Function Field Sieve

Recent advances in the index-calculus attack on the discrete logarithm problem

A Thesis

submitted to
Indian Institute of Science Education and Research Pune
in partial fulfillment of the requirements for the
BS-MS Dual Degree Programme

by

K Hariram

Indian Institute of Science Education and Research Pune
Dr. Homi Bhabha Road,
Pashan, Pune 411008, INDIA.

March, 2015

Supervisor: Dr. Ayan Mahalanobis
© K Hariram 2015
All rights reserved

This is to certify that this dissertation entitled Function Field Sieve: Recent advances in the index-calculus attack on the discrete logarithm problem submitted towards the partial fulfilment of the BS-MS dual degree programme at the Indian Institute of Science Education and Research Pune represents research carried out by K Hariram at IISER Pune under the supervision of Dr. Ayan Mahalanobis, Assisstant Professor, Department of Mathematics during the academic year 2014-2015.

Dr. Ayan Mahalanobis

Committee:
Dr. Ayan Mahalanobis
Dr. Bhaskar Balasubramanyam

Dedicated to my mother

Declaration

I hereby declare that the matter embodied in the report entitled Function Field Sieve: Recent advances in the index-calculus attack on the discrete logarithm problem are the results of the investigations carried out by me at the Department of Mathematics, IISER Pune under the supervision of Dr. Ayan Mahalanobis and the same has not been submitted elsewhere for any other degree.

K Hariram

Acknowledgments

I would like to thank my supervisor supervisor. He has guided me throughout the project and helped me with my shortcomings. He also organised talk sessions among students in his group to discuss work done by each member. This helped my reinforce my understanding through feedback from other members.

Abstract

In this paper we look into the discrete log problem over finite fields. The relative hardness of this problem defines the integrity of many cryptographic systems. Therefore the ease of solvalbility of this problem has been important to cryptographer for a long time. We study the index calculus method. In prticular we focus on the function field sieve. This algorithm works well to find discrete log in the multiplicative group of finite fields $\mathbb{F}_{q^n}$ with a medium or small sized subfield $\mathbb{F}_q$. It has sub-exponential time complexity. We investigate various recent improvements done to this algorithm by Antoine Joux.

Contents

Chapter 1

Introduction

In this thesis we look into one of the methods used to solve the discrete log problem. Some cryptographic systems rely on the fact that this problem is relatively hard to solve.

For a finite group generated by a single element we can define the discrete log of any element. Let G be a group generated by an element g. Any element h in this group can be expressed as $h = g^x$. Then we can say $\log_g h = x$. For certain groups the calculation of the discrete log is very easy. For example in the group $(\mathbb{Z}/n\mathbb{Z}, +)$ generated by the element 1, any element h has discrete log also as h. For other groups this can be much harder. Such groups include multiplicative groups of finite fields and the group of points on an elliptic curve over finite fields. For discrete log problem in multiplicative groups of finite fields there is a subexponential algorithm. However no such algorithm is known for the discrete log problem in elliptic curves.

Some early cryptographic systems which use the hardness of the discrete log problem are the Diffie-Hellman key exchange protocol and the ElGamal cryptosystem [4].

ElGamal cryptosystem Suppose Bob wants to transmit a message to Alice. Alice publically announces an element $A = g^a$ and the group G generated by g. She keeps a as secret. The value of A is called the public key for Alice. Under this cryptosystem anyone who knows A can safely send a message to Alice with out the fear of it being intercepted. Let us see how Bob can send his meesage. His message is an element of

the group G say m called the plain text. He now creates the cipher text as mA^b and g^b and announces them. The value of b is chosen randomly and is not announced at all. Then Alice can calculate $A^b = (g^b)^a$ as she know a. She can then proceed to find plain text m by multiplying mA^b with A^{-b}.

The above cryptosystem is an example of a public key cryptosystem. In such a system all communications can be done over public channels. This is better than priveate key cryptosystems which require sharing of cryptographic schemes or keys privately which may not be be always feasible. Instead the public key cryptosystems maintains the integrity of the messages by hardness of solving a problem. In the ElGamal cryptosystem, if anyone else was able to find $a = \log_g A$, then they can also read the message.

A brute force method to solve the discrete log problem would find all powers of the generator g till we are able to find the one equal to given element. The average search time for such an attack would be of the order of size of the group. There are several generic agorithm like Pollard-ρ method and Pohlig-Hellman. These algorithms take time of the order of $\sqrt{|G|}$. However in case of multiplicative groups of finite fields we have the index calculus method which can solve it much quicker.

Chapter 2

How does the function field sieve work

2.1 Index calculus method

The index calculus method is a sub exponential time algorithm to solve the discrete log problem over finite fields. The ideas for this method were formulated by Kraitchik [8] even before the advent of computing and cryptography. This method can be broken down into the following steps:

1. Choose a subset B of the given field. This set is called the smoothness basis. Now we try to create multiplicative relationships among the elements of B and with elements whose discrete log is known. We will call this **the sieving step**.

2. When these multiplicative relationships are rewritten in terms of log of elements from B, they become linear equations. If we have as many independent linear equations as the number of elements then we can solve them to get value of log of each element in B. We will call this **the linear algebra step**.

3. Now we want to find the log of a given element. We try to establish a multiplicative relationship between that given element and elements from B. Using this relation we can calculate the log of the given element. We will call this **the individual log step**.

I will now illustrate all these steps on the group $\mathbb{Z}/p\mathbb{Z}^*$ with a simple method for each of the step. We want to find x such that $g^x \equiv h \mod p$. In the sieving step, for

some $k > 0$ we choose our smoothness basis as

$$B = \{2 \leq P \leq k | \ P \text{ is prime in } \mathbb{Z}\}.$$

Here k is called the smoothness bound. Next we take elements of the form $A_i \equiv g^i \mod p$ that have all its prime factors in the set B. Such numbers are called B-smooth numbers. For each smooth number we get a multiplicative relationship of the form

$$A_i = 2^{a_{2,i}} 3^{a_{3,i}} \ldots P^{a_{P,i}} \ldots .$$

This search for smooth numbers can be done by trial division of each candidate with primes from B.

The above relationship can be rewritten as (with respect to log base g)

$$i = a_{2,i} \log(2) + a_{3,i} \log(3) + \ldots + a_{P,i} \log(P) + \ldots .$$

This is a linear equation with variables as $\log(P), P \in B$. If we have as many linealy independent equations as variables, then the set of linear equations can be solved using Gaussian elimination to get value of each individual $\log(Q)$.

Now to find the discrete log of an element h, we check for an element of the form hg^u which is B-smooth. Once we get one such value of u for which this happens, we have $hg^u = 2^{b_2} 3^{b_3} \ldots P^{b_P} \ldots$, we get

$$\log(h) = u + b_2 \log(2) + b_3 \log(3) + \ldots + b_P \log(P) + \ldots$$

The algorithm described above still has some considerations yet to be discussed. First we need to choose an optimum value for the smoothness bound k. If we select a small value for it, we may not find enough smooth values of A_i. On the other hand choosing too large value will encumber the linear algebra step as we need to find as many relations to solve the system of equations. This will also gretly increase the time taken for this step. Next we can improve the sieveing step by performing a sieving instead of trial division. Finally in the linear algebra step one can observe that the matrix of co-efficients for the linear equations has mostly zeroes. Such a matrix is called sparse matrix. Thus this step can be performed much faster by sparse matrix elimination [10] rather than simple Gaussian elimination.

2.2 The function field sieve

The index calculus method was improved for the charecterstic 2 case in the Coppersmith algorithm [3]. Later Adleman improved this algorithm for the general charecterstic into what is now called the function field sieve[1].

The function field sieve works well when we are working with the multiplicative group of a field which has a medium or small sub-field. Thus we are trying to find the discrete log of any element in the field $\mathbb{F}_{q^n}$. This has a subfield $\mathbb{F}_q$. Note that this subfield itself may or may not be a prime field. In order to represent elements in this field we shall use a defining polynomial. If $F(x)$ is an irreducible polynomial of degree n with co-efficients in $\mathbb{F}_q$ then $\mathbb{F}_q[x]/F(x) \cong \mathbb{F}_{q^n}$.

We first construct two bi-variate polynomials with co-efficients in $\mathbb{F}_q$.

$$f_1(x, y) = x - g_1(y), f_2(x, y) = y + g_2(x)$$

Let the degree of these polynomials be d_1 and d_2 respectively. Later we shall look into how to optimally select these degrees. We want to select the two polynomials such that $y+g_2(g_1(y))$ has an irreducible factor of degree exactly n. Let this factor be $F(y)$. We shall use this as our defining polynomial for $\mathbb{F}_{q^n}$ over $\mathbb{F}_q$. Let α be a root of multiplicity 1 of the polynomial $F(y)$. This allows us to represent elements in $\mathbb{F}_{q^n}$ as polynomials in α of degree at most $n-1$. Set $\beta = g_1(\alpha)$. Then we can see that the pair (β, α) satisfies both the polynomials f_1 and f_2. Note that we are currently assuming that we will be able to find the polynomials $g_1(y)$ and $g_2(x)$ satisfying the desired conditions. We won't be discussing how to find them (by brute force or otherwise) and whetherthey exist at all.

The sieving step Our elements are from the field $\mathbb{F}_q[y]/F(y)$. They can be represented in the form $a_0 + a_1\alpha + a_2\alpha^2 + \ldots + a_{n-1}\alpha^{n-1}$. We shall let the smoothness basis contain all elements which have degree in either α or β less than or equal to a parameter D. The algorithm is most simple when $D = 1$. Next we shall sieve elements of the form $A(\alpha)\beta + B(\alpha)$ where A and B are polynomials of degree at most D. Using f_1 and f_2 we get that

$$A(\alpha)g_1(\alpha) + B(\alpha) = A(g_2(\beta))\beta + B(g_2(\beta)). \tag{2.1}$$

We shall call one such equation for a particular value of A and B as a relation. Here the left hand side has degree $d_1 + D$ and the right hand side has degree $d_2D + 1$. We would like to sieve both the sides into terms of degree at most D. The probability of a element to be smooth decreases with increase in its degree. Hence the minimum possible degree for both the sides is attained when $d_1 = \sqrt{nD}$ and $d_2 = \sqrt{n/D}$. For the $D = 1$ case, this requirement becomes $d_1 = d_2$ or $d_1 + 1 = d_2$ and d_1d_2 is slightly greater than or equal to n.

When we are completely able to factorise both sides for a relation, it can be written as

$$\prod C_i(\alpha) = E \prod D_j(\beta).$$

Here $C_i(\alpha)$ and $D_j(\beta)$ are elements from the smoothness basis. We can take log on this relation to get a linear relation between the log of elements of the smoothness basis. If we take the log base as α (or any other element from the smoothness basis) the system of equations in non-homogeneous. From here we need to just continue with the other two steps of the index calculus. We just need to find a relationship between α and g as well in the end as we would be knowing only log_α of elements.

2.3 Complexity of the sieving step

The main advantage we gain in the function field sieve is that we are sieving polynomials of degree around $\sqrt{n}$. In the simpler index calculus method discussed in the previous section we would be trying to split a arbitrary polynomial into linear (or degree D) terms. An arbitrary polynomial is usually of degree $n - 1$. We ask what is the probability that a polynomial splits into linears. There are q^d polynomials of degree d over the field $\mathbb{F}_q$. On the other hand we can create a polynomial that splits completely by just selecting its d roots from $\mathbb{F}_q$. Since the order of selection doesn't matter, such polynomials are nearly $q^d/d!$ in number. Thus in the basic index calculus we would need to go through about $(n-1)!$ relations before we get one relation that works. In function field sieve however, this becomes $(\lceil\sqrt{n}\rceil)!^2$ relations for each hit (The squaring arises because we need to simultaneously get smooth terms on both sides). Because the factorial function grows very rapidly, the function field sieve gives a significant saving.

We have $2q$ elements in the smoothness basis. This is also the number of smooth relations we require to perform the linear algebra step. Hence we need to go through $C = 2q(\lceil\sqrt{n}\rceil)!^2$ relations to complete the sieving step when $D = 1$. We will have q^3 choices for the polynomials A and B. This will be the number of available relations. Thus we can sieve only when $q^3 > C$ i.e. $q^2 > 2(\lceil\sqrt{n}\rceil)!^2$. To calculate the asymptotic complexity of this algorithm, we assume that for a given finite field $\mathbb{F}_Q = \mathbb{F}_{p^k}$, We are freely able to choose q and n such that $Q = q^n$. Under all these constraints we would like to find the minimum value of C. Using approximation for the factorial for $\log C$, we get

$$\log C = \log(2q) + 2\sqrt{n}\log(\sqrt{n}).$$

Minimizing the above equation under the constraint $\log Q = n \log q$, we get

$$n = (\log Q)^{2/3}(\log\log Q)^{-2/3}$$

$$\log q = (\log Q)^{1/3}(\log\log Q)^{2/3}.$$

This yields us

$$\begin{aligned}\log C &= \log 2 + (\log Q)^{1/3}(\log\log Q)^{2/3} \\ &\quad +(\log Q)^{1/3}(\log\log Q)^{-1/3}\log((\log Q)^{2/3}(\log\log Q)^{-2/3}) \\ &= \log 2 + (\log Q)^{1/3}(\log\log Q)^{2/3}(1 + (\log\log Q)^{-1/3}) + \text{ smaller terms.}\end{aligned}$$

Giving us the asymptotic complexity of

$$C = e^{(2+o(1))(\log Q)^{1/3}(\log\log Q)^{2/3}}$$

$$= L_Q(1/3, 2)$$

Chapter 3

Recent advances in the function field sieve

Now we shall look into some improvements done by Antoine Joux in [7] and [5] to the function field sieve. A prominent one is called pinpointing. This helps in efficiently finding smooth relationships by allowing us to check for smoothness of only one side of the equation at a time. We are also encouraged to look at these improvements from the point of view of the number field sieve as well. Any corresponding improvement in this other sieve can be of great benefit as it has use in integer factorization as well.

3.1 Pinpointing

From this section onwards we shall only focus on the simple $D = 1$ case of the function field sieve. We will use the same setup as described in the previous chapter. However we shall try to cleverly choose the the function g_1. We just set it as $g_1(y) = y^{d_1}$. Hence now we need to find an appropriate $g_2(x)$ such that $y + g_2(y^{d_1})$ has irreducible factor of degree n. Note that with this restriction it may not always be possible to do so.

Next we can expand $A(y) = y + a$ and $B(y) = by + c$. Then we can write equation 2.1 in the form

$$\alpha^{d_1+1} + a\alpha^{d_1} + b\alpha + c = (g_2(\beta) + a)\beta + bg_2(\beta) + c$$

We modify the LHS by setting $U = \alpha/a, s = ba^{-d_1}, t = ca^{-d_1-1}$. Then it becomes $U^{d_1+1} + U^{d_1} + sU + t$. Now we sieve over this space to find elements which factorize to linears in U. Then we substitute back α giving us a factorization into linears of a polynomial in α Note that this factorizability remains true for any value of a. However each value of a gives a different RHS. Hence by finding one smooth element in U we have found q relations which have the LHS smooth.

3.2 Two sided pinpointing on Kummer extensions

In the special case where we have an irreducible polynomial of the type $F(y) = y^n + \gamma$, such that all its root are distinct, it is possible to implement a two sided version of the pinpointing technique discussed above. The extension defined by such a polynomial is called a Kummer extension. In this setting we will choose the two polynomials as

$$g_1(y) = y^{d_1}$$

$$g_2(x) = x^{d_2}/\gamma$$

such that $n = d_1 d_2 + 1$. This gives $y + g_2(g_1(y)) = y + y^{n+1}/\gamma$. We get the desired irreducible polynomial once we divide this by y/γ. Now the relations look like

$$\alpha^{d_1+1} + a\alpha^{d_1} + b\alpha + c = \beta^{d_1+1}/\gamma + a\beta^{d_1}/\gamma + b\beta + c$$

This can be modified by setting $U = \alpha/a$, $V = \beta/a$, $s = ba^{-d_1}$, $t = ba^{-d_2}$, $w = c/ab$.

$$U^{d_1+1} + U^{d_1} + s(U + w) = (V^{d_2+1} + V^{d_2})/\gamma + t(V + w)$$

Now any such relation that is smooth on both sides gives us q different actual relations.

3.3 Using the Frobenius map

When the base field is not prime, we can reduce the size of our smoothness basis. Suppose we are working with $\mathbb{F}_{q^n}$ where $q = p^m$. Consider the action of n^{th} power of the Frobenius map $\phi^n := x \mapsto x^{p^n}$ on the smoothness basis. The element α is root of degree n irreducible polynomial. This means ϕ^n fixes α. Then for a general element from the smoothness basis, we have $\phi^n(\alpha+u) = \alpha+\phi^n(u)$. This is just another element

in the smoothness basis. However we get a direct relation $\log(u) = p^n \log(\phi^n(u))$. We can use this to reduce the number of variables for the linear algebra step. We can repeatedly apply this map to get a string of related elements in the smoothness basis. The size of this string can be as high as m when $(m, n) = 1$. Thus we can reduce the number of variables by an order of m in certain cases.

3.4 Homographies

Using change of variable as illustrated in the pinpointing technique, it is possible to get multiple relations by just sieving for smooth element once. We now consider another map called a homography which looks like:

$$X \mapsto \frac{aX + b}{cX + d}$$

In order to preserve the polynomials we modify it and define the following map:

$$H_{abcd} : \mathbb{F}_q[x] \to \mathbb{F}_q[x]$$

$$H_{abcd}(f(X)) = (cX + d)^{\deg f} f\left(\frac{aX + b}{cX + d}\right)$$

If $f(X)$ has the factorization into irreducibles as $f(y) = \prod_{i=1}^{k} f_i(X)$. We have a corresponding factorization for $H_{abcd}(f(X))$ as

$$H_{abcd}(f(X)) = \prod_{i=1}^{k} \left((cX + d)^{\deg f_i} f_i\left(\frac{aX + b}{cX + d}\right)\right)$$

However this may not be a factorization into irreducibles. Still this shows that degree of the irreducble factors for $H_{abcd}(f(X))$ will be smaller than the corresponding ones in $f(X)$. This implies that if $f(X)$ is smooth, then so is $H_{abcd}(f(X))$. When $D \geq 2$, this technique can be used to create several smooth relations from a single one.

3.5 A faster individual logarithm step

Finding the value of $log_\alpha(h)$ can be speeded up by the following method called the descent method. First we look at elements of the form $\alpha^u h^v$. These arbitrary terms

are usually of degree $n-1$. We would like to factor any one of these into irreducibles of degree at most $\sqrt{n}$. Once we are able to find one such value of $\alpha^u h^v$, let r be one of its factors. Now we are left with finding the log of a few terms whose degree is bounded. Now search elements of the form $A(\alpha)\beta + B(\alpha)$ such that r divides its representation as a polynomial in α i.e., r divides the LHS of equation 2.1

$$A(\alpha)g_1(\alpha) + B(\alpha) = A(g_2(\beta))\beta + B(g_2(\beta)).$$

Now we try to find one such relationship where the remaining LHS and the whole RHS splits into factors of degree strictly less than that of r (preferably into linears). We repeat the above step with all other non-linear factors till we are able to descend to a point where we are able to form a multiplicative relationship with only linear terms and our current r. Then we will know the log of that particular factor. After this we can traceback all factors till we find the log of $\alpha^u h^v$. From this we can get the value of $log_\alpha(h)$.

Bibliography

[1] L. M. Adleman. The function field sieve. In *Algorithmic Number Theory, Proceedings of the ANTS-I conference*, volume 877, pages 108–121, 1994.

[2] I. F. Blake, R. C. Mullin, and S. A. Vanstone. Computing logarithms in $GF(2^n)$. *Advances in Cryptology - CRYPTO'84, LNCS 196*, 1985.

[3] Don Coppersmith. Fast evaluation of logarithms in fields of characteristic two. *IEEE Transactions on Information Theory*, 30(4):587–594, 1984.

[4] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Advances in Cryptology*, pages 10–18. Springer, 1985.

[5] Antoine Joux. Faster index calculus for the medium prime case application to 1175-bit and 1425-bit finite fields. In *Advances in Cryptology–EUROCRYPT 2013*, pages 177–193. Springer, 2013.

[6] Antoine Joux. A new index calculus algorithm with complexity L(1/4 + o(1)) in small characteristic. In *Selected Areas in Cryptography–SAC 2013*, pages 355–379. Springer, 2014.

[7] Antoine Joux and Reynald Lercier. The function field sieve in the medium prime case. In *Advances in Cryptology-EUROCRYPT 2006*, pages 254–270. Springer, 2006.

[8] M. Kraitchik. *Théorie des nombres*. Gauthier–Villards, 1922.

[9] Oliver Schirokauer. The impact of the number field sieve on the discrete logarithm problem in finite fields. In *Proceedings of the 2002 Algorithmic Number Theory workshop at MSRI*, 2008.

[10] D. H. Wiedemann. Solving sparse linear equations over finite fields. *IEEE Transactions on Information Theory*, 32:54–62, 1986.