# A New Index Calculus Algorithm for the Elliptic Curve Discrete Logarithm Problem and Summation Polynomial Evaluation

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. We propose an index calculus algorithm to solve the Elliptic Curve Discrete Logarithm Problem that makes use of a technique for fast evaluation of the summation polynomials, and unlike all other algorithms using summation polynomials, does not involve a Gröbner basis computation. We further propose another algorithm that does not involve Gröbner basis computations or summation polynomials. We give a complexity estimate 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 2.2). 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 2.1. Section 2.2 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], [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 2.3, we give a brief overview of the different approaches to the prime field case using summation polynomials. A recent paper by Amadori-Pintore-Sala [APS18] published in *Finite Fields and their Applications* 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 2.5) in section 2.3.

In section 3 we develop the algorithm in [APS18] 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.

In Section 5 we develop a method for fast evaluation of the summation polynomials. This improves the algorithms and allows us to evaluate summation polynomials $S_9$ and $S_{10}$ even though we cannot calculate the polynomials.

Section 6 contains an estimate of the complexity of the algorithm in [APS18], as well as a complexity estimate of our two algorithms presented here. Note that [APS18] did not contain an estimate of the complexity, indeed the authors state that they are "unable to estimate the complexity of solving our polynomial equation systems." We will see that Algorithm 4.1 is best, followed by Algorithm 3.1 and last comes Algorithm 2.5. 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.

Finally we present computational results for small primes in section 7, which happily agree with the complexity estimates.

# 2 Background

## 2.1 Summation Polynomials

**Definition 2.1:** [Sem04] Let $E$ be an elliptic curve over a field $K$. For $n \geq 2$, we define the summation polynomial $S_n = S_n(X_1, X_2, \ldots, X_n)$ of $E$ by the following property. Let $x_1, x_2, \ldots, x_n \in \overline{K}$, then $S_n(x_1, x_2, \ldots, x_n) = 0$ if and only if there exist $y_1, y_2, \ldots, y_n \in \overline{K}$ such that $(x_i, y_i) \in E(\overline{K})$ and $(x_1, y_1) + (x_2, y_2) + \ldots + (x_n, y_n) = \mathcal{O}$, 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 2.2:** 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
$S_2(X_1, X_2) = X_1 - X_2$,
$S_3(X_1, X_2, X_3) = (X_1 - X_2)^2 X_3^2 - 2((X_1 + X_2)(X_1 X_2 + A) + 2B)X_3 + ((X_1 X_2 - A)^2 - 4B(X_1 + X_2))$,
$S_n(X_1, \ldots, X_n) = \mathrm{Res}_X(S_{n-k}(X_1 \ldots, X_{n-k-1}, X), S_{k+2}(X_{n-k} \ldots, X_n, X))$ for $n \geq 4$ 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].

## 2.2 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 2.3:** (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, $R = P_1 + ... + P_m$, with $P_1, ..., P_m \in \mathcal{F}$. 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 $\beta$ is invertible modulo the group order $r$, then the discrete logarithm of $Q$ is $-\alpha/\beta \bmod r$.

Let $\mathcal{F} = \{P_1, P_2, \ldots, P_s\}$ 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 $R = P_1 + ... + P_m$, with $P_1, ..., P_m \in \mathcal{F}$, we use the $(m+1)^{th}$ summation polynomial: writing $R = (x_R, y_R)$, we try to find a solution $(x_1, \ldots, x_m)$ of $S_{m+1}(X_1, \ldots, X_m, x_R) = 0$ 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 $\varepsilon_1(x_1, y_1) + \ldots + \varepsilon_m(x_m, y_m) \pm R = \mathcal{O}$.

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

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

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 $S_{n+1}(X_1, \ldots, X_n, x_R) = 0$, 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 $\pm 1$ in the decomposition of $R$. Now writing $S_{n+1}(X_1, \ldots, X_n, x_R) = \sum_{i=0}^{n-1} \varphi_i(X_1, \ldots, X_n) t^i$, we instead solve $\varphi_i(X_1, \ldots, X_n) = 0$ 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.

## 2.3 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 $[0, ..., p-1]$ and treating them as integers in order to bound them). However, nobody knows 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 7: 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 [APS18] 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 an improvement in the running time. We will explain their approach now and give a complexity estimate in section 6.

Step 1. Let $s$ be the desired size of the factor base (we will show later how to select $s$). Compute random integers $a_1, ..., a_s, b_1, ..., b_s$. Then the factor base $\mathcal{F}$ is all points $\{a_1P + b_1Q, ..., a_sP + b_sQ\}$.
Step 2. Find a relation of the form $P_1 + \ldots + P_m = \mathcal{O}$ 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 2.4:** 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. $V = \{x|(x, y) \in \mathcal{F}\}$. Let $f(x) = \prod_{v \in V}(x - v)$. Then they solve (via Gröbner basis algorithms) the system

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

Hence, they only consider solutions to $S_m(X_1, \ldots, X_m) = 0$ of the form $(x_1, ..., x_m) \in V^m$, 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

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

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 6 on complexity estimates.

For completeness, we give the full algorithm from [APS18] with this approach:

**Algorithm 2.5:** [APS18]

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 $a_1, ..., a_s$, $b_1, ..., b_s$. The factor base $\mathcal{F}$ is all points $\{a_1P + b_1Q, ..., a_sP + b_sQ\}$. 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 $f_i(x) = \prod_{v \in V_i}(x - v)$, $i = 1, \ldots, m$.
2. Using a Gröbner basis algorithm, solve system (3). If there is no solution, go back to step 1.
3. If $\{x_1, \ldots, x_m\}$ is a solution to the above system, then each $x_i \in V_i$ and there exist $y_i$ such that $(x_1, y_1)+\ldots+(x_m, y_m) = \mathcal{O}$ 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 2.5, 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 $a_1, ..., a_s$, $b_1, ..., b_s$. The factor base $\mathcal{F}$ consists of all points $\{a_1P + b_1Q, ..., a_sP + b_sQ\}$. The corresponding set containing only the $x$-coordinates of the factor base points is denoted $V = \{x|(x, y) \in \mathcal{F}\}$.
2. Choose $\{x_1, \ldots, x_m\}$ a multiset of size $m$ with each $x_i \in V$ and check if $S_m(x_1, \ldots, x_m) = 0$. 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 $S_m(x_1, \ldots, x_m) = 0$ for some $\{x_1, \ldots, x_m\}$, then there exist $y_i$ such that $(x_1, y_1) + \ldots + (x_m, y_m) = \mathcal{O}$ 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 5.1 how to efficiently evaluate the summation polynomials $S_m$ in step 2.

**Remark 3.2:** In step 2, we can alternatively choose a multiset of $m$ points $\{P_1, \ldots, P_m\}$ 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 6 and section 7). 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 $a_1, ..., a_s$, $b_1, ..., b_s$. The factor base $\mathcal{F}$ is $\{a_1P + b_1Q, ..., a_sP + b_sQ\}$.
2. Choose $\{P_1, \ldots, P_{m-1}\}$ a multiset of size $m-1$ with each $P_i \in \mathcal{F}$. Choose $v \in \mathbb{F}_2^{m-1}$, and let $P_v = (-1)^{v_1}P_1 + \ldots + (-1)^{v_{m-1}}P_{m-1}$. 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 $(-1)^{v_1}P_1 + \ldots + (-1)^{v_{m-1}}P_{m-1} + P_m = \mathcal{O}$. 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 estimate of all algorithms in section 6.

**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 2.5 and Algorithm 3.1 use summation polynomials, and therefore the input value of $m$ must be $\leq 8$ 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 Summation Polynomial Evaluation

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 provides a speed-up.

The notation $m \ll s$ in this paper means that $m$ is constant and small, and $s$ is arbitrarily large. In practice $m$ is at most 10. We have in mind that $s = p^{1/m}$, as used in previous papers.

**Lemma 5.1:** Evaluating $S_m$ at a point $(x_1, \ldots, x_m)$ can be done in $O(\log^2 p)$ steps for $m \ll p$.

*Proof:* It follows from the statement of Theorem 2.2 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

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

for polynomials $f(x,y), g(x,y)$ whenever the leading coefficients are non-zero. 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 2.2,

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

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 2.2,

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

where $S_{m-1}(x_3,\ldots,x_m,X) = b_{2^{m-3}}X^{2^{m-3}} + b_{2^{m-3}-1}X^{2^{m-3}-1} + \cdots + b_0$ 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 $i^{\text{th}}$ and $j^{\text{th}}$ 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 $2^{m-4}$, ... , 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 $(17m-47)O(\log^2 p) + (17m-44)O(\log p) + 6(O(2^{3(m-3)}) + O(2^{3(m-4)}) + \ldots + O(2^{3\cdot 2}))$.

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 $S_m$. □

**Remark 5.2:** See `math.ie/ecdlp/ecdlp.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.

# 6 Complexity Estimates

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 here.

| 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).

**Remark 6.1:** The number of ways of choosing $m$ elements from a set of size $s$, allowing repetitions, is $\binom{s+m-1}{m}$ which is approximately $\frac{s^m}{m!}$ for $m \ll s$. Relations can be of the form $P_1 \pm \ldots \pm P_m = \mathcal{O}$ with $P_i \in \mathcal{F}$, so we get approximately $\frac{2^{m-1}s^m}{m!}$ possibilities. The number of points on the curve is approximately $p$. Therefore, the probability of obtaining a relation of length $m$ in $\mathcal{F}$ is approximately $\frac{2^{m-1}s^m}{p \cdot m!}$, where $s = |\mathcal{F}|$ and $m \ll s$.

**Remark 6.2:** There are $\frac{s}{m}$ ways of choosing each point in the relation giving $2^{m-1}(\frac{s}{m})^m$ possibilities for relations of the form $P_1 \pm \ldots \pm P_m = \mathcal{O}$ with each $P_i$ coming from the factor base partition $\mathcal{F}_i$ of size $\frac{s}{m}$. Therefore, 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 $\frac{2^{m-1}s^m}{p \cdot m^m}$.

**Remark 6.3:** The complexity of computing a factor base of size $s$ is bounded by $O(s \log^3 p)$, as can be seen from Table 1. (We have to do $2s$ point multiplications of $O(\log^3 p)$ and $s$ point additions of $O(\log^2 p)$.)

**Remark 6.4:** We would like the probability of finding a relation in the factor base to be close to 1, i.e. in the case of Remark 6.2, 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 [APS18] 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 2.5 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 2.5). 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 [APS18].

The following result estimates the complexity of *one* Gröbner basis computation. Let $\omega \leq 3$ be the linear algebra constant (the constant in the exponent of the complexity of multiplying matrices, see Chapter 12 of [vzGG13]). The notation $m \ll s$ in this paper means that $m$ is constant and small, and $s$ is arbitrarily large.

**Heuristic Result 6.5:** The complexity of computing a Gröbner basis of system (3) in graded reverse lexicographical order is bounded by $O(p^{\omega-\omega/m})$ for $s = p^{1/m}$ and $m \ll s$.

*Proof:* Let $D$ be the maximum degree reached during a Gröbner basis computation. There are $\binom{N+D}{N}$ monomials of degree at most $D$ in $N$ variables, therefore the complexity of a Gröbner basis computation in graded reverse lexicographical order can be bounded by $\binom{N+D}{N}^{\omega}$ as the linear algebra is the most costly part of the algorithms (see [MP15] and [JV11]).

The maximum degree $D$ reached during a Gröbner basis computation can be bounded by the Macaulay bound (see [Laz83]) $D \leq \sum_{i=1}^{l}(d_i - 1) + 1$, where $l$ is the number of polynomials and $d_i$ is the degree of the $i^{th}$ polynomial. (Refer to [CG17] for a more detailed discussion.) As system (3) has $m + 1$ polynomials in $m$ variables, $D$ can be bounded by $\sum_{i=1}^{m+1}(d_i - 1) + 1$. Also, $S_m$ has total degree $(m-1) \cdot 2^{m-2}$ (Theorem 2.2) and each $f_i$ has degree about $\frac{s}{m}$. So we get

$$D \leq (m-1) \cdot 2^{m-2} - 1 + m(\frac{s}{m} - 1) + 1 = (m-1) \cdot 2^{m-2} + s - m.$$

Thus,

$$\begin{aligned}\binom{N+D}{N} \leq & \binom{m + (m-1) \cdot 2^{m-2} + s - m}{m} \\ = & \frac{(s + (m-1) \cdot 2^{m-2})!}{m!(s + (m-1) \cdot 2^{m-2} - m)!} \\ = & \frac{(s + (m-1) \cdot 2^{m-2}) \ldots (s + (m-1) \cdot 2^{m-2} - m + 1)}{m!}.\end{aligned}$$

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

$$\binom{N+D}{N}^{\omega} \leq \frac{s^{\omega \cdot m - \omega}}{m!^{\omega}} = \frac{p^{\omega - \omega/m}}{m!^{\omega}}.$$ □

**Heuristic Result 6.6:** The complexity of Algorithm 2.5 is bounded by

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

for $s = p^{1/m}$ and $m \ll s$.

*Proof:* Steps 1 and 2 of Algorithm 2.5 may have to be computed $\frac{m^m}{2^{m-1}}$ times, each time with complexity $O(p^{1/m}\log^3 p) + O(p^{\omega-\omega/m})$ by Remark 6.3 and Heuristic Result 6.5. Once we get a factor base that yields a solution, we need to compute the solutions from the Gröbner basis in grevlex order. In theory, one may have to do a change of ordering algorithm to get a Gröbner basis in lexicographical ordering, from which we can compute the solutions. However, the probability of getting more than one solution to system (3) is negligible (by Remark 6.2, the probability of obtaining a relation of length $m$ is $\frac{2^{m-1}}{m^m}$ so the probability of obtaining two relations is the square of this). When we only find one solution the Gröbner basis elements have the form $x_i - a$ and therefore the change of ordering is trivial. □

**Remark 6.7:** Even if the algorithm does find a grevlex Gröbner basis with two solutions, the basis elements will be quadratic or linear, and again this is easy to solve and a change of ordering is not necessary.

**Remark 6.8:** 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 6.9:** With $\omega = 3$ and $m = 3$ the complexity of Algorithm 2.5 is $O(p^2)$. This roughly agrees with our experiments.

**Heuristic Result 6.10:** 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 Remark 6.1, 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 5.1, evaluating $S_m$ is $O(\log^2 p)$, giving a total of $\frac{s^m}{m!}O(\log^2 p)$ in the worst case. By Remark 6.1 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 $\frac{s^{m-1}}{m!} \geq \log p$. □

**Remark 6.11:** Replacing the evaluation of $S_m$ by adding $m$ points together as in Remark 3.2, gives a complexity of $(m-1)O(p\log^2 p) \approx O(p\log^2 p)$ for small $m$.

**Heuristic Result 6.12:** 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 $\pm P_1 \pm \ldots \pm P_{m-1}$, with $P_i \in \mathcal{F}$, allowing repetitions, for small $m$ and $s = |\mathcal{F}|$. Let $P_m = \pm P_1 \pm \ldots \pm P_{m-1}$. 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 $\pm P_1 \pm \ldots \pm P_{m-1} - P_m = \mathcal{O}$. So we still get a relation with probability as in Remark 6.1, 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 $m$. □

# 7 Experimental Results

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

- the Algorithm 2.5 first given in [APS18] (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 5.1 (called Type Eval $S_m$ †),
- 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 [APS18]. 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.)

**Remark 7.1:** 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. Both our algorithms are much faster than Algorithm 2.5 given in [APS18], and that paper shows that their algorithm is in turn faster than the one in [PKM16].

**Remark 7.2:** As we remarked in section 6, 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 [APS18] (Algorithm 2.5).

**Remark 7.3:** As expected by Remark 6.1 and Remark 6.2, 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.

**Remark 7.4:** Table 3 shows that using the evaluation function for $S_m$ described in Lemma 5.1 significantly speeds up Algorithm 3.1 for $m = 4$. It even outperforms Algorithm 4.1 for small values of $p$.

**Remark 7.5:** 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 Remark 6.1.) 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.

**Remark 7.6:** 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 2.5, 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 $S_m$ † | 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 $S_m$ † | 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 $S_m$ † | 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 $S_m$ † | 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 2.5, Eval $S_m$ denotes Algorithm 3.1, Eval $S_m$ † 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


[APS18] Alessandro Amadori, Federico Pintore, and Massimiliano Sala. On the discrete logarithm problem for prime-field elliptic curves. *Finite Fields and Their Applications*, 51:168 – 182, 2018.

[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).

[CG17] Alessio Caminata and Elisa Gorla. Solving multivariate polynomial systems and an invariant from commutative algebra. 06 2017. `https://eprint.iacr.org/2017/593`.

[Duz] S. V. Duzhin. Lecture 10 the sylvester resultant. `http://www.pdmi.ras.ru/~lowdimma/topics_nth/Resultants.pdf`.

[FHJ+14] Jean-Charles Faugère, Louise Huot, Antoine Joux, Guénaël 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 Aggelos Kiayias, editor, *Topics in Cryptology – CT-RSA 2011*, pages 356–375, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[JV13] Antoine Joux and Vanessa Vitse. Elliptic curve discrete logarithm problem over small degree extension fields. *J. Cryptol.*, 26(1):119–143, 2013.

[KV05] Erich Kaltofen and Gilles Villard. On the complexity of computing determinants. *Computational Complexity*, 13(3-4):91–130, feb 2005.

[Laz83] D. Lazard. Gröbner bases, gaussian elimination and resolution of systems of algebraic equations. In J. A. van Hulzen, editor, *Computer Algebra*, pages 146–156, Berlin, Heidelberg, 1983. Springer Berlin Heidelberg.

[MP15] Michael Monagan and Roman Pearce. A compact parallel implementation of f4. In *Proceedings of the 2015 International Workshop on Parallel Symbolic Computation*, PASCO '15, pages 95–100, New York, NY, USA, 2015. ACM.

[MvOV96] A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[PKM16] Christophe Petit, Michiel Kosters, and Ange Messeng. Algebraic approaches for the elliptic curve discrete logarithm problem over prime fields. *Public-Key Cryptography*, 9615:3–18, 2016.

[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.