UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

STUDY OF MATRICES RELATED TO DISCRETE LOGARITHM IN KUMMER EXTENSIONS

A THESIS

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

MASTER OF SCIENCE

By

XUECHENG CHEN
Norman, Oklahoma

2016

STUDY OF MATRICES RELATED TO DISCRETE LOGARITHM
IN KUMMER EXTENSIONS

A THESIS APPROVED FOR THE
SCHOOL OF COMPUTER SCIENCE

BY

Dr. Qi Cheng, Chair

Dr. Sridhar Radhakrishnan

Dr. S. Lakshmivarahan

Acknowledgements

First of all, I'd like to thank my advisor, my teachers, my fiends and my family for the encouragement and support in pursuing this degree.

With the help of my advisor, Dr. Cheng, I learned so much in knowledge and research in computer science, especially in topics related to security aspect. His solid knowledge base in cryptography and number theory helped me gain sights in this field and his critical thinking of the questions affected me in research methods.

I also want to thank my committee members and teachers in our department for the help of different topics which are interesting and useful. As a transfer student from other major, I have a lot of things to learn about computer science. The courses and activities in thee department helped me to fill the gaps and move on to the new major.

During the last two and half years, my friends and family was always encouraging me in the school and life. My parents cared about me for the health and passion all the time and my wife has always been accompanying with me and built a sweet home together with me.

At last I'd like to mention the tornadoes and earthquakes in Oklahoma, which are unique experiences for an international student who has never seen these before. I'll always remember the great football team , the peaceful campus, and foremost

the university of Sooner and Boomer.

Contents

List of Tables

List of Figures

Abstract

The discrete logarithm problem has been studied during the past decades for its important application in cryptography and other fields. It is very useful in the public key cryptography, which is widely used for Internet safety. Using current computers to solve general discrete logarithm problem seems still not possible within reasonable time, since no polynomial time algorithms has been found for general cases. However, over finite fields of small characteristic, the factor base discrete logarithm can be solved much faster with heuristic polynomial time algorithms.

This thesis is mainly based on the previous study of factor base discrete logarithm in Kummer extension ($\mathbb{F}_{q^{2(q-1)}}$) which is published recently by Xiao-Zhuang-Cheng [14] and we focused on further calculation in this study. The previous research, based on the hypothesis of the determinant of lattices and the discrete logarithm, was confirmed with calculation for all q's such that $\log_2(q^{2(q-1)}) \leq 5000$, and in this thesis we pushed the limit to $\log_2(q^{2(q-1)}) \leq 10000$. During the calculation, we tried different strategies to improve the efficiency, by transferring the matrices and splitting q's into several groups. We achieved 1000% speed-up for most q's in the range and discovered some possible structures to group q's in calculation.

In the thesis, we'll go through the basic backgrounds of the study, and then in-

troduce the main methods and experiments done in the study. We'll discuss the grouping of q's and the efficiency improvement. In the end we'll summarize the progress and the possible future work of this study.

Keywords: Cryptography, Discrete logarithm, Kummer extension, Lattice

Chapter 1

Introduction

1.1 Backgrounds

In the past several decades, public key cryptography was heavily based on discrete logarithm study [5], and we believe that it still takes longer than reasonable time to solve the general cases using current computing method and resources [13]. Although new methods were invented and improved as time went on, this assumption holds unless quantum computers are developed with productivity. However, some recent study showed that if the characteristic of the field is small, the calculation can be accelerated significantly [6, 7, 8, 9, 10, 11, 12]. With the help of index calculus, function field sieve and number field sieve [1, 2], we collect linear relations for the discrete logarithm, and then solve the discrete logarithm with the relations. The smoothness of the polynomial affects the efficiency if exhaustive search is used, but guided searching algorithm can accelerate the process with loss of correctness to assumptions of smoothness [3, 4].

In the public key cryptography, an important part is the Diffie-Hellman key

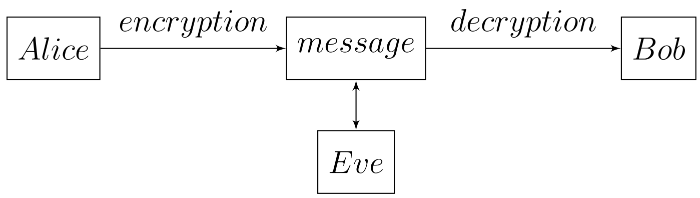

Figure 1.1: Cryptography in communication

exchange based on discrete logarithm. As is shown in Figure 1, if Alice and Bob wants to share the key in a safe channel, they can use this algorithm to get same secret while Eve can't solve it.

Assume Alice and Bob agree to use g and p as the base and modulus. At first Alice hold the secret a and Bob hold the secret b. Alice then transfers $A = g^a$ mod p to Bob and Bob transfers $B = g^b$ mod p to Alice. Now Alice can get the secret $s = B^a$ mod p and Bob can get the same secret $s = A^b$ mod p. Eve can see the g, p, A and B but can't get the secret s. Here we can see the algorithm relies on the difficulty of discrete logarithm with the current computing resources.

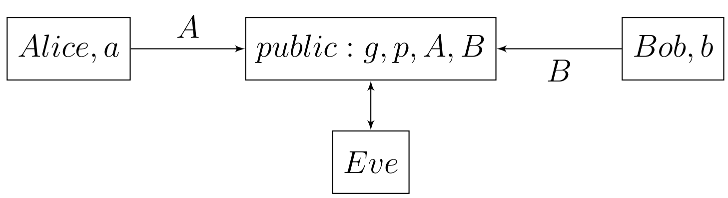

Figure 1.2: Diffie-Hellman public key exchange

In the previous research, matrices related to discrete logarithm in Kummer extension were studied in order to solve the factor base discrete logarithm problems [14]. The research went through the new algorithm from building the matrices related to the index calculus to solving and verifying the determinant of resulting matrices. The detailed steps will not be discussed in this study but main processes and definitions will be introduced in the later chapters.

1.2 Motivation

The previous research provided theoretical evidence to support the algorithm, as well as verified the results for all q's such that $\log_2(q^{2(q-1)}) \leq 5000$. However, it takes quite a long time to reach that limit with the current computers. Since the transformation of matrices related to the lattice is not suitable for parallel computing, it's important to speed up the process by shortening the time of each step. There are several ways of improving the calculation speed and the most efficient one we found was to reduce the size of elements in the matrices during the beginning period. However, it was not always working on current computers due to some possible special structures in the matrices. So we tried to reduce the matrix size by partitioning field at the beginning, and this worked for the outliers from the first case. We also try to minimize the space usage if it's possible although the new algorithm sacrifices space for time in most cases.

1.3 Progress

In this study, the main improvement from the previous work is to push the limit of the q's to $\log_2(q^{2(q-1)}) \leq 10000$ with several attempts in different aspects. The direct speed up from the new calculation of Hermite Normal Form accelerated the calculation by more than 1000% for eligible q's. For the outliers, the new calculation could not be handled by our computers, but by dividing those q's into cases and transforming the calculation to smaller matrices, we also got more than 1000% speed-up. As for the space usage, most cases were affordable in both cases, although the new algorithm has a weaker relation of space usage and q's. The calculation was done on a computer with 3.6GHz CPU and 32G memory.

We used SAGE under LINUX system for the study.

Chapter 2

Methods

2.1 Definitions

2.1.1 Finite Fields

We'll start the definitions from the group theory.

Definition 2.1. Group

The group is a structure of set of elements (G) and operations($\bullet$). To form a group, the operation need to be used on the elements and generate an element with the following properties:

Closure: $\forall$ a, b $\in G$, a $\bullet$ b $\in G$

Associativity: $\forall$ a, b, c $\in G$, (a $\bullet$ b) $\bullet$ c = a $\bullet$ (b $\bullet$ c)

Identity: $\exists$ e, $\forall$ a $\in G$, a $\bullet$ e = e $\bullet$ a = a

Invertibility: $\exists$ b, $\forall$ a $\in G$, a $\bullet$ b = b $\bullet$ a = e

From the definitions above, we denote the e as the identity element and $b = a^{-1}$ as the inverse of a. If the invertibility doesn't hold, we call that a monoid. If further the identity doesn't hold, we call that a semigroup.

Definition 2.2. Abelian Group

The abelian group is a group with the following property:

$\forall$ a, b $\in G$, a $\bullet$ b = b $\bullet$ a

With these definitions, we can see that integers ($\mathbb{Z}$) with addition (+) forms an abelian group $(\mathbb{Z}, +)$. If we remove 0 from $\mathbb{Q}$ (denoted as $\mathbb{Q}^* = \mathbb{Q} \setminus \{0\}$), it also forms an abelian group with multiplication $(\mathbb{Q}^*, *)$, otherwise it's a monoid.

Definition 2.3. Ring

The ring is a set(R) with two binary operations ($\bullet, \circ$) satisfying the following properties:

$(R, \bullet)$ is an abelian group.

$(R, \circ)$ is a semigroup.

Distributive law holds: a $\circ$ (b $\bullet$ c) = (a $\bullet$ b) $\circ$ (a $\bullet$ c)

For integers it's easy to see that with + and $*$ we can form an integer ring ($\mathbb{Z}$, +, *). For example, $(\mathbb{Z}/m\mathbb{Z}, +, *)$ is an integer ring where $(\mathbb{Z}/m\mathbb{Z}) = \{0, 1, 2, ..., m-1\}$. We can easily verify the properties needed above.

Definition 2.4. Zero divisor

A zero divisor in the ring ($\mathbb{R}$) is, a non-zero element a $\in \mathbb{R}$, $\exists b \neq 0$ such that b * a = 0.

For example, in $(\mathbb{Z}/6\mathbb{Z}, +, *)$, 2 is a zero divisor since 2 * 3 = 0 in $(\mathbb{Z}/6\mathbb{Z}, +, *)$. Also, 3 and 4 are zero divisors in this ring but 5 is not because it has inverse of 5. Zero divisor doesn't have multiplicative inverse.

Definition 2.5. Field

A field $\mathbb{F}$ in number theory is usually defined as a ring in which each nonzero element has its multiplicative inverse.

For example, $(\mathbb{Q}, +, *)$ is a field since every element except 0 has its inverse. For prime number p, $(\mathbb{Z}/p\mathbb{Z}, +, *)$ is a field. Let p = 3, $(\mathbb{Z}/3\mathbb{Z}, +, *) = \{0, 1, 2\}$. We can easily see that 1 has its inverse 1 and 2 has its inverse 2.
For non-prime number m, $(\mathbb{Z}/m\mathbb{Z}, +, *)$ is not a field. Let m = 4, $(\mathbb{Z}/4\mathbb{Z}, +, *) = \{0, 1, 2, 3\}$. We can see that 2 doesn't have its inverse.

Definition 2.6. Characteristic of field
The characteristic of a field $\mathbb{F}$, $Ch(\mathbb{F})$ is defined as 0 or the smallest positive integer n such that $n * 1_{\mathbb{F}} = 0$ if it exists.

Definition 2.7. Finite field
A finite field is a field with finite number of elements.
Denote $\mathbb{F}_q$ be a finite field with q elements, where q is a prime number or a prime power.

With these basic field definitions, now we can describe the discrete logarithm problem.

2.1.2 Discrete logarithm

Discrete logarithm has been used in public key cryptography for a long time, and the definition can be described as follow:

Definition 2.8. Discrete logarithm over finite field.
For $\mathbb{F}_q$, if $\alpha^x = \beta$ was given for $\alpha, \beta \in \mathbb{F}_q^*$, the discrete logarithm problem is to find the integer x.

Here's an example of a simple discrete logarithm problem:

x	1	2	3	4	5	6
β	3	2	6	4	5	1

Table 2.1: A simple discrete logarithm in $\mathbb{F}_7$ when $\alpha = 3$

Here with the table we can find x for given β, but when the field is much larger, it'll take very long time since no polynomial algorithm has been found.

As is mentioned above, the general algorithms solving discrete logarithm problem are sub-exponential time complexity, such as number field sieve and function field sieve.

However, special cases with small characteristic field can be solved much faster. In these cases, the Kummer extension is very useful in testing the algorithms due to its structure. So we focus on these cases in the study.

2.1.3 Kummer Extensions

When we try to solve the discrete logarithm problem, the Kummer extension could be a starting point although it's considered not safe enough to be used in real world cryptography. The Kummer extension can be described as follow:

Definition 2.9. Kummer extension:

In general, a field extension L/K is called a Kummer extension if for some given integer $n > 1$:

K contains n distinct n-th roots of unity.

L/K has abelian Galois group of exponent n.

For example, the Kummer extension $\mathbb{F}_{q^{2(q-1)}}$ could be very interesting since the recent breakthrough suggests fast algorithms in this field and the right hand of the relations would be automatically linear. In this study, $\mathbb{F}_{q^2}[x]/(x^{q-1} - A)$ is used to model the Kummer extension where $A \in \mathbb{F}_{q^2}$ and $x^{q-1} - A$ is irreducible over $\mathbb{F}_{q^2}$.

2.1.4 Lattice

Lattice study is growing popular because of its resistance to quantum computers. In this study, we built the $(q+1)$-dimensional lattice for the conjecture and calculated the determinant related to the matrices.

Definition 2.10. Lattice:

A lattice in $\mathbb{R}^m$ is a subgroup of $\mathbb{R}^m$ and it's isomorphic to $\mathbb{Z}^m$.

In this study, it's defined as

$$\mathcal{L} = \{\sum_{i=1}^{n} x_i \mathbf{b}_i | x_i \in \mathbb{Z}\},$$

where $n \leqslant m$ and $\mathbf{b}_1, \mathbf{b}_2, ...\mathbf{b}_n$ are linearly independent and $\mathbf{b}_i \in \mathbb{R}^n$ for $1 \leqslant i \leqslant n$.

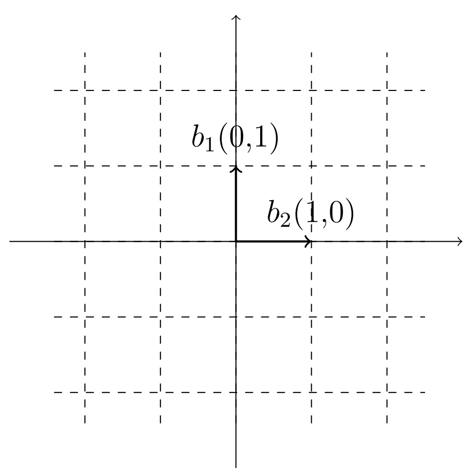

Figure 2.1: Lattice generated by $b_1(0, 1)$ and $b_2(1, 0)$

If m = n in the lattice defined above, the set of vectors $\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_n$ is called

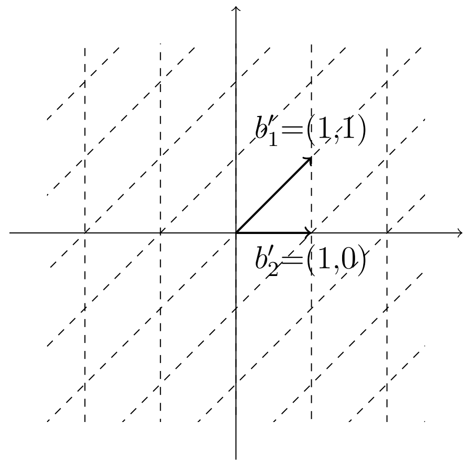

Figure 2.2: Lattice generated by $b'_1(1,1)$ and $b'_2(1,0)$

a lattice basis and it can be represented as

$$B = [\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n],$$

Definition 2.11. Determinant of a lattice:
For a lattice $\mathcal{L} = \mathcal{L}(B)$, the determinant of the lattice($det(\mathcal{L})$), is the n-dimensional volume of the fundamental parallelepiped $P(B)$.

Now we can go on to the matrices used in the study and explain the transformation chosen on the matrices representing the lattice.

2.1.5 Normal forms

In linear algebra, normal forms are used for matrix transformation in order to generalize and clarify the information of the matrix. Here are the two normal forms used in the study and some transforming methods.
Here we only use row transformation in the study, so the normal forms are transformed by rows unless otherwise specified.

Definition 2.12. Gaussian elimination:

The Gaussian elimination is an operation to change the left lower corner of the matrix into zeros by swapping rows, multiplying rows by non-zero numbers, or adding rows or multiplied rows to another.

Definition 2.13. (Row) Echelon form:

The resulting matrix from the Gaussian elimination is an echelon form.

Definition 2.14. GaussJordan elimination:

If we add the following restriction to Gaussian elimination, it's called Gauss-Jordan elimination: the resulting matrix has leading coefficient of 1 for all rows, and they are the only non-zero numbers in their columns.

Definition 2.15. Reduced (row) echelon form:

The resulting matrix of GaussJordan elimination is called the reduced echelon form. Each matrix only has one unique reduced echelon form.

Note that in this study, row echelon form is used in all calculation if not specified in the following.

Definition 2.16. Hermite normal form:

The Hermite normal form is basically similar with reduced echelon form but it's over integers $\mathbb{Z}$. All transformations should be done over $\mathbb{Z}$. Note that only swapping or adding rows, or addingrows with nonzero integer multiplication are allowed in the transformation since solely multiplying rows with integers may change the determinant.

Definition 2.17. Smith normal form:

The Smith normal form is defined for matrices with entries in a principal ideal domain (such as $\mathbb{Z}$) and calculated by by multiplying the left and right invertible square matrices.

Here's an example of the transformation between normal forms (row transformation).

Say we have matrix A:

$$A = \begin{pmatrix} 1 & 2 & 3 & 9 \\ 2 & 5 & 6 & 7 \\ 3 & 4 & 5 & 6 \\ 4 & 7 & 8 & 9 \end{pmatrix}$$

Here when we transform the matrix to reduced row echelon form (E). It's calculated in $\mathbb{Q}$.

$$E = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Here we transform the matrix to Hermite normal form (H). It's calculated in $\mathbb{Z}$. Note that it has the same absolute determinant value with A.

$$H = \begin{pmatrix} 1 & 0 & 3 & 1 \\ 0 & 1 & 0 & 4 \\ 0 & 0 & 4 & 3 \\ 0 & 0 & 0 & 5 \end{pmatrix}$$

Now we transform it into Smith normal form(S). Here we can see

S=$S_{left} * A * S_{right}$.

$$S = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 20 \end{pmatrix}, S_{left} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & -3 & -2 \\ -4 & -1 & 14 & 11 \end{pmatrix}, S_{right} = \begin{pmatrix} -2 & -1 & 3 & 15 \\ -1 & -4 & 5 & 24 \\ 2 & 4 & -7 & -33 \\ 0 & 0 & 1 & 4 \end{pmatrix}$$

The previous study was done mainly in Smith normal form, but we changed the verification calculation into Hermite normal form in this study. We'll explain the reason of the change in the later chapters.

2.2 Calculation

2.2.1 Matrix transformation

The first step is to prepare for parameters used in the calculation.

We start from the q used for the verification and calculate the N with the following method:
1. Calculate $q^{2(q-1)} - 1$.
2. Remove prime factors from $q^{2(q-1)} - 1$ until the smallest factor is larger than q^2.

Then we build the field with N from above. Now we can form the matrices need in the calculation:

$$G(x^k) = x^k \sum_{\alpha \in \mathbb{F}_q} x^{\log_g (g+\alpha)}$$

$$T(x^k) = x^k x^{\log_g \frac{Ag^{k(q-1)}-1}{g^q - g}}$$

The next step is to get $M = GT$ and relate it to the lattice. Here we define $\tilde{M}$ with a transformation from $\mathbb{F}_{q^2}$ to $\mathbb{F}_q$, and M_1 as in the matrix

$$M = U^{-1} \begin{pmatrix} M_0 & & & \\ & M_1 & & \\ & & \ddots & \\ & & & M_{q-2} \end{pmatrix} U$$

If we denote L as the map from an integer matrix to the lattice and construct the following:

$$\mathcal{L}_1 = L(\tilde{M}_1 - I) + N\mathbb{Z}^{q+1}.$$

Then We have $N|\det(\mathcal{L}_1)|N^{q+1}$.

The hypothesis need to verify $\det(\mathcal{L}_1) = N$ with q's as large as possible.

2.2.2 Example q = 7

Here's an example when q = 7.

First, we calculate N.

$$q^{2(q-1)} - 1 = 2^5 * 3^2 * 5^2 * 13 * 19 * 43 * 181$$

After removing factors no larger than $q^2 = 49$, we got N = 181.

Then we find the polynomial G.

$$G_{poly} = x^{38} + x^{36} + x^{31} + x^{11} + x^5 + x^2 + x$$

And we get $\tilde{G}$ from G with modulus of $x^{q+1} - q^2$.

$$\tilde{G}_{poly} = 180x^7 + x^5 + 132x^4 + 49x^3 + x^2 + x$$

α	0	1	2	3	4	5	6
DL	1	5	38	36	2	11	31

Table 2.2: Discrete logarithm of $g + \alpha$ in G

Next step is to build the matrix $\tilde{G}$ and $\tilde{T}$.

$$\tilde{G} = \begin{pmatrix} 0 & 1 & 1 & 49 & 132 & 1 & 132 & 180 \\ 132 & 0 & 1 & 1 & 49 & 132 & 1 & 132 \\ 133 & 132 & 0 & 1 & 1 & 49 & 132 & 1 \\ 49 & 133 & 132 & 0 & 1 & 1 & 49 & 132 \\ 133 & 49 & 133 & 132 & 0 & 1 & 1 & 49 \\ 48 & 133 & 49 & 133 & 132 & 0 & 1 & 1 \\ 49 & 48 & 133 & 49 & 133 & 132 & 0 & 1 \\ 49 & 49 & 48 & 133 & 49 & 133 & 132 & 0 \end{pmatrix}$$

$$\tilde{T} = \begin{pmatrix} 0 & 0 & 180 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 48 & 0 \\ 180 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 180 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 132 \\ 0 & 132 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 48 & 0 & 0 \end{pmatrix}$$

Now we calculate M and append the diagonal matrix of N.

$$\tilde{M}_1 = \begin{pmatrix} 179 & 48 & 0 & 132 & 132 & 133 & 48 & 132 \\ 180 & 131 & 49 & 180 & 49 & 1 & 0 & 48 \\ 0 & 48 & 47 & 180 & 1 & 48 & 1 & 133 \\ 49 & 133 & 132 & 180 & 1 & 1 & 49 & 132 \\ 48 & 132 & 48 & 49 & 180 & 180 & 180 & 132 \\ 132 & 132 & 133 & 48 & 132 & 47 & 49 & 0 \\ 48 & 0 & 132 & 132 & 133 & 48 & 131 & 48 \\ 133 & 48 & 132 & 48 & 49 & 0 & 180 & 179 \\ 181 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 181 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 181 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 181 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 181 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 181 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 181 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 181 \end{pmatrix}$$

At last we calculate the Hermite normal form of the matrix above. The resulting matrix will be as follow:

$$HNF = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 155 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 141 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 59 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 170 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 77 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 35 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 46 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 181 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

2.2.3 Verification

We follow the steps above to verify whether $\det(\mathcal{L}_1) = N$. In the previous study, all q's such that $\log_2(q^{2(q-1)}) \leq 5000$, which means $q \leq 307$, was verified. Here we increased the q's until $\log_2(q^{2(q-1)}) \leq 10000$, which means $q \leq 613$, and changed the algorithm of verification.

As for the output, we expect to get the diagonal of the resulting matrix as $[1, 1, 1, ...N]$, and only printed the diagonal instead of the full matrix which should be as follow:

$$HNF = \begin{pmatrix} 1 & & & x_1 \\ & 1 & & x_2 \\ & & \ddots & \\ & & & N \end{pmatrix}$$

2.2.4 Modification

The first change to the algorithm is using Hermite normal form instead of Smith normal form during the calculation. The reason is that we tested different combinations of normal form and transformation, and then found that the Smith normal form is relatively stable under transformation. In other words it's not suitable for the speed up. For the Hermite normal form, we found different speed when we attach the diagonal matrix of N to the $M = GT$ in different directions. Then we picked the faster one as the new algorithm.

The second change is only made for the outliers which doesn't work with the new algorithm. Due to limited time and space, some q's exceeded our computing resources when we apply the new algorithm. So we divided q's into several cases and treated the outliers with different algorithm. According to known factorization of specific structures, we divided the N's into smaller integers, and then applied the old algorithm on those partitions. The other steps are exactly the same.

2.2.5 Evaluation

The main concern of the study is the time efficiency of the verification, so we set a series of reading of CPU time during the calculation. The threshold of the whole process was the calculation of the Hermite normal form. To compare the efficiency of this part, we record the CPU time separately for different calculations.

We also considered the space consumption during the verification. Since some of the q's took way too much memory and caused overflow, we compare the memory usage in different cases.

Chapter 3

Results

3.1 Correctness

All q's such that $311 \leq q \leq 613$ were verified in addition to previous q's such that $q \leq 307$. Although some of the q's doesn't fit in the new algorithm, they were all verified by the second modification. Also, the outliers could be divided into three cases as shown in Table 3.1. According to $q-1$'s factors, we can see that they contain small factors such as 2 or 4, and another large prime factor. But not all the q's with these structures are outliers. For example, 317 and 383 has similar structures but they can go through the new algorithm.

For these outliers of the new algorithm, we partitioned the N into several factors with known factorization. Then we ran the smallest partition with old algorithm and other partitions with new algorithm. All the partitions returned

$4 \mid q-1$	389, 557
$2 \mid q-1$	467, 479, 503, 587
others	311, 313,... 613

Table 3.1: Grouped prime numbers $311 \leq q \leq 613$

correct relation and the product of the determinants is exact N.

We are going to show a sample of this situation, q = 389, in the third section in this chapter.

3.2 Efficiency

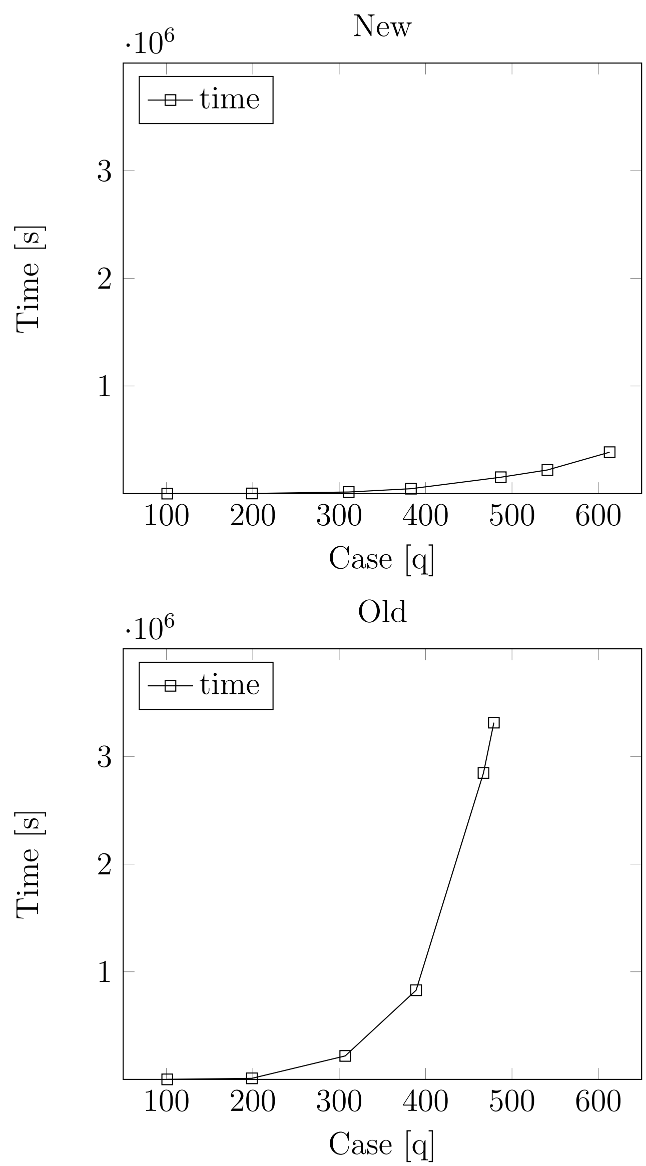

Figure 3.1: Selected HNF time of new and old algorithms

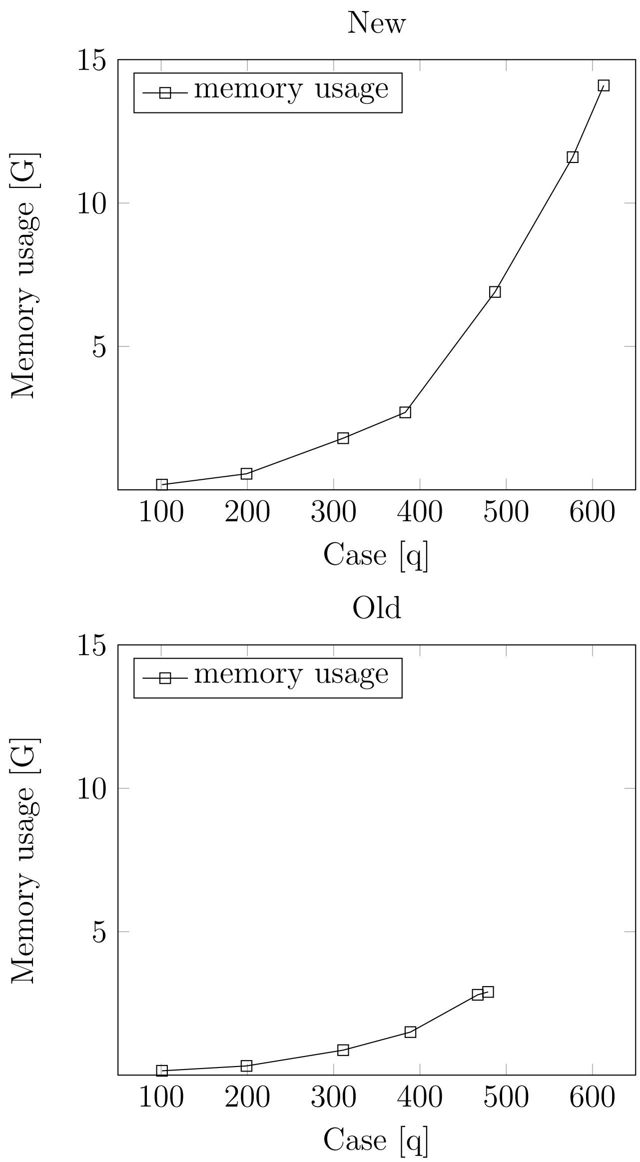

Figure 3.2: Selected memory usage of new and old algorithms

As is shown in Table 3.2, the largest part of time used in calculation is to calculate the HNF. Also, this study doesn't change the calculation method of getting G,T and M. So the efficiency evaluation in this section was referring to HNF time.

The general speed up of the algorithm is more than 1000% and vary with N's in

Time to get	Time s
N	0.644
G, T	873.816
M	40727.372
HNF	384903.052

Table 3.2: Calculation time of different steps in case q = 613

the new algorithm. The threshold of the outliers are using the old algorithm to calculate the smallest partition, but we still found more than 1000% speed-up. With this improvement, we took similar time computing the case where $q = 307$ with old algorithm and $q = 613$ with new algorithm.

3.3 q = 389

The first outlier of the new algorithm is q = 389.

Since $4 \mid q - 1$, we partitioned N into N_1 to N_4 by dividing q as follow: The original method calculates N from $389^{(2*389-2)} - 1$, which could be factored as $389^{388} + 1$, $389^{194} + 1$, $389^{97} + 1$, and $389^{97} - 1$.

Each partition of N is verified individually but we choose different algorithm for each partition. The first three partition are eligible for the new algorithm while the last partition only can be done by the old algorithm. However, due to the decrease of size N, the last part finished much faster than using the old algorithm on original N. The space consumption almost doubled using the partitioning method even without parallel computing. But it was still affordable at this data size.

	HNF time old	HNF time new
$389^{388}+1$	around 270000	around 10000
$389^{194}+1$	around 90000	around 3000
$389^{97}+1$	around 30000	around 1000
$389^{97}-1$	around 30000	unfinished

Table 3.3: Partitions and calculation time of q = 389

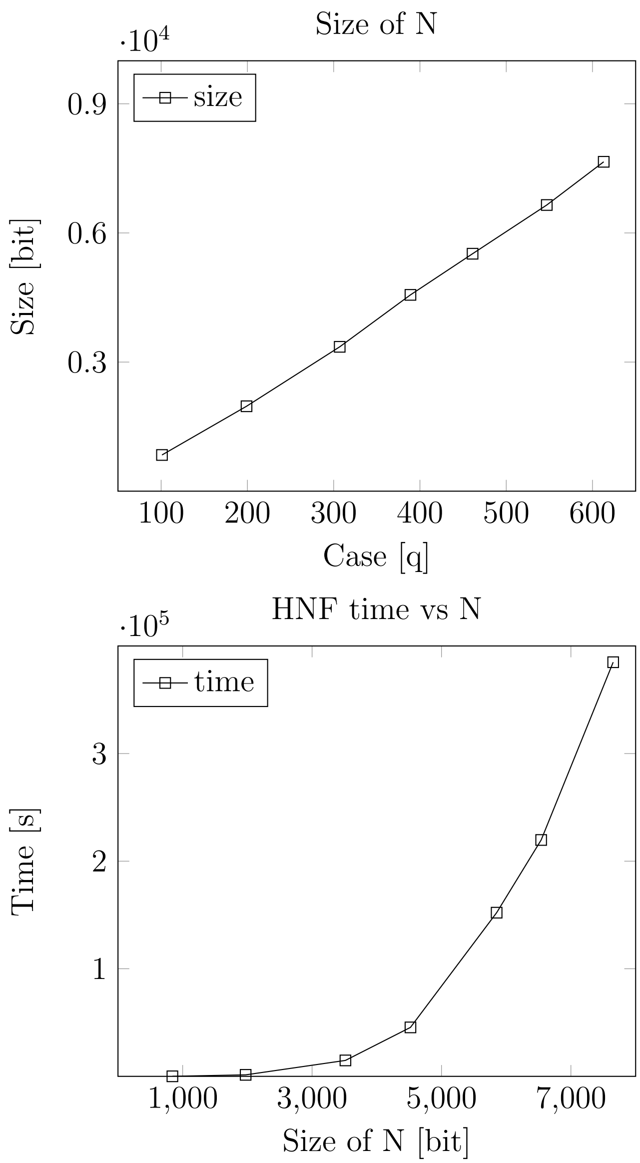

Figure 3.3: Size of N and HNF time vs Size of N

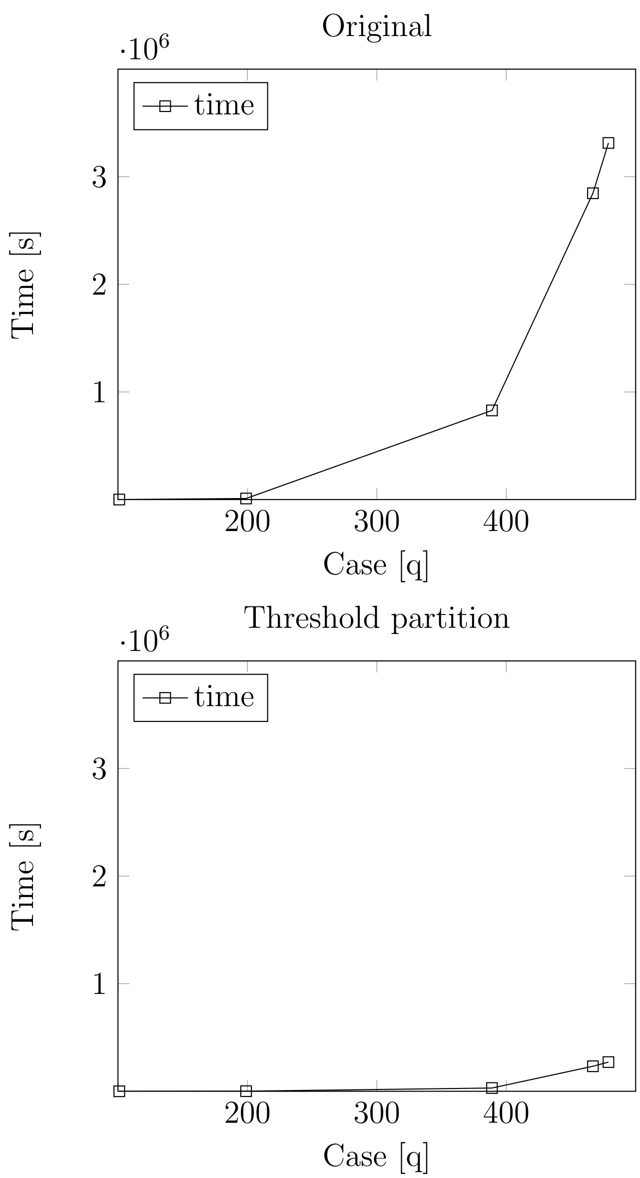

Figure 3.4: HNF time of threshold partitions and original

Chapter 4

Discussion

4.1 Matrices structure

During the calculation, the matrices M could be built in different ways. The previous algorithm appended GT after diagonal matrix with N and the new algorithm choose opposite direction. The difference between the time usage was great: the new algorithm is much faster for eligible q's. However the space usage increased a bit.

$$M_1 = \begin{pmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \\ N & & \\ & N & \\ & & N \end{pmatrix}$$

$$M_2 = \begin{pmatrix} N & & \\ & N & \\ & & N \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{pmatrix}$$

The possible reason for these results could be the default calculation process of the Hermite Normal Form in Sage. Elements in GT might be somehow sparse for eligible q's, which could end up with smaller space usage at the beginning and sharp increase by the end. Space usage is unstable with each q's structure. Smaller elements during the most calculation period reduced the time, but it also varies with q'.

The non-eligible q's could have much more dense structures during the calculation, So the new algorithm with unstable space usage might cause overflow on our computer. But the old algorithm could solve those cases even without partitioning, using longer time.

q	factorization of q-1
311	2 * 5 * 31
317	2 * 2 * 79
383	2 * 191
389	2 * 2 * 97
467	2 * 233
479	2 * 239
487	2 * 3 * 3 * 3 * 3 * 3
503	2 * 251
557	2 * 2 * 139
587	2 * 293
613	2 * 2 * 3 * 3 * 17

Table 4.1: q's and factor of $q-1$

4.2 Grouping q's

The outliers of the new algorithm could be grouped by their factors, since all cases was found to contain only 2's or 4's with another large prime factor of $q-1$'. The cause of the overflow during the matrix solving might be related to those large factors but that is not necessarily related.

If we could try different sequences during the calculation of solving the Hermite Normal Form in the future, we might find the relation to the structures and q's. For example, if printing the matrix during the transformation is possible, we may find out the size of elements and distribution of sizes.

Chapter 5

Conclusion

In summary, this study is an expansion of cases in the previous paper and improvement of algorithm from the old one. The speed-up was significant and space usage was reasonable. It's interesting to find the different cases of q's during the calculation.

In the future, we can look into the cause of grouping with more theoretical mathematics. Also, new way of verifying the determinant would be great if faster calculation or parallel computing could be introduced to the study.

References

[1] Adleman, L.: A subexponential algorithm for the discrete logarithm problem with applications to cryptography. In: Foundations of Computer Science, 1979., 20th Annual Symposium on. pp. 5560. IEEE (1979)

[2] Adleman, L.: The function field sieve. In: Algorithmic number theory ANTS I. Lecture Notes in Comput. Sci., vol. 877, pp. 108121. Springer (1994)

[3] Barbulescu, R., Gaudry, P., Joux, A., Thomé , E.: A heuristic quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic. In: Advances in Cryptology - EUROCRYPT 2014. LNCS, vol. 8441, pp. 1-16. Springer (2014)

[4] Cheng, Q., Wan, D., Zhuang, J.: Traps to the BGJT-algorithm for discrete logarithms. LMS Journal of Computation and Mathematics 17, 218-229 (2014)

[5] Diffie, W., Hellman, M.: New directions in cryptography. IEEE Transactions on Information Theory 22(6), 644654 (1976)

[6] Gäloğlu, F., Granger R., McGuire G., Zumbrägel J.: On the function field

sieve and the impact of higher splitting probabilities. In: Advances in Cryptology - CRYPTO 2013. LNCS, vol. 8043, pp. 109-128. Springer (2013)

[7] Granger, R., Kleinjung, T., Zumbrägel, J.: On the powers of 2. Cryptology ePrint Archive, Report 2014/300 (2014)

[8] Granger, R., Kleinjung, T., Zumbrägel, J.: On the discrete logarithm problem in finite fields of fixed characteristic. Cryptology ePrint Archive, Report 2015/685 (2015)

[9] Joux, A., Lercier, R.: The function field sieve in the medium prime case. In: Advances in Cryptology EUROCRYPT 2006. Lecture Notes in Comput. Sci., vol. 4005, pp. 254270. Springer (2006)

[10] Joux, A., Lercier, R., Smart, N., Vercauteren, F.: The number field sieve in the medium prime case. In: Advances in Cryptology CRYPTO 2006, Lecture Notes in Comput. Sci., vol. 4117, pp. 326344. Springer (2006)

[11] Joux, A.: Faster index calculus for the medium prime case. Application to 1175-bit and 1425-bit finite fields. In: Advances in Cryptology EUROCRYPT 2013, Lecture Notes in Comput. Sci., vol. 7881, pp. 177193. Springer (2013)

[12] Joux, A.: A new index calculus algorithm with complexity $L(1/4 + o(1))$ in very small characteristic. Cryptology ePrint Archive, Report 2013/095 (2013)

[13] Pohlig, S., Hellman, M.: An improved algorithm for computing logarithms

over $GF(p)$ and its cryptographic significance. IEEE Transactions on Information Theory 24(1), 106-110 (1978)

[14] Xiao, D., Zhuang, J. and Cheng, Q.: Factor base discrete logarithms in Kummer Extensions, Cryptology ePrint Archive: Report 2015/859 (2015)

Appendix

Sage code

q=31 code

```
##########################################################################
# This is the code when q = 31.
# Just change the initialization for other q's.
##########################################################################
# initialize q and calculate N

q=31
Adlog = -1
print "q=",q
t=cputime()

N=q^(2*q-2)-1
for i in range(2,q^2):
          while N%i==0:
                    N=N/i
```

```
print "N=", N #calculate N = large prime factors of q^(2*q-2)-1
# print "=", factor(N)

################################################################################
# initialize RING AND FIELD

II=Integers(N)
R.<x>=Integers(N)[]
F2=GF(q^2,'g')
F2_gen=F2.multiplicative_generator()
F1_gen=F2_gen^(q+1)
A=F2_gen^Adlog
# A can be changed here such as A=F2_gen
# Adlog can be changed here such as Adlog=1

################################################################################
# find G

t=cputime()
k=0
for i in (F2):
        if i^q==i:
                l=F2_gen+i
                for j in range (q^2):
```

```
                if F2_gen^j==1:
                    k=k+x^j
                    break
G=k
print "G=",G

klog = []
# klog stores the logarithm of k
for i in range(q+1):
    k=(A*F2_gen^(i*(q-1))-1)/(F2_gen^q-F2_gen)
    for j in range (q^2-1):
        if F2_gen^j==k:
            klog.append(j)
            break



#########################################################################
# calculate HNF


def test(q,b,Sbase):
    t=cputime()
    #calculate Gx,Tx
    S = R.quotient(x^(q+1)-Sbase^b, 'a')
    XK=[]
    GK=matrix(R,q+1,q+1)
```

```
YK=[]
TK=matrix(R,q+1,q+1)

for i in range (q+1):
        XK.append(S(G*x^i))
        YK.append(S(x^(i+klog[i])))

for i in range(q+1):
        k=[]
        m=0
        for j in XK[i]:
                k.append(j)
                m+=1
        if m!=q:
                for j in range(m,q):
                        k.append(0)
        GK.set_row(i,k)

for i in range(q+1):
k=[]
m=0
for j in YK[i]:
        k.append(j)
m+=1
if m!=q:
```

```
        for j in range(m,q):
                k.append(0)
TK.set_row(i,k)

GKS=matrix(II,q+1,q+1)
TKS=matrix(II,q+1,q+1)
I = matrix.identity(q+1)

for i in range (q+1):
        GKS[i]=GK[i]
        TKS[i]=TK[i]

M=GKS*TKS
MN= matrix(ZZ,2*q+2,q+1)
for i in range(q+1):
        MN[q+1+i,i]=N
        MN[i]=(M-I)[i]

t=cputime()
M0=MN.hermite_form()
N0=vector(ZZ,q+1)
print 'HNF_time=',cputime()-t
for i in range(q+1):
        N0[i]=M0[i][i]
print N0
```

```
##########################################################################
########################### main ###################################
##########################################################################

print 'begin test'
test(q,1,q^2)
```

q=31 output

begin test

q= 31

N= 142626898851388442193955125501359865760996717321109924032112373711122891

N time= 0.0

G,T time= 0.024

M time= 0.176

HNF time= 0.064

(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,

142626898851388442193955125501359865760996717321109924032112373711122891)

q	factor of (q-1)	bits of N
2	1	0
3	2	0
5	2^2	5
7	$2*3$	5
11	$2*5$	36
13	2^2*3	40
17	2^4	71
19	$2*3^2$	58
23	$2*11$	123
29	2^2*7	172
31	$2*3*5$	162
37	2^2*3^2	195
41	2^3*5	248
43	$2*3*7$	276
47	$2*23$	336
53	2^2*13	386
59	$2*29$	446
61	2^2*3*5	418
67	$2*3*11$	497
71	$2*5*7$	521
73	2^3*3^2	524
79	$2*3*13$	603
83	$2*41$	698
89	2^3*11	746

97	$2^5 * 3$	777
101	$2^2 * 5^2$	841
103	$2 * 3 * 17$	874
107	$2 * 53$	955
109	$2^2 * 3^3$	901
113	$2^4 * 7$	988
127	$2 * 3^2 * 7$	1075
131	$2 * 5 * 13$	1223
137	$2^3 * 17$	1287
139	$2 * 3 * 23$	1284
149	$2^2 * 37$	1433
151	$2 * 3 * 5^2$	1415
157	$2^2 * 3 * 13$	1467
163	$2 * 3^4$	1553
167	$2 * 83$	1656
173	$2^2 * 43$	1731
179	$2 * 89$	1820
181	$2^2 * 3^2 * 5$	1707
191	$2 * 5 * 19$	1929
193	$2^6 * 3$	1935
197	$2^2 * 7^2$	2030
199	$2 * 3^2 * 11$	1973
211	$2 * 3 * 5 * 7$	2092
223	$2 * 3 * 37$	2320
227	$2 * 113$	2425

229	$2^2 * 3 * 19$	2366
233	$2^3 * 29$	2472
239	$2 * 7 * 17$	2564
241	$2^4 * 3 * 5$	2462
251	$2 * 5^3$	2716
257	2^8	2727
263	$2 * 131$	2892
269	$2^2 * 67$	2962
271	$2 * 3^3 * 5$	2802
277	$2^2 * 3 * 23$	2984
281	$2^3 * 5 * 7$	3036
283	$2 * 3 * 47$	3121
293	$2^2 * 73$	3249
307	$2 * 3^2 * 17$	3354
311	$2 * 5 * 31$	3515
313	$2^3 * 3 * 13$	3409
317	$2^2 * 79$	3602
331	$2 * 3 * 5 * 11$	3611
337	$2^4 * 3 * 7$	3758
347	$2 * 173$	4024
349	$2^2 * 3 * 29$	3931
353	$2^5 * 11$	4068
359	$2 * 179$	4184
367	$2 * 3 * 61$	4182
373	$2^2 * 3 * 31$	4311

379	$2*3^3*7$	4261
383	$2*191$	4520
389	2^2*97	4561
397	2^2*3^2*11	4565
401	2^4*5^2	4651
409	2^3*3*17	4765
419	$2*11*19$	4990
421	$2^2*3*5*7$	4782
431	$2*5*43$	5126
433	2^4*3^3	5065
439	$2*3*73$	5256
443	$2*13*17$	5303
449	2^6*7	5333
457	2^3*3*19	5477
461	2^2*5*23	5517
463	$2*3*7*11$	5505
467	$2*233$	5676
479	$2*239$	5875
487	$2*3^5$	5852
491	$2*5*7^2$	5909
499	$2*3*83$	6081
503	$2*251$	6234
509	2^2*127	6291
521	2^3*5*13	6385
523	$2*3^2*29$	6420

541	$2^2 * 3^3 * 5$	6540
547	$2 * 3 * 7 * 13$	6649
557	$2^2 * 139$	7001
563	$2 * 281$	7096
569	$2^3 * 71$	7124
571	$2 * 3 * 5 * 19$	7100
577	$2^6 * 3^2$	7134
587	$2 * 293$	7435
593	$2^4 * 37$	7450
599	$2 * 13 * 23$	7577
601	$2^3 * 3 * 5^2$	7438
607	$2 * 3 * 101$	7702
613	$2^2 * 3^2 * 17$	7653

Table 5.1: Factor of (q-1)'s and bits of N's

Selected q's and determinants

q=311

N= 122204114574813417986481235029388738881698649384864194580617796845196337737183258073824232545608541218918723811960148552537251611400445013268402913197654341769186758968238727747340034553724661488958516434357852246260097149042358543566474704481025131325126599583952374463591105264547959591770978855776019997803605664063703021779873275332132319423303324268978937861857409559076526616995311676002163181851006919781308521219367694909589990562943465678451002497884123921039083816488641142353728376191016348566486234012077314529634625370181844816842906404093063223817023082959392471193484972880937337403803457934669663233675693327958131940763931843945850903774484374401790510896030433962809344951508908261064478006092070458662252217688262713139739477652483265601310413258125913623366106949825527193568742398811805617008805558732894305675319217173800499980105172307209099566550602475304814679644745021541797209529197980589944462562512768354692411110772521996505139930112902009219541998705719839610844963245559661036742756909186733357807104233727586841827964927187772560806972085911578824651602811461602030429777189275621717989030898859967382457754063422299246275821329622025628632603350724802562117287756308843578611640632913144436544488711590026871427353919150398969230289937105084891641612942355696565346756075867621220681810065792555872883091935927336158074926847653547863316973280269791307498781127370536534710586628598217910942589322707683545454030303573721800598831314975779863908526737346100064580115144368501881686966476777

q=317

N= 402178517315408532265557475313709128900642933088657156166009476408
2025096818735665065568606226286154152504064112564861937901952925617407 56
8286926568779646055184746606049316606024011899371771665022800676580767 03
4719619583954968280714396440590336994834530118991027936331801150427362 93
9693518242570076363136840365770932966144304372598975781945110481069726 26
1546285340849472299732286650610727616415294009235102545870596043211915 42
4586960233937332337958986039804601053649038967488623456945225989089033 40
1275419329294322513933100734565652842059670311848945669554893129720949 44
7954836951424304175908471967070682772300874480361890979938803146221788 10
5309105727259177491592506197903103055791894056811937490366344252647565 45
7597564766932511281000136846597587810392066535818039165649526995642433 60
8546500096708317548001282282350497793375323900631878335615233643196379 66
7536096467921713929336707535209851385721085526112592135089963195453144 24
6120147416810255297907994560299604353869770277890910576475371951544234 11
2263700193716596807098474711518620524142746755488794787322941516546432 93
2426171929361587888626006289671937521996598178773391476073536684327075 33
4892817166445708005669892476201578305359678934320690158091873793833445 60
6760277427677070493149308305030488142718346015582953457619985800358337 24
1346580760564307466215915620330738784938802568324196853364619202331253 49
3560712153255237113801078736976763236542229449221244087307270718233645 25
9175297320908538008271399073602614627495850660009294692396963268376106 94
68567179471807528013764566720005407715768914202768155 7

q=383

N= 14580769790786048579713073320512610759358484605246061948980515775207

74952211069325885209727342419187324857988537970859707731901013503830981

14868963679129330247949463670668019053277116714864234569372727999715403

86901101697195533486335856133921155426166323042195462985673651542303182

18601384052109394332710553067093351725149802854193796654545885751461736265

29686729187738867841177486232588221287649939713255931703080954603015560

64262854440090893435962828515919017600496019082806907459752774772149654

17320773174296193526683723202799256147587899707749736430575896042018802230

74846956864081784176847084405459821394611214834449663929832660479274386

31017956435460617872268689048536223008200601915584845567121435040154255

00489435241601908370115633665386847975801896382204023084106792525849140357

81400563983772002596321636229496235302375585140329803401909717949207033

69202249165169714902231036847494140896438546785548350557620139234129697

85107930991057963749985630832287416504887831741698032907058955547702842479

46607708322435162358562537573958487001585487997741045752827323618118857

63245707493651012635463603167780987920047275869960119345415739631674289

15757250488106937836616903820719219556478688451793141490477127006483478774

59402950707784881660863430408161452618287844712253121580372165304071210

09255667745904658762339035510401085623511888982417436321783954442230138

14219140252685440904215209050783670732155447802915178313873512367483600282

40917553360656692700968419301698674310732295954805049065228940814562682

15051621596295121066160620101958319152599732645858807987026779736321720

48986491893465889036114249872116026910548116975945295940076632308081785385

61783491841428912984353151422923319868099014428571890992195173149493389

14405526340356801855365944445252481277250224143631547123162141573825856 71

0131688320750454450402427841414245515834864048271866718354779613147208O5
667647392584241256388983691283678436605160723687265941636927616111371892
653376633932050977

q=389

N= 7195331750747484152085419439650343326167001327774918316175974049277
722938228436284822115935720940855808723841592194554852591261106514552587
998792697344006909328597127958485586198824809760493619528438029382516955
802437372015885868652676060704273108648005446756567866454037217153666707
116107955813259897581789965548166873227089745793038056876086093487364660
104165354375638730780144771940820461379144045407410633806525882471616992
313107313458518056156155372469123009498454107816512700653764419802046775
795056390593183426232868729935687347132474597447270299842803551483280382
464345737592680500610492028940899252744848503850147888866006672309743009
707608056808465671630041936151575721737832608056167755557287448599406240
664128644940787322334450386697752135044664759178397183268772058995455906
530214874403118896365825911606545712252543349530273195645766743047004882
772794343150019268224755541621832841084927152609167782631248686519979077
261704778279434581285089309982530290516450254651948711338697696523561128
237131690866523967123809890516412122433312579639731316482517399044222371
648027429394491857914758789554722411217192580708530710563303630609337512
375736374992413197099835298039698176561667502460196281220809685898103151
801536460824061171709409915407076163699749989106951416033064052305415369
172208716680222928232108669911514529435819037003548630822766056743609717
871400339970257506326522184356565974073087427738029035514597912641413281
864721228361243940548875106883409490452585792883551924553356666885340993
409525115851784559602381695501650437380918742644155483811707202620992640
671919692119264818017617725574319133657575175571011358782147102699466133
976139878846733780028611934562593296650810158727671483594135852975270136
343283327218261498217240273563540924577404997311033419523302409280733431

875376399609828318193709239918912628149158776296863968437043483848149113
291753871808384517843168494031720350845780796001506232612214295949526384
21937632933033111109633577780266405 7

q=467

N= 8193682819735904907936462652762652957783801140282061764680311398642
355238292209136200793745698408979142685271806667880695439656644767365658962804500507328924591277012527354972867083755868359662342631293941095260296836469017267512710855619435662717780121618969799191783589121739510344276659959452330107141588370842890533418683574136420161763154188864032344051335032890602342690990331545512878722845868516478220655672811984728099829584400348024077781024883841078871472427192910334006397072591593425575001582062495270466693923330813787033936932085463161319738777157723719205663510587261989045516277399792258676669025902061468414338936004777221651120128177284169726309676767243230700942651892885739372435031266603405194926736374504661605073635356729678170946671059228219580414102477943739962010943602405830702183270123588613295844379470805381224397382897311239774291035219450733243985454552294613261579819999222768501542040828030342324749500293990959881643288222147454465739982396633029518755264422499169413800948870145113541709861228904602480297159321730407550425161663814137762815328504113649745806201609675807776334603770728185435662442974547545292124484627139481264440306306276174236846371370615288136390337733795900869799059527013550349978911522961593782262111579795770566157261698765468287287036428769421593687671844613576392027427659500766509619007109126319228663188313182821761621096295894335877149917549224508427298360298958806603741672739378573845250923926503996897651348539802615982069497009131645174845900515957045945369600419581505046736732678816987857244113512133573495658735922450650020693185605825145435753481586659509230006830286160027557753020447567816102711727703321437248277987102699335416867799947690827644062178290893074122964705126667835005148297475296259944293090593568576122

562955880820521795084752549150431739105170356981387623138751580074331503049381562934343919188403920170632771201198722829539523581972559907051174426561278179057593819914649483436841887497547366195099487522115176214585885334508559059585758067301292625257032033576801754890512925239321569092559349164983924605586740828964064196155247822764786310935718332219453527564215371429071787341780771928056765790312442784701839664535122716221082064914125012462723860970516555936555767206809055654587364855169664462773211832019761426795537757795544128221807124125073672666982193841820192269309127941260744209247506370470043435144321437917720042890836220355755429062131737559487

q=479

N= 961815180868751842285732208661597056961427746259514552984597834345
4448618802693093806174301703041941859348535932869907153101083071411010 80
3456063630665900536331741409195151920885582430505379103614171689325778 48
9635851915467509624636121252747504888668412598846424286996049592018557 32
3913094689734727755921975094524127950642823880986148497506016970894935 90
3648565943867612479405588845250553752006483345380077674303557419432127 55
8208733421463971379344533049440806231439526701385635497006169364431805 78
3429180839792108139524530423628000442643282649451591297372208850213427 13
3461910687264293985531527946810733266739248693995449148735869497153921 10
5314389027122712751770246771398979638341844855238289658771572324015835 90
7245386690853156319489160004643572069455529184394600986105483302393781 14
1226048904218763968499253154495883940814778824960447766628815179984355 62
9221124676140341161660215389374376582213084770495696212722696372478379 34
6737464458621021349757102200677279480126467334425581581410412071179639 57
3301446871165160788413543928494829556943838364219869607425806655123612 45
8396781274885617707782277187153820621053025661615254016028798550167633 48
2289457284153091193172949445945819335159230225582435741301623331735746 65
1145967882164096617782018067111826095805262567896399690590696998147023 35
4649850475545490721757949210819004280539409488835155856513839335094074 64
5353167906795507284117756484968777385946448482170998180715473880896091 12
0114317136474201035595812200113297236470728953746360372060131798038630 07
4290449713416084132731428579735177431074571112277441815147037314899375 26
8506861013964291111722980506751006009106776900880219506015594339980232 81
8353321346278332751075426021729474680954306921091064570930765334723508 85
5070222800973472311420853428474908794653665552840099195956533374846893 07

3498402782591612074026572470534922077554209579350139370508964834533793808280967603229737128366508014325346385982891506143336622474822557598662355592724888000425807158888270586572211350974539619147958502595028877855443396676609019307188139368164436521234088698641598220269495261869699576011611641815688527725019221309831393749981545688456314906829255587458174654862955065758156707104898890483430251829060531678370531837509181530733074470861100503315958564220106551071648730545454135922433495504771358457492930279799705805251261156824547063742079194534530227269285571448290261176624433157617922891811797968020227354210533461833778707624586319989156303176891868381496721595692273740848513693233427861493435784587573288782820204786167152167955380970678 91

q=487

N= 10782720766975026508950644062220650443872163525537818263616134273 7

546361849057898469875599327816975774441578412233114599521742614494846887

740703080811148454047060485212052898351429291302607383413701329004937472

110130810594384628498903013384299898677662097586203992584420300123479688

785684393540422293410360973720913164496834267238412112247567596545168837

737144534589039088347178412795877575787183707026931850844179567530895640

048616446605895392791202991591300212536936775209605943209829037959038448

644375408328710718300289456458572242049271728769524363126273853715500193

992389397888943821783652200285128430559853028782821408152591885930253959

343363572383873083165126137898284523398720102296715451247068157020833701

559621319003673098042630112932862628501950564632735466206216603138321249

217171339976696682869969285858086091226851424749306290948023985863737595

479286567482749091395852685584018895956480265983348479431205342847195247

664202184545509444073196750969881471993537723451511123356835172969794677

005895270609159438063149957132594233024220238098666324875715524694568431

877984803514244863991450442722463359917319219876714736136981326149224271

917119983677231104646058922538799501312301306624431782348213725731474980

072268037174861637258178428492282775604518408282182883338962744713428765

351357038539796626115406098868926182873473137617788072302142999686731779

004538770966351787580252505338707237518234022336687560434124912382388971

097424646432301275534386052715373826920904552232580275869239823036727431

833531669999335834386165607257759778153553173938764606955523815952618365

383445341291463783562970897109625720657070919217228163652664563258546056

946645712882883254672309664786508152237835087204624488511667622924336500

614389360778481908856057067993634846367663690999501201392634269969314892

4378932395557253634087976106154814264444461900711295393120661914650623502832379340691253835124226483693428320243605068261288082505623702176148298333266322720608621778068987374449904958132661315380947330602852818657061469919021537196527652277068392161545708863351415110089462235727710111500949006978638726979081832600240994001969907020897723174013212365576625760481287046744192124566549615922774378805380656247559903099690092736600250680226860279698003144880843184623701517154967251909911487616335053831460587215690174974677652703489026202458923686267176112697154042850818617080803258117793872354821201303345158215561917205647447472447952814789172239582686120291642904317414205949691098521792189502811697498929450533471990402912812431381462l9

q=503

N= 147336834349706797044425656887744489349639385206508274656353112840
154188839157203929396413979603053484732170317150123836585299940217124069
418131155639820786383863808503837816134221204162370689398910672128830126
875535715790977173830346306391882707890705220333528612457580199764835975
303359021329156579459835055274985048399711308604360025915407798395681986
697460141421914534555594657656778406283679170195247096369765818165808327
612004043969219018782573596013565703189362803334814399574866472961024938
918885877805928920357033952882962667826061905099037118891324342946832542
542214752563044039371871097358602810888164251672740375316485427806529032
757599061664447724062151740934514802419176331498400804933058201107558702
429449303196408769093873187335791374587290764095580015486584401899629798
461954021625226347464078642531684664097855482709859829407763649948491337
884199710887347486437230796060472145666441721343041702272464424537452080
613248444661339289699499009332546379642297491058545027809442713117635144
082450453755036212814099124056641147665035766416557635641079692604973950
773920155950817728259665768496055451212444576214827193838926459482595718
972001461250141781841674253550053321923602906140009626099817970732673000
338829523739369243550841931370360592140997666100024640249461334097681741
208087212249089069367114071149458248157736163948506921189310510108140075
496373678977565853487063490343044675664978839002031659168493637210851138
425593781704876455710917916478494833374697463417597147391641221487862132
925680843603228165826146305863639709764245209221312894884648013450113706
256344098505539432089787128428887942976949173572879311265729934750968571
650606623246818250525624992923075964758291050700754231687302223856217821
543464371668958982566875788733981410144345204688702480286202103830814032

9590720707875739665416805562175412650243701800985143476969446402434001009974668900973052920981609196989369423018013418873498734526488465058527490932166194695334778574654762189425032641935018714994473998178327493809319913295814215589447716099380844306663061182030152956116355747239295912164727014721815762591310750838887925964329947900598106170181069097705739270543401771878151454649439174524410744405039162442184299814252752336124179950678897788639961005561319144335578466881923408040301169591562751251154804833692017286166408930083262604976566653282858156843775809252626582809936567152542604305545516231933646962241284113331633598956211209074430629525432625422487200262748189501789777158028552584279637263837611155769708096978474438991095479731343370182650110683123478736509300503895743359985184291703535457734572127408094622693223252048773702960489505145522039681428001882101017294768260292344835726600001

q=557

N= 314368359478155049397352092307869311778858723471788302913880806244666

652591775108235853834502874501624378601743448311570059757504028872827564

224061117949779642122256530153511874573785116643054301329404190330004853

980652341337937801879803091785690699711289861116864017307913133498507701

009528005456708412140367642183463205775452412251222188785968466102897363

328161980421329221327493873640039614073637232011066188807539543010166340

191445518111807007589518878489123477047343215769164314635530521990723232

814172643445247196280108340123505929636980082583878211875201178095926826

155653424350068425548870116149681450780222300890057094917940478625384600

504947544383504214710940333054122426703810208644984586405282256025901624

878896366918945606134849236766005021982356538755971074911440796801148491

171923334373974519490404860924481495434071312252189692917205833372789159

747628420114376012791164203937201914268064299245656874120072924534856648

458821871049333707703918724666059677942957503024430362974072144858166822

946661055915224440616236103637923315469012627641543519131399105681529076

143499712062446523436021618834840780879763419902016039611638172461233083

984225298628313414804720870707226321993502241670406079550703617001059809

861480023408341372770978896929819858736458197921710407847699341727588094

285385342733010116428226429638095606628398419087812417610221325060316161

159693363779446034883263778915734408121533618878419794217061219220254895

053070998712309817336438271848856606896337462238028355081891487779357855

076862950394244243888216155432350891820796041079610114743953860223017542

890222894206686692110393865956219485207057415283729576726228568693407835

309260158094190429367209226896834991732360891978767584506736862264040912

776346723372524403845778789099916824268533775799619624759398530870006058

4208179411883347769320549138080555021443135377998557798892660487157944688
884977147999578873473393616726182174081606185529856842321897210873112779
888466466278757879644288010669000985360411039692167727361997567427467201
148700848199330595785419572519503052525423627189213559908179808198430529
830962485788541703952406384496380180654819260477234312213501220447522384
529948466416205690131295720001116285887515737531007721088902498245843973
421891085058448440784625711190535561802202593564877697154649350623097905
532224681404066588537916024926239838587786385816025929279678857514304525
882612134670026079124654944813606521576121279776959051908976489628154035
498413285257800076875565865494381533459542084000698358157895228840199972
343094474812815029715713996232012861854066665385552565871909944200726505
973315982461118242816383707549822671416232255328602144916260163660248769
252841301900166902587391158832987087454412163053336834082769959075947198
691986442430155402063952090753815105872410480598238840487195388384426796
687923031076630171569631566283017340015016680340831914952761329207351169
281085588477150892767776013021396121764386929332762739675653306049918288
157962135589256681125696589296050957540003302416458845538757926020674004
861742776007

q=587

N= 2307128339166220356384869939572283870824394409144539542643795995890
193987178273115520241818054503417807301768222837967717792820981566830644
702359162800544300509265014734969625319556929612542769704066604717079466
6062544513837515303929517857632068794889342597622092659082052482366093 80
4952871776713360680499453434701662678703065477630908504506571157201332 77
4003301272162246612443579324214202683861935401795114509027045735155014 58
3817183717656673294951648451145170925309887812125289777958597237171887 52
3691039821603612552446805429411197527044726521448861096161666409113457 55
8772341539240687497090525226548361676078722763943539138842745164838222 50
8161955216140298878200816834113895826503528167906286098948217591949420 31
6180481766250335025450951844503542199599298741590889810178381499042950 37
8110360691803073748418624280060722780596816290160115117027369891382157 40
2369041890132947897631190193899974081262798306756298162484893112463081 56
8414563369163297192575389897069271613945012817237762853492038292957317 11
0052409883835034370439556632382664667738475683275991613195789236213035 82
4771879916824437315648665996264520557669152870998243590727795262760248 91
6887265423695511430734137170642631082427932166505851870788783189069672 04
9999786874039175394853473163853659347811084188469075688345112490019989 40
0325423548883609946104829570425381848713735659482465365012246468046744 99
0788918403774629418351630823934768986411779817749922200629821568974864 49
5273414083792941819127788658152706963101678089068546563277037355164685 68
4747027706356745573514296446694941305089466415491924600642176771366246 21
7136392551588979696991398177860444562225884533275162725957579966759194 42
0378489896787300751260615884362619181558771760475113043569485922788040 74
5683026560982632994462887080686683642305635007610302365492943223848632 14

517853204802788102862393626861718398381444091287427093733945278126404808796535973263119676398693284652821532664126160067352850484568003929998073516589928740417017156894461954894742021202796395663649737718769934996393011183684458362924201774667644541185171458687248709915964687235161539871267721373025668380519500345068690263201498564958347090356847808831537149474439507842599554178082870354464254085680369728831977964073127999220670270805081225668622838876565347713145838614988930954677767875707247470616606301530730299409426435834932437280589830089261923095809541554391430373491679119034287808812544898489197235741161261032303532511318530135554121398322602846871754957953766783121789863576016931215881208559731249867551409422414906690305733827311664391503575858345649906510250379160760862532712697014396404622128029250608057775171645321575304855323798022855936790469473625543324144479813493644015222680915489310217660209655279791513674624942831006689176141001330394601459923605020782149235052744558624710143970654049335051443926984242211495568883346306266480953407162443212331733260428910724324374253226952750538404615823972540066741727665744545152170149719073976507902640748242860693813050289265482156137954791737915062679940763903746780188380252172047689123943462356314807421611053197591302585151387544773186589927331086413744144424445703759714719109697828577019913239125701674792316881525364718657528643649544992742182599

q=613

N= 37967508226528476918073262558472181670648196920028343275312848652022
7310061373703070206769874420021839944188081432666694304807266004396888 08
7346129196518753546853360866753174517788362325417033707817976284212339 86
6211138821147502875504135893310257415564245942472726786254249333694799 52
9060355755369532222120240610678535959958970270865068012936942478315854 72
4732837939825044481209635395290455769169403113282073970101189069077331 34
4381560024630859886885781986242030435426404411724459403516420484790647 94
9016001356875374403068298640487806868880522160067280498348142252304658 43
3975536121778067767853125741114049545303372270758377663110473045485397 21
6142604205721191437379591350008902175158318159116649650532193237816729 89
9135617140593787482375967379402044852111905723282833871676861457062913 48
4430090691185959850373541279602208223228415374706420866175773941506892 32
6885475868143229015060395425269205090343210189216836087050254900037146 37
5949350045804375036588856776496387226349050066545149969193279421535009 29
3785738971019059668441255082657452741301234400630132133893662240187946 89
4442576270647858411858908630614582721610174926321996159837646409254923 32
7625822155060747020288400541053028400009650505736910054032781028404428 87
1618064890272376616628445236406024185335509653665135675532131895651307 68
7885071977016305358922355286437246842694801288231476160819120901057161 30
8884132731246188150793435384995219075353396529921406681567048557783416 15
9275792977252668655625058954574685060860323359291778236931429214234909 75
5695071928816137621680791011925148103300280604438426737717586081064697 39
2935859753163930052423589749008266463191904159193121619886869575789151 67
3542650956495196997256767843662855280039911282967949408635082785555182 85
7688769033097864928616450613534737573937397587530026254071563747581341 53

2998897247335710444370503299675384437920052234654005288590829239656643898

5350052674559216459096612068100305821814857097999465332120897993308966667

9434233158649565538933250759726167953829147677321705942685472203327216822

4479981191999027772981683466744373988756963097652862133167805164786664144

0263349366706053382249148283230272513948309024845461384141472221752103399

5470105886361735051439234403342449009595617986965445059252906893795739922

5150639379745126754725490368359850738037263213003473982271235566539490811

8418117070026795102806958117821621955179320420843936330066814410828362022

7186675566861168761455245503368736567246420532376561946224181505210394299

5655743041612512124863106335696486224939522507807958791530316702759504766

1144702080814455184685964518796175120951569949703130798908307229466305699

4276878547718436260925231658374338359946332900564218404388293847313603644

7137908362061690719016548469117424816818025352593148713393445110759851222

6103588823068577658455393707807159521318083902515629330110591772807495311

9214314363386055458666580762349248608492688603689876706878297633904508766

3454855851890336267022009045815205257100145804331659001973784550830153400

4129516958364250440179955239271025778647089539186105760236989181844573777

6642401901587625138158512605004726731502462201100888296742011370237716511

1408981678870370873065663825251028634974854584978821122442909456282391955

1721160794848110273696004569024088302630595556513751684958401979183001455

1079003165122253055867951710959900528175357990749843091473424740609760044

660467