Elliptic Curve Discrete Logarithm Problem over Small Degree Extension Fields

Application to the static Diffie-Hellman problem on $E(\mathbb{F}_{q^5})$

Antoine Joux[1] and Vanessa Vitse[2]

[1] DGA and Université de Versailles Saint-Quentin, Laboratoire PRISM, 45 avenue des États-Unis, F-78035 Versailles cedex, France
antoine.joux@m4x.org

[2] Université de Versailles Saint-Quentin, Laboratoire PRISM, 45 avenue des États-Unis, F-78035 Versailles cedex, France
vanessa.vitse@prism.uvsq.fr

Abstract. In 2008 and 2009, Gaudry and Diem proposed an index calculus method for the resolution of the discrete logarithm on the group of points of an elliptic curve defined over a small degree extension field $\mathbb{F}_{q^n}$. In this paper, we study a variation of this index calculus method, improving the overall asymptotic complexity when $\log q \leq cn^3$. In particular, we are able to successfully obtain relations on $E(\mathbb{F}_{p^5})$, whereas the more expensive computational complexity of Gaudry and Diem's initial algorithm makes it impractical in this case. An important ingredient of this result is a new variation of Faugère's Gröbner basis algorithm F4, which significantly speeds up the relation computation and might be of independent interest. As an application, we show how this index calculus leads to a practical example of an oracle-assisted resolution of the elliptic curve static Diffie-Hellman problem over a finite field on 130 bits, which is faster than birthday-based discrete logarithm computations on the same curve.

Key words: elliptic curve, discrete logarithm problem (DLP), index calculus, Gröbner basis computation, summation polynomials, static Diffie-Hellman problem (SDHP)

1 Introduction

Given a finite group G and two elements $g, h \in G$, the *discrete logarithm problem* (DLP) consists in computing – when it exists – an integer x such that $h = g^x$. The difficulty of this problem is at the heart of many existing cryptosystems, such that the Diffie Hellman key exchange protocol [11], the Elgamal encryption and signature scheme [13], DSA, or more recently in pairing-based cryptography. Historically, the DLP was first studied in the multiplicative group of finite fields. In such groups, now-standard index calculus methods allow to solve the DLP with a subexponential complexity. Therefore, the key size necessary to achieve a given level of security is rather large. For this reason, in 1985, Miller [26] and Koblitz [22] suggested using for G the group of points of an algebraic curve, thus introducing elliptic curves to the cryptography community.

Up to now, very few algorithms exist that solve the DLP in the group of points of an elliptic curve defined over a finite field (ECDLP). In most cases, only generic methods such as Baby-step Giant-step [33] or Pollard's rho and kangaroo algorithms [28, 29] are available. Their complexity is exponential in the size of the largest prime factor of the group cardinality; more precisely, the running time is of the order of the square root of this largest prime factor [27]. However, for some specific curves more powerful attacks can be applied; they usually move the DLP to another, weaker group. The first approach is to lift the ECDLP to a characteristic zero field, either global (i.e. $\mathbb{Q}$) or local (i.e. p-adic numbers $\mathbb{Q}_p$): so far, this works only for subgroups of $E(\mathbb{F}_{p^n})$ of order p^i [30, 32, 35]. The second approach is to transfer via the Weil or Tate pairing the DLP on $E(\mathbb{F}_q)$ to $\mathbb{F}^*_{q^k}$: this includes the MOV [25] and FR [17] attacks for elliptic curves with small embedding degree k. The last approach is to transfer via Weil descent the DLP on an elliptic curve defined over an extension field $\mathbb{F}_{q^n}$ to a second algebraic curve, defined over $\mathbb{F}_q$ but of greater genus g; this is efficient when the resulting genus g is small, which occurs only with specific curves [8].

In [31], Semaev proposed for the first time an index calculus method for the ECDLP, which unfortunately turned out to be impracticable. However, combining Semaev's ideas and Weil restriction tools, Gaudry and Diem [10, 18] independently came up with an index calculus attack of subexponential complexity for elliptic curves defined over small degree extension fields. More precisely, the complexity of their algorithm over $E(\mathbb{F}_{q^n})$ for n fixed is in $\tilde{O}(q^{2-2/n})$, but with a hidden constant in n that grows over-exponentially (in $O(n!\,2^{3n(n-1)})$). If one also allows n to go to infinity, then the complexity remains subexponential as long as n is upper bounded by $O(\sqrt{\log(q)})$.

In this article, we investigate a variant of Gaudry and Diem's attack and obtain the following result, which implies that this new approach is better than generic methods like Pollard's rho when $n \leq c \log q$, and better than Gaudry and Diem's when $n \geq c' \sqrt[3]{\log q}$ for some constants c and c'.

Theorem 1 *Let E be an elliptic curve defined over $\mathbb{F}_{q^n}$ and G a cyclic subgroup of its group of rational points. Then there exists an algorithm to solve the DLP in G with asymptotic complexity*

$$\tilde{O}\left((n-1)!\left(2^{(n-1)(n-2)} e^n\, n^{-1/2}\right)^{\omega} q^2\right)$$

where ω, $2 \leq \omega < 3$, is the linear algebra constant of [2].

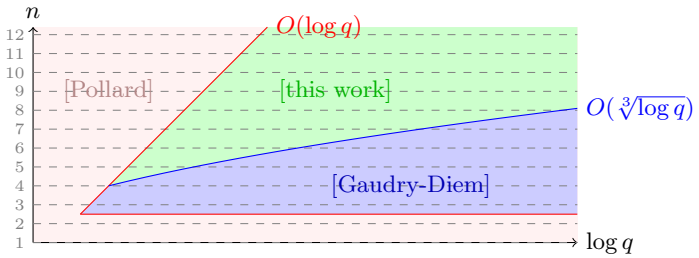

Fig. 1. Comparison of three attacks on the ECDLP over $\mathbb{F}_{q^n}, n \geq 1$.

The paper is organized as follows. First, we give a summary of Gaudry and Diem's index calculus; the main ideas are the use of the Weil restriction to obtain a convenient factor base and of Semaev's summation polynomials to test decomposition. More precisely, Gaudry and Diem check whether a given point can be decomposed as a sum of n points of this factor base, where n is the degree of the extension field. This amounts to solving a multivariate polynomial system of n equations in n variables of degree 2^{n-1}, arising from the $(n+1)$-th summation polynomial. Next, we introduce our variant: we check if a point decomposes as a sum of $n-1$ points instead of n. This reduces the likelihood of finding a relation, but greatly speeds up the decomposition process; as mentioned above, this trade-off is favorable when n is bigger than some multiple of $\sqrt[3]{\log q}$. We then give a detailed analysis of the complexity of our variant, enabling us to prove Theorem 1, and show that our trade-off is better than the one provided by the hybrid approach of [4].

The following sections are devoted to several optimizations of the decomposition process. As already noted by Gaudry, the polynomial system that has to be solved is inherently symmetrical, so it pays off to reduce its total degree by writing down the equations in terms of the elementary symmetric functions before the resolution. A convenient way to do so is to use (partially) symmetrized summation polynomials instead of Semaev's; in section 3, we sketch two different ways to directly compute these polynomials. More details are given in appendix A. The second optimization concerns the resolution of the symmetrized system. The fastest available method is to compute a Gröbner basis for an appropriate monomial order, using one of Faugère's algorithms [14, 15]. Since each relation search leads to a system with the same specific shape, we propose an ad hoc variant of F4 which takes advantage of this particularity to remove all reductions to zero.

Finally, in order to give a practical application of our idea on a problem of cryptographic interest, we present a variation of our algorithm which solves the oracle-assisted static Diffie-Hellman problem (SDHP, introduced in [6]) over $E(\mathbb{F}_{q^n})$. As in the case of finite fields presented in [21], solving the SDHP, after some oracle queries, is faster than solving the corresponding discrete logarithm problem. More precisely, we show that an attacker is able, after $\simeq q/2$ well-suited oracle queries, to compute an arbitrary SDHP instance reasonably quickly. To give a concrete example, we show that such an attack is currently achievable for a random elliptic curve defined over a finite field $\mathbb{F}_{q^5}$ of size 130 bits and requires much less computing power than a discrete logarithm computation using generic methods on the same group.

2 Index calculus attacks for elliptic curve over an extension field

We begin by briefly recalling the principle of index calculus methods. We consider a finite group G (for simplicity, we assume G has prime order r) and two elements $h, g \in G$ such that $h = [x]g$ (in additive notation), where x is the secret to recover. The basic outline of Dixon's "precomputation-and-descent" approach consists in four main steps:

1. Choice of a factor base, i.e. a family $\mathcal{F} = \{g_1, \ldots, g_N\}$ of elements of G
2. Relation search: for "random" integers a_i, decompose – if possible – $[a_i]g$ into the factor base, i.e. write
$$[a_i]g = \sum_{j=1}^{N} [c_{i,j}]g_j \tag{1}$$
3. Linear algebra: once k relations have been found where k is large enough, construct the vector $A = \left(a_i\right)_{1\leq i\leq k}$ and the matrix $M = \left(c_{i,j}\right)_{\substack{1\leq i\leq k \\ 1\leq j\leq N}}$, and solve the linear system $MX = A$. It admits a unique solution that can be computed with elementary linear algebra as soon as k is greater than N and the relations are linearly independent. The resulting vector X contains the logarithms w.r.t. g of the factor base elements.
4. Descent step: find an equation
$$[a]g + [b]h = \sum_{j=1}^{N} [c_j]g_j, \tag{2}$$
with $b \neq 0$ and deduce the logarithm of h.

For example, if G is the multiplicative group of $\mathbb{F}_p$, p prime, we can take for $\mathcal{F}$ the set of equivalence classes of prime integers smaller than a fixed bound B. An element is then decomposable in this factor base if its representative in $[1, p-1]$ is B-smooth. There is obviously a compromise to be found: if B is large, then most elements are decomposable, but many relations are necessary and the matrices involved in the linear algebra step are comparatively large. On the other hand, if the factor base is small, the required number of relations is small and the linear algebra step is fast, but finding a relation is much less probable. In any case, the matrix M is usually very sparse and appropriate techniques are used to compute its kernel quickly. Note that for discrete logarithm computations on finite fields, the descent step becomes quite involved (for example, see [21]), but in our setting it will be no different from a standard relation search. One major obstruction to index calculus when G is a subgroup of rational points of an elliptic curve defined over $\mathbb{F}_q$, is that there exist no obvious factor bases. The second, related difficulty is that decomposing an element as in (1) or (2) is really not straightforward. In [31], Semaev proposes the first efficient way to find such decomposition, yet, his approach could not work for lack of an adequate factor base.

2.1 The versions of Gaudry and Diem

In [10, 18], Gaudry and Diem propose to apply an index calculus method in the group of rational points of elliptic curves defined over small degree extension fields. To do so, they combine ideas from Semaev's index

calculus definition and Weil descent attack, to get a multivariate polynomial system that one can solve using Gröbner basis techniques. More precisely, if E is an elliptic curve defined over $\mathbb{F}_{q^n}$, their choice of factor base is the set of points whose x-coordinate lies in the base field: $\{P \in E(\mathbb{F}_{q^n}) : P = (x_P, y_P),\ x_P \in \mathbb{F}_q,\ y_P \in \mathbb{F}_{q^n}\}$. Actually, since this set is invariant under negation, it is possible to consider only one half of it; more precisely, if E is given in reduced Weierstrass form in characteristic different from 2 or 3, the factor base becomes:

$$\mathcal{F} = \{P \in E(\mathbb{F}_{q^n}) : P = (x_P, y_P),\ x_P \in \mathbb{F}_q,\ y_P \in S\}$$

where S is a subset of $\mathbb{F}_{q^n}$ such that $\mathbb{F}_{q^n} = S \cup (-S)$ and $S \cap (-S) = \{0\}$ (for example, assuming that -1 is not a square in $\mathbb{F}_{q^n}$, we can choose for S the set of quadratic residues together with 0). The same kind of twofold reduction can also be done for a general equation of E.

To compute the discrete logarithm of $Q \in \langle P \rangle$ with an index calculus algorithm, we first need to find *relations*, i.e. to decompose combinations of the form $R = [a]P$ or $R = [a]P + [b]Q$ where a, b are random integers, as sum of points in $\mathcal{F}$. Following Semaev's idea, Gaudry suggests to consider only relations of the form:

$$R = \pm P_1 \pm P_2 \pm \ldots \pm P_n \tag{3}$$

where n is the degree of the extension field and $P_i \in \mathcal{F}$ $(1 \leq i \leq n)$. Getting such relations can be done by using a Weil restriction process. One considers $\mathbb{F}_{q^n}$ as $\mathbb{F}_q[t]/(f(t))$ where $f(t)$ is an irreducible polynomial of degree n over $\mathbb{F}_q$, in order to represent points $P = (x_P, y_P) \in E(\mathbb{F}_{q^n})$ by $2n$ coordinates: $x_P = x_{0,P} + x_{1,P}t + \ldots + x_{n-1,P}t^{n-1}$ and $y_P = y_{0,P} + y_{1,P}t + \ldots + y_{n-1,P}t^{n-1}$. Instead of writing down an equation with $(n+1)n$ indeterminates from the decomposition (3), it is rather convenient to use the Semaev's summation polynomials to get rid of the y_{P_i} variables. We recall here the definition and properties of such polynomials.

Proposition 2 *Let E be an elliptic curve defined over a field K. The m-th Semaev summation polynomial is an irreducible symmetric polynomial $f_m \in K[X_1, \ldots, X_m]$, of degree 2^{m-2} in each variable, such that given $P_1 = (x_{P_1}, y_{P_1}), \ldots, P_m = (x_{P_m}, y_{P_m}) \in E(\overline{K}) \setminus \{O\}$, we have*

$$f_m(x_{P_1}, \ldots, x_{P_m}) = 0 \Leftrightarrow \exists \epsilon_1, \ldots, \epsilon_m \in \{1, -1\}, \epsilon_1 P_1 + \ldots + \epsilon_m P_m = O.$$

These summation polynomials can be effectively computed by induction, see [31] or the appendix. At this point, we replace (3) by the equivalent equation

$$f_{n+1}(x_{P_1}, \ldots, x_{P_n}, x_R) = 0, \tag{4}$$

using the $(n+1)$-th summation polynomial $f_{n+1} \in \mathbb{F}_{q^n}[X_1, \ldots, X_{n+1}]$. The unknowns $x_{P_1}, \ldots, x_{P_n}$ actually lie in $\mathbb{F}_q$, so if we sort (4) according to powers of t, we obtain $\sum_{i=0}^{n-1} \varphi_i(x_{P_1}, \ldots, x_{P_n})\, t^i = 0$ where each φ_i is a symmetric polynomial over $\mathbb{F}_q$, of degree at most 2^{n-1} in each variable (and depending on x_R, a and b). This leads to a system of n symmetric polynomials. As advised by Gaudry, it can be written as a system of polynomials of total degree 2^{n-1} in terms of the elementary symmetric functions $e_1, \ldots, e_n$ of the variables $x_{P_1}, \ldots, x_{P_n}$. Given the $(n+1)$-th summation polynomial, writing down such a system is almost immediate, but solving it is much more complicated. An efficient way to solve this multivariate polynomial system is to compute a reduced Gröbner basis for the lexicographic order. This yields a univariate polynomial whose degree is generically $2^{n(n-1)}$ in this case. Unfortunately, even the best known algorithms [14–16] have complexity at least polynomial in $2^{n(n-1)}$, which renders this attack unfeasible for $n \geq 5$ on current personal computers.

Once we get enough equations like (3), we proceed to the linear algebra step. After collecting $k > \#\mathcal{F} \simeq q/2$ distinct (independent) relations of the form:

$$[a_i]P = \sum_{j=1}^{N} [c_{i,j}]P_j, \quad \text{where } N = \#\mathcal{F}, \quad c_{i,j} \in \{0; 1; -1\} \quad \text{and} \quad \sum_{j=1}^{N} |c_{i,j}| = n,$$

we get a vector $A = (a_i)_{1 \leq i \leq k}$ and a matrix $M = (c_{i,j})$ whose right-kernel is trivial and which is very sparse since it has only n entries per row. The equation $MX = A$ has thus a unique solution in $\mathbf{Z}/ord(P)\mathbf{Z}$, which yields the discrete logarithms of the factor base elements. A single decomposition of $[a]P + [b]Q$ with $b \neq 0$ then suffices to obtain the logarithm of Q.

Complexity estimate of Gaudry and Diem's algorithm

The first main step of the algorithm consists of collecting around $\#\mathcal{F} \simeq q/2$ relations of the form (3) to build the matrix M. The $(n+1)$-th summation polynomial can be determined once for all using $Poly(e^{(n+1)^2} \log q)$ operations for a fixed elliptic curve [10]. The representation of this summation polynomial in terms of elementary symmetric functions can be done by using Gröbner elimination techniques for example, we refer to section 3 for improvements of this computation. The probability of finding one decomposition of a point $R \in E(\mathbb{F}_{q^n})$ is approximately

$$\frac{\#(\mathcal{F} \cup -\mathcal{F})^n/\mathfrak{S}_n}{\#E(\mathbb{F}_{q^n})} \simeq \frac{q^n}{n!}\frac{1}{q^n} = \frac{1}{n!},$$

and the cost of checking if the point R is actually decomposable in the factor base, noted $c(n,q)$, is the cost of the resolution of a multivariate polynomial system of n equations defined over $\mathbb{F}_q$ with n variables of total degree 2^{n-1}. As we need at least $\#\mathcal{F}$ relations, the total complexity of the first step is about $n!\, c(n,q)\, q/2$.

The estimation of the cost $c(n,q)$ is not straightforward as it thoroughly depends on the algorithm used. Following Diem's analysis [10], the polynomial system considered is generically of dimension 0 since it has a finite number of solutions over $\overline{\mathbb{F}_q}$, and using resultant techniques we get an upper bound for $c(n,q)$:

$$c(n,q) \leq Poly(n!\, 2^{n(n-1)} \log q).$$

The sparse linear algebra step can then be done in a time of $\tilde{O}(nq^2)$ with Lanczos or Wiedemann's algorithm [8]. However, in order to improve the complexity of the algorithm, Gaudry suggests to rebalance the matrix-building cost against the linear algebra cost using "large primes" techniques adapted from [19] and [36] (see [18] for more details). By doing so, he needs to collect approximately $q^{2-2/n}$ relations instead of q. The cost of the first main step becomes thus $n!\, c(n,q)\, q^{2-2/n}$. By contrast, the cost of the linear algebra step is reduced to $\tilde{O}(n\, q^{2-2/n})$, which is negligible compared to the previous step. As a result, the elliptic curve discrete logarithm problem over $\mathbb{F}_{q^n}$ for fixed n can be solved in an expected time of $\tilde{O}(q^{2-2/n})$, but the hidden constant grows extremely fast with n. A complete complexity estimate using Diem's bound is:

$$n!\, Poly(n!\, 2^{n(n-1)} \log q)\, q^{2-2/n}.$$

2.2 Our version

The bad behaviour (over-exponential) in n occurring in the complexity of Gaudry and Diem's algorithm remains a serious drawback and makes their approach practical only for very small extension degrees, namely $n = 3$ or 4. Since the cost of the multivariate system resolution heavily depends on the degree of the summation polynomial, the complexity can be considerably improved by considering only decompositions of combinations $R = [a]P$ or $R = [a]P + [b]Q$ as sum of $(n-1)$ points in $\mathcal{F}$, instead of n points as in [10, 18]. Even though we are lowering the probability of getting such a decomposition when q grows, the gain is sufficient to make this approach realistic for $n = 5$.

We can solve the equation:

$$[a]P + [b]Q = \pm P_1 \pm \ldots \pm P_{n-1}, \tag{5}$$

where a and b are random integers and the P_i belong to $\mathcal{F}$, in the same way as explained in the previous section. The differences are that only the n-th summation polynomial is involved, and that the resulting system of n polynomials is in $(n-1)$ variables and of total degree only 2^{n-2}. Since this system is overdetermined, its resolution is greatly sped up, as compared to the previous case. The trade-off is that it is less probable to find a decomposition. More precisely, the probability that a given point R decomposes into the factor base is about

$$\frac{\#(\mathcal{F} \cup -\mathcal{F})^{n-1}/\mathfrak{S}_{n-1}}{\#E(\mathbb{F}_{q^n})} = \underset{q\to\infty}{O}\left(\frac{1}{q(n-1)!}\right).$$

As mentioned, the cost of trying to find this decomposition, noted $\tilde{c}(n,q)$, is reduced to the cost of the resolution of an overdetermined multivariate polynomial system of n equations with $(n-1)$ variables of total degree 2^{n-2}. Consequently, the complexity of the relation search step becomes $(n-1)!\,\tilde{c}(n,q)\,q^2/2$.

The value of $\tilde{c}(n,q)$ has to be compared to the cost $c(n-1,q)$: clearly we have $\tilde{c}(n,q) < c(n-1,q)$. Indeed, solving a system of n equations with $(n-1)$ variables of degree 2^{n-2} can be achieved by solving the system consisting of the first $(n-1)$ equations, and by checking the compability of the solutions with the last equation. With such an upper bound of $\tilde{c}(n,q)$, we obtain the complexity for the first collecting step of

$$O\left((n-1)!\,q^2 Poly((n-1)!\,2^{(n-1)(n-2)}\log q)\right) \tag{6}$$

The linear algebra step has a complexity of $\tilde{O}(nq^2)$, which is negligible compared to the first step. Hence the total complexity of the algorithm is given by (6). We emphasize that because of the q^2 factor in the complexity, Pollard's rho or other generic methods remain faster than our variant for $n \leq 4$. Thus our approach is actually relevant only for $n \geq 5$. On the other hand, with estimate (6) it is asymptotically faster than generic methods as soon as $n \leq c\log q$ for some constant c.

However, the resultant method yielding (6) is not optimal. A much faster way of solving the overdetermined polynomial system is to compute a Gröbner basis of the corresponding zero-dimensional ideal. Generically, this ideal is the whole polynomial ring and the corresponding set of solutions is empty. Exceptionally, i.e. when the decomposition exists, the set of solutions contains a very small number of points (a single point in most cases), and this can be found directly by using a degree-reverse-lexicographic order Gröbner basis. This is in stark contrast with the situation in Gaudry and Diem's algorithm, where the solution set generically contains $2^{n(n-1)}$ points although most of them lie in an extension of $\mathbb{F}_q$. In their setting, the computation of a degrevlex Gröbner basis is not sufficient to solve the system; an elimination order basis is needed instead, whose computation (using for instance FGLM [16]) has a rather important cost.

In order to derive effective upper bounds for the complexity of a Gröbner basis computation, it is necessary to make some additional hypotheses on a zero-dimensional system $\{f_1,\ldots,f_m\}$. For instance, one can assume that the sequence $\{f_1,\ldots,f_m\}$ is semi-regular [2, 3] or that the set of solutions of the homogeneized system has no positive dimension component at infinity [24]. These properties hold generically and have been verified in all our experiments with systems arising from (5). They imply that the maximum degree of polynomials occurring during the computation of the Gröbner basis is bounded by the degree of regularity d_{reg} of the homogeneized system, which is itself smaller than the Macaulay bound $\sum_{i=1}^{m}(\deg f_i - 1) + 1$. Using the fact that the system in our variant is composed of n polynomials of degree 2^{n-2} in $n-1$ variables, we obtain that $d_{reg} \leq n2^{n-2} - n + 1$. The standard algorithms for Gröbner bases (e.g. Buchberger [7], Faugère's F4 and F5 [14, 15]) can be reduced to the computation of the row echelon form of the d_{reg}-Macaulay matrix (cf [24]), which has in our case at most $\binom{n-1+d_{reg}}{d_{reg}}$ columns and a smaller number of lines. Using fast reduction techniques, we obtain the following bound:

$$\tilde{c}(n,q) = \tilde{O}\left(\binom{n-1+d_{reg}}{d_{reg}}^{\omega}\right) = \tilde{O}\left(\left(2^{(n-1)(n-2)}e^n\, n^{-1/2}\right)^{\omega}\right),$$

where ω is the exponent in the complexity of matrix multiplication. This directly implies our main theorem:

Theorem 1 *With the above assumptions, the complexity of our algorithm is*

$$\tilde{O}\left((n-1)!\left(2^{(n-1)(n-2)}e^n\, n^{-1/2}\right)^{\omega} q^2\right) \tag{7}$$

In the same spirit, we can also try to sharpen the estimate of the complexity of Gaudry and Diem's algorithm. However, since $\omega < 3$, we find that the cost of the degrevlex Gröbner basis computation is dominated by the cost of the ordering change, whose complexity is in $O\left((2^{n(n-1)})^3\right)$. An easy computation then shows that our approach is asymptotically faster, provided $n \geq c' \sqrt[3]{\log q}$ for some constant c'.

2.3 Comparison with the hybrid approach

We have seen that the main difficulty in Gaudry and Diem's algorithm is the resolution of the polynomial system. Recently, Bettale et al. [4] have proposed a hybrid approach for solving such systems: the idea is to find a solution by exhaustive search on some variables and Gröbner basis computations of the modified systems where the selected variables have been *specialized* (i.e. evaluated). It is thus a trade-off between exhaustive search and Gröbner basis techniques. A natural choice here would be to specialize (or guess) one variable. The exhaustive search multiplies by q the number of polynomial systems, but these systems now consist of n equations in $n-1$ variables. At first sight, this seems quite similar to our version; however, the total degree of the equations in this hybrid approach is 2^{n-1} whereas it is only 2^{n-2} in our case. The following chart summarizes the number of multivariate systems to solve together with their parameters, in order to find one relation in $E(\mathbb{F}_{q^n})$. It shows that our version provides a better trade-off between the number of systems to solve and their complexity than the hybrid approach.

method	average number of systems	number of equations	number of variables	total degree
Gaudry-Diem	$n!$	n	n	2^{n-1}
Gaudry-Diem with hybrid approach	$n!\,q$	n	$n-1$	2^{n-1}
this work	$(n-1)!\,q$	n	$n-1$	2^{n-2}

2.4 Application to $\mathbb{F}_{q^5}$

The approach of Gaudry and Diem, while theoretically interesting, turns out to be impracticable on $\mathbb{F}_{q^n}$ as soon as $n \geq 5$. Not only is the computation of the 6-th summation polynomial problematic (cf. §3), but also, the fact that the system arising from (4) has generically $2^{5(5-1)} \simeq 10^6$ solutions in $\overline{\mathbb{F}_{q^5}}$ renders its resolution very cumbersome. Indeed, the complexity of the resolution (e.g. by using FGLM to obtain a lex order Gröbner basis) depends of the degree of the ideal generated by the equations, which is generically $2^{n(n-1)}$. A natural way of decreasing this degree would be to add the field equations $e_i^q - e_i$, but clearly this is not practical for large values of q. In particular, we have not been able to successfully run one complete relation search with their method, as the requested memory exceeded the capacity of our personal computer.

Nonetheless, using our algorithm and our own implementation of the F4 variant, we are able to check and if necessary compute a decomposition over $\mathbb{F}_{p^5}$ with $|p|_2 = 32$ bits in about 8.5 sec on a 2.6 GHz Intel Core 2 Duo processor. Needless to say, this is still much too slow to yield in a reasonable time the solution of the ECDLP over fields of size compatible with current levels of security. However, our approach provides an efficient attack of non-standard problems such as the oracle-assisted static Diffie-Hellman problem, as explained in section 5.

3 Computing symmetrized summation polynomials

The main difficulty of the previously investigated algorithms is the construction of relations of the form (3) or (5). Semaev's summation polynomials were first proposed in [31] to solve, or at least, to palliate this difficulty, allowing us to reduce this problem to the resolution of the polynomial equation $f_m(x_{P_1}, \ldots, x_{P_{m-1}}, x_R) = 0$, where $x_{P_1}, \ldots, x_{P_{m-1}}$ are the unknowns. The m-th polynomial f_m is computed only once and is evaluated in x_R for each relation search. As mentioned above, it is more efficient to express this equation in terms of the elementary symmetric functions of the unknowns before the resolution of the system. This symmetrizing operation greatly reduces the total degree of the system, and improves a lot its resolution by e.g. Gröbner techniques. It can be done once for all at the beginning of the relation search.

We propose two distinct improvements: both consider a direct computation of the symmetrized summation's polynomials, instead of rewriting the equation $f_m(x_{P_1}, \ldots, x_{P_{m-1}}, x_R) = 0$ in terms of elementary symmetric polynomials after the computation of Semaev's polynomials, as in [18]. These improvements reduce the computation time by a factor 10. Since these developments are not central to this article, the details will be given in appendix A; we only sketch here the main ideas.

Recall that Semaev's summation polynomials are defined recursively; each inductive step actually consists of a resultant computation. Our first improvement is to partially symmetrize after each step: it has the double benefit of reducing the size of the intermediate polynomials and the cost of the final symmetrization, by distributing it between the different steps. In our second improvement, we show, using principal divisors and Miller's technique, that $P_1 + P_2 + \ldots + P_m = O$ if and only if the polynomial $\prod(X - x_{P_i})$ has a specific shape. This condition is algebraically equivalent to the vanishing of the m-th Semaev's summation polynomial, whose symmetrized expression can be directly recovered using elimination theory. Thus, the computation of resultants and the symmetrization are replaced by an elimination order Gröbner basis computation.

4 An F4-like algorithm without reduction to zero

An efficient way to solve the multivariate polynomial system coming from (3) or (5) is to use Gröbner basis tools. The best known algorithms for constructing Gröbner bases are Faugère's F4 and F5 [14, 15], which are improvements of the classical Buchberger's algorithm. The second one, F5, is considered as the most efficient, since it includes a criterion to eliminate a priori almost all critical pairs that eventually reduce to zero. This criterion is based on the concept of "signature" of a polynomial; the main drawback is that many reductions are forbidden because they do not respect signature compatibility conditions. Hence, the polynomials considered in the course of the F5 algorithm are mostly "top-reduced" but their tails are left almost unreduced; this increases significantly the complexity of the remaining pairs' reduction. Furthermore, F5 generates many "redundant" polynomials, i.e. which are not members of a minimal Gröbner basis, but cannot be discarded for signature reasons [12]. The total number of computed critical pairs thus remains relatively important, at least compared to what could be expected from the F4 algorithm if all critical pairs reducing to zero were removed. These drawbacks are especially significant for overdetermined systems such as those we are considering. As mentioned by Faugère in [15], this is a consequence of the incremental nature of the F5 algorithm. Indeed, to determine a Gröbner basis of an ideal generated by m polynomials, F5 starts by computing a basis of the ideal generated by the first $m-1$ polynomials. Clearly, the additional equation of the overdetermined system cannot provide any speed-up at this point. Moreover, in our case, since the systems considered during the relation search always have the same shape, it is possible to extract from a precomputation the knowledge of the relevant critical pairs and to remove the pairs that lead to zero reductions. When such a precomputation is accessible, there is no reason to use F5 instead of F4.

Recall that during the course of F4 algorithm, a queue of yet untreated critical pairs is maintained. At each iteration of the main loop, some pairs are selected from this queue (according to some predefined

strategy, usually all pairs having the smallest lcm total degree) and treated, that is, their S-polynomials are computed and reduced simultaneously using linear algebra tools and former computations. The queue is then updated with the critical pairs involving the resulting new generators and satisfying the first and the second Buchberger's criteria [7, 20]. Here is a quick outline of the method we used for our computations:

1. For precomputation purposes, run a standard F4 algorithm on the first system, with the following modifications:
   - At each iteration, store the list of all selected critical pairs.
   - Each time there is a reduction to zero, remove from the stored list the critical pair that leads to the reduction.
2. For each subsequent system, run a F4 computation with the following modifications:
   - Do not maintain nor update a queue of untreated pairs.
   - At each iteration, instead of selecting pairs from the queue, pick directly from the previously stored list the relevant pairs.

As an illustration of this approach we give some examples of the speed gain it provides on $\mathbb{F}_{p^5}$, using the equations generated from the fifth summation polynomial. The system to solve is composed of 5 equations defined over $\mathbb{F}_p$ of total degree 8 in 4 variables. We run a degrevlex Gröbner basis computation of the corresponding ideal over three prime fields of size 16, 25 and 32 bits. To be fair, we compare our variant F4' with an implementation of F4 which uses the same primitives and structures (in language C), and also with the proprietary software Magma (V2.15-15) whose implementation is probably the best publicly available for the considered finite fields. All tests are performed on a 2.6 GHz Intel Core 2 Duo processor, the times are given in seconds.

	F4 (Magma)	F4	F4'
$\lvert p\rvert_2 = 16$	9.600	9.683	3.979
$\lvert p\rvert_2 = 25$	119.1	17.01	5.002
$\lvert p\rvert_2 = 32$	1046	24.43	8.496

The F4' algorithm requires a single precomputation of 11.31 sec to generate the list of relevant pairs. The above timings show that this overhead is largely compensated as soon as there are more than two subsequent computations. We emphasize that this precomputed list of relevant pairs is the same for the three cases $|p|_2 = 16,\ 25$ or 32 bits. In truth, for increased efficiency, this list was generated using an even smaller characteristic ($|p|_2 = 8$). We have also solved this system with our own implementation of the F5 algorithm[3]. The size of the Gröbner basis computed by F5 at the last step (before minimalization) is surprisingly large: it contains 17249 labeled polynomials whereas both versions of F4 never build more than 2789 polynomials at once, and construct bases containing at most 329 generators. Note that these figures do not depend on the implementation's details. The large number of polynomials that F5 computes has obvious consequences on its performances; in particular, the timings of F5 that we have obtained for this system are much worse than those of F4 or its variants. This shows that F5 as described in [15] is unsuitable for these specific systems.

5 Application to a practical oracle-assisted static Diffie-Hellman algorithm

Semaev's idea of decomposing points of $E(\mathbb{F}_{q^n})$ into a well-suited factor base leads naturally to an oracle-assisted resolution of the SDHP, similar to the finite field SDHP algorithm presented in [21]. We first recall here the definition of oracle-assisted SDHP from [6]:

Definition 3 *Let G be a finite group of order $|G|$ and $P, Q \in G$ such that $Q = [d]P$ where $d \in [1, |G| - 1]$ is a secret integer. An algorithm $\mathcal{A}$ is said to solve the SDHP in G if, given P, Q, and a challenge $X \in G$, it*

[3] At the present time, we have found no public implementation of F5 which achieves the computation of the complete Gröbner basis in a reasonable time.

outputs $[d]X \in G$.
The SDHP-solving algorithm $\mathcal{A}$ *is said to be oracle-assisted if, during a learning phase, it can make any number of queries* $X_1, \ldots, X_l$ *to an oracle that outputs* $[d]X_1, \ldots, [d]X_l$*, after which* $\mathcal{A}$ *is given a previously unseen challenge* X *and outputs* $[d]X$.

Generally, the ability to decompose points into a factor base $\mathcal{F} = \{P_1, \ldots, P_l\}$ gives the following oracle-assisted algorithm:

- learning phase: ask the oracle to compute $Q_i = [d]P_i$ for $1 \leq i \leq l$,
- decompose a challenge X as $X = \sum_i [c_i]P_i$ and answer $Y = \sum_i [c_i]Q_i$.

This methodology directly applies to the case $G = E(\mathbb{F}_{q^n})$ with the factor base $\mathcal{F} = \{P \in E(\mathbb{F}_{q^n}) : P = (x_P, y_P),\ x_P \in \mathbb{F}_q,\ y_P \in S\}$. The only minor difficulty is that a small fraction of points actually decompose (1 in $n!$ or $q\,(n-1)!$ depending of the details). However, we can use a simple variation of the descent step to circumvent this difficulty:

1. learning phase: ask the oracle to compute $Q = [d]P$ for each $P \in \mathcal{F}$
2. modified descent: given a challenge X, pick a random integer r coprime to the order of G and compute $X_r = [r]X$
3. check if X_r can be written as a sum of m points of $\mathcal{F}$: $X_r = \sum_{i=1}^{m} \epsilon_i P_i$, with $\epsilon_i \in \{-1; 1\}$
4. if X_r is not decomposable, go back to step 2; else output $Y = [s]\left(\sum_{i=1}^{m} \epsilon_i Q_i\right)$ where $s = r^{-1} \mod |G|$.

It should be noted that the same techniques can also be used to solve other variants of SDHP, such as the "Delayed Target" Discrete Logarithm or Diffie-Hellman problem (DTDLP and DTDHP) described in [23]. In practice, we have seen that, on a personal computer, we can only decompose a chosen point into at most four points of the factor base. If we are working with a finite field K whose size is approximately b bits, then assuming K is a degree 4 extension field, we obtain a complexity of $O(2^{b/4})$ oracle calls plus the decomposition cost of $4!c(4, 2^{b/4})$. On the other hand, assuming K is a degree 5 extension field, the complexity becomes $O(2^{b/5})$ oracle calls plus the decomposition cost of $4!2^{b/5}\tilde{c}(5, 2^{b/5})$. For practical applications, oracle queries are usually the limiting factor. As a consequence, for a given field size, the oracle-assisted elliptic curve SDHP is less secure over degree 5 than over degree 4 extension fields.

To our knowledge, the only other known method of solving the SDHP over a general elliptic curve consists in solving the underlying discrete logarithm problem, i.e. computing d given P and $Q = [d]P$. A state-of-the-art attack of the Certicom ECC2K-130 challenge is currently underway [1] and is expected to solve the DLP on a Koblitz elliptic curve defined over $\mathbb{F}_{2^{131}}$ in about one year with 3000 3GHz Core 2 CPUs. Since many specific characteristic 2 speed-ups are not available for finite fields of the form $\mathbb{F}_{q^5}$ (with q a prime power), generic discrete logarithm computations should be slightly slower on a random curve defined over $\mathbb{F}_{q^5}$ where $|q^5|_2 \approx 130$, but still several orders of magnitude faster than with our approach. However, by using index calculus to directly solve the SDHP on such a curve, we estimate the computation to a single month using the same 3000 3GHz Core 2 CPUs. This computation requires approximately $2^{25} \simeq 3.3 \times 10^7$ oracle queries.

6 Conclusion

In this article, we have shown that considering decomposition of points on $E(\mathbb{F}_{q^n})$ as sums of $n-1$ points improves the index calculus proposed by [10, 18], when $n \geq 5$ and $\log q \leq O(n^3)$. The key point of our approach is that such decompositions lead to overdetermined polynomial systems, which are easier to solve than the systems arising from Gaudry and Diem's original version. This resolution can be greatly sped up by using our modified Gröbner basis computation algorithm, which takes advantage of the common shape of the systems and is therefore faster than F4 and F5. Note that our "F4-like" algorithm may be applied to many other problems, as soon as one has to compute Gröbner bases of several same-shape systems.

The practicality of our algorithm is still too large to seriously threaten ECDLP-based cryptosystems with the current cryptographic key sizes. However, we further illustrate the weakness of elliptic curves defined over small degree extension fields by providing an efficient way of solving oracle-assisted Diffie-Hellman problems.

References

1. D. V. Bailey, L. Batina, D. J. Bernstein, P. Birkner, J. W. Bos, H.-C. Chen, C.-M. Cheng, G. van Damme, G. de Meulenaer, L. J. D. Perez, J. Fan, T. Gneysu, F. Gurkaynak, T. Kleinjung, T. Lange, N. Mentens, R. Niederhagen, C. Paar, F. Regazzoni, P. Schwabe, L. Uhsadel, A. V. Herrewege, and B.-Y. Yang. Breaking ECC2K-130. Cryptology ePrint Archive, Report 2009/541, 2009. `http://eprint.iacr.org/`.
2. M. Bardet. *Étude des systèmes algébriques surdéterminés. Applications aux codes correcteurs et à la cryptographie.* PhD thesis, Université Pierre et Marie Curie, Paris VI, 2004.
3. M. Bardet, J.-C. Faugère, B. Salvy, and B.-Y. Yang. Asymptotic behaviour of the degree of regularity of semi-regular polynomial systems. Presented at MEGA'05, Eighth International Symposium on Effective Methods in Algebraic Geometry, 2005.
4. L. Bettale, J.-C. Faugère, and L. Perret. Hybrid approach for solving multivariate systems over finite fields. *Journal of Mathematical Cryptology*, pages 177–197, 2009.
5. W. Bosma, J. J. Cannon, and C. Playoust. The Magma algebra system I: The user language. *J. Symb. Comput.*, 24(3/4):235–265, 1997.
6. D. R. L. Brown and R. P. Gallant. The static Diffie-Hellman problem. Cryptology ePrint Archive, Report 2004/306, 2004. `http://eprint.iacr.org/`.
7. B. Buchberger. Gröbner bases: An algorithmic method in polynomial ideal theory. In N. Bose, editor, *Multidimensional systems theory, Progress, directions and open problems, Math. Appl. 16*, pages 184–232. D. Reidel Publ. Co., 1985.
8. H. Cohen, G. Frey, R. Avanzi, C. Doche, T. Lange, K. Nguyen, and F. Vercauteren, editors. *Handbook of elliptic and hyperelliptic curve cryptography.* Discrete Mathematics and its Applications (Boca Raton). Chapman & Hall/CRC, Boca Raton, FL, 2006.
9. D. Cox, J. Little, and D. O'Shea. *Ideals, varieties, and algorithms.* Undergraduate Texts in Mathematics. Springer, New York, third edition, 2007.
10. C. Diem. On the discrete logarithm problem in elliptic curves. Preprint, available at: `http://www.math.uni-leipzig.de/~diem/preprints/dlp-ell-curves.pdf`, 2009.
11. W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Trans. Information Theory*, IT-22(6):644–654, 1976.
12. C. Eder and J. Perry. F5C: a variant of Faugère's F5 algorithm with reduced Gröbner bases. arXiv/0906.2967, 2009.
13. T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Advances in cryptology (Santa Barbara, Calif., 1984)*, volume 196 of *Lecture Notes in Comput. Sci.*, pages 10–18. Springer, Berlin, 1985.
14. J.-C. Faugère. A new efficient algorithm for computing Gröbner bases (F4). *Journal of Pure and Applied Algebra*, 139(1-3):61–88, June 1999.
15. J.-C. Faugère. A new efficient algorithm for computing Gröbner bases without reduction to zero (F5). In *Proceedings of ISSAC 2002*, New York, 2002. ACM.
16. J. C. Faugère, P. Gianni, D. Lazard, and T. Mora. Efficient computation of zero-dimensional Gröbner bases by change of ordering. *Journal of Symbolic Computation*, 16(4):329–344, 1993.
17. G. Frey and H.-G. Rück. A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves. *Math. Comp.*, 62(206):865–874, 1994.
18. P. Gaudry. Index calculus for abelian varieties of small dimension and the elliptic curve discrete logarithm problem. *J. Symbolic Computation*, 2008. doi:10.1016/j.jsc.2008.08.005.
19. P. Gaudry, E. Thomé, N. Thériault, and C. Diem. A double large prime variation for small genus hyperelliptic index calculus. *Mathematics of Computation*, 76:475–492, 2007.
20. R. Gebauer and H. M. Möller. On an installation of Buchberger's algorithm. *J. Symbolic Comput.*, 6(2-3):275–286, 1988.
21. A. Joux, R. Lercier, D. Naccache, and E. Thomé. Oracle assisted static Diffie–Hellman is easier than discrete logarithms. In M. G. Parker, editor, *IMA Int. Conf.*, volume 5921 of *Lecture Notes in Comput. Sci.*, pages 351–367. Springer, 2009.

22. N. Koblitz. Elliptic curve cryptosystems. *Math. Comp.*, 48(177):203–209, 1987.
23. N. Koblitz and A. Menezes. Another look at non-standard discrete log and Diffie-Hellman problems. *J. Math. Cryptol.*, 2(4):311–326, 2008.
24. D. Lazard. Gröbner bases, Gaussian elimination and resolution of systems of algebraic equations. In *Computer algebra (London, 1983)*, volume 162 of *Lecture Notes in Comput. Sci.*, pages 146–156. Springer, Berlin, 1983.
25. A. J. Menezes, T. Okamoto, and S. A. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Trans. Inform. Theory*, 39(5):1639–1646, 1993.
26. V. S. Miller. The Weil pairing, and its efficient calculation. *J. Cryptology*, 17(4):235–261, 2004.
27. S. Pohlig and M. Hellman. An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance. *IEEE Transactions on Information Theory*, IT-24:106–110, 1978.
28. J. M. Pollard. Monte Carlo methods for index computation (mod p). *Math. Comp.*, 32(143):918–924, 1978.
29. J. M. Pollard. Kangaroos, Monopoly and discrete logarithms. *J. Cryptology*, 13(4):437–447, 2000.
30. T. Satoh and K. Araki. Fermat quotients and the polynomial time discrete log algorithm for anomalous elliptic curves. *Comment. Math. Univ. St. Paul.*, 47(1):81–92, 1998.
31. I. Semaev. Summation polynomials and the discrete logarithm problem on elliptic curves. Cryptology ePrint Archive, Report 2004/031, 2004.
32. I. A. Semaev. Evaluation of discrete logarithms in a group of p-torsion points of an elliptic curve in characteristic p. *Math. Comp.*, 67(221):353–356, 1998.
33. D. Shanks. Class number, a theory of factorization, and genera. In *1969 Number Theory Institute (Proc. Sympos. Pure Math., Vol. XX, State Univ. New York, Stony Brook, N.Y., 1969)*, pages 415–440. Amer. Math. Soc., Providence, R.I., 1971.
34. J. H. Silverman. *The arithmetic of elliptic curves*, volume 106 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1986.
35. N. P. Smart. The discrete logarithm problem on elliptic curves of trace one. *J. Cryptology*, 12(3):193–196, 1999.
36. N. Thériault. Index calculus attack for hyperelliptic curves of small genus. In Heidelberg, editor, *Advances in Cryptology, ASIACRYPT 2003*, volume 2894 of *Lecture Notes in Comput. Sci.*, pages 75–92. Springer, September 2003.

A Symmetrized summation polynomials

In this appendix, we detail the two methods used to compute the symmetrized summation polynomials, as mentioned in section 3. The goal is to write the Semaev's summation polynomials in terms of the elementary symmetric functions of the unknowns

$$e_1 = \sum_i x_{P_i}, \quad e_2 = \sum_{i<j} x_{P_i} x_{P_j}, \quad \ldots, \quad e_{m-1} = \prod_i x_{P_i}.$$

The first method consists of symmetrizing after each resultant computation and is summarized by the following proposition:

Proposition 4 *Let E be an elliptic curve defined over a field K of characteristic different from 2 or 3, with reduced Weierstrass equation $y^2 = x^3 + ax + b$. The symmetrized summation polynomials are determined by the following induction. The initial value for $n = 3$ is given by*

$$\tilde{f}_3(e_{1,2}\,,\, e_{2,2}\,,\, X_3) = \left(e_{1,2}^2 - 4e_{2,2}\right) X_3^2 - 2\left(e_{1,2}(e_{2,2} + a) + 2b\right) X_3 + (e_{2,2} - a)^2 - 4b\, e_{1,2}$$

and for $m \geq 3$ by

$$\tilde{f}_{m+1}(e_{1,m}\,,\, \ldots, e_{m,m}\,,\, X_{m+1}) = Sym_m\left(Res_Y\left(\tilde{f}_m(e_{1,m-1}, \ldots, e_{m-1,m-1}, Y), f_3(e_{1,1}, X_{m+1}, Y)\right)\right),$$

where

* *$e_{r,n}$ is the r-th elementary symmetric polynomial in variables $X_1, \ldots, X_n$,*

* $f_3(X_1, X_2, X_3) = (X_1 - X_2)^2 X_3^2 - 2\left((X_1 + X_2)(X_1X_2 + a) + 2b\right) X_3 + (X_1X_2 - a)^2 - 4b(X_1 + X_2)$ *is the third Semaev's summation polynomial,*
* Sym_m *denotes the operation of rewriting a partially symmetrized polynomial in terms of elementary symmetric functions*

$$\begin{cases} e_{1,m} = e_{1,1} + e_{1,m-1} \\ e_{2,m} = e_{1,1}e_{1,m-1} + e_{2,m-1} \\ \qquad \vdots \\ e_{m-1,m} = e_{1,1}e_{m-2,m} + e_{m-1,m-1} \\ e_{m,m} = e_{1,1}e_{m-1,m-1} \end{cases}$$

Obviously it is also possible to define $\tilde{f}_m$ from resultants of $\tilde{f}_{m-j}$ and $\tilde{f}_{j+2}$ for $1 \leq j \leq m-3$, as in [31]. This has the advantage of reducing the number of resultant computations, but increases the complexity of the symmetrization step. In our context, since m is small ($m \leq 6$), the approach of Proposition 4 is the fastest.

We now detail the second method for the computation of the symmetrized summation polynomials. Let E be an elliptic curve of equation $y^2 = x^3 + ax + b$ defined over K of characteristic different from 2 or 3, and let $P_1, \ldots, P_m \in E(K)$ be such that $P_1 + \ldots + P_m = O$. We consider the principal divisor $D = (P_1) + \ldots + (P_m) - m(O) \in Div^0_K(E)$. Up to a constant, there exists a unique function $g_m \in K(E)$ such that $D = div(g_m)$ (cf. [34] Corollary 3.5). Same techniques as the ones used in Miller's algorithm [26] enable us to compute recursively the function g_m.

Let $l_i(X, Y) = 0$ ($1 \leq i \leq m-1$) be the equations of the lines passing through $P_1 + \ldots + P_i$ and P_{i+1} and $v_i(X, Y) = 0$ ($1 \leq i \leq m-2$) the equations of the vertical lines passing through $P_1 + \ldots + P_{i+1}$, then we have

$$g_m(X, Y) = \frac{l_1 \ldots l_{m-1}}{v_1 \ldots v_{m-2}}(X, Y).$$

An easy induction shows that

$$g_m(X, Y) = g_{m,1}(X) + Y g_{m,2}(X) \tag{8}$$

where $g_{m,1}$ and $g_{m,2}$ are two polynomials of degree lower than $d_{m,1}$ and $d_{m,2}$ respectively:

$$d_{m,1} = \begin{cases} m/2 \text{ if } m \text{ is even} \\ (m-1)/2 \text{ if } m \text{ is odd} \end{cases} \quad \text{and } d_{m,2} = \begin{cases} (m-4)/2 \text{ if } m \text{ is even} \\ (m-3)/2 \text{ if } m \text{ is odd} \end{cases}$$

Note that the function g_m is uniquely determined if the equations of l_i and v_i are normalized at the point at infinity. The intersection between the curve ($g_m = 0$) and E is exactly the set of points P_i, $1 \leq i \leq m$, thus the following proposition is quite immediate:

Proposition 5 *Let E be an elliptic curve of equation $y^2 = x^3 + ax + b$ defined over a field K of characteristic different from 2 or 3. Let $P_1, \ldots, P_m \in E(K)$ be such that $P_1 + \ldots + P_m = O$ and $g_{m,1}$ and $g_{m,2}$ be the polynomials given by equation (8). Then we have $g_{m,1}(x)^2 - g_{m,2}(x)^2(x^3 + ax + b) = 0$ if and only if x is the x-coordinate of one of the points P_i.*
Conversely, if $g_{m,1}$ and $g_{m,2}$ are two arbitrary polynomials in $K[X]$ with degree $d_{m,1}$ and $d_{m,2}$, then the roots in $\overline{K}$ of $F_m(X) = g_{m,1}(X)^2 - (X^3 + aX + b)\, g_{m,2}(X)^2$, counted with multiplicity, are the x-coordinates of points $Q_1, \ldots, Q_m \in E(\overline{K})$ such that $Q_1 + \ldots + Q_m = O$.

The second assertion comes from the fact that in $K(E)$, we have $F_m = (g_{m,1} + Y g_{m,2})(g_{m,1} - Y g_{m,2})$. Since $\deg(F_m) = m$, F_m has exactly m roots counted with multiplicity over $\overline{K}$, each of which is the x-coordinate of two opposite points $\pm Q_i \in E(\overline{K})$. Up to a change of sign, we can assume that Q_i is a zero of $g_{m,1} + Y g_{m,2}$

(and so $-Q_i$ is a zero of the second factor $g_{m,1} - Y g_{m,2}$). Thus, the principal divisor $Div(g_{m,1} + Y g_{m,2})$ is equal to $(Q_1) + \ldots + (Q_m) - m(O)$, which implies $Q_1 + \ldots + Q_m = O$.

We can now use this proposition to construct the symmetrized summation polynomials. Let $A = K[\alpha_0, \ldots, \alpha_{d_{m,1}}, \beta_0, \ldots, \beta_{d_{m,2}}, x_{P_1}, \ldots, x_{P_m}]$. We define the following elements of $A[X]$:

$$h_{m,1}(X) = \sum_{i=0}^{d_{m,1}} \alpha_i X^i, \quad h_{m,2}(X) = X^{d_{m,2}} + \sum_{i=0}^{d_{m,2}-1} \beta_i X^i,$$

$$F_m(X) = h_{m,1}(X)^2 - (X^3 + aX + b)\, h_{m,2}(X)^2$$

Finally, let I be the ideal of A generated by $F_m(x_{P_1}), \ldots, F_m(x_{P_m})$. We can easily find a different set of generators of I by identifying the coefficients of F_m with the elementary symmetric functions $e_1, \ldots, e_m$ of the variables $x_{P_1}, \ldots, x_{P_m}$, and consider the result as an ideal J of $K[\alpha_0, \ldots, \alpha_{d_{m,1}}, \beta_0, \ldots, \beta_{d_{m,2}}, e_1, \ldots, e_m]$. Elimination theory (cf. [9]) allows to compute efficiently (e.g. with appropriate Gröbner bases) a set of generators of the ideal $J' = J \cap K[e_1, \ldots, e_m]$. According to the second part of Proposition 5, a m-tuple $(e_1, \ldots, e_m)$ belongs to the algebraic set $\mathbf{V}(J')$ if and only if the roots of the polynomial $T^m + \sum_{i=1}^{m} (-1)^i e_i T^{m-i}$ are the x-coordinates of points of $E(\overline{K})$ whose sum is the point at infinity O. Actually, using Semaev's results, this elimination ideal J' is principal, generated by the m-th symmetrized summation polynomial. Hence, this elimination computes the m-th summation polynomial directly in terms of $e_1, \ldots, e_m$. Note that with this approach, it is also possible to compute directly the partially symmetrized polynomial $\tilde{f}_m$.

A worked example

For $m = 5$, following the previous construction, we obtain

$$F_5(X) = (\alpha_2 X^2 + \alpha_1 X + \alpha_0)^2 - (X^3 + aX + b)(X + \beta_0)^2$$

with $a, b \in K$. By identifying the coefficients of this polynomial with the elementary symmetric polynomials $e_1, \ldots, e_5$ of the variables $x_{P_1}, \ldots, x_{P_4}, x_{P_5}$, we deduce the polynomial system :

$$\begin{cases} e_1 = \alpha_2^2 - 2\beta_0 \\ e_2 = \beta_0^2 + a - 2\alpha_1\alpha_2 \\ e_3 = 2\alpha_0\alpha_2 + \alpha_1^2 \\ e_4 = a\beta_0^2 + 2b\beta_0 - 2\alpha_0\alpha_1 \\ e_5 = \alpha_0^2 - b\beta_0^2 \end{cases}$$

and using a Gröbner basis computation with an elimination order, we obtain the fifth summation polynomial f_5 directly in terms of $e_1, \ldots, e_5$.

Some comparisons

Here we give a comparison of computer times between the classical computation using resultants followed by a final symmetrization and our two methods. In all cases, we have used the software Magma (V2.16-4) [5] on a 3.6 GHz Intel Pentium D processor ; the symmetrizations have been done via an elimination order Gröbner basis computation. In view of the applications we have in mind, we chose to compute the 5-th symmetrized summation polynomial on an extension field $\mathbb{F}_{p^5}$, p prime.

$\log_2(p)$	resultant + symmetrization	1rst method	2nd method
8	$2.55 + 15.28 = 17.83\,\text{sec}$	$1.64\,\text{sec}$	$2.75\,\text{sec}$
16	$2.61 + 15.45 = 18.06\,\text{sec}$	$1.82\,\text{sec}$	$2.86\,\text{sec}$
32	$12.65 + 31.77 = 44.42\,\text{sec}$	$5.09\,\text{sec}$	$14.37\,\text{sec}$

We also tried to perform the same computations for the 6-th symmetrized summation polynomial. Unfortunately, in this case, both resultant and Gröbner based computations exceeded the capacity of our personal computer. However, note that we were able to obtain with our first method (partially symmetrized resultants) the 6-th symmetrized summation polynomial over $\mathbb{F}_{p^6}$ ($|p|_2 = 8$), for a fixed value of the last variable x_6, in less than 3 min, expressed in the variables $e_{1,5}, \ldots, e_{5,5}$. This means that, when using a decomposition into 5 points of the factor base, it would be possible to perform this computation, if we are willing to pay the price of redoing it for each relation search instead of precomputing it at the beginning.