Improvement of Faugère *et al.*'s method to solve ECDLP

Huang Yun-Ju[1], Christophe Petit[2⋆],
Naoyuki Shinohara[3], and Tsuyoshi Takagi[4]

[1] Graduate School of Mathematics, Kyushu University
y-huang@math.kyushu-u.ac.jp
[2] UCL Crypto Group
[3] NICT
[4] Institute of Mathematics for Industry, Kyushu University

Abstract. Solving the elliptic curve discrete logarithm problem (ECDLP) by using Gröbner basis has recently appeared as a new threat to the security of elliptic curve cryptography and pairing-based cryptosystems. At Eurocrypt 2012, Faugère, Perret, Petit and Renault proposed a new method using a multivariable polynomial system to solve ECDLP over finite fields of characteristic 2. At Asiacrypt 2012, Petit and Quisquater showed that this method may beat generic algorithms for extension degrees larger than about 2000.
In this paper, we propose a variant of Faugère *et al.*'s attack that practically reduces the computation time and memory required. Our variant is based on the idea of symmetrization. This idea already provided practical improvements in several previous works for composite-degree extension fields, but its application to prime-degree extension fields has been more challenging. To exploit symmetries in an efficient way in that case, we specialize the definition of factor basis used in Faugère *et al.*'s attack to replace the original polynomial system by a new and simpler one. We provide theoretical and experimental evidence that our method is faster and requires less memory than Faugère *et al.*'s method when the extension degree is large enough.

Keywords: Elliptic curve, Discrete logarithm problem, Index calculus, Multivariable polynomial system, Gröbner basis

1 Introduction

In the last two decades, elliptic curves have become increasingly important in cryptography. Elliptic curve cryptography requires shorter keys than factorization-based cryptography. Additionally, elliptic curve implementations have become increasingly efficient, and many cryptographic schemes have been proposed based on the hardness of the elliptic curve discrete logarithm problem (ECDLP) or another elliptic curve problem. These reasons led the American National Security

⋆ F.R.S.-FNRS posdoctoral fellow at Université catholique de Louvain.

Agency (NSA) to advocate the use of elliptic curves for public key cryptography in 2009 [1].

Given an elliptic curve E defined over a finite field K, some rational point P of E and a second point $Q \in \langle P \rangle \subset E$, ECDLP requires finding an integer k such that $Q = [k]P$. Elliptic curves used in practice are defined either over a prime field $\mathbb{F}_p$ or over a binary field $\mathbb{F}_{2^n}$. Like any other discrete logarithm problem, ECDLP can be solved with generic algorithms such as Baby-step Giant-step algorithm, Pollard's ρ method and their variants [2–5]. These algorithms can be parallelized very efficiently, but the parallel versions still have an exponential complexity in the size of the parameters. Better algorithms based on the *index calculus framework* have long been known for discrete logarithm problems over multiplicative groups of finite fields or hyperelliptic curves, but generic algorithms have remained the best algorithms for solving ECDLP until recently.

A key step of an index calculus algorithm for solving ECDLP is to solve the *point decomposition problem*. Given a predefined *factor basis* $\mathcal{F} \subset E$ and a random point $R \in E$, this problem asks the existence of points $P_i \in \mathcal{F}$ such that $R = \sum_i P_i$. In 2004, Semaev introduced the *summation polynomials* (also known as Semaev's polynomials) to solve this problem. The Semaev's polynomial s_r is a polynomial in r variables such that $s_r(x_1, \ldots, x_r) = 0$ if and only if there exist r points $P_i := (x_i, y_i) \in E$ such that $\sum_{i=1}^{r} P_i = O$. For a factor basis $\mathcal{F}_V := \{(x, y) | x \in V\}$ where $V \subset K$, the point decomposition problem now amounts to computing all x_i satisfying $s_{r+1}(x_1, \cdots, x_r, x(R)) = 0$ for the x-coordinate $x(R)$ of the given point R. Semaev's polynomials therefore reduce the decomposition problem on the elliptic curve E to algebraic problem over the base field K.

Solving Semaev's polynomials is not a trivial task in general, in particular if K is a prime field. For extension fields $K = \mathbb{F}_{q^n}$, Gaudry and Diem [6, 7] independently proposed to define V as the subfield $\mathbb{F}_q$ and to apply a *Weil descent* to further reduce the resolution of Semaev's polynomials to the resolution of a polynomial system of equations over $\mathbb{F}_q$. Diem generalized these ideas by defining V as a vector subspace of $\mathbb{F}_{q^n}$ [8]. Using generic complexity bounds on the resolution of polynomial systems, these authors provided attacks that can beat generic algorithms and can even have subexponential complexity for specific families of curves [6]. At Eurocrypt 2012, Faugère, Perret, Petit and Renault re-analized Diem's attack [8] in the case $\mathbb{F}_{2^n}$, and showed that the systems arising from the Weil descent on Semaev's polynomials are much easier to solve than generic systems [9]. Later at Asiacrypt 2012, Petit and Quisquater provided heuristic evidence that ECDLP is subexponential for that very important family of curves, and would beat generic algorithms when n is larger than about 2000 [10].

Even though these recent results suggest that ECDLP is weaker than previously expected for binary curves, the attacks are still far from being practical. This is mainly due to the large memory and time required to solve the polynomial systems arising from the Weil descent in practice. In particular, the experimental

results presented in [10] for primes n were limited to $n = 17$. In order to validate the heuristic assumptions taken in Petit and Quisquater's analysis and to estimate the exact security level of binary elliptic curves in practice, experiments on larger parameters are definitely required.

Hybrid methods (involving a trade-off between exhaustive search and polynomial system solving) have been proposed to practically improve the resolution of the polynomial systems [11]. More importantly, the special structure of these systems can be exploited. When n is composite and the Weil descent is performed on an intermediary subfield, Gaudry already showed in [7] how the symmetry of Semaev's polynomials can be exploited to accelerate the resolution of the polynomial system in practice. In that case, the whole system can be re-written with new variables corresponding to the fundamental symmetric polynomials, therefore reducing the degrees of the equations and improving their resolution. In the particular cases of twisted Edward curves and twisted Jacobi curves, Faugère *et al.* also exploited additional symmetry coming from the existence of a rational 2-torsion point to further reduce the degrees of the equations [12].

In this paper, we focus on Diem's version of index calculus for ECDLP over a binary field of prime extension degree n [8–10]. In that case, the Weil descent is performed on a vector space that is not a subfield of $\mathbb{F}_{2^n}$, and the resulting polynomial system cannot be re-written in terms of symmetric variables only. We therefore introduce a different method to take advantage of symmetries even in the prime degree extension case. Our re-writing of the system involves both symmetric and non-symmetric variables. The total number of variables is increased compared to [9, 10], but we limit this increase as much as possible thanks to an appropriate choice of the vector space V. On the other hand, the use of symmetric variables in our system allows reducing the degrees of the equations significantly. Our experimental results show that our systems can be solved faster than the original systems of [12, 21] as long as n is large enough.

Notations In this work, we are interested in solving the elliptic curve discrete logarithm problem on a curve E defined over a finite field $\mathbb{F}_{2^n}$, where n is a prime number. We denote by $E_{\alpha,\beta}$ the elliptic curve over $\mathbb{F}_{2^n}$ defined by the equation $y^2 + xy = x^3 + \alpha x^2 + \beta$. For a given point $P \in E$, we use $x(P)$ and $y(P)$ to indicate the x-coordinate and y-coordinate of P respectively. From now on, we use the specific symbols P, Q and k for the parameters and solution of the ECDLP: $P \in E$, $Q \in \langle P \rangle$, and k is the smallest non-negative integer such that $Q = [k]P$. We identify the field $\mathbb{F}_{2^n}$ as $\mathbb{F}_2[\omega]/h(\omega)$, where h is an irreducible polynomial of degree n. Any element $e \in \mathbb{F}_{2^n}$ can then be represented as $poly(e) := c_0 + c_1\omega + ... + c_{n-1}\omega^{n-1}$ where $c_i \in \mathbb{F}_2$. For any set S, we use the symbol $\#S$ to mean the order of S.

Outline. The remaining of this paper is organized as follows. In Section 2, we recall previous index calculus algorithms for ECDLP, in particular Faugère *et al.*'s attack on binary elliptic curves and previous work exploiting the symmetry of Semaev's polynomials when the extension degree is composite. In Section 3, we describe our variant of Faugère *et al.*'s attack taking advantage of the symmetries

even when the extension degree is prime. In Section 4, we provide experimental results supporting our method with respect to Faugère *et al.*'s original attack. Finally in Section 5, we conclude the paper and we introduce further work.

2 Index Calculus for Elliptic Curves

2.1 The Index Calculus Method

For a given point $P \in E_{\alpha,\beta}$, let Q be a point in $\langle P \rangle$. The index calculus method can be adapted to elliptic curves to compute the discrete logarithm of Q with respect to P.

Algorithm 1 Index Calculus for ECDLP [13]

Input: elliptic curve $E_{\alpha,\beta}$, point $P \in E_{\alpha,\beta}$, point $Q \in \langle P \rangle$
1 $F \longleftarrow$ a subset of $E_{\alpha,\beta}$
2 $M \longleftarrow$ matrix with $\#F + 2$ columns
3 **while** $Rank(M) < \#F + 1$ **do**
4 $\quad R \longleftarrow [a]P + [b]Q$ where a and b are random integers in $(0, \#\langle P \rangle)$
5 $\quad sol_m \longleftarrow$ `Decompose(`R, F`)`
6 $\quad M \longleftarrow$ `AddRelationToMatrix(`sol_m`)`
7 **end**
8 $M \longleftarrow$ `ReducedRowEchelonForm(`M`)`
9 $a', b' \longleftarrow$ last two column entries of last row
10 $k \longleftarrow -a'/b'$
Output: k, where $Q = [k]P$

As shown in Algorithm 1, we first select a *factor base* $F \subset E_{\alpha,\beta}$ and we perform a *relation search* expressed as the loop between the line 3 and 7 of Algorithm 1. This part is currently the efficiency bottleneck of the algorithm. For each step in the loop, we compute $R := [a]P + [b]Q$ for random integers a and b and we apply the **Decompose** function on R to find all tuples (sol_m) of m elements $P_{j_\ell} \in F$ such that $P_{j_1} + P_{j_2} + \cdots + P_{j_m} + R = O$. Note that we may obtain several decompositions for each point R. In the line 6, the **AddRelationToMatrix** function encodes every decomposition of a point R into a row vector of the matrix M. More precisely, the first $\#F$ columns of M correspond to the elements of F, the last two columns correspond to P and Q, and the coefficients corresponding to these points are encoded in the matrix. In the line 7, the **ReducedRowEchelonForm** function reduces M into a row echelon form. When the rank of M reaches $\#F + 1$, the last row of the reduced M is of the form $(0, \cdots, 0, a', b')$, which implies that $[a']P + [b']Q = O$. From this relation, we obtain $k = -a'/b' \bmod \#\langle P \rangle$.

A straightforward method to implement the **Decompose** function would be to exhaustively compute the sums of all m-tuples of points in F and to compare these sums to R. However, this method would not be efficient enough.

2.2 Semaev's Polynomials

Semaev's polynomials [13] allow replacing the complicated addition law involved in the point decomposition problem by a somewhat simpler polynomial equation over $\mathbb{F}_{2^n}$.

Definition 1 *The m-th Semaev's polynomial s_m for $E_{\alpha,\beta}$ is defined as follows: $s_2 := x_1 + x_2$, $s_3 := (x_1x_2 + x_1x_3 + x_2x_3)^2 + x_1x_2x_3 + \beta$, and $s_m := Res_X(s_{j+1}(x_1, ..., x_j, X), s_{m-j+1}(x_{j+1}, ..., x_m, X))$ for $m \geq 4$, $2 \leq j \leq m-2$.*

The polynomial s_m is symmetric and it has degree 2^{m-2} with respect to each variable. Definition 1 provides a straightforward method to compute it. In practice, computing large Semaev's polynomials may not be a trivial task, even if the symmetry of the polynomials can be used to accelerate it [11]. Semaev's polynomials have the following property:

Proposition 1 *We have $s_m(x_1, x_2, ..., x_m) = 0$ if and only if there exist $y_i \in \mathbb{F}_{2^n}$ such that $P_i = (x_i, y_i) \in E_{\alpha,\beta}$ and $P_1 + P_2 + ... + P_m = O$.*

In Semaev's seminal paper [13], he proposed to choose the factor base F in Algorithm 1 as

$$F_V := \{(x, y) \in E_{\alpha,\beta} | x \in V\}$$

where V is some subset of the base field of the curve. According to Proposition 1, finding a decomposition of a given point $R = [a]P + [b]Q$ is then reduced to first finding $x_i \in V$ such that

$$s_{m+1}(x_1, x_2, ..., x_m, x(R)) = 0,$$

and then finding the corresponding points $P_j = (x_j, y_j) \in F_V$.

A straightforward **Decompose** function using Semaev's polynomials is described in Algorithm 2. In this algorithm, Semaev's polynomials are solved by a

Algorithm 2 **Decompose** function with s_{m+1}

Input: $R = [a]P + [b]Q$, factor base F_V
$Set_m \longleftarrow \{e \in F_V^m\}$
$sol_m \longleftarrow \{\}$
for $e = \{P_1, P_2, .., P_m\} \in Set_m$ **do**
  if $s_{m+1}(x(P_1), x(P_2), ..., x(P_m), x(R)) = 0$ **then**
    if $P_1 + P_2 + ... + P_m + R = O$ **then**
      $sol_m \longleftarrow sol_m \cup \{e\}$
    end
  end
end
Output: sol_m contains the decomposition elements of R w.r.t. F_V

naive exhaustive search method. Since every x-coordinate corresponds to at most

two points on the elliptic curve $E_{\alpha,\beta}$, each solution of $s_{m+1}(x_1, x_2, ..., x_m, x(R)) = 0$ may correspond to up to 2^m possible solutions in $E_{\alpha,\beta}$. These potential solutions are tested in the line 5 of Algorithm 2. As such, Algorithm 2 still involves some exhaustive search and can clearly not solve ECDLP faster than generic algorithms.

2.3 Method of Faugère *et al.*

At Eurocrypt 2012, following similar approaches by Gaudry [7] and Diem [6, 8], Faugère *et al.* provided V with the structure of a vector space, to reduce the resolution of Semaev's polynomial to a system of multivariate polynomial equations. They then solved this system using Gröbner basis algorithms [9].

More precisely, Faugère *et al.* suggested to fix V as a random vector subspace of $\mathbb{F}_{2^n}/\mathbb{F}_2$ with dimension n'. If $\{v_1, \ldots, v_{n'}\}$ is a basis of this vector space, the resolution of Semaev's polynomial is then reduced to a polynomial system as follows. For any fixed $P' \in F_V$, we can write $x(P')$ as

$$x(P') = \bar{c}_1 v_1 + \bar{c}_2 v_2 + ... + \bar{c}_{n'} v_{n'}$$

where $\bar{c}_\ell \in \mathbb{F}_2$ are known elements. Similarly, we can write all the variables $x_j \in V$ in $s_{m+1} \mid_{x_{m+1}=x(R)}$ as

$$\begin{cases} x_j = c_{j,1} v_1 + c_{j,2} v_2 + \ldots + c_{j,n'} v_{n'}, \quad 1 \leq j \leq m, \\ x_{m+1} = r_0 + r_1 \omega + \ldots + r_{n-1} \omega^{n-1}, \end{cases}$$

where $c_{j,\ell}$ are binary variables and $r_\ell \in \mathbb{F}_2$ are known. Using these equations to substitute the variables x_j in s_{m+1}, we obtain an equation

$$s_{m+1} = f_0 + f_1 \omega + \ldots + f_{n-1} \omega^{n-1},$$

where $f_0, f_1, ..., f_{n-1}$ are polynomials in the binary variables $c_{j,l}$, $1 \leq j \leq m$, $1 \leq l \leq n'$.

We have $s_{m+1} \mid_{x_{m+1}=x(R)} = 0$ if and only if each binary coefficient polynomial f_l is equal to 0. Solving Semaev's polynomial s_{m+1} is now equivalent to solving the binary multivariable polynomial system

$$\begin{cases} f_0(c_{1,1}, \ldots, c_{0,l}, \ldots, c_{m,n'}) = 0, \\ f_1(c_{1,1}, \ldots, c_{1,l}, \ldots, c_{m,n'}) = 0, \\ \qquad \vdots \\ f_m(c_{1,1}, \ldots, c_{m,l}, \ldots, c_{m,n'}) = 0, \end{cases} \tag{1}$$

in the variables $c_{j,\ell}$, $1 \leq j \leq m, 1 \leq \ell \leq n'$.

The **Decompose** function using this system is described in Algorithm 3. It is denoted as Imp_{FPPR} in this work. We first substitute x_{m+1} with $x(R)$ in s_{m+1}. The **TransFromSemaevToBinaryWithSym** function transforms the equation $s_{m+1} \mid_{x_{m+1}=x(R)} = 0$ into System (1) as described above. To solve this

Algorithm 3 **Decompose** function with binary multivariable polynomial system (Imp_{FPPR}) [9]

Input: $R = [a]P + [b]Q$, factor base F_V

1 $F \longleftarrow$ `TransFromSemaevToBinary`$(s_{m+1} \mid_{x_{m+1}=x(R)})$

2 $GB(F) \longleftarrow$ `GroebnerBasis`$(F, \prec_{lex})$

3 $sol(F) \longleftarrow$ `GetSolutionFromGroebnerBasis`$(GB(F))$

4 $sol_m \longleftarrow \{\}$

5 **for** $e = \{P_1, P_2, .., P_m\} \in sol(F)$ **do**

6 | **if** $P_1 + P_2 + ... + P_m + R = O$ **then**

7 | | $sol_m \longleftarrow sol_m \cup \{e\}$

8 | **end**

9 **end**

Output: sol_m contains the decomposition elements of R w.r.t. F_V

system, we compute its Gröbner basis with respect to a lexicographic ordering using an algorithm such as F_4 or F_5 algorithm [14, 15]. A Gröbner basis for a lexicographic ordering always contains some univariate polynomial (the polynomial 1 when there is no solution), and the solutions of F can be obtained from the roots of this polynomial. However, since it is much more efficient to compute a Gröbner basis for a graded-reversed lexicographic order than for a lexicographic ordering, a Gröbner basis of F is first computed for a graded-reverse lexicographic ordering and then transformed into a Gröbner basis for a lexicographic ordering using FGLM algorithm [16].

After getting the solutions of F, we find the corresponding solutions over $E_{\alpha,\beta}$. As before, this requires to check whether $P_1 + P_2 + ... + P_m + R = O$ for all the potential solutions in the line 6 of Algorithm 3.

Although Faugère *et al.*'s approach provides a systematic way to solve Semaev's polynomials, their algorithm is still not practical. Petit and Quisquater estimated that the method could beat generic algorithms for extension degrees n larger than about 2000 [10]. This number is much larger than the parameter $n = 160$ that is currently used in applications. In fact, the degrees of the equations in F grow quadratically with m, and the number of monomial terms in the equations is exponential in this degree. In practice, the sole computation of the Semaev's polynomial s_{m+1} seems to be a challenging task for m larger than 7. Because of the large computation costs (both in time and memory), no experimental result has been provided yet when n is larger than 20.

In this work, we provide a variant of Faugère *et al.*'s method that practically improves its complexity. Our method exploits the symmetry of Semaev's polynomials to reduce both the degree of the equations and the number of monomial terms appearing during the computation of a Gröbner basis of the system F.

2.4 Use of Symmetries in Previous Works

The symmetry of Semaev's polynomials has been exploited in previous works, but always for finite fields $\mathbb{F}_{p^n}$ with *composite* extension degrees n. The approach was already described by Gaudry [7] as a mean to accelerate the Gröbner basis computations. The symmetry of Semaev's polynomials has also been used by Joux and Vitse's to establish new ECDLP records for composite extension degree fields [11, 17]. Extra symmetries resulting from the existence of a rational 2-torsion point have also been exploited by Faugère *et al.* for twisted Edward curves and twisted Jacobi curves [12]. In all these approaches, exploiting the symmetries of the system allows reducing the degrees of the equations and the number of monomials involved in the Gröbner basis computation, hence it reduces both the time and the memory costs.

To exploit the symmetry in ECDLP index calculus algorithms, we first rewrite Semaev's polynomial s_{m+1} with the elementary symmetric polynomials.

Definition 2 *Let $x_1, x_2, ..., x_m$ be m variables, then the elementary symmetric polynomials are defined as*

$$\begin{cases} \sigma_1 := \sum_{1 \leq j_1 \leq m} x_{j_1} \\ \sigma_2 := \sum_{1 \leq j_1 < j_2 \leq m} x_{j_1} x_{j_2} \\ \sigma_3 := \sum_{1 \leq j_1 < j_2 < j_3 \leq m} x_{j_1} x_{j_2} x_{j_3} \\ \quad \vdots \\ \sigma_m := \prod_{1 \leq j \leq m} x_j \end{cases} \tag{2}$$

Any symmetric polynomial can be written as an algebraic combination of these elementary symmetric polynomials. We denote the symmetrized version of Semaev's polynomial s_m by s'_m. For example for the curve $E_{\alpha,\beta}$ in characteristic 2, we have

$$s_3 = (x_1x_2 + x_1x_3 + x_2x_3)^2 + x_1x_2x_3 + \beta,$$

where x_3 is supposed to be fixed to some $x(R)$. The elementary symmetric polynomials are

$$\begin{aligned} \sigma_1 &= x_1 + x_2, \\ \sigma_2 &= x_1x_2. \end{aligned}$$

The symmetrized version of s_3 is therefore

$$s'_3 = (\sigma_2 + \sigma_1 x_3)^2 + \sigma_2 x_3 + \beta.$$

Since x_3 is fixed and the squaring is a linear operation over $\mathbb{F}_2$, we see that symmetrization leads to a much simpler polynomial.

Let us now assume that n is a composite number with a non-trivial factor n'. In this case, we can fix the vector space V as the subfield $\mathbb{F}_{p^{n'}}$ of $\mathbb{F}_{p^n}$. We note that all arithmetic operations are closed on the elements of V for this special choice. In particular, we have

$$if\ x_i \in V\ then\ \sigma_i \in V\ . \tag{3}$$

Let now $\{1, \omega_2, \ldots, \omega_{n/n'}\}$ be a basis of $\mathbb{F}_{p^n}/\mathbb{F}_{p^{n'}}$. We can write

$$\begin{aligned}\sigma_j &= d_{j,0} \; for \; 1 \leq j \leq m,\\ x_{m+1} &= r_1 + r_2\omega_2 + \ldots + r_{n/n'}\omega_{n/n'},\end{aligned}$$

where $r_\ell \in \mathbb{F}_{p^n}$ are known and the variables $d_{j,0}$ are defined over $\mathbb{F}_{p^{n'}}$. These relations can be substituted in the equation $s'_{m+1} \mid_{x_{m+1}=x(R)} = 0$ to obtain a system of n/n' equations in the m variables $d_{j,0}$ only. Since the total degree and the degree of s'_m with respect to each symmetric variable σ_i are lower than those of s_m with respect to all non-symmetric variables x_i, the degrees of the equations in the resulting system are also lower and the system is easier to solve. As long as $n/n' \approx m$, the system has a reasonable chance to have a solution.

Given a solution $(\sigma_1, \ldots, \sigma_m)$ for this system, we can recover all possible corresponding values for the variables $x_1, \ldots, x_m$ (if there is any) by solving the system given in Definition 2, or equivalently by solving the symmetric polynomial equation

$$x^m + \sum_{i=1}^{m} \sigma_i x^{m-i} = x^m + \sigma_1 x^{m-1} + \sigma_2 x^{m-2} + \ldots + \sigma_m.$$

Note that the existence of a non-trivial factor of n and the special choice for V are crucial here. Indeed, they allow building a new system that only involves symmetric variables and that is significantly simpler to solve than the previous one.

3 Using Symmetries with Prime Extension Degrees

When n is prime, the only subfield of $\mathbb{F}_{2^n}$ is $\mathbb{F}_2$, but choosing $V = \mathbb{F}_2$ would imply to choose $m = n$, hence to work with Semaev's polynomial s_{n+1} which would not be practical when n is large. In Diem's and Faugère *et al.*'s attacks [9, 8], the set V is therefore a generic vector subspace of $\mathbb{F}_{2^n}/\mathbb{F}_2$ with dimension n'. In that case, Implication (3) does not hold, but we now show how to nevertheless take advantage of symmetries in Semaev's polynomials.

3.1 A New System with both Symmetric and Non-Symmetric Variables

Let n be an arbitrary integer (possibly prime) and let V be a vector subspace of $\mathbb{F}_{2^n}/\mathbb{F}_2$ with dimension n'. Let $\{v_1, \ldots, v_{n'}\}$ be a basis of V. We can write

$$\begin{cases} x_j = c_{j,1}v_1 + c_{j,2}v_2 + ... + c_{j,n'}v_{n'}, \; for \; 1 \leq j \leq m \\ x_{m+1} = r_0 + r_1\omega + ... + r_{n-1}w^{n-1}, \end{cases}$$

where $c_{j,\ell}$ with $1 \leq j \leq m$ and $1 \leq \ell \leq n'$ are variables but r_ℓ, $1 \leq \ell \leq n$ are known elements in $\mathbb{F}_2$.

Like in the composite extension degree case, we can use the elementary symmetric polynomials to write Semaev's polynomial s_{m+1} as a polynomial s'_{m+1} in the variables σ_j only. However since V is not a field anymore, constraining x_j in V does not constrain σ_j in V anymore. Since $\sigma_j \in \mathbb{F}_{2^n}$, we can however write

$$\begin{cases} \sigma_1 = d_{1,0} + d_{1,1}\omega + \ldots + d_{1,n-1}\omega^{n-1}, \\ \sigma_2 = d_{2,0} + d_{2,1}\omega + \ldots + d_{2,n-1}\omega^{n-1}, \\ \qquad\qquad \vdots \\ \sigma_m = d_{m,0} + d_{m,1}\omega + \ldots + d_{m,n-1}\omega^{n-1}. \end{cases}$$

where $d_{j,\ell}$ with $1 \leq j \leq m$ and $1 \leq \ell \leq n$ are binary variables. Using these equations, we can substitute σ_j in s'_{m+1} to obtain

$$s'_{m+1} = f'_0 + f'_1\omega + \ldots + f'_{n-1}\omega^{n-1}$$

where $f'_0, f'_1, ..., f'_{n-1}$ are polynomials in the binary variables $d_{j,\ell}$. Applying a Weil descent on the symmetrized Semaev's polynomial equation $s'_m = 0$, we therefore obtain a polynomial system

$$f'_0 = f'_1 = \ldots = f'_{n-1} = 0$$

in the mn binary variables $d_{j,\ell}$.

The variables $d_{j,\ell}$ must also satisfy certain constraints provided by System (2). More precisely, substituting both the x_j and the σ_j variables for binary variables in the equation

$$\sigma_j = \sum_{\substack{I \subset \{1,\ldots,m\} \\ \#I = j}} \prod_{k \in I} x_k \ ,$$

we obtain

$$\begin{aligned} d_{j,0} + d_{j,1}\omega + ... + d_{j,n-1}\omega^{n-1} = \sigma_j &= \sum_{\substack{I \subset \{1,\ldots,m\} \\ \#I = j}} \prod_{k \in I} \sum_{\ell=1}^{n'} c_{k,\ell} v_\ell \\ &= g_{j,0} + g_{j,1}\omega + ... + g_{j,n-1}\omega^{n-1} \end{aligned}$$

where $g_{j,\ell}$ are polynomials in the mn' binary variables $c_{i,\ell}$ only. In other words, applying a Weil descent on each equation of System (2), we obtain mn new equations

$$d_{j,\ell} = g_{j,\ell}$$

in the $mn + mn'$ binary variables $c_{j,\ell}$ and $d_{j,\ell}$. The resulting system

$$\begin{cases} f'_j = 0, & 1 \leq j \leq n, \\ d_{j,\ell} = g_{j,\ell}, & 1 \leq j \leq m, 1 \leq \ell \leq n, \end{cases}$$

has $mn + n$ equations in $mn + mn'$ binary variables. As before, the system is expected to have solutions if $mn' \approx n$, and it can then be solved using a Gröbner basis algorithm.

In comparison with the simpler method of Faugère *et al.* (denoted as FPPR) [9], the number of variables is multiplied by a factor roughly $(m + 1)$. However, the degrees of our equations are also decreased thanks to the symmetrization, and this may decrease the degree of regularity of the system. In order to compare the time and memory complexities of both approaches, let D_{FPPR} and D_{Ours} be the degrees of regularity of the corresponding systems. The time and memory costs are respectively roughly N^{2D} and N^{3D}, where N is the number of variables and D is the degree of regularity . Assuming that neither D_{FPPR} nor D_{Ours} depends on n (as suggested by Petit and Quisquater's experiments [10]), that $D_{Ours} < D_{FPPR}$ (thanks to the use of symmetric variables) and that m is small enough, then the extra $(m+1)$ factors in the number of variables will be a small price to pay for large enough parameters. In practice, experiments are limited to very small n and m values. For these small parameters, we could not observe any significant advantage of this variant with respect to Faugère *et al.*'s original method. However, the complexity can be improved even further in practice with a clever choice of vector space.

3.2 A Special Vector Space

In the prime degree extension case, V cannot be a subfield, hence the symmetric variables σ_j are not restricted to V. This led us to introduce mn variables $d_{j,\ell}$ instead of mn' variables only in the composite extension degree case. However, we point out that some vector spaces may be "closer to a subfield" than other ones. In particular if V is generated by the basis $\{1, \omega, \omega^2, \ldots, \omega^{n'-1}\}$, then we have

$$if\ x_j \in V\ then\ \sigma_2 \in V'$$

where $V' \supset V$ is generated by the basis $\{1, \omega, \omega^2, \ldots, \omega^{2n'-2}\}$.

More generally, we can write

$$\begin{cases} \sigma_1 = d_{1,0} + d_{1,1}\omega + ... + d_{1,n'-1}\omega^{n'-1}, \\ \sigma_2 = d_{2,0} + d_{2,1}\omega + ... + d_{2,2n'-2}\omega^{2n'-2}, \\ \qquad \vdots \\ \sigma_m = d_{m,0} + d_{m,1}\omega + ... + d_{m,n-m}\omega^{n-m}. \end{cases}$$

Applying a Weil descent on $s'_{m+1} \mid_{x_{m+1}=x(R)}$ and each equation of System (2) as before, we obtain a new polynomial system

$$\begin{cases} f'_j = 0, & 1 \leq j \leq n, \\ d_{j,\ell} = g_{j,\ell}, & 1 \leq j \leq m, 0 \leq \ell \leq j(n'-1), \end{cases}$$

in $n + (n'-1)\frac{m(m+1)}{2} + m$ equations and $n'm + (n'-1)\frac{m(m+1)}{2} + m$ variables.

	s_{m+1}	s'_{m+1}	s'_{m+1} with specific V
variables number	mn'	$mn' + mn$	$mn' + (n'-1)\frac{m(m+1)}{2} + m$
polynomials number	n	$n + mn$	$n + (n'-1)\frac{m(m+1)}{2} + m$
degree of regularity	7 or 6	4 or 3	4 or 3

Table 1. Comparison for different multivariate polynomial system

When m is large and $mn' \approx n$, the number of variables is decreased by a factor 2 if we use our special choice of vector space instead of a random one. For $m = 4$ and $n \approx 4n'$, the number of variables is reduced from about $5n$ to about $7n/2$. For $m = 3$ and $n \approx 3n'$, the number of variables is reduced from about $4n$ to about $3n$ thanks to our special choice for V. In practice, this improvement turns out to be significant.

3.3 New Decomposition Algorithm

Our new algorithm for the decomposition problem is described in Algorithm 4. It is denoted as Imp_{Ours} in this work. The only difference between Imp_{FPPR}

Algorithm 4 **Decompose** function with binary multivariable polynomial system and symmetric elementary functions (Imp_{Ours})

Input: $R = [a]P + [b]Q$, factor base F_V
1 $F \longleftarrow$ `TransFromSemaevToBinaryWithSym(`$s_{m+1}\mid_{x_{m+1}=x(R)}$`)`
2 $F\_ \longleftarrow$ `GroebnerBasis(`F`)`
3 $sol(F) \longleftarrow$ `GetSolutionFromGroebnerBasis(`$F\_$`)`
4 $sol_m \longleftarrow \{\}$
5 **for** $e = \{P_1, P_2, .., P_m\} \in sol(F)$ **do**
6 **if** $P_1 + P_2 + ... + P_m + R = O$ **then**
7 $sol_m \longleftarrow sol_m \cup \{e\}$
8 **end**
9 **end**
Output: sol_m contains the decomposition elements of R w.r.t. F_V

and Imp_{Ours} comes from a different transformation function in the line 1 of Algorithm 4. Although the system solved in Imp_{Ours} contains more variables and equations than the system solved in Imp_{FPPR}, the degrees of the equations are smaller and they involve less monomial terms. We now describe our experimental results.

4 Experimental Results

To validate our analysis and experimentally compare our method with Faugère *et al.*'s previous work, we implemented both algorithms in Magma. All our experiments were conducted on a CPU with four AMD Opteron Processor 6276 with 16 cores, running at 2.3GHz with a L3 cache of 16MB. The Operating System was CentOS 6.3 with Linux kernel version 2.6.32-279.14.1.el6.x86_64 and 512GB memory. The programming platform was Magma V2.18-9 in its 64-bit version. Gröbner basis were computed with the *GroebnerBasis* function of Magma. Our implementations of Imp_{FPPR} and Imp_{Ours} share the same program, except that the former uses Algorithm 3 and the latter uses Algorithm 4 to set up the binary multivariate system. We first focus on the relation search, then we describe experimental results for a whole ECDLP computation.

4.1 Relation Search

The relation search is the core of both Faugère *et al.*'s algorithm and our variant. In our experiments, we considered a fixed randomly chosen curve $E_{\alpha,\beta}$, a fixed ECDLP with respect to P, and a fixed $m = 3$ for all values of the parameters n and n'. For random integers a and b, we used both Faugère *et al.*'s method and our approach to find factor basis elements $P_j \in F_V$ such that $P_1 + \cdots + P_m = [a]P + [b]Q$.

We focused on $m = 3$ (fourth Semaev's polynomial) in our experiments. Indeed, there is no hope to solve ECDLP faster than with generic algorithms using $m = 2$ because of the linear algebra stage at the end of the index calculus algorithm[5]. On the other hand, the method appears unpractical for $m = 4$ even for very small values of n because of the exponential increase with m of the degrees in Semaev's polynomials.

The experimental results are given in Table 2 and 3. For most values of the parameters n and n', the experiment was repeated 200 times and average values are presented in the table. For large values $n' = 6$, the experiment was only repeated 3 times due to the long execution time.

We noticed that the time required to solve one system varied significantly depending on whether it had solutions or not. Table 2 and 3 therefore present results for each case in separate columns. The table contains the following information: D_{reg} is the maximum degree appearing when solving the binary system with Magma's Gröbner basis routine; var is the number of $\mathbb{F}_2$ variables of the system; poly and mono are the number of polynomials and monomials in the system; rel is the average number of solutions obtained (modulo equivalent solutions through symmetries); t_{trans} and t_{groe} are respectively the time (in seconds) needed to transform the polynomial s_{m+1} into a binary system and to compute a Gröbner basis of this system; mem is the memory required by the experiment (in MB).

[5] In fact, even $m = 3$ would require a double large prime variant of the index calculus algorithm described above in order to beat generic discrete logarithm algorithms [7].

	n	n'	sol: yes							sol: no						
			D_{reg}	var	poly	mono	t_{trans}	t_{groe}	mem	D_{reg}	var	poly	mono	t_{trans}	t_{groe}	mem
Imp_{FPPR}	17	3	6	9	17	2070.59	3.95	1.08	21.51	6	9	17	2149.37	4.50	0.09	23.40
Imp_{Ours}	17	3	3	24	32	826.12	0.67	1.14	14.86	3	24	32	867.87	0.72	0.24	16.26
Imp_{FPPR}	19	3	6	9	19	2305.76	4.44	1.08	27.55	6	9	19	2401.07	4.97	0.11	29.59
Imp_{Ours}	19	3	3	24	34	912.57	0.75	1.13	19.75	3	24	34	962.67	0.79	0.31	20.90
Imp_{FPPR}	23	3	6	9	23	2792.97	5.47	1.06	29.10	6	9	23	2908.92	6.18	0.12	32.25
Imp_{Ours}	23	3	3	24	38	1079.60	0.91	1.04	15.59	3	24	38	1147.65	0.97	0.14	16.68
Imp_{FPPR}	29	3	6	9	29	3509.17	6.94	1.02	38.85	6	9	29	3669.15	7.75	0.07	43.14
Imp_{Ours}	29	3	3	24	44	1329.85	1.15	0.95	17.16	3	24	44	1427.97	1.22	0.17	18.43
Imp_{FPPR}	31	3	6	9	31	3739.76	7.38	1.03	41.12	5	9	31	3922.40	8.38	0.06	46.33
Imp_{Ours}	31	3	3	24	46	1428.49	1.24	0.90	17.59	3	24	46	1515.79	1.30	0.04	18.87
Imp_{FPPR}	37	3	6	9	37	4450.86	8.90	1.00	48.88	6	9	37	4677.23	9.99	0.06	54.81
Imp_{Ours}	37	3	3	24	52	1673.42	1.48	0.88	19.23	3	24	52	1800.79	1.58	0.05	20.85
Imp_{FPPR}	41	3	6	9	41	4921.38	9.81	0.98	54.35	6	9	41	5182.97	11.17	0.06	61.70
Imp_{Ours}	41	3	3	24	56	1847.03	1.64	0.87	20.58	3	24	56	1983.08	1.75	0.05	22.60
Imp_{FPPR}	43	3	6	9	43	5175.86	10.47	0.99	57.69	6	9	43	5436.94	11.73	0.06	64.74
Imp_{Ours}	43	3	3	24	58	1931.96	1.76	0.87	21.28	3	24	58	2076.11	1.86	0.05	23.24
Imp_{FPPR}	47	3	6	9	47	5631.62	11.29	1.00	63.77	5	9	47	5947.98	12.85	0.06	72.47
Imp_{Ours}	47	3	3	24	62	2116.38	1.92	0.83	23.17	3	24	62	2263.80	2.02	0.06	25.32
Imp_{FPPR}	53	3	6	9	53	6358.94	12.86	1.03	72.06	5	9	53	6706.36	14.57	0.07	81.22
Imp_{Ours}	53	3	3	24	68	2348.50	2.12	0.79	24.89	2	24	68	2541.59	2.28	0.04	27.52
Imp_{FPPR}	17	4	7	12	17	8997.76	15.47	6.81	58.16	7	12	17	9028.92	16.53	1.20	55.37
Imp_{Ours}	17	4	3	33	38	1622.88	1.31	3.91	31.52	3	33	38	1641.84	1.33	2.23	24.88
Imp_{FPPR}	19	4	7	12	19	9915.47	17.04	6.88	67.24	7	12	19	10072.64	17.85	1.54	64.78
Imp_{Ours}	19	4	3	33	40	1823.58	1.51	3.26	32.97	3	33	40	1823.69	1.46	1.57	27.11
Imp_{FPPR}	23	4	6	12	23	12059.19	21.06	6.83	95.66	6	12	23	12201.94	22.31	4.67	91.23
Imp_{Ours}	23	4	3	33	44	2173.29	1.83	3.19	29.63	3	33	44	2173.69	1.81	1.72	22.75
Imp_{FPPR}	29	4	6	12	29	15048.54	26.63	6.56	125.32	6	12	29	15361.50	27.80	1.37	129.78
Imp_{Ours}	29	4	3	33	50	2652.74	2.30	3.11	32.95	3	33	50	2716.43	2.29	1.06	27.88
Imp_{FPPR}	31	4	6	12	31	16130.71	28.94	3.37	136.23	6	12	31	16443.60	30.19	1.56	142.69
Imp_{Ours}	31	4	3	33	52	2839.32	2.49	3.20	35.30	3	33	52	2907.78	2.48	1.24	29.22
Imp_{FPPR}	37	4	6	12	37	19466.94	35.03	2.43	172.56	6	12	37	19611.72	35.68	0.88	176.13
Imp_{Ours}	37	4	3	33	58	3314.88	2.93	2.45	31.32	3	33	58	3437.06	2.96	0.49	32.45
Imp_{FPPR}	41	4	6	12	41	21095.65	37.58	2.79	189.16	6	12	41	21756.74	39.80	0.84	201.77
Imp_{Ours}	41	4	3	33	62	3668.86	3.24	2.23	33.84	3	33	62	3783.47	3.33	0.56	35.49
Imp_{FPPR}	43	4	6	12	43	22472.30	40.59	2.24	207.05	6	12	43	22868.33	41.39	0.85	210.59
Imp_{Ours}	43	4	3	33	64	3857.07	3.41	2.23	35.02	3	33	64	3965.76	3.48	0.60	36.51
Imp_{FPPR}	47	4	6	12	47	24264.24	43.37	2.10	225.73	6	12	47	24955.58	46.01	0.66	239.89
Imp_{Ours}	47	4	3	33	68	4197.12	3.73	2.12	37.93	3	33	68	4336.85	3.84	0.67	39.78
Imp_{FPPR}	53	4	6	12	53	27655.34	50.63	1.86	272.55	6	12	53	28043.51	52.26	0.37	279.83
Imp_{Ours}	53	4	3	33	74	4701.09	4.19	1.75	40.46	3	33	74	4824.09	4.36	0.46	42.63

Table 2. Comparison of the relation search ($m = 3$, $n' = 3, 4$) with two strategies, Imp_{FPPR} and Imp_{Ours}. D_{reg}, var, poly and mono are the degree of regularity, the number of variables, the number of polynomials and the number of monomials in the system. t_{trans} and t_{groe} are the transformation time and solving Gröbner basis time (seconds). men is the memory consumptions for solving the system (MB).

	n	n'	sol: yes							sol: no						
			D_{reg}	var	poly	mono	t_{trans}	t_{groe}	mem	D_{reg}	var	poly	mono	t_{trans}	t_{groe}	mem
Imp_{FPPR}	17	5	7	15	17	29408.19	46.53	218.87	723.08	7	15	17	29562.07	48.06	59.82	725.07
Imp_{Ours}	17	5	4	42	44	2680.14	2.21	485.10	596.46	4	42	44	2687.94	2.16	136.93	492.88
Imp_{FPPR}	19	5	7	15	19	32812.55	50.50	91.61	401.17	7	15	19	32300.00	54.03	41.80	348.01
Imp_{Ours}	19	5	4	42	46	3264.00	1.97	516.67	619.63	4	42	46	2922.50	2.67	182.92	492.82
Imp_{FPPR}	23	5	7	15	23	40168.90	64.67	70.46	475.55	7	15	23	39659.80	65.07	55.75	381.39
Imp_{Ours}	23	5	4	42	50	3572.00	3.01	157.86	323.60	4	42	50	3619.30	3.07	17.83	253.16
Imp_{FPPR}	29	5	7	15	29	50156.00	81.75	109.40	587.39	7	15	29	50403.80	80.99	50.75	530.53
Imp_{Ours}	29	5	4	42	56	4414.90	3.67	140.47	372.59	4	42	56	4356.70	3.82	20.03	278.07
Imp_{FPPR}	31	5	7	15	31	53222.10	84.08	70.64	547.86	7	15	31	53415.30	85.50	53.56	410.47
Imp_{Ours}	31	5	4	42	58	4781.80	3.99	130.07	362.76	4	42	58	4800.60	4.13	20.98	279.23
Imp_{FPPR}	37	5	7	15	37	63941.80	101.06	158.23	828.44	7	15	37	64215.10	103.29	88.29	690.51
Imp_{Ours}	37	5	3	42	64	5586.20	4.85	11.68	118.00	3	42	64	5496.80	4.87	6.85	57.52
Imp_{FPPR}	41	5	6	15	41	70895.30	113.85	230.40	889.70	7	15	41	71215.80	114.09	69.12	930.24
Imp_{Ours}	41	5	3	42	68	6042.50	5.33	13.26	126.19	3	42	68	5986.60	5.34	8.53	58.99
Imp_{FPPR}	43	5	6	15	43	75145.70	118.87	75.46	600.95	6	15	43	74671.20	118.31	39.69	615.72
Imp_{Ours}	43	5	3	42	70	6223.40	5.41	11.35	89.33	3	42	70	6470.90	5.74	8.21	56.86
Imp_{FPPR}	47	5	6	15	47	81488.60	128.63	65.03	674.87	6	15	47	81215.20	131.95	45.34	693.31
Imp_{Ours}	47	5	3	42	74	7043.30	6.07	9.57	109.38	3	42	74	7183.40	6.26	4.71	60.15
Imp_{FPPR}	53	5	6	15	53	91642.50	147.66	80.76	810.08	6	15	53	92314.60	150.41	23.31	814.76
Imp_{Ours}	53	5	3	42	80	8034.10	6.83	6.68	59.58	3	42	80	7849.50	6.96	1.36	59.91
Imp_{FPPR}	23	6	7	18	23	107008.67	163.45	3888.70	6656.13	7	18	23	105744.33	156.11	3309.43	5025.06
Imp_{Ours}	23	6	4	51	56	5270.00	4.36	5150.12	4791.31	4	51	56	5510.33	4.42	3082.15	4428.07
Imp_{FPPR}	29	6	7	18	29	136465.67	198.99	4511.74	6685.01	7	18	29	137194.33	204.07	1681.27	6528.03
Imp_{Ours}	29	6	4	51	62	6093.33	5.67	2848.46	3368.01	4	51	62	6263.33	5.76	932.65	2681.20
Imp_{FPPR}	31	6	7	18	31	145504.00	209.98	4664.25	7336.11	7	18	31	145700.33	206.29	1205.29	7276.85
Imp_{Ours}	31	6	4	51	64	6538.33	5.82	2811.99	3257.82	4	51	64	6916.67	6.09	1049.14	2616.21
Imp_{FPPR}	37	6	7	18	37	171914.33	248.24	4733.79	9777.27	7	18	37	175419.00	256.90	1126.29	9812.93
Imp_{Ours}	37	6	4	51	70	8223.00	6.77	1101.04	1327.00	4	51	70	8459.33	7.10	146.14	927.36
Imp_{FPPR}	41	6	7	18	41	189028.67	279.05	1045.53	4416.99	7	18	41	192778.33	266.44	653.92	3062.68
Imp_{Ours}	41	6	4	51	74	9297.67	7.87	953.60	1361.59	4	51	74	9246.00	8.31	87.61	896.38
Imp_{FPPR}	43	6	7	18	43	203094.33	298.13	1444.41	4288.28	7	18	43	199325.67	280.46	787.02	3796.57
Imp_{Ours}	43	6	4	51	76	9899.33	8.02	920.13	1340.39	4	51	76	8958.33	8.33	82.37	918.05
Imp_{FPPR}	47	6	7	18	47	222208.67	326.22	1278.79	4524.33	7	18	47	221999.67	326.08	463.62	3287.07
Imp_{Ours}	47	6	4	51	80	10789.00	9.06	858.66	1296.09	4	51	80	10438.33	9.24	80.54	919.39
Imp_{FPPR}	53	6	7	18	53	245891.33	366.92	2967.03	7311.44	7	18	53	248212.33	359.03	1857.65	6677.92
Imp_{Ours}	53	6	3	51	86	11748.00	10.48	34.82	151.04	3	51	86	11744.00	10.70	31.21	151.02

Table 3. Comparison of the relation search ($m = 3$, $n' = 5, 6$) with two strategies, Imp_{FPPR} and Imp_{Ours}.

n	$\#E_{\alpha,\beta}$	Imp_{FPPR}	Imp_{Ours}
7	4*37	1.574	0.864
11	4*523	8.625	6.702
13	4*2089	49.698	31.058
17	4*32941	2454.470	1364.742
19	4*131431	22474.450	9962.861

Table 4. Comparison of two ECDLP strategies, Imp_{FPPR} and Imp_{Ours}. The last two columns are computing time in seconds.

	probability to get an answer $\frac{2^{mn'}}{m!2^n}$	complexity $N^{\omega D}$
m increases	probability increases	D increases, N increases.
n' increases	probability increases	N increases.

Table 5. Trade-off for choosing m and n'. N: total number of variables. D: degree of regularity.

The experiments show that the degrees of regularity of the systems occurring during the relation search are decreased from values between 6 and 7 in Faugère *et al.*'s method to values between 3 and 4 in our method. This is particularly important since the complexity of Gröebner basis algorithms is exponential in this degree. However, this huge advantage of our method comes at the cost of a significant increase in the number of variables, which itself tends to increase the complexity of Gröbner basis algorithms. Our experimental results confirm the analysis of Section 3: while our method may require more memory and time for small parameters (n, n'), it becomes more efficient than Faugère *et al.*'s method when the parameters increase. We remark that although the time required to *solve* the system may be larger with our method than with Faugère *et al.*'s method for small parameters, the time required to *build* this system is always smaller. This is due to the much simpler structure of s'_{m+1} compared to s_{m+1} (lower degrees and less monomial terms).

4.2 Whole ECDLP Computation

In a next step, we also implemented the whole ECDLP algorithm with the two strategies Imp_{FPPR} and Imp_{Ours}. For n in $\{7, 11, 13, 17, 19\}$, we ran the whole attack using $m = 3$ and several values for n'. The orders of the curves we picked in our experiments are shown in Table 4 together with the experimental results for the best value of n', which turned out to be 3 in all cases. Timings provided in the table are in seconds and averaged over 20 experiments. Table 4 clearly shows that our method (Imp_{Ours}) is more efficient than Faugère *et al.*'s method (Imp_{FPPR}).

It may look strange that $n' = 3$ leads to optimal timings at first sight. Indeed, the ECDLP attacks described above use $mn' \approx n$ and a constant value for n'

leads to a method close to exhaustive search. However, this is consistent with the observation already made in [9, 10] that exhaustive search is more efficient than index calculus for small parameters. Table 5 also shows that while increasing n' increases the probability to have solutions, it also increases the complexity of the Gröebner basis algorithm. This increase turns out to be significant for small parameters.

5 Conclusion and Future work

In this paper, we proposed a variant of Faugère *et al.*'s attack on the binary elliptic curve discrete logarithm problem (ECDLP). Our variant takes advantage of the symmetry of Semaev's polynomials to compute relations more efficiently. While symmetries had also been exploited in similar ECDLP algorithms for curves defined over finite fields with *composite* extension degrees, our method is the first one in the case of extension fields with *prime* extension degrees, which is the most interesting case for applications.

At Asiacrypt 2012, Petit and Quisquater estimated that Faugère *et al.*'s method would beat generic discrete logarithm algorithms for any extension degree larger than roughly 2000. We provided heuristic arguments and experimental data showing that our method reduces both the time and the memory required to compute a relation in Faugère *et al.*'s method, unless the parameters are very small. Our results therefore imply that Petit and Quisquater's bound can be lowered a little.

Our work raises several interesting questions. On a theoretical side, it would be interesting to prove that the degrees of regularity of the systems appearing in the relation search will not rise when n increases (in all our experiments for various parameter sizes, they were equal to either 3 or 4). It would also be interesting to provide a more precise analysis of our method and to precisely estimate for which values of the parameters it will become better than Faugère *et al.*'s method.

On a practical side, it would be interesting to improve the resolution of the systems even further. One idea in that direction is pre-computation. The relation search involves solving a large number of closely related systems, where only the value $x(R)$ changes from one system to the other. The transformation of Semaev's polynomial into a binary multivariate system could therefore be done in advance, and its cost be neglected. In fact, even the resolution of the system could potentially be improved using special Gröebner basis algorithms such as F_4 trace [18, 14]. A second direction on the practical side is parallelization. A powerful feature of Pollard's ρ method and its variants is their highly-parallelized structure. Since our method saves memory compared to Faugère *et al.*'s method, it is also more suited to parallelization.

Using Gröbner basis algorithms to solve ECDLP is a very recent idea. We expect that the index calculus algorithms that have recently appeared in the literature will be subject to further theoretical improvements and practical optimizations in a close future.

References

1. National Security Agency: The case for elliptic curve cryptography. `http://www.nsa.gov/business/programs/elliptic_curve.shtml` (January 2009)
2. Shanks, D.: 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), Providence, R.I. (1971) 415–440
3. Pollard, J.M.: A Monte Carlo method for factorization. BIT Numerical Mathematics **15 (3)** (1975) 331–334
4. Brent, R.P.: An improved Monte Carlo factorization algorithm. BIT Numerical Mathematics **20** (1980) 176–184
5. Pollard, J.M.: Kangaroos, monopoly and discrete logarithms. Journal of Cryptology **13** (2000) 437–447
6. Diem, C.: An index calculus algorithm for plane curves of small degree. In Hess, F., Pauli, S., Pohst, M.E., eds.: ANTS. Volume 4076 of Lecture Notes in Computer Science., Springer (2006) 543–557
7. Gaudry, P.: Index calculus for abelian varieties of small dimension and the elliptic curve discrete logarithm problem. Journal of Symbolic Computation **44**(12) (2009) 1690 – 1702
8. Diem, C.: On the discrete logarithm problem in elliptic curves. Compositio Mathematica **147** (2011) 75–104
9. Faugère, J.C., Perret, L., Petit, C., Renault, G.: Improving the complexity of index calculus algorithms in elliptic curves over binary field. In: Proceedings of Eurocrypt 2012. Volume 7237 of Lecture Notes in Computer Science., Springer Verlag (2012) 27–44
10. Petit, C., Quisquater, J.J.: On polynomial systems arising from a Weil descent. In Wang, X., Sako, K., eds.: Advances in Cryptology ASIACRYPT 2012. Volume 7658 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2012) 451–466
11. Joux, A., Vitse, V.: Elliptic curve discrete logarithm problem over small degree extension fields. Journal of Cryptology (2011) 1–25
12. Faugère, J.C., Gaudry, P., Huot, L., Renault, G.: Using symmetries in the index calculus for elliptic curves discrete logarithm. IACR Cryptology ePrint Archive **2012** (2012) 199
13. Semaev, I.: Summation polynomials and the discrete logarithm problem on elliptic curves. IACR Cryptology ePrint Archive **2004** (2004) 31
14. Faugère, J.C.: A new efficient algorithm for computing Gröbner bases (F_4). Journal of Pure and Applied Algebra **139**(1-3) (1999) 61–88
15. Faugère, J.C.: A new efficient algorithm for computing Gröbner bases without reduction to zero (F_5). In: Proceedings of the 2002 international symposium on Symbolic and algebraic computation. ISSAC '02, New York, NY, USA, ACM (2002) 75–83
16. Faugère, J., Gianni, P., Lazard, D., Mora, T.: Efficient computation of zero-dimensional Gröbner bases by change of ordering. Journal of Symbolic Computation **16**(4) (1993) 329 – 344
17. Joux, A., Vitse, V.: Cover and decomposition index calculus on elliptic curves made practical - application to a previously unreachable curve over $\mathbb{F}_{p^6}$. In Pointcheval, D., Johansson, T., eds.: EUROCRYPT. Volume 7237 of Lecture Notes in Computer Science., Springer (2012) 9–26
18. Joux, A., Vitse, V.: A variant of the F4 algorithm. In Kiayias, A., ed.: CT-RSA. Volume 6558 of Lecture Notes in Computer Science., Springer (2011) 356–375