

A few more index calculus algorithms for the Elliptic Curve Discrete Logarithm Problem

Gary McGuire* Daniela Mueller†

School of Mathematics and Statistics
University College Dublin
Ireland

Abstract

The introduction of summation polynomials for elliptic curves by Semaev has opened up new avenues of investigation in index calculus type algorithms for the elliptic curve discrete logarithm problem, and several recent papers have explored their use. Most papers use Gröbner basis computations at some point. We propose a faster algorithm to solve the Elliptic Curve Discrete Logarithm Problem using summation polynomials that does not involve Gröbner basis computations. Our algorithm also does not involve a linear algebra step, and makes use of a technique for fast evaluation of the summation polynomials, which we describe in the paper. We further propose an even faster algorithm that does not involve Gröbner basis computations, or a linear algebra step, or summation polynomials. We give a complexity analysis of our algorithms and provide extensive computational data.

Keywords

elliptic curves, ECDLP, index calculus, summation polynomials.

*Research supported by Science Foundation Ireland Grant 13/IA/1914.

†Research supported by a Postgraduate Government of Ireland Scholarship from the Irish Research Council.

1 Introduction

Let E be an elliptic curve over a finite field \mathbb{F}_q , where q is a prime power. In practice, q is often a prime number or a large power of 2. Let P and Q be points on E . The Elliptic Curve Discrete Logarithm Problem (ECDLP) is finding an integer l (if it exists) such that $Q = lP$. The integer l is called the discrete logarithm of Q to base P .

The ECDLP is a hard problem that underlies many cryptographic schemes and is thus an area of active research. The introduction of summation polynomials by [Sem04] has led to algorithms that resemble the index calculus algorithm of the DLP over finite fields. We outline how the algorithm works in general first.

Let G be a cyclic group with given generator g . We wish to find the discrete logarithm of a target element h to the base g . A sketch of the index calculus algorithm for G is the following.

1. **Factor Base step.** Define a subset $\mathcal{F} \subseteq G$, called the factor base.
2. **Relation step.** Collect linear relations involving factor base elements.
3. **Linear Algebra step.** Combine and solve relations using linear algebra.
4. **Solving step.** Use the results to find the discrete logarithm of the target element h .

When the group G is the multiplicative group of a finite field, typically the first three steps do not depend on the target element. Steps 1-3 will result in the logs of the factor base elements, and only in the final step will the target element be used, when its log will be calculated. This is different from typical index calculus algorithms for the ECDLP, where the relations in Step 2 depend on the target element, although the choice of factor base in Step 1 does not (see section 1.2). In the algorithms under discussion in this paper, the choice of factor base in Step 1 *does* depend on the target element.

It is a priori not clear how to choose the factor base, and a feature of all the algorithms under discussion in this paper is that the factor base is chosen randomly. One advantage of this is that the size of the factor base is very easy to change.

It is also not a priori clear how to find relations in Step 2. Summation polynomials enable a decomposition over the factor base in certain cases for elliptic curves, and we give their definition in section 1.1. Section 1.2 shows how this decomposition can be achieved for certain choices of factor base.

Most papers have focused on elliptic curves over an extension field \mathbb{F}_{q^n} , and use subfields in the algorithm, see for example [Gau09], [FGHR14], [FHJ⁺14], [JV13]. The case of elliptic curves over prime order fields seems to be much harder to tackle. Our algorithms in this paper are aimed at prime order fields, although they are valid for any finite field. In section 2, we give a brief overview of the different approaches to the prime field case using summation polynomials. A recent article [APS17] has shown how to simplify these algorithms to avoid the linear algebra step and reduce the number of Gröbner basis computations. We summarize their algorithm (as Algorithm 2.2) in section 2.

In section 3 we develop the algorithm in [APS17] to a new algorithm (Algorithm 3.1) which, unlike all other algorithms using summation polynomials, does not involve a Gröbner basis computation. This leads to a significant speedup over the other prime field algorithms.

In section 4 we then further develop our Algorithm 3.1 to Algorithm 4.1 which does not use summation polynomials at all, as well as not using Gröbner bases and not using a linear algebra step. This algorithm is fastest among all the algorithms discussed here, both in practice and in complexity.

Section 5 contains a complexity analysis of the algorithm in [APS17], as well as a complexity analysis of our two algorithms presented here. We also develop a method for fast evaluation of the summation polynomials. We will see that Algorithm 4.1 is best, followed by Algorithm 3.1 and last comes Algorithm 2.2. The algorithms have exponential complexity, which one would expect with a randomly chosen factor base. Our analysis shows that all these algorithms are worse than the well known square-root algorithms such as Pollard-Rho. Nevertheless, we claim that Algorithm 4.1 is the best index calculus algorithm for prime order fields at the present time. There seems to be no advantage to using Gröbner bases or summation polynomials with a random factor base, over prime order fields anyway.

Finally we present computational results for small primes in section 6, which happily agree with the complexity analysis.

1.1 Summation Polynomials

Definition 1.1: [Sem04] Let E be an elliptic curve over a field K . For $n \geq 2$, we define the summation polynomial $S_n = S_n(X_1, X_2, \dots, X_n)$ of E by the following property. Let $x_1, x_2, \dots, x_n \in \overline{K}$, then $S_n(x_1, x_2, \dots, x_n) = 0$ if and only if there exist $y_1, y_2, \dots, y_n \in \overline{K}$ such that $(x_i, y_i) \in E(\overline{K})$ and $(x_1, y_1) + (x_2, y_2) + \dots + (x_n, y_n) = \mathcal{O}$, where \mathcal{O} is the identity element of E .

Semaev showed in [Sem04] how to compute the summation polynomials for elliptic curves in Weierstrass form:

Theorem 1.2: Let E be an elliptic curve given by $Y^2 = X^3 + AX + B$ over a field K with characteristic $\neq 2, 3$. Then the summation polynomials are given by

$$S_2(X_1, X_2) = X_1 - X_2,$$

$$S_3(X_1, X_2, X_3) = (X_1 - X_2)^2 X_3^2 - 2((X_1 + X_2)(X_1 X_2 + A) + 2B)X_3 + ((X_1 X_2 - A)^2 - 4B(X_1 + X_2)),$$

$$S_n(X_1, \dots, X_n) = \text{Res}_X(S_{n-k}(X_1, \dots, X_{n-k-1}, X), S_{k+2}(X_{n-k}, \dots, X_n, X)) \text{ for } n \geq 4 \text{ and any } 1 \leq k \leq n - 3.$$

Furthermore, the polynomials S_n , $n \geq 3$, are symmetric, of degree 2^{n-2} in each variable, of total degree $(n-1)2^{n-2}$, and absolutely irreducible.

For more detail and for other characteristics, see [Sem04].

1.2 Point Decomposition with Summation Polynomials

The following is a more detailed version of the index calculus algorithm as normally used for elliptic curves, see [Gau09] for example. We include it for comparison with our algorithms developed in this paper.

Definition 1.3: (Index Calculus) Let G be a cyclic group of points on an elliptic curve defined over \mathbb{F}_q (here we use additive notation), let P be a generator of G , and Q another element in G whose discrete logarithm we wish to compute. The index calculus algorithm for G is the following.

1. **Factor Base step.** Define a subset $\mathcal{F} \subseteq G$, called the factor base.

2. **Relation step.** Collect relations that decompose over the factor base: Let $R = aP + bQ$ (a, b random integers), and try to write R as a sum of factor base elements, $R = P_1 + \dots + P_m$, with $P_1, \dots, P_m \in \mathcal{F}$. Store the relations in matrix and vector format.
3. **Linear Algebra step.** Perform linear algebra on the matrix-vector equation to get an equation of the form $\alpha P + \beta Q = 0$.
4. **Solving step.** If β is invertible modulo the group order r , then the discrete logarithm of Q is $-\alpha/\beta \pmod{r}$.

Let $\mathcal{F} = \{P_1, P_2, \dots, P_s\}$ be the factor base of points on E , where $s = |\mathcal{F}|$ is the size of the factor base. Let r_1, r_2 be random integers and let $R = r_1P + r_2Q$. In order to write $R = P_1 + \dots + P_m$, with $P_1, \dots, P_m \in \mathcal{F}$, we use the $(m+1)^{th}$ summation polynomial: writing $R = (x_R, y_R)$, we try to find a solution (x_1, \dots, x_m) of $S_{m+1}(X_1, \dots, X_m, x_R) = 0$ such that $\exists y_i$ with $(x_i, y_i) \in \mathcal{F}$, $1 \leq i \leq m$. Then $\exists \varepsilon_i = \pm 1$ such that $\varepsilon_1(x_1, y_1) + \dots + \varepsilon_m(x_m, y_m) \pm R = \mathcal{O}$.

Once we have found at least $s+1$ independent relations of this form, we can find $\log_P(Q)$ by solving the matrix equation

$$\begin{pmatrix} \varepsilon_{1,1} & \dots & \varepsilon_{1,s} \\ \varepsilon_{2,1} & \dots & \varepsilon_{2,s} \\ & \dots & \\ \varepsilon_{s+1,1} & \dots & \varepsilon_{s+1,s} \end{pmatrix} \begin{pmatrix} \log_P(P_1) \\ \log_P(P_2) \\ \dots \\ \log_P(P_s) \end{pmatrix} = \begin{pmatrix} r_{1,1} \\ r_{2,1} \\ \dots \\ r_{s+1,1} \end{pmatrix} + \begin{pmatrix} r_{1,2} \\ r_{2,2} \\ \dots \\ r_{s+1,2} \end{pmatrix} \log_P(Q)$$

where $\varepsilon_{i,j} \in \{0, 1, -1\}$, $1 \leq i \leq s+1$, $1 \leq j \leq s$.

Gaudry suggests in [Gau09] a way to solve $S_{n+1}(X_1, \dots, X_n, x_R) = 0$, if E is defined over $\mathbb{F}_{q^n} = \mathbb{F}_q[t]/f(t)$, q a prime power, f irreducible of degree n . He defines the factor base to be all points with x -coordinate in \mathbb{F}_q , $\mathcal{F} = \{(x, y) \in E(\mathbb{F}_{q^n}) : x \in \mathbb{F}_q\}$. Note that we only need to include one of $\{(x, y), (x, -y)\}$ in the factor base if we allow coefficients ± 1 in the decomposition of R . Now writing $S_{n+1}(X_1, \dots, X_n, x_R) = \sum_{i=0}^{n-1} \varphi_i(X_1, \dots, X_n)t^i$, we instead solve $\varphi_i(X_1, \dots, X_n) = 0$ over \mathbb{F}_q , $0 \leq i \leq n-1$, obtaining a polynomial system of n equations in n unknowns (Weil descent). We then solve this system with Gröbner basis techniques.

In [JV13] and [JV11], the authors use a variation of the F4 algorithm (originally due to [Fau99]) to speed up the Gröbner basis computation in the index calculus algorithm for ECDLP.

2 Factor base over prime fields

If the elliptic curve is defined over a prime field (\mathbb{F}_p for p a prime number) Semaev suggests in [Sem04] to define the factor base to be all points with "small" x -coordinate (taking the finite field elements to lie in the interval $[0, \dots, p-1]$ and treating them as integers in order to bound them). However, we don't know how to find these small points efficiently.

Petit et al. showed in [PKM16] how to define the factor base as points on the curve with x -coordinate a solution of the composition of some small-degree rational maps. The decompositions are then found by solving the polynomial system obtained from these rational maps and summation polynomials. Their approach seems to be the first working case for curves defined over prime fields, but it is only feasible for small parameters. (The largest field in their experiments is $\mathbb{F}_{4206593}$ and one Gröbner basis computation takes 4975.07 sec. Compare this with our results in section 6: the largest field in our experiments is $\mathbb{F}_{30951732491}$ and the total time to solve the ECDLP is 6850.10 sec.)

Amadori-Pintore-Sala [APS17] showed a different way of defining the factor base that enabled them to significantly reduce the number of polynomial systems that need to be solved, and also avoid the linear algebra step, leading to a huge improvement in the running time. We will explain their approach now and give a complexity analysis in section 5.

Step 1. Let s be the desired size of the factor base (we will show later how to select s). Compute random integers $a_1, \dots, a_s, b_1, \dots, b_s$. Then the factor base \mathcal{F} is all points $\{a_1P + b_1Q, \dots, a_sP + b_sQ\}$.

Step 2. Find a relation of the form $P_1 + \dots + P_m = \mathcal{O}$ with $P_i \in \mathcal{F}$.

Step 3. Substitute each P_i with the corresponding $a_iP + b_iQ$ and get the relation

$$\sum_{i=1}^m a_iP + \sum_{i=1}^m b_iQ = \mathcal{O}. \tag{1}$$

Then $Q = -\sum_{i=1}^m (a_i/b_i)P$ provided $\sum_{i=1}^m b_i$ is invertible modulo the order of E (if $\sum_{i=1}^m b_i$ is not invertible, start again). We have thus solved for the discrete logarithm of Q without doing a linear algebra step.

Remark 2.1: Note that the factor base is chosen randomly, as opposed to the methods mentioned in the first two paragraphs of this section. The algorithm may fail, and if so then it is run again and the re-run will involve a different choice of random factor base. This is in contrast to the other methods, where the factor base is clearly defined, and re-running the algorithm does not result in a different factor base.

In step 2, Amadori et al. propose the following system of polynomial equations to find relations. Let V be the set of x -coordinates of all the points in the factor base, i.e. $V = \{x \mid (x, y) \in \mathcal{F}\}$. Let $f(x) = \prod_{v \in V} (x - v)$. Then they solve (via Gröbner basis algorithms such as F4 or F5, see [Fau99] and [Fau02] respectively) the system

$$\begin{aligned} S_m(X_1, \dots, X_m) &= 0 \\ f(X_1) &= 0 \\ \dots \\ f(X_m) &= 0. \end{aligned} \tag{2}$$

Hence, they only consider solutions to $S_m(X_1, \dots, X_m) = 0$ of the form $(x_1, \dots, x_m) \in V^m$, i.e. corresponding to points in the factor base.

Since f has degree s , which is the size of the factor base and could be quite large, the resolution of the system could be slow. So they propose instead using m different polynomials, by partitioning the factor base into m different factor bases \mathcal{F}_i of more or less equal size $\frac{s}{m}$. V is partitioned into m sets V_i accordingly, giving m polynomials $f_i(x) = \prod_{v \in V_i} (x - v)$. They then solve the system

$$\begin{aligned} S_m(X_1, \dots, X_m) &= 0 \\ f_1(X_1) &= 0 \\ \dots \\ f_m(X_m) &= 0. \end{aligned} \tag{3}$$

Now each of the f_i only has degree $\frac{s}{m}$ approximately, and therefore the Gröbner basis computation is less expensive. However, this also reduces the probability of finding a solution in the factor base. We shall give more details about this in section 5 on complexity analysis.

For completeness, we give the full algorithm from [APS17] with this approach:

Algorithm 2.2: [APS17]

Input: elliptic curve E over \mathbb{F}_p , points P and Q on E , integers m, s , summation polynomial S_m

Output: $\log_P(Q)$

1. Let s be the size of the factor base. Compute random integers $a_1, \dots, a_s, b_1, \dots, b_s$. The factor base \mathcal{F} is all points $\{a_1P + b_1Q, \dots, a_sP + b_sQ\}$. The corresponding set containing only the x -coordinates of the factor base points is $V = \{x | (x, y) \in \mathcal{F}\}$. Partition this set into m sets V_i of approximately equal size. Let $f_i(x) = \prod_{v \in V_i} (x - v)$, $i = 1, \dots, m$.
2. Using a Gröbner basis algorithm like F4 or F5, solve system (3). If there is no solution, go back to step 1.
3. If $\{x_1, \dots, x_m\}$ is a solution to the above system, then each $x_i \in V_i$ and there exist y_i such that $(x_1, y_1) + \dots + (x_m, y_m) = \mathcal{O}$ where either (x_i, y_i) or $-(x_i, y_i)$ are in \mathcal{F} . Substituting each $\pm(x_i, y_i)$ with the corresponding $\pm(a_iP + b_iQ)$, we get a relation of the form $\sum_{i=1}^m \pm a_iP + \sum_{i=1}^m \pm b_iQ = \mathcal{O}$ and can solve for the discrete logarithm of Q , provided $\sum_{i=1}^m \pm b_i$ is invertible modulo the order of E .

3 Avoiding Gröbner basis computations and Linear Algebra step

While systems (2) and (3) are a way of algebraically describing that the solutions to S_m lie in the factor base, it seems to us that there should be a better way to solve this problem than feeding the polynomial system into a Gröbner basis algorithm. This approach essentially treats the polynomials f_i (or f) as input polynomials to find their common roots with S_m even though we already know their complete factorisation.

We therefore propose the following alternative to Algorithm 2.2, which does not use a Gröbner basis algorithm.

Algorithm 3.1:

Input: elliptic curve E over \mathbb{F}_p , points P and Q on E , integers m, s , summation polynomial S_m

Output: $\log_P(Q)$

1. Let s be the size of the factor base. Compute random integers $a_1, \dots, a_s, b_1, \dots, b_s$. The factor base \mathcal{F} consists of all points $\{a_1P + b_1Q, \dots, a_sP + b_sQ\}$. The corresponding set containing only the x -coordinates of the factor base points is denoted $V = \{x | (x, y) \in \mathcal{F}\}$.
2. Choose $\{x_1, \dots, x_m\}$ a multiset of size m with each $x_i \in V$ and check if $S_m(x_1, \dots, x_m) = 0$. If not, repeat with another multiset. If S_m is non-zero for all multisets of size m , go back to step 1.
3. If $S_m(x_1, \dots, x_m) = 0$ for some $\{x_1, \dots, x_m\}$, then there exist y_i such that $(x_1, y_1) + \dots + (x_m, y_m) = \mathcal{O}$ where either (x_i, y_i) or $-(x_i, y_i)$ are in \mathcal{F} . Substituting each $\pm(x_i, y_i)$ with the corresponding $\pm(a_iP + b_iQ)$, we get (as in (1)) a relation of the form $\sum_{i=1}^m \pm a_iP + \sum_{i=1}^m \pm b_iQ = \mathcal{O}$ and can solve for the discrete logarithm of Q , provided $\sum_{i=1}^m \pm b_i$ is invertible modulo the order of E .

We will show later in Lemma 5.10 how to efficiently evaluate the summation polynomials S_m in step 2.

Remark 3.2: In step 2, one could alternatively use the multi-point multivariate polynomial evaluation algorithm by Kedlaya and Umans, see [KU11]. This does not immediately appear to give an improvement in the complexity.

Remark 3.3: In step 2, we can alternatively choose a multiset of m points $\{P_1, \dots, P_m\}$ from the factor base, and sum those points to see if they give the point at infinity. This avoids using summation polynomials, and is in fact faster in theory and in practice (see section 5 and section 6). We omit the details for this algorithm. We refine this idea in the next section, and obtain an even faster algorithm.

4 Avoiding Summation Polynomials and Gröbner bases and Linear Algebra step

The following algorithm is a variation of our Algorithm 3.1. Here, we choose a multiset of $m - 1$ points from the factor base, and check if the sum of those points lies in the factor base:

Algorithm 4.1:

Input: elliptic curve E over \mathbb{F}_p , points P and Q on E , integers m, s

Output: $\log_p(Q)$

1. Let s be the size of the factor base. Compute random integers $a_1, \dots, a_s, b_1, \dots, b_s$. The factor base \mathcal{F} is all points $\{a_1P + b_1Q, \dots, a_sP + b_sQ\}$.
2. Choose $\{P_1, \dots, P_{m-1}\}$ a multiset of size $m - 1$ with each $P_i \in \mathcal{F}$. Choose $v \in \mathbb{F}_2^{m-1}$, and let $P_v = (-1)^{v_1}P_1 + \dots + (-1)^{v_{m-1}}P_{m-1}$. Check if $P_v \in \mathcal{F}$.
If $P_v \notin \mathcal{F}$ for all $v \in \mathbb{F}_2^{m-1}$, repeat with another multiset.
If there is no solution for all multisets of size $m - 1$, go back to step 1.
3. If $P_v \in \mathcal{F}$ for some v then let $P_m = -P_v$ and we get the relation $(-1)^{v_1}P_1 + \dots + (-1)^{v_{m-1}}P_{m-1} + P_m = \mathcal{O}$. Substituting each $\pm P_i$ with the corresponding $\pm(a_iP + b_iQ)$, we get (as in (1)) a relation of the form $\sum_{i=1}^m \pm a_iP + \sum_{i=1}^m \pm b_iQ = \mathcal{O}$ and can solve for the discrete logarithm of Q , provided $\sum_{i=1}^m \pm b_i$ is invertible modulo the order of E .

We will provide a complexity analysis of all algorithms in section 5.

Remark 4.2: The motivation for the three given algorithms was the ECDLP over prime fields. However, none of the algorithms require the field to be of prime order. They all work for any finite field.

Remark 4.3: Algorithm 2.2 and Algorithm 3.1 use summation polynomials, and therefore the input value of m must be ≤ 8 because the largest summation polynomial that has been computed so far is S_8 as far as we are aware (see [FHJ⁺14]). Algorithm 4.1 does not suffer from this problem, and larger values of m can readily be used.

5 Complexity Analysis

Table 1 summarises the complexity of operations in \mathbb{F}_p obtained from [MvOV96] and of operations on an elliptic curve over \mathbb{F}_p from [Sil09]. Please note that there may be faster algorithms for these operations that lead to better complexities, but these improvements will only affect the log factors of our algorithm complexities and will not provide a significant improvement to the overall complexity.

Operation	Bit complexity
Addition	$O(\log p)$
Multiplication	$O(\log^2 p)$
Inversion	$O(\log^2 p)$
Point addition	$O(\log^2 p)$
Point multiplication	$O(\log^3 p)$
Searching a sequence of length s	$O(s)$

Table 1: Bit complexity of basic operations in \mathbb{F}_p and on an elliptic curve over \mathbb{F}_p (not best possible).

Lemma 5.1: The probability of obtaining a relation of length m in \mathcal{F} is $\frac{2^{m-1}s^m}{p \cdot m!}$, where $s = |\mathcal{F}|$ and $m \ll s$.

Proof: The number of ways of choosing m elements from a set of size s , allowing repetitions, is $\binom{s+m-1}{m} \approx \frac{s^m}{m!}$ for $m \ll s$. Relations can be of the form $P_1 \pm \dots \pm P_m = \mathcal{O}$ with $P_i \in \mathcal{F}$, so we get $\approx \frac{2^{m-1}s^m}{m!}$ possibilities. The number of points on the curve is approximately p . \square

Lemma 5.2: The probability of obtaining a relation of length m with each point coming from a different partition of the factor base of size $\frac{s}{m}$ is $\frac{2^{m-1}s^m}{p \cdot m^m}$.

Proof: There are $\frac{s}{m}$ ways of choosing each point in the relation giving $2^{m-1} \left(\frac{s}{m}\right)^m$ possibilities for relations of the form $P_1 \pm \dots \pm P_m = \mathcal{O}$ with each P_i coming from the factor base partition \mathcal{F}_i of size $\frac{s}{m}$. \square

Lemma 5.3: The complexity of computing a factor base of size s is $O(s \log^3 p)$.

Proof: See Table 1. We have to do $2s$ point multiplications of $O(\log^3 p)$ and s point additions of $O(\log^2 p)$. \square

Remark 5.4: There may be faster ways of computing the points $aP + bQ$ at the expense of more memory.

We would like the probability of finding a relation in the factor base to be close to 1, i.e. in the case of Lemma 5.2, we want $\frac{2^{m-1}s^m}{m^m} \approx p$, so we should choose the factor base size s accordingly. However, the authors of [APS17] propose $s = p^{1/m}$ as was chosen in other papers, e.g. [Gau09]. With this choice we will have to run (steps 1 and 2 of) Algorithm 2.2 an expected number of $\frac{m^m}{2^{m-1}}$ times. Therefore, even though we only require the computation of one Gröbner basis each time we choose a factor base, we will in general have computed several factor bases before finding a relation, and thus we require several Gröbner basis computations in the overall discrete logarithm algorithm (Algorithm 2.2). If one were to increase s in order to reduce the number of Gröbner basis computations needed, then the polynomial degrees are increasing accordingly, making each Gröbner basis computation slower. It may therefore be better to keep $s = p^{1/m}$, but it should be noted that this choice requires several Gröbner basis computations and not only one as is claimed in [APS17].

The following theorem gives the complexity of *one* Gröbner basis computation.

Theorem 5.5: The complexity of solving the system (3) is $O(p^{\omega-\omega/m})$ for $s = p^{1/m}$ and $m \ll s$, where $\omega \leq 3$ is the linear algebra constant.

Proof: The complexity of a Gröbner basis computation can be approximated with $\binom{N+D_{reg}-1}{D_{reg}}^\omega$, (see [BFS04]) where N is the number of variables of the input polynomials, D_{reg} is the degree of regularity, and ω is the linear algebra constant. If the number of polynomials is $N + 1$, then D_{reg} can be calculated using the formula $D_{reg} = \sum_{i=1}^{N+1} \frac{d_i-1}{2}$ for N large (see [BFS04] and also pages 80 and 98 of [Bar04]) where d_i is the degree of the i^{th} input polynomial. Here $N = m$, S_m has total degree $(m-1) \cdot 2^{m-2}$ (Theorem 1.2) and each f_i has degree $\approx \frac{s}{m}$. So we get

$$D_{reg} = \frac{(m-1) \cdot 2^{m-2} - 1 + m(\frac{s}{m} - 1)}{2} = (m-1) \cdot 2^{m-3} + \frac{s - m - 1}{2}.$$

Thus,

$$\begin{aligned}
& \binom{N + D_{reg} - 1}{D_{reg}} = \binom{N + D_{reg} - 1}{N - 1} \\
&= \binom{m + (m - 1) \cdot 2^{m-3} + \frac{s-m-1}{2} - 1}{m - 1} \\
&= \frac{(\frac{s}{2} + (m - 1) \cdot 2^{m-3} + \frac{m}{2} - \frac{3}{2})!}{(m - 1)! (\frac{s}{2} + (m - 1) \cdot 2^{m-3} - \frac{m}{2} - \frac{1}{2})!} \\
&= \frac{(\frac{s}{2} + (m - 1) \cdot 2^{m-3} + \frac{m}{2} - \frac{3}{2}) \dots (\frac{s}{2} + (m - 1) \cdot 2^{m-3} - \frac{m}{2} + \frac{1}{2})}{(m - 1)!}.
\end{aligned}$$

There are $m - 1$ factors in the numerator, each dominated by $\frac{s}{2}$. So we approximate $\binom{N+D_{reg}-1}{D_{reg}} \approx \frac{(\frac{s}{2})^{m-1}}{(m-1)!}$. Thus,

$$\binom{N + D_{reg} - 1}{D_{reg}}^\omega \approx \frac{(\frac{s}{2})^{\omega \cdot m - \omega}}{(m - 1)!^\omega} = \frac{p^{\omega - \omega/m}}{(2^{m-1}(m - 1)!)^\omega}. \quad \square$$

Remark 5.6: It may be possible that the degree of regularity of system (3) is smaller than the one we calculate here using a rather generic formula that holds for large N . We do not investigate this possibility in this paper.

Corollary 5.7: The complexity of Algorithm 2.2 is

$$\frac{m^m}{2^{m-1}} (O(p^{\omega - \omega/m}) + O(p^{1/m} \log^3 p)) \approx O(p^{\omega - \omega/m})$$

for $s = p^{1/m}$ and $m \ll s$, where $\omega \leq 3$ is the linear algebra constant.

Remark 5.8: It is reasonable to assume that m is small when using summation polynomials, since the largest summation polynomial that has been computed so far is S_8 (see [FHJ⁺14]).

Remark 5.9: With $\omega = 3$ and $m = 3$ the complexity of Algorithm 2.2 is $O(p^2)$. This roughly agrees with our experiments.

Next we will discuss a method for evaluating a summation polynomial S_m which is faster than a straightforward brute force evaluation. For $m = 3$ there is no difference, so in our experiments with S_3 we did not need to implement this method. For experiments with $m \geq 4$ this method is worth using.

Lemma 5.10: Evaluating S_m at a point (x_1, \dots, x_m) can be done in $O(\log^2 p)$ steps for $m \ll p$.

Proof: It follows from the statement of Theorem 1.2 that S_3 can be evaluated using at most 8 multiplications and 11 additions, and thus has complexity $8O(\log^2 p) + 11O(\log p)$.

For larger m , we make heavy use of the fact that

$$\text{Res}_X(f(a, X), g(a, X)) = \text{Res}_X(f, g)(a)$$

whenever the leading coefficients are non-zero, see [Duz]. Write $S_3(x_1, x_2, X) = a_2X^2 + a_1X + a_0$ and $S_3(x_3, x_4, X) = b_2X^2 + b_1X + b_0$. By Theorem 1.2,

$$\begin{aligned} S_4(x_1, x_2, x_3, x_4) &= \text{Res}_X(S_3(x_1, x_2, X), S_3(x_3, x_4, X)) \\ &= \det \begin{pmatrix} a_2 & a_1 & a_0 & 0 \\ 0 & a_2 & a_1 & a_0 \\ b_2 & b_1 & b_0 & 0 \\ 0 & b_2 & b_1 & b_0 \end{pmatrix} \\ &= a_2(b_0(a_2b_0 - 2a_0b_2 - a_1b_1) + a_0b_1^2) \\ &\quad + b_2(a_1(a_1b_0 - a_0b_1) + a_0^2b_2). \end{aligned}$$

Using our optimisation for S_3 to evaluate the a_i and b_i , we can evaluate S_4 with at most 21 multiplications and 24 additions, and thus with complexity $21O(\log^2 p) + 24O(\log p)$.

For $m \geq 5$, again by Theorem 1.2,

$$\begin{aligned} S_m(x_1, \dots, x_m) &= \text{Res}_X(S_3(x_1, x_2, X), S_{m-1}(x_3, \dots, x_m, X)) \\ &= \det \begin{pmatrix} a_2 & a_1 & a_0 & 0 & 0 & \dots & 0 \\ 0 & a_2 & a_1 & a_0 & 0 & \dots & 0 \\ \dots & & & & & & \\ b_{2^{m-3}} & b_{2^{m-3}-1} & \dots & b_0 & 0 & & \\ 0 & b_{2^{m-3}} & \dots & b_1 & b_0 & & \end{pmatrix} \\ &= a_2(a_2C_{12} - a_1C_{13} + a_0(C_{14} - C_{23})) \\ &\quad + a_1(a_1C_{23} - a_0C_{24}) + a_0^2C_{34} \end{aligned}$$

where $S_{m-1}(x_3, \dots, x_m, X) = b_{2^{m-3}}X^{2^{m-3}} + b_{2^{m-3}-1}X^{2^{m-3}-1} + \dots + b_0$ and $S_3(x_1, x_2, X) = a_2X^2 + a_1X + a_0$ and C_{ij} is the determinant of the above matrix (Sylvester matrix of S_3 and S_{m-1}) with the i^{th} and j^{th} column removed and the first and second row removed.

We can evaluate this expression with at most nine multiplications, six additions, and six determinant computations of 2^{m-3} by 2^{m-3} matrices. Also,

we can evaluate the b_i by $m - 4$ recursive calls to this algorithm ($m - 4$ calls because once we reach S_4 , we can evaluate it directly using the expression earlier in the proof). Thus, we can evaluate S_m using a total of $8(m - 4) + 21 + 9(m - 4)$ multiplications, $11(m - 4) + 24 + 6(m - 4)$ additions, and six determinant computations of matrices of size 2^{m-3} by 2^{m-3} , six of size 2^{m-4} by 2^{m-4} , \dots , and six of size 2^2 by 2^2 .

We assume that the complexity of computing the determinant of an n by n matrix is $O(n^3)$ (see [vzGG13], although this can be improved, see e.g. [KV05]). This gives a total complexity of $(17m - 47)O(\log^2 p) + (17m - 44)O(\log p) + 6(O(2^{3(m-3)}) + O(2^{3(m-4)}) + \dots + O(2^{3 \cdot 2}))$.

So far, the proof holds for any m . Finally, we assume $m \ll p$ and we get a complexity of $O(\log^2 p)$ for evaluating S_m . \square

Remark 5.11: See math.ieuclidp/ecldlp.html for a Magma implementation of the method described in this proof. In particular for $m \geq 4$ it is significantly faster than first computing S_m and then evaluating it with Magma's in built evaluation function. This method even allows us to evaluate S_9 and S_{10} , although nobody has actually computed them yet as far as we know.

Theorem 5.12: The complexity of Algorithm 3.1 is $O(p \log^2 p)$ for $m \ll s$ and $m \ll p$ and $\frac{s^{m-1}}{m!} \geq \log p$.

Proof: As noted in the proof of Lemma 5.1, there are $\frac{s^m}{m!}$ ways of choosing m elements from the set \mathcal{F} of size s , allowing repetitions, for small m . By Lemma 5.10, evaluating S_m is $O(\log^2 p)$, giving a total of $\frac{s^m}{m!} O(\log^2 p)$ in the worst case. Now by Lemma 5.1, we need an expected number of $\frac{p \cdot m!}{2^{m-1} s^m}$ trials, giving $\frac{p \cdot m!}{2^{m-1} s^m} (\frac{s^m}{m!} O(\log^2 p) + O(s \log^3 p)) \approx O(p \log^2 p)$ when $\frac{s^{m-1}}{m!} \geq \log p$. \square

Remark 5.13: Replacing the evaluation of S_m by adding m points together as in Remark 3.3, gives a complexity of $(m - 1)O(p \log^2 p) \approx O(p \log^2 p)$ for small m .

Theorem 5.14: The complexity of Algorithm 4.1 is $O(p)$ for $m \ll s$ and $s \geq (m - 2) \log^2 p$.

Proof: There are $2^{m-1} \frac{s^{m-1}}{(m-1)!}$ different ways of forming the sum $\pm P_1 \pm \dots \pm P_{m-1}$, with $P_i \in \mathcal{F}$, allowing repetitions, for small m and $s = |\mathcal{F}|$. Let $P_m = \pm P_1 \pm \dots \pm P_{m-1}$. The complexity of each sum is $(m - 2)O(\log^2 p)$. For each combination, we check if P_m is in \mathcal{F} , which is $O(s)$. If the sum is in \mathcal{F} , we get a relation of the form $\pm P_1 \pm \dots \pm P_{m-1} - P_m = \mathcal{O}$. So we still get a relation with probability as in Lemma 5.1, so the complexity of this

algorithm is $\frac{p \cdot m!}{2^{m-1} s^m} (2^{m-1} \frac{s^{m-1}}{(m-1)!} ((m-2)O(\log^2 p) + O(s)) + O(s \log^3 p))$. If $s \geq (m-2) \log^2 p$ then this is $\frac{p \cdot m}{s} O(s) \approx O(p)$ for small m . \square

6 Experimental Results

We ran experiments in Magma V2.21-6 [BCP97] with $m = 3$ and $m = 4$ to time

- the Algorithm 2.2 first given in [APS17] (called Type Gröbner),
 - our Algorithm 3.1 (called Type Eval S_m),
 - our Algorithm 3.1 using the evaluation for S_m described in Lemma 5.10 (called Type Eval $S_m \dagger$),
 - our Algorithm 4.1 (Type Sum in \mathcal{F}),
- all with the same parameters.

See math.ie/ecdlp/ecdlp.html for a Magma implementation of all three algorithms.

We have used a factor base size of $s = \lceil p^{1/m} \rceil$ in line with [APS17]. The results are summarised in Tables 2 and 3, where

- $T_{\mathcal{F}}(s)$ denotes the time in seconds it took to compute the factor bases (step 1 of the algorithms),
- $T_{solve}(s)$ denotes the time in seconds it took to find a solution (step 2).
- We did not include the timings for step 3 as they are negligible.
- The column “trials” shows the number of times we had to compute a factor base before finding a solution (i.e. how many times step 1 and 2 were done.)

The experiments in Table 2 clearly show that for those field sizes our Algorithm 4.1 is the fastest, which agrees with the complexity analysis. So, our algorithms are faster than Algorithm 2.2 first given in [APS17], and they show that their algorithm is faster than the one in [PKM16].

As we remarked in section 5, the experiments also show that we need to run several Gröbner basis computations in order to solve the discrete logarithm problem using the approach reported in [APS17] (Algorithm 2.2).

As expected by Lemma 5.1 and Lemma 5.2, when $m = 4$, more trials are needed before a relation in the factor base is found, suggesting that the size of the factor base is too small. In fact, the complexity of our Algorithm 3.1 and Algorithm 4.1 grows with m so it may be an advantage to keep to $m = 3$ and increase the size of the factor base.

Table 3 shows that using the evaluation function for S_m described in Lemma 5.10

significantly speeds up Algorithm 3.1 for $m = 4$. It even outperforms Algorithm 4.1 for small values of p .

Table 4 shows experimental results for $s = (m! \cdot p \cdot 2^{1-m})^{1/m}$ using our Algorithm 4.1, over prime fields of bigger size. (This choice of s gives a probability of obtaining a relation in the factor base of approximately 1 according to Lemma 5.1.) The other algorithms could not finish in reasonable time with p this size. They again show that $m = 3$ is faster than $m = 4$. For $m = 2$, step 2 of the algorithm ($T_{solve}(s)$) is faster than for $m = 3$, but building the factor base ($T_{\mathcal{F}}(s)$) takes more time, so overall $m = 2$ is slower than $m = 3$. So it seems that for Algorithm 4.1, $m = 3$ is the best choice.

Both versions of our algorithm require much less memory than the Gröbner basis approach.

It is also worth noting that our algorithms are embarrassingly parallel, while using a Gröbner basis to solve system (3) is much harder to parallelize.

Type	p	m	s	trials	$T_{\mathcal{F}}(s)$	$T_{solve}(s)$	Mem(MB)
Gröbner	55673	3	39	5.00	6.80E-3	0.31	33.62
Eval S_m	55673	3	39	1.88	2.38E-3	0.06	33.62
Sum in \mathcal{F}	55673	3	39	1.86	2.53E-3	0.05	33.62
Gröbner	719267	3	90	7.98	0.03	16.94	67.24
Eval S_m	719267	3	90	2.03	9.32E-3	0.86	33.56
Sum in \mathcal{F}	719267	3	90	1.95	7.79E-3	0.34	33.62
Gröbner	6443737	3	187	6.57	0.07	290.50	917.50
Eval S_m	6443737	3	187	2.00	0.02	7.30	33.62
Sum in \mathcal{F}	6443737	3	187	2.05	0.02	1.97	33.62
*Gröbner	30056657	3	311	7.88	0.14	96384.00	4555.01
Eval S_m	30056657	3	311	2.33	0.04	42.62	33.56
Sum in \mathcal{F}	30056657	3	311	2.16	0.05	14.08	33.62

Table 2: Average values on 100 experiments for each p (the one marked with * was only run 8 times). Type Gröbner denotes Algorithm 2.2, Eval S_m denotes Algorithm 3.1, Sum in \mathcal{F} denotes Algorithm 4.1

Type	p	m	s	trials	$T_{\mathcal{F}}(s)$	$T_{solve}(s)$	Mem(MB)
Gröbner	55673	4	16	33.89	0.02	0.50	33.62
Eval S_m	55673	4	16	3.35	2.84E-3	0.79	33.62
Eval $S_m \dagger$	55673	4	16	3.14	2.02E-3	0.18	33.62
Sum in \mathcal{F}	55673	4	16	2.76	1.93E-3	0.20	33.62
Gröbner	719267	4	30	35.47	0.06	19.19	33.62
Eval S_m	719267	4	30	3.42	4.86E-3	9.50	33.62
Eval $S_m \dagger$	719267	4	30	2.89	3.72E-3	1.22	33.62
Sum in \mathcal{F}	719267	4	30	3.23	3.67E-3	1.54	33.56
Gröbner	6443737	4	51	32.92	0.09	412.00	169.34
Eval S_m	6443737	4	51	3.17	8.19E-3	66.25	33.62
Eval $S_m \dagger$	6443737	4	51	3.72	8.70E-3	11.08	33.62
Sum in \mathcal{F}	6443737	4	51	3.69	9.00E-3	9.03	33.62
Eval S_m	30056657	4	75	3.06	0.01	288.50	33.56
Eval $S_m \dagger$	30056657	4	75	3.08	0.01	38.75	33.62
Sum in \mathcal{F}	30056657	4	75	3.42	0.02	37.00	33.62

Table 3: Average values on 100 experiments for each p . Type Gröbner denotes Algorithm 2.2, Eval S_m denotes Algorithm 3.1, Eval $S_m \dagger$ denotes Algorithm 3.1 with Lemma 5.10, Sum in \mathcal{F} denotes Algorithm 4.1

Type	p	m	s	trials	$T_{\mathcal{F}}(s)$	$T_{solve}(s)$	Mem(MB)
Sum in \mathcal{F}	55673	2	236	2.80	0.03	3.67E-3	33.56
Sum in \mathcal{F}	55673	3	44	1.48	2.27E-3	0.04	33.62
Sum in \mathcal{F}	55673	4	21	1.47	1.02E-3	0.17	33.62
Sum in \mathcal{F}	719267	2	849	2.43	0.11	0.05	33.62
Sum in \mathcal{F}	719267	3	103	1.62	6.34E-3	0.30	33.62
Sum in \mathcal{F}	719267	4	39	1.52	2.76E-3	1.05	33.56
Sum in \mathcal{F}	6443737	2	2539	2.45	0.62	0.51	33.62
Sum in \mathcal{F}	6443737	3	214	1.56	0.02	1.68	33.62
Sum in \mathcal{F}	6443737	4	67	1.65	3.93E-3	6.71	33.62
Sum in \mathcal{F}	30056657	2	5483	2.59	5.73	7.40	33.56
Sum in \mathcal{F}	30056657	3	356	1.54	0.04	11.58	33.62
Sum in \mathcal{F}	30056657	4	98	1.39	7.87E-3	24.81	33.62
Sum in \mathcal{F}	75426619	2	8685	2.77	14.94	20.72	33.56
Sum in \mathcal{F}	75426619	3	484	1.46	0.05	25.59	33.56
Sum in \mathcal{F}	75426619	4	123	1.84	0.01	90.12	33.56
Sum in \mathcal{F}	161532773	3	624	1.73	0.09	66.25	33.62
Sum in \mathcal{F}	4911016471	3	1946	1.69	0.91	1126.00	33.56
Sum in \mathcal{F}	30951732491	3	3595	1.67	2.10	6848.00	33.62

Table 4: Average values on 100 experiments for each p using Algorithm 4.1

References

- [APS17] Alessandro Amadori, Federico Pintore, and Massimiliano Sala. On the discrete logarithm problem for prime-field elliptic curves. Cryptology ePrint Archive, Report 2017/609, 2017. <https://eprint.iacr.org/2017/609>.
- [Bar04] Magali T. Bardet. *Étude des systèmes algébriques surdéterminés. Applications aux codes correcteurs et à la cryptographie*. PhD thesis, Université Paris 6, 2004.
- [BCP97] Wieb Bosma, John Cannon, and Catherine Playoust. The Magma algebra system. I. The user language. *J. Symbolic Comput.*, 24(3-4):235–265, 1997. Computational algebra and number theory (London, 1993).
- [BFS04] Magali T. Bardet, Jean Charles Faugère, and Bruno Salvy. On the complexity of Gröbner basis computation of semi-regular overde-

- terminated. In *Proceedings of the 2004 International Conference on Polynomial System Solving*, ICPSS '04, pages 71–75, 2004.
- [Duz] S. V. Duzhin. Lecture 10 the sylvester resultant. http://www.pdmi.ras.ru/~lowdimma/topics_nth/Resultants.pdf.
- [Fau99] Jean Charles Faugère. A new efficient algorithm for computing Gröbner bases (F4). *Journal of Pure and Applied Algebra*, 139(1):61 – 88, 1999.
- [Fau02] Jean Charles Faugère. A new efficient algorithm for computing Gröbner bases without reduction to zero (F5). In *Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation*, ISSAC '02, pages 75–83, New York, NY, USA, 2002. ACM.
- [FGHR14] Jean-Charles Faugère, Pierrick Gaudry, Louise Huot, and Guénaél Renault. Using symmetries in the index calculus for elliptic curves discrete logarithm. *Journal of Cryptology*, 27(4):595–635, 2014.
- [FHJ+14] Jean-Charles Faugère, Louise Huot, Antoine Joux, Guénaél Renault, and Vanessa Vitse. Symmetrized summation polynomials: Using small order torsion points to speed up elliptic curve index calculus. *Advances in Cryptology – EUROCRYPT 2014 Lecture Notes in Computer Science*, 8441:40–57, 2014.
- [Gau09] Pierrick Gaudry. Index calculus for abelian varieties of small dimension and the elliptic curve discrete logarithm problem. *Journal of Symbolic Computation*, 44(12):1690–1702, 2009.
- [JV11] Antoine Joux and Vanessa Vitse. A variant of the f4 algorithm. In *Proceedings of the 11th International Conference on Topics in Cryptology: CT-RSA 2011*, CT-RSA'11, pages 356–375, Berlin, Heidelberg, 2011. Springer-Verlag.
- [JV13] Antoine Joux and Vanessa Vitse. Elliptic curve discrete logarithm problem over small degree extension fields. *J. Cryptol.*, 26(1):119–143, 2013.
- [KU11] Kiran S. Kedlaya and Christopher Umans. Fast polynomial factorization and modular composition. *SIAM J. Comput.*, 40(6):1767–1802, December 2011.

- [KV05] Erich Kaltofen and Gilles Villard. On the complexity of computing determinants. *Computational Complexity*, 13(3-4):91–130, feb 2005.
- [MvOV96] A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.
- [PKM16] Christophe Petit, Michiel Kisters, and Ange Messeng. Algebraic approaches for the elliptic curve discrete logarithm problem over prime fields. *Public-Key Cryptography*, 9615:3–18, 2016. https://doi.org/10.1007/978-3-662-49387-8_1.
- [Sem04] Igor Semaev. Summation polynomials and the discrete logarithm problem on elliptic curves. Cryptology ePrint Archive, Report 2004/031, 2004. <http://eprint.iacr.org/2004/031>.
- [Sil09] Joseph H. Silverman. *The Arithmetic of Elliptic Curves, 2nd Edition*. Graduate Texts in Mathematics, Springer, 2009.
- [vzGG13] Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra, Third Edition*. Cambridge University Press, 2013.