Solving the Elliptic Curve Discrete Logarithm Problem Using Semaev Polynomials, Weil Descent and Gröbner Basis Methods – An Experimental Study

Michael Shantz and Edlyn Teske

University of Waterloo, Canada
{mcshantz,eteske}@uwaterloo.ca

Johannes Buchmann zu Ehren

Abstract. At ASIACRYPT 2012, Petit and Quisquater suggested that there may be a subexponential-time index-calculus type algorithm for the Elliptic Curve Discrete Logarithm Problem (ECDLP) in characteristic two fields. This algorithm uses Semaev polynomials and Weil Descent to create a system of polynomial equations that subsequently is to be solved with Gröbner basis methods. Its analysis is based on heuristic assumptions on the performance of Gröbner basis methods in this particular setting. While the subexponential behaviour would manifest itself only far beyond the cryptographically interesting range, this result, if correct, would still be extremely remarkable. We examined some aspects of the work by Petit and Quisquater experimentally.

1 Introduction

Throughout this paper, let E be an elliptic curve over $\mathbb{F}_{2^n}$,

$$E/\mathbb{F}_{2^n} : y^2 + xy = x^3 + a_2x^2 + a_6$$

where $a_2, a_6 \in {\mathbb{F}_{2^n}}^*$ and such that the trace $\mathrm{Tr}_{\mathbb{F}_{2^n}/\mathbb{F}_2}(a_2) = 1$. (In particular, if n is odd, then we can set $a_2 = 1$.) For such an elliptic curve, and an integer $m \geq 2$, the m-th *Semaev polynomial* S_m [10] is the unique polynomial in m variables with the following property: $S_m(\overline{x}_1, \ldots, \overline{x}_m) = 0$ for $\overline{x}_1, \ldots, \overline{x}_m \in \overline{\mathbb{F}_{2^n}}$ if and only if there exist $\overline{y}_1, \ldots, \overline{y}_m \in \overline{\mathbb{F}_{2^n}}$ with $(\overline{x}_i, \overline{y}_i) \in E(\overline{\mathbb{F}_{2^n}})$ and such that $\sum_{i=1}^{m}(\overline{x}_i, \overline{y}_i) = \infty \in E(\overline{\mathbb{F}_{2^n}})$. Here, $\sum_{i=1}^{m}(\overline{x}_i, \overline{y}_i)$ denotes the addition of m points on the elliptic curve E, and ∞ denotes the point at infinity. For example, $S_2(x_1, x_2) = x_1 + x_2$ since $(\overline{x}_1, \overline{y}_1) + (\overline{x}_2, \overline{y}_2) = \infty$ in $E(\overline{\mathbb{F}_{2^n}})$ if and only if $\overline{x}_1 = \overline{x}_2$. For a fixed integer m, Semaev polynomials can be used to find relations on the set of points in $E(\mathbb{F}_{2^n})$. This is done by combining Weil descent and Gröbner basis techniques to find certain zeroes of S_m. This yields an index-calculus type algorithm to solve the *elliptic curve discrete logarithm problem (ECDLP)*: given $P \in E(\mathbb{F}_{2^n})$ and $Q \in \langle P \rangle$, find ℓ such that $Q = \ell P$. Variants of this algorithm

M. Fischlin and S. Katzenbeisser (Eds.): Buchmann Festschrift, LNCS 8260, pp. 94–107, 2013.

are due to Gaudry [7] and Diem [3]. We will refer to it as *Diem's algorithm* for the remainder of this paper.

This paper is motivated by the works of Faugère, Perret, Petit and Renault [6], and Petit and Quisquater [9] who both emphasize the special form of the multivariate polynomial equations arising in Diem's algorithm, thereby suggesting a significant speed-up in the running time of special-purpose Gröbner basis techniques. While Diem [3] gives a runtime complexity of $\exp(O(n(\log n)^{1/2}))$, Faugère, Perret, Petit and Renault [6] derive complexity bounds for solving $S_m = 0$ based on the so-called *Linearization Method* and suggest a runtime complexity of $O(2^{\omega t})$ with $2.376 \leq \omega \leq 3$ (ω = the linear algebra constant) and $t \approx n/2$. (Compare this with the running time of the parallelized Pollard Rho method of $O(2^{n/2})$). This running time is under the unproven yet plausible heuristic assumption that a certain set of equations arising in the algorithm is linearly independent. Petit and Quisquater [9] conduct a more aggressive analysis of Diem's algorithm, and claim a running time complexity that is subexponential in the input size. More specifically, under the assumption on bounds on the so-called degree of regularity of a the system of equations that arises in the computation, the ECDLP over $\mathbb{F}_{2^n}$ is said to be solved in time $O(2^{cn^{2/3}\log n})$, where $c = 2\omega/3$. Petit and Quisquater also give estimated running times for specific values of the extension degree n, which indicate that Diem's algorithm outperforms the Pollard rho method for $n > 2000$. On the other hand, both papers [6,9] suggest that for large enough n (and m), the use of the so-called hybrid method (for computing a Gröbner basis) should produce even further speed-up. While these results seem to be no threat for current ECDLP-based cryptographic systems, they do require further study. The purpose of this paper is to do exactly this.

In particular, the contributions of this paper are the following:

- We confirm and extend the Petit-Quisquater experimental data [9] on the degree of regularity, up to $n = 29$. (Petit-Quisquater give data for $n = 11, 17$ only.)
- We suggest the *Delta Method* to achieve speed-up.
- We study the effect of various realizations of the hybrid method.
- Lastly, for the case n even only, we report on experiments with subfield-based factor-bases.

2 Preliminaries

2.1 Semaev's Summation Polynomials

For an integer $m \geq 2$, the m-th Semaev polynomial has been defined in the introduction. Further,

$$S_3(x_1, x_2, x_3) = x_1^2 x_2^2 + x_1^2 x_3^2 + x_2^2 x_3^2 + x_1 x_2 x_3 + a_6$$

([3, Lemma 3.4]), and recursively, for $m \geq 4$:

$$S_m(x_1, \ldots, x_m) = \mathrm{Res}_X(S_{m-k}(x_1, \ldots, x_{m-k-1}, X), S_{k+2}(x_{m-k}, \ldots, x_m, X)),$$

where $1 \leq k \leq m-3$. By definition S_m is symmetric, and S_m has degree 2^{m-2} in each x_i for $m \geq 2$. For Semaev polynomials to solve the ECDLP, we think of m to be small. For example, in the running time analysis by Petit and Quisquater, best running times are achieved with $m \leq 4$ if $n \leq 2000$, and with $m \leq 14$ for $n \leq 100,000$. Petit and Quisquater argue that using a method by Collin's [2], the calculation of S_m can be done in time $O(2^{m(m+1)})$.

2.2 An Index Calculus for the ECDLP

In our description of Diem's algorithm for the ECDLP, using Semaev's summation polynomials, we follow Petit and Quisquater [9].

Input:

- $\mathbb{F}_{2^n} = \mathbb{F}_2[z]/(f(z))$, where $\deg f = n$ and f irreducible/$\mathbb{F}_2$.
  We view $\mathbb{F}_{2^n}$ as a vector space over $\mathbb{F}_2$ of dimension n.
- Elliptic curve $E/\mathbb{F}_{2^n}$, $P \in E(\mathbb{F}_{2^n}), Q \in < P >$.

Output:

- The least positive integer ℓ such that $Q = \ell P$.

Algorithm:

- *Find a Factor Basis $\mathcal{F}_V$:*
  - Fix $n' \in [1, n] \cap \mathbb{Z}$.
  - Choose a subspace $V \subseteq \mathbb{F}_{2^n}/\mathbb{F}_2$ of $\dim V = n'$.
  - Set $\mathcal{F}_V = \{(x, y) \in E(\mathbb{F}_{2^n}); x \in V\}$
  (among pairs (x, y), (x, y') with $y \neq y'$, take only those (x, y) with lexicographically smaller y).

- *Compute Relations:*
  - Fix m.
  - Do about $2^{n'}$ times:

    - *REPEAT*
      - Take $a, b \in_R [0, 2^n] \cap \mathbb{Z}$.
      - Set $R = aP + bQ =: (x_R, y_R)$.
      - Look for $(x_1, \ldots, x_m) \in V^m$ with $S_{m+1}(x_1, \ldots, x_m, x_R) = 0$.
      UNTIL such $x_1, \ldots, x_m$ are found.

    - For $j = 1, \ldots, m$, compute $y_1, \ldots, y_m$ such that $R_j := (x_j, y_j) \in \mathcal{F}_V$.
    - Find $e_j \in \{\pm 1\}$ such that $R + \sum_{j=1}^{m} e_j R_j = \infty$.

  - *Result: about $2^{n'}$ Relations:*

$$\sum_{i=1}^{\#\mathcal{F}_V} \epsilon_{ik} R_i + a_k P + b_k Q = \infty$$

  where $\epsilon_{ik} = \{0, 1, -1\}$.

– *Linear Algebra Step:*
Use sparse matrix Linear Algebra to find a linear dependency among the relations.
Then easily obtain a solution ℓ to the ECDLP $Q = \ell P$.
Return ℓ.

Notes:

– According to Faugère et al. [6], one should choose m, n' such that $mn' \approx n$.
– If $\dim V = n'$, then Diem [3] shows that $\#\mathcal{F}_V \approx 2^{n'}$.
– If $\dim V = n'$, the probability that $S_{m+1}(x_1, \ldots, x_m, x_R) = 0$ with $(x_1, \ldots, x_m) \in V$ is, on average, $\approx \frac{2^{mn'-n}}{m!}$.

Weil Descent. It remains to discuss how to solve $S_{m+1}(x_1, \ldots, x_m, x_R) = 0$ with the added constraint that $x_1, \ldots, x_m \in V$. This is achieved via Weil descent, followed by Gröbner basis techniques. More specifically, one does the following:

– Choose a basis $\{\Theta_1, \ldots, \Theta_n\}$ of $\mathbb{F}_{2^n}/\mathbb{F}_2$.
– Choose a basis $\{v_1, \ldots, v_{n'}\}$ of $V/\mathbb{F}_2$.
– Introduce mn' new variables x_{ij}, $1 \le i \le m, 1 \le j \le n'$:
Write
$$x_i = \sum_{j=1}^{n'} x_{ij} v_j, \qquad i = 1, \ldots, m.$$
– Substitute the x_i into S_{m+1}.
Decompose each v_j, $j = 1, \ldots, n'$, and x_R, into the Θ_s, $s \in \{1, \ldots, n\}$.
Reduce any $x_{ij}^2 - x_{ij} = 0$.
– Obtain equation:

$$\begin{aligned} 0 &= S_{m+1}(x_1, \ldots, x_m, x_R) \\ &= S_{m+1}(\sum_{j=1}^{n'} x_{1j} v_j, \ldots, \sum_{j=1}^{n'} x_{mj} v_j, x_R) \\ &= [f]_1 \Theta_1 + \cdots + [f]_n \Theta_n \end{aligned}$$

where $[f]_s \in \mathbb{F}_2[x_{11}, \ldots, x_{mn'}]$ for $s = 1, \ldots, n$.

Thus, and after adding the field equations, one has to solve the set of polynomial equations in mn' variables

$$\begin{aligned} [f]_s &= 0, & s &= 1, \ldots, n, \\ x_{ij}^2 - x_{ij} &= 0, & i &= 1, \ldots, m; j = 1, \ldots, n', \end{aligned} \tag{2.1}$$

for which one can use Gröbner basis techniques such as F4 [4] or F5 [5]. See [6] and [9] for details.

2.3 The Hybrid Method

Reconsider the system (2.1). The hybrid method [1] (see also [8]) works as follows:

- Choose k variables among the x_{ij}, label them $y_1, \ldots, y_k$. There are 2^k possible choices to assign values to the k-tuple $(y_1, \ldots, y_k)$.
- For each such assignment, try to solve the new system in $mn' - k$ variables via a Gröbner basis calculation.
- With probability $\approx \frac{2^{mn'-n}}{m!}$, one of these new systems yields a solution to (2.1).

Using the hybrid method will require doing more Gröbner basis calculations, but since we are fixing k variables each time, the number of variables in the system is reduced. This produces an over-determined system which causes the Gröbner basis algorithms to run much faster.

Fixing k variables could require doing up to 2^k times as many Gröbner basis calculations. Thus we require a speed up of at least 2 each time k goes up by 1. Consequently, setting k too high causes the solution time for a single polynomial system to start increasing, since the decrease in running time for each of the 2^k required Gröbner basis calculations is not sufficient to make up for the increased number of calculations.

Note that setting $k = n$ corresponds to an exhaustive search. This approach is only efficient for very small n.

Faugère et al. [6] observed in experiments that with a suitable choice of k and some tweaking, the hybrid method is faster than solving (2.1) directly. They speculate that the hybrid method gives speedup by a factor m in the exponent.

3 Supporting the Petit-Quisquater Analysis, and More Experimental Evidence

In their Table 3, Petit and Quisquater [9] give running time estimates for Diem's algorithm for the ECDLP in $E(\mathbb{F}_{2^n})$ for various extension degrees $50 \leq n \leq 100,000$. These data illustrate the claimed subexponential behaviour, and suggest that for large enough n ($n \geq 2000$), Diem's algorithm outperforms the Pollard rho method. We reproduce their table in Table 3.1.

The Petit-Quisquater analysis is based on the assumption that for the Semaev polynomial equations (2.1),

$$\text{degree of regularity} = \text{first fall degree } + o(1). \tag{3.1}$$

Here, the degree of regularity D_{reg} is the degree of the largest Macaulay matrix appearing in a Gröbner basis computation with the algorithm F5 (cf. [9]), while the first fall degree $D_{\text{firstfall}}$ is the degree at which a non-trivial degree fall occurs during a Gröbner basis computation (see [9, Definition 2]).

We have the following facts ([9]):

- In general: $D_{\text{reg}} \geq D_{\text{firstfall}}$.

Table 3.1. Petit-Quisquater complexity estimates for the ECDLP in $E(\mathbb{F}_{2^n})$. Here, t_S = time to compute the mth Semaev polynomial, t_R = time to generate $2^{n'}$ relations, t_{LA} = time for the linear algebra step, and $T = \max\{t_S, t_R, t_{LA}\}$.

n	m	n'	t_S	t_R	t_{LA}	T
50	2	25	6	97	57	97
100	2	50	6	137	108	137
160	2	80	6	177	168	177
200	2	100	6	202	209	209
500	3	167	12	393	344	393
1000	3	250	20	664	512	664
2000	4	500	20	965	1013	1013
5000	6	833	42	1926	1682	1926
10000	7	1429	56	3020	2873	3020
20000	9	2222	90	4986	4462	4986
50000	11	4545	132	9030	9110	9110
100000	14	7143	210	14762	14306	14762

- (3.1) is true for many systems analyzed in the context of multivariate cryptosystems.
- $D_{\text{firstfall}} \leq m^2 + 1$.
- The running time to solve (2.1) with $F4$ or $F5$ is $O(n^{\omega D_{\text{reg}}})$. Memory requirements: $O(n^{2D_{\text{reg}}})$.

In their Table 2, Petit and Quisquater support (3.1) with experimental evidence for the Semaev polynomial equations, for $n = 11, 17$, $m = 2, 3$.

3.1 Extending the Petit-Quisquater Data

Mimicking the Petit-Quisquater experiments [9], we reproduced and expanded their Table 2. For this, we did the following; We fixed n, n' and m. We constructed $\mathbb{F}_{2^n} = \mathbb{F}_2[z]/(f(z))$ with f a Conway polynomial of degree n. We chose a random elliptic curve over $\mathbb{F}_{2^n}$ of order twice a prime. We chose the vector space V of dimension n' with basis $\{1, z, \ldots, z^{n'-1}\}$. We picked a random point $R = (x_R, y_R)$ of prime order on the elliptic curve and used Magma on an AMD Opteron Processor 6168 to solve the $(m+1)$-st Semaev polynomial associated with R, that is, to solve $S_{m+1}(x_1, \ldots, x_m, x_R) = 0$. This was done for 20 random curves. We measured the average degrees of regularity, the average time for solving $S_{m+1} = 0$, and the maximum memory requirement for that computation. Selected results are shown in Table 3.2 (for $m = 2$), and in Table 3.3 (for $m = 3$). They agree with the Petit-Quisquater data (given for $n = 11, 17$) and also confirm that the average degrees of regularity are lower than the assumed upper bound $m^2 + 1$. Further, for $10 \leq n \leq 21$ and $m = 3$, we observed $D_{\text{reg}} = 7, 8$ most often, and always $D_{\text{reg}} \leq 9$.

We repeated the computations for 20 random points R on the Koblitz curves $y^2 + xy = x^3 + x^2 + 1$ for various values of n, and $m = 2, 3$. Results are given in Table 3.4.

Table 3.2. Solving $S_3(x_1, x_2, x_R) = 0$. Random curves: $a_6 \in_R \mathbb{F}_{2^n}$. Average degrees of regularity, average running times and maximum memory use.

n	n'	m	m^2+1	D_{reg}	Time (s)	Max Mem (MB)
11	6	2	5	3.0	0	11
11	5	2	5	2.7	0	11
13	7	2	5	3.0	0	11
13	6	2	5	3.2	0	11
15	8	2	5	3.1	0	11
15	7	2	5	3.2	0	11
17	9	2	5	3.1	0	11
17	8	2	5	2.8	0	11
23	12	2	5	4.0	0	29
23	11	2	5	3.0	0	12
29	15	2	5	4.0	3	97
29	13	2	5	3.0	0	13

Table 3.3. Solving $S_4(x_1, x_2, x_3, x_R) = 0$. Random curves: $a_6 \in_R \mathbb{F}_{2^n}$. Average degrees of regularity, average running times and maximum memory use.

n	n'	m	m^2+1	D_{reg}	Time (s)	Max Mem (MB)
11	4	3	10	7.0	1	24
11	3	3	10	6.4	0	11
13	4	3	10	7.0	1	23
13	3	3	10	6.0	0	11
15	5	3	10	7.0	15	188
15	3	3	10	6.0	0	11
17	6	3	10	7.2	220	2143
17	3	3	10	6.0	0	11
21	7	3	10	7.0	6910	27235
21	3	3	10	6.0	0	11

4 The Delta Method

So far, we always worked with parameters m and n' such that $mn' \approx n$. In fact, previous work [9,6] used work with $n' = \lceil n/m \rceil$. A closer look at the data in Tables 3.2 and 3.3 however suggests that choosing $n' < n/m$ is favourable: for each value of n, the first row gives the data for $n' = \lceil n/m \rceil$ while the second row gives the data for an optimized $n' < n/m$. We notice lower average degrees of regularity, running times and memory requirements almost throughout. This gives rise to the *Delta method*. Let

$$\Delta := n - mn' > 0.$$

Table 3.4. Solving $S_3(x_1, x_2, x_R) = 0$ and $S_4(x_1, x_2, x_3, x_R) = 0$. Koblitz curves $y^2 + xy = x^3 + x^2 + 1$. Average degrees of regularity, average running times and maximum memory use.

n	n'	m	m^2+1	D_{reg}	Time (s)	Max Mem (MB)
11	6	2	5	3.0	0	11
11	4	3	10	7.1	1	24
13	7	2	5	3.1	0	11
13	4	3	10	7.0	1	23
15	8	2	5	3.1	0	11
15	5	3	10	7.0	16	189
17	9	2	5	3.0	0	11
17	6	3	10	7.1	211	2139
23	12	2	5	4.0	0	28
29	15	2	5	4.0	2	95

There is a tradeoff to consider when picking the value of n', as a lower n' value will reduce the chances of finding a decomposition of R into m points in the factor base, but will also reduce the factor base and therefore the number of relations needed in Diem's algorithm. The important experimental observation is that lowering n' causes the complexity of the Gröbner basis calculation (using the F4 algorithm) to decrease dramatically.

More precisely, decreasing n' by 1 decreases the chance of a decomposition being found by a factor of 2^m. It also decreases the number of relations needed by a factor of 2, since $2^{n'} + c$ relations are needed to solve the ECDLP (for some small constant c). Thus we require a speedup of 2^{m-1} in Gröbner basis calculation times in order for the Delta method to offer an improvement.

Also note that we expect that for the majority of polynomial systems for which we are trying to calculate a Gröbner basis, the system will have **no** solution. This follows from that an elliptic curve point R can be decomposed into m points of the factor basis if and only if the $m+1$-st Semaev polynomial associated with R has a solution. Thus the Delta method with $\Delta > 0$ is useful as long as the time required to calculate a Gröbner basis for systems with no solution decreases by a factor of slightly over 2 each time n' is reduced by 1 (and the time required for systems with a solution does not increase by too much). For $n = 26$, we do in fact get this required decrease for $n' \geq 11$. For $n = 34$, we get the decrease for $n' \geq 14$. Table 4.1 shows some selected average degrees of regularity and Gröbner basis running times (using Magma on an AMD Opteron 6168); to obtain these data, we used the same experiment as in Section 3.1, but separated the data into the cases that the system $S_3(x_1, x_2, x_R) = 0$ had a solution or not. We took averages over 5 calculations in each case. Observe that looking at solvable and unsolvable systems separately, we can see that solvable systems have a degree of regularity between 3 and 5, while unsolvable systems are between 2 and 4.

Table 4.1. Average degree of regularity and Gröbner basis running time, for Semaev polynomial systems with, or without solution

			Degree of Regularity		Gröbner Basis Running Time	
n	m	n'	No Sol	Sol	No Sol	Sol
26	2	13	2.8	4.0	0.39	0.99
26	2	12	2.4	3.0	0.06	0.14
26	2	11	2.4	3.0	0.01	0.03
26	2	10	2.6	4.0	0.01	0.02
34	2	17	2.8	4.0	12.31	55.20
34	2	16	3.2	4.0	1.42	3.14
34	2	15	2.8	3.0	0.21	0.47
34	2	14	2.6	3.0	0.03	0.07
34	2	13	2.0	4.0	0.02	0.04

Obviously and unfortunately, we cannot increase Δ arbitrarily in order to decrease the ECDLP running time. As we continue to increase Δ the Gröbner basis times stop decreasing as rapidly, and the overall running time starts going back up. Our experimental data suggests that for the case $m = 2$, the optimal value of Δ is given by

$$\Delta = \begin{cases} 2\lfloor \frac{n-15}{6} \rfloor & \text{if } n \text{ is even} \\ 2\lfloor \frac{n-15}{6} - 1 \rfloor & \text{if } n \text{ is odd} \end{cases}$$

At present, our best explanation for the remarkable decrease in Gröbner basis calculation time offered by the Delta method is that decreasing n' gives a system with the same number of equations, but in fewer variables. This creates a more over-determined system which can be solved more efficiently by F4. Why the F4 algorithm works so remarkably well needs more investigation. Decreasing n' often results in the polynomial systems having a lower degree of regularity. However, this is not always the case. In fact, there are cases where a decrease in n' results in both faster F4 times and a higher average degree of regularity. Clearly, a more detailed theoretic explanation for the Delta method's success is still needed.

The above optimal Δ values are based on experimental results for $m = 2$ and values of n from 25 to 40. For each n value, we decreased n' from $\lceil \frac{n}{m} \rceil$ until the overall ECDLP running time started going up. For each choice of n and n', we solved Semaev polynomials until we found five systems with a solution. Data for $n = 42$ and $n = 48$ was also generated, and our formula remains valid at these higher n values.

There are also some results available for $m = 3$ and values of n from 10 to 20. Using a value of $m = 3$ is much slower than using $m = 2$, as is expected based on the complexity analysis in [9]. We do however expect that for higher values of n, a choice of $m = 3$ will be preferable. Interestingly, the total ECDLP times for $m = 3$ increase faster than the times for $m = 2$, so our experimental results suggest that using $m = 2$ will always remain the better option. This result is

actually to be expected, as the theoretical running time determined by Petit and Quisquater increases at a faster rate for $m = 2$ than it does for $m = 3$ until we reach $n > 200$. The relation gathering stage is in fact faster for $m = 2$ than $m = 3$ until $n > 600$. Note that the theoretical switch to $m = 3$ occurs earlier, around $n = 290$, but that this is due to the complexity of the linear algebra stage, not the relation gathering stage.

5 Experiments with the Hybrid Method

Recall the hybrid method from Section 2.3, in which we fix k variables $y_1, \ldots, y_k \in \mathbb{F}_2$ among the mn' variables and perform a Gröbner basis computation for each of the 2^k possible assignments for $(y_1, \ldots, y_k) \in \mathbb{F}_2^k$ (or until a solution to $S_{m+1} = 0$ has been found). We experimentally determined optimal values of k for various n (as we could not make sensible use of existing Magma code for the hybrid method that allegedly determines such k). We worked with $m = 2$ and $21 \leq n \leq 40$, and used the same set-up for our experiments as before, with the exception that we used a Magma implementation of the hybrid method by Bettale [8] instead of the built-in Magma function GröbnerBasis. For each value of n, we used $n' = \lfloor \frac{n}{m} \rfloor$ (corresponding to $\Delta = 0$ or 1) and incremented k from 0 until the average running time to solve $S_3(x_1, x_2, x_R) = 0$ stopped improving. Optimal values for k are given in Table 5.1.

Table 5.1. Optimal k-values in the hybrid method

n	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
n'	11	12	12	13	13	14	14	15	15	16	16	17	17	18	18	19	19	20
k	0	1	0	3	1	3	2	3	2	5	3	5	4	5	4	6	5	7

We confirmed some speed-up for optimal k over $k = 0$, but the observed speed-up for the overall ECDLP running time was not always as much as with the Delta method. Specifically, for $n > 28$, an optimal choice of Δ produced better results with the Delta method than the hybrid method with an optimal choice of k.

5.1 Block Hybrid Method versus Standard Hybrid Method

When using the hybrid method, we can choose which of the mn' variables occurring in the final system to fix. Let $X_1, X_2, \ldots, X_m$ be the m variables occurring in the original Semaev polynomial. Let $x_1, x_2, \ldots, x_{mn'}$ be the mn' variables we get after doing the Weil descent, where X_i is a function of the block of variables $x_{(i-1)n'+1}, \ldots, x_{in'}$, for $i = 1, \ldots, m$.

The above results are based on fixing the first k variables, x_1 to x_k. Thus we fix all the variables corresponding to X_i before starting to fix the variables corresponding to X_{i+1}.

Other approaches are certainly possible however. For example, in the block hybrid method, the order in which we fix the variables is to fix the first variables $x_{(i-1)n'+1}$ occurring in each X_i, then to fix the second variables $x_{(i-1)n'+2}$, and so on. So for $n' = 3$, $m = 2$ and $k = 3$ we would fix the variables x_1, x_2, x_4.

We ran the same tests for the block hybrid method as we ran for the standard hybrid method, although only for n from 21 to 28. For optimal k values, the block hybrid method performed worse than the standard hybrid method for every value of n. The optimal block hybrid times were always within a factor of 2 of the optimal standard hybrid times, however. Even for non-optimal k values, the block hybrid method was normally slower. In particular, if we take $k = m$ so that we fix the first variable in each block, then the block hybrid method was slower than the standard hybrid method by a factor of over 10 times.

In conclusion, simply fixing the first k variables is preferable to the block hybrid method.

5.2 Combining the Hybrid and Delta Methods

Since the hybrid and Delta methods both give rise to faster ECDLP algorithms, we asked whether combining the two methods could give an additional speed-up?

We tried combining the two methods for n-values of 27, 36 and 40 (using $m = 2$). We used values of k from 0 to one more than the optimal k-value in the hybrid method, and values of n' from $\lceil \frac{n}{m} \rceil$ to one less than the optimal value for the n' in the Delta method. Our data were generated in the same way as before. For $n = 27$, the best combination of k and Δ was simply to take $k = 0$ and n' optimal as determined in Section 4. For $n = 36$ and 40, the best times were obtained with $k = 1$ and n' one higher than optimal. However, these times were very similar to those for $k = 0$ and n' optimal.

In conclusion, combining the hybrid and Delta methods doesn't seem to offer any significant speedup compared to just using the Delta method.

6 Exploiting the Existence of Subfields

Another choice that we have when implementing Diem's algorithm is how to choose the factor basis. Recall that the factor basis $\mathcal{F}_V$ is the set of points on the elliptic curve whose x-coordinate lies in an n'-dimensional subspace V of the underlying finite field. So we can change the factor basis by changing the basis for V. The standard basis we used to generate our data so far is given by $\{1, z, \ldots, z^{n'-1}\}$, where z is a generator for $\mathbb{F}_{2^n}$. However if $m|n$, then one can choose a basis of the form $\{1, a, \ldots, a^{n'-1}\}$, where a is a generator for $\mathbb{F}_{2^{n/m}}$, so that V is an n'-dimensional subfield of $\mathbb{F}_{2^n}$ and $\mathcal{F}_V$ contains only points $(x, y) \in V \times V$. We do not expect that using this alternate basis will affect the probability that a random elliptic curve point can be written as a sum of m points in $\mathcal{F}_V$. Hence we will still need to do the same *number* of Gröbner basis calculations.

In this section we report on experiments for the case that $m = 2$ and n is even, so $n' = n/2$. Using a subfield-based basis for V, we performed the same experiments outlined in Section 4, but repeated our experiment until we had data for 50 solvable systems and 50 unsolvable systems. The reason that we used a greater number of systems was that the calculations required less time than they did for the Delta method. For even n values between 26 and 40, we set $m = 2$, $n' = n/2$ and calculated the average time required to do a Gröbner basis calculation. From theses calculations we estimated the expected time required to solve an ECDLP instance.

Gröbner basis calculations went much faster both for both systems with a solution and systems without a solution. For $n = 40$, the calculation took 0.02 seconds for systems with no solution and 0.08 seconds for systems with a solution. In comparison, similar calculations using the standard basis take 374.77 seconds and 388.09 seconds, respectively. Significant speedups were also observed for all lower n values.

We noted that using a subfield-based basis produced systems of polynomials with a much lower degree of regularity. For each n-value tested, every single solvable systems had a degree of regularity of 3. This is lower than the average degree of regularity observed when using the standard basis. Similar results hold for systems with no solution, where the average degree of regularity ranged between 2.1 and 2.3, depending on the value of n.

See Table 6.1 for a complete comparison of average degrees of regularity. Here the data for the standard basis are taken from our Delta method results.

Table 6.1. Average degrees of regularity for subfield-based basis and standard basis

			Average Degree of Regularity			
			Standard Basis		Subfield Basis	
n	m	n'	No Sol	Sol	No Sol	Sol
26	2	13	2.5	4.0	2.2	3.0
28	2	14	2.6	4.0	2.1	3.0
30	2	15	4.0	4.0	2.2	3.0
32	2	16	2.7	4.0	2.3	3.0
34	2	17	2.6	4.0	2.1	3.0
36	2	18	4.0	4.0	2.3	3.0
38	2	19	3.0	4.0	2.3	3.0
40	2	20	4.0	4.0	2.2	3.0

At present we do not have a good explanation for why the degree of regularity is so much lower. The decrease is not the result of any cancellation during the Weil descent, as the systems of polynomials produced have the same number of terms and same total degrees no matter which basis we chose. Nonetheless, the F4 algorithm is much better at finding low degree combinations of the polynomials when a subfield is used. It is normally able to determine if a system is solvable or not after only one iteration.

7 Bonus Track: Using the Magma "PairsLimit" Parameter

When using values of $m > 2$, the default Magma implementation of the F4 algorithm tends to start using massive amounts of memory if n is increased as high as even 19 or 20. Looking at the detailed output from the Magma implementation reveals a way to decrease the memory usage.

Let I be the ideal for which we are trying to find a Gröbner basis. The F4 algorithm goes through a sequence of steps in which it takes linear combinations of polynomials in the current basis for I, and adds some of them to the basis. Magma's default behaviour is to add all new polynomials having minimal degree among the new polynomials. In large systems, this can result in thousands of polynomials being added in a single step. However, it may turn out that only a fraction of these systems needed to be added to the basis in order to find a Gröbner basis. Thus by limiting the number of new polynomials that are added each steps, we can make sure that Magma's memory usage never increases by too much at once.

The Magma function "Gröbner Basis" takes an optional parameter "PairsLimit" that can be used to include at most k new pairs at each step. Experimental results show that setting an appropriate value for this parameter can significantly improve the running time. See Table 7.1 for running time data based on $n = 19$, $n' = 6$, $m = 3$ which uses a single system for each value of "PairsLimit".

Table 7.1. Effect of PairsLimit

n	m	n'	PairsLimit	Running Time (s)	Memory (MB)
19	3	6	500	305.5	2091
19	3	6	1000	238.8	2483
19	3	6	1750	197.1	2468
19	3	6	2500	234.4	3009

Unfortunately, there is no known formula for determining optimum values to use for "PairsLimit". Using too high of a value can result in high memory usage and can also cause the final F4 step to take a very long time. However, using too low of a value can result in needing a large number of steps before a Gröbner basis is found.

Some experimentally determined values that give low running times have been chosen as defaults when working with $m > 2$. The benefits described above are also present when using $m = 2$, although not to as large of an extent. Note that using the "PairsLimit" parameter is primarily helpful when using n' values that require working with systems that take lots of time and memory to solve. As such, using this parameter likely can't help improve times for the Delta method, since the systems that result from optimal Δ values can be solved very quickly and with very little memory.

8 Conclusion

Further research, both experimentally and theoretically is needed to determine the true complexity of the ECDLP index calculus method based on Semaev polynomials, Weil descent and Gröbner basis methods!

Acknowledgments. We used the Computer Algebra System Magma in our work. We would like to thank Koray Karabina and Brandon Weir for their helpful advice and comments. We are grateful to Christophe Petit for commenting on an earlier version of this paper. We would also like to thank NSERC of Canada for providing partial funding for this research.

References

1. Bettale, L., Faugère, J.-C., Perret, L.: Hybrid approach for solving multivariate systems over finite fields. J. Math. Crypt. 3, 177–197 (2009)
2. Collins, G.: The calculation of multivariate polynomial resultants. Journal of the Association of Computing Machinery 18, 515–522 (1971)
3. Diem, C.: On the discrete logarithm problem in elliptic curves. Compositio Mathematica 147(1), 75–104 (2011)
4. Faugère, J.-C.: A new efficient algorithm for computing Gröbner bases (F4). Journal of Pure and Applied Algebra 139, 61–88 (1999)
5. Faugère, J.-C.: A new efficient algorithm for computing Gröbner bases without reduction to zero (F5). In: Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation (ISSAC), pp. 75–83. ACM Press (2002)
6. Faugère, J.-C., Perret, L., Petit, C., Renault, G.: Improving the complexity of index calculus algorithms in elliptic curves over binary fields. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 27–44. Springer, Heidelberg (2012)
7. Gaudry, P.: Index calculus for abelian varieties of small dimension and the elliptic curve discrete logarithm problem. J. Symbolic Computation 44(12), 1690–1702 (2009)
8. Bettale, L.: Hybrid approach for solving multivariate polynomial systems over finite fields, `http://www-polsys.lip6.fr/~bettale/hybrid`
9. Petit, C., Quisquater, J.-J.: On polynomial systems arising from a Weil descent. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 451–466. Springer, Heidelberg (2012)
10. Semaev, I.: Summation polynomials and the discrete logarithm problem on elliptic curves, Cryptology ePrint Archive Report 2004/031 (2004), `http://eprint.iacr.org/2004/031/`