The Lattice Sieve Field Selection of Cado-NFS

The Lattice Sieve Field Selection of Cado-NFS

Ping Xue[1,2,*], Shaozhen Chen[1,2] and Zhixin Fu[1,2]

[1.] Institute of Cyberspace Security, The PLA Information Engineering University, Hennan, 450001，China

[2.] State Key Laboratory of Mathematical Engineering and Advanced Computing, Hennan, 450001，China

[*]Corresponding author's e-mail: xueping941110@163.com

Abstract. The software application of the discrete logarithms on the "large" finite field is studied. The most effective algorithm for the problem is the general number field sieve (GNFS). Focusing on the theory of GNFS of solving software Cado-NFS and lattice sieves, we find the selection of lattice sieve which improve efficiency of CADO-NFS. This is the first parameter modification suggest. The efficiency of our selection is better than the current by experiment.

1. Introduction

The Discrete logarithms problem over $\mathbb{Z}_N$ Means: let N be a odd prime and $g \in \mathbb{Z}_N^*$, for $y \in \mathbb{Z}_N^*$, finding the solution x which satisfies the congruence equation $y \equiv g^x (\mathrm{mod}\ N)$. For prudently selected large prime number N, the discrete logarithm problem over $\mathbb{Z}_N$ is often computationally difficult to solve.

The famous Diffie-Hellman key exchange protocol [1] (DH key exchange protocol for short), elgamal encryption system, elgamal digital signature system, DSA encryption system and DSA signature system, etc, have been effectively applied to the Internet, include electronic mail, the family banking business and the safety of the web browser, financial services, such as electronic cash, credit card transactions, Immediate withdrawal services and wireless communication. The security of these encryption systems and protocols is based on the difficulty of solving discrete logarithm over $\mathbb{Z}_N$.

The algorithm for solving discrete logarithm problem can be divided into exponential algorithm and sub-exponential algorithm according to the time complexity. Exponential algorithms mainly include baby-step giant-step algorithm, Polloard rho algorithm [1] and Polloard lambda, etc. Sub-exponential algorithm originated from Gaussian integer method (COS) presented by Coppersmith [2] in 1986, The complexity of solving the discrete logarithm problem over the finite domain $\mathbb{Z}/N\mathbb{Z}$ is reduced to $L_N(1/2, 1)$, where

$$L_N(\alpha, c) = \mathrm{Exp}\left(\left(c + o(1)\right)(\mathrm{Log}\, N)^{\alpha}(\mathrm{Log}\,\mathrm{Log}\, N)^{1-\alpha}\right).$$

In 1993, Gordon presented the general number field sieve method (GNFS) for the solution of discrete logarithms problem over prime fields. Further reducing the solution complexity from $L_N(1/2, 1)$ to $L_N(1/3, 3^{2/3})$, which is a breakthrough in solving the discrete logarithm problem over number fields; Subsequently, Schirokauer improved Gordon's GNFS, Further reduce the complexity to $L_N(1/3, (64/9)^{1/3}) \approx L_N(1/3, 1.923)$.When the modulus has some special properties, the problem can also be solved by the special number field sieve method (SNFS). Whose complexity is $L_N(1/3, (32/9)^{1/3}) \approx L_N(1/3, 1.526)$.in 2016, Joshua [4] gives an example of the discrete logarithm problem solution using SNFS on a thousand-bit prime modulus. Table 1 summarizes the time complexity and memory complexity of the above various algorithms.

GNFS is the most known and well-known algorithm for factoring "large" integers and solving discrete logarithms problem over a "large" finite field . The number field sieve method (NFS) is divided into two phases: offline phase and online phase. The main task of offline phase is to solve the discrete logarithm of a given factor base; the main task of online phase is to represent the required discrete logarithm as a linear combination with the discrete logarithm of the factor base, and then obtain the corresponding discrete logarithm.

Cado-NFS is a tool written by C / C ++ for implementing NFS, can be used to factor integers and calculate discrete logarithms over finite fields. It contains open source programs and common scripts that can be run in parallel over a computer network. There are four groups of parameter files in the program: p30, p60, p100 and p155, corresponding to 30-bit, 60-bit, 100-bit and 155-bit modules respectively. For actual discrete logarithm problems, Cado-NFS will match the actual module N to the corresponding modulus field automatically, then call the corresponding parameter files for compute. When the given large prime number is larger than 100-bit, the time to solve the discrete logarithm with the original program parameters is usually much longer than the time estimated according to the time complexity. Therefore, for public key cryptoanalysis. It is of profound significance to find the corresponding parameter allocation of Cado-NFS when solving the real discrete logarithm problem.

Table 1. Algorithms' complexity

Algorithms	Time Complexity	Memory Complexity
Baby-step Giant-step	$O(\sqrt{N})$	$O(\sqrt{N})$
Polloard rho	$O(\sqrt{N})$	$O(1)$
Pohlig-hellman	$O(\sum_i e_i(\mathrm{Log}\, N + \sqrt{p_i}))$[1]	$O(1)$
Cos	$L_N(1/2, 1)$	−
GNFS	$L_N(1/3\,;\,(64/9)^{1/3})$	−
SNFS	$L_N(1/3\,;\,(32/9)^{1/3})$	−

Cado-NFS is a C/C++ implementation of the Number Field Sieve (NFS) tool that can be used to decompose integers and compute discrete logarithms over a finite field.

At present, Cado-NFS program contains six algorithms: kleinjung polynomial selection [4,8-13], screening relationship pairs by lattice method [14], cavallar relationship pair filtering [13,15,16] , schirokauer mapping for construct sparse linear equations [10,17,18], wiedemann algorithm for solving large sparse linear equations [19-21]and single discrete logarithm solving algorithm [18], which correspond to the six most efficient theoretical algorithms of the GNFS.

On the one hand, there is no literature to match the corresponding parameters of the number field sieve method with the existing software; on the other hand, there is no achievement to set the boundary for the relevant parameters of the practical problems of the number field sieve method. Based on the existing work, this paper further explores the parameter setting of Cado-NFS about discrete logarithm solution.

In terms of theory: summarize the six algorithms of GNFS. Find a better selection of lattice sieve field of CADO-NFS. In the combination of theory and practice: Compare the solution time of this papers chosen parameters with solution time of current parameters. The outline of this paper is as follows. The basic concept in algebraic theory is given in Section 2, in Section 3 we introduce the theory number field sieve method and the Six theories implanted in Cado-NFS, including Kleinjung Polynomial selection, Screening relationship pairs by lattice method, Cavallar relationship pair filtering, Schirokauer mapping for Construct sparse linear equations, Wiedemann algorithm for solving large sparse linear equations and single discrete logarithm solving algorithm. In Section 4 we introduce the selection of the lattice sieve field, experiment and analysis is given in Section 5. In Section 6 we conclude this paper.

2. Algebraic number theory

The basic definitions of algebraic number field, algebraic domainand smooth can be find in [6].

[1] Let $N = \prod_i p_i^{e_i}$

Definition 1 (Norm): $K = \mathbb{Q}(\alpha)$ *is a* field, the minimum polynomial of α is$f(x) = a_n x^n + \cdots + a_1 x + a_0, a, b \in \mathbb{Q}$, then norm of $a - b\alpha$ *is* $N(a - b\alpha) = b^n f(a/b)$, which means the product of all the roots in f.

Definition 2 (Norm of Ideal): let A and B *is* ideal of O_K, $A = \mathfrak{p}_1^{e_1}\mathfrak{p}_2^{e_2} \ldots \mathfrak{p}_r^{e_r}$, where $\mathfrak{p}_1, \mathfrak{p}_2, \ldots, \mathfrak{p}_r$ Are different prime ideal in O_K, $e_i \geq 1$, then

- $N(A) = N(\mathfrak{p}_1)^{e_1} N(\mathfrak{p}_2)^{e_2} \ldots N(\mathfrak{p}_r)^{e_r}$
- $N(AB) = N(A)N(B)$
- If $A = (\alpha)$ $(\alpha \in O_K)$ Is main ideal, then $N(A) = |N_{K/\mathbb{Q}}(\alpha)|$.

3. GNFS of Cado-NFS

The GNFS is the most effective algorithm known for solving discrete logarithms on "large" finite field. GNFS solves discrete logarithms into off-line and on-line phase. The task of off-line phase is to solve the discrete logarithm of given factor base, and the task of online phase is to represent discrete logarithm as a linear combination of factor base. The off-line phase can be calculated only once. The principle of the GNFS can be described by figure 1:

Step 1: Let $f(x), g(x)$ be two irreducible polynomials over $\mathbb{Z}$, and have a common root m over $\mathbb{Z}/N\mathbb{Z}$. Let α and β are a root of $f(x)$ and $g(x)$ over complex field $\mathbb{C}$ respectively. $\mathbb{Q}(\alpha)$ and $\mathbb{Q}(\beta)$ is the corresponding number field, O_α and O_β are algebraic domain of $\mathbb{Q}(\alpha)$ and $\mathbb{Q}(\beta)$ respectively.

Step2: For a given sieve area $\mathcal{A} = [-A, A] \times [1, B]$ and given smooth bounds B_α and B_β, the smooth relation pairs $(a, b) \in \mathcal{A}$ were selected by the lattice sieve technique , so that the following three conditions hold simultaneously:

- $\gcd(a, b) = 1$;
- Norm $N(a - b\alpha)$ is B_α-smooth;
- Norm $N(a - b\beta)$ is B_β-smooth.

Step 3: Choosing an appropriate factor basis (the size of factor basis is usually related to modulus N and computing resources. The larger the size of factor basis, the higher the computational complexity in the discrete stage. If the factor basis scale is too small, it is very likely that a single discrete logarithm cannot be solved). Then, based on the algebraic number theory, the factorization of $a - b\alpha$ and $a - b\beta$ on the factor basis is given.

Step 4: Using Schirokauer homomorphic mapping

$$\varphi_\alpha: \begin{cases} O_\alpha \to \mathbb{Z}/N\mathbb{Z} \\ \alpha \mapsto m \end{cases} \qquad \varphi_\beta: \begin{cases} O_\beta \to \mathbb{Z}/N\mathbb{Z} \\ \beta \mapsto m \end{cases}$$

build the equation $\varphi_\alpha(a - b\alpha) = (a - bm) = \varphi_\beta(a - b\beta) \bmod N$. Pick up enough smooth relations (a, b), using the above equations about factor matrix of the discrete logarithm sparse linear equations system.

Step 5: Solving the sparse linear equations to obtain the discrete logarithm of the corresponding factor basis.

Step 6: Expressing the desired discrete logarithm to a linear combination of the discrete logarithms about the factor basis, from which the corresponding discrete logarithm is obtained.

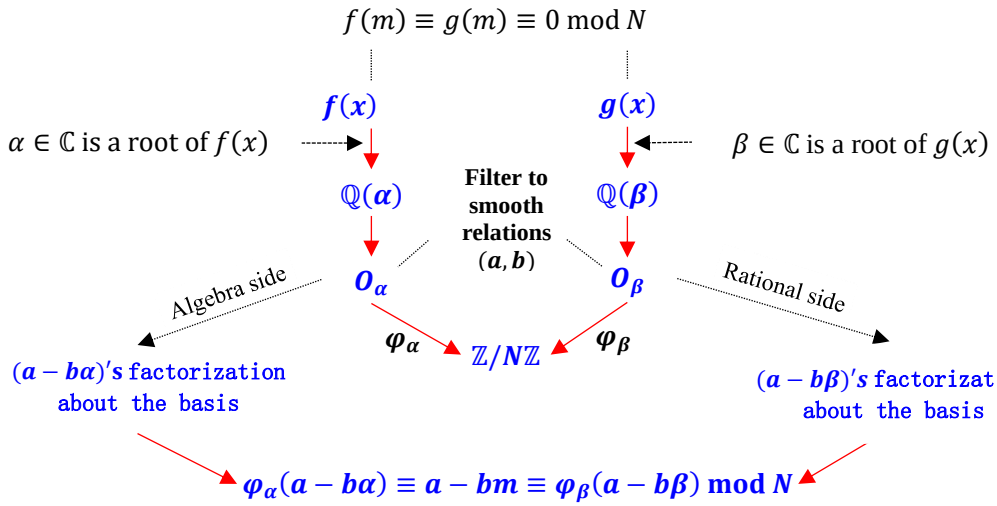

Figure 1. Principle of NFS

Cado-NFS follows the above process for solving the discrete logarithm, which is mainly divided into six steps:

- Kleinjung polynomial selection [4,8-13];
- sieving relationship pairs with lattice method [14];
- Cavallar relational pair filtering [13,15,16];
- Schirokauer mapping constructs sparse linear equations [10,17,18];
- Wiedemann algorithm solves large sparse linear equations [19-21];
- Single discrete logarithm computation [18].

4. The lattice sieve filed selection

The lattice sieve expend the most time of GNFS for solving problems, so we find the lattice sieve field to improve efficiency.

4.1. Kleinjung polynomial selection

The first step in number field sieve method (NFS) is to select two irreducible polynomials over Z, so that they have a common root m over $\mathbb{Z}/N\mathbb{Z}$.

We choose the degree of $f(x)$ and $g(x)$, and $\deg(f) = d \geq 1, \deg(g) = 1$. The degree d of optimal polynomial $f(x)$ is given by Literature [4]:

$$d = \left(3^{1/3} + o(1)\right)\left(\frac{\log N}{\log\log N}\right)^{1/3}.$$

In 2006, Kleinjung[4] presented an improved algorithm for generating candidate polynomials in the first stage ,the main idea is to look for polynomial pairs with the form $f = \sum_{i=0}^{d} a_i x^i$, $g = lx - h$, s.t. f and g have a common root m under module N, and the first three coefficients of f are controlled in a small range.

4.2. Sieving relationship pairs by lattice method

Suppose $f(x)$ and $g(x)$ are relationship pairs selected by Kleinjung algorithm, α and β are roots of $f(x)$ and $g(x)$ over the complex domain $\mathbb{C}$ respectively, $\mathbb{Q}(\alpha)$ and $\mathbb{Q}(\beta)$ are the corresponding number fields, O_α and O_β are algebraic domains of $\mathbb{Q}(\alpha)$ and $\mathbb{Q}(\beta)$ respectively. Given the prime smooth bounds of f- side and g- side: B_0 and B_1, the factor bases can established as follows:

$\mathcal{S}_\alpha = \{\langle p, \alpha - r\rangle \mid f(r) \equiv 0 \bmod p, p \leq B_0, p \text{ is prime}\}$,
$\mathcal{S}_\beta = \{\langle p, \beta - r\rangle \mid g(r) \equiv 0 \bmod p, p \leq B_1, p \text{ is prime}\}$.

Let sieve field as $\mathcal{A} = [-A, A] \times [1, B]$, A, B are given positive integers. the task of sieving is to sift through enough pairs of relationships $(a, b) \in \mathcal{A}$, so that:

•$\gcd(a,b)=1$;
•$N(a-b\alpha)$ is $B_0-smooth$;
•$N(a-b\beta)$ is $B_1-smooth$.
We call relation pairs satisfying the above three conditions smooth relation pairs

Sieving method is the core of the number field sieve method and also the most time-consuming part of the number field sieve method. The computational complexity of sieve method is closely related to the size of sieve interval, factor basis and the selection of polynomial. The sieve method used in Cado-NFS is the lattice sieve method, which was proposed by J.M. Pollard in 1993. In order to understand the basic principles of the lattice sieve method, it is necessary to introduce the classical sieve method before.

4.2.1. Lattice sieve

Although the classical sieve method can effectively screen out smooth relation pairs, it still has some shortcomings. We take the f-side as an example to illustrate. The classical sieve method needs to judge whether the norm $N(a-b\alpha)$ is smooth or not. When the norm value is large, it is time-consuming to judge the smoothness. Therefore, in 1993, J.M. Pollard put forward the lattice sieve algorithm 1, whose core idea was to introduce the large prime number q in the smooth boundary: on the one hand, divide the sieve field into small sub-grids.

Each relation pairs (a,b) in small sub-grids satisfys $N(a-b\alpha)\equiv 0 \bmod q$, so that the smoothness judgment problem of norm $N(a-b\alpha)$ converts to smoothness judgment problem of a smaller integer $N(a-b\alpha)/q$, and improves the efficiency of smoothness judgment; on the other hand, In view of the relatively sparse distribution of sieve points in the sub-grid, to improve the density of mesh points in the mesh area and the efficiency of the mesh method, the planar transformation of mesh points was carried out by using the lattice reduction technique. Although lattice sieving will lose the relations whose norm are not divisible by q, the effect on the improvement of efficiency is negligible.

The lattice sieve method is further subdivided into row sieve and vector sieve. Row sieve is the classical sieve method after transferring flat (a,b) to flat (c,d), vector sieve is the classical sieve method that first transfers flat (a,b) to flat(c,d) and then to flat (e,f). The row sieve is suitable for the sub-lattice with a small prime p, while the vector sieve is suitable for the sub-lattice with a large prime p due to its large computational cost because the lattice reduction operation involved in the transfer process of sieve points.

Let's take the f- side as an example to introduce the basic principle of the lattice sieve. We always assume that $\gcd(a,b)=1$.

As what mentioned before, B_α is smooth boundary of f- side, q is a big prime number less than but close to B_α , let $\mathcal{S}_q=\{\langle q,\alpha-s\rangle \mid f(s)\equiv 0 \bmod q\}$. We can know $\mathcal{S}_q$ is a subset of f- side factor base $\mathcal{S}_\alpha$, so $\mathcal{S}_q\subseteq\mathcal{S}_\alpha$ .For any ideal $\mathfrak{q}=\langle q,\alpha-s\rangle\in\mathcal{S}_q$, we definition that $L(\mathfrak{q})=\{(a,b)\in\mathbb{Z}^2 \mid a-bs\equiv 0 \bmod q\}$.For any $(a,b)\in L(\mathfrak{q})$, has $\mathfrak{q}\mid\langle a-b\alpha\rangle$, therefore, $q=N(\mathfrak{q})\mid N(\langle a-b\alpha\rangle)=N(a-b\alpha)$.

4.2.2. Divide lattice $L(\mathfrak{q})$ into sub-lattice $L(\mathfrak{pq})$

Similarly, for any ideal $\mathfrak{p}=\langle p,\alpha-r\rangle\in\mathcal{S}_\alpha, p<q$, we definition that $L(\mathfrak{p})=\{(a,b)\in\mathbb{Z}^2 \mid a-br\equiv 0 \bmod p\}$.

For any $(a,b)\in L(\mathfrak{p})$, has $\mathfrak{p}\mid\langle a-b\alpha\rangle$, therefore, $p=N(\mathfrak{p})\mid N(\langle a-b\alpha\rangle)=N(a-b\alpha)$.Define sublattice $L(\mathfrak{pq})=L(\mathfrak{p})\cap L(\mathfrak{q})$, It can be known from formula (1) and (2), for any $(a,b)\in L(\mathfrak{pq})$, has $pq\mid N(a-b\alpha)$.

In fact, the intersection of $L(\mathfrak{pq})$ and sieve field $\mathcal{A}=[-A,A]\times[1,B]$ is the point we need to sieve. This moment, the distribution of sieve points in sieve area $\mathcal{A}$ is very sparse. In order to improve the efficiency of sieve method, Pollard proposed to use lattice reduction technology to carry out flat conversion of sieve points.

4.2.3. From flat (a,b) to flat (c,d)

Find basis $V_1=(a_1,b_1)$,$V_2=(a_2,b_2)$in $L(\mathfrak{q})$. Lattice point (a,b) in $L(\mathfrak{q})$ can be expressed as $(a,b)=c\cdot V_1+d\cdot V_2=(ca_1+da_2,cb_1+db_2)$ .According to the definition of $L(\mathfrak{pq})$, $\forall(a,b)\in L(\mathfrak{pq})$,

$c(a_1 - rb_1) + d(a_2 - rb_2) \equiv 0 \bmod p$. Let $u_1 = a_1 - rb_1$, $u_2 = a_2 - rb_2$, then the above equation can be simplified as $cu_1 + du_2 \equiv 0 \bmod p$ .Therefore, we convert the lattice point (a, b) on the "(a, b) flat" into the lattice point (c, d) on the "(c, d) flat". Since the lattice point on the "(c, d) flat" is often denser than that on the "(a, b) flat", the flat transformation will help improve the efficiency of the sieve method.

The lattice point (a, b) on the flat (a, b) corresponds to the lattice point (c, d) on the flat (c, d). hence, on the "flat(c, d)" for lattice point (c, d), the substitution (3) can get "(a, b) flat" on the sieve (a, b). Therefore, we will only discuss how to obtain the sieve point (c, d) on the "(c, d) flat".

- If $\gcd(u_1, p) = p, \gcd(u_1, p) = 1$, then $(c, d) = (c, kp), \forall c$;
- If $\gcd(u_1, p) = 1, \gcd(u_1, p) = p$, then $(c, d) = (kp, d), \forall d$;
- If $\gcd(u_1, p) = \gcd(u_1, p) = 1$, then $(c, d) = (kp + c_0, d)$, where $c_0 = -du_1^{-1}u_2 \bmod p$;
- If $\gcd(u_1, p) = \gcd(u_1, p) = p$, means $a_1 - rb_1 \equiv a_2 - rb_2 \equiv 0 \bmod p$, then from the definition, $V_1 = (a_1 , b_1) \in L(\mathfrak{p}), V_2 = (a_2 , b_2) \in L(\mathfrak{p})$.

Notice that V_1, V_2 is A reduced basis of $L(\mathfrak{q})$. Thus to know $L(\mathfrak{p}) = L(\mathfrak{q})$ and $p = q$, it's contradiction. Therefore, this will not happen.

4.2.4. Lattice sieve

Since the principle of f-side and g-side sieving method is completely similar, we will only take f-side as an example to introduce it.

As mentioned above, the lattice sieve method is further divided into row sieve and vector sieve, among which row sieve is suitable for sub- lattice with small prime p, while vector sieve is suitable for sub- lattice with large prime p. Before sieving, selecting an auxiliary prime number bound $B_0 < B_\alpha$, dividing factor basis $\mathcal{S}_\alpha$ into two non-intersecting subsets $\mathcal{S}'_\alpha$ and M_α, where $\mathcal{S}'_\alpha = \{\langle p, \alpha - r\rangle \in \mathcal{S}_\alpha \mid p < B_0\}$, $M_\alpha = \mathcal{S}_\alpha - \mathcal{S}'_\alpha$. Row sieve is used for the prime ideal of $\mathcal{S}'_\alpha$, and vector sieve is used for the prime ideal of M_α.

Specifically, if $\mathfrak{q} = \langle q, \alpha - s\rangle \in \mathcal{S}_q$ is a given prime ideal, empty array $Arrayf_{(c,d)}$, for each prime ideal $\langle p, \alpha - r\rangle$ in $\mathcal{S}'_\alpha$, calculate the sieve point (c, d) according to (III), and add $\log p$ to the vaule of $Arrayf_{(c,d)}$. For each prime ideal $\langle p, \alpha - r\rangle$ of M_α, on account of that prime number p is large, lattice points are still relatively sparse, so at first we calculate a reduced basis $v_1 = (c_1, d_1), v_2 = (c_2, d_2)$ of sub-lattice $L(\mathfrak{pq})$, then lattice point of "(c, d) flat" can be expressed to lattice point of "(e, f) flat". Acquire lattice point (e, f) by classical sieve method, calculate the sieve point (c, d) corresponding to sieve point (e, f), and add $\log p$ to the vaule of $Arrayf_{(c,d)}$. finally, transform the lattice points (c, d) which exceed the threshold value l_α in $Arrayf_{(c,d)}$ to $e \cdot v_1 + f \cdot v_2 = (ec_1 + fc_2, ed_1 + fd_2)$, thus transform "$(c, d)$ flat" to lattice point (a, b) of "(a, b) flat". These sieve points are what we want. f in "(e, f) flat" is a integer, not the polynomial $f(x)$ mentioned before.

The pseudo code of lattice sieve as follows:

Algorithm1：Lattice sieve

Input： Sieve bound $\mathcal{A} = [-A, A] \times [1, B]$, smooth bound B_α, B_β, where $B_0 < B_\alpha$, $B_1 < B_\beta$, factor base $\mathcal{S}_\alpha, \mathcal{S}_\beta$, big prime q ranged $[q_{\min}, q_{\max}]$, the threshold value l_α, l_β, the number of smooth relationships required: Num

Output： set of smooth relation pairs: Set, the norm of every smooth relation pair (a, b) are N(a-bα) and n(a-bβ), output the standard factorization of it into cache files

1.$q \leftarrow q_{\min}$
2.$Set \leftarrow \emptyset$
3.While $|Set| \leq Num$ do
4. $\mathcal{S}_q \leftarrow \{\langle q, \alpha - s\rangle \in S_\alpha \mid f(s) \equiv 0 \bmod q\}$
5. $S \leftarrow \emptyset$
6. $\mathcal{S}'_\alpha \leftarrow \{\langle p, \alpha - r\rangle \in \mathcal{S}_\alpha \mid p < B_0\}$, $M_\alpha \leftarrow \mathcal{S}_\alpha - \mathcal{S}'_\alpha$
7. $\mathcal{S}'_\beta \leftarrow \{\langle p, \beta - r\rangle \in \mathcal{S}_\beta \mid p < B_1\}$, $M_\beta \leftarrow \mathcal{S}_\beta - \mathcal{S}'_\beta$
8. while $\langle q, \alpha - s\rangle \in \mathcal{S}_q$ do
9. $V_1 \leftarrow (a_1, b_1)$, $V_2 \leftarrow (a_2, b_2)$ /*calculate $\langle q, \alpha - s\rangle$ 的 reduced basisV_0, V_1*/
10. $Arrayf_{(c,d)} \leftarrow 0$, $Arrayg_{(c,d)} \leftarrow 0$ /*array*/
11. for each $\langle p, \alpha - r\rangle \in S_\alpha$ and $cV_1 + dV_2 \in \mathcal{A}$ do
12. do row sieve to $\mathcal{S}'_\alpha$, do vector sieve to M_α
13. for every sieve point (c, d), let $Arrayf_{(c,d)} \mathrel{+}= \log p$
14. end for
15. if $Arrayf_{(c,d)} \geq l_\alpha$
16. $S \leftarrow S \cup \{(c, d)\}$
17. end if
18. for each $\langle p, \beta - r\rangle \in \mathcal{S}_\beta$ do
19. do row sieve to $\mathcal{S}'_\beta$, do vector sieve to M_β
20. for every sieve point (c, d), let $Arrayg_{(c,d)} \mathrel{+}= \log p$
21. end for
22. for $(c, d) \in S$ do
23. if $Arrayg_{(c,d)} \geq l_\beta$
24. $(a, b) \leftarrow cV_1 + dV_2$
25. if $N(a - b\alpha)$ and $N(a - b\beta)$ are $B_\alpha - smooth$ and $B_\beta - smooth$ respectively
26. $Set \leftarrow Set \cup (a, b)$
27. output standard factorization of $N(a - b\alpha)$ and $N(a - b\beta)$into cache files
28. end if
29. end for
30. end while
31. $q \leftarrow$ nextprime in $[q_{\min}, q_{\max}]$
32.End while

5. Experiment and analysis

The lattice sieve filed in CADO-NFS is parameters---task.i and task.polyselect.p, which control lattice sieve filed and polynomal number. z is denoted as the digits of N. We get foluma of task.i and task.polyselect.p by curve fitting.

Task.i= $\mathbf{7.564365459102716 \times 10^{-7} x^3 - 0.0003818181818181724 x^2 + 0.09626475279106744 x + 6.435269993164756}$

Task.polyselect.p = $\mathbf{0.7032631943495017 x^3 - 130.26905454545297 x^2 + 7298.590118022242 x - 120431.66069719614}$

We selected N, g, y_a for solving x_a, where $g^{x_a} = y_a \bmod N$. First factor $N-1$ and solve the discrete logarithm of the square power factor of the modular prime number, respectively. Then, the Chinese Residual Theorem(CRT) is applied to obtain the discrete logarithm of modular $N-1$.

5.1. Computing platform and performance

The computer platforms and performance we used are shown in table 2.

Table2. The computer platforms and performance

Order	Property	Model
Property	Notebook	Server
Model	Hasee p7xxdm2-g	Poweredge-r930
Cpu	Intel® core™ i7-7700 cpu @ 3.60ghz × 8	Intel xeon(r) cpu e7-4820 v4 @ 2.00ghz × 57
Memory	16gb	2.0 tb

5.2. Experiment and comparison

According to (IV.C), task.i ≈ 180000 and task.polyselect.p ≈ 180000 for the solution of the discrete logarithm problem over 130-digit strong primes.

Take data 1(see appendix) as an example, where N is 133-digit, the standard factorization of $N-1$ is as follows:

$$N - 1 = 2 \times q_0 \times q_1.$$

Table 3 shows the application platform, software tools and solution time of two logarithmic solutions of modular factor.

Table 3. Process of solving logarithm of 2 modulus factor of data

Q	q_0	q_1
Software	C-ntl:Pollard rho	Cado-nfs
Platform	1	2
Time	7.483h	196.25h

The total solution time is 196.25h. With current parameters, we cannot have a result with same computer on limit time.

6. Summary

We have sorted out the most efficient algorithm for solving discrete logarithm problems over "large" finite fields—general number field sieve method (GNFS). The lattice sieve expend the most time of GNFS for solving problems, so we find the lattice sieve field to improve efficiency.

For parameter configuration, we only consider parameter selection of task.i and polynomial selection number task.polyselect.p of the lattice. In the next step, parameter configuration of other values can be considered to increase the influence of single logarithm's prime factor bound and large prime number bound on the solving efficiency of discrete logarithm.

APPENDIX

Data 1(133-digit)

p
=45992933402075745330668784328919628957859520056454767396667335306613126261688142541064796460840362246501407092541683873197660223782 7

g =2

y_a
=24646608434430150096328834471403771151616233462883871289489054016574018822235654321917328287596878941595797623884843805348560706349 3

References

[1] Stinson D R. Cryptography: Theory and practice. 3rd ed[J]. IEEE Transactions on Medical Imaging, 2007, 26(7):917-24.
[2] Coppersmith D, odlzyko A M, Schroeppel R. Discrete logarithms in GF (p)[J]. Algorithmica, 1986, 1(1-4):1-15.
[3] Gordon D M. Discrete logarithms in GF(P) using the number field sieve[M]. Society for Industrial and Applied Mathematics, 1993.
[4] Kleinjung T. On Polynomial Selection for the General Number Field Sieve[J]. Mathematics of Computation, 2006, 75(256):2037-2047.
[5] Fried J, gaudry P, heninger N, et al. A Kilobit Hidden SNFS Discrete Logarithm Computation[J]. 2016.
[6] Keqin F. Algebraic Number Theory [M]. Science Press, 2000.
[7] Yujie Z, dengguo F. Public Key Cryptography Algorithm and Its Fast Implementation [M]. National Defence Industry Press, 2002.
[8] Shi B, tho E, and E, et al. Factorisation of RSA-704 with CADO-NFS[J]. HAL - INRIA, 2012.
[9] Bai S, bouvier C, kruppa A, et al. Better polynomials for GNFS[J]. Mathematics of Computation, 2015, 85(298).
[10] Xueqiao Z. Implementation of GNFS algorithm for discrete logarithm problem on GF(p)[D]. Shandong: Shandong University, 2009.
[11] Bai S, brent R P. On the Efficiency of Pollard's Rho Method for Discrete Logarithms.[C]// Theory of Computing Proc Fourteenth Computing: the Australasian Theory Symposium. DBLP, 2008:125-131.
[12] Murphy B, brent R P. On Quadratic Polynomials for the Number Field Sieve[J]. Australian Computer Science Communications, 1997, 20:199-213.
[13] Kleinjung T. Polynomial Selection[R]. http://cado.gforge.inria.fr/workshop/slides/kleinjung.pdf
[14] Pollard J M. The lattice sieve[M]// The development of the number field sieve. Springer Berlin Heidelberg, 1993:43-49.
[15] Cavallar S. Strategies in Filtering in the Number Field Sieve[M]// Algorithmic Number Theory. Springer Berlin Heidelberg, 2000:209-231.
[16] Timothy V. Pivoting To Preserve Sparsity[R]. https://vismor.com/documents/network_analysis/matrix_ algorithms/S8.SS3.php
[17] Schirokauer O. Discrete Logarithms and Local Units[J]. Philosophical Transactions Physical Sciences & Engineering, 1993, 345(1676):409-423.
[18] Joux A, lercier R. Improvements to the general number field sieve for discrete logarithms in prime fields. A comparison with the gaussian integer method[J]. Mathematics of Computation, 2003, 72(242):953-967.
[19] Wiedemann D H. Solving sparse linear equations over finite fields[J]. IEEE Transactions on Information Theory, 2008, 32(1):54-62.
[20] Joux A. Algorithmic Cryptanalysis[M]. Chapman & Hall/CRC, 2009.
[21] Coppersmith D. Solving Homogeneous Linear Equations Over GF(2) via Block Wiedemann Algorithm[J]. Mathematics of Computation, 1994, 62(205):333-350.
[22] Joux A, pierrot C. The Special Number Field Sieve in F_{P^n} [M]. Pairing-Based Cryptography – Pairing 2013. Springer International Publishing, 2013:45-61.
[23] Hayasaka K, takagi T. An Experiment of Number Field Sieve over GF(p) of Low Hamming Weight Characteristic[J]. Lecture Notes in Computer Science, 2011, 109:191-200.