Dixon's Factorization method

Nikithkumarreddy yellu

December 2015

Contents

1 Introduction

Dixon's Factorization method is an integer factorization algorithm. It is the prototypical factor method. The only factor base method for which a run-time bound not dependent on conjectures about the smoothness properties of values of a polynomial is known. Dixon's technology depends on discovering a congruence of squares modulo the integer.[2] Using Fermat's factorization algorithm we can find a congruence by selecting a pseudo-random x values and hoping that $x^2 mod N$ is a perfect square.

Dixon's algorithm tries to find x and y efficiently by computing $x, y \epsilon Z n$ such that $x^2 \equiv y^2 \, (mod N)$. Then with probability $\geq \frac{1}{2}$, $x \not\equiv \pm y \, (mod N)$, hence gcd $(x - y, n)$ produces a factor of n with probability $\geq \frac{1}{2}$.

2 History

In 1981, John D. Dixon, a mathematician at Carleton University,[3] developed the integer factorization method that bears his name. Dixon's algorithm is not used in practice, because it is quite slow, but it is important in the realm of number theory because it is the only sub-exponential factoring algorithm with a deterministic (not conjectured) run time, and it is the precursor to the quadratic sieve factorization algorithm, which is eminently practical. This approach was discovered by Micheal Morrison and John Brillhart and published in 1975.

Figure 1: John D. Dixon

Dixon didn't know the whole history when he published his 1981 paper, but

he included it in a later paper. In what seems to be a theme with important work in cryptology from the last 38 years. Dixon's 1981 paper was rejected by the first journal to which he submitted it. Dixon didn't suggest that the randomized version which he described would be competitive in practice with algorithms which were currently in use.

3 Method

Suppose we are trying to factor the composite number N. We choose a bound B, and identify the factor base (which we will call P), the set of all primes less than or equal to B. Next, we search for positive integers z such that $z^2 mod N$ is B-smooth. We can therefore write, for suitable exponents ak,

$$z^2 \equiv \prod_{p_i \epsilon P} p_i^{a_i} \, (mod N)$$

When we have generated enough of these relations (it's generally sufficient that the number of relations be a few more than the size of P), we can use the methods of linear algebra (for example, Gaussian elimination) to multiply together these various relations in such a way that the exponents of the primes on the right-hand side are all even

$$\mathrm{z}_1^2 z_2^2 ... z_k^2 = \prod_{p_i \epsilon P} p_i^{a_{i,1}+a_{i,2}+...+a_{i,k}} \, (mod N)$$

This gives us a congruence of squares of the form $a2 \equiv b2(mod N)$, which can be turned into a factorization of N, $N = gcd(a+b, N)(N/gcd(a+b, N))$. This factorization might turn out to be trivial (i.e. N = N 1), which can only happen if $a \equiv b(mod N)$, in which case we have to try again with a different combination of relations; but with luck we will get a nontrivial pair of factors of N, and the algorithm will terminate.

3.1 Factor-base

Factor base is a small set of prime numbers commonly used as a mathematical tool in algorithms involving extensive sieving for potential factors of a given integer.
If we want to factorize an integer N. We need to generate a large number of integer pairs (x,y) for which $x \not\equiv \pm y$, $x^2 \equiv y^2 \, (mod N)$ and $x^2 \, (mod N)$ and $y^2 \, (mod N)$ can be completely factorized over the chosen factor base—that is, all their prime factors are in P.

3.2 B-smooth

A positive integer is called B-smooth if none of its prime factors is greater than B. For example, 720 has prime factorization $2^4 3^2 5^1$: therefore 720 is 5-smooth because none of its prime factors are greater than 5.[1]

4 Examples

4.1 Example1

We will try to factor N = 84923 using bound B = 7. Our factor base is then P = 2, 3, 5, 7. We then search randomly for integers between $4\lceil\sqrt{84923}\rceil = 292$ and N whose squares are B-smooth. Suppose that two of the numbers we find are 513 and 537:

$513^2 mod 84923 = 8400 = 2^4.3.5^2.7$

$537^2 mod 84923 = 33600 = 2^6 \cdot 3 \cdot 5^2 \cdot 7$

So

$(513 \cdot 537)^2 84923 = 2^{10} \cdot 3^2 \cdot 5^4 \cdot 7^2$

For the below calculations we apply some rules of modular arithmetic and division of integers.

$(513 \cdot 537)^2 mod 84923 = (275481)^2 mod 84923$

$= (84923.\ 3 + 20712)^2 mod 84923 (weusetheEuclideanalgorithm 275481 = q * 84923 + rwithqand0 \leq r < 84923 (q = 3andremainderr = 20712)$

$=(84923.\ 3)^2 + 2.(84923.3.20712) + 20712^2 mod 84923$

$= 0 + 0 + 20712^2 mod 84923$ (we use $a^n modb = (amodb)^n$
(ab)modc=(a mod c).(bmodc)

With the fact that integer multiples of the modulus 8492384923 are zero

That is, $20712^2 mod 84923 = (2^5.3.5^2.7)^2 mod 84923 = 16800^2 mod 84923$.

The resulting factorization is 84923 = gcd(20712 - 16800, 84923) * gcd(20712 + 16800, 84923) = 163 * 521

4.2 Example2

Say we want to factor n=23449 over s = 2,3,5,7.

$x = [\sqrt{n}] = 154$. Starting here, the first related squares we get are:

$970^2 mod(23449) = 2940 = 2^2 * 3 * 5 * 7^2$

$8621^2 mod(23449) = 11760 = 2^4 * 3 * 5 * 7^2$

So, (970* 8621)$^2 \equiv (2^3 * 3 * 5 * 7^2)^2 (mod 23449)$

That is, $14526^2 \equiv 5880^2 (mod 23449)$

Now , we find:

gcd(14526-5880,23449) = 131

gcd(14526+5880,23449) = 179

Factors are n = 131 * 179

5 Algorithm

Here are the steps of the algorithm.

1. Randomly select a ϵ Zn.
2. Let $b = a^2 (mod n)$
3. Check if b is k-smooth [k to be defined later]
4. If YES, let b = $\prod_{i=1}^{t} {p_i}^{\alpha^i} where \{p1, ..., pt\}$ is the set of primes $\leq k$.
5. Collect t + 1 such pairs (a1, b1),(a2, b2), $\cdots$,(at+1, bt+1).
6. Let bj= $\prod_{i=1}^{t} {p_i}^{\alpha^i j}$
7. Find $\beta j' ssuchthat \sum_{j=1}^{t+1} \beta j \ \alpha ij$ is even for each i
8. x = $\prod_{j=1}^{t+1} {a_j}^{\beta_j}$ and $y = \left(\prod_{j=1}^{t+1} {b_j}^{\beta j} \right)^{1/2}$

6 Optimizations

The quadratic sieve is an optimization of Dixon's method. Dixon's elegant factorization method was improved upon the very year it was published. Carl Pomerence published his method, the Quadratic Sieve, in 1981 while at the University of Georgia. It selects values of x close to the square root of N such that x 2 modulo N is small, thereby largely increasing the chance of obtaining a smooth number. Where Dixon's method blindly tries to factor each f(x) over the prime base S, the 7 Quadratic Sieve only considers primes in S that have a quadratic residue of n, that is:

$n \equiv t^2 (mod p)$

For some integer t, and where n is the number we are attempting to factor. This way, it is easy to see when a given prime will divide $f(x) = x^2$, since if the residue of p is t, p| $f(x) if x = t, (t + 7), (t + 27), (t + 37), ...$

This extension of Dixon's Method is much faster, and is actually the fastest algorithm for factoring numbers with less than 115 decimal digits[5].

Other ways to optimize Dixon's method include using a better algorithm to solve the matrix equation, taking advantage of the sparsity of the matrix: a number z cannot have more than log2z factors, so each row of the matrix is almost all zeros. In practice, the block Lanczos algorithm is often used. Also, the size of the factor base must be chosen carefully: if it is too small, it will be difficult to find numbers that factorize completely over it, and if it is too large, more relations will have to be collected.

The optimal complexity of Dixon's method is

In big-O notation:

$O\left(exp\left(2\sqrt{2}\sqrt{lognloglogn}\right)\right)$

In L-notation:

$L_n\left[\frac{1}{2}, 2\sqrt{2}\right]$

7 Conclusion

In this paper we explained how to implement Dixon's factorization method for prime factorization. In addition to the improvement in efficincy, this algorithm can be run in parallel by many machines, making this a great method for factoring very large numbers. This type of parallel attack was used to factor RSA-129 in 1994 over a time period of 8 months.

References

[1] *https://en.wikipedia.org/wiki/Smoothnumber.*

[2] Thorsten: et al. Kleinjung. *Factorization of a 768-bit RSA modulus.* 2010.

[3] Craig P.Bauer. *Secret history : The story of Cryptology.* 2013.