Mathematical and Computer Modelling 55 (2012) 161–169

Contents lists available at SciVerse ScienceDirect

Mathematical and Computer Modelling

journal homepage: www.elsevier.com/locate/mcm

Discrete logarithm problem using index calculus method

R. Padmavathy [a,*], Chakravarthy Bhagvati [b]

[a] National Institute of Technology, Warangal, Andhra Pradesh, India
[b] University of Hyderabad, Hyderabad, Andhra Pradesh, India

ARTICLE INFO

Article history:
Received 25 September 2010
Received in revised form 1 February 2011
Accepted 14 February 2011

Keywords:
Discrete logarithm problem
Index calculus method
Linear sieve
Lanczos method

ABSTRACT

This paper presents a new methodology for the pre-computation phase of the index calculus method (ICM), which is a popular attack on solving the Discrete Logarithm Problem (DLP). For a prime field Z_p^* of a multiplicative cyclic group, with a given generator $g \in Z_p^*$ and an element $y \in Z_p^*$, the problem of finding x, such that $g^x = y \pmod p$, is known as the DLP. The ICM has two steps: pre-computation and individual logarithm computation. In the pre-computation step, the logarithms of elements from a subset of the group, known as a *factor base*, is computed. In the second step, the DLP is computed with the help of the pre-computed logarithms of a factor base. The present work focuses on the pre-computation step. Three steps that have a significant impact on the performance of the pre-computation step are generating a system of equations on the logarithms of the primes in the factor base, reducing its size for computation efficiency, and solving the system for logarithms of elements in the factor base. It is shown that the performance of ICM is improved through reduction in size of the system of equations producing a smaller size matrix for the third step by combining the reduction and generation steps. The size of the factor base, sieve length (length of elements to be searched for generating the linear relations) and the ratio between the rows and columns in the linear relations are viewed in combination and studied in detail. We have achieved 30%–40% improvement in the performance of ICM. Even for a smaller size problem (≈100 bits), the running time is reduced to 667 s from 937 s.

1. Introduction

In this paper we propose a new methodology for the pre-computation step of the Index Calculus Method (ICM) to solve the Discrete Logarithm Problem (DLP). Let a group $(G, *)$ consist of a set G and a binary operation $*$. The order of an element, say a, of a finite group G is defined as the smallest value t such that $a^t = a * a * a * \ldots = 1$. For a given prime field Z_p^* of a multiplicative cyclic group, a generator $g \in Z_p^*$ and an element $y \in Z_p^*$, the problem of finding x, in the range of $0 \leq x \leq p-2$, such that $g^x = y (mod\ p)$, is known as the DLP.

Apart from the exhaustive search to solve the DLP, a well known deterministic algorithm is Shank's baby-step-giant-step algorithm [1]. Its space and time complexities are both $O(\sqrt{n})$. Pollard Rho, a probabilistic algorithm, has a similar square root running time but avoids the large space requirements [2]. The Pohlig–Hellman method [3] reduces the DLP in a field to small subgroups. For example, if $p - 1$ is a product of small factors q_i that are relatively prime to each other, then the method reduces the discrete logarithm $x\ mod\ p$ to $x_i\ mod\ q_i$, computes $x_i\ mod\ q_i$ in each q_i and finally combines the results using the Chinese Remainder Theorem.

* Corresponding author. Tel.: +91 9440173819.
E-mail addresses: r_padma3@rediffmail.com (R. Padmavathy), chakcs@uohyd.ernet.in (C. Bhagvati).

0895-7177/$ – see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.mcm.2011.02.022

The best known algorithm for DLP over integer fields is ICM [4]. DLP can be computed in sub-exponential time using ICM if there is more structure to the group beyond the set of elements and the group operation. ICM uses a small subset of group elements called the factor base B and expresses all the elements as a product of the members of the factor base [5]. In a prime field F_p, where we identify the field elements with integers in $0, 1 \ldots p-1$, the factor base is almost always chosen to contain all prime numbers less than a prescribed bound B.

ICM is also one of the most researched methods for DLP. An efficient way of solving DLP using ICM was first presented by Coppersmith and Odlyzko [4]. Later LaMacchia and Odlyzko [6] pointed out that even for the naïve approach one can obtain a running time of the form $\exp((c + o(1))(\log p)^{\frac{1}{2}}(\log\log p)^{\frac{1}{2}})$ for some constant c. Research is in progress for obtaining better values of c. Another popular variant of ICM is the number sieve field with a heuristic running time of the form $\exp((c + o(1))(\log p)^{\frac{1}{3}}(\log\log p)^{\frac{2}{3}})$ [7–9]. Discovery of methods analogous to ICM on an elliptic curve group is a well-known open problem [10]. Recently, ICM is formulated for hyper elliptic curves [11].

1.1. Index calculus method

The Index Calculus Methods are the most prominent collection of algorithms that have successfully used additional knowledge of the underlying groups to provide sub-exponential algorithms. The basic idea, which goes back to Kraitchik [5] is that if

$$\prod_{i-1}^{m} x_i = \prod_{j-1}^{n} y_i \tag{1}$$

for some elements of $GF(q)^*$, then

$$\sum_{i-1}^{m} \log_g x_i \equiv \sum_{j-1}^{n} y_j q - 1. \bmod \tag{2}$$

If we obtain many equations of the above form, and they do not involve too many x_i and y_i, then the system becomes computationally feasible. The algorithm has two steps:

- A pre-computation step where the logarithms of $\log_g b$ of all members of the factor base are obtained, where g is a generator and $b \in B$ is an element of the factor base.
- A computation step, which computes $g^a y$ for an $a \in G$ until the result factors over the factor base, in turn leading to the required logarithm $log_g y$ [12].

The general algorithm for DLP may now be described in detail as:

INPUT: a generator g of a cyclic group G of order n and an element y
OUTPUT: $\log_g y$

Step 1. Precomputation

- Select a factor base $S = \{p_1, p_2, \ldots, p_t\}, p_i \in G$ for $1 \leq i \leq t$ such that a significant number of elements of G may be efficiently expressed as a products of elements from S.
- Generate a linear system with the logarithms of p_i as unknowns. Such a system may be generated as follows:
  (i) Select a random integer k, such that $0 \leq k \leq n-1$ and compute g^k
  (ii) Try to write g^k as a product of elements in S

$$g^k = \prod_{i=1}^{t} p_i^{c_i}, \quad c_i > 0, \tag{3}$$

  for any k. Then, $k \equiv \sum_{i=1}^{t} c_i \log_g p_i$
  (iii) Repeat the above steps to get the $t + c$ equations
- The linear system is reduced to a smaller size using the structured Gaussian method. This step is optional one and used when a large system is generated in the previous step.
- Solve the linear system to obtain $\log_g p_i$ for $1 \leq i \leq t$.

Step 2. Computation of the discrete logarithm

- Select a random integer, k, $(0 \leq k \leq n-1)$ and compute yg^k
- Try to write yg^k as a product of elements in S

$$yg^k = \prod_{i=1}^{t} p_i^{d_i} \tag{4}$$

for any k. Then, $\log_g y = (\sum_{i=1}^{t} d_i \log_g p_i - k) \bmod n$

Table 1
Optimal values for parameters.

Parameter	Theoretical	Experimental
B	$O(L_p(\frac{1}{2}; \frac{1}{2}+O(1)))$	$3.33L_p(0.5, 0.476)$
C	$O(L_p(\frac{1}{2}; \frac{1}{2}+O(1)))$	$2.78L_p(0.5, 0.417)$

The focus of this paper is on the *pre-computation step* and the steps involved in it are

Step 1. *Generating* a linear system of equations with logarithms of elements of factor base as unknowns

Step 2. Optionally, *reducing* the system into smaller size. This step is an optional one and used when a large system is generated in the previous step.

Step 3. *Solving* the system to find logarithms of elements in the factor base.

Progress in improving the performance of ICM is by efficiently generating, reducing and solving the linear system. Coppersmith and Odlyzko [4] presented three versions of generating the linear system namely *linear sieve, Gaussian integer* and *cubic sieve*. Later LaMacchia and Odlyzko [6] reported the implementation of two (linear sieve and Gaussian integer) of these three methods. An implementation of cubic sieve method is reported by Abhijit Das and Veni Madhavan [13].

The reduction step is also studied in detail. LaMacchia and Odlyzko investigated this problem through the experimental work on structured Gaussian elimination for a problem size of 192 bits, in which it is mentioned that the linear system ought to be sparse and the number of relations generated should be considerably larger than the number of unknowns, i.e., the size of the factor base. They stated that this condition is necessary for obtaining a good reduction using the structured Gaussian approach [14]. Recently Roberto Avanzi et al. [11] proposed a new filtering technique called the *harvesting method* to reduce the size of the linear system. Their method removes duplicate equations and singletons from the system. It was claimed that such a reduction led to improving the performance of ICM by more than 30%.

The solving phase in pre-computation is the most significant step in ICM with respect to the computational aspects. Index calculus algorithms require solutions of large sets of linear equations over finite fields. The introduction of structured Gaussian elimination [14] and of the finite field versions of the Lanczos and conjugate gradient algorithms [6], and the subsequent discovery of the Wiedemann [15] algorithm led to a reduction in the estimate of the difficulty of the equation solving phase. Still, the solving phase remains a bottleneck.

From the above discussion, it is observed that the third step in the pre-computation is the most significant and has a dominant influence on the running time of ICM. One way to improve the performance in this step is by reducing the size of the linear system to be solved. Thus, the key result in the present paper is the discovery of relationships between the parameters involved in generation and reduction phases that lead to small linear systems for the third phase.

1.2. Problem statement

From the several methods proposed in the literature for the pre-computation step, we focus on the linear sieve method for generating the linear system of relations. An improvement in the performance based on a different method, known as the random method, is reported in [16] and a variant of ICM, when $p-1$ has many small factors is reported in [17]. The linear sieve method uses a linear sieve to generate the linear system, structured Gaussian method for reduction and any of linear algebra techniques such as Lanczos, conjugate gradient and Wiedemann for solving the linear system. In our paper, we analyze the linear sieve generation method in conjunction with structured Gaussian elimination and Lanczos method.

We identify the parameters that influence the running time significantly as the bound (B) based on which the factor base is defined, the sieve length (C) i.e., a range of numbers chosen for the *smoothness* test. The smoothness test is used to form the linear relations. Traditionally, the steps in ICM are viewed into two groups such as generation as one group and reduction and solving as another. Certain optimal selections of B and C are proposed in the literature with this perspective, i.e, to *generate* as small a system as possible.

The time complexity of the pre-computation step, optimal Bound (B) and optimal sieve length (C) are derived in the literature in terms of $L_p[s; c]$ function. $L_p[s; c] = \exp^{c(\log p)^s(\log\log p)^s}$. These two parameters are also analyzed experimentally by considering half the time for generation and half the time for solving in the paper [12] and the results are given in Table 1.

We show that if the phases in the pre-computation step are grouped into generation and reduction as one group and solving as another, it allows us to define a *third* parameter, *reduction ratio* (R) in addition to B and C. Reduction ratio may be defined as the ratio of the number of relations going *into*, and the number resulting *after* the structured Gaussian elimination phase. By optimizing the tuple (B, C, R), we show empirically that the running time improves by a factor of 30%–40% over the results published in the literature.

The paper is organized as follows: the following section discusses the approach followed to obtain the optimal tuple (B, C, R). Section 3 presents the algorithm developed for the pre-computation step of linear sieve method with the experimental results. Section 4 discusses the comparative analysis between the traditional and the proposed algorithm and Section 5 is the conclusion.

2. Analysis of the linear sieve method

Traditionally, the steps in linear sieve method are viewed into two groups with generation as one group, and reduction and solving as another. In the present study too, the steps are viewed as two groups. However, we consider generation and reduction of relations as one group and solving as another. The linear system is generated and reduced such that a smaller matrix is produced for the third step. Odlyzko [14] hypothesized, based on empirical evidence, that the more number of relations than unknowns, the more is the reduction using a structured Gaussian method. Thus, the problem is formulated as to investigate how the parameters that generate and reduce the linear system be chosen to generate as small a matrix as possible for the third step.

The following methodology is used to obtain an optimal tuple (B, C, R) empirically.

- Analyze the structured Gaussian method to obtain R that produces as high a reduction is possible.
- Obtain a suitable pair (B, C), which will produce a linear system with the required R.

2.1. Empirical analysis on structured Gaussian elimination

In the first phase of our experiments, the performance of structured Gaussian method is analyzed. This method reduces a large system into a smaller system and works efficiently when the matrix has a greater number of relations than unknowns [14,18]. The performance of structured Gaussian elimination is analyzed using an appropriate database of safe primes and we found there appears to be an optimal ratio between the rows and columns of linear system. As safe primes are considered hard for exponential time algorithms such as Shanks, Pollard-Rho, Pollard-Lambda to name a few, they are analyzed in this paper and solved through linear sieve method.

The experiments are done by first creating a file of test cases. The file consists of a list of tuples (p, g, q, m, M, s, S) where p is the safe prime for generating the group G, g is a generator of the group G, q is the largest prime factor of $p - 1$, m is the minimum bound, M is the maximum bound, s is the minimum sieve length, S is the maximum sieve length.

Given the above, the tuples are computed as follows:

- Select a safe prime p approximately of problem size between 25 and 40 digits and obtain the largest prime factor, say q of $p - 1$.
- Find the generator of Z_p^*.
- Calculate (m, M) and (s, S) approximately using the minimum and maximum bound and sieve length for linear sieve from the literature [12].

Having built up the file the following procedure is computed.

Algorithm 1 Finding Optimal Reduction Ratio

1: Read the tuple from the data file
2: BEGIN
3: For every B from $m \leq B \leq M$
4: BEGIN
5: For every C from $s \leq C \leq S$ repeat the following
6: BEGIN
7: Generate the Linear relation for p, B and c
8: If the ratio between rows and column is less than 1 skip the following step
9: Reduce the system using structured Gaussian elimination method and solve the linear system using Lanczos.
10: END
11: Plot the reduced size of matrix to the ratio between the rows and the columns and obtain the ratio that minimizes the size of the reduced matrix for a bound B.
12: END
13: From the results of optimal ratio and the size of reduced matrix of each bound B, obtain the ratio that minimizes the size of reduced matrix for the given problem p.
14: END
15: Plot the problem size to the ratio, which is obtained in the previous step and obtain the ratio that can be considered as the ratio that achieves the maximum reduction.

The first set of experiments on the structured Gaussian method are designed to find the ratio R, which produces the best reduction for the set of problems taken from the data file. The plot of the ratio versus the size of reduced matrix for a given problem shows that when the ratio increases the size of reduced matrix decreases and after reaching some point it starts increasing. The valley gives the best reduction for a given problem. Fig. 1(a) shows the reduction when ratio increases for a specific problem of size 25 digits. This figure shows that the size of reduced matrix decreases, when the ratio increases and starts increasing after reaching some point. In particular, Fig. 1(a) also shows that the linear relations of a 25 digits problem is reduced to the smallest system, when the ratio is around 1.8. This ratio is considered as the optimal ratio for this problem. Other problems are also analyzed similarly and optimal ratios are obtained.

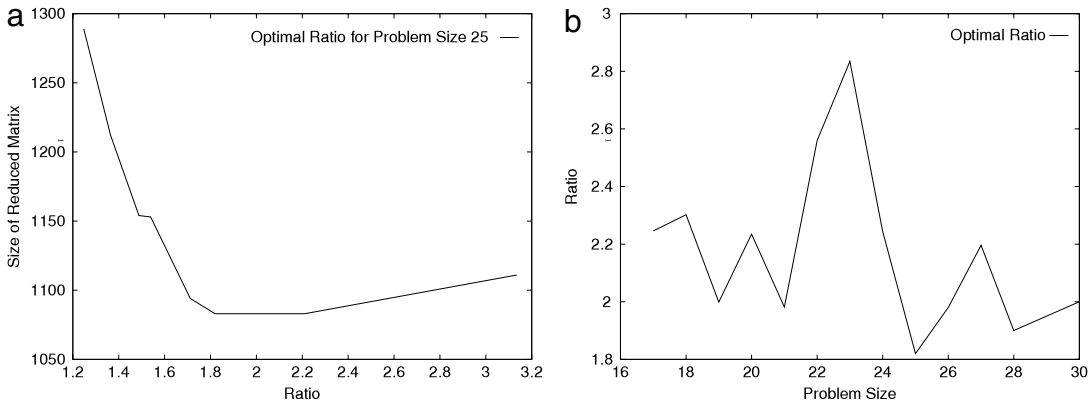

Fig. 1. (a) Optimal ratio for problem of size 25. (b) optimal ratio of problems from 17 digits to 30 digits.

From the different sized problems, it is observed that the reduced size does not vary much for ratios between 1.7 and 2.4. Fig. 1(b) shows the optimal ratio between the rows and the columns obtained by implementing the above heuristic algorithm for problems of size between 17 and 30 digits based on the bound and sieve length values given in literature. Fig. 1(b) shows that the optimal ratio ranges from 1.8 to 3.0.

We have chosen the optimal ratio as 2.0 for deciding the number of relations to be generated in the generation step of pre-computation step. In other words, we generate twice as many relations as there are unknowns.

2.2. Analysis on bound (B) and sieve length (C) for the ratio (R)

After fixing R as 2.0, the next set of experiments are conducted on obtaining the relationship between B and C, that could produce the linear relations in the required ratio 2.0. The following procedure is adapted for retrieving the above mentioned pair.

Algorithm 2 Compute B and C for achieving optimal R

1: Read the tuple from the data file.
2: BEGIN
3: For every B from $m \leq B \leq M$ repeat the following
4: BEGIN
5: Choose the sieve length (C) from the range $s \leq C \leq S$
6: Generate the linear relation till the ratio is met otherwise increment or decrement the sieve length
7: Obtain the sieve length
8: END
9: Plot bound vs sieve length
10: END
11: Find the relationship between bound and sieve length for fixed ratio 2.0 that minimizes the size of the reduced matrix from the above results

The plots of bound versus sieve length for a set of different size problems show that the sieve length is exponentially high when the bound is small. It starts decreasing as the bound increases, and finally saturates. The number of primes in the factor base of smaller bounds are less compared with larger bounds. This makes the search for smooth elements difficult for smaller bounds and needs a greater number of elements (sieve length) to be searched. This leads to increasing the time for the generation step. On the other hand for larger bounds, the search for smooth elements is easy and needs a greater number of smooth elements to form the linear system. This leads to increase in the time for solving step.

Fig. 2 shows that the result on sieve length against the bound for the fixed optimal ratio for the problems of size around 25 digits. Fig. 2 reports that the sieve length is exponentially high for small B. Such a small B will drastically increase the running time of generation step because of the large range of numbers to be tested for smoothness. On the other hand for large B, the number of primes in the factor base is large leading to increased time complexity of the solving step even though the sieve length is small. Thus, there is an optimal choice of B and C which simultaneously reduces the generation and solution phases. Fig. 2 also shows that the optimal bound ranges from 2500 to 10,000 and the sieve length ranges from 1000 to 6000. With the above results we developed an algorithm for the pre-computation step of the linear sieve method. In the proposed algorithm the linear system is generated iteratively by satisfying the ratio and the relationship between the bound and sieve length. The initial bound is the lower bound for the linear sieve method. The following section demonstrates the algorithm developed for the pre-computation step of the linear sieve method along with the experimental results.

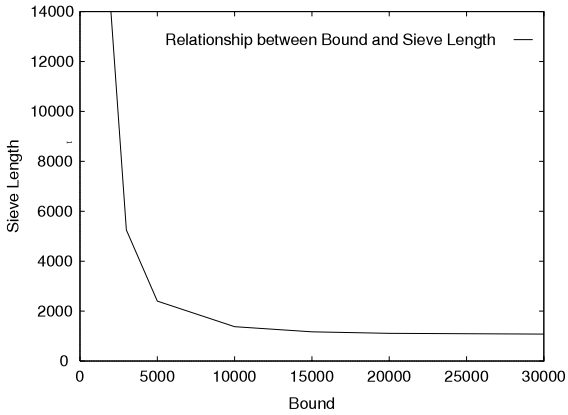

Fig. 2. Relationship between bound and the sieve length for problem of size 25.

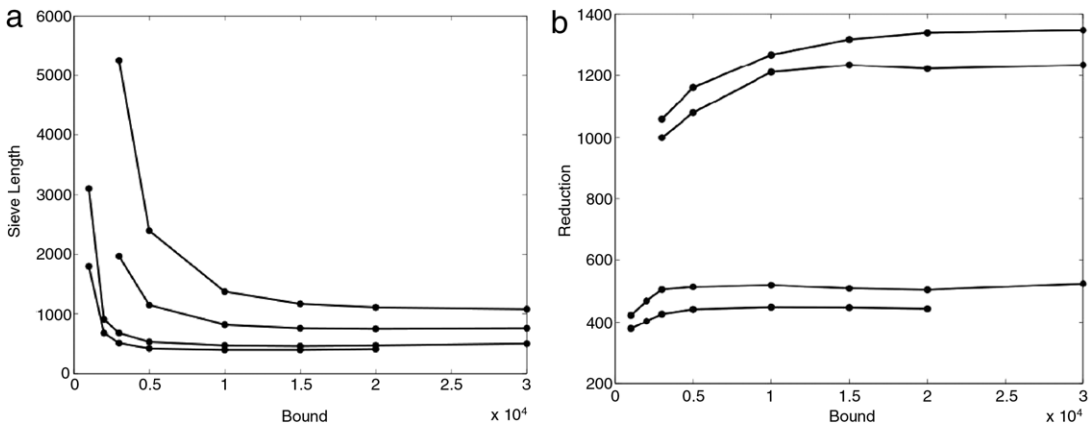

Fig. 3. (a) Relationship between bound and sieve length of different size problems. (b) Size of reduced matrix for different size problems.

Fig. 3(a) shows the relationship between Bound and Sieve length for different size of problems. Similarly, Fig. 3(b) shows the size of reduced matrix, when bound increases for different sizes of problems. This figure shows that the size of the reduced matrix is small, when the bound is small. It starts increasing as the bound increases and saturates after reaching some point. From the figure, it can be said that the bound also influences the percentage of reduction. The reduction is high (the size of reduced matrix is low) for smaller bounds than higher bounds.

3. Algorithm for pre-computation step of linear sieve method

In the present study the parameters bound, sieve length and the ratio are studied collectively as a group. An algorithm is developed based on the results we obtained. The performance of the algorithm is compared with the information given in the Table 1 reported in Section 1.2. The algorithm presented in the following section is based on two pieces of evidence from the empirical study of the previous section. One is the performance of structured Gaussian elimination as it decides the optimal ratio between the rows and the column and the other is the dependency between the bound and the sieve length, that corresponds to optimal ratio. These are considered as key issues as they are independent from an implementation point of view and need to be known to develop an algorithm to improve the performance of the linear sieve method in all aspects. Initial value for the bound is computed from the lower bound for the linear sieve reported in the literature [12]. From the conjecture of bound and the sieve length presented in the Section 1.2 and the empirical values reported in the Table 1, the initial value for the sieve length is assigned as the initial value of the bound. The linear relation is generated iteratively until the optimal ratio and the relationship between the bound and the sieve length are satisfied.

3.1. Improved algorithm for linear sieve method

In Algorithm-3, step 8 is to generate the relation in required ratio, step 22 is to reduce the linear system into smaller one and step 23 is to solve the linear system. The terminating condition $C \leq k * B$ is chosen from the heuristic analysis given above, where k may be chosen arbitrarily ≤ 2. In the current study, the k is chosen as 1. The variable l is chosen from a

Algorithm 3 Algorithm for the pre-computation step of linear sieve method

INPUT : p (Problem to be solved)
OUTPUT : RESULT (The discrete logarithm for the primes in the factor base)
VARIABLES : $n.B$, B, C, $Ratio$, $tag = 0$, $flag = 0$, p

1: Read the input(p)
2: Set $n.B = \lceil e(\sqrt{(\log(p))}\rceil$; $C = n.B$; $B = n.B$.
3: Repeat the following till the required size of matrix is obtained for the bound and sieve length
4: BEGIN
5: $C = k * B * m$, where m is to be close to 0
6: Repeat the following steps 7 to 20
7: BEGIN
8: Generate the relations using the linear sieve method
9: Compute the ratio between the rows and columns
10: **if** $C \leq k * B$ and ratio is equal to 2.0 **then**
11: Go to step 22
12: **else**
13: **if** $C = k * B$ and ratio is $\neq 2.0$ **then**
14: $B = B + l * n.B$, where l is close to zero when ratio is close to 2.0 or close to 1 when ratio is close to zero.
15: Go to step 5
16: **end if**
17: **else**
18: $C = k * B * m$, where m is to be varied based on the value of C and the ratio
19: **end if**
20: END
21: END
22: Reduce the relation using the structured Gaussian method
23: Solve the linear system using the Lanczos method

set {1, .5, .333} based on the ratio set {<1.2, between 1.2 and 1.6, >1.6}. Similarly, the variable m is chosen from a set {1, .75, .5} based on C and the ratio set as {<.75, between.75 and 1.3, ≥1.3}.

3.2. Performance study and numerical results

We implemented Algorithm-3 to measure the efficiency of the algorithm and conducted experiments on an appropriate data base of safe primes i.e., primes of the form $2q + 1$, where q is also a prime. The use of safe prime is considered as safe for cryptographic applications. The advantage of using safe primes is that, all group elements of Z_p^* other than ± 1 are known to have order either q or $2q$. Consequently $g = 2$ known to be of generator for full group of order $2q$ elements if 2 is quadratic non-residue *modp* and generator for the group of order q when 2 is quadratic residue *modp*. In modular exponentiation using $g = 2$, the multiples cost is decreased substantially and it almost disappears, only the squaring cost remains in the exponentiation. The another usage of this prime is that it precludes the attacks, which are working on the small factors of $p-1$. Apart from the large prime factor q, to retrieve the partial information about the exponent x, the partial Pohlig–Hellman decomposition is used for random primes. This way of obtaining the partial information about the x yields only one bit of information from safe primes. In the present study, the pre-computation step of the linear sieve method is implemented using the algorithm discussed in the previous section to solve the DLP defined on the safe primes. The data base of safe primes is created based on the following algorithm.

Algorithm 4 Algorithm to create a data base of safe primes

1: Select k between 15 and 50.
2: Select q, an integer of size k digits.
3: Check $2q + 1$ is a prime or not using probabilistic primality test algorithm.
4: Find the generator g such that g should be a prime and $g \in FB$, where FB is a factor base formed from the lower bound of linear sieve.
5: Create a 3-tuple as (p, q, g), where p is a prime of the form $2q + 1$, q is a prime factor of $p - 1$ and g is a generator.

Having built up the database the following algorithm is executed to test the Algorithm-3 discussed in the previous section.

The above algorithm is executed on a data base of safe primes and the reports are tabulated in Table 2. Table 2 list the average Running time and the number of iterations for problems of size from 17 digits to 50 digits. The algorithm is tested

Algorithm 5 Algorithm to apply the newly proposed Algorithm-3

1: Read the 3-tuple (p, q, g).
2: Generate the linear relations using the **Algorithm-3**.
3: The matrix of linear relations is reduced using structured Gaussian elimination.
4: The reduced linear system is of the form $Ax = 0$, where A is the coefficient matrix and x is a one dimensional unknown matrix.
5: BEGIN
6: This linear system is transformed into the form $Ax = B$, where B is formed by using the logarithm of the generator.
7: The linear system is to be solved for logarithms $mod p - 1$.
8: The linear system is solved for logarithms $mod\ q$ by using the Lanczos method.
9: The logarithms $mod p - 1$ of factor base elements can be obtained by using Chinese Remainder Theorem from logarithms $mod\ q$ and $mod\ 2$.
10: END
11: keep the running time for each tuple.
12: Repeat the above procedure for all the tuples.

Table 2
Results on improved linear sieve method with $k = 1$.

Problem size in digits	Running time in s	Bound	Sieve length	No of iteration
17	4	1042	795	2
18	7	1250	895	2
19	11	1492	1067	2
20	14	1770	1255	2
21	26	2094	2094	2
22	32	2466	2466	2
23	53	2894	2894	2
24	72	3384	3384	2
25	121	4000	4000	2
26	171	4586	4586	2
27	238	5314	5314	2
28	338	6140	6140	2
29	459	7200	7200	2
30	677	8200	8200	2
31	985	10,897	10,897	3
40	6 h	38,000	38,000	3
45	–	81,000	81,000	3
50	–	160,000	160,000	4

Table 3
Comparison between the running time of improved linear sieve method and the empirical results in Table 1 with $k = 1$.

Problem size in digits	Bound from Table 1	Bound obtained from Algorithm	Running time using results from Table 1 in sec	Running time of Algorithm-3 in s
17	998	1042	4	4
18	1233	1250	6	7
19	1515	1492	10	11
20	1900	1770	13	14
21	2260	2064	26	26
22	2740	2466	38	32
23	3310	2894	58	53
24	4000	3384	88	72
25	4800	4000	130	120
26	5730	4586	191	171
27	6840	5314	265	238
28	8140	6140	395	338
29	9660	7200	578	459
30	11440	8200	947	677

for both sieving and solving phase up to 40 digits, beyond that only linear relation is generated and the parameters bound and sieve length are obtained.

Similarly the traditional linear sieve method is executed with the same data base of safe primes except that the linear relation is created using the bound and sieve length computed from the Table 1 discussed in Section 1.2. The running time is compared with the running time of the linear sieve method implemented using the proposed algorithm discussed in the previous section. Table 3 lists the difference in the running time between the traditional and Algorithm-3.

4. Discussion

Table 3 ensures the performance of an improved pre-computation step of the linear sieve method. The investigation of the relationship among the parameters in the tuple (B, C, R) aid in improving the performance of the linear sieve method. This section reports the comparative analysis between the parameters listed in the Table 1 discussed in Section 1.2 with tuple (B, C, R) investigated in the present study. The bound (B) and the sieve length (C) in the Table 1 are functions of problem size (p). The bound and the sieve length in the tuple (B, C, R) are related such that $C \leq 2B$ and this is obtained through the experimental results for the fixed ratio R. The bound B chosen for the list of problems in the newly proposed pre-computation step is comparatively less with the empirical results given in the Table 1. The sieve length presented in the Table 1 is comparatively less with the sieve length chosen in the newly proposed algorithm.

For example a 20 digit problem is solved by using a bound $B \approx 1900$ and the sieve length of $C \approx 800$ obtained from the Table 1, this shows that the B and C are related with independent of ratio as $C = .421B$. The bound and the sieve length chosen for the same problem are $B \approx 1770$ and $C \approx 1400$ with dependence on the ratio as $R = 2.0$, when it is solved by using the newly proposed pre-computation step of the linear sieve method. Through the experimental results we obtained $C \approx .5B$ to $.9B$ for smaller problems and $C = B$ for problems of size ≥ 21. In the present study, the problems of size up to $\approx$40 digits are solved by using both generation and solving steps of Algorithm-3 and the parameters in the tuple (B, C, R) are $R \approx 2$ and $C = B$. The problems of size up to $\approx$55 digits are solved by using a generation step and the tuple (B, C, R) are $R \approx 2$ and $C = B$.

The use of tuning certain parameters by considering them as a whole (B, C, R) together to produce the most reduced system of equations for any general solver such as Lanczos is demonstrated in the current study. The selection of B, C, R and experimentation presented in the paper is specific to the linear sieve and structured Gaussian reduction. However, a similar approach may be easily be defined for any approach based on generating, reducing and solving equations for discrete logarithms of a factor base such as cubic sieve.

5. Conclusion

In this paper, we discussed a new approach to improve the performance of ICM. An improved algorithm is proposed for the pre-computation step of ICM and new results are reported on the parameters used to generate and solve the relations. The parameters are analyzed as a group on the performance of ICM. We showed that including reduction alongside generation permits the selection of B and C such that the overall time needed for the pre-computation step of ICM is reduced. Also, we reported that, if the number of relations is twice the number of unknowns, we get improved performance in the structured Gaussian step resulting in a smaller set of relations for solving by the Lanczos method. Finally we conclude that the performance of ICM is improved by 30% to 40% by selecting the smallest bound B such that $C \leq 2B$.

References

[1] D.E. Knuth, The art of Computer Programming, vol. 3, Addison-Wesley, 1973.
[2] J.M. Pollard, Monte Carlo methods for index computation (mod p), Mathematics of Computation 32 (143) (1978) 106–110.
[3] S. Pohlig, M. Hellman, An improved algorithm for computing logarithms over GF(p) and its cryptographic significance, IEEE Transaction on Information Theory 24 (1) (1978) 106–110.
[4] D. Coppersmith, A.M. Odlyzko, R. Schroeppel, Discrete logarithms in GF(p), Algorithmica 1 (1986) 15.
[5] McCurely, The discrete logarithm problem, Symposium of Applied Mathematics 42 (1990) 49–74.
[6] B.A. LaMacchia, A.M. Odllyzko, Computation of discrete logarithms in prime fields, Designs, Codes and Cryptography 1 (1) (1991) 46–62.
[7] D.M. Gordon, Discrete logarithms in GF(p) using the number field sieve, SIAM Journal of Discrete Mathematics, 6 (1) 124–138.
[8] D. Weber, Computing discrete logarithms with the general number field sieve, in: Proceeding of ANTS II, in: Lecture Notes in Maths, vol. 1122, 1996, pp. 99–114.
[9] D. Weber, T. Denny, The solution of McCurleys discrete log challenge, in: Proceeding of Crypto98, in: LNCS, vol. 1462, 1998 pp. 458–471.
[10] J. Silverman, The xedni calculus and the elliptic curve logarithm problem, Design Codes and Cryptography 20 (2000) 5–40.
[11] R.M. Avanzi, H. Cohen, C. Doche, G. Frey, T. Lange, K. Nguyen, F. Vercauteren, Handbook of Elliptic and Hyperelliptic Curve Cryptography, CRC Press, 2005.
[12] C. Studholme, Discrete logarithm problem, Research paper requirement (milestone) of the Ph.D. program at the University of Toronto, June 21, 2002.
[13] D. Abhijit, C.E. Veni Madhavan, On the cubic sieve method for computing discrete logarithms over prime fields, International Journal of Computer Mathematics 82 (12) (2005) 1481–1495.
[14] B.A. LaMacchia, A.M. Odlyzko, Solving large sparse linear systems over finite fields, in: LNCS, vol. 537, 1991 pp. 109–133.
[15] D.H. Wiedemann, Solving sparse linear equations over finite fields, IEEE Transaction on Information Theory 32 (1986) 54–62.
[16] R. Padmavathy, Chakravarthy Bhagvati, Performance analysis of index calculus method, Journal of Discrete Mathematical Sciences and Cryptography 12 (3) (2009) 72–91.
[17] R. Padmavathy, Chakravarthy Bhagvati, Index calculus method based on smooth numbers of ± 1 over Z_p^*, International Journal of Network Security (accepted).
[18] R. Lambert, Computational Aspects of Discrete Logarithms, Ph.D. Thesis, 1996.