FBA

Abstract

One of the most difficult challenges in number theory and cryptography is the problem of factoring large integers since factoring large numbers more than 200 digits remain difficult. This difficulty has been useful in proving the strength of the RSA encryption scheme, which uses two large numbers of similar size. In this work, we introduce factorisation by bases algorithms (FBA) and implement it on the RSA-numbers defined by the RSA challenge. We analyse the time spent on factoring to check the speed and efficiency of the algorithm.

Introduction

Integer factorization is the task of computing the divisors of natural numbers. It is a problem with a long and fascinating history, and it is certainly among the most influential in algorithmic number theory. While there is a variety of algorithms significantly faster than the brute-force search for divisors, it is still an open problem to construct a technique that efficiently factors general numbers with hundreds to thousands of digits. The hardness of this problem is fundamental for the security of widely used cryptographical schemes, most prominently the RSA cryptosystem. Nevertheless, there is no proof for its hardness besides the fact that decades of efforts have failed to construct a more efficient technique. Quite regularly, there are set new records1 concerning the factorization of numbers of certain size, mostly due to improved implementations of the best available algorithms and advances in the hardware and computing power. In addition, the bound for the deterministic integer factorization problem has been improved multiple times in recent years ([11], [12], [10], [14]). On the other hand, there has only been little progress in the development of new techniques for practical integer factorization since the invention of the Number Field Sieve ([19]) in the 1990s. One of the earlier algorithms with sub-exponential runtime was by Dixon ([7]) in 1981.

Background

Let N be the number we want to factorize. We will always assume that N is odd, composite and not a perfect power of another number. We will also assume that N is made up of 2 prime factors, p and q. We will explain properties of factorisation by bases techniques using one number in base 10.

Lemma: N can be converted into a polynomial given a base.

Example: 667 in base 10 is

Now 667 is built by two prime factors which are 23 and 29. If both of them are converted into base 10 they become, and respectively.

which is different from the first form of N gotten. (A1)

From this derivation we notice that cannot be factorised to give us p and q because the discriminant was imaginary. Furthermore, we also notice that for any converted form of N we get we cannot factorise if also the discriminant is imaginary. From (A1) we noticed that a change of the form happened with h=2 and k=2 (A2) which results in, . This becomes the obtained equation.

If we say

We obtain the discriminant: which reduces to

(A3)

To solve such a problem is quite complex. We want to reduce the equation to 2 variables which can be easily solved either as Diophantine equations or through some ingenious method. There are many ways to do this.

CASE I (A4)

Given Here if we insert a value for m for example, 12 we get: which is a conic section. We will refer to this paper written by on solving conic sections.

Note: Since this one is a conic section and to find the values of h,k we will need to factorise N using SIQS or ECM, hence this one will not have an algorithm.

CASE II (A5)

Given Let if we solve it, we get We will now only be left with finding m of which by Diophantine solutions, we know that m is of a specific parametric solution.

Let

Note: This one will have an algorithm. Name it ALGORITHM 1.

CASE III (A8)

If we are to go back to the original form with only variables, we notice that and . For the discriminant we get,

Making unknowns subjects of the knowns, for example, making subject of B, we get.

And for N=667 in base 10, we obtain.

Well, this means that we now have 2 equations of which we can use to find solutions. We now have four variables of which we need a third equation to reduce all the variables to one variable which can easily be solved with. From the other cases above, we noticed that Case II gave us a breakthrough by giving us values of k and h as variables of m.

LET

(T1)

(T2)

Then insert either m or o as fixed value, then iterate the one left as free.

Note: ALGORITHM 2 Will have to remove this part since we know that the combination of m and o is very strict hence its complex to solve.

CASE IV

In this case we introduce a rather unpopular method of factorising. Suppose we want to factorise N, we derive a secondary N, name it to N2, then we factorise it instead. The factorisation of N2 will lead to the factorisation of N using relationships which existed between the two numbers. This is the approach in this scenario.

Given we have,

(T1)

(T2)

We establish a relationship between h and k except that of the form and then insert a value for o and solve using (T2). This will give results of h and k basing on the square of m, ie, both h and k will consist of some solution in which we have to find where all the values are integers. We solve assuming that it won’t relate to N, hence factorising N will not happen if we are to solve it by Alpetron calculators.

Example.

(T2)

The solutions are:

The solutions are:

Which shows that they are not related to N.

Note: Make ALGORITHM 3

CASE V

Let’s have then it show that for the discriminant we have

Here we can insert any value for b and m and get deterministic results.

For example, for N = 667, x = 10, we have:

If b=30 and m=12 we get: . The solution set for 6 values is shown in the table below.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| a | -26 | -10 | -2 | 2 | 4 | 5 |
| c | 8 | 9 | 11 | 15 | 23 | 39 |

And the solutions

If we substitute x with 10 for all the solutions, we will find none equating to 667 and its multiples. However, if we try to find solutions for x such that we get the prime solutions, for example.

For

Using x=27, for the solutions of x to find p and q, we get:

But finding solutions such that takes time especially when resolving large numbers, since current methods require us to factorise N first.

The solution to this problem is covered in the next step.

Here we put C to be C-N for f(x).

Using N=667, x=10, we get.

Let b=-24 and m=10, we get:

As for the solution sets, we have:

And if we take all the solutions of we have:

We had no f(10)=667 or its multiples.

Assuming S is a group and using the preliminary laws of algebra, a group is said to be associative, commutative and contains and identity element as well as an inverse. In our case, let the identity be the prime factor we want to find. Then we have whereby \* is addition.

Examples:

Therefore p=gcd(e, N) ALGORITHM 4