Impact of Decreasing Polynomial Degree in Time Needed to Factor a 100 Digits Integer by General Number Field Sieve Algorithm

Jamal A. Othman
Iraq Commission for Computers & Informatics/ Baghdad
Email:Jamalothman2003@yahoo.com

Received on: 6/7/2011 & Accepted on: 5 /1/2012

ABSTRACT

Factoring is very important in the field of cryptography, specifically in the Rivest, Shamir, Adleman(RSA) public-key cryptosystem, one of the most prevalent methods for transmitting and receiving secret data which its security relies on the fact that factoring large composite numbers (may be 300 digits) is computationally intensive task . The General Number Field Sieve (GNFS) algorithm is the fastest known method for factoring large integers (100 digits and more), selecting a polynomial is the first and the most important step of the general number field sieve. Choosing a "good" polynomial as a first step in GNFS algorithm widely affect the time needed to factor the integer which we intend to factor .In this paper we concern about polynomial selection step in GNFS, we try to examine practically the affect of choosing two different degrees polynomial on the time needed to factor a 100 digits integer. Base – m method is a reasonable first step for generating a suitable polynomial for the GNFS, in this method we first choose the degree of the polynomial (d=4 or 5 in our case) and looking for $m \approx N^{\overline{d+1}}$ and a polynomial f of degree d for which $f(m) = 0 \mod n$,we begin with $f(x) = \sum_{i=0}^{d} a_i x^i$ where the a_i are the coefficients of the base-m representation, Brian Murphy in his PhD thesis considered being the first how deeply study the effect of choosing a good polynomial on the time needed to factor a number n, he came out with a parameter $\alpha(f)$ to measure without sieving the quality of the polynomial ,many programs have been written using Murphy parameter $(\alpha(f))$ and iterate on the leading coefficient of the base-m polynomial till reaching the required $(\alpha(f))$ value which we intend to reach, we get use from a program freely available on the net which help us to choose a good polynomial after feeding the program with the required parameter . We find a 4th degree polynomial need less time than a 5th degree polynomial to factor an integer n of a 100 digits, we present the result of factoring showing the two factors and the coefficients of the two polynomials and other information related to the factoring jobs using a python script written by Brian Gladman's available for free use on the net called (factmsieve.py) in two appendices.

Keywords: Public key Cryptosystem (PKC), The Integer Factorization Problem (IFP), RSA public key cryptosystem, General Number Field Sieve (GNFS), Polynomial selection in GNFS.

https://doi.org/10.30684/etj.30.4.5
2412-0758/University of Technology-Iraq, Baghdad, Iraq

تأثير خفض درجة متعددة الحدود على الوقت اللازم لتجزئة عدد صحيح من 100 مرتبة بواسطة الخوارزمية العامة لغربلة حقل الاعداد

الخلاصة

تعتبر تجزأه الاعداد الصحيحه من المفاهيم المهمه في التشفير خاصة فيما يتعلق بالتشفير بأستخدام المفتاح العام الذي يعتبر من أساليب التشفيرالاكثر استخداما في ارسال وأستلام البيانات بشكل سري عبر وسائل الاتصالات المختلفه والذي طوره ثلاث مختصين تم نسبه اليهم هم رايفز ، شامير ، أدلمن (أر ، أس ، أي) والذي تعتمد موثوقيته على حقيقه أن تجزأة عدد صحيح كبير نسبيا (أحيانا مكون من 300 مرتبه) من المهام الصعبة من ناحية الوقت اللازم لانجازها . تعتبر الخوارزمية العامة لغربلة حقل الاعداد أسرع الطرق المعروفة لتجزأة الاعداد الصحيحة الكبيرة نسبيا (100 مرتبة وأكثر). أن الخطوة الاولى في هذه الخوارزمية هي اختيار متعددة حدود ضمن شروط معينة . يؤثر الاختيار المناسب لمتعددة الحدود هذه بشكل كبير على الوقت اللازم لتجزأة العدد الصحيح المزمع تجزأته . تعتبر متعددة الحدود للأساس – m (Base-m) أختيار أولي مناسب لخوارزمية غربلة حقل الاعداد (GNFS) حيث نحدد أولا درجة متعددة الحدود وعدد m بحيث $m \approx N^{\frac{1}{d+1}}$ ومتعددة حدود f ذات درجة d بحيث $f(m) = 0 \mod n$ ومنها ستكون متعددة الحدود الابتدائية $f(x) = \sum_{i=0}^{d} a_i x^i$ ، يعتبر براين مورفي Brian (Murphy) من أشهر الذين درسوا بشكل معمق تأثير الاختيار المناسب لمتعددة الحدود على سرعة تجزأة عدد n وخرج بمقياس لجودة متعددة الحدود ($\alpha(f)$) يدعى مقياس مورفي .

نهتم في هذا البحث بأختيار متعددة الحدود في خوارزميه غربلة حقل الاعداد أذ نحاول أن نختبر بشكل عملي تأثير أختيار متعددتي حدود مختلفتين على الوقت اللازم لتجزأه عدد صحيح (n) مكون من 100 مرتبة تمت الاستعانة بأحد البرامج المتاحة الاستخدام على الشبكة العنكبوتية والتي تستخدم مقياس مورفي وتعمل بشكل تكراري على تغيير قيمة المعامل الرائد (Leading Coefficient) لمتعددة الحدود لحين الحصول على متعددة حدود تحقق مقياس مورفي المحدد ولقد وجدنا أن أستخدام متعددة حدود رباعية في تجزأه العدد (n) أستغرق وقت أقل من متعدده حدود خماسية أستخدمناها في تجزأه نفس العدد . . قدمنا بملحقين في نهايه البحث البيانات الخاصه بالتجزأه وأهمها مركبي العدد (n) المركب الاول (r1) والمركب الثاني (r2) ومعاملات متعددتي الحدود وبيانات أخرى تخص الخوازمية وقد أسنخدمنا سناريو مكتوب بلغه بايثون مكتوب من قبل برين كلادمان مشاع الاستخدام على الشبكه العنكبوتيه .

أن المعادلة التي تستخدم لقياس تعقيد خوارزمية غربلة حقل الاعداد (Algorithm Complexity)لاتحوي ضمن معالمها (Parameters) على درجة متعددة الحدود لذلك لانستطيع نظريا مناقشة تأثير درجة متعددة الحدود على تعقيد الخوارزمية فلجئنا الى الاختبار العملي لقياس تأثير تغيير درجة متعددة الحدود على الوقت اللازم للتجزأة . على الرغم من التعقيد النسبي لخوارزمية التجزاة وكون عملية التجزأة تتطلب عدة ساعات إلا أن تدقيق النتيجة لم يتطلب إلا ثواني فقط من خلال أستخدام البرنامج ماتلاب (MATLAB) تم التأكد من صحة عملية التجزأة أذ قمنا بضرب العددين الناتجين من التجزأة r1 و r2 وكانت النتيجة ان حصلنا على العدد n أي

أننا واثقين بشكل تام من صحة ناتج التجزأة ،أما بالنسبة لمنطقة الغربلة Sieving Region فلكــون متعددة الحدود الرباعية تأخذ قيم أصغر من متعددة الحدود الخماسية لذلك تم تصغير منطقة الغربلة في حالة متعددة الحدود الرباعية .أن النتيجة التي حصلنا عليه لايمكن تعميمها لآي عــدد n لعــدم وجود أساس نظري لذلك .

INTRODUCTION

The ability to conduct secure electronic communications and transactions has become an issue of vital concern. There are two distinct paradigms for approaching security, One is the symmetric or private-key path [1], which can be viewed as a descendant of the very early attempts of security. Any such system assumes that, if two individuals wish to communicate securely, they must both possess a unique common secret key, which is used for both encryption and decryption. While many such systems have evolved over time, it is the difficulties of sharing and distributing the common key and keeping it a secret that leads to their vulnerability and the need for a new paradigm. By the early 1970's a significantly new and different perspective on security began to take shape contributing to the emergence of public-key cryptosystem. With Public Key Cryptosystem (PKC) [2] cryptosystem, the first depend upon the intractability of the Discrete Logarithm Problem (DLP) while the second depend upon the intractability of Integer Factorization Problem (IFP). Factoring numbers is very important in the field of cryptography, it is a mean by which we could break the security of the cryptosystem, one of the most prominent systems for securing electronic information, known as RSA, relies upon the fact that it is computationally difficult to factor a "large" integer into its component prime integers. If an efficient algorithm is developed that can factor any arbitrarily large integer in a reasonable amount of time, the security value of the RSA system would be nullified. The General Number Field Sieve(GNFS) algorithm is the fastest known method for factoring large integer, selecting a polynomial is The first and the most important step of the general number field sieve, literally there can be billions of choices for the polynomials it has been seen that in many cases some polynomials are remarkably better than others. Polynomial selection still remains an underdeveloped area. We try in this paper to compare the time consumed to factor a 100 digits integer by two different degrees polynomials (fourth and fifth degree) using GGNFS. We find that a forth degree polynomial is faster in factoring a 100 digits integer than the fifth degree polynomial. In the coming sections we will speak first briefly about subjects related to our work's so we will speak about public-key cryptosystem (PKC) , integer factorization problem (IFP), and the main topics in which our paper concerned ,the general number field sieve (GNFS) and polynomial selection in GNFS .

Public-Key Cryptosystem (Pkc)

With Public-Key Cryptosystem (PKC) each user has a specific private-key which only the user knows and a mathematically related public-key which can be made public and freely distributed. Based upon the mathematical problem, we can distinguish between two main kinds of public key cryptosystem, the first depend upon the intractability of the Discrete Logarithm Problem (DLP) while the second depend upon the intractability of Integer Factorization Problem (IFP). The first published work on Public Key Cryptosystem was in a groundbreaking paper by Whitfield Diffie and Martin Hellman [3] titled "New Directions in Cryptography" in November, 1976. The paper described the key concepts of PKC .This paper revolutionized the world of cryptography and galvanized dozens of researchers around the world to work on practical implementations of a public key cryptography algorithm. It was the first practical method for establishing a shared secret over an unprotected communications channel. In Diffie-Hellman key exchange, a finite field $GF(p)$ and a generator $g \in GF(p)$ are chosen and made public. Suppose that two users "A" and "B" wish to agree upon a key. User "A" selects a random integer $2 \leq x \leq p-2$, and transmits g^x to "B" over a public channel. User "B" also selects a random integer $2 \leq y \leq p-2$, and transmits g^y to "A". The users "A" and "B" having common key g^{xy} , compute $(g^x)^y = g^{xy}$ and $(g^y)^x = g^{xy}$ respectively. Another important public Key encryption algorithm is the ElGamal [4] encryption system which is an asymmetric key encryption algorithm for public-key cryptography based on the Diffie–Hellman key exchange. It was described by Taher Elgamal in 1985. The algorithm works as follows: User "A" selects a finite field GF (q) and a generator $g \in GF(q)$, and an integer x then he publishes (g, g^x) as the public-key and keeps x secret. User "B", who requires to send a message $m \in GF(q)$ to "A", selects an integer y, $2 \leq y \leq q-2$ randomly, computes $m.(g^x)^y = m.g^{xy}$, and sends the pair $(g^y , m . g^{xy})$ to "A". User "A" who knows x , recovers m by computing $m . g^{xy} (g^y)^{-x} = m . g^{xy} . G^{-xy} = m$. Finding an efficient discrete logarithm algorithm (DLP) would make this system unsecure, since g , g^x and g^y are all known, and x, y or g^{xy} can be computed. Another way for breaking this scheme is to compute g^{xy} from g^x and g^y , without computing either x or y. The best known attack against the above schemes is the index calculus method which is sub-exponential in nature. Elliptic Curve Cryptosystem (ECC) is another important public key cryptosystem, the first elliptic curve scheme was proposed by Koblitz [5] and Miller [6] independently, it is based on a group of points on an elliptic curve which are defined over a finite field. There is no sub exponential-time algorithm that could solve the Discrete Logarithm Problem (DLP) in these groups. On the other hand, the best known attack against Elliptic curve cryptosystems is exponential in nature. So the use of an elliptic curve group that is smaller in size maintains same level of security and offers potential reductions in bandwidth, storage, processing power, electrical power and message sizes. The above cryptosystems are group based and depend upon the discrete logarithm problem (DLP). There are other important cryptosystem schemes which depend upon intractability of integer factorization (The integer factorization (IF) problem), RSA (Rivest, Shamir,

Adleman)[7] is the most used one of them. It is based on the product of two prime numbers .The keys for the RSA algorithm are generated as follows: Choose two distinct uniformly at random prime numbers p and q Compute $n = pq$ (n is used as the modulus for both the public and private keys) Compute $\varphi(pq) = (p-1)(q-1)$. (φ is Euler's totient function), Choose an integer e such that $1 < e < \varphi(pq)$, e is released as the public key, Determine $d = e^{-1} \bmod ((p-1)*(q-1))$,it will satisfy $d * e \equiv 1 \bmod \varphi(pq)$ and d is kept as the private key[8].

User "A", transmits his public key (n, e) to User "B", and keeps the private key d secret. If User "B" wishes to send message m to User "A", He first turns m into an integer $< n$, He then computes the cipher text $c = m^e \bmod n$, to decrypt user "A" recover m from c by using his private key d throw the computation

$$m = c^d \bmod n .$$

The Integer Factorization Problem (IFP)

Many important cryptosystem schemes depend upon intractability of (IFP), Factoring integers is quite an old challenge of great interest, most modern method of factoring are variants of Fermat's method in which the number n to be factored is written as a difference of two squares $n = x^2 - y^2$,and so we conclude that $n = (x-y)(x+y)$. for a large number n , as it is the case in the public key cryptosystem (PKC) in which the key is an integer composed of around 300 digits , finding the two numbers x and y with their squares differ by n is not an easy task and it is the cornerstone of the security of the PKC , Kraitchik came up with an interesting enhancement, instead of trying to find integers x and y such that $x^2 - y^2 = n$ it might be suffice to find x and y with $x^2 \equiv y^2 \bmod n$ [9], and it is this enhancement that is at the basis of most modern factoring algorithms . Many sieving algorithms developed to improve finding these numbers starting from the linear sieve up to the algebraic sieve. A sieve algorithm searches a lot of numbers satisfying a certain property. Then it makes some tests systematically on all these numbers, and at the end keeps the ones that have passed all the tests successfully, so sieving is really the act of filtering. The general number field sieve (GNFS) considered being the state of the art for the factoring algorithms, the key idea is to use smooth numbers in a number rings deferent from Z. For the RSA public key cryptosystem which is depend upon factoring, if we could factor the integer n which is part of the public key (the public key used to be equal to (n, e) to two prime numbers p and q such that $n = p * q$,then, we may recover the private key (d) from the relation $d = e^{-1} \bmod ((p-1)*(q-1))$ since we choose from the beginning the private key (d) to satisfy the relation

$e\,d \equiv 1 \bmod (p-1)*(q-1)$. [10]

GENERAL NUMBER FIELD SIEVE (GNFS)

Factoring is very important in the field of cryptography, specifically in the Rivest, Shamir, Adleman(RSA) Public-Key Crypto system , one of the most prevalent methods for transmitting and receiving secret data. The General Number Field Sieve (GNFS) algorithm itself uses Ideas and results from diverse fields of mathematics and computer science. Algebraic number theory, finite fields, linear algebra, and even real and complex analysis all play vital roles in GNFS. [11]

The General Number Field Sieve algorithm is the fastest known method for factoring hard composite numbers [12]. It operates by finding congruent squares mod n, which lead to a non-trivial factorization of n. This approach is a generalization of the idea that all odd composite numbers can be represented as a difference of two squares, thus providing a non-trivial factorization. One can't efficiently use this fact to factor, as simply choosing values for x and checking to see if $n - x^2$ is a square. One can extend this approach by noting that if$x^2 \equiv y^2 (\, mod\, n)$, non-trivially (that is, $x \not\equiv \pm y \ \ (mod\, n)$) Then gcd $(x - y, n)$and gcd $(x + y, n)$ are non-trivial factors of n. The question remains how to generate such values of x. GNFS use a two main steps process to arrive at these numbers. First, numbers of a particular form are sieved for smoothness and then a matrix reduction step is used to find subsets of the smooth numbers that can be multiplied together to form the candidate congruent squares. We expect roughly half of these candidates to be non-trivial, so by generating many such candidates we can have a very high probability that we will find a non-trivial factorization. The key idea of the algorithm is finding two perfect squares in two different rings Z and Z[α] , suppose one can find a $\beta^2 \in Z[\alpha]$ that is a perfect square and a $y^2 \in Z$ that is a perfect square. Then one can produce a difference of squares congruence that can be used to factor n. GNFS consist of the following steps:

1. Polynomial Selection
2. Sieving
3. Matrix Reduction
4. Square Root.

In this paper we are mainly concerned about polynomial selection step which has been considered as a major open problem since the inception of the number field sieve. The problem is to choose polynomial which ensure a good supply of smooth values, The asymptotic advantage of the number field sieve is that its polynomial guarantee a better supply of smooth values than is the case of other sieving algorithms[12], The polynomial selection problem concerns how to exploit this advantage in practice. The aim is to choose polynomial which generate many smooth values and so reduce the effort required in the time consuming in the sieving step.

The GNFS algorithm

The GNFS algorithm which is considered to be relatively complicated had been presented and discussed in many papers, the equations which is used her taken from Murphy [13], J. Buhler, H. W. Lenstra, Jr. and C. Pomerance[14], *Carl Pomerance* [15],we also get use from a guide written by Michael Case from Oregon State

University titled " A Beginner's Guide To The General Number Field Sieve" ,we are going to introduce briefly the steps within GNFS. Let n be a large composite integer to be factored, to achieve this we will do the following:

(a) Choose an $m \in Z$ and find a corresponding polynomial f satisfying $f(m) \equiv 0 \ (mod\, n)$ by the base-m expansion method, let α be the root of the polynomial

$$f(\alpha) = 0 .$$

(b) Define a rational factor base R such that R has finitely many elements and $\forall x \in R, x$ is prime. Let k be the number of elements in R , we will looking values which is factored completely upon the rational factor base R we call these values a smooth values.

(c) Define an algebraic factor base A such that A has finitely many elements and$\forall (r,p) \in A$, p is prime and r satisfies$(r) \equiv 0 \ (mod\, p)$. Let l be the number of elements in A.we will looking for elements which are completely factored upon the rational factor base; we will call such values smooth values.

The main object of the sieving step is to find values which are smooth on both the rational factor base and algebraic factor base stimulatingly and after that to find a subset of these values , a set of pairs $\{(a_1, b_1), \ldots, (a_s, b_s)\}$ such that

1. $\prod(a_i - b_i m) \in Z$ is a perfect square,
2. $\prod(a_i - b_i \alpha) \in Z[\alpha]$ is a perfect square.

(d) Define a quadratic character base Q with finitely many elements so that $\forall (s,q) \in Q, q$ is prime and $f(s) \equiv 0 \ (mod\, q)$ ensure that$\forall (s,q) \in Q, (s,q) \notin A$. Let u be the number of elements in Q.

(e) Build sieve arrays as follow:

Fix$b \in Z$, and let S be an arbitrary positive integer. Let a vary from $-S$ to S Create two arrays: one for the various values of $a + b\alpha$ that will result and another for the various values of $a + bm$ that will result. Note the elements of the sieve arrays that are smooth i.e. to record elements for which $q_i \in R$ and $(r_i, p_i) \in A$ are divisors. Repeat this process for various b as necessary until more than $1 + k + l + u$ pairs (a,b) have been found such that $a + bm$ is smooth in Z and $a + b\,\alpha$ is smooth in$Z[\alpha]$. Let y be the number of smooth (a,b) found.

(f) Populate a $y * (1 + k + l + m)$ matrix X , which is a matrix all its element is 0 or 1 depend on whether the power of the decomposition of the smooth elements (a,b) found upon the factor base is even or odd.

(g) Solve the equation

$$X^T \begin{bmatrix} A_1 \\ \vdots \\ A_y \end{bmatrix} \equiv 0 \ (\mathrm{mod}\ 2)$$

For $\{A_1, \ldots, A_y\}$. Let the subset $V \subset U$ be defined by $\forall (a_j, b_j) \in U, (a_j, b_j) \in V$ if $A_j = 1$. Then

$\prod_{(a_j,b_j)\in V}(a_j + b_j m)$ Is a perfect square in Z and $\prod_{(a_j,b_j)\in V}(a_j + b_j\alpha)$ is a perfect square in Z[α].

(h) With a homomorphism $\emptyset : Z[\alpha] \to Z_n$ we get

$$\emptyset\left(\prod_{(a_j,b_j)\in V}(a_j + b_j\alpha)\right) \equiv \prod_{(a_j,b_j)\in V}(a_j + b_j m) \pmod n$$

Use this to attempt factoring n using the difference of square factorization method, if no factorization is found, go to step (b) and repeat this process until we get a congruence square mod n from which we could factor the integer n .

GNFS Complexity Estimates

Most of the complexity measurements in cryptography are often based on experimentation and heuristics, and simply give an estimate of the expected running time.

GNFS is one of the "congruent squares" Algorithms family typically have heuristic asymptotic run-times described by the $L-$ function [16] were

$$L_n[\alpha, c] = \exp\left[(c + o(1))(\log n)^{\alpha}(\log\log n)^{1-\alpha}\right]$$

And in the case of GNFS $\alpha = 1/3$ and $c = (64/9)^{1/3}$ [11], so for the integer which we factored the heuristic asymptotic run-times will be

$$L_n\left[\frac{1}{3}, \sqrt[3]{\frac{64}{9}}\right] = \exp\left(\left(\sqrt[3]{\frac{64}{9}} + o(1)\right)(\log n)^{\frac{1}{3}}(\log\log n)^{\frac{2}{3}}\right)$$

We easily could note that the above L-function formulas haven't any parameter related to the polynomial choice; the only parameter which appears is the number (n) which we intend to factor. GNFS considered being a complicated algorithm (actually it is a set of related algorithms) in which we has to perform computations and factorizations in two different number fields, This results in the complicated aspects of the algorithm but once we factored the intended number there is no doubt about the result i.e. we have a highly reliable result. We check the result of our factorization by using the Matlab program by multiplying the two factors (r55) and (r45) to get the number (n).

Polynomial Selection in GNFS

The GNFS algorithm includes two free parameters that must be chosen first. These free parameters will be used along with a composite integer n that is to be factored. The first such parameter is a polynomial $f : R \to R$ with integer coefficients, and the second parameter is a natural number $m \in N$ that satisfies $f(m) \equiv 0\ (mod\ n)$.

Consider the base-m expansion of n . [9]

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

By defining the function f as

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

$f(m) = n$. Therefore $f(m) \equiv 0 \pmod n$, and so f and m meet the above criteria. Polynomial selection is the first step in the GNFS algorithm steps and it has been considered as a major open problem since the inception of the number field sieve. The problem is to choose polynomials which ensure a good supply of smooth values, The asymptotic advantage of the number field sieve is that its polynomials guarantee a better supply of smooth values than is the case of other sieving algorithm[11], The polynomial selection problem concerns how to exploit this advantage in practice. The aim is to choose polynomials which generate many smooth values and so reduce the effort required in the time consuming sieving step. Brian Murphy considered being the first how deeply study the effect of choosing a good polynomial on the time needed to factor a number n in his PhD thesis, he came out with a parameter $\alpha(f)$ to measure without sieving the quality of the polynomial ,many programs have been written using Murphy parameter $(\alpha(f))$ and iterate on the leading coefficient of the base-m polynomial till reaching the required value which we intend to reach, we get use from a program freely available on the net which help us to choose a good polynomial after feeding the program with the required parameters .

The matrix step is also troublesome. An advantage of better polynomial selection is that the saving in sieving time is sufficient that, in effect, sub-optimal smoothness bounds can be chosen to decrease the matrix size.

The main aim in doing so is to decrease sieving times.

In this paper we try to compare the time consumed to factor a 100 digits integer by two different degrees polynomials (fourth and fifth degree) using two GNU programs named GGNFS and MSIEVE thought a python script named factmsieve.py written by Brian Gladman's , we presents the result in appendixes (appendix A & appendix B).

GGNFS & MSIEVE - A Number Field Sieve implementation

GGNFS is one of the earliest public domain implementations of the General Number Field Sieve, MSIEVE [17] is another open source project for implementing the General Number Field Sieve (GNFS), as a result many people involved in public factoring projects are using the two programs together in order to obtain the best features of both, To make things easier on Windows in particular a python script named (factmsieve.py) has been developed by Brian Gladman's which is a Python driver for GGNFS and MSIEVE, we use this script (factmsieve.py) to factor a 100 digits integer n and compare the time needed to factor n using two different degree polynomials.

Factoring a 100 digits integer by GNFS

We are going to factor the following 100 digit integer (n) using the General Number Field Sieve (GNFS) with Brian Gladman's factmsieve.py python script which uses both GGNFS and MSIEVE tools:

n =288103982745789597188162705313753073463879082516612749606667432024157144649476238662044295382073545 3.

The GNFS algorithm uses a 3 phase's method, the first being polynomial selection, then sieving, and finally linear algebra. Before factoring can begin, a polynomial must be selected. The factmsieve.py script will run the appropriate tool to select a polynomial which is the first step in the algorithm, in this range of 100 digits, we are right around the break-point between wanting degree 4 and degree 5, we use factmsieve.py script in two ways, first by choosing to use a 5th degree polynomial as a first step in factoring process and secondly by choosing a 4th degree polynomial as a first step in the factoring process and compare the time needed to factor (n) , we find that by using a 4th degree polynomial we factor a 100 digits number faster than a 5th degree polynomial by a factor of 0.47 hour , the time needed to factor the 100 digits number n using 5th degree polynomial equal to 2.78 hours while we factor the same number using a 4th degree polynomial in a 2.29 hour the divisor of the number n founded are :

r1=6181628341868659693893363741554871982772656 79 (pp45)

r2=4660648728983566373964395375209529291596595400646068307 (pp55)

The 5th degree polynomial used to factor a 100 digits integer

We list down the coefficients of the 4th degree polynomial which have been used to factor n.

c0: -38188640167192634569245

c1: 71784960098507618961

c2: -9629246254240576

c3: -14887528966064

c4: -1300662016

c5: 204120

The 4th degree polynomial used to factor a 100 digits integer

We list down the coefficients of the 4th degree polynomial which have been used to factor n.

c0: 158848928508311969623933113 59

c1: 54279151636160866273417

c2: -19225420015742387

c3: 191972171

c4: 840

CONCLUSIONS

selecting a polynomial is The first and the most important step of the general number field sieve(GNFS), literally there can be billions of choices for the polynomials .From real-world experiments with the GNFS it has been seen that in many cases some polynomials are remarkably better than others, where a polynomial $f(x)$ is considered "better" than another polynomial $g(x)$ if more (a, b) pairs with $a + bm$ and $a + b\alpha$ smooth are found with $f(x)$ than with $g(x)$ using the same sieve region, factor base sizes, and quadratic character base sizes, Brian Murphy came out with a parameter $\alpha(f)$ to measure without sieving the quality of the

polynomial many programs have been written using Murphy parameter $(\alpha(f))$ and iterate on the leading coefficient of the base-m polynomial till reaching the required value which we intend to reach, we get use from a program freely available on the net which help us to choose a good polynomial after feeding the program with the required parameters. We find a 4th degree polynomial need less time than a 5th degree polynomial to factor an integer n of a 100 digits.

REFERENCES

[1] Bellare , M. A. Desa , E. Jokipii , P. Rogawayy, *" A Concrete Security Treatment of Symmetric Encryption: Analysis of the DES Modes of Operation "*, Proceedings of the 38th Symposium on Foundations of Computer Science, IEEE, 1997.

[2] Boneh, D. G. Di Crescenzo, R. Ostrovsky, and G. Persiano, *" Public key encryption with keyword search",* proceedings of Eurocrypt 2004, LNCS 3027, pp. 506-522, 2004 .

[3] Diffie.W. and M. Hellman, "***New directions in cryptography***", IEEE Transactions on Information Theory, No. 22 (1976), P (644-654).

[4] El-Gamal, T. "***A public key cryptosystem and a signature scheme based on discrete logarithms***", IEEE Transactions on Information Theory, No. 31 (1985), P (469- 472).

[5] Koblitz, N. "***Elliptic curve cryptosystems***", Mathematics of Computation, No. 48 (1987), P (203-209).

[6] Miller, V. "***Uses of elliptic curves in cryptography***", Advances in Cryptology-CRYPTO '85, Lecture Notes in Computer Science, Springer-Verlag, No. 218 (1986), P (417-426).

[7] Rivest, R. L. A. Shamir and L. Adleman, "***A method for obtaining digital signatures and public key cryptosystems***", Communication of the ACM, No, 21 (1978), P 120-126.

[8] Peter L. Montgomery, "***A Survey of Modern Integer Factorization Algorithms"***, CWI Quarterly, Volume 7 (4) 1994.pp 337 – 365.

[9] Kostas Bimpikis and Ragesh Jaiswal***,"Modern Factoring Algorithms"***, University of California, San Diego.

[10] Menezes, A. P. van Oorschot and S. Vanstone ***"Handbook of Applied Cryptography"*** , CRC Press, Inc , USA, 1997.

[11] Matthew E. Briggs, "***An Introduction to the General Number Field Sieve***" ,M.sc thesis, Virginia Polytechnic Institute and State University, Blacksburg, Virginia ,USA, 1998.

[12] Peter Stevenhagen, "***The number fields sieve***".Mathematical Sciences Research Institute Publications, Cambridge University Press, 2008.

[13]. Brian Antony Murphy, "***Polynomial Selection for the Number Field Sieve Integer Factorization Algorithm***", The Australian National University, PHD thesise, 1999.

[14] Buhler, J. H. W. Lenstra, Jr. and C. Pomerance, "***Factoring integers with the number field sieve***", The development of the number field sieve, A. K. Lenstra

and H. W. Lenstra, Jr., eds., Lecture Notes in Math. 1554, pp. 50–94, Springer-Verlag,Berlin, 1993.

[15] Carl Pomerance, "***A Tale of Two Sieves***", the Notices of the American Mathematical Society (AMS), DECEMBER 1996, pp. 1473

[16] Pavol Zajac," ***REMARKS ON THE NFS COMPLEXITY"***, Tatra Mt. Math. Publ. 41 (2008), 79-91, Bratislava, SLOVAKIA

[17] Papadopoulos, J. msieve, http://sourceforge.net/projects/msieve/

Appendix A: Factoring a 100 digits integer using 5th degree polynomial by GGNFS & MSIEVE free software.

Number: example

N = 288103982745789597188162705313753073463879082516612749606667432024157144649476238662044295382073545 3 (100 digits)

Divisors found:

r1=6181628341868659693893363741554871982772656 79 (pp45)

r2=4660648728983566373964395375209529291596595400646068307 (pp55)

Version: Msieve v. 1.48

Total time: 2.78 hours.

Factorization parameters were as follows:

name: example

n: 288103982745789597188162705313753073463879082516612749606667432024157144649476238662044295382073545 3

skew: 3780.87

norm 1.16e+14

c5: 204120

c4: -1300662016

c3: -14887528966064

c2: -9629246254240576

c1: 71784960098507618961

c0: -38188640167192634569245

alpha -6.1

Y1: 12865768213

Y0: -6759799176610485742

Murphy_E 3.54e-09

#M 240869040966887297249882522310401210868018415971057420663293912153490808276484920003347431431043 2048

type: gnfs

rlim: 1800000

alim: 1800000

lpbr: 26

lpba: 26

mfbr: 48

mfba: 48
rlambda: 2.5
alambda: 2.5
qintsize: 100000
Factor base limits: 1800000/1800000
Large primes per side: 3
Large prime bits: 26/26
Sieved algebraic special-q in [0, 0)
Total raw relations: 4662917
Relations: 259460 relations
Pruned matrix : 164760 x 164985
Polynomial selection time: 0.50 hours.
Total sieving time: 2.15 hours.
Total relation processing time: 0.04 hours.
Matrix solve time: 0.06 hours.
time per square root: 0.03 hours.
Prototype def-par.txt line would be: gnfs,99,5,58,1500,0.003,0.4,220,15,10000,2000,1800000,1800000,26,26,48,48,2.5,2.5,100000
total time: 2.78 hours.
x86 Family 6 Model 23 Stepping 6, GenuineIntel
Windows-XP-5.1.2600-SP2
processors: 2, speed: 2.53GHz

Appendix B: Factoring a 100 digits integer using 4th degree polynomial by GGNFS & MSIEVE free software.

Number: example
N = 288103982745789597188162705313753073463879082516612749606667432024157144649476238662044295382073545 3 (100 digits)
Divisors found:
r1=6181628341868659693893363741554871982772656 79 (pp45)
r2=4660648728983566373964395375209529291596595400646068307 (pp55)
Version: Msieve v. 1.48
Total time: 2.29 hours.
Factorization parameters were as follows:
n:
288103982745789597188162705313753073463879082516612749606667432024157144649476238662044295382073545 3
Y0: -136086903206958504042 5502
Y1: 64064546057551
c0: 158848928508311969623933113 59
c1: 54279151636160866273417
c2: -19225420015742387
c3: 191972171

c4: 840
Skew: 4053539.86
Type: gnfs
Factor base limits: 1800000/1800000
Large primes per side: 3
Large prime bits: 26/26
Sieved algebraic special-q in [0, 0)
Total raw relations: 4532684
Relations: 307082 relations
Pruned matrix: 186767 x 186993
Polynomial selection time: 0.35 hours.
Total sieving time: 1.81 hours.
Total relation processing time: 0.03 hours.
Matrix solves time: 0.08 hours.
Time per square root: 0.02 hours.
Prototype def-par.txt line would be: gnfs,99,4,58,1500,0.003,0.4,220,15,10000,2000,1800000,1800000,26,26,48,48,2.5,2.5,100000
Total time: 2.29 hours.
x86 Family 6 Model 23 Stepping 6, GenuineIntel
Windows-XP-5.1.2600-SP2
Processors: 2, speed: 2.53GHz