element of $\mathcal{A}$ divides $a+b\theta$. The final u entries are used in conjunction with the QCB $\mathcal{Q}$. For each entry, if for corresponding $(s,q)\in\mathcal{Q}$, $\left(\frac{a+bs}{q}\right)=1$, then set it to 0, otherwise set it to 1.

Suppose the set S of smooth pairs (a,b) have j elements. Let X be a $j\times(1+k+l+u)$ matrix with each row being the aforementioned vector representation of $(a,b)\in S$. Finding a subset $U\subset S$ in order to get perfect squares is equivalent to finding a column $A=[A_1,A_2,\cdots,A_n]^T$ where $A_i\in\{0,1\}$ such that

$$X^TA\equiv 0\pmod{2}$$

If $j>1+k+l+u$, this congruence is guaranteed to have a nontrivial solution A. If

$A_i = 1$, then put $(a_i, b_i) \in S$ into U. Then $\prod_{(a_i,b_i)\in U}(a_i + b_i m)$ is a perfect square in $\mathbb{Z}$ and $\prod_{(a_i,b_i)\in U}(a_i + b_i \theta)$ is a perfect square in $\mathbb{Z}[\theta]$. With the mapping φ from the subsection 3.1, we have

$$\varphi\left(\prod_{(a_i,b_i)\in U}(a_i + b_i\theta)\right) \equiv \prod_{(a_i,b_i)\in U}(a_i + b_i m) \pmod{n}.$$

The remain question is to compute the square root of $\prod_{(a_i,b_i)\in U}(a_i + b_i\theta)$ in $\mathbb{Z}[\theta]$ since the above method can easily derive the square root of $\prod_{(a_i,b_i)\in U}(a_i + b_i m)$. For the sake of simplicity, we will not discuss the specific implementation here.

Reference

[1] Case, M. (2003). A Beginner's Guide To The General Number Field Sieve.

[2] E. Briggs, Matthew. (2018). An Introduction to the General Number Field Sieve.

[3] 刘新星, 邹潇湘, & 谭建龙. (2014). 大数因子分解算法综述. 计算机应用研究(11), 3201-3207.