# Faster Initial Splitting for Small Characteristic Composite Extension Degree Fields

Madhurima Mukhopadhyay and Palash Sarkar

Applied Statistics Unit
Indian Statistical Institute
Kolkata, INDIA
Email:{madhurima_r,palash}@isical.ac.in

March 18, 2019

**Abstract**

Let $p$ be a small prime and $n = n_1 n_2 > 1$ be a composite integer. For the function field sieve algorithm applied to $\mathbb{F}_{p^n}$, Guillevic (2019) had proposed an algorithm for initial splitting of the target in the individual logarithm phase. This algorithm generates polynomials and tests them for $B$-smoothness for some appropriate value of $B$. The amortised cost of generating each polynomial is $O(n_2^2)$ multiplications over $\mathbb{F}_{p^{n_1}}$. In this work, we propose a new algorithm for performing the initial splitting which also generates and tests polynomials for $B$-smoothness. The advantage over Guillevic splitting is that in the new algorithm, the cost of generating a polynomial is $O(n \log(1/\pi))$ multiplications in $\mathbb{F}_p$, where $\pi$ is the relevant smoothness probability.

Keywords: Discrete Log, Finite fields, Function Field Sieve, Cryptography

MSC: 94A60, 12E20, 11N35, 11Y99

## 1 Introduction

Designing efficient algorithms for the discrete logarithm problem over finite fields is an important problem. Let $p$ be a prime, $n \geq 1$ be an integer and $\mathbb{F}_{p^n}$ be the finite field of $p^n$ elements. Let $\alpha$ be a generator of $\mathbb{F}_{p^n}^*$. Given an element $\beta \in \mathbb{F}_{p^n}^*$, the discrete logarithm of $\beta$ to base $\alpha$ is an integer $i \in [0, p^n - 2]$ such that $\beta = \alpha^i$, i.e., $i = \log_\alpha \beta$. The discrete logarithm problem in $\mathbb{F}_{p^n}$ is to find $i$, given $\alpha$ and $\beta$. When $\alpha$ is understood, one simply writes $\log \beta$ instead of $\log_\alpha \beta$.

Let $L_Q(a, c)$ be the usual sub-exponential notation:

$$L_Q(a, c) \quad = \quad \exp\left((c + o(1))(\ln Q)^a (\ln \ln Q)^{1-a}\right); \text{ where } Q = p^n, a \in [0, 1], c \neq 0. \tag{1}$$

Let $p = L_Q(a, c)$. Fields for which $a < 1/3$ are usually referred to as small characteristic fields. For small characteristic fields, the algorithm of choice is the function field sieve (FFS) algorithm [4, 5]. This is a complex algorithm consisting of three major steps, namely, relation collection, linear algebra and the individual logarithm steps. The individual logarithm step itself consists of two sub-tasks, initial splitting followed by descent. In this paper, our focus will be on the task of initial splitting for small characteristic, composite extension degree fields.

Let $\mathbb{F}_{p^n}$ be the finite field of $p^n$ elements where $p$ is a small prime and $n = n_1 n_2 > 1$ is a composite integer. For practical scenarios, $n_1 \ll n_2$. The standard algorithm for initial splitting for such fields is the Waterloo algorithm [8, 7]. The Waterloo algorithm iteratively generates a pair of polynomials of degrees at most $n_2/2$ and tests both of them for $B$-smoothness for an appropriate choice of $B$. The cost of generating the pair of polynomials is $O(n_2^2)$ multiplications over $\mathbb{F}_{p^{n_1}}$. Considering a multiplication in $\mathbb{F}_{p^{n_1}}$ to require $O(n_1^2)$ $\mathbb{F}_p$ multiplications, the cost of generating the pair of polynomials is $O(n^2)$ $\mathbb{F}_p$ multiplications.

Guillevic [19] proposed a different algorithm for initial splitting over such fields. This algorithm iteratively generates a polynomial of degree $n_2 - d/n_1$ and tests it for $B$-smoothness, where $d$ is the largest non-trivial divisor of $n$. The key insight utilised in [19] is that multiplying the target by an element of a proper subfield does not change the discrete logarithm modulo $\Phi_n(p)$, where $\Phi_n(x)$ is the $n$-th cyclotomic polynomial. This insight was earlier stated in [18]. The amortised cost of generating a polynomial in the Guillevic splitting algorithm is the same as that of the Waterloo algorithm which as mentioned above is $O(n^2)$ multiplications over $\mathbb{F}_p$. The advantage of Guillevic splitting is that only a single polynomial is required to be $B$-smooth while for the Waterloo algorithm two polynomials are required to be $B$-smooth.

In this paper, we present a new algorithm for initial splitting for FFS applied to small characteristic, composite extension degree fields. We also utilise the insight that multiplying the target by elements of a proper subfield does not change the logarithm modulo $\Phi_n(p)$. Our utilisation of this insight, however, is different from that in [19]. The main improvement that we obtain over Guillevic splitting is that the cost of generating a polynomial to be tested for $B$-smoothness is $O(n \log_p(1/\pi_2))$ operations over $\mathbb{F}_p$ where $\pi_2$ is the probability that a polynomial of degree $n_2 - s - 1$ over $\mathbb{F}_{p^{n_1}}$ is $B$-smooth. The parameter $s$ is new to our work and can be chosen such that the degrees of the polynomials generated by the new method is only slightly larger than those generated by the Guillevic splitting algorithm. Consequently, the smoothness probabilities and the times for smoothness testing of both the Guillevic splitting and the new algorithm are almost the same. Since $\log_p(1/\pi_2) \ll n$, the time for generating a polynomial by the new method is significantly lower than the time for generating a polynomial using the Guillevic splitting algorithm.

### Previous Works

Coppersmith [9] described an $L(1/3, \cdot)$ algorithm to solve the discrete logarithm problem over small characteristic fields. This was followed by the function field sieve algorithm [4, 5]. For small characteristic fields, there has been a substantial progress in the efficiency of the FFS algorithm [13, 12, 10, 21, 6, 20, 14, 3, 17, 23, 16, 3, 17]. A quasi-polynomial algorithm was described in [6]. In characteristic 2, Granger, Kleinjung and Zumbrägel [15] have performed a record computation of discrete log in the field $\mathbb{F}_{2^{9234}}$. In characteristic 3, record discrete log computation as well as concrete analysis for certain fields have been described in [1, 3, 2, 22].

## 2 Preliminaries

Let $p$ be a prime and $n > 1$ be a composite integer. Let $\mathbb{F}_{p^n}$ be the finite field of $p^n$ elements. Write $n = n_1 n_2$. Let $h(y)$ be an irreducible polynomial of degree $n_1$ over $\mathbb{F}_p$. The finite field $\mathbb{F}_{p^{n_1}}$ is represented as $\mathbb{F}_p[y]/\langle h(y)\rangle$. Let $I(x)$ be an irreducible polynomial of degree $n_2$ over $\mathbb{F}_{p^{n_1}}$. The field $\mathbb{F}_{p^n}$ is represented as $\mathbb{F}_{p^{n_1}}[x]/\langle I(x)\rangle$. We will denote this as the $(I(x), h(y))$-representation of the field $\mathbb{F}_{p^n}$. We will further assume that a generator $\alpha$ of $\mathbb{F}_{p^n}^*$ is available.

Using the $(I(x), h(y))$-representation, any element $T \in \mathbb{F}_{p^n}$ can be written as follows.

$$T(x) \quad = \quad t_0(y) + t_1(y)x + \cdots + t_{n_2-1}(y)x^{n_2-1}$$

where $t_0(y), t_1(y), \ldots, t_{n_2-1}(y)$ are polynomials of degree at most $n_1 - 1$ over $\mathbb{F}_p$.

Let $\Phi_n(x)$ be the $n$-th cyclotomic polynomial and $\ell$ be a non-trivial prime divisor of $\Phi_n(p)$. Given a target $T_0 \in \mathbb{F}_{p^n}^*$, the goal is to find its discrete logarithm modulo $\ell$.

The function field sieve algorithm has three broad computational steps. In the first step, a suitable factor basis is identified and linear relations among the discrete logarithms of the factor basis elements are obtained. The second step applies sparse linear algebra computation to obtain the discrete logarithms of the factor basis elements. The logarithm of the target element $T_0(x)$ is obtained in the third step which is called the individual logarithm step.

The goal of the individual logarithm step is to express the logarithm of $T_0$ as an $\mathbb{F}_\ell$-linear combination of logarithms of elements which are either in the factor basis or whose logarithms are known. For FFS over small characteristic and composite extension degree fields, the individual logarithm step is carried out in two parts. An *initial splitting* step followed by descent to factor basis elements.

Let $B$ be a positive integer. A polynomial is said to be $B$-smooth, if all its irreducible factors have degrees less than or equal to $B$. The initial splitting step expresses the logarithm of $T_0$ in terms of logarithms of one or more polynomials each of which is $B$-smooth for a suitable value of $B$. The descent step attempts to descend the irreducible factors of the $B$-smooth polynomial(s) to the factor basis. In this paper, we will only be concerned with the initial splitting step.

A simple and important result proved by Guillevic [18] and extensively used in [19] is the following.

**Lemma 1.** *Let $p$ be a prime and $n > 1$ be an integer. Let $T \in \mathbb{F}_{p^n}$ and $u$ be an element in a proper subfield of $\mathbb{F}_{p^n}$. Then $\log T \equiv \log uT \bmod \Phi_n(p)$ and hence $\log T \equiv \log uT \bmod \ell$ for any divisor $\ell$ of $\Phi_n(p)$.*

The importance of Lemma 1 stems from the fact that one may try to multiply the target element $T_0$ with a proper subfield element $u$ to obtain $W = uT_0$ such that the degree of $W$ is substantially less than that of $T_0$. Then for any integer $B$, the chance of $W$ being $B$-smooth is significantly higher than the chance of $T_0$ being $B$-smooth.

Let $g$ be the generator of the order $\ell$ subgroup of $\mathbb{F}_{p^n}$ which is of interest. Given the target $T_0$ and an integer $t \in \{0, \ldots, \ell - 1\}$, let $T_t = g^t T_0$ so that $\log T_t = t + \log T_0 \bmod \ell$. Since $t$ is known, it is sufficient to find the logarithm of $T_t$. A well known method for initial splitting is the Waterloo algorithm [8, 7]. This algorithm expresses $T_t$ as $T_t(x) = N(x)/D(x) \bmod I(x)$ using the extended Euclidean algorithm such that the degrees of $N(x)$ and $D(x)$ are much smaller than that of $T_t(x)$. The procedure is repeated for random choices of $t$ until both $N(x)$ and $D(x)$ are obtained to be $B$-smooth for a pre-defined choice of $B$. Once this is obtained, initial splitting is said to have been achieved.

Improvement to the Waterloo algorithm based on Lemma 1 has been described by Guillevic [19]. We call this the Guillevic splitting (GS) algorithm and the complete description is shown in Algorithm-1.

**Algorithm 1:** Guillevic splitting for small characteristic composite order fields.

```
   Input: An $(I(x), h(y))$-representation of $\mathbb{F}_{p^n}$; generator $g$ of the order $\ell$ (where $\ell | \Phi_n(p)$)
          subgroup over which logarithms are to be computed; a target element $T_0(x)$; and a
          smoothness bound $B$
   Output: A $B$-smooth polynomial $P(x)$ such that $\log P(x) \equiv \log T_0(x) \bmod \ell$
 1 Let $d$ be the largest non-trivial divisor of $n$
 2 Set $\mathfrak{d} = \gcd(d, n_1)$ and $d' = d/\mathfrak{d}$
 3 Obtain $U(x) \in \mathbb{F}_{p^n}$ such that $\{1, U, \ldots, U^{d'-1}\}$ is a basis for $\mathbb{F}_{p^{d'}}$
 4 repeat
 5     Choose $t$ randomly from $\{1, 2, \ldots, \ell - 1\}$
 6     $T \leftarrow g^t T_0$
 7     Define $L = \begin{bmatrix} T \\ UT \\ \vdots \\ U^{d'-1}T \end{bmatrix}$ a $d' \times n_2$ matrix over $\mathbb{F}_{p^{n_1}}$
 8     $M \leftarrow \text{RowEchelonForm}(L)$ (with $\mathbb{F}_{p^{\mathfrak{d}}}$-linear combinations)
 9     Set $P(x)$ as the polynomial obtained from the first row of $M$
10 until $P$ is $B$-smooth;
11 return $(t, P(x))$.
```

Guillevic [19] proved the following properties of Algorithm 1.

1. $P(x)$ is a polynomial of degree at most $n_2 - d/n_1$ over $\mathbb{F}_{p^{n_1}}$. So, the degree of $P(x)$ is substantially less than that of $T_0(x)$ which has degree $n_2 - 1$.

2. $P(x) = uT(x) = ug^tT_0(x)$ for some $u \in \mathbb{F}_{p^{d'}}$ and so $\log P(x) = t + \log T_0(x) \bmod \ell$ (using Lemma 1).

It was shown in [19] that the efficiency of Algorithm 1 can be further improved using the following two ideas.
*Improvement-1:* In Algorithm 1, $M$ is obtained as the row echelon form of $L$. This requires running a Gaussian elimination on $L$. Another round of Gaussian elimination (again with $\mathbb{F}_{p^{\mathfrak{d}}}$-linear combinations) is run on $M$ from the reverse side to obtain a matrix $M'$ of the following form.

$$\begin{bmatrix} * & \ldots & * & * & 0 & \ldots & 0 \\ 0 & \ddots & & & & \ddots & \vdots \\ \vdots & \ddots & \ddots & & & \ddots & 0 \\ 0 & \ldots & 0 & * & \ldots & * & * \end{bmatrix} \tag{2}$$

The $i$-th row of $M'$ is of the form $x^{e_i}P_i(x)$, with degree of $P_i(x) \le n_2 - d/n_1$ and $e_i \approx (i-1)\mathfrak{d}/n_1$. The element $x$ is a member of the factor basis, and so it is sufficient to obtain the logarithm of $P_i$. Incorporating this into Algorithm 1 requires two rounds of $\mathbb{F}_{p^{\mathfrak{d}}}$-linear Gaussian elimination. The advantage is that after the two rounds, it provides a set of $d'$ polynomials which are to be tested for $B$-smoothness. This is to be contrasted with the basic description of Algorithm 1 where one round of $\mathbb{F}_{p^{\mathfrak{d}}}$-linear Gaussian elimination results in only one polynomial to be tested for $B$-smoothness.

*Improvement-2:* In each iteration, it is possible to further increase the number of polynomials to be tested for smoothness by taking $\mathbb{F}_{p^{\mathfrak{d}}}$-linear combinations of a small number of consecutive rows. This will increase the degrees of the resulting polynomials by one or two which does not significantly affect the probability of the polynomials being $B$-smooth.

**Cost of Algorithm 1:** A one-time computation is required by Algorithm 1 to obtain $U(x)$. Given $\alpha$, $U(x)$ is computed as $U(x) = \alpha^{(p^n-1)/(p^{d'}-1)}$. So, obtaining $U(x)$ requires an exponentiation which in turn requires $O(n \log p)$ multiplications over $\mathbb{F}_{p^n}$. Each multiplication in $\mathbb{F}_{p^n}$ requires $O(n^2)$ operations over $\mathbb{F}_p$. Using Karatsuba this cost would be $O(n^{1.59})$ operations over $\mathbb{F}_{p^n}$ and using the Fast Fourier Transform will provide even lower asymptotic costs. We take the cost of a multiplication in $\mathbb{F}_{p^n}$ to be $O(n^2)$ so that the cost of obtaining $U(x)$ is $O(n^3 \log p)$. In practice, the actual time for obtaining $U(x)$ is negligible in comparison to the cost of generating and testing polynomials for smoothness. So, the best asymptotic cost for computing $U(x)$ is not very relevant in practice. Note that it is not required to compute the basis $\{1, U, \ldots, U^{d'-1}\}$.

Apart from smoothness testing, the cost per iteration of Guillevic splitting algorithm consists of the following.

1. An exponentiation over $\mathbb{F}_{p^n}$ to compute $g^t$.
2. A total of $d'$ multiplications over $\mathbb{F}_{p^n}$ to compute $g^t T_0$ and the products $UT, \ldots, U^{d'-1}T$.
3. Two rounds of $\mathbb{F}_{p^{\mathfrak{d}}}$-linear Gaussian eliminations (considering Algorithm 1 along with Improvement-1).

Let $\pi_1$ be the probability that a polynomial of degree $n_2 - d/n_1$ over $\mathbb{F}_{p^{n_1}}$ is $B$-smooth. The generated polynomials are not statistically independent. Heuristically however, trying out about $1/\pi_1$ polynomials of degrees $n_2 - d/n_1$, it is likely to obtain one that is $B$-smooth. It has been shown in [19] that the amortised cost of obtaining one polynomial to be tested for smoothness by Algorithm 1 plus Improvement-1 is $O(n_2^2)$ multiplications over $\mathbb{F}_{p^{n_1}}$ which is the same as that of the Waterloo algorithm.

A single multiplication over $\mathbb{F}_{p^{n_1}}$ consists of a polynomial multiplication followed by a reduction. Asymptotically, the cost of the reduction step is negligible in comparison to the polynomial multiplication step though in practice, reduction consumes a significant fraction of the time for the entire field multiplication. Using the schoolbook method to perform polynomial multiplication requires $(n_1^2)$ multiplications over $\mathbb{F}_p$ and so the $O(n_2^2)$ multiplications over $\mathbb{F}_{p^{n_1}}$ has a cost of $O(n^2)$ multiplications over $\mathbb{F}_p$. Using Karatsuba's algorithm or an asymptotically faster algorithm will yield lower asymptotic costs. However, for small values of $n_1$, as is typically the case, the schoolbook method will be faster and it may be assumed that for Algorithm 1, the cost of generating a polynomial to be tested for smoothness is $O(n^2)$ multiplications over $\mathbb{F}_p$.

The total cost of Algorithm 1 is the one-time cost plus the cost of generating and testing about $1/\pi_1$ polynomials for smoothness. This cost is $O(n^3 \log p + (n^2 + \mathfrak{t}_1)/\pi_1)$ operations over $\mathbb{F}_p$, where $O(\mathfrak{t}_1)$ is the number of $\mathbb{F}_p$ operations required to test a polynomial of degree $n_2 - d/n_1$ over $\mathbb{F}_{p^{n_1}}$ for $B$-smoothness.

## 3 A New Algorithm for Initial Splitting

The setting is as in Section 2. Given a prime $p$ and a composite integer $n$, the finite field $\mathbb{F}_{p^n}$ is represented by $(I(x), h(y))$ where $h(y)$ is an irreducible polynomial of degree $n_1$ over $\mathbb{F}_p$ and $I(x)$ is an irreducible polynomial of degree $n_2$ over $\mathbb{F}_{p^{n_1}} = \mathbb{F}_p[y]/\langle h(y)\rangle$ so that $\mathbb{F}_{p^n} = \mathbb{F}_{p^{n_1}}[x]/\langle I(x)\rangle$. Also, a generator $\alpha$ of $\mathbb{F}_{p^n}^*$ is available. Given a target element $T_0(x) \in \mathbb{F}_{p^n}$, the goal is to compute the logarithm of $T_0$ modulo $\ell$ where $\ell$ is a prime divisor of $\Phi_n(p)$.

Let $d$ be the largest non-trivial divisor of $n$ and $U(x) \in \mathbb{F}_{p^n}$ be such that $\{1, U(x), \ldots, U^{d-1}(x)\}$ is a polynomial basis for $\mathbb{F}_{p^d}$. Note the difference to Guillevic splitting, where $\{1, U(x), \ldots, U^{d'-1}(x)\}$ is a polynomial basis for $\mathbb{F}_{p^{d'}}$ for $d' = d/\mathfrak{d}$ and $\mathfrak{d} = \gcd(d, n_1)$. In our method, we will not require either $\mathfrak{d}$ or $d'$.

We also make use of Lemma 1. Let

$$\mathbf{a} = (a_0, \ldots, a_{d-1})^T \in \mathbb{F}_p^d. \tag{3}$$

Then

$$V = a_0 + a_1 U + \cdots + a_{d-1} U^{d-1} \tag{4}$$

is an element of $\mathbb{F}_{p^d}$. Define

$$W = V T_0. \tag{5}$$

By Lemma 1, the logarithms of $W$ and $T_0$ are equal modulo $\ell$, i.e.,

$$\log W \equiv \log T_0 \bmod \ell. \tag{6}$$

We introduce a new parameter $s$ which is a positive integer less than $n_2$. The first step is to obtain $\mathbf{a}$ such that $W$ is a monic polynomial of degree $n_2 - s - 1$ over $\mathbb{F}_{p^{n_1}}$. Next, the polynomial $W$ is tested for $B$-smoothness.

A single polynomial may not be smooth. Let $\pi_2$ be the probability that a monic polynomial of degree $n_2 - s - 1$ over $\mathbb{F}_{p^{n_1}}$ is $B$-smooth. By generating and testing about $1/\pi_2$ random polynomials $W$ it is likely to obtain a polynomial which is $B$-smooth. We use linear algebra to generate $1/\pi_2$ polynomials over $\mathbb{F}_{p^{n_1}}$ each of degree $n_2 - s - 1$.

Given $T_0$, define

$$U_i = U^i T_0, \text{ for } i = 0, \ldots, d-1. \tag{7}$$

Each $U_i$ is a polynomial of degree at most $n_2 - 1$ over $\mathbb{F}_{p^{n_1}}$. The polynomial $U_i$ can be represented by a (column) vector $\mathbf{u}_i$ in $\mathbb{F}_p^n$.

We define an $n \times d$ matrix $M$ with entries from $\mathbb{F}_p$ as follows.

$$M = [\mathbf{u}_0 \ \mathbf{u}_1 \ \cdots \ \mathbf{u}_{d-1}] = \begin{bmatrix} M_0 \\ M_1 \end{bmatrix} \tag{8}$$

where $M_0$ is an $(n_1(n_2 - s - 1)) \times d$ matrix and $M_1$ is an $(n_1(s+1)) \times d$ matrix. Note that, given $T_0$ and $U$, the matrix $M$ is fixed and needs to be computed only once.

The polynomial $W(x)$ can be represented as a vector $\mathbf{w} \in \mathbb{F}_p^n$. We write $\mathbf{w}$ as

$$\mathbf{w} = \begin{bmatrix} \mathbf{w}_0 \\ \mathbf{w}_1 \end{bmatrix}$$

where $\mathbf{w}_0$ is in $\mathbb{F}_p^{n_1(n_2-s-1)}$ and $\mathbf{w}_1$ is in $\mathbb{F}_p^{n_1(s+1)}$.

The relation

$$W = VT_0 = (a_0 + a_1U + \cdots + a_{d-1}U^{d-1})T_0 = a_0U_0 + a_1U_1 + \cdots + a_{d-1}U_{d-1}$$

can be written in matrix notation as follows.

$$\begin{bmatrix} \mathbf{w}_0 \\ \mathbf{w}_1 \end{bmatrix} = M\mathbf{a} = \begin{bmatrix} M_0 \\ M_1 \end{bmatrix} \mathbf{a} \tag{9}$$

From (9), we obtain the following two equations.

$$\mathbf{w}_0 = M_0\mathbf{a}; \tag{10}$$
$$\mathbf{w}_1 = M_1\mathbf{a}. \tag{11}$$

Recall that our goal is to obtain $W(x)$ as a monic polynomial of degree $n_2 - s - 1$ over $\mathbb{F}_{p^{n_1}}$. This puts constraints on the coefficients of $W(x)$, namely, the coefficient of $x^{n_2-s-1}$ has to be one and the coefficients of $x^i$ for $i = n_2 - s, \ldots, n_2 - 1$ have to be zeros. These conditions define the vector $\mathbf{w}_1$ to be the following:

$$\mathbf{w}_1 = [\underbrace{0, 0, \ldots, 0, 1}_{n_1}, \underbrace{0, 0, \ldots, 0}_{n_1 s}]^T. \tag{12}$$

Given $\mathbf{w}_1$ and the matrix $M_1$, the inhomogeneous system of equation given by (11) is to be solved for $\mathbf{a}$ and the resulting solution is to be substituted in (10) to obtain $\mathbf{w}_0$. This $\mathbf{w}_0$ provides the polynomial

$$W(x) = x^{n_2-s-1} + W_0(x) \tag{13}$$

where $W_0(x) \in \mathbb{F}_{p^{n_1}}[x]$ is the polynomial represented by $\mathbf{w}_0$ and is of degree less than $n_2 - s - 1$. The polynomial $W(x)$ is to be tested for smoothness.

$M_1$ is an $n_1(s+1) \times d$ matrix and so a necessary condition for a solution to (11) to exist is $d \geq n_1(s+1)$. Define

$$r = d - n_1(s+1). \tag{14}$$

Then a general solution $\mathbf{a}$ to (11) can be written as

$$\mathbf{a} = B\mathbf{b} + \mathbf{c} \tag{15}$$

where $B$ is an $d \times r$ matrix over $\mathbb{F}_p$ which is a basis for the null space of $M_1$, $\mathbf{b}$ is in $\mathbb{F}_p^r$ and $\mathbf{c} \in \mathbb{F}_p^d$ is a particular solution to (11).

Substituting the general solution given by (15) into (10) we obtain

$$\mathbf{w}_0 = M_0\mathbf{a} = M_0 (B\mathbf{b} + \mathbf{c}) = L\mathbf{b} + \mathbf{d} \tag{16}$$

where

$$L = M_0B \quad \text{and} \quad \mathbf{d} = M_0\mathbf{c}. \tag{17}$$

$L$ is a matrix of order $n_1(n_2 - s - 1) \times r$ and $\mathbf{d} \in \mathbb{F}_p^{n_1(n_2-s-1)}$. Once the system (11) is solved, $B$ and $\mathbf{c}$ are obtained and from these it is possible to obtain $L$ and $\mathbf{d}$.

Suppose $r$ is chosen such that

$$r = d - n_1(s+1) \quad \geq \quad \lceil \log_p 1/\pi_2 \rceil. \tag{18}$$

Then varying $\mathbf{b}$ over all possible vectors in $\mathbb{F}_p^r$, it is possible to generate more than $1/\pi_2$ distinct polynomials $W(x) = x^{n_2-s-1} + W_0(x)$ of degrees $n_2 - s - 1$. These polynomials are not statistically independent and so theoretically the smoothness estimate does not apply to these polynomials. Heuristically however, it is likely that one of these polynomials is $B$-smooth. Our experiments confirm this heuristic assumption.

A necessary condition for the method to work is given by (18). Using Theorem 1 of [25] and using the estimate $\rho(u) \approx u^{-u}$ of the Dickman function, an estimate of the probability that a polynomial of degree $n_2 - s - 1$ is $B$-smooth is as follows.

$$\pi_2 \quad \approx \quad \left(\frac{n_2 - s - 1}{B}\right)^{-\left(\frac{n_2-s-1}{B}\right)} \tag{19}$$

Combining with (18), we obtain the condition

$$d - n_1(s+1) \quad \geq \quad \left\lceil \left(\frac{n_2 - s - 1}{B}\right) \log_p \left(\frac{n_2 - s - 1}{B}\right) \right\rceil. \tag{20}$$

Increasing the value of $s$ decreases the degree of $W(x)$ to be tested for smoothness and hence increases the smoothness probability $\pi_2$. On the other hand, increasing the value of $s$ reduces the left hand side of (20) and (20) may fail to hold. So, the goal is to choose the maximum possible value of $s$ such that (20) holds.

An algorithmic description of the above theory is given in Algorithm 2.

**Algorithm 2:** The new algorithm for initial splitting.

```
Input: An $(I(x), h(y))$-representation of $\mathbb{F}_{p^n}$; a target element $T_0(x)$; a smoothness bound $B$;
       and an $s$ satisfying (20);
Output: A $B$-smooth polynomial $W(x)$ such that $\log W(x) \equiv \log T_0(x) \bmod \ell$ where $\ell$ is a
        divisor of $\Phi_n(p)$
1  Choose the maximum value of $s$ such that (20) holds
2  Let $r = d - n_1(s+1)$
3  Let $U(x)$ be such that $\{1, U(x), \ldots, U^{d-1}(x)\}$ is a polynomial basis for $\mathbb{F}_{p^d}$
4  Compute the matrices $M_0$ and $M_1$ as given in (8)
5  Set $\mathbf{w}_1$ as given in (12)
6  Solve (11) to obtain $B$ and $\mathbf{c}$ as given in (15)
7  Compute $L = M_0 B$ and $\mathbf{d} = M_0 \mathbf{c}$
8  for each $\mathbf{b} \in \mathbb{F}_p^r$ do
9      Compute $\mathbf{w}_0 = L\mathbf{b} + \mathbf{d}$ and let $W_0(x) \in \mathbb{F}_{p^{n_1}}[x]$ be the polynomial represented by $\mathbf{w}_0$
10     Set $W(x) = x^{n_2-s-1} + W_0(x)$
11     if $W(x)$ is $B$-smooth then
12         break
13 return $W(x)$.
```

**Remarks:**

1. A difference between Algorithm 1 and Algorithm 2 pertains to the target. In Algorithm 2, the original target $T_0$ remains unchanged, whereas in Algorithm 1, for every $d'$ candidate polynomials a fresh target $T_t = g^t T_0$ is computed.

2. In her paper [19], Guillevic had also proposed an initial splitting algorithm for the number field sieve (NFS) algorithm in the large characteristic case. Our method of generating polynomials does not help in improving the efficiency of Guillevic splitting for the NFS algorithm.

## 3.1 Implementation Issues

Since $p$ is small, it is possible to speed up the computation of $L\mathbf{b} + \mathbf{c}$ at the cost of extra storage. For $k = 1, \ldots, p-1$, let $L^{(k)} = kL$ and denote by $L^{(k)}_{*,j}$ the $j$-th column of $L^{(k)}$. Suppose the matrices $L^{(1)}, \ldots, L^{(p-1)}$ have been computed and stored. Suppose $\mathbf{b} = (b_1, \ldots, b_r)$. Then $L\mathbf{b}+\mathbf{c}$ can be computed as follows.

```
sum ← $\mathbf{c}$;
for $j = 1, \ldots, r$ do
    if $b_j > 0$, then sum ← sum + $L^{(b_j)}_{*,j}$
end for;
return sum.
```

This method of computation avoids all the multiplications over $\mathbb{F}_p$ during the generation of the polynomial $W(x)$.

**Parallelism:** In Algorithm 2, after the matrix $L$ and the vector $c$ has been generated, the generation of the polynomials and testing them for smoothness can be completely parallelised. It is possible to allocate non-intersecting subsets of $\mathbb{F}_p^r$ to different processes. Each process independently uses the vectors in its alloted subset to generate polynomials and test them for smoothness. There is no need for any coordination between the processes.

Guillevic splitting in Algorithm 1 also supports parallelism though of a somewhat restricted kind. The completion of the doubly reduced row echelon form provides a total of $d'$ polynomials to be tested for smoothness by independent processes. The testing of these polynomials can be done in parallel. However, once these polynomials have been checked, the processes have to halt until the next batch of $d'$ polynomials have been generated. Alternatively, there can be a process which successively generates batches of $d'$ polynomials and feeds them to other processes to be tested for smoothness. Neither of these options is as simple as the parallelism that can be obtained from Algorithm 2.

## 3.2 Degrees of Polynomials Generated by Algorithm 2

Each iteration generates a polynomial $W(x)$ of degree $n_2 - s - 1$. As mentioned above, $s$ is a positive integer less than $n_2$ which is to be chosen as the maximum value satisfying (20). We determine the value of $s$ and hence the degrees of $W(x)$ for the two examples considered in [19].

*Example-1:* For this example, $p = 3$, $n_1 = 6$, $n_2 = 509$ and so $d = 3 \cdot 509$. For $26 \le B \le 32$, the maximum value of $s$ satisfying (20) is $s = 250$. So, the degrees of the corresponding $W(x)$'s are 258.

*Example-2:* For this example, $p = 3$, $n_1 = 5$ and $n_2 = 479$ and so $d = n_2 = 479$. For $42 \leq B \leq 50$, the maximum value of $s$ satisfying (20) is $s = 91$. So, the degrees of the corresponding $W(x)$'s are 387.

Using Algorithm 1, the degrees of the generated polynomials for Example-1 and Example-2 are 254 and 383 respectively. So, compared to Algorithm 1, the polynomials generated by Algorithm 2 have slightly larger degrees. This has two consequences.

1. The smoothness probability $\pi_2$ for Algorithm 2 is slightly smaller than the smoothness probability $\pi_1$ for Algorithm 1. For Example-1, $\pi_1/\pi_2$ is in the range $[1.33, 1.46]$ for $32 \geq B \geq 26$ while for Example-2, $\pi_1/\pi_2$ is in the range $[1.20, 1.26]$ for $50 \geq B \geq 42$.

2. The cost of smoothness checking $\mathfrak{t}_2$ in Algorithm 2 is slightly greater than the cost of smoothness checking $\mathfrak{t}_1$ in Algorithm 1. For the degrees considered, it is reasonable to assume $\mathfrak{t}_1 \approx \mathfrak{t}_2$.

We note that Algorithm 1 combined with both Improvement-1 and Improvement-2 result in polynomials whose degrees are one or two more than those obtained from Algorithm 1 combined only with Improvement-1. So, the degrees of polynomials generated by Algorithm 1 plus Improvement-1 and Improvement-2 are even closer to the degrees of polynomials generated by Algorithm 2.

## 3.3 Cost of Algorithm 2

The one-time cost of Algorithm 2 consists of the following components.

1. Computation of $U(x) = \alpha^{(p^n-1)/(p^d-1)}$. As in the case of Algorithm 1, this cost is $O(n^3 \log p)$. Also, as in the case of Algorithm 1, it is not required to compute the basis $\{1, U(x), \ldots, U^{d-1}(x)\}$.

2. Computation of $M_0$ and $M_1$ requires the elements $\{T_0, UT_0, \ldots, U^{d-1}T_0\}$. This requires a total of $d - 1$ multiplications in $\mathbb{F}_{p^n}$ which we estimate as $O((d-1)n^2)$ operations over $\mathbb{F}_p$.

3. Solving (11) to obtain $B$ and $\mathbf{c}$. Since $M_1$ is an $n_1(s+1) \times d$ matrix over $\mathbb{F}_p$, the cost for this step is $O(n_1(s+1)d^2)$ operations over $\mathbb{F}_p$.

4. Computing $L = M_0B$ and $\mathbf{d} = M_0\mathbf{c}$ where $M_0$ is an $n_1(n_2 - s - 1) \times d$ matrix over $\mathbb{F}_p$, $B$ is a $d \times r$ matrix over $\mathbb{F}_p$ and $\mathbf{c}$ is in $\mathbb{F}_p^d$. The cost for the matrix multiplication $M_0B$ is $O(n_1(n_2 - s - 1)dr)$ operations over $\mathbb{F}_p$ and the cost for the matrix-vector multiplication $M_0\mathbf{c}$ is $O(n_1(n_2 - s - 1)d)$ operations over $\mathbb{F}_p$.

The total one-time cost is $O(n^3 \log p + (d-1)n^2 + n_1(s+1)d^2 + n_1(n_2 - s - 1)dr) = O(n^3)$ operations over $\mathbb{F}_p$. In practice, the one-time computation is negligible in comparison to the time for generating and testing polynomials for smoothness.

The cost of generating a polynomial to be tested for smoothness is the cost of computing the matrix-vector multiplication $L\mathbf{b}$. Since $L$ is an $n_1(n_2 - s - 1) \times r$ matrix over $\mathbb{F}_p$ and $\mathbf{b} \in \mathbb{F}_p^r$, this cost is $O(n_1(n_2 - s - 1)r)$ operations over $\mathbb{F}_p$. Noting that $n_1n_2 = n$ and $r \approx \log_p(1/\pi_2)$, the cost of generating each polynomial is $O((n - n_1(s+1))r) = O((n - n_1(s+1))\log_p(1/\pi_2))$ $\mathbb{F}_p$ operations. The value of $s$ is less than $n_2$, and so, the cost of generating a polynomial is $O(n \log_p(1/\pi_2))$ $\mathbb{F}_p$-operations.

The total cost of Algorithm 2 is the cost of one-time computation plus the cost for generating and testing the polynomials for $B$-smoothness. This cost is $O(n^3 \log p + (\mathfrak{t}_2 - n \log \pi_2)/\pi_2)$ operations over

$\mathbb{F}_p$, where $O(\mathfrak{t}_2)$ is the number of $\mathbb{F}_p$ operations required to test a polynomial of degree $n_2 - s - 1$ over $\mathbb{F}_{p^{n_1}}$ for $B$-smoothness.

In contrast, the total cost of Algorithm 1 is $O(n^3 \log p + (\mathfrak{t}_1 + n^2)/\pi_2)$ operations over $\mathbb{F}_p$. Based on the discussion in Section 3.2, we may take $\pi_1$ and $\pi_2$ to be approximately equal and denote by $\pi$ this common smoothness probability. Similarly, we may take $\mathfrak{t}_1$ and $\mathfrak{t}_2$ to be approximately equal and denote by $\mathfrak{t}$ to be the time for smoothness checking in both the algorithms. Then the total time for Algorithms 1 and 2 are respectively $O(n^3 \log p + (\mathfrak{t} + n^2)/\pi)$ and $O(n^3 \log p + (\mathfrak{t} - n \log \pi)/\pi)$ operations over $\mathbb{F}_p$. The main cost for Algorithm 1 is $O((\mathfrak{t} + n^2)/\pi)$ while for Algorithm 2 it is $O((\mathfrak{t} - n \log \pi)/\pi)$.

From [11], for a degree $\delta$ polynomial over $\mathbb{F}_{p^{n_1}}$, the costs of square-free factorisation, distinct degree factorisation and equal degree factorisation are respectively $O(\delta^2)$, $O(\delta^3 \log p^{n_1})$ and $O(\delta^2 \log p^{n_1})$. So, the cost $\mathfrak{t}$ of smoothness checking is substantial. In comparison to Algorithm 1, the main advantage of Algorithm 2 is that it makes the cost of generating a polynomial negligible in comparison to the cost of testing the polynomial for smoothness.

## 4 Computational Results

To demonstrate that Algorithm 2 works, we made a basic Magma implementation for Example-1 mentioned in Section 3.2, i.e., $p = 3$, $n_1 = 6$ and $n_2 = 509$. The field $\mathbb{F}_{p^n}$ is represented using $(I(x), h(y))$. The polynomials $h(y)$ and $I(x)$ and the generator $x + y^2$ are given in [19]. Further, we also used the target $T_0$ that was used in [19]. The largest prime divisor of $n$ is $d = 3 \cdot 509$.

We ran Algorithm 2 for two values of $B$, namely $B = 28$ and $B = 30$. In both cases, the value of $s$ satisfying (20) is 250 and so the degrees of the generated polynomials are $n_2 - s - 1 = 258$. Since $d = 3 \cdot 509$, $n_1 = 6$ and $s = 250$, the value of $r$ from (14) is 21.

Given $\alpha$, $U(x)$ is uniquely defined. Given $T_0(x)$ and $U(x)$, the matrices $M_0$ and $M_1$ are completely defined. Since $\mathbf{w}_1$ is fixed by (12), given $M_1$, the matrix $B$ and the particular solution $\mathbf{c}$ to (11) are completely defined. Further, given $B$ and $c$, the matrix $L$ and $\mathbf{d}$ are also completely defined. So, given $\alpha$ and $T_0(x)$, the matrix $L$ and the vector $\mathbf{d}$ are defined. Different values of $\mathbf{b}$ generates different values of $\mathbf{w}_0$ (equivalently, different values of $W_0(x)$) and so different values of $W(x)$. From (6), we have $\log W \equiv \log T_0 \bmod \ell$.

Below we provide the obtained values of $\mathbf{b}$ such that the corresponding $W(x)$'s are $B$-smooth for $B = 28$ and $B = 30$. Along with the $b$'s we also provide the smoothness probabilities.

$B = 28$: $\pi_2 \approx 2^{-29.5}$, $\mathbf{b} = (0, 1, 1, 1, 1, 0, 2, 1, 2, 1, 1, 1, 2, 1, 2, 1, 1, 0, 0, 2, 0)$.

$B = 30$: $\pi_2 \approx 2^{-26.7}$, $\mathbf{b} = (0, 0, 1, 2, 2, 1, 1, 0, 2, 0, 0, 0, 2, 2, 2, 0, 0, 2, 0, 0, 0)$.

In both the cases, the degrees of the generated polynomials are 258. So, the time for one-time computation and the average times for generating a polynomial and smoothness checking are also the same. The one-time computation took less than 2 hours.

To obtain an estimate of the times required per iteration, we averaged over 1000 iterations. The average time to generate a polynomial is about $10^{-5}$ seconds while the average time to check the polynomial for smoothness is about 0.4 seconds. As mentioned earlier, the advantage of the new method is that the time for generating a polynomial is negligible in comparison to the time for smoothness checking.

For the actual computations of $\mathbf{b}$ for $B = 28$ and $B = 30$, the iterative part of Algorithm 2 was parallelised as mentioned in Section 3.1. The subspace $\mathbb{F}_3^r$ was divided into disjoint subspaces to be searched by 320 parallel processes running on four servers have 100 cores each. The computation for $B = 30$ took less than a day while the computation for $B = 28$ took about 10 days. We did not have exclusive access to the servers during the execution of the programs. The server was loaded with long running R and Matlab programs by other users. Due to this, the exact times for the completion of our programs are not informative and so we do not report these times.

## 5 Conclusion

For small characteristic, composite extension degree fields, we have shown that in the initial splitting step, the cost of generating polynomials to be tested for smoothness can be brought down to $O(n \log(1/\pi))$ operations in $\mathbb{F}_p$ from the cost $O(n_2^2)$ multiplications in $\mathbb{F}_{p^{n_1}}$ that is required by the Guillevic splitting algorithm [19]. This improvement should help in the computation of future record discrete logarithm computations over such fields.

## References

[1] Gora Adj, Alfred Menezes, Thomaz Oliveira, and Francisco Rodríguez-Henríquez. Weakness of $\mathbb{F}_{6^{6\cdot 509}}$ for discrete logarithm cryptography. In Zhenfu Cao and Fangguo Zhang, editors, *Pairing-Based Cryptography - Pairing 2013 - 6th International Conference, Beijing, China, November 22-24, 2013, Revised Selected Papers*, volume 8365 of *Lecture Notes in Computer Science*, pages 20–44. Springer, 2013.

[2] Gora Adj, Alfred Menezes, Thomaz Oliveira, and Francisco Rodríguez-Henríquez. Computing discrete logarithms in $\mathbb{F}_{3^{6\cdot 137}}$ and $\mathbb{F}_{3^{6\cdot 163}}$ using Magma. In Çetin Kaya Koç, Sihem Mesnager, and Erkay Savas, editors, *Arithmetic of Finite Fields - 5th International Workshop, WAIFI 2014, Gebze, Turkey, September 27-28, 2014. Revised Selected Papers*, volume 9061 of *Lecture Notes in Computer Science*, pages 3–22. Springer, 2014.

[3] Gora Adj, Alfred Menezes, Thomaz Oliveira, and Francisco Rodríguez-Henríquez. Weakness of $\mathbb{F}_{6^{6\cdot 1429}}$ and $\mathbb{F}_{2^{4\cdot 3041}}$ for discrete logarithm cryptography. *Finite Fields and Their Applications*, 32:148–170, 2015.

[4] Leonard M. Adleman. The function field sieve. In Leonard M. Adleman and Ming-Deh A. Huang, editors, *ANTS*, volume 877 of *Lecture Notes in Computer Science*, pages 108–121. Springer, 1994.

[5] Leonard M. Adleman and Ming-Deh A. Huang. Function field sieve method for discrete logarithms over finite fields. *Inf. Comput.*, 151(1-2):5–16, 1999.

[6] Razvan Barbulescu, Pierrick Gaudry, Antoine Joux, and Emmanuel Thomé. A heuristic quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, volume 8441 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2014.

[7] Ian F. Blake, Ryoh Fuji-Hara, Ronald C. Mullin, and Scott A. Vanstone. Computing logarithms in finite fields of characteristic two. *SIAM Journal on Algebraic Discrete Methods*, 5(2):276–285, 1984.

[8] Ian F. Blake, Ronald C. Mullin, and Scott A. Vanstone. Computing logarithms in $GF(2^n)$. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings*, volume 196 of *Lecture Notes in Computer Science*, pages 73–82. Springer, 1984.

[9] Don Coppersmith. Fast evaluation of logarithms in fields of characteristic two. *IEEE Trans. Information Theory*, 30(4):587–593, 1984.

[10] Jérémie Detrey, Pierrick Gaudry, and Marion Videau. Relation collection for the function field sieve. In Alberto Nannarelli, Peter-Michael Seidel, and Ping Tak Peter Tang, editors, *IEEE Symposium on Computer Arithmetic*, pages 201–210. IEEE Computer Society, 2013.

[11] Philippe Flajolet, Xavier Gourdon, and Daniel Panario. The complete analysis of a polynomial factorization algorithm over finite fields. *J. Algorithms*, 40(1):37–81, 2001.

[12] Faruk Göloglu, Robert Granger, Gary McGuire, and Jens Zumbrägel. On the function field sieve and the impact of higher splitting probabilities - application to discrete logarithms in $\mathbb{F}_{2^{1971}}$ and $\mathbb{F}_{2^{3164}}$. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 109–128. Springer, 2013.

[13] Faruk Göloglu, Robert Granger, Gary McGuire, and Jens Zumbrägel. Solving a 6120-bit DLP on a desktop computer. In Lange et al. [24], pages 136–152.

[14] Robert Granger, Thorsten Kleinjung, and Jens Zumbrägel. Breaking '128-bit secure' supersingular binary curves – (or how to solve discrete logarithms in $\mathbb{F}_{2^{4\cdot1223}}$ and $\mathbb{F}_{2^{12\cdot367}}$). In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part II*, volume 8617 of *Lecture Notes in Computer Science*, pages 126–145. Springer, 2014.

[15] Robert Granger, Thorsten Kleinjung, and Jens Zumbrägel. Discrete logarithms in $GF(2^{9234})$. *NMBRTHRY list*, January 2014.

[16] Robert Granger, Thorsten Kleinjung, and Jens Zumbrägel. On the powers of 2. *IACR Cryptology ePrint Archive*, 2014:300, 2014.

[17] Robert Granger, Thorsten Kleinjung, and Jens Zumbrägel. On the discrete logarithm problem in finite fields of fixed characteristic. *IACR Cryptology ePrint Archive*, 2015:685, 2015.

[18] Aurore Guillevic. Computing individual discrete logarithms faster in $gf(p^n)$ with the NFS-DL algorithm. In Tetsu Iwata and Jung Hee Cheon, editors, *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part I*, volume 9452 of *Lecture Notes in Computer Science*, pages 149–173. Springer, 2015.

[19] Aurore Guillevic. Faster individual discrete logarithms in finite fields of composite extension degree. *Math. Comput.*, 88(317):1273–1301, 2019.

[20] Antoine Joux. Discrete Logarithms in GF $\left(2^{6168}\right)\left[=\mathrm{GF}\left(2^{257^{24}}\right)\right]$ . *NMBRTHRY list*, may 2013.

[21] Antoine Joux. A new index calculus algorithm with complexity $L(1/4+o(1))$ in small characteristic. In Lange et al. [24], pages 355–379.

[22] Antoine Joux and Cécile Pierrot. Discrete logarithm record in characteristic 3, GF $\left(3^{5.479}\right)$ a 3796-bit field. *NMBRTHRY list*, Sept 2014.

[23] Antoine Joux and Cécile Pierrot. Improving the polynomial time precomputation of Frobenius representation discrete logarithm algorithms - simplified setting for small characteristic finite fields. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, volume 8873 of *Lecture Notes in Computer Science*, pages 378–397. Springer, 2014.

[24] Tanja Lange, Kristin E. Lauter, and Petr Lisonek, editors. *Selected Areas in Cryptography - SAC 2013 - 20th International Conference, Burnaby, BC, Canada, August 14-16, 2013, Revised Selected Papers*, volume 8282 of *Lecture Notes in Computer Science*. Springer, 2014.

[25] Daniel Panario, Xavier Gourdon, and Philippe Flajolet. An analytic approach to smooth polynominals over finite fields. In Joe Buhler, editor, *Algorithmic Number Theory, Third International Symposium, ANTS-III, Portland, Oregon, USA, June 21-25, 1998, Proceedings*, volume 1423 of *Lecture Notes in Computer Science*, pages 226–236. Springer, 1998.