On Generalized First Fall Degree Assumptions

Yun-Ju Huang[1], Christophe Petit[2], Naoyuki Shinohara[3], and Tsuyoshi Takagi[4,5]

[1] Information Securty Laboratory, Institute of Systems, Information Technologies and Nanotechnologies (ISIT)
cs@isit.or.jp
[2] University College London, Information Security Group
christophe.petit@ucl.ac.uk
[3] National Institute of Information and Communications Technology (NICT)
[4] Institute of Mathematics for Industry, Kyushu University
[5] CREST, Japan Science and Technology Agency

Abstract. The first fall degree assumption provides a complexity approximation of Gröbner basis algorithms when the degree of regularity of a polynomial system cannot be precisely evaluated. Most importantly, this assumption was recently used by Petit and Quisquater's to conjecture that the elliptic curve discrete logarithm problem can be solved in subexponential time for binary fields (binary ECDLP). The validity of the assumption may however depend on the systems in play.
In this paper, we theoretically and experimentally study the first fall degree assumption for a class of polynomial systems including those considered in Petit and Quisquater's analysis. In some cases, we show that the first fall degree assumption seems to hold and we deduce complexity improvements on previous binary ECDLP algorithms. On the other hand, we also show that the assumption is unlikely to hold in other cases where it would have very unexpected consequences.
Our results shed light on a Gröbner basis assumption with major consequences on several cryptanalysis problems, including binary ECDLP.

Keywords: Elliptic curves, Discrete logarithm problem, Index calculus, Semaev's summation polynomials, Multivariate polynomial systems, Gröbner basis

Remark This paper has been developed between February 2013 and now, following a visit of the second author to Kyushu University, and has appeared in the first author's Ph.D thesis [28]. Very recently, we have learned that three other teams had been developing partly similar ideas independently, including [42,31,32]. Compared to their attacks, ours uses different vector spaces for each coordinate of the summation polynomial, resulting in an improved complexity. We also take a different point of view by focusing on the first fall degree assumption.

1 Introduction

Solving polynomial systems of equations is a fundamental problem with many applications in engineering, computer science and mathematics. Gröbner basis algorithms are one of the main tools to solve polynomial systems. When the number of solutions is "small", the runtime complexity of these algorithms can be approximated by $\binom{n+D_{reg}}{D_{reg}}^{\omega}$ [5], where n is the number of variables and D_{reg} is a characteristic of the system called the *degree of regularity.* While the value of this degree for "generic" systems is well-understood, evaluating the degree of regularity of systems with particular distributions is usually a very challenging task.

In cryptanalysis applications, the degree of regularity has sometimes been approximated by another parameter of the system called the *first fall degree*, which is usually easier to evaluate [20,26,14,13,37,27]. Most notably, the first fall degree assumption (approximating the degree of regularity of a system by its first fall degree) was used by Petit and Quisquater to support a conjecture that the binary elliptic curve discrete logarithm problem (ECDLP) can be solved in subexponential time using an index calculus algorithm of Diem [37,11]. Follow-ups of this work [29,35,23,30] have used the same assumption. In [30], the experiments even comes to $n = 53$.

The first fall degree assumptions is supported in most of these cases by experimental results as well as by some intuition on the behaviour of Gröbner basis algorithms. On the other hand, it is also well-known that the approximation may be very bad for specific systems.

The case of *polynomial systems arising from a Weil descent* is of particular interest as they are central in Petit and Quisquater's analysis. These systems are obtained by projecting a single polynomial f over a finite field $\mathbb{F}_{2^n}$ onto the n components of a basis of $\mathbb{F}_{2^n}$ over $\mathbb{F}_2$ (see Section 2.3). The validity of the first fall degree assumption in this case remains an open problem, even if a rigorous proof of its validity when f is univariate has been provided in [36].

1.1 Contributions

In this paper, we study the theoretical implications and the experimental validity of the first fall degree assumption for polynomial systems obtained by applying a Weil descent on a *set of polynomials* f_i over $\mathbb{F}_{2^n}$ instead of a *single polynomial* over $\mathbb{F}_{2^n}$ as in [21,37].

We first consider the case of polynomials over $\mathbb{F}_{2^n}$ that can be written as the resultant of two polynomials with lower degrees. Our generalization of the first fall degree assumption to sets of polynomials leads to a new, more efficient algorithm for this particular case. This directly impacts binary ECDLP since the polynomial in play in this application is a resultant of several lower degree polynomials.

When the number of polynomials f_i remains small, our experimental results provide some support for this generalized first fall degree assumption. In particular, we obtain significant timing improvements over previous works [37,29,30] for the computation of relations for binary ECDLP, and we could complete the computation for some parameters that were previously intractable. Most importantly, the degrees of regularity were consistently smaller with our methods than with previous methods.

When the number of polynomials f_i increases, our experimental results suggest a moderate increase in the degree of regularity as well, suggesting that the first fall degree assumption should be relaxed in that case.

Finally when the number of polynomials f_i is as large as n, we show that the first fall degree assumptions implies some results that seem unlikely to be true. In particular, we describe algorithms for binary ECDLP, two algorithms for DLP and ECSSP that all run in *expected polynomial time* under our generalized first fall degree assumptions. Even more, we argue that 3SAT can also be seen as a particular instance of our systems, which is even stronger evidence that the generalized first fall degree assumption must be false in that case. Surprisingly, the experimental results on our binary ECDLP and DLP algorithms do provide some support for our analysis, but the instances we could solve using our methods are ridiculously small compared to what can be solved with other methods.

1.2 Outline

This paper is organized as follows. We provide preliminary background on Gröbner basis algorithms and the recent index calculus developments for binary ECDLP in Section 2. We detail our idea for resultant polynomials in Section 3 and we describe further applications of the generalized first fall degree assumption to binary ECDLP, binary DLP, ECSSP and 3-SAT in Section 4. We provide our experimental results in Section 5. We then conclude the paper in Section 6.

2 Preliminaries

2.1 Gröbner Basis Algorithms

Let K be a field, let $R = K[x_1, \ldots, x_N]$ be a polynomial ring over K and let $f_1, \ldots, f_n \in R$. By solving the polynomial system $\mathcal{F} := \{f_i = 0\}_{1 \leq i \leq n}$, we mean finding one, several or all the tuples $\bar{x} = (x_1, \ldots, x_N) \in K^N$ such that $f_i(\bar{x}) = 0$ for all i. We will usually assume that the number of solutions is "small" in this paper. When all polynomials f_i have degree 1, this problem can be solved in time polynomial in n and N

using Gaussian elimination to derive a triangular system. In general however, polynomial system solving is a daunting computational task.

Gröbner basis algorithms are one of the most efficient tools to solve polynomial systems of equations. Given a monomial ordering on K, a Gröbner basis for an ideal $I \in R$ is a set of polynomials $\{g_i\} \subset I$ such that for any $f \in I$, there exists $g_i \in I$ such that $LT(g_i)|LT(f)$. A Gröbner basis for I allows for an efficient resolution of the ideal membership problem (given $f \in R$, decide whether $f \in I$). For a lexicographic ordering, a Gröbner basis will contain a "triangular" set of polynomials involving less and less variables, allowing to solve the system "one variable at the time".

In practice, computing a Gröbner basis seems easier for a graded ordering than for a lexicographic ordering. Given a set of polynomials $\{f_i\}$, the usual strategy is to first compute a Gröbner basis of their ideal for the graded reverse lexicographic ordering using F4 or F5 [16,17], then to convert it into a Gröbner basis for a lexicographic order using FGLM [18] or Gröbner walk [8], and finally to solve this "triangular" system for one variable at the time. The complexity of the second step is negligible when the number of solutions is small, so we will neglect it in this paper and focus on the first step.

When the system is defined over a "small" finite field of size q, adding the field equations $x_i^q - x_i = 0$ to it before computing the graded reverse lexicographic ordering Gröbner basis will generally accelerate the resolution.

Most Gröbner basis algorithms can be seen as reducing the general problem to the simpler case of linear polynomials, where it can be solved using Gaussian elimination [33]. The input polynomials f_i are multiplied by all monomials m_j such that $g_{ij} := m_j f_i \leq d$ has degree smaller than some degree d. The coefficients of the polynomials g_{ij} are encoded in a large *Macaulay matrix*, one row per polynomial and one column per monomial term. Gaussian elimination on this matrix (elimination the largest monomial terms first) will then produce new polynomials. When d is large enough, it can be proved that this strategy will produce a set of polynomials whose linear combinations generate all polynomials of degree at most d in the ideal: a Gröbner basis is then easily deduced.

Generic bounds are known for the minimal d to use in this strategy, but they tend to be very loose for particular systems. In practice, Gröbner basis algorithms will start at the maximal degree d of the equations in the system and perform a first Gaussian elimination step. If a Gröbner basis is found they will stop. If new, lower degree equations are found, they will multiply these new equations by monomial terms and add all resulting equations to the system, perform a new Gaussian elimination step and iterate. If no new equations of lower degree are found they will increase the degree, add new equations accordingly, perform a Gaussian elimination step, and iterate. This strategy takes maximal advantage of low degree equations found during the computation, and in some cases may reduce the maximal degree d needed. The most efficient Gröbner basis algorithms such as F4 or F5 [16,17] use block Gaussian elimination algorithms and prevent trivial reductions to the zero polynomials to occur.

The overall cost of these algorithms is determined by the cost of Gaussian elimination. With N variables at degree d, the number of monomials at degree d is $\binom{N+d}{d}$, which can be approximated by N^d when $d << N$. The time and memory complexities of Gröbner basis algorithms can therefore be approximated by $N^{\omega D_{reg}}$ and $N^{2D_{reg}}$ where ω is the linear algebra constant and the *degree of regularity* D_{reg} is the maximal degree occuring during the computation.

The degree of regularity can be precisely evaluated for random systems, but known bounds can be very loose for specific families of polynomial systems, and deriving better bounds for these systems seems very difficult. In several previous cryptanalysis works, the degree of regularity has therefore been approximated by the *first fall degree.* We said that D_{ff} is the first fall degree of the system $\{f_{ij}\}$ if for each f_{ij} exists some g_{ij} such that $D_{ff} = \min\{deg(g_{ij}f_{ij})\}$ where $deg(\sum g_{ij}f_{ij}) < deg(g_{ij}f_{ij})$ and $\sum g_{ij}f_{ij}$ is not zero. [37].

The *first fall degree assumptions* says that the degree of regularity approximation obtained by this way is reasonably good. This is intuitively justified as follows: when degree falls start to occur in a Gröbner basis computation, they will produce new polynomials of lower degrees, which can in turn be multiplied by all monomials to obtain new polynomials, which can contribute to even more degree falls later, etc.

The intuition seems even better when not only one, but many degree fall occurs at degree D_{ff}, since all these low degree polynomials are likely to contribute to even more degree falls later.

It is well-known that the first fall degree assumption can fail for some specific polynomial systems, but this fact does not prevent it to hold with a large probability for randomly generated systems of a particular type. It certainly holds for "random" systems (for which both degrees are expected to be $\sum_i(\deg f_i - 1) + 1$) and there is some experimental evidence that it holds for polynomial systems arising from a Weil descent on one polynomial (see Section 2.3 below). In this paper, we will consider the first fall degree assumption on a larger class of polynomial systems arising from a Weil descent on a *system* of polynomials.

2.2 Index Calculus

Let G be a finite cyclic group and let g be a generator of G. Let h be another element in G. The *discrete logarithm problem* asks for an integer k such that $h = [k]g$. The hardness of this problem underlies the security of many cryptographic schemes, including for example El Gamal cryptosystem and Diffie-Hellman key exchange [15,12]. Although generic algorithms such as Pollard's rho and BSGS algorithms [39,43] can solve this problem in any group with a complexity $O(\sqrt{|G|})$, the difficulty of this problem highly depends on the particular group used. The most popular groups in cryptography are multiplicative groups of finite fields and cyclic subgroups of elliptic curves or hyperelliptic curves over finite fields.

Index calculus algorithms have been the most successful algorithms against the discrete logarithm problems used in practice. These algorithms first define a *factor basis* $\mathcal{F} \subset G$. In their simplest variants, they then construct about $|\mathcal{F}|$ relations

$$[a_i]g + [b_i]h = \sum_{f_i \in \mathcal{F}} [e_i]f_i.$$

Finally, Gaussian eliminination is performed on these relations to obtain a relation

$$[a]g + [b]h = 0,$$

from which the discrete logarithm is easily recovered. More elaborate index calculus algorithms may first compute the discrete logarithms of all the elements in the factor basis, then use a dedicated *descent* strategy to compute the discrete logarithm of a particular target element h.

Index calculus algorithms are particularly efficient for the multiplicative groups of finite fields. The most efficient index calculus algorithms can now solve discrete logarithm problems in subexponential time

$$L_q(c, 1/3) = \exp\left(c(\log q)^{1/3}(\log\log q)^{2/3}\right)$$

in any finite field $\mathbb{F}_q$ (for some constant c), and in quasi-polynomial time for finite fields with small to medium characteristics [9,25,40,1,4]. Index calculus algorithms have also been proposed for hyperelliptic curves; their complexity is subexponential for families of curves with increasing genus [2].

Index calculus algorithms differ from each other in the definition of the factor basis and in the algorithm used to compute relations. For finite fields, the factor basis naturally consists of all the "small" elements in some canonical representation, hence the relation search amounts to finding elements $[a_i]g+[b_i]h$ whose canonical representation is smooth. A similar definition can be taken for hyperelliptic curves. On the other hand, elliptic curves elements do not have a "natural" definition of smoothness.

The first step in the design of index calculus algorithms for elliptic curves was obtained by Semaev who introduced his *summation polynomials* [41].

Proposition 1. *Let E be an elliptic curve defined over K. For any integer m, there exists a unique* summation polynomial $S_m \in K[x_1, \ldots, x_m]$ *such that*

$$S_m(x_1, \ldots, x_m) = 0 \Leftrightarrow \exists y_i \in K \text{ with } (x_i, y_i) \in E \text{ and } \sum_i (x_i, y_i) = 0.$$

Semaev showed how to compute these polynomials recursively with resultants. Defining $\mathcal{F}_V := \{(x,y) \in E | x \in V\}$ for some $V \subset K$, finding a relation involving the point $(X,Y) = [a_i]g + [b_i]h$ amounts to solving the polynomial equation $S_{m+1}(x_1, \ldots, x_m, X) = 0$ together with the constraints $x_j \in V$.

When $K := \mathbb{F}_{q^m}$, Gaudry [24] and Diem [10] later proposed to use $V := \mathbb{F}_q$. For any basis $\{\theta_1, \ldots, \theta_m\}$ of $\mathbb{F}_{q^m}/\mathbb{F}_q$, we can write $S_{m+1}(x_1, \ldots, x_m, X) = \sum_\ell f_\ell(x_1, \ldots, x_m)\theta_\ell$ for some polynomials f_ℓ with coefficients in $\mathbb{F}_q$. The equation $S_{m+1}(x_1, \ldots, x_m, X) = 0$ and the constraints $x_j \in \mathbb{F}_q$ then lead to a system of polynomial equations $f_\ell(x_1, \ldots, x_m) = 0$ over the subfield $\mathbb{F}_q$. This transformation is usually called a *Weil descent* or *Weil restriction*. This system can then be solved using resultants, Gröbner basis algorithms, or any other technique. Gaudry and Diem obtained improvements over generic algorithms in this way. Diem [11] also proposed to take V any vector subspace instead of a subfield. This turned out to be particularly useful when no exploitable subfield is available, such as elliptic curves over binary fields $\mathbb{F}_{2^n}$ with n prime.

In 2012, Faugère, Perret, Petit and Renault [21] and Petit and Quisquater [37] revisited Diem's algorithm for binary curves [11]. While Diem had previously analyzed his algorithm using generic bounds on the complexity of solving polynomial systems, they showed that the systems arising from a Weil restriction were in fact much easier to solve than generic systems. Petit and Quisquater even conjectured that binary ECDLP can be solved in subexponential time

$$2^{\omega T} \text{ where } T \approx 2n^{2/3} \log n.$$

Their conjecture is supported by assumption arguments and experimental results for small parameters. They estimated that Diem's algorithm could beat generic ones when n is larger than about 2000 [37].

2.3 Polynomial Systems Arising from a Weil Descent

The core of Petit and Quisquater's analysis is a study of the following problem: given a polynomial f in m variables over $\mathbb{F}_{2^n}$ and given a vector space $V \subset \mathbb{F}_{2^n}$ of dimension n', find a tuple $(x_1, \ldots, x_m)$ such that $x_i \in V$ and $f(x_1, \ldots, x_m) = 0$. For simplicity they assume that f has degree bounded by $2^t - 1$ in each variable, and they also assume $mn' \approx n$ such that the expected number of solutions is small.

The standard way to find such a solution is to apply a *Weil descent* on the variables. We first define small field variables x_{ij} such that $x_i = \sum_{i=1}^{n'} x_{ij} v_j$ and we substitute the x_i variables by these linear relations in f. For any basis $\{\theta_1, \ldots, \theta_n\}$ of $\mathbb{F}_{2^n}/\mathbb{F}_p$, the equation $f = 0$ then leads to n polynomial equations $f_\ell(x_{ij}) = 0$ such that $f(x_1, \ldots, x_m) = \sum_{\ell=1}^{n} f_\ell(x_{ij}, y_{ij})\theta_\ell$. We finally use Gröbner basis algorithms to solve the polynomial system

$$\begin{cases} f_\ell(x_{ij}, y_{ij}) = 0 & \ell = 1, \ldots, n \\ x_{ij}^2 - x_{ij} = 0 & 1 \leq i \leq m, 1 \leq j \leq n' \end{cases} \tag{1}$$

where the last equations are field equations. It is easy to see that the polynomials f_ℓ have degree bounded by t in each block of variables (each block corresponding to one variable over $\mathbb{F}_{2^n}$) and by mt in total.

It turns out that for this type of systems, many degree falls occur already at degree $mt + 1$. This could be shown in [21,37] by considering the Weil descent of a polynomial mf (where m is an arbitrary monomial of degree 1) and in [27] using different techniques. Under the first fall degree assumption, the degree of regularity of the system is therefore close to $mt + 1$ as well, whereas generic systems with the same degrees would be expected to have much larger degrees of regularity.

The first fall degree assumption implications on the degree of regularity were experimentally verified in [37] both for "generic" polynomials f and for Semaev's polynomials, but only for sufficiently small parameters. Further evidence in favor of the first fall degree assumption for polynomial systems arising from a Weil descent comes from various assumption arguments [20,26,14,13,6] and a proof [36] that it holds in the case of a univariate polynomial f.

Petit and Quisquater then showed that the degree of regularity of the Weil descent systems arising in Diem's algorithm will be bounded by roughly m^2 under the first fall degree assumption. The overall cost of Diem's algorithm can then be approximated by

$$2^{O(m^2)} + m!2^{n'}n^{O(m^2)} + 2^{O(n')}.$$

The first term in this formula comes from the computation of the $(m+1)$th summation polynomial, the second term comes from the relation search and the last term comes from linear algebra. In the second term, the $n^{O(m^2)}$ factor comes from the Gröbner basis resolution of one system, the $2^{n'}$ accounts for the number of relation needed, and the $m!$ factor comes from the probability that one random point leads to a relation given the symmetry of summation polynomials [24,11]. Using $n' \approx n^{2/3}$ and $m \approx n^{1/3}$, the algorithm will run in time $2^{O(n^{2/3}\log n)}$, a major progress compared to previous generic, exponential-time algorithms.

Further experimental results supporting this analysis were presented in [44] and variants of the attack were studied in [29,23,30]. An extension of the attack to hyperelliptic curves was also presented in [35]. All these results rely on some form of the first fall degree assumption.

3 Further Consequences of First Fall Degree Assumptions

In this section, we consider the theoretical implications of first fall degree assumptions when they are generalized even further by applying the resultant structure of Semaev polynomial. In our new approach, the first fall degree decrease more than the previous work.[21,37,29,44,23,30]. We also show the experimental evidence of the first fall degree assumptions in the later section.

3.1 Splitting up the Resultant

Let n be a positive integer and let $p = 2$. Let $f^{(1)}, f^{(2)}$ be two multivariate polynomials over the field $\mathbb{F}_{p^n}$, respectively with $m_1 + 1$ and $m_2 + 1$ variables. To simplify the exposition of our idea, we will assume that $m_1 = m_2 = m$ and that the degrees of $f^{(1)}$ and $f^{(2)}$ are bounded by D with respect to all variables individually. Let

$$f(x_1, \ldots, x_m, y_1, \ldots, y_m) := \mathrm{Res}_z X(f^{(1)}(x_1, \ldots, x_m, z), f^{(2)}(y_1, \ldots, y_m, z)).$$

The polynomial f has degree bounded by $2D$ with respect to all its variables.

Let $n' \approx n/2m$ and let $V = \langle v_1, \ldots, v_{n'}\rangle \subset \mathbb{F}_{p^n}$ be a vector space of dimension n' over $\mathbb{F}_p$, generated by $\{v_1, \ldots, v_{n'}\}$. For "random" polynomials $f^{(1)}$ and $f^{(2)}$ with the above characteristics, we expect the equation $f(x_1, \ldots, x_m, y_1, \ldots, y_m) = 0$ to have about one solution such that $x_i \in V$. As recalled in Section 2.3, the standard way to find such a solution is to introduce small field variables to model the vector space constraints and to apply a Weil descent on f.

To exploit the resultant structure of f, we suggest to introduce n additional small field variables z_i such that $z = \sum_{i=1}^{n} z_i\theta_i$, and to perform a Weil restriction on $f^{(1)}$ and $f^{(2)}$ separately instead of doing the Weil restriction on their resultant. This leads to a polynomial system

$$\begin{cases} f_\ell^{(1)}(x_{ij}, z_j) = 0 & \ell = 1, \ldots, n \\ f_\ell^{(2)}(y_{ij}, z_j) = 0 & \ell = 1, \ldots, n \\ x_{ij}^p - x_{ij} = 0 & 1 \le i \le m, 1 \le j \le n' \\ y_{ij}^p - y_{ij} = 0 & 1 \le i \le m, 1 \le j \le n' \\ z_j^p - z_j = 0 & 1 \le j \le n. \end{cases} \tag{2}$$

Note that the polynomials f_ℓ^k have degree bounded by $\lceil \log_p(D) \rceil$ in each block of variables, and by $(m+1)\lceil \log_p(D) \rceil$ in total.

3.2 Complexity Analysis

Following the reasoning of [21,37], it is easy to see that the first degree fall relations will appear at degree $1 + \max(\deg f_\ell^{(1)}, \deg f_\ell^{(2)})$ for System (2). For the parameters chosen above, the first fall degree of our new system is therefore bounded by $1 + (m+1)(p-1)\lceil \log_p(D) \rceil$.

We summarize the main characteristics of both the FPPR method and our new method in Table 1. For $2mn' \sim n$ and large m, the new method requires about twice as many variables, but on the other hand the first fall degree is about twice as small.

Table 1: Comparing previous and new methods

	Previous method	New method
Number of small field variables	$2mn'$	$2mn' + n$
First fall degree	$1 + 2m\lceil \log_2(2D) \rceil$	$1 + (m+1)\lceil \log_2(D) \rceil$

A rigorous estimation of the degree of regularity of System (2) appears out of reach today since we do not even have a good rigourous bound for the degree of regularity of System (1). We can nevertheless prove the following partial result.

Lemma 1. *The degree of regularity System* (2) *is at most as large as the degree of regularity of System* (1).

Proof. Since $f(x_1, \ldots, x_m, y_1, \ldots, y_m) := \mathrm{Res}_z(f^{(1)}(x_1, \ldots, x_m, z), f^{(2)}(y_1, \ldots, y_m, z))$, there exist polynomials $g^{(1)}, g^{(2)}$ such that

$$\begin{aligned} f(x_1, \ldots, x_m, y_1, \ldots, y_m) = {} & f^{(1)}(x_1, \ldots, x_m, z) g^{(1)}(x_1, \ldots, x_m, y_1, \ldots, y_m, z) \\ & + f^{(2)}(y_1, \ldots, y_m, z) g^{(2)}(x_1, \ldots, x_m, y_1, \ldots, y_m, z). \end{aligned}$$

Applying a Weil restriction on all components, we have

$$\begin{aligned} \sum_{i=1}^{n} f_\ell(x_1, \ldots, x_m, y_1, \ldots, y_m)\theta_\ell = {} & \left(\sum_{k=1}^{n} f_k^{(1)}(x_1, \ldots, x_m, z)\theta_k \right) \left(\sum_{k=1}^{n} g_k^{(1)}(x_1, \ldots, x_m, y_1, \ldots, y_m, z)\theta_k \right) \\ & + \left(\sum_{k=1}^{n} f_k^{(2)}(y_1, \ldots, y_m, z)\theta_k \right) \left(\sum_{k=1}^{n} g_k^{(2)}(x_1, \ldots, x_m, y_1, \ldots, y_m, z)\theta_k \right). \end{aligned}$$

Multiplying and rearranging terms in the right-hand side, we deduce

$$\begin{aligned} f_\ell(x_1, \ldots, x_m, y_1, \ldots, y_m) = {} & \sum_k h_{k\ell}^{(1)}(x_1, \ldots, x_m, y_1, \ldots, y_m, z) f_k^{(1)}(x_1, \ldots, x_m, z) \\ & + \sum_k h_{k\ell}^{(2)}(x_1, \ldots, x_m, y_1, \ldots, y_m, z) f_k^{(2)}(y_1, \ldots, y_m, z) \end{aligned}$$

for some $h_{k\ell}^{(1)}$ and $h_{k\ell}^{(2)}$, modulo the field equations. This shows that the polynomials of System (1) are algebraic combinations of the equations of System (2) at degree $\max \deg h_{k\ell}^{(i)} f_k^{(i)}$.

This bound might not be tight. Assuming that the first fall degree is (also) a reasonable approximation of the degree of regularity for systems of the form (2), the degree of regularity of the new system is also roughly twice as small as the degree of regularity of the previous system. Since the complexity of Gröbner basis algorithms depends polynomially on the number of variables but exponentially on the degree of regularity, the new method will perform significantly better than the previous one when n is large enough. Detailed experimental results are provided in Sections 5.4.

3.3 Application to ECDLP

For an elliptic curve with equation $y^2 + xy = x^3 + a_2x + a_6$ over $\mathbb{F}_{2^n}$, the second and third summations polynomials are defined by $S_2(x_1, x_2) = x_1 + x_2$ and $S_3(x_1, x_2, x_3) = x_1^2x_2^2 + x_1^2x_3^2 + x_2^2x_3^2 + x_1x_2x_3 + a_6$. The next summation polynomials can be recursively defined by

$$S_r(x_1, \ldots, x_r) = \mathrm{Res}_X(S_{r-k}(x_1, \ldots, x_{r-k-1}, X), S_{k+2}(x_{r-k}, \ldots, x_r, X))$$

for an arbitrary integer $k \in [1, r - 3]$.

Although the asymptotic analysis in [37] suggests using $m = n^{1/3}$ for optimal efficiency, the huge memory requirements following from a (predicted) degree of regularity approximately equal to m^2 have so far limited m to either 3 or 4 in all experiments. In fact even when additional symmetries coming from composite extension degrees are exploited, the maximal value of m reportedly used in experiments is 6, and the largest summation polynomial computed is S_8 [19]!

Splitting the resultant as above can potentially allow for much larger m values, without even needing to computing the corresponding summation polynomial. To compute a relation in Diem's algorithm with $m = 3$, one can apply a Weil descent on both polynomials in the set $\{S_3(x_1, x_2, w), S_3(w, x_3, X)\}$ and solve the resulting system using Gröbner basis algorithms. Assuming that the first fall degree assumptions works in both cases, the degree of regularity of the last system will be approximately equal to its first fall degree, which will be approximately equal to the degree of regularity of a Weil descent system obtained from the single, smaller polynomial $S_3(x_1, x_2, w)$.

This reasoning can be extended to larger m values. Assuming that the first fall degree assumptions works in all cases, splitting the $(m + 1)$th summation polynomial into $m - 1$ summation polynomials S_3 will keep the degree of regularity bounded by a constant D. The method will introduce $m - 2$ variables w_i that are unconstrained over $\mathbb{F}_{2^n}$ (corresponding to x-coordinates of partial sums of the first points involved in the summation), so solving the $(m+1)$th summation polynomial in that way would then cost roughly $(mn)^D$, which is polynomial in n.

Assuming that the first fall degree assumptions works in all cases, it is tempting to increase m as much as possible. This will decrease the size of the vector spaces, hence the cost of linear algebra in the second phase of index calculus. The parameter m can however not be increased too much (say not up to $m \approx n$) since the factor $m!$ arising from the symmetry of Semaev's polynomials will then significantly decrease the probability that one random point can be decompoed as a sum of factor basis elements.

In practice, we could solve $(m+1)$th summation polynomials with m up to 6 using this splitting strategy. For smaller values of m, we could solve them for n values larger than in previous works. Although the experiments are limited by their memory requirements, they do provide some support for the generalized first fall degree assumption in that case. Detailed experimental results are provided in Sections 5.1 to 5.3.

4 New binary ECDLP Algorithms and Variants

In this section, we consider the theoretical implications of first fall degree assumptions when they are generalized even further. Although some of them are partly backed by small experimental results, we believe that the very surprising results we obtain are further evidence that first fall degree assumptions must be taken with great care.

4.1 Binary ECDLP Algorithm

We first focus on binary ECDLP. Let n be a prime number, let $K := \mathbb{F}_{2^n}$ and let E be the elliptic curve given by the equation $y^2 + xy = x^3 + a_2x^2 + a_6$ with $a_2, a_6 \in K$. The second and third summation polynomials are given by $S_2(z_1, z_2) := z_1 + z_2$ and $S_3(z_1, z_2, z_3) := z_1^2z_2^2 + z_1^2z_3^2 + z_2^2z_3^2 + z_1z_2z_3 + a_6$. Let $P \in E$ and let $Q \in < P >$. Let N be the order of P, and thus we can set $N = O(2^n)$ since the order of the elliptic curve is in the interval $[2^n + 1 - 2 * 2^{(n/2)}, 2^n + 1 + 2 * 2^{(n/2)}]$.

For any $(R_1, \ldots, R_n) \in E^n$, let x_i be the x-coordinate of $\pm R_i$ and let w_i be the x-coordinate of $Q_i := \sum_{j=1}^{i}(-1)^{s_j} R_j$ where $s_j \in \{0,1\}$. Our attack goes as follows

1. Let $R_i = (x_i, y_i) = a_i P$ for a_i randomly generated in $\{0, \ldots, N-1\}$, $i = 1, \ldots, n$.
2. Let $R = (X, Y) = aP + bQ$ for a, b randomly generated in $\{0, \ldots, N-1\}$
3. Build the system
$$\begin{cases} S_3(x_1, x_2, w_2) = 0 \\ S_3(w_2, x_3, w_3) = 0 \\ S_3(w_3, x_4, w_4) = 0 \\ \qquad \cdots \\ S_3(w_{n-1}, x_n, X) = 0. \end{cases}$$
4. Apply a Weil descent on each equation of this system, "restricting" each w_i in the whole field $\mathbb{F}_{2^n}$. Add the field equations to this system. This results in a system with $n(n-2)$ variables over $\mathbb{F}_2$ (corresponding to the variables $w_2, \ldots, w_{n-1}$ over $\mathbb{F}_{2^n}$) and $2n(n-2)$ equations, each of them of degree at most 2.
5. Solve this system with F4. If there is no solution, go back to Step 2.
6. Reconstruct the corresponding solutions w_i over $\mathbb{F}_{2^n}$.
7. After obtaining the solutions of the algebraic equation, we first recover s_n and Q_{n-1} by checking $Q_{n-1} + (-1)^{s_n} R_n + R = O$ where $s_n \in 0, 1$ and the x-coordinate of $Q_{n-1} = w_{n-1}$. Next we go to the next check with $Q_{n-2} + (-1)^{s_{n-1}} R_{n-1} + Q_{n-1} = O$ where $s_{n-1} \in 0, 1$ and the x-coordinate of $Q_{n-2} = w_{n-2}$. Such $2(n-2)$ tests are required to obtain the correct point decomposition of one R.
8. Deduce the discrete logarithm $Q = b^{-1}\left(-a + \sum_{i=1}^{n}(-1)^{s_i} a_i\right) P$

By heuristic probabilistic arguments, each random choice of a and b is expected to produce one relation on average. (This is because we expect the function $\varphi : \{0,1\}^n \to \mathbb{Z}_N : s \to \sum(-1)^{s_i} a_i \bmod N$ to be close to a bijective function, since $N = O(2^n)$.) Note that we saved the factor $m!$ ($m = n$ here) present in other attacks by constraining each x_i in a distinct "factor basis" $\{\pm P_i\}$, similarly to Galbraith and Gebregiyorgis's [23].

If the first fall degree assumption holds for the system generated in Step 4, then the degree of regularity of this system should remain very small even for large n. This implies that the runtime complexity of the attack is roughly
$$\left(n^{2\omega D_{reg}}\right).$$
Previous experiments with the third summation polynomial suggest that D_{reg} might be equal to the constant 4 [38,29,30], in which case this attack runs in *expected polynomial time.* Detailed experimental results are provided in Sections 5.5.

4.2 Reduction to binary ECSSP

A subset sum problem (SSP) is a classical hard problem related to decoding theorem. An elliptic curve subset sum problem (ECSSP) is the specific problem on the elliptic curve. In the paper of Cheng [7], it is stated that the ECSSP is NP-complete and the ECDLP is not harder than ECSSP. We are going to reduce our algorithm to solve ECSSP in this section.

First, we define the elliptic curve subset sum problem. Let E be a curve and $R_1, R_2, ..., R_n, R \in E$. The elliptic curve subset sum problem is to find out that if $R = \sum_{1 \le i \le n} e_i R_i$, where $e_i \in \{0,1\}$.

Then we modified the binary ECDLP method as followed:

1. At Step 2 in the binary ECDLP method, leet $R = (X, Y) = (\sum_{1 \le i \le n} a_i + a)P$ for a randomly generated in $\{0, \ldots, N-1\}$
2. At Step 8 in the binary ECDLP method, by modifying Step 2, finally we deduce the subset sum $aP = \sum_{1 \le i \le n} e_i(2a_i P)$.

By this reduction, we can solve the problem whether aP is a subset sum of $\{2a_iP\}$ as well as the solution in Step 2. Now we claim that the binary ECDLP is also a solution of ECSSP, which is an NP-complete problem. A similar attack has been developed independently in [22,7] If the first fall degree assumptions works, then we will have a polynomial time algorithm to solve an NP-complete problem.

4.3 Binary DLP Algorithm, First Variant

We now consider the discrete logarithm problam over the multiplicative group of a finite field of characteristic 2. Let $K := \mathbb{F}_{2^n}$, let g a generator of K^* and let $h = g^k$. We suggest the following algorithm to compute k:

1. Let a_i be randomly chosen elements in $\mathbb{F}_{2^n}^*$, $i = 1, \dots, n$.
2. Let $\mathcal{F} := \{g, h\} \cup \{e + a_i, e \in \mathbb{F}_2, i = 1, \dots, n\}$.
3. Let $\mathcal{R} := \emptyset$
4. Relation search: until $\#\mathcal{R} > 2n + 1$ do
   (a) Let $r = g^a h^b$ for a, b randomly generated in $\{0, \dots, 2^n - 2\}$
   (b) Build the system
$$\begin{cases} w_2 = (x_2 + a_2)(x_1 + a_1) \\ w_3 = (x_3 + a_3)w_2 \\ \qquad \dots \\ w_{n-1} = (x_{n-1} + a_{n-1})w_{n-2} \\ r = (x_n + a_n)w_{n-1} \end{cases}$$
   (c) Apply a Weil descent on each equation of this system, restricting x_i in $\mathbb{F}_2$ and w_i in the whole field $\mathbb{F}_{2^n}$. Add the field equations to this system. This results in a system with $n(n-1)$ variables over $\mathbb{F}_2$ (corresponding to the variables $w_2, \dots, w_{n-1}$ over $\mathbb{F}_{2^n}$ and the variables x_i over $\mathbb{F}_2$) and $2n(n-1)$ equations, each of them of degree at most 2.
   (d) Solve this system with F4. If there is no solution, go back to Step 4a.
   (e) Add the relation $g^a h^b = \prod (x_i + a_i)$ in $\mathcal{R}$.
5. Do linear algebra on the relations to recover one relation $g^a h^b = 1$, and deduce the discrete logarithm.

The algorithm uses a very small factor basis $\mathcal{F}$. Unlike in the previous algorithm, the discrete logarithms of the factor basis elements are not known a priori, so $2n + 1$ relations are needed and a (very small) linear algebra step is performed at the end.

The runtime complexity of the algorithm can be estimated as

$$n^{2\omega D_{reg}+1}$$

where D_{reg} is the degree of regularity of the above system. Under a generalization of the first fall degree assumption to this system, we expect D_{reg} to remain small even for large values of n. In fact it is easy to see that the first fall degree is equal to 3 for this system, so we expect the algorithm to run in polynomial time under an appropriate generalized first fall degree assumption.

In the next subsection, we provide another DLP algorithm which is closer in spirit to the algorithm of Section 4.3, also running in expected polynomial time under an appropriate generalization of the first fall degree assumption. Note that a similar generalization to hyperelliptic curves is straightforward given previous works such as [34]. Detailed experimental results are provided in Sections 5.6.

4.4 Binary DLP Algorithm, Second Variant

Semaev's polynomials project the addition law on point elements into an addition law constraint on their x-coordinate. The projection from one point to its x-coordinate is mostly 2-to-1 since the x-coordinates of P and $-P$ are identical.

To extend the algorithm of the previous section to finite fields, we suggest to project any field element x onto the element $z := x + x^{-1}$, which ensures that every element and its inverse have the same image. The second and third summation polynomials then naturally become

$$S_2(z_1, z_2) := z_1 + z_2, \qquad S_3(z_1, z_2, z_3) := z_1^2 + z_2^2 + z_3^2 + z_1 z_2 z_3.$$

Indeed, we will have such properties,

Lemma 1 *Properties for the DLP summation polynomials:*

i). $S_2(z_1, z_2) := z_1 + z_2 = 0$ if and only if $\exists e_1, e_2 \in \{-1, 1\}, x_1^{e_1} x_2^{e_2} = 1$,

ii). $S_3(z_1, z_2, z_3) := z_1^2 + z_2^2 + z_3^2 + z_1 z_2 z_3 = 0$ if and only if $\exists e_1, e_2, e_3 \in \{-1, 1\}, x_1^{e_1} x_2^{e_2} x_3^{e_3} = 1$,

the remaining summation polynomials are then defined inductively using resultants.

In order to fit the requirement here, we modified the binary ECDLP method in Section 4.1 as followed:

For any $(r_1, \ldots, r_n) \in K^*$, let $z_i = r_i + r_i^{-1}$ and let $w_i = q_i + q_i^{-1}$ where $q_i = \prod_{j=1}^{i} r_j^{e_j}$ where $e_j \in \{1, -1\}$.

1. Let $r_i = g^{a_i}$ for a_i randomly generated in $\{1, \ldots, |K^*|\}$, $i = 1, \ldots, n$.
2. Let $r = g^a h^b$ for a, b randomly generated in $\{1, \ldots, |K^*|\}$.
3. Build the system
$$\begin{cases} S_3(z_1, z_2, w_2) = 0 \\ S_3(w_2, z_3, w_3) = 0 \\ S_3(w_3, z_4, w_4) = 0 \\ \qquad \cdots \\ S_3(w_{n-1}, z_n, X) = 0. \end{cases}$$
4. Doing the same procedures in the binary ECDLP method in Section 4.1.
5. Reconstruct the corresponding solutions w_i over K^*.
6. Reconstruct the e_i.
7. Deduce the discrete logarithm $k = b^{-1}\left(-a + \sum_{i=1}^{n} e_i a_i\right)$.

It is also clear that it has expected polynomial complexity $O\left(n^{2\omega D_{reg}}\right)$ under an appropriate generalization of the first fall degree assumption. Detailed experimental results are provided in Sections 5.6.

4.5 Circuit Satisfiability

It is well-known that any NP-complete can be converted into a 3-SAT problem, which in turn can be modeled as a set of polynomials

$$\begin{cases} f_1(x_{i_1}, x_{j_1}, x_{k_1}) = 0 \\ \qquad \cdots \\ f_m(x_{i_m}, x_{j_m}, x_{k_m}) = 0 \end{cases}$$

where $x_{i_\ell}, x_{j_\ell}, x_{k_\ell} \in \{x_\ell : \ell = 1, \ldots, N\}$ are the binary variables. Looking at these polynomials and variables over $\mathbb{F}_{2^n}$ with the constraints $x_\ell \in \mathbb{F}_2$, we do get a (very special) instance of a polynomial system arising from a Weil descent, where all components of the Weil descent but one are identically zero for all equations. When it has a small number of solutions, this system can be solved in time roughly $N^{\omega D_{reg}}$ using Gröebner basis algorithms. The first fall degree of the above system is bounded by 4, so the problem would be solvable in polynomial time under a generalization of the first fall degree assumption.

5 Experimental Results

In order to investigate the validity of our generalized first fall degree assumptions, we first performed experiments on Weil descent systems coming from splitting strategies of Semaev's polynomials. We then also studied the Weil descent of "generic resultant polynomials". Finally, we implemented the Algorithms of Sections 4.1. All algorithms were implemented in Magma, and we used the *GroebnerBasis* function of Magma to compute Gröbner bases.

The experiments of Sections 5.1 to 5.4 were conducted on a CPU with four 16-cores AMD Opteron Processor 6276, running at 2.3GHz with a L3 cache of 16MB. The Operating System was Linux Mint 14 Nadia with kernel version GNU/Linux 3.5.0-17-generic x86_64 and 512GB memory. The programming platform was Magma V2.18-9 in its 64-bit version. We usually provide average results on at least 10 experiments, except when an experiment takes more than 1000 seconds, in which case we provide average numbers on at least 2 experiments. The time unit is the second, and the memory unit is a MB.

The experiments of Sections 5.5 to 5.6 were conducted on a CPU with four 4-cores Intel Xeon Processor 5550, running at 2.67GHz with 8MB cache. The Operating System was Linux Ubuntu 12.04.5 LTS with kernel version GNU/Linux 3.5.0-17-generic x86_64 and 24GB memory. The programming platform was Magma V2.18-5 in its 64-bit version. The time unit is the second, and the memory unit is a MB.

5.1 Splitting up Semaev 4

We first applied our splitting strategy to compute relations in Diem's algorithm when $m = 3$. We chose a random curve, a random point (X, Y) on the curve and a random vector space V of appropriate size, and we solved $S_4(x_1, x_2, x_3, X) = 0$ by applying a Weil descent to two polynomials $S_3(x_1, x_2, w)$ and $S_3(w, x_3, X)$ where we left the variable w unconstrained. Table 2 compares the experimental results obtained using this approach with previous works denoted as FPPR [21] and HPST [29].

The table suggests that the degree of regularity remains at 4 in our splitting method, independently of the value of n: this is consistent with the generalized first fall degree assumption in this case. The new method is also faster than both FPPR and HPST by an order of magnitude, and it requires less memory in most cases. Empty lines in the table occur when we could not finish the computation: we observe that our splitting strategy allows for larger parameters.

5.2 Splitting up Semaev 5

We then applied our splitting strategy to compute relations in Diem's algorithm when $m = 4$. We repeated the same steps as the previous section, and we split $s_5(x_1, x_2, x_3, x_4, X)$ into three parts $\{s_3(x_1, x_2, w_1), s_3(w_1, X, w_2), s_3(w_2, x_3, x_4)\}$, where the new variables w_1 and w_2 were left unconstrained. We remark that it is better to split the polynomial in this way than as $\{s_3(x_1, x_2, w_1), s_3(w_1, x_3, w_2), s_3(w_2, x_4, X)\}$ to obtain a better balance of variables in the three equations.

Table 3 summarizes the experimental results obtained using this approach. Note that we could not finish the computation using previous methods (neither FPPR [21] nor HPST [29]), so we only show the experiments results for the new method here. The table shows that D_{reg} remains at 4 in this case, which is still consistent with the generalized first fall degree assumption.

	n	n'	sol: yes				sol: no			
			D_{reg}	t_{trans}	t_{groe}	mem	D_{reg}	t_{trans}	t_{groe}	mem
ImpFPPR	17	4	7	15.47	6.81	58.16	7	16.53	1.2	55.37
ImpHPST	17	4	3	1.31	3.91	31.52	3	1.33	2.23	24.88
ThisWork	17	4	4	0.66	4.96	37.38	4	0.65	3.16	39.63
ImpFPPR	17	5	7	46.53	218.87	723.08	7	48.06	59.82	725.07
ImpHPST	17	5	4	2.21	485.1	596.46	4	2.16	136.93	492.88
ThisWork	17	5	4	0.84	10.32	68.55	4	0.85	9.19	61.67
ImpFPPR	19	5	7	50.5	91.61	401.17	7	54.03	41.8	348.01
ImpHPST	19	5	4	1.97	516.67	619.63	4	2.67	182.92	492.82
ThisWork	19	5	4	1.19	18.6	91.91	4	1.11	16.19	86.61
ImpFPPR	19	6								
ImpHPST	19	6								
ThisWork	19	6	4	1.46	135.17	400.89	4	1.44	68.60	380.98
ImpFPPR	23	5	7	64.67	70.46	475.55	7	65.07	55.75	381.39
ImpHPST	23	5	4	3.01	157.86	323.6	4	3.07	17.83	253.16
ThisWork	23	5	4	1.37	49.14	136.1	4	1.41	34.61	108.24
ImpFPPR	23	6	7	163.45	3888.7	6656.13	7	156.11	3309.43	5025.06
ImpHPST	23	6	4	4.36	5150.12	4791.31	4	4.42	3082.15	4428.07
This Work	23	6	4	1.77	168.12	794.3	4	1.69	146.35	767.47
ImpFPPR	23	7								
ImpHPST	23	7								
ThisWork	23	7	4	2.07	538.77	1981.37	4	2.09	476.41	1936.13
ImpFPPR	29	6	7	198.99	4511.74	6685.01	7	204.07	1681.27	6528.03
ImpHPST	29	6	4	5.67	2848.46	3368.01	4	5.76	932.65	2681.2
ThisWork	29	6	4	2.53	636.97	2195.45	4	2.44	489.95	2201.95
ImpFPPR	29	7								
ImpHPST	29	7								
ThisWork	29	7	4	3.02	1907.13	5555.25	4	2.83	1519.04	5545.99
ImpFPPR	29	8								
ImpHPST	29	8								
ThisWork	29	8	4	3.45	5450.35	12297.73	4	3.36	4868.43	11449.37
ImpFPPR	31	6	7	209.98	4664.25	7336.11	7	206.29	1205.29	7276.85
ImpHPST	31	6	4	5.82	2811.99	3257.82	4	6.09	1049.14	2616.21
ThisWork	31	6	4	2.79	1088.8	2984.91	4	2.61	827.2	2988.82

Table 2: Comparison FPPR [21], HPST [29] and our method when $m = 3$

5.3 Splitting up Semaev 6 and 7

We then considered larger summation polynomials. We followed the same steps as in the previous two sections, and we solved $s_{m+1}(x_1, x_2, ..., x_m, X) = 0$ by applying a Weil descent to split polynomials $s_3(x_1, x_2, w_1)$, $s_3(w_1, x_3, w_2)$, ..., and $s_3(w_{m-2}, x_m, X)$ where we left the variables $w_1, w_2, ..., w_{m-2}$ unconstrained. The results are shown in Table 4.

The table shows that we could solve Semaev's polynomials up to $m = 6$ (s_7) using our splitting method. The degree of regularity remains reasonably small, but unlike for smaller m values it does seem slightly larger than 4 when $m = 5$ and $m = 6$, so the first fall degree assumption must probably be relaxed in these cases. We point out that the experiments here were only repeated twice, so the average results may still be affected by variability effects such as the number of solutions.

5.4 Generic resultant polynomials

We then studied the case of "generic resultant polynomials". To this aim, we generated two random polynomials $f^{(1)}$ and $f^{(2)}$ with $m+1$ variables over the base field F_{2^n}, with degree $2^t - 1$ in each variables, such that f_1 and f_2 share exactly one variable. We applied a Weil descent on both f and the set $\{f^{(1)}, f^{(2)}\}$,

n	m	n'	sol: yes				sol: no			
			D_{reg}	t_{trans}	t_{groe}	mem	D_{reg}	t_{trans}	t_{groe}	mem
13	4	3	4	1.05	246.38	803.26	4	0.99	178.89	514.82
17	4	3	4	1.77	427.45	2538.02	4	1.47	545.94	2258.43
17	4	4	4	1.90	4252.28	10987.02	4	1.72	2460.63	10483.68
23	4	3	4	3.49	3132.48	14068.59	4	4.46	2061.40	11451.40
23	4	4	4	3.94	27501.20	57511.55	4	4.56	18092.68	57493.99
23	4	5	4	3.73	24890.21	57657.05	4	5.36	16672.26	57553.95
29	4	3	4	7.38	17053.36	57943.28	4	8.05	11804.60	44238.16
29	4	4	4	8.74	122559.83	220688.44	4	7.00	431414.36	671703.84

Table 3: Experiment results for the splitting strategy when $m = 4$

n	m	n'	sol: yes				sol: no			
			D_{reg}	t_{trans}	t_{groe}	mem	D_{reg}	t_{trans}	t_{groe}	mem
13	5 (s_6)	2	4.00	1.55	42.39	395.97	4.00	1.45	31.80	276.18
13	5 (s_6)	3	4.33	1.87	686.53	2784.53	4.00	1.74	668.06	3711.62
13	6 (s_7)	2	5.00	2.08	213.70	887.97	4.00	2.00	238.17	932.53
17	5 (s_6)	3	4.33	2.53	4378.38	12444.42	4.00	2.64	572.30	4518.55
17	6 (s_7)	2	4.50	3.15	1295.83	3618.48	4.00	4.00	305.57	2164.90
17	6 (s_7)	3	5.00	3.70	25543.69	25533.75	4.00	3.79	14361.97	15185.44
23	5 (s_6)	3	4.67	4.84	8780.20	28556.62	4.00	5.66	3602.93	23407.68
23	6 (s_7)	3	5.00	6.27	88161.21	43730.85	4.00	5.97	4666.71	16058.17
29	5 (s_6)	3	4.33	10.39	48122.94	102263.43	4.00	9.63	16588.69	81579.91

Table 4: Experiment results for the splitting strategy when $m > 4$

we solve both systems $F^{(res)}$ and $F^{(1+2)}$ using Gröbner basis algorithms, and we compare the timings and the degree of regularity obtained. The experiment results with $t = 1, 2, 3$, $m = 1, 2$ and $7 \leq n \leq 29$ are shown in Fig. 1, Fig. 2, Fig. 3 and Fig. 4.

For the largest parameters t and m we tested, solving the system was actually faster than generating it. We believe this is because we did not optimize the Weil descent process (to focus on its resolution) but we did not investigate this further. As an example in the case $(t = 2, m = 1)$, it takes more than 20,000 seconds to setup the system and seconds to solve it. We also noted that the computation sometimes ran out of memory or took too much time to solve $F^{(res)}$ system in the step Gröbner walk no matter how small t and m were, while it worked well in solving $F^{(1+2)}$. The numbers we report are average values on the cases that could be completed.

The results obtained for generic polynomials are not as spectacular as for Semaev polynomials. On the one hand, we consistently observe smaller degrees of regularity with our splitting method, on the other hand these smaller degrees of regularity do not compensate for the larger number of variables at the parameters we could test, and for most parameters the splitting method was actually slower than solving the Weil descent of the resultant polynomial. We point our that this does not contradict the first fall degree assumption, which predicts that the new method becomes better for sufficiently large n values.

Another important remark is that for $(t = 1, m = 1)$, the resultant polynomial has actually the same degrees as both $f^{(1)}$ and $f^{(2)}$, so it definitely makes sense that $F^{(res)}$ can be solved faster than $F^{(1+2)}$ in this case. In fact for larger parameters such as $(t = 3, m = 2)$ shown in Fig 4, we observed that the splitting method became more efficient.

Interestingly, the splitting strategy performs much better for s_4 than for generic resultant polynomials with $(t = 2, m = 2)$ (arguably the closest case to s_4). We believe this is because of the sparsity and the symmetric structure of the Semaev's polynomial system. The sparsity of a system is quite important in solving the system, as it affects the matrix size occurring in F_4 calculation, hence the run time.

n	$F^{(1+2)}$		$F^{(res)}$	
	D_{reg}	time	D_{reg}	time
7	3.00	0.00	3.00	0.00
8	3.60	0.01	3.00	0.00
9	3.00	0.01	3.00	0.00
10	3.20	0.02	3.00	0.00
11	3.00	0.07	3.00	0.01
12	3.60	0.03	3.00	0.00
13	3.20	0.20	3.00	0.00
14	3.00	0.21	3.00	0.00
15	3.00	0.38	3.00	0.01
16	3.20	0.44	3.00	0.01
17	3.60	0.68	3.00	0.02
18	4.00	0.85	3.00	0.01
19	3.20	1.07	3.20	0.03
20	3.20	1.31	3.00	0.02
21	3.60	1.82	3.00	0.11
22	3.20	2.15	3.00	0.03
23	3.80	3.09	3.00	0.17
24	3.60	3.08	3.00	0.16
25	4.00	3.78	3.00	0.27
26	3.20	4.79	3.00	0.24
27	3.20	33.85	3.00	0.39
28	3.50	7.14	3.25	0.34
29	4.00	58.44	3.20	0.60

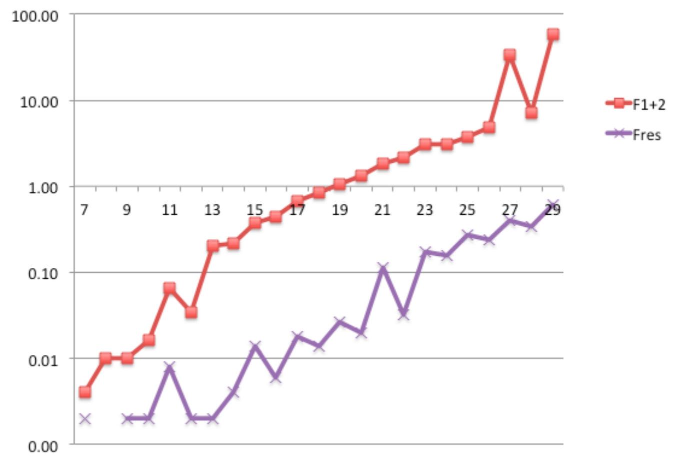

Fig. 1: Experiments for the splitting strategy with generic resultant polynomial ($t = 1, m = 1$).

5.5 New Binary ECDLP Algorithm

We implemented the binary ECDLP algorithm of Section 4.1. We observed the timings and degrees of regularity obtained and we compared them with previous methods (see [29]). The results are provided in Table 5. Perhaps surprisingly, the degree of regularity remained below 4 in all experiments we did, but we point out that the parameters we could test remained quite small. Table 5 also reports the number of times Step 2 of the algorithm was executed.

n	binary ECDLP in 4.1				FPPR	HPST
	D_{reg}	loop	time	mem	time	time
7	4	3	0.180	11.030	1.57	0.86
8	3	3	0.210	11.190		
9	4	2	0.360	13.380		
10	4	2	1.180	35.030		
11	4	1	1.120	24.160	8.63	6.70
12	4	1	4.440	48.500		
13	4	1	19.010	88.410	49.70	31.06
14	4	2	213.350	231.380		
15	4	1	597.570	364.380		
17					2454.47	1364.74

Table 5: Comparison of ECDLP algorithms

5.6 New Binary DLP Algorithms

We finally implemented the binary DLP algorithms of Sections 4.3 and 4.4, for which we report results in Table 6. As for ECDLP algorithm, the degree of regularity surprisingly remained below 4 in all experiments

n	$F^{(1+2)}$		$F^{(res)}$	
	D_{reg}	time	D_{reg}	time
7	4.20	0.02	4.80	0.00
8	4.20	0.03	4.00	0.01
9	4.20	0.06	4.00	0.00
10	4.80	0.75	5.00	0.05
11	4.40	1.00	5.00	0.04
12	4.20	1.38	5.00	0.06
13	4.20	1.60	5.00	0.10
14	4.80	30.47	5.00	1.36
15	4.60	22.38	5.00	1.62
16	4.00	30.95	5.00	2.23
17	4.20	41.86	5.00	2.64
18	5.00	2599.71	5.00	53.18
19	5.00	3419.45	5.00	47.66
20	5.00	4666.46	5.20	46.81

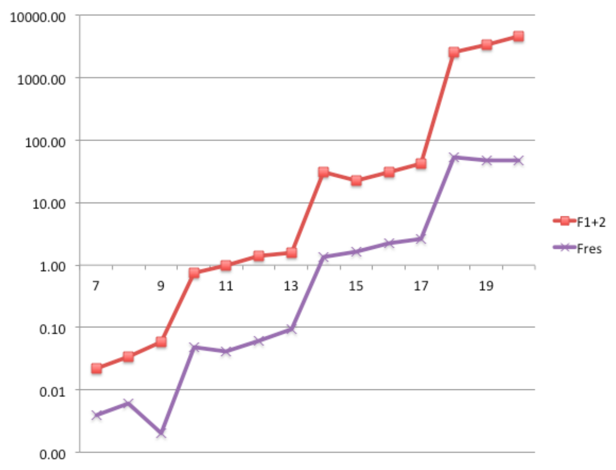

Fig. 2: Experiments for the splitting strategy with generic resultant polynomial ($t = 1, m = 2$).

n	$F^{(1+2)}$		$F^{(res)}$	
	D_{reg}	time	D_{reg}	time
7	5	0.10	7	0.01
8	5	0.13	7	0.01
9	5	0.84	7	0.09
10	5	1.34	7	0.09
11	5	4.38	7	0.39
12	5	6.91	7	0.54
13	5	221.04	7	4.87
14	5	327.19	7	5.60
15	5	1620.49	7	37.19

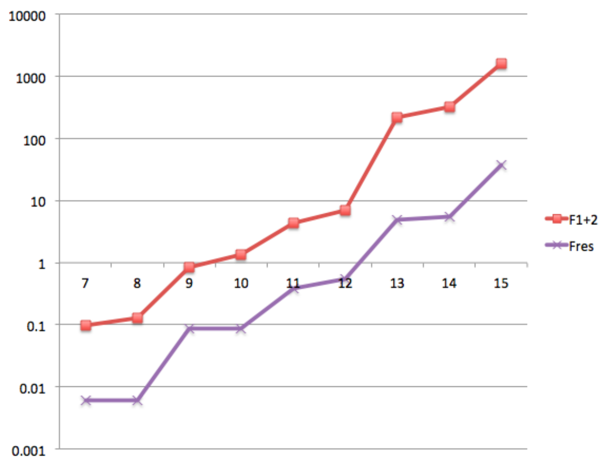

Fig. 3: Experiments for the splitting strategy with generic resultant polynomial ($t = 2, m = 1$).

we did, but again the parameters we could test were quite small, particularly with respect to the latest DLP records. For the second variant, we also report the second step of the algorithm was executed.

6 Conclusion

The first fall degree assumption approximates the degree of regularity of a polynomial system by its first fall degree. In this paper, we studied the validity of this assumption for polynomial systems arising from a Weil descent on a *set of polynomials* over $\mathbb{F}_{2^n}$. These systems generalize those studied in [21,37], where the set contained a single polynomial.

In the case of a single polynomial, Petit and Quisquater's experimental results [37] suggest that the assumption might hold and they lead to the conjecture that binary ECDLP can be solved in subexponential time. The validity of the assumption remains an open problem even in this case, except for *univariate* polynomial where it is known to hold [36].

When the set contains a small number of polynomials, our experimental results suggest that the assumption might still hold. We showed that this would allow for a more efficient resolution of resultant polynomials, in particular Semaev's summation polynomials involved in ECDLP index calculus algorithms. We obtained significant timing improvements over previous works [37,29,30] for the computation

t	m	n	$F^{(1+2)}$ D_{reg}	$F^{(1+2)}$ time	$F^{(res)}$ D_{reg}	$F^{(res)}$ time
2	2	7	7	0.119	8	0.01
2	2	13	5	10.612	9	3.406
3	1	7	5	0.115	7	0.113
3	1	13	5	219.813	7	13.534
3	2	13	7	39.359	12	625.939

Fig. 4: Experiments for the splitting strategy with generic resultant polynomial for larger t and m.

n	binary DLP 1st variant			binary DLP 2nd variant			
	D_{reg}	time	mem	D_{reg}	loop	time	mem
7	3	0.14	11.03	4	1	0.15	11.03
8	4	0.18	13.25	3	1	0.17	11.50
9	4	0.47	15.81	4	1	0.35	14.72
10	3	0.71	17.28	3	2	1.69	22.06
11	3	1.53	35.84	4	3	13.60	242.66
12	4	23.62	94.75	4	1	96.19	864.59
13	4	126.53	235.62	4	4	1150.25	3330.28
14	4	503.40	1713.34	4	1	1105.41	5308.00
15	4	2286.90	6486.28	4	1	2647.24	8130.47
16	4	5550.42	2901.28				
17	4	22589.91	12900.94				
18	3	15497.95	15763.69				

Table 6: Comparison of the DLP variants

of relations for binary ECDLP, and could complete the computation for some parameters that were previously intractable.

For an intermediate number of polynomials in the set, our experiments suggest a moderate increase in the degree of regularity at constant first fall degree. This suggests that some practical gain can be obtained in binary ECDLP algorithms, even though a precise estimation of these gains is impossible at this stage.

Finally when the number of polynomials is as large as the field extension degree, we describe a number of consequences of the first fall degree heuristic that strongly suggest that it must be false: new polynomial time algorithms for binary ECDLP and binary DLP, and a polynomial time resolution of ECSSP and 3-SAT with Gröbner basis algorithms.

Our results provide a synthetic picture of the validity of first fall degree assumptions for these systems, from cases where they are proven to hold to cases where they now seem very unlikely.

Acknowledgements We thank anonymous reviewers of CT-RSA 2015 for pointing us out [22,7].

References

1. Leonard M. Adleman. The function field sieve. In Adleman and Huang [3], pages 108–121.
2. Leonard M. Adleman, Jonathan DeMarrais, and Ming-Deh A. Huang. A subexponential algorithm for discrete logarithms over the rational subgroup of the Jacobians of large genus hyperelliptic curves over finite fields. In Adleman and Huang [3], pages 28–40.
3. Leonard M. Adleman and Ming-Deh A. Huang, editors. *Algorithmic Number Theory, First International Symposium, ANTS-I, Ithaca, NY, USA, May 6-9, 1994, Proceedings*, volume 877 of *Lecture Notes in Computer Science*. Springer, 1994.
4. Razvan Barbulescu, Pierrick Gaudry, Antoine Joux, and Emmanuel Thomé. A heuristic quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic. In *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, pages 1–16, 2014.

5. Magali Bardet, Jean-Charles Faugère, and Bruno Salvy. On the complexity of Gröbner basis computation of semi-regular overdetermined algebraic equations. In *International Conference on Polynomial System Solving - ICPSS*, pages 71 –75, Nov 2004.
6. Luk Bettale, Jean-Charles Faugère, and Ludovic Perret. Cryptanalysis of HFE, Multi-HFE and Variants for Odd and Even Characteristic. *Des. Codes Cryptography*, pages 1–42, 2012. accepted.
7. Qi Cheng. Hard problems of algebraic geometry codes. *IEEE Transactions on Information Theory*, 54(1):402–406, 2008.
8. S. Collart, M. Kalkbrener, and D. Mall. Converting bases with the Gröbner walk. *Journal of Symbolic Computation*, 24:465–469, 1997.
9. Don Coppersmith. Fast evaluation of logarithms in fields of characteristic two. *IEEE Transactions on Information Theory*, 30(4):587–593, 1984.
10. Claus Diem. On the discrete logarithm problem in elliptic curves. *Compositio Mathematica*, 147:75–104, 2011.
11. Claus Diem. On the discrete logarithm problem in elliptic curves II. 2011. `http://www.math.uni-leipzig.de/~diem/preprints/dlp-ell-curves-II.pdf`.
12. Whitfield Diffie and Martin E. Hellman. New directions in cryptography. IT-22(6):644–654, 1976.
13. Jintai Ding and Timothy J. Hodges. Inverting HFE systems is quasi-polynomial for all fields. In Phillip Rogaway, editor, *CRYPTO*, volume 6841 of *Lecture Notes in Computer Science*, pages 724–742. Springer, 2011.
14. Vivien Dubois and Nicolas Gama. The degree of regularity of HFE systems. In Masayuki Abe, editor, *ASIACRYPT*, volume 6477 of *Lecture Notes in Computer Science*, pages 557–576. Springer, 2010.
15. Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.
16. Jean-Charles Faugère. A new efficient algorithm for computing Gröbner bases (F4). *Journal of Pure and Applied Algebra*, 139(1–3):61–88, June 1999.
17. Jean-Charles Faugère. A new efficient algorithm for computing Gröbner bases without reduction to zero (F5). In *Proceedings of the 2002 international symposium on Symbolic and algebraic computation*, ISSAC '02, pages 75–83, New York, NY, USA, 2002. ACM.
18. Jean-Charles Faugère, Patrizia M. Gianni, Daniel Lazard, and Teo Mora. Efficient computation of zero-dimensional gröbner bases by change of ordering. *J. Symb. Comput.*, 16(4):329–344, 1993.
19. Jean-Charles Faugère, Louise Huot, Antoine Joux, Guénaël Renault, and Vanessa Vitse. Symmetrized summation polynomials: Using small order torsion points to speed up elliptic curve index calculus. In *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, pages 40–57, 2014.
20. Jean-Charles Faugère and Antoine Joux. Algebraic cryptanalysis of Hidden Field Equation (HFE) cryptosystems using Gröbner bases. In Dan Boneh, editor, *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 44–60. Springer, 2003.
21. Jean-Charles Faugère, Ludovic Perret, Christophe Petit, and Guénaël Renault. Improving the complexity of index calculus algorithms in elliptic curves over binary fields. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT*, volume 7237 of *Lecture Notes in Computer Science*, pages 27–44. Springer, 2012.
22. Steven Galbraith. Hard problems of algebraic geometry codes, by Qi Cheng., August 2011. `https://ellipticnews.wordpress.com/2011/08/04/hard-problems-of-algebraic-geometry-codes-by-qi-cheng/`.
23. Steven D. Galbraith and Shishay W. Gebregiyorgis. Summation polynomial algorithms for elliptic curves in characteristic two. Personal communication, 2014.
24. Pierrick Gaudry. Index calculus for abelian varieties of small dimension and the elliptic curve discrete logarithm problem. *J. Symb. Comput.*, 44(12):1690–1702, 2009.
25. Daniel M. Gordon. Discrete logarithms in *f(p)* using the number field sieve. *SIAM J. Discrete Math.*, 6(1):124–138, 1993.
26. Louis Granboulan, Antoine Joux, and Jacques Stern. Inverting HFE is quasipolynomial. In Cynthia Dwork, editor, *CRYPTO*, volume 4117 of *Lecture Notes in Computer Science*, pages 345–356. Springer, 2006.
27. Timothy Hodges, Christophe Petit, and Jacob Schlather. First fall degree and Weil descent. Submitted to Finite Fields and their Applications, 2012.
28. Yun-Ju Huang. *Efficient Algorithms to Solve the Elliptic Curve Discrete Logarithm Problem over Finite Fields of Characteristic Two*. PhD thesis, Graduate School of Mathematics, Kyushu University, Fukuoka, Japan, January 2015.
29. Yun-Ju Huang, Christophe Petit, Naoyuki Shinohara, and Tsuyoshi Takagi. Improvement of faugère et al.'s method to solve ecdlp. In Kazuo Sakiyama and Masayuki Terada, editors, *IWSEC*, volume 8231 of *Lecture Notes in Computer Science*, pages 115–132. Springer, 2013.
30. Yun-Ju Huang, Christophe Petit, Naoyuki Shinohara, and Tsuyoshi Takagi. Improvement of FPPR method to solve ECDLP. *Pacific Journal of Mathematics for Industry*, 7(1), 2015.
31. Koray Karabina. Point decomposition problem in binary elliptic curves. Cryptology ePrint Archive, Report 2015/319, 2015. `http://eprint.iacr.org/`.
32. Michiel Kosters and Sze Ling Yeo. Notes on summation polynomials. *ArXiv e-prints*, March 2015. `http://arxiv.org/pdf/1503.08001v1.pdf`.
33. Daniel Lazard. Gröbner-bases, Gaussian elimination and resolution of systems of algebraic equations. In *Proceedings of the European Computer Algebra Conference on Computer Algebra*, volume 162 of *Lecture Notes in Computer Science*, Berlin, Heidelberg, New York, 1983. Springer Verlag.

34. Koh-ichi Nagao. Decomposition formula of the Jacobian group of plane curve. Cryptology ePrint Archive, Report 2013/548, 2013. `http://eprint.iacr.org/`.
35. Koh-ichi Nagao. Equations system coming from Weil descent and subexponential attack for algebraic curve. Cryptology ePrint Archive, Report 2013/549, 2013. `http://eprint.iacr.org/`.
36. Christophe Petit. Bounding HFE with SRA. `http://perso.uclouvain.be/christophe.petit/files/SRA_GB.pdf`, 2014.
37. Christophe Petit and Jean-Jacques Quisquater. On polynomial systems arising from a Weil descent. In Xiaoyun Wang and Kazue Sako, editors, *Asiacrypt*, volume 7658 of *Lecture Notes in Computer Science*, pages 451–466. Springer, 2012.
38. Christophe Petit and Jean-Jacques Quisquater. On polynomial systems arising from a Weil descent. Preprint., 2012.
39. John M. Pollard. Monte Carlo methods for index computation (mod p). *Mathematics of Computation*, 32, 1978.
40. Oliver Schirokauer. Discrete logarithms and local units. Philos. Trans. Roy. Soc. London Ser. A, vol 345 (1676), pp 40900F1423, 1993., 1993.
41. Igor Semaev. Summation polynomials and the discrete logarithm problem on elliptic curves. Available at `http://www.isg.rhul.ac.uk/~ppai034/_pub/papers/Semaev%20%28Feb%29.pdf`, 2004.
42. Igor Semaev. New algorithm for the discrete logarithm problem on elliptic curves. Cryptology ePrint Archive, Report 2015/310, 2015. `http://eprint.iacr.org/`.
43. Daniel 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.
44. Michael Shantz and Edlyn Teske. Solving the elliptic curve discrete logarithm problem using Semaev polynomials, Weil descent and Gröbner basis methods - an experimental study. In Marc Fischlin and Stefan Katzenbeisser, editors, *Number Theory and Cryptography - Papers in Honor of Johannes Buchmann on the Occasion of His 60th Birthday*, volume 8260 of *Lecture Notes in Computer Science*, pages 94–107. Springer, 2013.