A Brief Survey of the Discrete Logarithm Problem

by

Gretel S. Sia

Submitted in Partial Fulfillment
of the Requirements for
the Degree of
Master of Arts in Mathematics

University of Hawaii at Manoa

August 2011

Abstract

The Discrete Logarithm Problem (DLP) has been the subject of interest among many mathematicians and cryptographers in recent times because of its computational difficulty. For the former, the enormity of the mathematics involved and the intellectual challenge that it entails are certainly motivating factors; for the latter, its usefulness in the field of cryptography. Cryptosystems rest their security on some assumptions that certain mathematical problems are difficult to solve. The ElGamal cryptosystem, for instance, is considered secure because of the computational assumption that it is difficult to solve the Discrete Logarithm Problem. The difficulty of the DLP lies in the fact that it has a "one-way" property. Its computational complexity is roughly measured by the computing time of the algorithm used to solve this mathematical problem. This paper is a brief survey of some of the best known algorithms for solving the DLP, examines their computing time, and considers the DLP over two particular finite groups.

Acknowledgement

It was my late father, Sebastian de Padua Salimbagat, who by insinuation or otherwise, had instilled in my young heart and mind, how happy and proud I could make him if someday I would become a teacher. For all that I have accomplished and have become, to him, my loving gratitude.

I thank the members of the Math Department graduate committee and my panelists, Dr. Pavel Ghuerzhoy and Dr. Claude Levesque, for giving me the privilege to present this paper.

The subject on cryptography was introduced to me by my adviser, Dr. Michelle Manes, during one of those directed reading courses I had with her, and in which this thesis came about as a natural conception. I am especially thankful for the many help she had extended to me and her patience in guiding me and helping me understand any sort of mathematical concept, from the simplest to the most abstract of ideas. Her great passion for math is itself an inspiration.

I am equally thankful to the many individuals who had made this part of my academic journey possible: Dr. Robert Little, for opening the door for me that I could do my graduate studies in math here in the United States; Dr. James B. Nation, for giving me all the chances to propel in my academic endeavor; Dr. Heiner Dovermann and Dr. Wayne Smith, for being there when help was mostly needed; Dr.Tom Craven and Dr. George Csordas, both with whom I did the most part of my basic graduate math courses and whose teaching styles worked well for me, something I can emulate; my former colleague and my friend, Chee Chen, who has always been there with his kindness and generosity; my other colleagues, Zachary Kent, Shashi Singh, Lukasz Grabarek, Michael Joyce, Matthew Chasse, and William De Meo, who in one way or another, had helped me along the way; my family, especially my son, Kyle Christopher Sia, whose constant faith in me is my source of strength; and Dr. Nilo G. Arnaiz, for paving the way for me, and in myriad ways, a pillar whereon I lean for support.

Above all, I am grateful to God, the Almighty, by whose power and presence, I seek to do only that which is good.

Table of Contents

Section Page

1. Introduction 5

2. General Attacks and Running Time 7

Shank's Baby-step Giant-step Algorithm............................... 8

Pollard's Rho Algorithm 10

3. The Index-Calculus Method 12

4. The Discrete Logarithm Problem Over Some Specific Groups 16

5. The Elliptic Curve Discrete Logarithm Problem.................... 23

Bibliography 31

1. Introduction

The Discrete Logarithm Problem (DLP) presents itself as a simple mathematical problem but there is a computational presumption that it is difficult. It is important because of its wide applications in the field of cryptography. Nevertheless, its study always bears great academic significance. Indeed, it has become the subject of interest among cryptographers and mathematicians in recent times because of its computational difficulty. Cryptosystems are considered secure under certain computational assumptions. For instance, the RSA scheme of Rivest, Shamir, and Adleman rests its security on the difficulty of the Factoring Problem [9]. Many others, such as ElGamal [3], are based on the assumption that the Discrete Logarithm Problem is difficult to compute for certain groups.

Definition 1.1 The Discrete Logarithm Problem states: " Given a multiplicative group G and elements g , $h \in G$, find an integer n, if it exists, such that $g^n = h$ ". This number n is the discrete logarithm of h to the base g, written more concisely as $n = \log_g(h)$.

In cryptographic applications, the existence of such an integer n is naturally presumed. Consequently, the problem is reduced to finding the number n. The word "discrete" is used to distinguish those situations involving finite groups, like the ones being dealt herein, from the classical (continuous) case.

In 1976, Whitfield Diffie and Martin Hellman published a paper in which they proposed the Discrete Logarithm Problem as a good source of a "one-way" function [2]. That marked the inception of the Discrete Logarithm Problem in cryptography. For the purpose of this study, we may think of a "one-way" function as a function $f : X \to Y$ for which given $x \in X$, it is easy to compute $f(x)$, however, given $y \in Y$, it is difficult to compute a value $x \in X$ such that $f(x) = y$, at least for most values of y [6]. In other words, from the standpoint of realistic computability, the function f is not invertible, without further information, and it is for this reason that such function is otherwise known as a "trapdoor" function.

Application

In a public key cryptosystem, anyone who has an enciphering key, and thus knows how to encipher, cannot make use of this enciphering key to find the deciphering key. In other words, any enciphering function is a "one-way" function.

The most notable example of a public key cryptosystem whose security rests on the assumption that discrete logarithms are hard to compute is the Diffie-Hellman Key Exchange Protocol. This is a method whereby two users, say Alice and Bob, hereinafter being referred to as A and B, respectively, exchange a cryptographic key (string of bits) over an insecure channel of communication. Under the traditional cryptosystems, in order for two parties to be able to communicate with each other, they needed to meet beforehand to agree upon a secret key, or were constrained to use a courier for the purpose. This was a great obstacle to the free and spontaneous flow of secure communications. The Diffie-Hellman key exhange algorithm solved this problem.

Under this cryptographic scheme, the key exhange between A and B is accomplished as follows:

(1) A and B agree on a finite group G and an element $g \in G$ of large order.

(2) A randomly chooses an exponent a, computes g^a and sends this value to B. She keeps the exponent a private.

(3) Similarly, B chooses an exponent b at random, computes g^b and sends this value to A. He also keeps the exponent b private.

(4) Using the value g^b received from B, A computes

$$K_a = (g^b)^a.$$

(5) Similarly, from the value g^a which he received from A, B computes

$$K_b = (g^a)^b.$$

From this fairly simple exchange of values by A and B, they now have created their secret key:

$$K_a = (g^b)^a = g^{ba} = g^{ab} = (g^a)^b = K_b,$$

and they can use this private key to communicate each other using any cryptographically safe communication protocol. We note that since the channel of communication being used by the parties is insecure, during this key exchange, the values g, g^a, and g^b are publicly visible. Thus, an efficient discrete logarithm algorithm would enable the cryptanalyst or any eavesdropper to determine a, and then eventually, determine the key used by A and B, making this scheme insecure. Diffie and Hellman have conjectured that breaking their scheme is equivalent in difficulty to computing discrete logarithms. This conjecture, however, remains unproven. Thus, although unlikely, there exists the possibility that one can find a way to compute g^{ab} by mere knowledge of g^a and g^b without computing either of the discrete logarithms a or b.

The Diffie-Hellman scheme has found a widespread use in practical cryptosystems, as for example in the optional security features of the NFS file system of the SunOs operating system (NFS and SUNOS are trademarks of Sun Microsystem, Inc.) [6].

In the sections that follow, we will deal with the most commonly known computational algorithms for solving the Discrete Logarithm Problem (DLP), examines the running time of each of them, and describes the DLP over two particular groups: $\mathbb{Z}/m_1\mathbb{Z} \times \mathbb{Z}/m_2\mathbb{Z} \times ... \times \mathbb{Z}/m_k\mathbb{Z}$ and over P_m, the group of all signed permutation matrices of order m.

2. General Attacks and Running Time

The computational complexity of the DLP is roughly measured by the running time of the algorithm used to solve this mathematical problem. Thus, one who analyzes a cryptographic algorithm is usually interested to know how much computing time and how much storage it requires, that is, how long it takes to solve the problem in terms of the size of the input. Typically, one measures the size of the input by its number of bits since that is how much storage it takes to record the input. An integer n satisfying $b^{k-1} \leq n < b^k$ can easily be shown to have k digits to the base b, and so a k-bit number, in particular, is approximately 2^k.

Definition 2.1. (Order Notation) Let $f(x)$ and $g(x)$ be functions of x taking values that are positive. We say that " f is big O of g " and write $f(x) = O(g(x))$ if there are positive constants c and C such that $f(x) \leq cg(x)$ for all $x \geq C$. In particular, we write $f(x) = O(1)$ if $f(x)$ is bounded for all $x \geq C$.

Proposition 2.2. If the limit $\lim\limits_{x\to\infty} \frac{f(x)}{g(x)}$ exists and is finite, then $f(x) = O(g(x))$.

(See [4] p. 76.)

Definition 2.3. Suppose that there is a constant $A \geq 0$ independent of the size of the input, such that if the input is $O(k)$ bits long, then it takes $O(k^A)$ steps to solve the problem. Then the problem is said to be solvable in *polynomial time.*

Polynomial time algorithms are considered to be fast algorithms.

If there is a constant $c > 0$ such that for inputs of size $O(k)$ bits, there is an algorithm to solve the problem in $O(e^{ck})$ steps, then the problem is solvable in *exponential time.*

Exponential time algorithms are considered to be slow algorithms.

Intermediate between polynomial time algorithms and exponential time algorithms are *subexponential time* algorithms. These have the property that for every $\epsilon > 0$ they solve the problem in $O_\epsilon(e^{\epsilon k})$ steps. This notation means that the constants c and C appearing in the definition of the order notation are allowed to depend on ϵ.

Proposition 2.4. (Trivial Bound for the DLP) Let G be a group and let $g \in G$ be an element of order N. Then the Discrete Logarithm Problem $g^x = h$ can be solved in $O(N)$ steps, where each step consists of multiplication by g.

Proof: We simply create a list of the values g^x for $x = 0, 1, 2, ..., N-1$, where each successive value is obtained by multiplying the previous value by g. If a solution to $g^x = h$ exists, then h will certainly appear in our list. ■

From the standpoint of an attacker of a cryptographic scheme, such mathematical equation always has a solution inasmuch as he knows the existence of a secret key that is actually being shared only by A and B, and for which he seeks to uncover by solving the underlying discrete logarithm problem. We may observe at this point that this brute-force-like method runs in exponential time since we measure the input by the number of bits and N is approximately 2^k, that is, exponential in k.

General Attacks

Algorithms that involve finding matching elements from within one or more lists are variably called either as collision algorithms, meet-in-the-middle algorithms, birthday paradox algorithms, and square-root algorithms. The last is so called because of the fact that the running time of a collision algorithm is generally a small multiple of the square root of the running time required by an exhaustive search [4]. When anyone of these algorithms is used to break a cryptosystem, the word “attack” instead of “algorithm” is used. Thus, meet-in-the-middle attacks, square-root attacks, etc., are familiar phrases in cryptography. This may involve searching a space of keys or plaintexts or ciphertexts, or for public key cryptosystems, they may be aimed at solving the underlying hard mathematical problems, like that of the DLP. On the other hand, an algorithm is called “generic” if it does not exploit any special properties of the objects to which it is applied.

The following is a discussion of two of the best known “generic attack” algorithms for solving the DLP. These are the Shanks' Baby-step Giant-step Algorithm and the Pollard's Rho Algorithm. They both work in the absence of any extra information concerning the group, that is, they work for arbitrary groups.

Shanks' Baby-step Giant-step Algorithm

Proposition 2.5 Let G be a group and let $g \in G$ be an element of order $N \geq 2$. The following algorithm solves the discrete logarithm problem $g^x = h$ in $O(\sqrt{N}\ \log N)$ steps.

(1) Let $n = 1 + \lfloor \sqrt{N}\ \rfloor$, where $\lfloor \sqrt{N} \rfloor$ denotes the greatest integer less than or equal to $\sqrt{N}$, so in particular, $n > \sqrt{N}$.

(2) Create two lists:

$$\begin{aligned} \textit{List } 1 \quad &: \quad e,\ g,\ g_2,\ g_3, ...,\ g_n. \\ \textit{List } 2 \quad &: \quad h,\ hg^{-n},\ hg^{-2n},\ hg^{-3n}, ...,\ hg^{-n^2}. \end{aligned}$$

(3) Find a match between the two lists, say $g^i = hg^{-jn}$.
(4) Then $x = i + jn$ is a solution to $g^x = h$.

Proof: In creating *List* 2, we start by computing the quantity $u = g^{-n}$, and then compile *List* 2 by computing $h, hu, hu^2, hu^3, ..., hu^n$. Thus, creating the two lists takes approximately $2n$ multiplications. Assuming that a match exists, we can find a match in a small multiple of $\log(n)$ steps using standard sorting and searching algorithms, so step (3) takes $O(\log n)$ steps. Since $\lim_{n\to\infty} \frac{2n+n\log n}{n\log n} = 1$, the total running time for the algorithm is $O(n\log n)$. Since $n \approx \sqrt{N}$, $n\log n = \sqrt{N}\ \log\sqrt{N} = \frac{1}{2}\sqrt{N}\log N$, and thus the total running time for the algorithm is $O(n\log n) = O(\sqrt{N}\log N)$.

In order to show that we can always find a match from *List* 1 and *List* 2, let x be the unknown solution to $g^x = h$, and write x as $x = nq + r$, where $0 \le r < n$. Since the order of g is N, we know that $1 \le x < N$, and so $q = \frac{x-r}{n} < \frac{N}{n} < n$ since $n > \sqrt{N}$. Thus, we can rewrite the equation $g^x = h$ as $g^r = hg^{-nq}$, where $0 \le r < n$, and $0 \le q < n$. Now, g^r belongs to *List* 1 and hg^{-nq} belongs to *List* 2, showing that *List* 1 and *List* 2 always have a common element. ■

Pomerance [7] notes that one of the ground rules on these algorithms to work is the assumption that we can label group elements in such a way that they can be sorted. For the Baby-step Giant-step algorithm, we need to sort the elements in the first list. Then sequentially run through the second list to check for membership in the first list. The sorting can be done in $O(n\log n)$ comparisons, and each membership check, via a binary search, can be done in $O(\log n)$ comparisons. A binary search involves identifying the midpoint of the sorted list, deciding if the searched-for element is in the first half or the second, and then iterating in the appropriate half. So in total we do $O(n\log n) = O(\sqrt{N}\log N)$ comparisons after the list.

Example:

We illustrate Shanks' Baby-step Giant-step Method to solve $5^x \equiv 3 \pmod{2017}$, that is, the discrete logarithm of 3 to the base 5 in $(\mathbb{Z}/2017\mathbb{Z})^*$. For this problem, $g = 5 + 2017\mathbb{Z}$, $h = 3 + 2017\mathbb{Z}$, $n = \lfloor\sqrt{2017}\rfloor + 1 = 45$, and $g^{-1} = 807 + 2017\mathbb{Z}$. The baby-steps ($g^k$) and the giant-steps ($hg^{-nk}$) are listed as follows:

k	$\mathbf{g}^k$	$\mathbf{hg}^{-nk}$	**k**	$\mathbf{g}^k$	$\mathbf{hg}^{-nk}$	**k**	$\mathbf{g}^k$	$\mathbf{hg}^{-nk}$
1	5	269	16	1121	414	31	10	1282
2	25	1261	17	1571	816	32	50	656
3	125	790	18	1804	556	33	250	1673
4	625	914	19	952	102	34	1250	82
5	1108	603	20	726	1078	35	199	1974
6	1506	1627	21	1613	517	36	995	1523
7	1479	1336	22	2014	639	37	941	751
8	1344	1464	23	2002	821	38	671	1451
9	669	167	24	1942	332	39	1338	346
10	1328	183	25	1642	859	40	639	1442
11	589	273	26	142	1050	41	1178	1556
12	928	275	27	710	1368	42	1856	1693
13	606	1799	28	1533	1644	43	1212	1203
14	1013	1295	29	1614	171	44	9	968
15	1031	477	30	2	1214	45	45	1411

So we have $5^{40} = 639 = 3 \cdot 5^{-22(45)}$, and therefore, $5^{40+22(45)} = 3$. The solution to the DLP is $x = 40 + 22(45) = 1030$.

Pollard's Rho Algorithm

This is another method that has a "square-root" complexity, but in contrast to Shanks' Baby-step Giant-step method, this has negligible space requirements. The down side is that the ρ-method requires the group order, while Shanks' algorithm will work even if n is only an upper bound for the order of the group element g. Again, we want to solve the discrete logarithm problem $g^x = h$.

The first step of the Pollard Rho Algorithm is to split the elements of the group into three disjoint subsets G_1, G_2, and G_3 of roughly equal size, so $G_1 \cap G_2 = G_2 \cap G_3 = G_1 \cap G_3 = \varnothing$ and $G_1 \cup G_2 \cup G_3 = G$. Then we define a function $f : G \to G$ by

$$f(\beta) = \begin{cases} g\beta & \text{if } \beta \in G_1, \\ \beta^2 & \text{if } \beta \in G_2, \\ h\beta & \text{if } \beta \in G_3. \end{cases}$$

Now, let $\beta_0 = g^{x_0} h^{y_0}$, where x_0 is a random element in the set $\{1, 2, ..., n\}$, so β_0 is a random element of G. Then we compute the sequence (β_i) by the recursion $\beta_{i+1} = f(\beta_i)$. Each element in the sequence can then be written as $\beta_i = g^{x_i}h^{y_i}$ with

$$x_{i+1} = \begin{cases} x_i + 1 \pmod{n} & \text{if } \beta_i \in G_1, \\ 2x_i \pmod{n} & \text{if } \beta_i \in G_2, \\ x_i \pmod{n} & \text{if } \beta_i \in G_3. \end{cases}$$

and

$$y_{i+1} = \begin{cases} y_i \pmod{n} & \text{if } \beta_i \in G_1, \\ 2y_i \pmod{n} & \text{if } \beta_i \in G_2, \\ y_i + 1 \pmod{n} & \text{if } \beta_i \in G_3. \end{cases}$$

Since G is a finite group, there will eventually be two elements in the sequence, β_i and β_{i+l} for some $i \geq 0$ and $l > 0$, such that $\beta_i = \beta_{i+l}$, and therefore, $g^{x_i}h^{y_i} = g^{x_{i+l}}h^{i+l}$. It follows that $g^{x_i - x_{i+l}} = h^{y_{i+l} - y_i}$, and since $h = g^x$, then $g^{x_1 - x_{i+l}} = g^{x(y_{i+l} - y_i)}$. This implies that $x_i - x_{i+l} \equiv x(y_{i+l} - y_i)(\bmod\, n)$.

Now, if $\gcd(y_{i+l} - y_i, n) = 1$, then the element $(y_{i+l} - y_i)$ is invertible, and solving for x using the last congruence gives $x \equiv (x_i - x_{i+l})(y_{i+l} - y_i)^{-1} \bmod n$. If, on the other hand, $(y_{i+l} - y_i)$ is not invertible, then we reduce the congruence by dividing it by the greatest common divisor d of all the three elements $(x_i - x_{i+l})$, $(y_{i+l} - y_i)$ and n, and then finding z such that $\frac{x_i - x_{i+l}}{d} \equiv \frac{z(y_{i+l} - y_i)}{d} \bmod(\frac{n}{d})$. This gives $x \equiv z + k(\frac{n}{d})$ where $0 \leq k < d$. We then test all possible values of x and choose that which solves the DLP. If there are exceedingly many values of x to test, we start the algorithm again using a different value for x_0.

Pollard's Rho Method to solve the Discrete Logarithm Problem was first introduced in 1978 [7]. It is called as such because the algorithm, in effect, produces a sequence of numbers such that if we present them graphically by connecting successive elements of the sequence by a line, the trail looks like the Greek letter, ρ. The rationale behind it and the reason why it works is based on the famous "birthday paradox" in statistics which states that if there are at least 23 (approximately $\sqrt{366}$) people in a room then the probability that some pair of them have the same birthday is more than 1/2. In more familiar terms, the paradox is otherwise stated that the probability of finding two numbers in a randomly chosen sequence that are congruent modulo p is greater than 1/2 once approximately $\sqrt{p}$ numbers have been chosen. More generally, everytime elements are randomly selected from a set, say of size n, we only need to select $O(\sqrt{n})$ of them in order to have more than 50% chance of selecting the same element twice. Indeed, this is the fundamental idea behind any collision algorithm [4]. Applying this to the Pollard Rho Algorithm tells us that the probability of finding a match, $\beta_i = \beta_{i+l}$, in the sequence (β_i) is greater than 1/2 once we have $O(\sqrt{|G|})$ group elements in it. To have that many elements of the sequence corresponds to having gone once around the loop of the ρ, which is of period l. This implies that $(i+l)$ is $O(\sqrt{|G|})$ and that the method requires $O(\sqrt{N})$ steps to solve the DLP, where $N = |G|$.

Example:

Using the Pollard Rho Algorithm, we again solve the congruence, $5^x \equiv 3 \pmod{2017}$. We let all the residue classes to be represented by their smallest nonnegative representatives and set

$$\begin{aligned} G_1 &= \{1, ..., 672\} \\ G_2 &= \{673, ..., 1344\} \\ G_3 &= \{1345, ..., 2016\}. \end{aligned}$$

As a starting value, we use $x_0 = 1037$. A few of the stored triples and the final triple which is a match are given below:

j	$\boldsymbol{\beta}_j$	$\mathbf{x}_j$	$\mathbf{y}_j$
0	1209	1037	1
1	1373	58	2
2	85	58	3
4	108	60	3
8	793	125	6
16	1366	1008	53
32	1580	31	453
45	704	1024	442
79	704	860	816

So we have $\beta_{45} = 5^{1024}3^{442} = 704 = 5^{860}3^{816}$, which gives $5^{164} = 3^{374} = 5^{x(374)}$. We then have to solve the congruence $374x \equiv 164 \pmod{2016}$ which can be reduced to $187x \equiv 82 \bmod 1008$. The solution is $x = 22 + k(1008)$, $0 \le k < 2$. We find $k = 1$, giving us the discrete logarithm $x = 1030$.

3. The Index Calculus Method

For a given group, an algorithm may exist that takes advantage of some special arithmetic properties of the group. This algorithm is no longer generic since it is not applicable to any group with a different structure. The index calculus method is one such method. For a given prime p, it is used to solve the discrete logarithm problem in a finite field $\mathcal{F}_p$. It makes use of the fact that it has certain elements that can be labeled as "smooth" and can therefore be written as a product of other elements from some relatively small factor base.

Definition 3.1 Let B be a given natural number. An integer n is called B-smooth if all of its prime factors are less than or equal to B.

Definition 3.2 Let p be a prime number. A *primitive root modulo* p is a natural number $A < p$ such that for every natural number B relatively prime to p there exists some $e \in \mathbb{Z}$ such that $A^e \equiv B \bmod p$.

We now consider the discrete logarithm problem over a finite field $\mathcal{F}_p$. The DLP is to solve $g^x \equiv h \pmod{p}$, where the prime p and the integers g and h are given. We further assume that g is a primitive root modulo p, so that it generates all of $\mathcal{F}_p^*$. The first step is to choose B and solve the DLP $g^x \equiv l \pmod{p}$ for all primes $l \leq B$. Then we look at the quantities $hg^{-k} (\bmod\, p)$ for $k = 1, 2, 3, \ldots$ until we obtain a value k such that $hg^{-k} (\bmod\, p)$ is B-smooth. For this particular value of k, we have $hg^{-k} = \prod_{l \leq B} l^{e_l} (\bmod\, p)$ for certain exponents e_l. Writing this in terms of the discrete logarithms, we have

$$\log_g(h) = k + \sum_{l \leq B} e_l (\log_g(l)) (\bmod\, p-1). \qquad (*)$$

Since we have already computed $\log_g(l)$ for every prime $l \leq B$, this gives the value $\log_g(h)$, and this solves the DLP.

Solving $\log_g(l)$ for all primes $l \leq B$ works as follows: For a randomly chosen exponent i, we compute $g_i = g^i (\bmod\, p)$ with $0 < g_i < p$. If g_i is B-smooth, we write g_i as $g_i = \prod_{l \leq B} l^{u_l(i)}$, otherwise, we discard it. For these values of i in which g_i is B-smooth, we then express g_i as a product of powers of its prime factors and obtain

$$\begin{aligned} i &= \log_g(g_i) \pmod{p-1} \\ &= \sum_{l \leq B} u_l(i) (\log_g(l)) \pmod{p-1}. \qquad (**) \end{aligned}$$

The unknowns in this equation are the discrete logarithm values, $\log_g(l)$, for $l \leq B$. Thus, in order to solve these $\log_g(l)$ "variables", we must have at least $\pi(B)$ equations like $(**)$, where $\pi(B)$ is the number of primes less than or equal to B, and use linear algebra to solve for the values of these "variables".

Example: Suppose we want to solve $2^x \equiv 13 \pmod{2027}$. We first choose a bound B, say $B = 11$. The factor base is the set of primes $\{2, 3, 5, 7, 11\}$ and $\pi(B) = 5$. We then take random powers of $g = 2 \pmod{2027}$ and pick out the ones that are B-smooth. We obtain

$$\begin{aligned}
3^2(7) &= 63 \equiv 2^{293} \pmod{2027},\\
5(7)(11) &= 385 \equiv 2^{983} \pmod{2027},\\
2^7(11) &= 1408 \equiv 2^{1318} \pmod{2027},\\
3(11) &= 33 \equiv 2^{1593} \pmod{2027},\\
2^6(5^2) &= 1600 \equiv 2^{1918} \pmod{2027}.
\end{aligned}$$

This gives relations for the discrete logarithms of $2, 3, 5, 7$, and 11. For convenience, we let

$$\begin{aligned}
x_2 &= \log_g(2),\\
x_3 &= \log_g(3),\\
x_5 &= \log_g(5),\\
x_7 &= \log_g(7),\\
x_{11} &= \log_g(11),
\end{aligned}$$

and by substitution, the above system of congruences becomes

$$\begin{aligned}
293 &\equiv 2x_3 + x_7 \pmod{2026},\\
983 &\equiv x_5 + x_7 + x_{11} \pmod{2026},\\
1318 &\equiv 7x_2 + x_{11} \pmod{2026},\\
1593 &\equiv x_3 + x_{11} \pmod{2026},\\
1918 &\equiv 6x_2 + 2x_5 \pmod{2026}.
\end{aligned}$$

Now, $2026 = 2(1013)$ and 1013 is prime, so we can solve the system $\pmod 2$ and $\pmod{1013}$. We obtain

$$\begin{aligned}
x_3 + x_{11} &\equiv 1 \pmod 2,\\
x_5 + x_7 + x_{11} &\equiv 1 \pmod 2,\\
x_2 + x_{11} &\equiv 0 \pmod 2,\\
x_7 &\equiv 1 \pmod 2.
\end{aligned}$$

Clearly, $x_2 = 1$ since $g = 2$. Consequently, we get $x_2 \equiv x_5 \equiv x_7 \equiv x_{11} \equiv 1 \pmod 2$ and $x_3 \equiv 0 \pmod 2$. Similarly, we compute the discrete logarithms of the same factor base elements $\pmod{1013}$. Again, since $x_2 = 1$, we obtain

$$\begin{aligned}
x_3 + x_{11} &\equiv 580 \pmod{1013},\\
x_5 + x_7 + x_{11} &\equiv 983 \pmod{1013},\\
x_{11} &\equiv 298 \pmod{1013},\\
2x_3 + x_7 &\equiv 293 \pmod{1013},\\
2x_5 &\equiv 899 \pmod{1013}.
\end{aligned}$$

We easily find $x_{11} \equiv 298 \pmod{1013}$, and $x_5 \equiv 956 \pmod{1013}$ since $2^{-1} \pmod{1013} \equiv 507 \pmod{1013}$. We next obtain $x_7 \equiv 742 \pmod{1013}$ and $x_3 \equiv 282$

(mod 1013). Combining this solution with that of the system (mod 2), we finally obtain

$$\begin{aligned} x_2 &= 1, \\ x_3 &= 282, \\ x_5 &= 1969, \\ x_7 &= 1755, \\ x_{11} &= 1311. \end{aligned}$$

Finally, we compute the value of $13 \cdot 2^{-k} (\bmod 2027)$ for random values of k until we find a value that is B-*smooth.* We find $2(5)(11) = 110 \equiv 13 \cdot 2^{1397} (\bmod 2027)$. From $(*)$, we get

$$\log_2 13 = (-1397 + 1 + 1969 + 1311)(\bmod 2026) = 1884,$$

and thus, we have solved the given DLP over $\mathcal{F}_p$, with $p = 2027$.

Definition 3.3 Let x, α, c be real numbers and let x be greater than the Euler constant $e = 2.718...$ We define $L_x[\alpha, c] = e^{c(\log x)^\alpha (\log\log x)^{1-\alpha}}$. Thus, if $\alpha = 0$, $L_x[0, c] = (\log x)^c$; if $\alpha = 1$, $L_x[1, c] = e^{c \log x}$.

(1) If an algorithm has running time $L_x[0, c]$, then it is a *polynomial* time algorithm. Its complexity is bounded by a polynomial in the size of the input. The algorithm is considered efficient, although its real efficiency depends on the degree c of the polynomial.

(2) If the algorithm has running time $L_x[1, c]$, then it is *exponential*. Its complexity is bounded by an exponential function in the length of the input. The algorithm is considered inefficient.

(3) If the algorithm has running time $L_x[\alpha, c]$ with $0 < \alpha < 1$, then it is *subexponential*. The algorithm is slower than polynomial but faster than exponential.

Proposition 3.4 Let N be a large number, and $B = L_N[1/2, 1/\sqrt{2}]$. Then we expect to check approximately $L_N[1/2, \sqrt{2}]$ random numbers modulo N in order to find $\pi(B)$ numbers that are B-*smooth.* (See [4], p.150.)

The running time of the index calculus method can be roughly estimated as follows: Using a factor base consisting of primes less than B, we need to find approximately $\pi(B)$ numbers of the form $g^i (\bmod p)$ that are B-*smooth*. Proposition 3.4 above suggests that we take $B = L_p[1/2, 1/\sqrt{2}]$, and then we will have to check approximately $L_p[1/2, \sqrt{2}]$ values of i. Choosing an optimal value of B, and using Lenstra's elliptic curve factoring method to recognize B-*smooth* integers, the expected complexity of the method is $\exp(\sqrt{2}+o(1))\sqrt{\log p \log\log p}$, where $o(1)$ denotes a function that tends to 0 as $p \to \infty$. Moreover, once the initial work is done to find the discrete logarithms of the small primes, the additional time to find the discrete logarithms of a given group element takes only about $((1/\sqrt{2}+o(1))\sqrt{\log p \log\log p}$ [4]. This shows that the index calculus is a subexponential algorithm for solving the discrete logarithm problem in $\mathcal{F}_p^*$.

4. The Discrete Logarithm Problem Over Some Specific Groups

In this section, we consider the Discrete Logarithm Problem over the group, $G = (\mathbb{Z}/m_1\mathbb{Z}) \times (\mathbb{Z}/m_2\mathbb{Z}) \times ... \times (\mathbb{Z}/m_k\mathbb{Z})$, where $m_i \in \mathbb{Z}^+$, $i = 1, 2, ..., k$, and over the group, P_m of all signed permutation matrices of order m.

The DLP over $\mathbf{G} = (\mathbb{Z}/\mathbf{m}_1\mathbb{Z}) \times (\mathbb{Z}/\mathbf{m}_2\mathbb{Z}) \times ... \times (\mathbb{Z}/\mathbf{m}_k\mathbb{Z})$

Let $g, h \in G = (\mathbb{Z}/m_1\mathbb{Z}) \times (\mathbb{Z}/m_2\mathbb{Z}) \times ... \times (\mathbb{Z}/m_k\mathbb{Z})$, where $g = (g_1, g_2, ..., g_k)$ and $h = (h_1, h_2, ..., h_k)$, $g_i, h_i \in \mathbb{Z}/m_i\mathbb{Z}$ for every i. The discrete logarithm problem (DLP) in this case, consists in computing, if it exists, an integer n such that $h = ng$. This involves solving a system of k linear congruences modulo m_i, $i = 1, 2, ..., k$, which may or may not have a solution. As a problem in cryptography, however, such an integer n is guaranteed to exist so that the problem is reduced to that of finding n.

The idea behind the Algorithm for solving for the integer n relies largely on the *Chinese Remainder Theorem.*

Theorem 4.1 (The Chinese Remainder Theorem) Let $m_1, m_2, m_3, ..., m_r$ be pairwise relatively prime positive integers. Then the system of congruences

$$\begin{cases} x \equiv a_1 \pmod{m_1}, \\ x \equiv a_2 \pmod{m_2}, \\ . \quad . \\ . \quad . \\ . \quad . \\ x \equiv a_2 \pmod{m_2}, \end{cases}$$

has a unique solution modulo $M = m_1 m_2 \cdots m_r$.

Theorem 4.2 The system of linear congruences

$$\begin{cases} x \equiv c_1 \ (\text{mod } \ m_1), \\ x \equiv c_2 \ (\text{mod } \ m_2), \\ . \quad\quad . \\ . \quad\quad . \\ . \quad\quad . \\ x \equiv c_r \ (\text{mod } \ m_r), \end{cases}$$

has a solution if and only if or all $i \neq j$, $c_i \equiv c_j$ (mod(m_i, m_j)), where (m_i, m_j) denotes the greatest common divisor of the integers m_i and m_j. The solution, if it exists, is unique mod$[m_1, m_2, ..., m_r]$, where $[m_1, m_2, ..., m_r]$ denotes the least common multiple (LCM) of the integers $m_1, m_2, ..., m_r$.

Proof of Theorem 4.2

We consider the first case when $r = 2$. If the system of congruences has a solution $x = x_0$, then we can write

$$x_0 \equiv c_1 \ (\text{ mod } \ m_1) \text{ and } x_0 \equiv c_2 \ (\text{mod } \ m_2)$$

which implies

$$x_0 = k_1 m_1 + c_1 \text{ and } x_0 = k_2 m_2 + c_2$$

for some integers k_1, k_2. Then

$$c_1 - c_2 = k_2 m_2 - k_1 m_1 = (m_1, m_2)(k_2 \frac{m_2}{(m_1, m_2)} - k_1 \frac{m_1}{(m_1, m_2)})$$

and therefore,

$$c_1 \equiv c_2 \,\text{mod}(m_1, m_2).$$

Now suppose that $c_1 \equiv c_2 \,\text{mod}(m_1, m_2)$. This means $(m_1, m_2)|(c_1, c_2)$. Then $c_1 - c_2$ can be written as a linear combination of m_1 and m_2, say $c_1 - c_2 = k_2 m_2 - k_1 m_1$. Then

$$c_1 + k_1 m_1 = c_2 + k_2 m_2.$$

Let $x_0 = c_1 + k_1 m_1$. Then x_0 is a solution to $x \equiv c_1$ (mod m_1) and

$$x_0 = c_1 + k_1 m_1 = c_2 + k_2 m_2 \equiv c_2 (\text{mod}\, m_2).$$

Note that if $x \equiv c_1 (\text{mod}\, m_1)$ then

$$x \equiv c_1 + k_1 m_1 = c_1 + k_1 (m_1, m_2)(\frac{m_1}{(m_1, m_2)}),$$

and thus

$$x \equiv c_1 (\text{mod}\, \frac{m_1}{(m_1, m_2)}).$$

Hence, any solution to $x \equiv c_1 (\text{mod}\, m_1)$ and $x \equiv c_2 (\text{mod}\, m_2)$ is also a solution to $x \equiv c_1 (\frac{m_1}{(m_1,m_2)})$ and $x \equiv c_2$ (mod m_2). By the Chinese Remainder Theorem, there can be at most one solution mod $\frac{m_1}{(m_1,m_2)} m_2 = [m_1, m_2]$.

For $r = 2$, we apply mathematical induction on r. Suppose that the first r congruences has a unique solution A (mod M), where $M = [m_1, m_2, ..., m_r]$. Then by the first part of the proof, the system $x \equiv A$ (mod M), $x \equiv a_{r+1}$(mod m_{r+1}) has a unique solution mod$([M, m_{r+1}]) =$ mod$([m_1, m_2, ..., m_{r+1}])$.

An Algorithm for Solving the DLP

Given $g = (g_1, g_2, ..., g_k)$ and $h = (h_1, h_2, ..., h_k) \in G = (\mathbb{Z}/m_1\mathbb{Z}) \times (\mathbb{Z}/m_2\mathbb{Z}) \times ... \times (\mathbb{Z}/m_k\mathbb{Z})$, we want to find an integer n such that $h = ng$. We consider the following cases:

Case 1. $g^{-1} = (h'_1, h'_2, ..., h'_k)$ exists.

If $g^{-1} \in G$ then $n = (ng)g^{-1} = hg^{-1} = (n_1, n_2, ..., n_k) \in G$. This gives the following system of congruences:

$$\begin{cases} n \equiv n_1 \ (\text{mod} \ m_1), \\ n \equiv n_2 \ (\text{mod} \ m_2), \\ \quad . \\ \quad . \\ \quad . \\ n \equiv n_k (\text{mod}\, m_k). \end{cases}$$

Note that g^{-1}exists if $\forall i = 1, 2, 3, ..., k, \ \ (g_i, m_i) = 1$. If $(m_i, m_j) = 1$ for $i \neq j$, then by the *Chinese Remainder Theorem*, the system has a unique solution mod $M = m_1 m_2 \cdots m_k$. If, on the other hand, $(m_i, m_j) = d > 1$ for some $i \neq j$, then by *Theorem 4.2,* a unique solution mod $[m_1, m_2, ..., m_k \ ]$ exists.

Examples:

(1) Let $G = (\mathbb{Z}/6\mathbb{Z}) \times (\mathbb{Z}/29\mathbb{Z})$, $g = (5, 11)$ and $h = (0, 11)$. We want to find $n \in \mathbb{Z}^+$ such that $h = ng$.

Solution: We have $g^{-1} = (5, 8)$ and $n = (ng)g^{-1} = hg^{-1} = (0, 11)(5, 8) = (0, 1)$. Thus, $n \equiv 0 \pmod 6$ and $n \equiv 1 \pmod{29}$. Note that $(6, 29) = 1$ and we can easily infer that $n = 30 \pmod{174}$.

(2) Let $G = (\mathbb{Z}/9\mathbb{Z}) \times (\mathbb{Z}/10\mathbb{Z}) \times (\mathbb{Z}/12\mathbb{Z})$, $g = (5, 7, 5)$, and $h = (6, 1, 9)$. We want to find $n \in \mathbb{Z}^+$ such that $h = ng$.

Solution: We have $g^{-1} = (2, 3, 5)$ and $n = (ng)g^{-1} = hg^{-1} = (6, 1, 9)(2, 3, 5) = (3, 3, 9)$. This gives rise to the following system of congruences:

$$\begin{cases} n \equiv 3 \pmod{9}, \\ n \equiv 3 \pmod{10}, \\ n \equiv 9 \pmod{12}, \end{cases}$$

which is equivalent to the system:

$$\begin{cases} n \equiv 3 \pmod{9}, \\ n \equiv 1 \pmod{2}, \ n \equiv 3 \pmod{5}, \\ n \equiv 1 \pmod{4}, \ n \equiv 0 \pmod{3}. \end{cases}$$

We can reduce the system to give us

$$\begin{cases} n \equiv 3 \pmod{9}, \\ n \equiv 3 \pmod{5}, \\ n \equiv 1 \pmod{4}. \end{cases}$$

Applying the method used in the proof of the Chinese Remainder Theorem, we find that $M = 9(5)(4) = 180$. It follows that:

$$\begin{cases} M_1 = 20 \text{ and } y_1(20) \equiv 1 \pmod{9}, \text{ and so } y_1 = 5; \\ M_2 = 36 \text{ and } y_2(36) \equiv 1 \pmod{5}, \text{ and so } y_2 = 1; \\ M_3 = 45 \text{ and } y_3(45) \equiv 1 \pmod{4}, \text{ and so } y_3 = 1. \end{cases}$$

Thus, $n = 3(5)(20) + 3(1)(36) + 1(1)(45) \equiv 93 \pmod{180}$.

Case 2. g^{-1} does not exist.

This gives rise to the following system of congruences:

$$\begin{cases} ng_1 \equiv h_1 \pmod{m_1}, \\ ng_2 \equiv h_2 \pmod{m_2}, \\ \quad\quad . \\ \quad\quad . \\ \quad\quad . \\ ng_k \equiv h_k \pmod{m_k}. \end{cases}$$

In this case, we divide each congruence through by (g_i, m_i) and then multiply by the inverse of the coefficient $\frac{g_i}{(g_i,m_i)}$ mod t_i, where $t_i = \frac{m_i}{(g_i,m_i)}$. The simplified form is the system

$$\begin{cases} n \equiv s_1 \pmod{t_1}, \\ n \equiv s_2 \pmod{t_2}, \\ \quad\quad . \\ \quad\quad . \\ \quad\quad . \\ n \equiv s_k \pmod{t_k}, \end{cases}$$

which can be solved using the method in Case 1.

Example:

Let $G = (\mathbb{Z}/6\mathbb{Z}) \times (\mathbb{Z}/9\mathbb{Z}) \times (\mathbb{Z}/10\mathbb{Z})$, $g = (5, 6, 3)$, and $h = (1, 3, 9)$. We want to find n such that $h = ng$.

Solution: Note that since $(9, 6) = 3 > 1$, g^{-1} does not exist. This gives the following system of congruences:

$$\begin{cases} 5n \equiv 1 \pmod{6}, \\ 6n \equiv 3 \pmod{9}, \\ 3n \equiv 9 \pmod{10}, \end{cases}$$

which can be reduced to the following system:

$$\begin{cases} n \equiv 5 \pmod{30}, \\ 2n \equiv 1 \pmod{3} \Longleftrightarrow n \equiv 2 \pmod{6}, \\ n \equiv 63 \pmod{70}. \end{cases}$$

Now,

$$\begin{cases} n \equiv 5 \pmod{30} \Longleftrightarrow n \equiv 1 \pmod{2}; n \equiv 2 \pmod{3}; n \equiv 0 \pmod{5}, \\ n \equiv 2 (\bmod 6) \Longleftrightarrow n \equiv 2 \pmod{3}; n \equiv 0 \pmod{2}, \\ n \equiv 63 (\bmod 70) \Longleftrightarrow n \equiv 1 \pmod{2}; n \equiv 3 \pmod{5}; n \equiv 0 \pmod{7}. \end{cases}$$

The resulting system is

$$\begin{cases} n \equiv 1 \pmod{2}, \\ n \equiv 2 \pmod{3}, \\ n \equiv 3 \pmod{5}. \end{cases}$$

By the Chinese Remainder Theorem, $M = (2)(30(5) = 30$. This gives

$$\begin{cases} M_1 = 15, \quad y_1(15) \equiv 1 \pmod{2}, \text{ and so } y_1 = 1; \\ M_2 = 10, \quad y_2(10) \equiv 1 \pmod{3}, \text{ and so } y_2 = 1; \\ M_3 = 6, \quad y_3(6) \equiv 1 \pmod{5}, \text{ and so } y_3 = 1. \end{cases}$$

Therefore, $n = 1(1)(15) + 2(1)(10) + 3(1)(6) = 23$ (mod 30).

The computing time for the above described algorithm can be roughly estimated as follows: The elements of $\mathbb{Z}/m\mathbb{Z}$ which have multiplicative inverses are those which are relatively prime to m, that is, all elements a such that $\gcd(a, m) = 1$. Moreover, if $\gcd(a, m) = 1$, we can use the Euclidean algorithm to find an element b such that $ab \equiv 1(\bmod m)$, and such an inverse can be found in $O(\log^3 m)$ bit operations [5]. In other words, inverting an element in $\mathbb{Z}/m\mathbb{Z}$ can be done in *polynomial* time in the length of m. Since the Euclidean algorithm involves adding and multiplying two elements in $\mathbb{Z}/m\mathbb{Z}$, we can fairly infer that either operation can be done in at most *quadratic* time (also *polynomial*) in the length of m. Thus, the computing time for this algorithm, which uses the Chinese Remainder Theorem, can only be $O(\log^n m)$ for some $n \in \mathbb{N}$. It is a *polynomial* time algorithm with a running time that is bounded by the size of the input, m. The group G is not, therefore, an ideal object over which to define the DLP if we are to protect any cryptosystem that relies its security on the computational difficulty of this problem.

The Discrete Logarithm Problem Over P_m

Definition 4.4 (Signed Permutation Matrices) Let e_j denote the column vestors of $\mathbb{R}^n$ whose jth entry is 1 and the remaining entries are each equal to 0. A *signed permutation matrix* is a square matrix of size m whose columns are of the form $(\pm e_{\sigma(1)}, \pm e_{\sigma(2)}, ..., \pm e_{\sigma(m)})$, where $\sigma \in \Sigma_m$ is a permutation. The set of all signed permutation matrices of size m forms a group P_m of order $2^m m!$ called the hyper-octahedral group.

Let $A, B \in P_m$. The discrete logarithm problem over P_m involves finding an integer n such that $B = A^n$. We recall some important concepts in linear algebra.

Definition 4.5 Let A be an $n \times n$ matrix over a field k. The characteristic polynomial of A denoted by $p_A(\lambda)$ is the polynomial defined by $p_A(\lambda) = \det(\lambda I - A)$, where I denotes the $n \times n$ identity matrix and the determinant is being taken in $k[\lambda]$, the ring of polynomials in λ over k.

The polynomial $p_A(\lambda)$ is monic and its degree is n. It encodes several important properties of the matrix, most notably its eigenvalues, its determinants, and its trace. Its constant coefficient is equal to $(-1)^n \det A$, and the coefficient of λ^{n-1} is equal to $-tr(A)$, the trace of the matrix A.

Theorem 4.6 (Cayley-Hamilton Theorem) Every matrix satisfies its own characteristic equation.

By its properties, the characteristic polynomial of an $m \times m$ signed permutation matrix A is of the form $\lambda^m - 1$, $\lambda^m + 1$, or a product of any of the factors of these two polynomials whose degree is n. Consequenty, by the Cayley-Hamilton Theorem, the element of P_m having the highest order is that whose characteristic polynomial is $\lambda^m + 1$, that is, whose order is $2m$.

Let $\{\lambda_1, \lambda_2, ..., \lambda_m\}$ and $\{\beta_1, \beta_2, ..., \beta_m\}$ be the sets of eigenvalues of A and B, respectively. Since $B = A^n$, $\beta_j = \lambda_i^n$ for some $i, j = 1, 2, 3, ..., m$. For any eigenvalue λ of A, define the order of λ, $|\lambda|$ to be the smalllest positive integer k, such that $\lambda^k = 1$. Then the order of A, $|A| = \text{lcm}(|\lambda_1|, |\lambda_2|, |\lambda|_3, ..., |\lambda|_m)$. Suppose $|A| = r$ and $|B| = k$, then $k \leq r$, and k divides r. Moreover, we have $B^k = A^{nk} = A^r = I_m$.This says nk is a multiple of r. Hence, to solve for n, we only have to check such powers n' of A for which $n'k = rd$ for some integer positive integer d, where $2 \leq n' \leq r - 1$. If $k = r$, all possible values of n' may need to be checked as the right value for n, while if $k < r$, fewer such exponentiations of A are needed to find n. In either case, granting we have determined all the eigenvalues of both A and B, this still requires multiples of m multiplications. Hence, the run-time is still $O(m) = O(2^k)$, i. e., exponential.

Example:

Let us consider P_3, and let $A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & -1 \\ 1 & 0 & 0 \end{bmatrix}$ and $B = \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 1 \\ -1 & 0 & 0 \end{bmatrix}$. We want to find n such that $B = A^n$.

Solution: The characteristic polynomial of A is $p_A(\lambda) = \lambda^3 + 1$, which implies the order of A is 6, while the characteristic polynomial of B is $p_B(\lambda) = \lambda^3 - 1$, implying its order is 3. We have $2 \leq n' \leq 5$, so we only have to check for $n' = 2$ and $n' = 4$. This gives $n = 4$.

5. The Elliptic Curve Discrete Logarithm Problem (ECDLP)

Definition 5.1. An elliptic curve over a field of characteristic $\neq 2, 3$ is the set of solutions to a Weierstrass equation $E : Y^2 = X^3 + Ax + B$, together with an extra point O, where the constants A and B must satisfy $4A^3 + 27B^2 \neq 0$.

The set where A, B, x, and y belong to will have to be specified. Usually, they are taken to be elements of a field. The quantity $\Delta_E = 4A^3 + 27B^2$ is called the *discriminant* of E. The condition that $\Delta_E \neq 0$ is equivalent to the condition that the cubic polynomial $X^3 + Ax + B$ has no repeated roots. The addition law defined below does not work well on curves in which $\Delta_E = 0$.

Definition 5.2. (Addition Law on E) Let P and Q be two points on E. Let L be the line connecting P and Q, or the tangent line to E at P if $P = Q$. Then the intersection of E and L consists of three points P, Q, and R, counted with appropriate multiplicities and with the understanding that O lies on every vertical line. Writing $R = (a, b)$, the sum of P and Q is defined to be the reflection $R' = (a, -b)$ of R across the $X - axis$. This sum is denoted by $P \oplus Q$, or simply by $P + Q$.

Theorem 5.3. Let E be an elliptic curve. Then the addition law on E has the following properties:

$$\begin{cases} P + O = O + P = P & \text{for all } P \in E & \text{[Identity];} \\ P + (-P) = O & \text{for all } P \in E & \text{[Inverse];} \\ (P + Q) + R = P + (Q + R) & \text{for all } P, Q, R \in E & \text{[Associativity];} \\ P + Q = Q + P & \text{for all } P, Q \in E & \text{[Commutativity].} \end{cases}$$

In other words, the addition law makes the points of E into an abelian group.

Theorem 5.4 (Elliptic Curve Addition Algorithm). Let

$$E : Y^2 = X^3 + Ax + B$$

be an elliptic curve and let P_1 and P_2 be points on E.

(a) If $P_1 = O$, then $P_1 + P_2 = P_2$.

(b) Otherwise, if $P_2 = O$, then $P_1 + P_2 = P_1$.

(c) Otherwise, write $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$.

(i) If $x_1 = x_2$ and $y_1 = -y_2$, then $P_1 + P_2 = O$.

(ii) Otherwise, define λ by

$$\lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} & \text{if } P_1 \neq P_2, \\ \frac{3x_1^2 + A}{2y_1} & \text{if } P_1 = P_2, \end{cases}$$

and let

$$x_3 = \lambda^2 - x_1 - x_2 \text{ and } y_3 = \lambda(x_1 - x_3) - y_1.$$

Then $P_1 + P_2 = (x_3, y_3)$.

In order to apply the theory of elliptic curves to cryptography, we need to look at elliptic curves whose points have coordinates in a finite field $\mathcal{F}_p$. Then an elliptic curve over $\mathcal{F}_p$, where $p \neq 2$, 3 is an equation of the form

$$E : Y^2 = X^3 + Ax + B \text{ with } A, B \in \mathcal{F}_p \text{ satisfying } 4A^3 + 27B^2 \neq 0,$$

and then we look at the points of E with coordinates in $\mathcal{F}_p$. We denote this set by

$$E(\mathcal{F}_p) = \{(x, y) : x, y \in \mathcal{F}_p \text{ satisfy } y^2 = x^3 + Ax + B\} \cup \{O\}.$$

We next define the sum of $P_1 + P_2$ to be the point (x_3, y_3) obtained by applying the elliptic curve addition algorithm above (Theorem 5.4). Since the only operations in the algorithm are addition, subtraction, multiplication and division involving the coefficients of E and the coordinates of P and Q which are all belonging to $\mathcal{F}_p$, the result is a point (x_3, y_3) also in $\mathcal{F}_p$.

Theorem 5.5. Let E be an elliptic curve over $\mathcal{F}_p$ and let P and Q be points in $E(\mathcal{F}_p)$, where $p \neq 2$, 3.

(a) The elliptic curve addition algorithm applied to P and Q yields a point in $E(\mathcal{F}_p)$. We denote this point by $P + Q$.

(b) This addition law on $E(\mathcal{F}_p)$ satisfies all the properties listed in Theorem 5.3. In other words, this addition law makes $E(\mathcal{F}_p)$ into a finite abelian group.

Definition 5.6. Let E be an elliptic curve over the finite field $\mathcal{F}_p$, and let P and Q be points in $E(\mathcal{F}_p)$.The *Elliptic Curve Discrete Logarithm Problem (ECDLP)* is the problem of finding an integer n such that $Q = nP$. We denote this integer n by

$$n = \log_P(Q)$$

and we call n the elliptic discrete logarithm of Q with respect to P.

Since the points of $E(\mathcal{F}_p)$,together with the addition operation, forms a finite abelian group, it is isomorphic to the finite abelian group $G = (\mathbb{Z}/m_1\mathbb{Z}) \times (\mathbb{Z}/m_2\mathbb{Z}) \times ... \times (\mathbb{Z}/m_k\mathbb{Z})$ in section 4. However, because of the complicated nature of the addition of points on elliptic curves, it is quite difficult, if not impossible, to determine such an isomorphism between the two groups. Thus, the algorithm described previously for solving the DLP over G will not be applicable.

The collision algorithms can easily be adopted to any group, and so for the group of points $E(\mathcal{F}_p)$ on an elliptic curve. The Index Calculus Method is seen as a faster way to solve the Discrete Logarithm Problem over $\mathcal{F}_p^*$, to wit, it has a *subexponential* running time. The principal reason that elliptic curves are used in cryptography is the fact that there are yet no index calculus algorithms to solve the ECDLP in fewer than $O(\sqrt{p})$ steps. Currently, Pollard's Rho Method is the fastest algorithm available to solve the ECDLP, running $\sqrt{\pi n/2}$, where n is the order of point P [17]. There are, however, special cases of elliptic curves for which an attack to the ECDLP could run faster. One such case is described below.

A Special Case of the ECDLP

One special situation in which the ECDLP can be reduced to an essentially trivial additive *Discrete Logarithm Problem* is when $\#E(\mathcal{F}_p) = p$. Rightly called "anomalous curves" will be the subject of this last section, but first, we need some preliminaries.

The Formal Group of An Elliptic Curve

Let E be an elliptic curve given by a Weierstrass equation

$$(1) \qquad Y^2Z + a_1XYZ + a_3YZ^2 = X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3.$$

In order to define the addition law of the elliptic curve "close to the origin", we first make a change of variables:

$$Z = -X/Y \text{ and } W = -Z/Y$$

Dividing (1) by Y^3 and then doing the change in variables, (1) becomes

$$(2) \qquad W = Z_3 + a_1ZW + a_2Z^2W + a_3W^2 + a_4ZW^2 + a_6W^3.$$

The specified basepoint O on E is now the point $(Z,W)=(0,0)$. Next, we express W as a power series by substituting (2) into itself recursively as follows:

$$\begin{aligned} W &= Z^3+(a_1Z+a_2Z^2)W+(a_3+a_4Z)W^2+a_6W^3 \\ &= Z^3+(a_1Z+a_2Z^2)[Z^3+(a_1Z+a_2Z^2)W+(a_3+a_4Z)W^2+a_6W^3] \\ &\quad +a_6[Z^3+(a_1Z+a_2Z^2)W+(a_3+a_4Z)W^2+a_6W^3]^3 \\ &= Z^3+a_1Z^4+(a_1^2+a_2)^5+(a_1^3+2a_1a_2+a_3)Z^6+... \\ &= Z^3(1+A_1Z+A_2Z^2+...), \end{aligned}$$

where $A_n\in\mathbb{Z}[a_1,a_2,a_3,...,a_6]$ is a polynomial in the coefficients of E. What has been done is we have obtained $W(Z)=W$, by defining a sequence of polynomials by

$$f_1(Z,W)=f(Z,W) \text{ and } f_{m+1}(Z,W)=f_m(Z,f(Z,W)).$$

In order to show that this sequence of polynomials converge to $W(Z)\in\mathbb{Z}[a_1,a_2,a_3,...,a_6][[Z]]$, and we need $W(Z)=f(Z,W(Z))$ to be true in the ring $\mathbb{Z}[a_1,a_2,a_3,...,a_6][[Z]]$.

Proposition 5.7. (a) The procedure described above gives a power series

$$W(Z)=Z^3(1+A_1Z+A_2Z^2+...)\in\mathbb{Z}[a_1,a_2,a_3,...,a_6][Z].$$

(b) The series $W(Z)$ is the unique power series in $\mathbb{Z}[a_1,a_2,a_3,...,a_6][[Z]]$ satisfying $W(Z)=f(Z,W(Z))$.

Claims $(a)$and (b) of the preceding proposition are special cases of Hensel's Lemma whose proof is given below. First, we need the following definition.

Definition 5.8. Let R be a ring, $I\subset R$ an ideal, and let a_n be a sequence in R. (i) a_n converges to a, denoted by $\lim\limits_{n\to\infty} a_n=a$ if and only if for each $\epsilon\in\mathbb{N}$, there exists N such that for all $n\geq N$, $a_n\to a\in I^\epsilon$. We call a the limit of a_n. In mod notation, $a_n\equiv a \bmod I^\epsilon$. (ii) A sequence a_n is a Cauchy sequence if and only if for each $\epsilon\in\mathbb{N}$, there exists an N such that for all $n,m\geq N$, we have $a_m-a_n\in I^\epsilon$. In other words $a_m\equiv a_n (\text{mod } I^\epsilon)$. (iii) A ring R is I-complete if and only if every Cauchy sequence converges to a unique limit in R.

Lemma 5.9. (Hensel's Lemma). Let A be a ring which is complete with respect to an ideal $I\subset A$, and let $F(w)\in A[w]$ be a polynnomial. If for some $m\geq 1$, we have

$$F(0)\in I^m \text{ and } F'(0)\equiv 1 \bmod (I),$$

then there is an element $\alpha\in I^m$ with $F(\alpha)=0$ and the recursion

$$w_0=0,\quad w_{n+1}=w_n-F(w_n) \text{ for } n\geq 0$$

converges to α. If moreover, A is a domain, α is the unique *zero* of F in I.

Proof: Note that since $F(0) \in I$, $w_n \in I^m$ implies $w_n - F(w_n) \in I^m$. Hence, by induction on n, $w_n \in I^m$ for all $n \geq 0$. Next, we prove by induction on n, that

$$w_{n+1} \equiv w_n (\text{mod } I^{m+n}) \quad \text{for } n \geq 0.$$

Suppose that the congruence holds for $n-1$. Let x and y be new variables, and write

$$F(x) - F(y) = (x-y)(F'(0) + xG(x,y) + yH(x,y)), \quad \text{where } G, H \in A[x,y]$$

are certain polynomials. Then we have

$$\begin{aligned} w_{n+1} - w_n &= (w_n - F(w_n)) - (w_{n-1} - F(w_{n-1})) \\ &= (w_n - w_{n-1}) - (F(w_n) - F(w_{n-1})) \\ &= (w_n - w_{n-1}) - (w_n - w_{n-1})(F'(0) + w_n G(w_n, w_{n-1}) \\ &\quad + w_{n-1} H(w_n, w_{n-1})) \\ &= (w_n - w_{n-1})(1 - F'(0) - w_n G(w_n, w_{n-1}) - w_{n-1} H(w_n, w_{n-1})) \\ &\in I^{m+n}, \end{aligned}$$

since $(w_n - w_{n-1}) \in I^{m+n-1}$ by the induction hypothesis and the second factor is in I. This proves that $(w_{n+1} - w_n) \in I^{m+n}$ for all $n \geq 0$.

Since A is complete with respect to I, the sequence $\{w_n\}_{n \geq 0}$ converges to a unique element $\alpha \in A$. Moreover, since $w_n \in I^m$ for all $n \geq 0$, then $\alpha \in I^m$. Taking the limit of the relation $w_{n+1} = w_n - F(w_n)$ as $n \to \infty$ gives $\alpha = \alpha - F(\alpha)$, and so $F(\alpha) = 0$.

To prove uniqueness, suppose $\beta \in I^m$ also satisfies $F(\beta) = 0$. Then

$$0 = F(\alpha) - F(\beta) = (\alpha - \beta)(F'(0) + \alpha G(\alpha, \beta) + \beta H(\alpha, \beta)).$$

If $\alpha \neq \beta$, then $F'(0) + \alpha G(\alpha, \beta) + \beta H(\alpha, \beta) = 0$, and so $F'(0) = -\alpha G(\alpha, \beta) - \beta H(\alpha, \beta) \in I$. This is a contradiction since $F'(0) \equiv 1 (\text{mod } I)$. Hence, $\alpha = \beta$. ■

From the power series $W(Z)$, we obtain Laurent series for X and Y

$$\begin{aligned} X(Z) &= \frac{Z}{W(Z)} = \frac{1}{Z^2} - \frac{a_1}{Z} - a_2 - a_3 Z - (a_4 + a_1 a_3) Z^2 - \ldots \\ Y(Z) &= -\frac{1}{W(Z)} = -\frac{1}{z^3} + \frac{a_1}{Z^2} + \frac{a_2}{Z} + a_3 + (a_4 + a_1 a_3) Z - \ldots \end{aligned}$$

and the invariant differential

$$\begin{aligned}\omega(Z) &= \frac{dX(Z)}{2Y(Z)+a_1X(Z)+a_3} \\ &= (1+a_1Z+(a_1^2+a_2)Z^2+a_1^3+2a_1a_2+2a_3)Z^3 \\ &\quad +(a_1^4+3a_1^2+6a_1a_3+a_2^2+2a_4)Z^4+...)dZ\end{aligned}$$

all of whose coefficients are in $\mathbb{Z}[a_1,a_2,a_3,...,a_6][[Z]]$. The pair $(X(Z),Y(Z))$ provides a formal solution to the Weierstrass equation

$$E: Y^2+a_1XY+a_3Y = X_3+a_2X^2+a_4X+a_6$$

that is, a solution in the quotient field of the ring of formal power series.

Definition 5.10. (The Formal Addition Law). Let Z_1 and Z_2 be independent indeterminates, and let $W_1 = W(Z_1)$ and $W_2 = W(Z_2)$. Then

$$\begin{aligned}F(Z_1,Z_2) &= Z_1+Z+2-a_1Z_1Z_2-a_2(Z_1^2Z_2+Z_1Z_2^2) \\ &\quad +(2a_3Z_1^3Z_2+(a_1a_2-3a_3)Z_1^2Z_2^2+2a_3Z_1Z_2^3)+... \\ &\in \mathbb{Z}[a_1,a_2,a_3,...,a_6][[Z_1,Z_2]].\end{aligned}$$

$F(Z_1,Z_2)$ has the following properties:

$$\begin{cases} F(Z_1,Z_2)=F(Z_2,Z_1) & \text{(Commutativity);} \\ F(Z_1,F(Z_2,Z))=F(F(Z_1,Z_2),Z) & \text{(Associativity);} \\ F(Z,i(Z))=0 & \text{(Inverse),} \end{cases}$$

where $i(Z) = \frac{X(Z)}{Y(Z)+a_1X(Z)+a_3} = \frac{Z^{-2}-a_1Z^{-1}-...}{-Z^{-3}+2a_1Z^{-2}+...} \in \mathbb{Z}[a_1,a_2,a_3,...,a_6][[Z]]$.

Definition 5.11. (Formal Group). Let R be a ring. *A (one-parameter commutative) formal group $\mathcal{F}$ over* R is a power series $F(X,Y)\in R[X,Y]$ with the following properties:

$$\begin{cases} F(X,Y)=X+Y+(\text{terms of degree } \geq 2); \\ F(X,F(Y,Z))=F(F(X,Y),Z) \quad \text{(Associativity);} \\ F(X,Y)=F(Y,X) \quad \text{(Commutativity);} \\ \text{There is a unique power series } i(T)\in R[T] \text{ such that } F(T,i(T))=0 \text{ (inverse);} \\ F(X,0)=X \text{ and } F(0,Y)=Y. \end{cases}$$

We call $F(X,Y)$ *the formal group law of* $\mathcal{F}$. The *formal additive group,* denoted by $\widehat{G}_a$ is defined by

$$F(X,Y)=X+Y$$

and the *formal multiplicative group*, denoted by $\widehat{G}_m$ is defined by

$$F(X,Y) = X + Y + XY = (1+X)(1+Y) - 1.$$

Definition 5.12. (The Formal Group Associated to an Elliptic Curve). Let E be an elliptic curve given by a Weierstrass equation (1) with coefficients in a ring R. Then the formal group asssociated to E, denoted by $\widehat{E}$, is defined by the power series $F(Z_1, Z_2)$ described in Definition 5.10.

In other words, $\widehat{E}(X,Y) \in R[[X,Y]]$ and it satisfies the following conditions:

$$\begin{aligned} \widehat{E}(X,Y) &= X + Y + (\text{terms of degree } \geq 2) \\ \widehat{E}(X,\ \widehat{E}(Y,Z)) &= \widehat{E}\ (\widehat{E}(X,Y), Z) \\ \widehat{E}(X,Y) &= \widehat{E}(Y,X). \end{aligned}$$

A formal group can be thought of as merely a group operation with no underlying group. However, if R is a complete local ring, and if the variables are assigned values in the maximal ideal $\mathcal{M}$ of R, then power series defining the formal group converge and give $\mathcal{M}$ the structure of a group [14].

Let R be a complete local ring , and let K be the quotient field of R. Let $\mathcal{M}$ be the maximal ideal of R, and let k be the residue field $R/\mathcal{M}$. Let $\mathcal{F}$ be a formal group defined over R, with formal group law $F(X,Y)$, and let $\mathcal{F}(\mathcal{M})$ denote the set $\mathcal{M}$ endowed with the group operations

$$\begin{cases} X \oplus_F Y = \widehat{E}(X,Y) & \text{(addition)} \quad \text{for } X, Y \in \mathcal{M}, \\ \ominus_F X = \iota(X) & \text{(inversion)} \quad \text{for } X \in \mathcal{M}. \end{cases}$$

The assumption that R is complete ensures that the power series $\widehat{E}(X,Y)$ and $\iota(X)$ converge in R for all $X, Y \in \mathcal{M}$. The formal group axioms immediately imply that $\mathcal{F}(\mathcal{M})$ is a group. Similarly, $\mathcal{F}(\mathcal{M}^n)$ is the subgroup of $\mathcal{F}(\mathcal{M})$ consisting of the set $\mathcal{M}^n$ with the above-described group operations.

Note that, in particular, if $R = \mathbb{Z}_p$, the ring of p-adic integers, then we have $K = \mathbb{Q}_p$,the field of p-adic numbers, $\mathcal{M} = p\mathbb{Z}_p$, and $k = \mathbb{Z}_p/p\mathbb{Z}_p = \mathbf{F}_p$.

We now let p be a prime number, and E be an elliptic curve defined over $\mathbb{Q}_p$, given by a Weierstrass equation (1), with coefficients in $\mathbb{Z}_p$, and assumed to have a good reduction at p. What this means is that if we reduce the coefficients of E modulo p, we can obtain an elliptic curve $\widetilde{E}$ over $\mathbf{F}_p$, where $\mathbf{F}_p$ is the residue field $k = \mathbb{Z}_p/p\mathbb{Z}_p$. Let $E_1(\mathbb{Q}_p)$ denote the set of points in $E(\mathbb{Q}_p)$ which reduce to zero modulo p, and let $E_0(\mathbb{Q}_p)$ denote the set of points in $E(\mathbb{Q}_p)$ which reduce modulo p to an element of $\widetilde{E}(\mathbf{F}_p)$. Since E is assumed to have a good reduction at p, $E(\mathbb{Q}_p) = E_0(\mathbb{Q}_p)$. Then we have an exact sequence of abelian groups [14],

$$0 \to E_1(\mathbb{Q}_p) \to E_0(\mathbb{Q}_p) \xrightarrow{\pi} \widetilde{E}(\mathbf{F}_p) \to 0,$$

where π is the reduction map, and $E_1(\mathbb{Q}_p)$ is $\ker(\pi)$. Thus, if we multiply an element of $E_0(\mathbb{Q}_p)$ by the number of elements in $\widetilde{E}(\mathbf{F}_p)$, or a multiple thereof, what we get is an element in $E_1(\mathbb{Q}_p)$.The p-adic elliptic logarithm of E is given by

$$\begin{aligned} \log(z) &= \int \omega(z)dz \\ &= z + \frac{a_1}{2}z^2 + \frac{a_1^2 + a_2}{3}z^3 + \ldots \in \mathbb{Q}_p[[z]]. \end{aligned}$$

This p-adic elliptic logarithm induces a group homomorphism:

$$\begin{aligned} \log_E &: E_1(\mathbb{Q}_p) \to \mathbb{Q}_p^+, \\ z &\mapsto \log_E(z), \end{aligned}$$

where $\mathbb{Q}_p^+$ denotes the additive group of $\mathbb{Q}_p$. Moreover, the group $E_1(\mathbb{Q}_p)$ is isomorphic to the formal group associated to E, $\mathcal{F}(\mathcal{M}) = \widehat{E}(\mathcal{M}) = \widehat{E}\,(p\mathbb{Z}_p)$ and the isomorphism is given by

$$\begin{aligned} \widehat{E}\,(p\mathbb{Z}_p) &\to E_1(\mathbb{Q}_p), \\ z &\mapsto \begin{cases} O \text{ if } z = 0 \\ (\frac{z}{w(z)}, -\frac{1}{w(z)}), \text{ if otherwise,} \\ i.e., z = \frac{-x}{y}. \end{cases} \end{aligned}$$

(See [14], p.191).

Proposition 5.13. (Semaev [11], Satoh-Arakaki [12], Smart [15]) Let $p \geq 3$ and let $E/\mathcal{F}_p$ be an (anomalous) elliptic curve satisfying

$$\#E(\mathcal{F}_p) = p.$$

The following algorithm solves the ECDLP in $E(\mathcal{F}_p)$.

(1) Let $P, Q \in E(\mathcal{F}_p)$ be nonzero points satisfying $Q = [m]P$, where modulo p, the integer m is not known.
(2) Choose an elliptic curve $E'/\mathbb{Q}_p$ whose reduction modulo p is $E/\mathcal{F}_p$.
(3) Use Hensel's Lemma to lift the points P, Q to points P', $Q' \in E'(\mathbb{Q}_p)$.
(4) The points $[p]P'$ and $[p]Q'$ are in the formal group $E_1^{'}(\mathbb{Q}_p)$. Let

$$\log_E : E_1'(\mathbb{Q}_p) \to \hat{G}_a(p\mathbb{Z}_p) \cong p\mathbb{Z}_p^+$$

be the formal logarithm map, and compute $pa = \log_E([p]P') \in p\mathbb{Z}_p$ and $pb = \log_E([p]Q') \in p\mathbb{Z}_p$.
(5) Then $m \equiv a^{-1}b (\operatorname{mod} p)$.

Proof: Using the fact that $\#E(\mathcal{F}_p) = p$, we have

$$\widetilde{[p]P} = [p]P = O \text{ and } \widetilde{[p]Q'} = O \text{ in } E(\mathcal{F}_p),$$

so $[p]P'$ and $[p]Q'$ are in the kernel of reduction modulo p. Hence, they are in the formal group $E_1(\mathbb{Q}_p)$ by Theorem 5.14. Similarly, if we let $R' = Q' - [m]P'$, then the reduction of R' modulo p is

$$\widetilde{R'} = \widetilde{Q'} - [m]P' = [m]P = O \text{ in } E(\mathcal{F}_p)$$

so $R' \in E_1'(\mathbb{Q}_p)$. We then compute

$$\begin{aligned}
\log([p]Q' &= \log_E([p]([m]P' + R)) && \text{(since } R' = Q' - [m]P')\\
&= m\log_E[p]([p]P' + p\log_E(R') && \text{(valid since } [p]P', R' \in E_1(\mathbb{Q}_p))\\
&= m\log_E([p]P')(\operatorname{mod} p^2) && \text{(since } \log_E(R') \in p\mathbb{Z}_p).
\end{aligned}$$

Substituting the values $\log_E([p]P') = pa$ and $\log_E([p]Q') = pb$ of the algorithm gives $pb \equiv mpa(\operatorname{mod} p^2)$, so $m \equiv a^{-1}b(\operatorname{mod} p)$ [14].

We note, in conclusion, that currently, the best known algorithms to solve the Discrete Logarithm Problem may well be categorized into two: collision search algorithms and index-calculus algorithms. They differ in the kind of objects on which they can be applied and also in their computing time. A collision search algorithm works for any finite group, while an index-calculus method requires certain arithmetic properties of the group for it to be successful. The former has purely *exponential* running time, the best general method of which, so far, is the Pollard Rho Algorithm which runs $O(\sqrt{n})$, with n the size of the group. The latter runs in *subexponential* running time, not as fast as *polynomial* time algorithms, but considerably faster than the *exponential* time methods. Because of its requirement for special properties of the group, the index-calculus method cannot be used to attack any elliptic curve discrete logarithm based cryptosystem. For the Elliptic Curve Discrete Logarithm Problem (ECDLP), the fastest algorithm currently available is the Pollard Rho Algorithm which takes about $\sqrt{\pi n/2}$ steps, where a step here is an elliptic curve addition, and n the order of a point $P \in E(\mathbb{Z}_p)$ in the ECDLP as defined in section 5 of this paper .

Quite remarkably, for over three decades, the Discrete Logarithm Problem has received enormous attention from leading mathematicians and cryptographers around the world, but no significant weakness of it has yet been discovered. Until the Diffie-Hellman conjecture is controverted, and proven otherwise, the Discrete Logarithm Problem will continue to fascinate people fortunately endowed with the mathematical mind and curiosity, and whence, it will never be devoid of its practical or academic significance.

Bibliography

[1] Buchmann, J. A., Introduction to Cryptography, Springer-Verlag, NY 2001.

[2] Diffie, W. and Hellman, M., New Directions in Cryptography, IEEE Trans. Information Theory (1976), 472-492.

[3] ElGamal, T., A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms, Advances in Cryptography - Crypto '84, volume 196 of Lecture Notes in Compute Science, Springer-Verlag, NY 1985.

[4] Hoffstein, J., Pipher, J., and Silvermann, J., An Introduction to Mathematical Cryptography, Springer-Verlag, NY 2008.

[5] Koblitz, N., A Course in Number Theory and Cryptography, Springer-Verlag, NY 1994.

[6] McCurley K.S., The Discrete Logarithm Problem, Cryptology and Computational Number Theory, volume 42, pages 49-74, American Mathematical Society, 1990.

[7] Pollard, J.M., Monte Carlo Methods for Index Computation (mod p), Mathematics of Computation volume 32, pages 918-924, 1978.

[8] Pomerance, C., Elementary Thoughts on Discrete Logarithms, Arithmetic Number Theory, MSRI Publications, Volume 44, 2008.

[9] Rivest, Shamir, Adleman, A Method of Obtaining Digital Signatures and Public Key Cryptography, Communications of the ACM, 21(2): 120-126, 1978.

[10] Rosen, K.H., Elementary Number Theory and its Applications, Addison-Wesley Publishing Company, NY 1993.

[11] Satoh, T., and Araki, K., Fermat Quotients and the Polynomial Time Discrete Logarithm Algorithm for Anomalous Elliptic Curves, Comment. Math. Univ. St. Paul., 47(1):81-92, 1998.

[12] Semaev, I.A., Evaluation of Discrete Logarithms in a Group of p-Torsion Points of an Elliptic Curve in Characteristic p, Mathematics of Computation, 67(221):353-356, 1998.

[13] Silverman, Joseph and Tate, John, Rational Points on Elliptic Curves, Springer-Verlag, NY 1992.

[14] Silverman, J. H., The Arithmetic of Elliptic Curves, Springer-Verlag, NY, 1992.

[15] Smart, N.P., The Discrete Logarithm Problem on Elliptic Curves of Trace One, J. Cryptology, 12(3):193-196, 1999.

[16] Online material @www.strumpfer.net/discrete_log_attacks.

[17] Online material @www.home.esat.kuleuven.be/~fvercaut/talks/ECDL.pdf.

[18] Washington, Lawrence C., Elliptic Curves, Chapman & Hall/CRC, Florida 2003.