Index Calculus Attack Algorithms on the DLP

Samira C. Oliva Madrigal
samira.oliva@sjsu.edu
San José State University
San Jose, California, USA

ABSTRACT

The Index Calculus Method (ICM) is a method for solving the discrete logarithm of a finite field with sub-exponential complexity. It was first developed in 1922 by Kraitchik. Index calculus algorithms conform to finite fields and apply to specific families of elliptic curves. ICM algorithms are the most important family of algorithms that yield sub- exponential time algorithms by making use of additional information in underlying groups. In this paper we present the classical ICM method and explore contemporary ICM algorithms that can be used to attack cryptographic schemes based on the Discrete Logarithm Problem (DLP).

KEYWORDS

index-calculus, discrete-logarithm problem

ACM Reference Format:
Samira C. Oliva Madrigal. 2019. Index Calculus Attack Algorithms on the DLP. In *Woodstock '19: ACM Symposium on Neural Gaze Detection, May 06–07, 2019, Woodstock, NY*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

The logarithmic function, $x = log_a(y)$ that solves the log of y for $y = a^x$ applicable to $\mathbb{R}_{>0}$ can also be applied to modular arithmetic and defined for any group. Some of the most common cryptographic schemes, such as the Diffie-Hellman Key-Exchange algorithm, the RSA public-key scheme, and Elliptic-Curve Cryptography rely on the hardness of the DLP. Currently, known the best attack algorithms on the DLP problem which are applicable to any group have exponential complexity [4]. However, ICM methods offer an efficient means of solving the DLP problem in sub-exponential time for specific groups.

2 DLP

For modular arithmetic, we can define the DLP as follows. If we let p be a prime number and a be a primitive root of p whose powers from 1 to $p-1$ are all different from each other and generate all nonzero elements in $mod(p)$. Then, we know that for any $y, k \in mod(p)$ that $y \equiv kmod(p)$ where $k \in [0, p-1]$. It follows that for any y, we can compute its discrete log for base $amod(p)$ as the number x such that $y \equiv a^x mod(p)$ for $x \in [0, p-1]$. For groups, we can define the DLP for any group G, similarly by the following equation.

$$y = a^x \text{for } a, y \in G \tag{1}$$

In this case, the discrete logarithm of y with base a is the number x applied to a such that $a^x = y$. In this case, we are interested in the smallest such number x. In general, it should be easy to compute y provided a and x but hard to find x given y and a. When considering any group, the problem has exponential complexity.

2.1 DLP-based Cryptosystems

DLP is fundamental to cryptography as is it the problem on which the some of the most used public-key cryptography schemes are based on. In the following subsections we discuss D-H and a variant.

2.1.1 D-H. The D-H key-exchange algorithm offers a way for two parties to generate a shared secret key. In this case, two parameters are public, a primitive root a of a prime p and p. In order for two end points A and B to generate a secret key, each must generate their private key as a random number r_A and r_B respectively, both $< p$. Then each party computes their public keys as $p_A = a^{r_A} mod(p)$ and $p_B = a^{r_B} mod(p)$ and transmit to the other. Upon receipt A computes the shared secret key as $k = p_B^{r_A} mod(p)$ and B similarly, $k = p_A^{r_B} mod(p)$.

The best attack on D-H in entails computing the discrete log or a private key in the scheme with which computing the shared key is then trivial. D-H DLP problem is hard depending on how the parameters are chosen.

2.1.2 ECC-DH. In elliptic curve cryptography, we consider points defined over an elliptic curve and operations on these. In general, Elliptic curves are described equations with the form of the Weierstrass equation $y^2 + axy + by = x^3 + cx^2 + dx + e$ in two variables and coefficients a, b, c, d, and e. For cryptography, the variables as well as coefficients are defined over a finite abelian group. An ECC-based schemes works as follows. The public parameters consists of $E_p(a, b)$ which denotes the set of points (a, b) which satisfy the elliptic curve equation, G a base point in the set with large order k such that $Gk = 0$, and the order k. Similarly, two endpoints A and B compute their private keys as r_A and r_B, respectively, both less than k. Then, they each compute their public key as a points on the set $p_A = r_A G$ and $p_B = r_B G$. A then computes the shared key as $k = r_A p_B$ and B as $k = r_B p_A$.

The DLP in this scheme is then, given a point $p = rG$ on the set, what is a private key r given only rG and G. It's difficulty is depending on the complexity required to solve this problem.

3 INDEX CALCULUS

The index calculus method refers to solving the discrete logarithm or index for the DLP and refers to sieving techniques. It's a probabilistic algorithm that originates from number theory and is applicable to finite fields and specific types of elliptic curves. ICM computes the discrete log using linear algebra taking as input the discrete logs of small primes. Some of the steps are very easy to run in parallel and others are not.

3.1 Classical IC algorithm

The IC method was originally discovered by Maurice Kraitchik in his work from number theory in 1922. His method is tries to answer the question on how to compute the index for a given number or the indices for a set of number bounded by some number according the group in question. The method consist of finding relations among small primes and putting intermediate results together using techniques from linear algebra. For a given prime number p and a primitive root of p, a, consider a set of primes $\{\rho_1, \rho_2, ..., \rho_k\}$ such that they are all $\leq$ a fixed bound S. Then, as our first step, we are interested in finding relations between products of powers of these primes that are congruent to a power of the base a to the $mod(p)$ which is expressed by the following equation.

$$\prod_j \rho_j^{x_j} \equiv a^x mod(p) \tag{2}$$

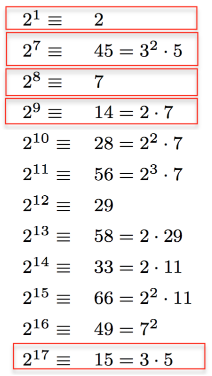
This relation can be expressed more explicitly in term of the discrete log as:

$$\prod_j a^{\xi_{amod(p)}(\rho_j)x_j} \equiv a^x mod(p) \tag{3}$$

where ξ is the index or discrete log function meaning that for some e, $\xi(e)$ computes the discrete log or index of e for base a to the $mod(p)$. And hence, we can expresses the above relation linearly, as a sum (using laws of logarithms for exponents and products) on the indices.

$$\sum_j x_j \xi_{amod(p)}(\rho_j) \equiv xmod(p-1) \tag{4}$$

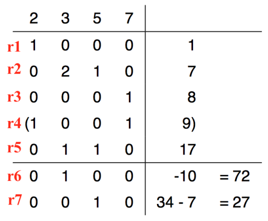
For clarity, consider an example from [1]: let prime $p = 83$ with primitive root $a = 2$ and four smalls primes $\leq S = 7$, $\rho_1 = 2$, $\rho_2 = 3$, $\rho_3 = 5$, and $\rho_4 = 7$. Then, we can find relations for powers of a that are congruent to a product of powers of these primes in $mod(83)$. From figure 1, we can see the left column corresponds to powers of the form $a^x mod(p)$ and the left column correspond products of numbers to some powers which are congruent to the left side. For example, for the first row we have $a^1 mod(83) \equiv \rho^1$, in the second row we have $a^7 mod(83) \equiv \rho_2^2 \rho_3^1$, in the third row $a^8 mod(83) \equiv \rho_4^1$, and so on. Hence, if we select some of these relations, we can generate a linear system according to equation 4 as in figure 2. We are interested in the powers of a which are congruent to the power of a single prime. For instance the first row in Figure 2 corresponds to the first row of 1 if we consider the primes are variables, then $a^1 mod(p) \equiv 2^1 = \rho_1^1$ can be expressed the linear equation $a^1 \equiv \rho_1^1 + \rho_2^0 + \rho_3^0 + \rho_4^0$, using the indices: $1 = 1 + 0 + 0 + 0$ or equivalently, $1 + 0 + 0 + 0 = 1$. Hence, we take all rows selected in 1 to get the top part (first five rows) of linear system in $\mathbb{Z}/82\mathbb{Z}$. We are only interested in the row 1 (r1) and row 3 (r3) since for these equations only have an exponent for one of the primes, ρ_1 and ρ_4



$2^1 \equiv$	2
$2^7 \equiv$	$45 = 3^2 \cdot 5$
$2^8 \equiv$	7
$2^9 \equiv$	$14 = 2 \cdot 7$
$2^{10} \equiv$	$28 = 2^2 \cdot 7$
$2^{11} \equiv$	$56 = 2^3 \cdot 7$
$2^{12} \equiv$	29
$2^{13} \equiv$	$58 = 2 \cdot 29$
$2^{14} \equiv$	$33 = 2 \cdot 11$
$2^{15} \equiv$	$66 = 2^2 \cdot 11$
$2^{16} \equiv$	$49 = 7^2$
$2^{17} \equiv$	$15 = 3 \cdot 5$

Figure 1: Classical ICM relations [2]

respectively. In order to generate such rows for the other primes, we need to apply elementary row operations (EROS). For example, to generate row 6 (r6) and isolate a 1 in the second variable, we can compute $-1xr5 + r2 = r6$ and to generate row 7 (r7) and isolate a 1 in the third variable, we can compute $-1xr2 + 2xr5 = r7$. All results are in $mod(p-1)$.



	2	3	5	7		
r1	1	0	0	0	1	
r2	0	2	1	0	7	
r3	0	0	0	1	8	
r4	(1	0	0	1	9)	
r5	0	1	1	0	17	
r6	0	1	0	0	-10	= 72
r7	0	0	1	0	34 - 7	= 27

Figure 2: Classical ICM relations to indecis [2]

From figure 2, we can see that the discrete logs of the small primes are $\xi_{amod(p)}(\rho_1) = 1$, $\xi_{amod(p)}(\rho_2) = 72$, $\xi_{amod(p)}(\rho_3) = 7$, and $\xi_{amod(p)}(\rho_4) = 8$. If we wanted to compute the discrete log of a larger prime, we could do so in terms of these smaller primes. To do so, we first state the laws of exponents in $\mathbb{R}_{>0}$ and the modular equivalents. For exponentiation and multiplication, in $\mathbb{R}_{>0}$, we have $log_a(b^r) = rxlog_a(b)$ and $log_a(bxc) = log_a(b) + log_a(c)$, respectively. Similarly, for modular arithmetic, we have $\xi_{amod(p)}(y^r) = [rx\xi_{amod(p)}(y)]mod(p)$ and $\xi_{amod(p)}(yxz) = [\xi_{amod(p)}(y) + \xi_{amod(p)}(z)]mod(p-1)$.

Now, consider the case where we want to compute the discrete log of a large number such as 31, so $\xi_{amod(p)}(31)]mod(p)$. We can proceed as follows using the exponent law for logs, since $31^2 \equiv 48mod(p) = 2^4x3$, we can rewrite that as $2x\xi_{amod(p)}(31) \equiv 48mod(p) = 2^4x3 = \xi_{amod(p)}(2^4)x\xi_{amod(p)}(3) = 4x\xi_{amod(p)}(2) + \xi_{amod(p)}(3) = 4x1 + 72 = 76mod(p-1)$, and $2x\xi_{amod(p)}(31) = 76/2 = 38$. As a consequence, we have the following theorem.

THEOREM 3.1. *If a if a primitive root of a prime p, and some number y, the discrete logarithm of y can be computed to the base amod(p) in time:* $L_p[1/2, \sqrt{2}] = e^{(\sqrt{2}+o(1))(log(p)^{1/2})(loglog(p)^{1-1/2})}$ *which is sub-exponential in* $log(p)$.

This theorem applies to the finite group of integers in mod(p) which in turn is generalized to finite fields of integers over mod(p) and mod(q). More formally, we can say that if $(G, +)$ denotes any finite group, then for $x, y \in G$ such that $y \in \langle x \rangle$ the cyclic subgroup of x, then we can say that the discrete logarithm of y to the base x is the smallest index $i \in \mathbb{Z}^+$ such that $y = x^i$. In this case we call x a generator of G, the element whose powers (which are also in G) generate all nonzero elements in G. Hence, this applies to $\mathbb{F}_p^*$ with x as the generator element. This can generalized to $\mathbb{F}_q^*$, where q has form p^k and in both cases, the field is cyclic since its generators are primitive roots. In general, for any cyclic group G, its generator x, for $y \in \langle x \rangle$, and $ord(x)$ prime, we can compute $\xi_x(y)$ in $O(\sqrt{ord(x)})$. The idea is that, we examine multiples of x and y of the form x^r and y^t until we find an r and t such that $x^r = y^t$, then if t has an inverse in $mod(ord(x))$, the discrete log is computed as $i = r/t$ which is in $\mathbb{Z}/ord(x)Z$. This is less attempts than the birthday paradox and for a non-prime $ord(x)$, we apply the Chinese Remainder Theorem (CRT).

Algorithm 1 ICM main steps

```
1: define suitable set of small primes $\{\rho_1, ..., \rho_k\} \subseteq G$
2: find relations between inputs and the $\rho_i$ (Equation 2)
3: use linear algebra to compute the indices for the $\rho_i$ (Equation 4)
4: compute indices of larger primes in $G$ in terms of the $\rho_i$
```

In general, we only assume that $ord(x)$ given but not that x is a generator of the group G.

Algorithm 2 General ICM Algorithm

```
inputs: $G, x, y \in G$
1: define suitable set of small primes (i.e. the factor base)
   $\{\rho_1, ..., \rho_k\} \subseteq G$
   define: column vectors $\alpha := (\alpha_i)_i$ and $\beta := (\beta_i)_i$.
   relations matrix $R := ((x_{i,j}))_{i,j}$ on which the following relations
   are stored, each $(x_{i,j})$ is a vector that becomes and a row in $R$
2: collect $k + 1$ linear relations of the form: $\sum_j x_{i,j}\rho_j = \alpha_i a + \beta_i y$
3: compute $\gamma$, a non-trivial vector in $\mathbb{Z}/ord(a)\mathbb{Z}^{1x(k+1)}$
   such that $R\gamma = 0$.
4: now, we can express the following:   $\sum_i \gamma_i\alpha_i a + \sum_i \gamma_i\beta_i y = 0$
   if $\sum_i \gamma_i\beta_i \in \mathbb{Z}/ord(a)\mathbb{Z}^*$
      then, $\xi_a(y) = (\sum_i \gamma_i\alpha_i)/(\sum_i \gamma_i\beta_i)$
      or $\xi_a(y) = \sum_i \gamma_i\alpha_i(\sum_i \gamma_i\beta_i)^{-1}$
```

In the above algorithm, each entry in R for row i and column j is defined by the formula $x_{i,j}$, meaning that each $x_{i,j}$ in step 2 constitutes an entry in R. Similarly, each α_i forms the i-th entry of vector α and the same follows for vector β. The subscript outside the parenthesis are a bit superfluous and meant to specify valid dimensions for the matrix and vectors. In fact, R is $(k + 1)xk$ and α and β are $1x(k+1)$, defined over $\mathbb{Z}/ord(a)\mathbb{Z}^{(k+1)xk}$ and $\mathbb{Z}/ord(a)\mathbb{Z}^{1x(k+1)}$, respectively. The α and β elements may be generated uniformly and at random from $\mathbb{Z}/ord(a)\mathbb{Z}$ depending on the implementation and group [3][6]. In step 2, we find a γ such that we can express $R\gamma = 0$ as a homogenous systems of equations (i.e. with form $Ax = 0$) but γ cannot be the trivial solution vector ($Ax = 0$, with $x = 0$), but rather $\gamma \neq 0$, a non-trivial solution vector for the linear system as required. Finally, in step 4 we can compute the discrete logarithm of y with respect to the base a.

If we consider the same problem over $\mathbb{F}_p^*$, we consider a surjective homomorphism between monoids and lifting from $\mathbb{F}_p^*$ to $\mathbb{N}'$ where $\mathbb{N}'$ is composed of all elements in $\mathbb{N}$ not equal to prime p, provided by the mapping from $[N]_p$ to $N \iff N \in [1, p)$ [2][10][11]. Then for a factor base with all elements $\leq S$ such that each element p_i in the factor base is defined as $[p_i]_p$, we can proceed in the following manner. For $e \in \mathbb{F}_p^*$, we "lift" e to $\mathbb{N}$, i.e. let N uniquely represent e which is $< p$. Then, we try factoring N over the factor base. If N can be factored with form $\prod_j p_j^{r_j}$, we can obtain a relation of the for e of the form $\prod_j p_j^{r_j}$. In the following table we provide the complexity for solving the DLP in several different types of groups. However,

Complexity	Group type
sub-exponential	$(\mathbb{Z}/n\mathbb{Z})^*$ $\mathbb{F}_p$ class groups from a number field
quasi-polynomial	$\mathbb{F}_p$ with small p
exponential	EC groups in general hyper EC groups of genus 2

Table 1: Complexity For Sample Group Types [5]

Gaundry has presented an algebraic method when considering elliptic curves defined over extension fields which can solve the DLP in polynomial time [2]. In this case we consider an elliptic curve E defined over a finite field $\mathbb{F}$ of order $q = p^n$ for a prime number p and positive integer n which is generated from equation $y^2 = f(x)$. Then, we can consider all points P in $E(\mathbb{F}_p)$ such that xP is in $\mathbb{F}_p$. In this manner, by we can solve multivariate systems defined in $\mathbb{F}_p$ to generate relations. This gives us a result that allows for solving the DLP in time that is polynomial in p if $\epsilon > 0$ and $a > \epsilon + 2 > 0$ [2].

4 ICM FAMILY

IC which has led to an interesting family of algorithms, namely, the function field sieve, the Number Field Sieve for Discrete Logarithms addressing fields of form $\mathbb{F}_q$ for $p \gg q$, the Number Field Sieve in High Degree and Joux when for fields with small characteristic [14][9]. For specific types of elliptic curves, the problem can be solved in subexponential time or polynomial time in p as noted earlier. In this section we discuss Guillevic's improvement to the last step fo the Number Field Sieve, Joux's algorithm addressing fields with small charactestic, and the significance of ICM in cryptography.

4.1 Number Field Sieve for Discrete Logarithms

The Number Field Sieve for Discrete Logarithms (NFS-DL) is currently the best classical attack algorithm when considering finite fields of the form $\mathbb{F}_{p^n}$ where p is in the average to large scale (whenever $p \gg q$) and n is a small integer ≥ 1. NFS-DL follows steps similar to 2 except that in the first step (see Figure 6) we select $f(x)$

and $g(x)$, irreducible polynomials, that define K_f and K_g, number fields. We also consider two polynomial rings defined over $\mathbb{Z}$ inside modulo each defining polynomial and maps ρ_f and ρ_g which maps the rings into $\mathbb{F}_{p^n}$ which is defined by a monic, $\psi(x)$ with degree n that is computed as the $gcd(f, g) mod(p)$. The last three steps are as usual, finding relations, applying linear algebra methods, and then computing discrete logarithms for only a subset from each ring. For a target element defined in $\mathbb{F}_{p^n}$, there exists a mapping in either of the rings. If such preimage can be expressed as a product of known logarithms of smaller elements, then, its discrete log can be computed.

$$\begin{tikzcd}
 & \mathbb{Z}[x] \arrow[dl] \arrow[dr] & \\
R_f = \mathbb{Z}[x]/(f(x)) \arrow[dr, "\rho_f : x \mapsto z"'] & & \mathbb{Z}[y]/(g(y)) = R_g \arrow[dl, "\rho_g : y \mapsto z"] \\
 & \mathbb{F}_{p^n} = \mathbb{F}_p[z]/(\psi(z)) &
\end{tikzcd}$$

Figure 3: Number Field Sieve mappings [8]

As we showed, previously, we can compute the discrete logarithm of a larger element in terms of the discrete logs of smaller ones. To compute the discrete logarithm of an element $e \in \mathbb{F}_{p^n}$, the last steps provides us with the preimage of one of the two number fields. Consequently, if we can express e as product consisting of elements for which we know the discrete log, then, we can compute the discrete log of e similar as in our earlier example. If this can be done efficiently, then DLP schemes can be broken depending on other factors. For instance, the Logjam attack from 2015 leveraged the fact the only the first three steps of the ICM or NFS in this case, rely on the $\#G$ and not on the target element for which we wish to compute the discrete log. In this case, the authors of Logjam found that by saving the computations of first three steps, the discrete log of any element of the group can be computed efficiently, applicable only for keys in the 512-1024-bit size. In the case of elements in the two 512-bit fields, the discrete log for each element in the last step was in 1 minute which means it is vulnerable to man-in-the-middle attacks [1].

During the early work from Joux et al. (JLSV), Barbulescu analyzed the complexity of the last step in three steps [8]. The first is smoothing (or booting) factoring the target element into smaller elements. This is followed by descent, in which the factors from the first step are each in turn factored recursively their degree is within a smoothness bound B_1 and output are elements of smaller degree that are in the factor base. The third computes the log of the target in terms of the logs of smaller elements into which it was decomposed. When the characteristic of the field is in the scale of medium to large and non-prime for $n \geq 3$, the last part in the NFS is significantly slower. In [8] Guillevic optimizes the the booting step applicable to extension fields on the small scale for $n \in [2, 6]$ as well as for those with medium or large p with $n \geq 1$.

The improvement in booting is made by making the norm of preimage in the ring significantly smaller because the complexity of smoothing is dependent on it. The norm of the preimage is contingent on the degree of the defining polynomial for the number field. More interestingly, the norm decreases as the degree and coefficients of the defining polynomial decrease. Different implementations have leveraged this by trying to output polynomials for the two rings such that one has small coefficients and the other has them $O(p)$.

In Algorithm 3, we see the three main steps involved in the last step of the NFS. [8] combines computing the preimage of the target and smoothing into the "booting step". Given a generator of the field, and a database of discrete logs for a subset of elements, this algorithm approaches the steps involved in computing the discrete logarithm for a target that is in the field and for which the discrete logarithm can be found in terms of the logs of smaller elements.

Algorithm 3 Booting Step [7]

```
Input: generator $G$, $database$ of logs for elements $p_i < B_0$
Ouput: $DL(T_0) \in \mathbb{F}^*_{p^k}$
1: boot
  do:
    1.1: compute preimage of target
      $t :=$ randomly selected from $[1, \ell]$
      $T := T_0 G^t$
    1.2: smooth preimage
      norm(T) = $q_1 x q_2 ... x q_i \in database$
  until (all the $q_i \leq B_1$)
2: recursive descent
  $S := NIL$
  for each $q_i$
    find relation $q_i = q_1 x q_2 ... x q_j$ s.t. $\forall q_i < B_j < q_i < B_i$
    $S \leftarrow \{q_i\} \cup \{q_j\}_j \in J$
3: compute discrete log of target
```

$$Norm_f(T) := Res(f, T) \leq A||T||^d_\infty eg(f)||f(x)||^{(d)}_\infty \tag{5}$$

If we assume $f(x)$ is monic, the norm computation for $T := t_0 + t_1 x + t_2 x^2 + ... + t_d x^d \in R_f$ with degree less than that of $f(x)$ is defined by Equation 5. Where $Res(f, T)$ is a number, meaning the residue of $f(x)$ on hole T. A hole refers to a value that can be reduced up to a certain limit. That norm, must be within the bound to the right where $||m(x)||_\infty$ refers to the absolute value of the largest coefficient m_i for some polynomial $m(x)$. Formally, we define $||f(x)||_\infty := max_{1 \leq i \deg deg(f)} |f_i|$ [7][8]. Figure provides an example for $deg(g) = 3$, max norm of $g = O(p^{1/2}$, and $k = 3$.

$$\begin{aligned}
p &= 908761003790427908077548955758380356675829026531247 \\
\mathbf{T} &= 314159265358979323846264338327950288419716939937510+ \\
&\quad 582097494459230781640628620899862803482534211706798x+ \\
&\quad 214808651328230664709384460955058223172535940812829x^2 \\
f &= 12x^6 - 24x^5 - 85x^4 + 70x^3 + 215x^2 + 96x + 12 \\
g &= x^3 + 870230335309004989831690 1x^2 + 870230335309004989831689 8x - 1
\end{aligned}$$

$$\text{Norm}_f(\mathbf{T})(\approx ||\mathbf{T}||^6_\infty ||f||^2_\infty) = \mathbf{1017}\text{bits} \sim p^6$$
$$\text{Norm}_g(\mathbf{T})(\approx ||\mathbf{T}||^3_\infty ||g||^2_\infty) = \mathbf{665}\text{bits} \sim p^4$$

Figure 4: Computing the norm [7]

In order to reduce the norm, we consider reducing the degree T and/or $||T(x)||_\infty$. Guillevic makes use of the Lemma which states that is we let the target be an element in F_{p^k}, then its log can be computed as $log(\eta T) mod(\ell)$ for η belonging to a properly selected subfield. F_p is selected as such subfield and the target is divided by the leading coefficient so that we have the discrete log of T = the discrete log of $T/t_d mod(\ell)$. Assuming the target is monic, we compute a lattice L (triangular matrix) using coefficients p, from $T(x)$, and the defining polynomial of F_{p^k}. Then, it applies the $Lenstra-Lenstra-Lovsz(LLL)$ redution which takes in a lattice and outputs a reduced basis of that lattice. In our case, the output is a short vector v composed from linear combinations of rows of L. Then, the log of the mapping for corresponding ring $\xi\rho(v)$ gives us the discrete log of the target mod ℓ. Figure shows an example for $\mathbb{F}_{p^3}$ and sample parameters.

$$f = x^6 + 19x^5 + 90x^4 + 95x^3 + 10x^2 - 13x + 1$$
$$\varphi = x^3 - yx^2 - (y+3)x - 1 \quad y \in \mathbb{Z}$$
$$\mathbf{T} = t_0 + t_1 x + x^2$$
$$\text{define } L = \begin{bmatrix} p & 0 & 0 & 0 & 0 & 0 \\ 0 & p & 0 & 0 & 0 & 0 \\ t_0 & t_1 & 1 & 0 & 0 & 0 \\ \varphi_0 & \varphi_1 & \varphi_2 & 1 & 0 & 0 \\ 0 & \varphi_0 & \varphi_1 & \varphi_2 & 1 & 0 \\ 0 & 0 & \varphi_0 & \varphi_1 & \varphi_2 & 1 \end{bmatrix} \begin{array}{l} \rho(p) = 0 \in \mathbb{F}_{p^k} \\ \\ T \\ \rho(\varphi) = 0 \in \mathbb{F}_{p^k} \\ \\ \\ \end{array}$$
LLL(L) outputs a short vector r, linear combination of L's rows.
$$r = \lambda_0 p + \lambda_1 px + \lambda_2 T + \lambda_3 \varphi + \lambda_4 x\varphi + \lambda_5 x^2 \varphi.$$
$$r = r_0 + \ldots + r_5 x^5, \ ||r_i||_\infty \le C \det(L)^{1/6} = O(p^{1/3})$$
$$\log \rho(r) = \log(T) \pmod{\ell}$$

Figure 5: Triangular matrix [7]

For an order of 508 bits, this work offered norm of T with respect to $f(x)$ of 340 bits for JSVL version this was 508, a norm with respect to $g(x)$ 508 bits and 847 for JSVL, and complexity $L[1/3, 1.26]$ in time $2^4 2$ and $L[1/3, 1.144]$ in time $2^4 8$. Overall, the improvements by Guillevic offer a very fast speed-up on the last step when n is small and used in conjunction with *Conjugation* or general Joux-Lercer (gJL) methods [7][8].

4.2 Joux

The initial work by Joux and Lercier in 2003 addressed large prime finite fields achieved $L_p[1/3,\ 1.44]$ for last step independent of the second and third steps. Later in 2006, Joux et. al (JSVL), addressed fields for non-primes p in the medium to large scale. Both use the special-q descent algorithm. In 2009, Lercier, Naccache and Thomé achieved a speed up in the relations step using an oracle. More recently, In 2013, Joux proposed an improvement to the NSF which applies to finite fields of small characteristic with a heuristic complexity of $L(\alpha = 1/4 + o(1))$ for $\mathbb{F}$ is of small characteristic p which require $q = p^\ell$ for $\ell \ge 2$ and p to be small with respect to q.

This was accomplished by introducing a new method for relations for elements that have small smoothness and a new method for the descent step which is meant to be a complement of the classical one and is a bilinear system descent. With respect to the smoothness and relations, we first consider the parameters: a finite field of small characteristic, $\mathbb{F}_{p^n}$, embed this field into one with form $\mathbb{F}_{q^{2k}}$ for $k \le q$. Figure 6 provides an overview of the parameters and mappings from the bivariate polynomial finite field.

$$\begin{tikzcd}
 & \mathbb{F}_q[X, X'] \arrow[dl, "X' \leftarrow X^q"'] \arrow[dr, "X' \leftarrow h_0(X)/h_1(X)"] & \\
\mathbb{F}_q[X] \arrow[dr, "X \leftarrow x"'] & & \mathbb{F}_q[X] \arrow[dl, "X \leftarrow x"] \\
 & \mathbb{F}_{q^{2k}} \arrow[d, dash, "\text{Degree } e \text{ extension}"] & \\
 & \mathbb{F}_{p^n} &
\end{tikzcd}$$

Figure 6: Joux parameters for small p [9]

Joux's approach includes includes polynomials of degree 2 and consider changes of form in Equation 6.

$$.Y = \frac{aX^2 + bX + c}{dX^2 + eX + c} \tag{6}$$

If the left side can be factored into polynomials with max degree of 2, then we have an equation which accounts for the extension of the basis. In this manner, once we have generated enough relations, we apply linear algebra and are able to recover extra logarithms. This is improved by placing polynomials of degree 2 into subsets that can be independently accessed as in Equation .

$$Y = \frac{a(X^2 + \alpha X) + b}{c(X^2 + \alpha X) + d} \tag{7}$$

It may be helpful to further extend the smoothness with higher degree polynomials (3+) contingent on parameters and the desired logs required to be computed.

5 SIGNIFICANCE

As noted earlier, commonly used public-key cryptographic systems rely on the hardness of solving the DLP problem. Some examples would be D-H and variants, the DSA, and ElGamal [15]. The ICM family of algorithms represent the classical state-of-the-art in solving the DLP offering a sub-exponential complexity (see Table 2) and in particular cases, polynomial time. However with the prospective of quantum machines and algorithms, DLP can be craked in polynomial time [12][13].

Algorithm	Complexity
Number Field Sieve for DLs case: p large with respect to q	$e^{ln(n)^{1/3} lnln(n)^{2/3} (\sqrt{64/9}^{1/3} + o(1))}$
Function Field Sieve case: q is large with respect to p	$e^{ln(n)^{1/3} lnln(n)^{2/3} (\sqrt{32/9}^{1/3} + o(1))}$
Number Field Sieve in High Degree case: p,q with similar dimension	$e^{ln(n)^{1/3} lnln(n)^{2/3} (c + o(1))}$ $c > 0$
Joux case: q large with respect to p	$e^{ln(n)^{1/4+\epsilon} lnln(n)^{3/4-\epsilon} (c + o(1))}$

Table 2: State-of-the-art in ICM [14]

Any notable improvements of these or new algorithms which offer an efficient solution to the DLP will break schemes based on this problem. If the solutions targets parameters to the point they must too large for practical purposes all such schemes will becomes obsolete.

6 CONCLUSION

We have explored the potential contemporary ICM-based algorithms which offer sub-exponential complexity for solving the DLP for specific types of groups and fields. In particular instances we have seen polynomial and quasi-polynomial cases. IC and quantum counterparts are active areas of research and significant work remains to be done in both. At the moment, the only way to safeguard against these attacks is select schemes and parameters for which the complexity is exponential and start shifting to post-quantum resistant alternatives.

REFERENCES

[1] David Adrian, Karthikeyan Bhargavan, Zakir Durumeric, Pierrick Gaudry, Matthew Green, J. Alex Halderman, Nadia Heninger, Drew Springall, Emmanuel Thomé, Luke Valenta, Benjamin VanderSloot, Eric Wustrow, Santiago Zanella-Béguelin, and Paul Zimmermann. Imperfect forward secrecy: How diffie-hellman fails in practice. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, pages 5–17, New York, NY, USA, 2015. ACM.

[2] Claus Diem. What is Index Calculus? Accessed: 2019-04-12.

[3] Claus Diem. Diem C. (2006) An Index Calculus Algorithm for Plane Curves of Small Degree. In: Hess F., Pauli S., Pohst M. (eds) Algorithmic Number Theory. ANTS 2006. Lecture Notes in Computer Science. 4096, 2006.

[4] Chris Fox. Attacking the Elliptic Curve Discrete Logarithm Problem. 2010.

[5] Pierrick Gaudry. The discrete logarithm problem. 1 - Generic algorithms. Accessed: 2019-05-01.

[6] Robert Granger, Thorsten Kleinjung, and Jens Zumbrägel. Diem C. (2006) An Index Calculus Algorithm for Plane Curves of Small Degree. In: Hess F., Pauli S., Pohst M. (eds) Algorithmic Number Theory. ANTS 2006. Lecture Notes in Computer Science. 4096, 2006.

[7] Aurore Guillevic. Individual Discrete Logarithm in GF(p^k) (last step of the Number Field Sieve algorithm). 2015.

[8] Aurore Guillevic. Computing Individual Discrete Logarithms Faster in GF(p^n) with the NFS-DL Algorithm. 2016.

[9] Antoine Joux. A new index calculus algorithm with complexity L(1/4 + o(1)) in small characteristic. 2013.

[10] Juan Klopper. Surjective homomorphisms in abstract algebra. Accessed: 2019-05-04.

[11] David R. Kohel. Complex multiplication and canonical lifts. 2007.

[12] A. D. MYASNIKOV and A. USHAKOV. QUANTUM ALGORITHM FOR THE DISCRETE LOGARITHM PROBLEM FOR MATRICES OVER FINITE GROUP RINGS. 2012.

[13] Peter Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J.Sci.Statist.Comput. 26 (1997) 1484*, 1995.

[14] Francesco Sovrano. A proposito di Crittografia a chiave asimmetrica e numeri primi: tecniche note e proposta di un nuovo test di primalitá euristico e deterministico, 2016.

[15] Stallings, William. *Cryptography and Network Security: Principles and Practice, 7th Edition.* Person, 2017.