Math 626: Quadratic Sieve Algorithm

Basile Hurat

May 2019

Abstract

This paper looks at the implementation and optimization of the Quadratic Sieve factoring algorithm for large integers which was proposed in 1981 as a possible approach for cryptanalyzing the RSA cryptosystem. we discuss the steps involved in the Quadratic Sieve, and give some optimal parameters in search of a factor. We then give an example and go over both runtime and parallelization approaches of the Quadratic Sieve. In discussing the complexity of the algorithm, we acknowledge that it is not a sufficiently fast algorithm for cryptanalysing RSA, as it is subexponential in complexity, and is slower than another subexponential factoring algorithm proposed in 1993 called the Number Field Sieve.

1 Introduction

Ever since the proposal of the RSA cryptographic system in 1977, the study of its cryptanalysis has been vastly important. Since the cryptanalysis of RSA is equivalent to the factoring of large primes, it is important to study the work done so far on this problem to see how secure RSA cryptosystems are to integer factorization attacks so that we may determine secure key sizes. This paper will look at the Quadratic Sieve Algorithm, which is currently the second fastest algorithm that has been properly implemented for factoring large integers. It will go over some brief history of the Quadratic Sieve Algorithm and walk through the steps of the algorithm, including an in depth example. We will then discuss of the algorithms properties related to performance, such as runtime and possible parallelization.

2 Introduction to the Quadratic Sieve

The Quadratic Sieve was proposed in 1981 by Carl Pomerance [3] and is an extension of the Dixon Random Squares Algorithm, which in turn is an elaboration on Fermat's difference of squares approach from the 1600's. While it was outpaced by the Number Field Sieve in 1993 [5], the Quadratic Sieve is still faster for numbers that are less than 110 digits. Also, in 1994, the RSA-129 challenge was cracked using a Quadratic Sieve algorithm that ran parallel on over 1600 computers. We now discuss how the Quadratic Sieve works.

3 Implementation of Quadratic Sieve

As mentioned above, the Quadratic Sieve is an extension of the Dixon Random Squares and Fermat's algorithm. This means that if we are trying to factor a large integer n, we are looking for x, y such that

$$x^2 = y^2 \mod n \quad \text{and} \quad x \neq \pm y \mod n$$

Should such x, y exist, then it is easy to see that $(x-y)(x+y) = 0 \mod n$ and assuming that they are not trivial factors (1 and n), then we have successfully factored n. So, we consider $Q(x) = x^2 - n$. Then, we will find $\{x_i\}$ such that

$$Q(x_1)Q(x_2)\dots Q(x_i) = (x_1, x_2, \dots x_i)^2 \mod n.$$

Getting to this, we have many steps. We begin by constructing a factor base.

3.1 Constructing Our Factor Base

To begin looking for a product of $\{Q(x_i)\}$ that is a perfect square, we want to get the prime factorization of some $\{Q(x_i)\}$. As such, we must construct a factor base of prime numbers (including -1) with which to decompose our $\{Q(x_i)\}$. However, we should not consider every prime number. First, we limit ourselves to primes less than some given β, to aid with computation. Furthermore, we note that if some prime p divides $Q(x)$, then

$$Q(x) = x^2 - n = 0 \mod p \implies x^2 = n \mod p.$$

If this is true, then n must be a quadratic residue of p. As such, we construct our factor base thusly:

$$B = \left\{ p \middle| p \text{ is prime or } -1,\ p \leq \beta, \text{ and } \left(\frac{n}{p}\right) = 1 \right\}$$

When considering the size of our factor base, we should choose carefully. Too small a factor base leads to less fully factorable numbers found within our sieve, while too large leads to excess computation. [2] states that an ideal number of factors is given by

$$|B| = \left\lfloor \left(e^{\log n \log \log n}\right)^{\frac{\sqrt{2}}{4}} \right\rfloor.$$

We now consider our sieving interval.

3.2 Choosing a Sieving Interval

When it comes to choosing our interval, we begin by centering our first interval around $\lfloor \sqrt{n} \rfloor$, leading to an interval of $\left[\lfloor \sqrt{n} \rfloor - \lfloor M/2 \rfloor, \lfloor \sqrt{n} \rfloor + \lfloor M/2 \rfloor\right]$, where M is our interval size. Once again, we must exercise care when choosing M. On one hand, a sieve size that is too small may lead to having to sieve multiple times to get the desired amount of x_i (we will discuss this desired amount in section 3.4). On the other hand, too large a sieve might mean operating on more numbers than we need to. Generally, the ideal size of our sieve is the cube of our factor base's size.[2] Therefore, we get

$$M = |B|^3 = \left(e^{\log n \log \log n}\right)^{\frac{3\sqrt{2}}{4}}.$$

3.3 Sieving

Once we have our factor base and our sieving interval, we may begin sieving. We take each integer in our sieving interval and see if it factors completely with our factor base. If it does, it is said to be smooth and is kept. If it does not, we throw it away. However, rather than working on our interval one number at a time, we instead sieve through the entire interval. To do this, we consider a prime p that divides $Q(x_i)$. If $p|Q(x_i)$, then $p|Q(x_i+p)$. Thus, we can consider the equation

$$Q(x) = x^2 + n = 0 \mod p.$$

Then, if $s^2 = n \mod p$ we can use Tonelli-Shanks algorithm for modular square roots to give us s_{1_p} and $s_{2_p} = p - s_{1_p}$. Then the $Q(x_i)$ are divisible by p if $x_i = s_{1_p} + kp$ for some integer k or $x_i = s_{2_p} + kp$ for some integer k. For each of these elements $Q(x_i)$, we then divide the highest power of p and record that power $\mod 2$ in a vector. We do this for all $p \in B$. By the end, we will have a set of smooth $\{Q(x_i)\}$ with corresponding vectors V_i. For example, if we had the factor base $B = \{-1, 2, 5, 17, 23\}$ and $Q(x_i) = 2 * 5^2 * 17^3$, then $V_i = (0, 1, 0, 1, 0)$.

At this point, we consider whether we have enough vectors to reliably get our factorization. We will discuss how to judge this in the following section.

3.4 Constructing our Matrix

If $k = |\{Q(x_i)\}|$, then we can use our vectors V_i to build a matrix $k \times |B|$. If $k \leq |B|$, we do not have enough vectors and we repeat the above process on a new interval. This gets us a new set of smooth $\{Q(x_i)\}$ and V_i which we can add to our previous set and recheck to see if we have enough vectors.

If $k > |B|$, then we have $k - |B|$ linear dependencies. However, the probability of having a trivial factor is $1/2$. Thus, after constructing our matrix, we have a probability of $1 - 2^{k-|B|}$ to find a nontrivial factor. When using the algorithm, therefore, you may choose a stopping point given the reliability you want for guaranteeing a nontrivial factor.

Once we have a sufficient set of vectors V_i, you build a $k \times |B|$ matrix, A. We then want to find a linear combination of e_i's where $e_i \in \{0, 1\}$ such that if a_i is a row of our matrix A, we get

$$\sum_{i=1}^{k} a_i e_i = 0$$

This means that we must solve

$$eA = 0,$$

where $e = (e_1, e_2, \dots, e_k)$.

By performing a modular Gaussian elimination, we can then get a spanning set of $|B|$ vectors. All other vectors can then be summed up with a linear combination of our spanning set to get $eA = 0$. Once we have a linear combination such that $\sum_{i=1}^{k} a_i e_i = 0$, we consider

$$\prod_{i|e_i=1} Q(x_i) = \left[\prod_{i|e_i=1} x_i \right]^2 \mod n$$

which gets us our factor x and y. At this point, we simply have to check to see if x and y are trivial factors. If they are, we simply try the next linear combination. If they are not, we check the primality of x and y and reapply the Quadratic Sieve Algorithm if they are composite. Of course, this last step is unnecessary for RSA cryptanalysis, as we are guaranteed for x and y to be prime.

3.5 Example of Quadratic Sieve

Say we wish to factor $n = 202669$. First, we create our factor base. We know that an optimal size is given by

$$|B| = \left\lfloor \left(e^{\log n \log \log n} \right)^{\frac{\sqrt{2}}{4}} \right\rfloor = 7.$$

Thus, we find the first 7 primes (including -1) p_i whose Legendre symbol $\left(\frac{n}{p_i}\right) = 1$. This gets us

p_i	-1	2	3	5	7	11	13	17	19	23
$\left(\frac{n}{p_i}\right)$	1	1	1	1	-1	1	1	-1	-1	1

meaning our factor base is $B = \{-1, 2, 3, 5, 11, 13, 23\}$. With that, we consider our sieving interval. Once again, we know that our optimal sieve size is given by $M = |B|^3 = 343$. Also, $\lfloor \sqrt{n} \rfloor = 450$. Thus, our sieving interval will be $[450 - \lfloor 343/2 \rfloor, 450 + \lfloor 343/2 \rfloor] = [279, 621]$. We now perform the task of sieving. I give the first 10 numbers before and after sieving to show the process.

x_i	279	280	281	282	283
$Q(x_i)$	-124828	-124269	-123708	-123145	-122580
Post-Sieve	2837	1801	61	2239	227

x_i	284	285	286	287	288
$Q(x_i)$	-122013	-121444	-120873	-120300	-119725
Post-Sieve	4519	30361	40291	401	4789

The smooth numbers found are given below with their corresponding vectors

x_i	337	387	392	395
$Q(x_i)$	-89100	-52900	-49005	-46644
V_i	$(1,0,0,0,1,0,0)$	$(1,0,0,0,0,0,0)$	$(1,0,0,1,0,0,0)$	$(1,0,1,0,0,0,1)$

x_i	418	433	437	447
$Q(x_i)$	-27945	-15180	-11700	-2860
V_i	$(1,0,1,1,0,0,1)$	$(1,0,1,1,1,0,1)$	$(1,0,0,0,0,1,0)$	$(1,0,0,1,1,1,0)$

x_i	450	455	463	473
$Q(x_i)$	-169	4356	11700	21060
V$_i$	$(1,0,0,0,0,0,0)$	$(0,0,0,0,0,0,0)$	$(0,0,0,0,0,1,0)$	$(0,0,0,1,0,1,0)$

x_i	487	499	502
$Q(x_i)$	34500	46332	49335
V_i	$(0,0,1,1,0,0,1)$	$(0,0,0,0,1,1,0)$	$(0,0,1,1,1,1,1)$

x_i	513	554	612
$Q(x_i)$	60500	104247	171875
V_i	$(0,0,0,1,0,0,0)$	$(0,0,0,0,1,1,0)$	$(0,0,0,0,1,0,0)$

Looking at these, we see that x_{14} and x_{17} have the same V_i. Thus, we begin by considering

$$46332 \cdot 104247 = (554 \cdot 499)^2 \mod n \implies$$

$$(\sqrt{46332 \cdot 104247} - 554 \cdot 499)(\sqrt{46332 \cdot 104247} + 554 \cdot 499) = 0 \mod n.$$

Thus, we have

$$(\sqrt{46332 \cdot 104247} - 554 \cdot 499) = 69498 - 276446 = -206948 = 198390 \mod n.$$

Getting our GCD, we get

$$\gcd(198390, n) = 389.$$

Thus, one factor is 389. By taking $n/389$, we get our second factor of 521. Both of these are prime. Before moving on, we note that other possible choices for our linear combination would lead to the same results, such as $\imath \in \{12, 13, 14\}$ or $i \in \{10, 11, 15\}$. However, choosing the linear combination of $i \in \{3, 7, 14\}$ does not work, as the resulting factors are trivial (1 and n).

4 More On The Quadratic Sieve

With our understanding of the Quadratic Sieve, we now turn our discussion to the performance of the Quadratic Sieve algorithm.

4.1 Runtime

Given the choice for $|B|$ and M given above, the Quadratic Sieve has a running time of roughly

$$\mathcal{O}\big(e^{(1.125 \log n \log \log n)}\big).$$

However, we can reduce this slightly by choosing a better algorithm for our modular Gaussian elimination. There are two choices for this: Lanczos and Wiedermann [1]. Wiedermann has better performance than Lanczos for the finite field $GF(2)$ [6], which is the field we are working in. By using this method instead of regular Gaussian elimination, we approach an order of

$$\mathcal{O}\big(e^{\log n \log \log n}\big),$$

which is a significant improvement. Of course, this is not faster than the Number Field Sieve, which has a running time of

$$\mathcal{O}\big(e^{1.9223((\log(n))1/3(\log(\log(n)))2/3)}\big)$$

and the Quadratic Sieve is still subexponential and not polynomial. However, in the next section we discuss ways to shorten practical runtime via parallelization.

4.2 Parallelization

There exist a few approaches to parallelize the Quadratic Sieve Algorithm. The first is simply to break up our sieving interval into subintervals and sieve them parallelly.

Another, less naive way, is to instead choose polynomials different from $Q(x) = x^2 - n$. This approach is known as the Multiple Polynomial Quadratic Sieve and it constructs polynomials of the form

$$Q(x) = ax^2 + 2bx + 4c.$$

where our coefficients are chosen in the following ways. First, we let a be a square. Then, we choose a nonnegative $b < a$ such that $b^2 = n \mod a$. However, this constraint will only hold true if n is a square $\mod q$ for every q that divides a, which means we must check the Legendre symbol $\left(\frac{n}{q}\right) = 1$ for each q that divides a, which places further constraints on a. Lastly, we choose c such that $b^2 - 4ac = n$. With these choices for our parameters, we see that

$$aQ(x) - n = a^2x^2 + 2abx + 4ac = (ax + b)^2 - b^2 + 4ac = (ax + b)^2 - n$$

Thus,

$$(ax + b)^2 = aQ(x) \mod n,$$

and the Quadratic Sieve method can be used to find factors of n.

Using the Multiple Polynomial Quadratic Sieve method has a few challenges however. First, for each polynomial, one must find an ideal sieve length to optimize the search for our factors. Second, there is a computational cost to switching polynomials, mostly due to the calculation of $Q(x) = 0 \mod p$ for each prime in our factor base. However, the main benefit, besides simply parallelization, is a reduced size in both factor base and sieve interval size. The optimal sizes for these is still a topic of study.

5 Conclusions

In summary, the Quadratic Sieve algorithm is a subexponentional factoring algorithm. It takes advantage of various number theory properties when choosing its factor base and in the sieving process. It also has optimal numbers for factor base and interval sizes, which guarantee a faster and reliable runtime. Lastly, despite speed increases due to choice of Gaussian elimination methods and the structure for some parallelization, the Quadratic Sieve's complexity remains much too high to cryptanalyze the RSA cryptosystem, whose encryption and decryption is polynomial time, or even compete with the Number Field Sieve, which is a faster factoring method for sufficiently large integers.

References

[1] Eric Landquist, "The Quadratic Sieve Factoring Algorithm", 2001.

[2] Joseph L. Gerver, "Factoring Large Numbers with a Quadratic Sieve", Mathematics of Computation, Vol. 41, No. 163, 1983.

[3] Carl Pomerance, "Analysis and Comparison of Some Integer Factoring Algorithms, in Computational Methods in Number Theory", , H.W. Lenstra, Jr. and R. Tijdeman, eds., 1982

[4] Carl Pomerance, "Smooth Numbers and the Quadratic Sieve", Algorithmic Number Theory, MSRI Publications, Volume 44, 2008.

[5] Carl Pomerance, "The Number Field Sieve", Proceedings of Symposia in Applied Mathematics, Volume 48, 1994.

[6] D. Wiedemann, "Solving Sparse Linear Equations over Finite Fields", IEEE Trans. Inform. Theory, 32, 1986.

[7] Douglas R Stinson, Maura B. Paterson, "Cryptography: Theory and Practice 4e", CRC Press, 2018.

[8] G. L. Honaker, Jr., "Prime Curios", shttps://primes.utm.edu/curios/home.php

[9] Rosetta Code, "Tonelli-Shanks Algorithm", "https://rosettacode.org/wiki/Tonelli-Shanks_algorithm".