Faster individual discrete logarithms in non-prime finite fields with the NFS and FFS algorithms

Aurore Guillevic[1,2]

[1] University of Calgary, Canada
[2] Pacific Institute for the Mathematical Sciences, CNRS UMI 3069
aurore.guillevic@ucalgary.ca

Abstract. Computing discrete logarithms in finite fields is a main concern in cryptography. The best algorithms known are the Number Field Sieve and its variants in large and medium characteristic fields (e.g. GF(p^2), GF(p^{12})); the Function Field Sieve and the Quasi Polynomial-time Algorithm in small characteristic finite fields (e.g. GF($3^{6 \cdot 509}$)). The last step of the NFS and FFS algorithms is the individual logarithm computation. It computes a smooth decomposition of a given target in two phases: an initial splitting then a descent tree. While new improvements have been made to reduce the complexity of the dominating relation collection and linear algebra steps of NFS and FFS, resulting in a smaller factor basis (database of known logarithms of small elements), the last step remains of same difficulty. Indeed, we have to find a smooth decomposition of a typically large element in the finite field.
The method we propose improves the initial splitting phase and applies to any finite field of composite extension degree. It exploits the available subfields with a cheap (polynomial-time) linear algebra step, resulting in a much more smooth decomposition of the target. This leads to a new trade-off in the asymptotic complexity of the initial splitting step: for instance it is improved by a factor 2 in the exponent for FFS and $2^{1/3}$ in the exponent for NFS, for any finite field of even extension degree, and with a much smaller smoothness bound. In medium and large characteristic, it can be combined with Pomerance's Early Abort strategy. In small characteristic, it replaces the Waterloo algorithm of Blake, Fuji-Hara, Mullin and Vanstone. Moreover it reduces the width and the height of the subsequent descent tree.

Keywords: Finite field, discrete logarithm, number field sieve, function field sieve.

1 Introduction

1.1 The discrete logarithm problem (DLP)

The discrete logarithm problem is a major building block in cryptography since the Diffie-Hellman key exchange protocol in 1976 [19]. Given a cyclic finite group G, a generator g, and a scalar $0 \leq x < \#G$, the exponentiation $(g, x) \mapsto g^x$ is computed in polynomial time in the size of x. Given again G, g and an element y, computing a discrete logarithm x of y in basis g such that $g^x = y$ is intractable in well-chosen groups: it requires an exponential time e.g. for elliptic curves. The first groups proposed were finite fields, from prime fields to characteristic two fields. Nowadays the group of points of an elliptic curve is preferred, but for pairing-based cryptography purpose, large characteristic finite fields such as $\mathbb{F}_{p^{12}}$ are intensively used.

Since 2014 and the publication of discrete logarithms records in small characteristic fields (of prime extension degree [39,8] or composite extension degree [1,33,26]), and the publication of a Quasi Polynomial-Time algorithm (QPA) [10], computing DL in $\mathbb{F}_{2^n}$, $\mathbb{F}_{3^n}$ and small characteristic finite fields is much easier than was thought before and these fields should be avoided for DLP instances. The finite fields of the form $\mathbb{F}_{p^2}$, $\mathbb{F}_{p^3}$, ... $\mathbb{F}_{p^{12}}$ (of large and medium characteristic) are still used in pairing-based cryptography. The difficulty of computing DL in these non-prime fields was less studied than for prime fields and small characteristic fields.

The algorithms used for computing DL are the Number Field Sieve (NFS) [23,29], the Function Field Sieve (FFS) [17,3,4] and the QPA [10,26]. The NFS and FFS algorithms are made of four phases: polynomial selection (two polynomials are chosen), relation collection where relations between small elements are obtained, linear algebra (computing the kernel of a huge sparse matrix over an auxiliary

large prime finite field), and individual discrete logarithm computation. In NFS and FFS, the relation collection and linear algebra are the most costly parts [7,15,40]. In the practical implementations that mix the FFS and QPA algorithms, the recent improvements speed-up the relation collection and linear algebra when n is smooth. As a comparison, in Kleinjung's record [39] in $\mathrm{GF}(2^{1279})$ where 1279 is prime, the relation collection and linear algebra took 3 core-years, and the individual logarithm took 0.5 core-year. In Granger, Kleinjung and Zumbragel's record [47] in $\mathrm{GF}(2^{12\cdot 367})$ (a 4404-bit field), the relation collection and linear algebra took 0.56 core-year, the initial splitting took 0.07 core-year and the tree descent took 5.3 core-years.

1.2 Individual discrete logarithm

The complexity of this step depends on the initial parameters of the algorithm: the polynomials defining the two number fields or function fields. Let $q = p^n$ ($q = p$ for a prime finite field). The asymptotic complexity formulas for these subexponential algorithms use the L-notation:

$$L_q[\alpha, c] = \exp\Big(\big(c + o(1)\big)(\log q)^{\alpha}(\log\log q)^{1-\alpha}\Big) \quad \text{with } \alpha \in [0,1] \text{ and } c > 0\ .$$

The α parameter measures the gap between polynomial time ($\alpha = 0$) and exponential time ($\alpha = 1$). When the complexity relates to an algorithm for a prime field $\mathbb{F}_p$, we write $L_p[\alpha, c]$. We will also need the definition of smoothness. An integer is said to be B-smooth if all its prime divisors are less than B. An ideal in a number field is said to be B-smooth if it factors into prime ideals whose norms are bounded by B. A polynomial is said to be B-smooth w.r.t. its degree if all its irreducible factors have a degree smaller than B.

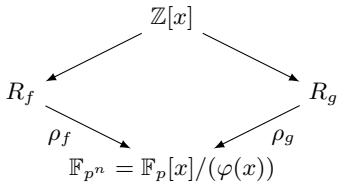

Fig. 1: NFS-DL diagram for $\mathbb{F}_{p^n}$

We now focus on the NFS algorithm. The first step of the NFS algorithm chooses two irreducible polynomials f and g defining two rings $R_f = \mathbb{Z}[x]/(f(x))$ and $R_g = \mathbb{Z}[x]/(g(x))$. After the relation collection step and the linear algebra step, a database of discrete logarithms of prime ideals in R_f and R_g of norm smaller than a bound B_0 are known. These elements form the *factor basis*. At the age of $L_q[1/2]$ complexity algorithms (for prime fields, $q = p$), the factor base was quite large (the smoothness bound B_0 was $L_q[1/2, \cdot]$) and the individual discrete logarithm was not a big challenge. The decreasing complexity from the Coppersmith–Odlzyko–Schroeppel (COS) algorithm [18] in $L_q[1/2, 1]$ to Gordon's algorithm in $L_q[1/3, 9^{1/3} \approx 2.08]$ was done by introducing a new polynomial selection allowing to handle elements of much smaller size in the relation collection: $L_q[2/3]$ instead of $L_q[1] = O(q)$. Combined with the linear algebra improvements of the block-Wiedemann algorithm, this allowed a new trade-off in the size of the parameters, and a much smaller complexity. Now the elements in the factor basis are bounded by $B = L_q[1/3, \beta]$, where $\beta = 0.96$ in the (generic) large characteristic case, and $\beta = 1.10$ in the medium characteristic case, which in both cases is much smaller than the previous bound of $L_q[1/2, 1/2]$. But at the end of the algorithm, we still have to decompose a given target from the finite field $\mathbb{F}_{p^n}$ into small elements of the factor basis. The factor basis is much smaller than before, whereas the target will always be any given element, large. This is the same issue with the FFS algorithm, and the QPA algorithm: the individual discrete logarithm is not so easy, and balancing its cost w.r.t. the relation collection and linear algebra is still challenging in the current practical implementations of the QPA algorithm.

1.3 Previous work

The target is an element in the finite field $\mathbb{F}_{p^n}$. A preimage of the target is an element r in one of the rings R_f or R_g such that $\rho(r)$ equals the target in $\mathbb{F}_{p^n}$. The aim of the individual logarithm computation is to obtain a smooth decomposition of this preimage: if it factors into prime ideals of known logarithms (in the factor basis), then one can obtain the logarithm of the preimage, then of the target.

Since there is no chance for a preimage of a target T_0 to be B-smooth, the individual discrete logarithm is done in two steps: an *initial splitting* of the target[3], then a *descent* phase[4]. The initial splitting is an iterating process that tries various targets $g^t T_0 \in \mathbb{F}_q^*$, where t is a known exponent (taken uniformly at random), until a B_1-smooth decomposition of the preimage is found, where $B_1 = L_q[2/3, \gamma] \gg B_0 = L_q[1/3, \beta]$ (B_0 is the factor basis smoothness bound: the discrete logarithm of any prime smaller than B_0 is known). The second phase starts a recursive process for each element less than B_1 but greater than B_0 obtained after the initial splitting phase. These elements are also called *special-*$\mathfrak{q}$.

Each of the special-$\mathfrak{q}$ elements are processed until a complete decomposition over the factor basis is found. Each special-$\mathfrak{q}$ obtained from the initial splitting is at the root of its descent tree. One finds a relation of prime ideals involving the special-$\mathfrak{q}$ itself and other ones whose norm is strictly smaller than the norm of the special-$\mathfrak{q}$. These smaller ideals form the new leaves of the descent tree. For each leaf, the process is repeated, until all the leaves are in the factor basis. The discrete logarithm of a special-$\mathfrak{q}$ output by the initial splitting can be computed by a tree traversal. This strategy is considered in [18, §6], [41, §7], [30, §3.5], [16, §4].

We will now focus on the initial splitting phase. This phase takes a preimage of the target and processes it with different techniques (in polynomial time) to improve its success probability. The generic process is sketched in Algorithm 1. The running-time of the whole step depends directly on the polynomials chosen to define the number fields or function fields, because it determines the size of the norm of the preimage in that number field. We will now explain the previous methods to improve the preimage smoothness probability.

Algorithm 1: Generic Initial Splitting (or Boot or Smoothing step)

Input: Target $T_0 \in \mathbb{F}_q$, generator g, subgroup order ℓ, bound B
Output: $t \in \mathbb{Z}/\ell\mathbb{Z}$, $\boldsymbol{T} \in \mathbb{Z}[x]$ a preimage of $T = g^t T_0$, such that $\boldsymbol{T}$ is B-smooth // $\log_g T_0 = \log_g \rho(\boldsymbol{T}) - t$

```
repeat
    take t at random in {1, ..., ℓ − 1}
    T ← g^t T_0
    T ← PREIMAGE(T)
until T is B-smooth
return T, t
```

Small characteristic fields: the Waterloo algorithm. In small characteristic, the finite field is most of the time $\mathbb{F}_{2^n}$ or $\mathbb{F}_{3^n}$, where n is prime in general setting, or composite in pairing-friendly setting. The first initial splitting algorithm, also known as the *Waterloo algorithm*, was introduced by Blake, Fuji-Hara, Mullin and Vanstone in 1984 [13,14] for computing discrete logarithms in $\mathbb{F}_{2^n}^*$.

The randomized target $T \in \mathbb{F}_{2^n}^*$ is represented by a polynomial $T(x)$ of degree $n-1$ of $\mathbb{F}_2[x]$. Let $I(x) \in \mathbb{F}_2[x]$ be an irreducible polynomial of degree n such that $\mathbb{F}_{2^n} = \mathbb{F}_2[x]/(I(x))$. The extended Euclidean algorithm is run on $T(x)$ and $I(x)$, to compute the GCD of the two polynomials (which is 1 since $I(x)$ is irreducible). At each iteration, the following equation holds [13, §2]:

$$v_i(x)T(x) + w_i(x)I(x) = u_i(x) \tag{1}$$

Reducing this equation modulo $I(x)$, one obtains $T(x) \equiv u_i(x)/v_i(x) \bmod I(x)$. The degree of $w_i(x)$ decreases while the degree of $v_i(x)$ increases, and the degree of $u_i(x)$ is less than the degree of $w_i(x)$. By

[3] also called *boot* or *smoothing step* in large characteristic finite fields

[4] in order to make no confusion with the mathematical *descent*, which is not involved in this process, we mention that in this step, the norm (with NFS) or the degree (with FFS) of the preimage *decreases*.

stopping the extended Euclidean algorithm at the state i where $\deg u_i(x), \deg v_i(x) \leq \lfloor n/2 \rfloor$, one obtains the initial splitting of $T(x)$ of degree $n-1$ into two polynomials $u_i(x), v_i(x)$ of degree at most $\lfloor n/2 \rfloor$. The joint probability of two polynomials of degree $n/2$ to be b-smooth is higher than the probability of one polynomial of degree $n-1$ to be b-smooth. The asymptotic complexity of the Waterloo algorithm was proved in [45,20]. This initial splitting algorithm was never improved in the pairing context, contrary to the other steps.

Prime fields. The prime field counterpart [41, §7] of the initial splitting uses the Rational Reconstruction algorithm: given a target $T \in \mathbb{Z}/p\mathbb{Z}$, one computes a rational reconstruction $u/v \equiv T \bmod p$ so that the preimage of T (the lift of T in $\mathbb{Z}$) and the ratio u/v have the same discrete logarithm, moreover u and v are two integers of size $p^{1/2}$, so that their joint probability to be smooth is higher than for the preimage of T, which is an integer of size p.

Large characteristic fields. In 2006 a generalization of the prime field and small characteristic field was proposed [31]. In 2015 a new technique [27] was proposed to use the subfields to reduce the size of the norm of the preimage. The idea was developed for extension degree n that are prime or even.

1.4 Contributions

We generalize the idea in [27] to any composite n by introducing a linear algebra step. We prove the following theorem which is a generalization of [27, Theorem 1].

Theorem 1. *Let $T \in \mathbb{F}_{p^n}^*$ an element which is not in a proper subfield of $\mathbb{F}_{p^n}$. We want to compute its discrete logarithm modulo a (large) prime ℓ, where $\ell \mid \Phi_n(p)$. Let K be a number field given by a polynomial selection method. Let d be the largest divisor of n, $d < n$ and $d = 1$ if n is prime.*

Then there exists a preimage $\boldsymbol{T}$ in K of $T \in \mathbb{F}_{p^n}^*$, such that $\log \rho(\boldsymbol{T}) \equiv \log T \bmod \ell$ and whose norm in K is bounded by $O(q^e)$, where $q = p^n$ and q^e equals

1. *$q^{1-d/n}$ for the GJL, Conjugation, Joux-Pierrot, Sarkar-Singh and TNFS-like methods (and for all the possible methods where $\|f\|_\infty = o(p)$);*
2. *$q^{\frac{3}{2}-\frac{d}{n}-\frac{1}{2n}}$ for the $JLSV_1$ method;*
3. *$q^{2-\frac{d}{n}-\frac{2}{D+1}}$ for the $JLSV_2$ method, where D is the degree of g (see Table 1).*

From this generalization, we develop a small characteristic counterpart to speed-up the initial splitting step when n is composite. Instead of using the Waterloo algorithm that splits a degree n polynomial into two degree $\lfloor n/2 \rfloor$ polynomials, we compute (with cheap linear algebra) a new polynomial of degree $n-d$ where $d \mid n$, $1 < d < n$, such that its discrete logarithm is the same modulo the order $\Phi_n(p)$ of cyclotomic subgroup. When n is even, one can take $d = n/2$ and compute a single degree $n/2$ polynomial (instead of two polynomials of degree $n/2$ with the Waterloo algorithm). This provides a very good practical improvement for the FFS and QPA algorithms. We compare the splitting probabilities of our new technique for two cryptographic-size examples: $\mathrm{GF}(3^{6\cdot 509})$ (a 4841-bit field) [2] and $\mathrm{GF}(3^{5\cdot 479})$ (a 3796-bit field) [33]. We show that our technique provides a very important improvement for the first case because n is even, and a competitive variant for the second case (we compute a degree $\frac{4}{5}479 = 383$ polynomial over $\mathbb{F}_{3^5}$ instead of two polynomials of degree $478/2 = 239$). Moreover, much less elements, each of much smaller degree, are obtained after the initial splitting step, which considerably reduces the depth and the width of the descent tree. This is also an important saving.

1.5 Organization of the paper.

In Section 2 we recall the preliminaries and the previous work of [27]. In Section 3 we adapt Barbulescu's work on Pomerance's Early Abort Strategy to non-prime finite fields. In Section 4 we generalize the previous work of [27] and give the algorithm to obtain the bounds of Theorem 1. In Section 5 we explain the generalization to small characteristic. Finally in Section 6 we provide two examples of cryptographic size in small characteristic and analyze the splitting probabilities.

2 Preliminaries

2.1 Polynomial Selection methods

There exists several polynomial selection methods to parameterize the NFS algorithm for large and medium characteristic finite fields. We give in appendix the algorithms and provide in Table 1 the properties of the polynomials (degree, coefficient size) that are relevant to compute a bound for the norm of a given element in each number field.

The properties are:

1. $\deg f, \deg g \geq n$;
2. f and g are irreducible and define two non-isomorphic number fields;
3. $\gcd(f \bmod p, g \bmod p) = \varphi$ is irreducible modulo p
4. the norm of the elements considered in the relation collection step is as small as possible.

Table 1: Properties: degree and coefficient size of the main polynomial selection methods for NFS-DL in $\mathbb{F}_q$, where $q = p^n$. The coefficient sizes are in $O(x)$. To lighten the notations, we simply write the x term. In the Joux–Pierrot method, the prime p can be written $p = P(x_0)$, where P is a polynomial of tiny coefficients and degree at least 2. Since the area is moving very fast since 2015, we mention that this table takes into account the methods known in Spring 2016. Very recently, Sarkar and Singh combined Kim's method with their method in [51,49,50]. Our method should apply as well.

method	$\deg f$	$\deg g$	$\|\|f\|\|_\infty$	$\|\|g\|\|_\infty$	f monic	g monic
JLSV_1	n	n	$q^{1/2n}$	$q^{1/2n}$	yes	no
JLSV_2	n	$D > n$	$q^{1/(D+1)}$	$q^{1/(D+1)}$	yes	no
GJL	$D+1 > n$	$D \geq n$	$\log p$	$q^{1/(D+1)}$	yes	no
Conjugation	$2n$	n	$\log p$	$q^{1/2n}$	yes	no
Joux-Pierrot	$n(\deg P)$	n	$\log p$	$q^{1/(n \deg P)}$	yes	yes
Sarkar-Singh[48], $n = d_1 d_2,\ D \geq d_2$	$d_1(D+1)$	$d_1 D$	$\log p$	$q^{\frac{1}{d_1(D+1)}}$	yes	no
TNFS+GJL [11]	$D > 1$	1	$\log p$	$q^{1/(D+1)}$	yes	no
TNFS+JLSV2 [36,37], $n = d_1 d_2$	d_1	$D \geq d_1$	$p^{d_1/(D+1)} = q^{1/(d_2(D+1))}$	$p^{d_1/(D+1)} = q^{1/(d_2(D+1))}$	yes	no
TNFS+Conj [6,37,28], $n = d_1 d_2$, $d_1 = O(\frac{1}{12^{\frac{1}{3}}}(\frac{\log q}{\log\log q})^{1/3})$	$2d_1$	d_1	$\log p$	$p^{1/2}$	yes	no

2.2 Norm Upper Bound in a Number Field

Let f be a monic irreducible polynomial over $\mathbb{Q}$ and let $K_f = \mathbb{Q}[x]/(f(x))$ a number field. Write $T \in K_f$ as a polynomial in x: $T = \sum_{i=0}^{\deg f - 1} t_i x^i$. The norm is defined by a resultant computation:

$$\text{Norm}_{K_f/\mathbb{Q}}(T) = \mathcal{N}(T) = \text{Res}(f, T)\ . \tag{2}$$

We use Kalkbrener's bound [35, Corollary 2] for an upper bound:

$$|\operatorname{Res}(f,T)| \leq \kappa(\deg f, \deg T)\|f\|_\infty^{\deg T}\|T\|_\infty^{\deg f}, \tag{3}$$

where $\kappa(n,m) = \binom{n+m}{n}\binom{n+m-1}{n}$ and $\|f\|_\infty = \max_{0\leq i\leq \deg f}|f_i|$ the absolute value of the greatest coefficient. An upper bound for $\kappa(n,m)$ is $(n+m)!$. We will use the following bound in Section 4:

$$\text{Norm}_{K_f/\mathbb{Q}}(T) \leq (\deg f + \deg T)!\|f\|_\infty^{\deg T}\|T\|_\infty^{\deg f}\ . \tag{4}$$

2.3 LLL algorithm

We recall an important property of the LLL algorithm [42] that we will widely use in this paper. Given a lattice $\mathcal{L}$ of $\mathbb{Z}^n$ defined by a basis given by an $n \times n$ matrix L, and parameters $\frac{1}{4} < \delta < 1$, $\frac{1}{2} < \eta < \sqrt{\delta}$, the LLL algorithm outputs a (η, δ)-*reduced basis* of the lattice. the coefficients of the first (shortest) vector are bounded by

$$(\delta - \eta^2)^{\frac{n-1}{4}} \det(L)^{1/n} .$$

With (η, δ) close to $(0.5, 0.999)$ (as in NTL or Magma), the approximation factor $C = (\delta - \eta^2)^{\frac{n-1}{4}}$ is bounded by 1.075^{n-1}. In the remaining of this paper, we will simply denote by C this LLL approximation factor.

2.4 Quadratic subfield cofactor simplification

In 2015 a new technique [27] was proposed to use the subfields to reduce the size of the norm of the preimage. The idea was developed for extension degree n that are prime or even, and uses the following lemma[5].

Lemma 1 ([27, Lemma 1]). *Let $T \in \mathbb{F}_{p^n}^* = \sum_{i=0}^{\deg T} t_i x^i$, with $\deg T < n$. Let ℓ be a non-trivial prime divisor of $\Phi_n(p)$. Let $T' = u \cdot T$ with u in a proper subfield of $\mathbb{F}_{p^n}$. Then*

$$\log T' \equiv \log T \bmod \ell . \tag{5}$$

Given an element $T \in \mathbb{F}_{p^n}^*$ (where n is even) not in a proper subfield, one of the ideas in [27] was to write T with one less coefficient, i.e. to find u in $\mathbb{F}_{p^2}$ such that $uT = T'$, where T' is represented by a polynomial of degree $n-2$ instead of $n-1$. It used a hand-written trick to compute u, given the finite field representation in a certain form. This was not a generic computation. Here is an example for $\mathbb{F}_{p^4}$.

Example 1. Let

$$p = \texttt{0xe3f367d542c82027f33dc5f3245769e676a5755d} \text{ and}$$
$$\ell = \texttt{0x6b455e0a014f1e30eaef7300bd4bb4258290fc5}$$

be a 160-bit prime and a 155-bit prime resp., such that $\ell \mid \Phi_4(p) = p^2 + 1$. We want to compute the individual discrete logarithm of a given target $T \in \mathbb{F}_{p^4}$ of 640 bits, modulo ℓ. The JLSV_1 method produces these kind of polynomials.

$$f = x^4 + \texttt{0xf19192168b16c1097c6b}x^3 - 6x^2 - \texttt{0xf19192168b16c1097c6b}x + 1$$
$$g = \texttt{0xf19192168b16c1593e06}\ x^4 + \texttt{0x96fafb4e168ad47e85f25}\ x^3 - \texttt{0x5a9696c87428888177424}\ x^2$$
$$-\texttt{0x96fafb4e168ad47e85f25}\ x + \texttt{0xf19192168b16c1593e06}$$

Let

$$T = \texttt{3141592653589793238462643383280742024009442661800} + \texttt{3653133493545225277827870597641597742307593336420}\ x$$
$$+\texttt{1992465999275654673118382378448634927673590850810}\ x^2 + \texttt{2424852442757209242435671622984325523344802298840}\ x^3$$

be our target whose coefficients are given by the decimals of π. Compute a monic equivalent of T to be $T^{(0)} = T/\text{LT}(T)$ (i.e. divides T by its leading coefficient). Compute a monic degree 2 equivalent $T^{(1)}$ of T. Define the lattice made of the coefficients $t_i^{(1)}$ of $T^{(1)}$ and $t_i^{(0)}$ of $T^{(0)}$:

$$L = \begin{bmatrix} p & 0 & 0 & 0 \\ 0 & p & 0 & 0 \\ t_0^{(1)} & t_1^{(1)} & 1 & 0 \\ t_0^{(0)} & t_1^{(0)} & t_2^{(0)} & 1 \end{bmatrix}$$

Compute a reduced lattice with the LLL algorithm, then define $R = r_0 + r_1 x + r_2 x^2 + r_3 x^3$ made of the coefficients of the shortest vector output by LLL. The coefficients are bounded by $Cp^{1/2}$, where C is the LLL factor. Then $\log \rho(R) \equiv \log T \bmod \ell$ by construction since R is a $\mathbb{F}_{p^2}$-linear combination of T. Moreover,

[5] As stated in [27], this idea was also used in pairing implementation to remove the denominators in the Miller loop computation [12].

a bound of the norm of T in the number field K_f is $O(\|R\|_\infty^4 \|f\|_\infty^3) = O(q^{7/8})$. In this example we obtain $R = 209330360557196451749 74x^3 + 312478973450133244613423x^2 + 233943694070133655633312x - 345482152256021836890951$ where $\log_2 \|R\|_\infty = 78.2$ and

$$\mathcal{N}_{K_f/\mathbb{Q}}(R) = -685765181338230588100436183388401246932895620798857777307572356644718568737152308 2\backslash$$
$$438210533901754982264350608123592428549388771530262577675862993372597335974129921 80$$

of 548 bits.

We explain in Section 4 how to generalize this idea of quadratic subfield simplification to any subfield, to speed-up the initial splitting of the individual discrete logarithm with NFS. Moreover, we show in Section 5 how to apply it to small characteristic finite fields and dramatically improve the initial splitting step in that case as well. We first recall the *early abort strategy* of Pomerance, analyzed by Barbulescu for finite fields, to speed-up the factorization of the norm in the initial splitting step.

3 Pomerance's Early Abort Strategy

Before going into the specificities of finite field extensions, we recall a generic strategy to improve the initial splitting step developed in the late 70's and early 80's. It considers the B-smooth test as a costly black box test. More precisely, the larger B, the more costly test. This strategy is known as the Early Abort Strategy. This section presents the results from Pomerance and Barbulescu adapted to the notations of this paper. The results of this section are entirely stated from Pomerance [46] and Barbulescu [5, Chapter 4].

Morrison–Brillhart presented their factorization of the seventh Fermat number $F_7 = 2^{2^7} + 1$ in [44] with the continued fraction method. They needed to factor integers Q_n of size about $\sqrt{F_7}$ over a factor basis of up to a thousand of small primes. They sped-up their algorithm by exploiting the fact that if all the prime factors of Q_n less than the factor basis bound B were already found, and if the remaining part of Q_n is less than B^2, then this last part should be a prime, and the factorization is actually complete. They extended this strategy in [44, Remark 4.2]: if after trial division of Q_n by half of the primes in the factor basis, the remaining part of Q_n is still larger than 10^{15}, the factorization is aborted. These algorithmic improvements lead to more than a factor two speed-up in practice in their computations.

Pomerance formulated and analyzed this early-abort strategy, to improve the running-time of the sub-exponential factorization algorithms of Dixon and Morrison–Brillhart in [46]. These two algorithms need to test the smoothness of large integers, of size bounded by N or $\sqrt{N}$, where N is the number to be factored. Barbulescu adapted this strategy to the smoothing step of the individual discrete logarithm computation in prime fields $\mathbb{F}_q$, in [5, Chapter 4], where again integers smaller than q are tested for smoothness. Morrison-Brillhart and Pomerance, and later Barbulescu consider different smoothness tests (trial division in 1982, elliptic curve method (ECM) in 2013) which both have a non-negligeable cost.

Pomerance's early abort strategy can be adapted to extensions of finite fields $\mathbb{F}_q = \mathbb{F}_{p^n}$ in the same way as Barbulescu's work. In this setting, integers smaller than q^e are tested for B_1-smoothness, where e is a fixed integer determined by the polynomial selection, and B_1 is a smoothness bound of the form $B_1 = L_q[\alpha, \gamma]$, where $0 < \alpha < 1$ and γ is a non-zero real number. Assuming that the norm of the randomized target is of size $q^e = L_q[1, e]$, this strategy improves the expected running time to find a smooth initial splitting from $L_q[1/3, (3e)^{1/3}]$ to $L_q[1/3, (3e)^{1/3}(23/27)^{2/3}]$, where $(23/27)^{2/3} \approx 0.9$. The optimal B_1 (a.k.a. "special-q") size is however increased from $L_q[2/3, (e^2/3)^{1/3}]$ to $L_Q[2/3, (e^2/3)^{1/3}(27/23)^{2/3}]$, where $(23/27)^{2/3} \approx 1.1$. The technique is the same and this is only a matter of rewriting formulas with an additional e parameter. However the L notation changed since 1982 because the constant α is no longer equal to $1/2$ but to $1/3$ or $2/3$. We use nowadays notations $L_q[\alpha, c]$.

Pomerance observed (see [46, Section 3]) that one smoothness test costs $L_N[1/2, 1/2]$ (a $L_N[1/2, 1/2]$-smoothness test was made by trial division at that time) and because of the low smoothness probability, in most of the cases the test withdraw the candidate. So he proposed to reduce the smoothness bound B_1 to B_1^θ where $0 < \theta < 1$, so that the test costs less. However the number tested will have a much smaller chance to be B_1^θ-smooth, hence Pomerance proposed to look at the size of the remaining part, which is not B_1^θ-smooth. If this remaining part is small enough (smaller than N^{1-b} for a $0 < b < 1$ to be determined), then it has a good chance to actually be B_1-smooth, hence the whole number will have a good chance to be B_1-smooth. To the contrary if the remaining part is larger than N^{1-b}, then the

probability of being B_1-smooth is so small that one can abort the test at this point and start with a new candidate. The general idea is to decide earlier if the integer tested has a good chance to be B_1-smooth, by performing only a B_1^θ-smoothness test. This will reduce the overall cost of the initial splitting step.

3.1 Early Abort Strategy with one test

We write in Algorithm 2 Pomerance's Early Abort Strategy. We give the results for non-prime fields in the following. A technical proof can be found in Appendix E.

Algorithm 2: Pomerance's Early Abort Strategy

Input: target $T_0 \in \mathbb{F}_q^*$ of prime order ℓ, generator g, `PREIMAGE` function and $e > 0$ such that Norm(`PREIMAGE`$(\cdot)) \leq q^e$, smoothness bound B_1, real numbers $\theta, b \in]0,1[$ (it was shown in [5, Chapter 4] that $\theta = 4/9$ and $b = 8/23$ are the optimal values)

Output: t such that Norm(`PREIMAGE`$(g^t T_0)$) is B_1-smooth, and its B_1-smooth decomposition

```
repeat
    repeat
        pick t in {1, ..., ℓ − 1}
        m ← Norm(PREIMAGE(g^t T_0))
        m_0 ← ECM (m, B_1^θ)                          // cost: L_{B_1^θ}[1/2, √2]
        m_1 ← m/m_0                                   // m_0 is the B_1^θ-smooth part of m
    until m_1 ≤ q^{e(1−b)}
    m_2 ← ECM(m_1, B_1)                               // cost: L_{B_1}[1/2, √2]
until m_1 is B_1-smooth, i.e. m_1 = m_2
return t, B_1-smooth decomposition of m
```

Lemma 2 (Pomerance's Early Abort Strategy with one test). *The asymptotic complexity of the Initial Splitting step of an integer of size q^e with one early-abort test is*

$$L_q\left[\frac{1}{3}, (3e)^{1/3}\left(\frac{23}{27}\right)^{2/3}\right],$$

where the smoothness bound is

$$B = L_q\left[2/3, \gamma = (e^2/3)^{1/3}\left(\frac{27}{23}\right)^{2/3}\right],$$

and the two parameters b and θ of the Early Abort Strategy as in Algorithm 2 are

$$b = 8/23 \text{ and } \theta = 4/9.$$

Proof (Proof of Lemma 2). Barbulescu showed in [5, eq. (4.16), (4.17), (4.18)] that the expected running time of the initial splitting step of an integer N is $L_N[1/3, c]$ where $c = (23/3)^{2/3}/3$, $\theta = 4/9$ and $b = 8/23$. We replace N by q^e and obtain an expected running-time of

$$\begin{aligned} L_{q^e}[1/3, c] &= \exp\left((c + o(1))(\log q^e)^{1/3}(\log\log q^e)^{2/3}\right) \\ &= \exp\left((c + o(1))e^{1/3}(\log q)^{1/3}(\log e + \log\log q)^{2/3}\right) \\ &= L_q[1/3, ce^{1/3}] = L_q[1/3, (3e)^{1/3}(23/27)^{2/3}] \end{aligned}$$

The parameters θ and b do not depend on N.

3.2 Early Abort Strategy with several tests

Algorithm 3 presents Barbulescu's admissibility strategy with several tests.

Algorithm 3: Pomerance's early abort strategy with several tests

Input: target $T_0 \in \mathbb{F}_q^*$ of prime order ℓ, generator g, `PREIMAGE` function and $e > 0$ such that Norm(`PREIMAGE`$(\cdot)$)$\leq q^e$, smoothness bound B_1, number of tests $k \geq 0$,
array of positive real numbers $\mathbf{b} = [b_0, b_1, \ldots, b_k]$ where $0 < b_i \leq 1$, and $\sum_{i=0}^{k} b_i = 1$
array of positive real numbers $\boldsymbol{\theta} = [\theta_0, \ldots, \theta_k = 1]$ where $\theta_i < \theta_{i+1}$
Output: t such that Norm(`PREIMAGE`$(g^t T_0)$) is B_1-smooth, and its B_1-smooth decomposition

repeat
  pick t in $\{1, \ldots, \ell - 1\}$
  $m \leftarrow$ Norm(`PREIMAGE`$(g^t T_0)$)
  $m_i \leftarrow m$
  $i \leftarrow 0$
  $\Sigma_b \leftarrow 0$
  repeat
    $s_i \leftarrow$ `ECM` $(m_i, B_1^{\theta_i})$ `// cost:` $L_{B_1^{\theta_i}}[1/2, \sqrt{2}]$
    $m_i \leftarrow m_i / s_i$ `//` s_i `is the` $B_1^{\theta_i}$`-smooth part of` m_i
    $\Sigma_b \leftarrow \Sigma_b + b_i$
    $i \leftarrow i + 1$ `// at this point,` $\Sigma_b = \sum_{j=0}^{i-1} b_j$
  until $(i > k)$ `OR` $(m_i == 1)$ `OR` $(m_i > q^{e(1-\Sigma_b)})$
until $m_i == 1$ *so that* m *is* B_1*-smooth*
return t, B_1*-smooth decomposition of* m

Lemma 3. *The asymptotic complexity of the Initial Splitting step of an integer of size q^e with k early-abort tests is $L_q[1/3, c]$ where*

$$c = (3e)^{1/3} \left(1 - \frac{4}{19}\left(1 - \left(\frac{2}{3}\right)^{3k}\right)\right)^{2/3},$$

the smoothness bound is $B = L_q[2/3, \gamma]$, where

$$\gamma = (e^2/3)^{1/3} \left(1 - \frac{4}{19}\left(1 - \left(\frac{2}{3}\right)^{3k}\right)\right)^{-2/3},$$

the bound b_i for $0 \leq i \leq k-1$ on the remaining part m_i in Algorithm 3 is

$$b_i = \left(\frac{2}{3}\right)^{3(k-i)} \left(1 - \frac{4}{19}\left(1 - \left(\frac{2}{3}\right)^{3k}\right)\right)^{-1}$$

and the exponent θ_i for $0 \leq i \leq k$ is

$$\theta_i = \left(\frac{4}{9}\right)^{k-i}.$$

Proof (Proof of Lemma 3). Again we replace N by q^e in Barbulescu's formulas [5, eq. (4.27), (4.28), (4.29)] to obtain Lemma 3. An extended proof is provided in Appendix E.

We will combine this strategy with our new initial splitting step to improve the overall initial splitting step running-time of the individual logarithm computation.

4 Faster initial splitting step in large characteristic

The idea in [27] was to reduce as much as possible the degree of the polynomial in $\mathbb{F}_p[x]$ representing the target T in $\mathbb{F}_{p^n}$, which was at most $n-1$. For n even, [27] proposed a hand-written trick to cancel the leading term. The generalization of this idea is to choose the largest proper divisor d of n, $1 \leq d < n$, and compute the sequence of d elements $T^{(i)}$ in $\mathbb{F}_{p^n}$ for $1 \leq i \leq d$ such that $T^{(i)}$ is of degree $n - i$ instead of $n-1$, is monic, and $\log T^{(i)} \equiv \log T \bmod \ell$. This requires elementary linear algebra. Let $[1, U, \ldots, U^{d-1}]$ be a polynomial basis of $\mathbb{F}_{p^d}$, where U is represented by an element in $\mathbb{F}_{p^n}$. One defines the $d \times n$ matrix

whose rows are made of the coefficients (in $\mathbb{F}_p$) of the elements U^iT. Then one computes a reduced row-echelon form of this matrix. Reduced means that the most right non-zero element is equal to 1 (in other words, each row is made of the coefficients of a monic element of $\mathbb{F}_p[x]$). We use this as a building block in Algorithm 4 in combination with the LLL algorithm, to reduce the size of the coefficients of a preimage in $\mathbb{Z}[x]$ of T. For clarity, we give an example for $\mathbb{F}_{p^6}$ right now, before giving the generic algorithm.

Example 2. Let $p = 314159265358979323846343 59$ a 85-bit prime made of the first 26 decimals of π, $p = \lfloor 10^{25}\pi \rfloor + 7926$, so that $\mathbb{F}_{p^6}$ is a 509-bit finite field. Moreover $\Phi_6(p) = p^2 - p + 1$ is the 170-bit prime $\ell = 98696044010893586188394702151308074053683373870652 3$. We want to compute discrete logarithms in the order-ℓ cyclotomic subgroup of $\mathbb{F}_{p^6}$. The JLSV_1 method computes two polynomials f, g where $\deg f = \deg g = 6$, and $\|f\|_\infty \approx \|g\|_\infty \approx p^{1/2}$. In our example, we have $\log_2 \|f\|_\infty = 44.67$ and $\log_2 \|g\|_\infty = 46.67$ (and $\frac{1}{2}\log_2 p = 42.35$).

$$\begin{aligned} f &= x^6 - 11209975711932x^5 - 28024939279845x^4 - 20x^3 + 28024939279830x^2 + 11209975711938x + 1 \\ g &= 5604994576830x^6 + 20986447533158x^5 - 31608799819555x^4 - 1120998915366003x^3 - 52466118832895x^2 \\ &\quad + 12643519927822x + 5604994576830 \end{aligned}$$

Let T_0 be our target in $\mathbb{F}_{p^6} = \mathbb{F}_p[x]/(f(x))$ whose coefficients are made of the decimals of π (starting at the 26-th decimal, since the first 25 ones were already used for p).

$$\begin{aligned} T_0 = {} & 642770498858150816216245 5x^5 + 162400524326938996131777 38x^4 + 450939028378094990902013 9x^3 \\ & + 38683743594457576475914 44x^2 + 820975591360211292080812 2x + 327950288419716939937510 5 \end{aligned}$$

Let $g = x + 3$ a generator of $\mathbb{F}_{p^6}$ ($x + 3$ is primitive). Let $(1, U, U^2)$ be a polynomial basis of $\mathbb{F}_{p^3}$ considered as an implicit subfield of $\mathbb{F}_{p^6}$, where $U = g^{1+p^3} = \mathrm{Norm}_{\mathbb{F}_{p^6}/\mathbb{F}_{p^3}}(g)$. We compute the reduced row-echelon form E of the matrix $\begin{bmatrix} T \\ UT \\ U^2T \end{bmatrix}$ to be $E =$

$$\begin{bmatrix} 762145752956632917638767 7 & 264722713391354819914554 67 & 756132945945174411593996 & 1 & 0 & 0 \\ 134438789144443053821511 70 & 201541926965205753519795 51 & 143024722625793988922595 22 & 117078875388522571611920 89 & 1 & 0 \\ 103738016569731364064693 82 & 175542042568905028233328 99 & 845362156961145873638013 7 & 465847860837277531023067 7 & 227680000081689917458593 86 & 1 \end{bmatrix}$$

Then we reduce with LLL the following lattice defined by the (6×6)-matrix, where $E_{i,j}$ stands for the coefficient of the i-th row and j-th column of the above matrix E:

$$L = \begin{bmatrix} p & 0 & 0 & 0 & 0 & 0 \\ 0 & p & 0 & 0 & 0 & 0 \\ 0 & 0 & p & 0 & 0 & 0 \\ E_{1,1} & \dots & E_{1,3} & 1 & 0 & 0 \\ E_{2,1} & \dots & & E_{2,4} & 1 & 0 \\ E_{3,1} & \dots & & & E_{3,5} & 1 \end{bmatrix}$$

The coefficients of the shortest vector of $\mathrm{LLL}(L)$ gives us a reduced target R such that $\log_2 \|R\|_\infty = 41.36 < \frac{1}{2}\log_2 p = 42.34$ and $\log R \equiv \log T_0 \bmod \ell$ (in other words, $(T/R)^{\frac{p^6-1}{\ell}} = 1$).

$$\begin{aligned} R = {} & 2833685018960x^5 - 2185315247519x^4 + 1284018747482x^3 + 2833796663216x^2 \\ & - 104495515254x + 1247379641337\ . \end{aligned}$$

The norm of $R \in \mathbb{Z}[x]$ mapped to $\boldsymbol{R}$ in $K_f = \mathbb{Q}[x]/(f(x))$ is

$$\begin{aligned} \mathcal{N}_{K_f/\mathbb{Q}}(\boldsymbol{R}) = {} & 15638474059380084621275254534833796503863915102901358863234502536978593 \backslash \\ & 95113777528998482652719242035434612948213969643192470447970552596060189 \end{aligned}$$

of 470 bits, which is very close to $\log_2 q^{11/12} = 466$ bits.

Algorithm 4: Initial splitting with a degree d subfield cofactor

Input: Finite field $\mathbb{F}_{p^n}$, composite n
irreducible polynomial ψ s.t. $\mathbb{F}_{p^n} = \mathbb{F}_p[x]/(\psi(x))$
prime order subgroup $\ell \mid \Phi_n(p)$
generator g (of the order ℓ subgroup)
target $T_0 \in \mathbb{F}_{p^n}$
smoothness bound B_1
Output: $t \in \{1, \ldots, \ell - 1\}$, $R \in \mathbb{Z}[x]$ s.t. $\log_g \rho(R) \equiv t + \log_g T_0$, and $\mathrm{Norm}_{K_f}(\boldsymbol{R})$ is B_1-smooth

$d \leftarrow$ the largest divisor of n, $1 < d < n$
Compute a polynomial basis $(1, U, U^2, \ldots, U^{d-1})$ of the subfield $\mathbb{F}_{p^d}$, where $U \in \mathbb{F}_{p^n}$
repeat
  take $t \in \{1, \ldots, \ell - 1\}$ uniformly at random
  $T \leftarrow g^t T_0$ in $\mathbb{F}_{p^n}$
  Define the $d \times n$ matrix $A_{(d,n)} = \begin{bmatrix} T \\ UT \\ \vdots \\ U^{d-1}T \end{bmatrix}$
  $E_{(d,n)} \leftarrow \mathrm{ReducedRowEchelonForm}(A_{(d,n)})$
  Define a $\deg f \times \deg f$ lattice by the lower triangular matrix

$$L = \begin{bmatrix} pI_{(d,d)} & & & \\ E_{(d,n)} & & & \\ & \psi(x) & & \\ & & \ddots & \\ & & & x^{\deg f - n - 1}\psi(x) \end{bmatrix}$$

$$= \begin{bmatrix} p & & & & & & & & \\ & \ddots & & & & & & & \\ & & p & & & & & & \\ E_{1,1} & & \cdots & E_{1,n-d-1} & 1 & & & & \\ \vdots & & & & \ddots & & & & \\ E_{d,1} & & & \cdots & & E_{d,n-1} & 1 & & \\ \psi(x) & & & & & & & 1 & \\ & \ddots & & & & & & & \ddots \\ & & x^{\deg f - n - 1}\psi(x) & & & & & & 1 \end{bmatrix}$$

  Compute $\mathrm{LLL}(L)$
  $R \leftarrow$ the shortest vector output by LLL
until $\mathrm{Norm}_f(\boldsymbol{R})$ *is* B_1*-smooth*
return t, R

Proposition 1. *The size of the coefficients of the element R output by the LLL algorithm in Algorithm 4 is $||R||_\infty = Cp^{(n-d)/\deg f}$ where C is the LLL approximation factor. Its norm is* $\mathrm{Norm}_f(\boldsymbol{R}) = O(p^{n-d}||f||_\infty^{\deg f-1})$.

Proof (Proof of Proposition 1). The determinant of L is p^{n-d} since L is triangular. L is a $\deg f \times \deg f$ matrix. The bound on the size of the coefficients of the shortest vector output by LLL is $||R||_\infty \leq C(\det L)^{1/\deg f} = Cp^{(n-d)/\deg f}$. We use the approximation on the norm of an element to get $\mathrm{Norm}_f(\boldsymbol{R}) = O(p^{n-d}||f||_\infty^{\deg f-1})$.

Proposition 2. *The reduced element R output by Algorithm 4 is such that* $\log_g R \equiv \log_g g^t T_0 = \log_g T_0 + t \bmod \ell$.

Proof (Proof of Proposition 2). Each row of the row-echelon matrix E represents a $\mathbb{F}_p$-linear combination of the d elements U^iT, $0 \leq i \leq d-1$, i.e. an element $\sum_{i=0}^{d-1} \lambda_i U^i T$, where $\lambda_i \in \mathbb{F}_p$. We can factor T in the expression. The element $\sum_{i=0}^{d-1} \lambda_i U^i$ lies in $\mathbb{F}_{p^d}$ by construction. So each row represents an element $T_i = u_i T$, where $u_i \in \mathbb{F}_{p^d}$, so that $\log T_i \equiv \log T \bmod \ell$ by Lemma 1.

The second part of the proof uses the same argument: the short vector output by the LLL algorithm is a linear combination of the rows of the matrix L. Each row represents either 0 or a $\mathbb{F}_{p^d}$-multiple of T, hence the short vector is also a $\mathbb{F}_{p^d}$-multiple of T. We conclude with Lemma 1 that $\log R \equiv \log T \bmod \ell$.

We state Lemma 4 from [27, Lemma 1] on the running-time to find boots whose norm is bounded by a given bound B_1. The time needed to compute the reduced row-echelon $d \times n$ matrix E is in $O(n^3)$ which is polynomial in n [21]. This time is then negligible compared to any $L_q[\alpha > 0]$.

Lemma 4 ([27, Lemma 1] Running-time of B-smooth decomposition). *Let $T_0 \in \mathbb{F}_q$ of order ℓ. Take at random $t \in [1, \ell-1]$ and assume that the norm $\mathcal{N}_t$ of a preimage of $g^t T_0 \in \mathbb{F}_q$, in the number field K_f, is bounded by $q^e = L_q[1, e]$. Write $B = L_q[\alpha_B, \gamma]$ the smoothness bound for $\mathcal{N}_t$. Then the lower bound of the expected running time for finding t s.t. the norm $\mathcal{N}_t$ of $T_0 g^t$ is B-smooth is $L_q[1/3, (3e)^{1/3}]$, obtained with $\alpha_B = 2/3$ and $\gamma = (e^2/3)^{1/3}$.*

We apply the formulas $\|R\|_\infty = p^{(n-d)/\deg f}$ and $\mathrm{Norm}_f(\boldsymbol{R}) \approx \|R\|_\infty^{\deg f}\|f\|_\infty^{\deg R}$ to obtain the norm bounds of Table 2.

Table 2: Norm bound of the preimage, and booting step complexity. The various TNFS versions of NFS use one of the classical polynomial selections, and the norm of the target is the same as for the NFS algorithm with the same polynomial selection method.

polynomial selection	norm bound				booting step $L_q[\frac{1}{3}, c]$			
	naive	JLSV06 [31]	[27]	this work	old e	old $c = (3e)^{1/3}$	Early-abort 1 test [5, (4.18)] $c = (3e)^{1/3}(\frac{23}{27})^{2/3}$	this work, 1 test $c = (3e)^{1/3}(\frac{23}{27})^{2/3}$
GJL	$q^{1+\frac{1}{n}}$	q	$q^{1-1/n}$	$q^{1-d/n}$	1	1.442	1.296	$\left(3(\frac{23}{27})^{2/3}(1-\frac{d}{n})\right)^{1/3}$
Conj	q^2	q	$q^{1-1/n}$	$q^{1-d/n}$	1	1.442	1.296	$\left(3(\frac{23}{27})^{2/3}(1-\frac{d}{n})\right)^{1/3}$
JLSV$_1$	$q^{\frac{3}{2}-\frac{1}{2n}}$	q^2	$q^{\frac{3}{2}(1-\frac{1}{n})}$	$q^{\frac{3}{2}-\frac{d}{n}-\frac{1}{2n}}$	$\frac{3}{2}$	1.651	1.484	$\left(3(\frac{23}{27})^{2/3}(\frac{3}{2}-\frac{d}{n}-\frac{1}{2n})\right)^{1/3}$
JLSV$_2$	q^2	$q^{3-\frac{2}{D+1}}$	$q^{2-\frac{1}{n}-\frac{2}{D+1}}$	$q^{2-\frac{d}{n}-\frac{2}{D+1}}$	2	1.817	1.633	$\left(3(\frac{23}{27})^{2/3}(2-\frac{d}{n}-\frac{2}{D+1})\right)^{1/3}$
Sarkar-Singh	$q^{\frac{D+1}{d_2}}$	q	$q^{1-1/n}$	$q^{1-d/n}$	1	1.442	1.296	$\left(3(\frac{23}{27})^{2/3}(1-\frac{d}{n})\right)^{1/3}$

polynomial selection	practical values of c, without an Early Abort test, with one test, with k tests where $k = 5$														
	$n = 2$				$n = 3$				$n = 4$				$n = 5$				$n = 6$			
	e	c			e	c			e	c			e	c			e	c		
		0 test	1 test	k tests		0 test	1 test	k tests		0 test	1 test	k tests		0 test	1 test	k tests		0 test	1 test	k tests
GJL	$\frac{1}{2}$	1.145	1.029	0.98	$\frac{2}{3}$	1.260	1.132	1.08	$\frac{1}{2}$	1.145	1.029	0.98	$\frac{4}{5}$	1.339	1.203	1.14	$\frac{1}{2}$	**1.145**	**1.029**	**0.98**
Conj	$\frac{1}{2}$	1.145	1.029	0.98	$\frac{2}{3}$	1.260	1.132	1.08	$\frac{1}{2}$	1.145	1.029	0.98	$\frac{4}{5}$	1.339	1.203	1.14	$\frac{1}{2}$	**1.145**	**1.029**	**0.98**
JLSV$_1$	$\frac{3}{4}$	1.310	1.178	1.12	1	1.445	1.296	1.23	$\frac{7}{8}$	1.379	1.240	1.18	$\frac{6}{5}$	1.533	1.377	1.31	$\frac{11}{12}$	**1.401**	**1.259**	**1.20**
JLSV$_1$ Gröbner																	$\frac{5}{6}$	**1.357**	**1.220**	**1.16**

Example 3. Assume that n is even and let $T \in \mathbb{F}_{p^n}$. Compute a polynomial basis $(1, U, U^2, \ldots, U^{n/2-1})$ of the subfield $\mathbb{F}_{p^{n/2}}$. Let

$$A = \begin{bmatrix} T \\ UT \\ \vdots \\ U^{n/2-1}T \end{bmatrix} \text{ and compute } E = \begin{bmatrix} E_{1,1} & \ldots & E_{1,\frac{n}{2}-1} & 1 & 0 & \ldots & 0 \\ \vdots & & & & \ddots & \ddots & \vdots \\ \vdots & & & & & \ddots & 0 \\ E_{\frac{n}{2}} & & & \ldots & & E_{\frac{n}{2},\frac{n}{2}-1} & 1 \end{bmatrix}$$

to be the reduced echelon form of A. Then we define the lower triangular matrix made of the $n/2 \times n/2$ identity matrix with p on the diagonal in the upper left quarter, the $n/2 \times n/2$ zero matrix in the upper right quarter, and the $n/2 \times n$ matrix E in reduced echelon form in the lower half. Moreover if $\deg(f) > n$ then we add $(\deg f - n - 1)$ rows made of the coefficients of $x^i\psi$ where $\mathbb{F}_{p^n} = \mathbb{F}_p[x]/(\psi(x))$, for $0 \leq i < \deg f - n - 1$. Finally we apply the LLL algorithm to this matrix. The short vector gives us a preimage $\boldsymbol{R}$ whose norm is bounded by $q^{1/2}$ with a polynomial selection such that $\|f\|_\infty = O(1)$ (such as Conj and GJL). Applying Lemma 4, we set the bound B_1 to be $B_1 = L_q[2/3, ((1/2)^2/3)^{1/3} \approx 0.436]$. The running-time of Algorithm 4 will be $L_q[1/3, (3/2)^{1/3} \approx 1.144]$. We obtain preimages $\boldsymbol{R}$ whose norm is bounded by $q^{1-\frac{1}{2n}}$ with the JLSV$_1$ polynomial selection method as shown in Example 2. Applying Lemma 4, we set the bound B_1 to be $B_1 = L_q[2/3, ((1-\frac{1}{2n})^2/3)^{1/3}]$. The running-time of Algorithm 4 will be $L_q[1/3, (3(1-\frac{1}{2n}))^{1/3}]$.

4.1 Other variant with Gröbner basis or resultants for $\mathbb{F}_{p^6}$

We consider the finite field $\mathbb{F}_{p^6}$. We will use the two subfields $\mathbb{F}_{p^2}$ and $\mathbb{F}_{p^3}$ to cancel three coefficients. Let $U \in \mathbb{F}_{p^6}$ such that $(1, U, U^2)$ is a basis of $\mathbb{F}_{p^3} \subset \mathbb{F}_{p^6}$. Let $V \in \mathbb{F}_{p^6}$ such that $(1, V)$ is a basis of $\mathbb{F}_{p^2} \subset \mathbb{F}_{p^6}$. We want to solve

$$uvwT = (u_0 + u_1U + u_2U^2)(v_0 + v_1V)wT = T'$$

where $u \in \mathbb{F}_{p^3}$, $v \in \mathbb{F}_{p^2}$, $w \in \mathbb{F}_p$ and $T' \in \mathbb{F}_{p^6}$ is represented by an element of degree 2. To simplify, we set $u_2 = v_1 = 1$ so that we obtain equations where we can eliminate recursively the variables by computing resultants. We compute u, v such that $uvT = T'$, where $T' = t'_0 + t'_1 + x^2$ is monic of degree 2. We define the lattice

$$L = \begin{bmatrix} p & 0 & 0 \\ 0 & p & 0 \\ t'_0 & t'_1 & 1 \end{bmatrix}$$

The determinant of L is p^2 hence LLL(L) computes a short vector R of coefficient size bounded by $Cp^{2/3}$, with C the LLL approximation factor (we can take $C \approx 1$ in this practical case). The norm of $\boldsymbol{R}$ will be in the JLSV$_1$ case $\mathrm{Norm}_f(\boldsymbol{R}) \approx \|R\|_\infty^6 \|f\|_\infty^2 = p^5 = q^{5/6}$. This is better than the bound $q^{11/12}$ obtained with the cubic subfield cofactor method. This specific method can be generalized to specific cases of finite fields where reducing as most as possible the degree of the target is the best strategy.

Example 4. We take the same finite field parameters as in Example 2, where $\mathbb{F}_{p^6} = \mathbb{F}_p[x]/(f(x))$. $g = x+3$ is a generator of $\mathbb{F}_{p^6}$. $(1, U, U^2)$ where $U = g^{1+p^3}$ is a basis of $\mathbb{F}_{p^3}$ and $(1, V)$ where $V = g^{1+p^2+p^4}$ is a basis of $\mathbb{F}_{p^2}$. We solve the system $(u_0 + u_1U + U^2)(v_0 + V)T = T'$ where $u_i, v_i \in \mathbb{F}_p$ and T' is monic and represented by a polynomial of degree 2 instead of 5. We did not obtain a solution for T so we do the same again for $g \cdot T$. We obtain

$$\begin{aligned} u &= 300871845097296285096582 39 + 280487161211463875274103 79U + U^2 \\ v &= 31157628467046376019551 36 + V \\ w &= 11308798867395222415537575 \\ T' &= uvw(gT) = X^2 + 17959435401841404261854961X + 69905327539651112256254 65 \end{aligned}$$

We checked that $(T'/(gT))^{\frac{p^6-1}{\ell}} = 1$, meaning that $\log_g T' = 1 + \log_g T$. Then we reduce the lattice defined by the matrix

$$\begin{bmatrix} p & 0 & 0 \\ 0 & p & 0 \\ 69905327539651112256254 65 & 17959435401841404261854961 & 1 \end{bmatrix}$$

to get the short vector $R = -4629536925891637x^2 + 58704739058037245x + 3540200535823573$ whose norm (when lifted in K_f) is

$$\mathcal{N}_{K_f/\mathbb{Q}}(\boldsymbol{R}) = -25315060406729661121215922516982142474458783496360727513629965\backslash 0\backslash 206829219162296152492424600507858835567777143398826374256422 83$$

of 413 bits, which is even smaller than $\log_2 q^{5/6} = 423$ bits. We still have $\log_g \rho(R) \equiv \log_g T + 1 \bmod \ell$.

4.2 Extended Tower Number Field Sieve for $\mathbb{F}_{p^6}$ and $\mathbb{F}_{p^{12}}$

Very recently, Kim proposed in [36] a new variant of the Tower Number Field Sieve that exploits the composite extension degree in another way than Sarkar and Singh. The first version combined the JLSV2 method with the TNFS method, then is was generalized to several previous polynomial selection methods [37,28,51,49,50]. We take the example of $\mathbb{F}_{p^6}$ as a first approach. This example will also help to introduce our idea for small characteristic in Section 5.

The idea of the ExtendedTNFS polynomial selection is to define a tower of two extensions, where the first one is of degree η, such that η divides the finite field extension n, and is smaller than n. Then, two other polynomials of degree at least $\kappa = n/\eta$ are defined, with any former polynomial selection, as if we would like to compute polynomials for $\mathbb{F}_{p^\kappa}$ instead of $\mathbb{F}_{p^n}$. The computation of the norm of an element in the top number field will be smaller by construction, thanks to the tower structure. Since in the initial splitting step, the elements are not of a special form as in the relation collection step, we simply obtain the same norms and bounds with or without ExtendedTNFS. A comparison is provided in Example 4 and Example 6.

Example 5. We take the same parameters as in Example 2: let $p = 31415926535897932384634359$ a 85-bit prime made of the first 26 decimals of π, $p = \lfloor 10^{25}\pi \rfloor + 7926$, so that $\mathbb{F}_{p^6}$ is a 509-bit finite field. Moreover $\Phi_6(p) = p^2 - p + 1$ is the 170-bit prime $\ell = 986960440108935861883947021513080740536833738706523$. We want to compute discrete logarithms in the order-ℓ cyclotomic subgroup of $\mathbb{F}_{p^6}$. The ExtendedTNFS method computes a first polynomial h of degree $\eta = 2$ or $\eta = 3$ to define a first extension. We take $h = y^2 + 1$ since $p = 3 \bmod 4$. Then we define two different polynomials f and g that share a common factor $\varphi(x)$ of degree 3 modulo p. Since $\gcd(2, 3) = 1$, we can take f and g whose coefficients are in $\mathbb{F}_p$ instead of $\mathbb{F}_{p^2}$. We can use any of the previous polynomial selection methods for computing f and g: Conjugation, JLSV1, or GJL for example. We start with the Conjugation method and we will consider the JLSV1 method in the next example (Example 6).

To compute f and g we proceed as if we were running the polynomial selection algorithm for $\mathbb{F}_{p^3}$. We take f in the cyclic family of degree 6: f is defined by $f_t(x) = x^6 - 2tx^5 - (5t+15)x^4 - 20x^3 + 5tx^2 + (2t+6)x + 1$ where t is a small parameter. The number field of f, denoted by $K_f = \mathbb{Q}[x]/(f(x)) = \mathbb{Q}(\alpha_f)$, admits a quadratic subfield defined by the auxiliary polynomial $P(y) = y^2 - 2ty - 3t - 9$. Over this quadratic field $K_2 = \mathbb{Q}[y]/(P(y)) = \mathbb{Q}(\gamma)$ where γ is a root of P, the polynomial f factors as $(x^3 - \gamma x^2 - (\gamma + 3)x - 1)(x^3 - \overline{\gamma}x^2 - (\overline{\gamma} + 3)x - 1)$. We obtain

$$\begin{aligned} t &= 1 \\ f(x) &= x^6 - 2x^5 - 20x^4 - 20x^3 + 5x^2 + 8x + 1 \\ P(y) &= x^2 - 2x - 12 \\ \varphi(x) &= x^3 - 333984620695869407825113x^2 - 33398462069586940782511 6x - 1 \\ g(x) &= 4968577858400x^3 + 1210792524713x^2 - 13694941050487x - 4968577858400 \end{aligned}$$

The two number fields used in ExtendedTNFS will be K_f and K_g, defined as the following.

$$\begin{array}{ccc} K_f = \mathbb{Q}(\alpha_f, \iota) = R[x]/(f(x)) & & K_g = \mathbb{Q}(\alpha_g, \iota) = R[x]/(g(x)) \\ | & & | \\ R = \mathbb{Q}[t]/(h(t)) = \mathbb{Q}[\iota] & \text{and} & R = \mathbb{Q}[t]/(h(t)) = \mathbb{Q}[\iota] \\ | & & | \\ \mathbb{Q} & & \mathbb{Q} \end{array}$$

and the finite field is

$$\begin{array}{ll} \mathbb{F}_{p^6} = \mathbb{F}_{p^2}[x]/(\varphi(x)) & \ni T_0 = (a_{00} + a_{01}t) + (a_{10} + a_{11}t)x + (a_{20} + a_{21}t)x^2 \\ | & \\ \mathbb{F}_{p^2} = \mathbb{F}_p[t]/(h(t)) & \ni a_{00} + a_{01}t \\ | & \\ \mathbb{F}_p & \end{array}$$

A given target $T_0 =$ in $\mathbb{F}_{p^6}$ will be of the form indicated above. As in Example 2 we take our target T_0 in $\mathbb{F}_{p^6}$ whose coefficients are made of the decimals of π (starting at the 26-th decimal, since the first 25 ones were already used for p).

$$\begin{aligned} t_0 &= (327950288419716939937510 5 + 820975591360211292080812 2\ t) \\ &= +(386837435944575764759144 4 + 450939028378094990902013 9\ t)x \\ &= +(162400524326938996131777 38 + 642770498858150816216245 5\ t)x^2 \end{aligned}$$

We proceed as before and try to reduce its degree and/or its coefficient size, in order to reduce the norm of its preimage in K_f or K_g. By construction, a basis for the elements in the subfield $\mathbb{F}_{p^3}$ is $\langle 1, x, x^2 \rangle$ since $\varphi(x)$ is irreducible of degree 3 and of coefficients in $\mathbb{F}_p$. We compute a Reduced Row Echelon Form of the matrix

$$\begin{bmatrix} T \\ xT \bmod \varphi(x) \\ x^2T \bmod \varphi(x) \end{bmatrix} = \begin{bmatrix} a_{00} & a_{01} & a_{10} & a_{11} & a_{20} & a_{21} \\ -a_{20}\varphi_0 & -a_{21}\varphi_0 & a_{00} - a_{20}\varphi_1 & a_{01} - a_{21}\varphi_1 & a_{10} - a_{20}\varphi_2 & a_{11} - a_{21}\varphi_2 \\ \multicolumn{6}{c}{\dots} \end{bmatrix}$$

so that we eliminated two coefficients *in* $\mathbb{F}_p$. We obtain the same form of matrix as in Example 2, the difference being the interpretation of the coefficients for each column. $E =$

$$\begin{bmatrix} 970207087198066030962261 9 & 157568365849050364086443 04 & 309678678246487480883056 44 & 1 & 0 & 0 \\ 520841848900047326120453 4 & 103781230538820669132773 76 & 299208261162983441337911 65 & 128568457259963026174471 30 & 1 & 0 \\ 285184831082867809977545 86 & 124143852976248850983924 9 & 217505362672804484561492 & 154987419210381114782594 26 & 145067889365867165547865 98 & 1 \end{bmatrix}$$

Then we reduce with the LLL algorithm the lattice defined by the lower triangular (12×12)-matrix

$$\begin{bmatrix} p & & & & & & & & & & & \\ 0 & p & & & & & & & & & & \\ 0 & 0 & p & & & & & & & & & \\ E_{1,1} & \dots & E_{1,3} & 1 & & & & & & & & \\ E_{2,1} & \dots & & E_{2,4} & 1 & & & & & & & \\ E_{3,1} & \dots & & & E_{3,5} & 1 & & & & & & \\ \varphi_0 & 0 & \varphi_1 & 0 & \varphi_2 & 0 & 1 & & & & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & & & & \\ & & \varphi_0 & 0 & \varphi_1 & 0 & \varphi_2 & 0 & 1 & & & \\ & & & & & & & & & \ddots & & \\ & & & & & & & & & & \ddots & \\ & & & & & & & & & & & 1 \end{bmatrix}$$

Since φ does not depend on t, $\varphi = (\varphi_0 + 0\ t) + (\varphi_1 + 0\ t)x + (\varphi_2 + 0\ t)x^2 + (1 + 0\ t)x^3$. A shift of one column to the right corresponds to a multiplication by t, a shift of two columns: by x, of three columns: tx, of four columns: x^2, of five columns: tx^2.

After LLL reduction we obtain a (12×12)-matrix whose coefficients are bounded by $Cp^{1/4}$. The norm of the corresponding element in K_f will be bounded by $O(\|r\|_\infty^{12}) = O(p^3) = O(q^{1/2})$, according to the bound formula in [38, §A Lemma 2]:

$$\|N_{K_f/\mathbb{Q}}\left(\sum_{i=0}^{d} a_i(\alpha_h)\alpha_f^i\right) = A^{\deg h \deg f}\|f\|_\infty^{d \deg h}\|h\|_\infty^{(d+\deg f)(\deg h - 1)} D(\deg h, \deg f)\ .$$

In our example $q^{1/2}$ is 254-bit long.

$$\begin{aligned} r &= (-237367 - 407674\ \alpha_h) + (147730 - 788036\ \alpha_h)\ \alpha_f + (797431 + 430785\ \alpha_h)\ \alpha_f^2 \\ &= +(14296 + 1161565\ \alpha_h)\ \alpha_f^3 + (652082 - 519659\ \alpha_h)\ \alpha_f^4 + (1035194 - 67214\ \alpha_h)\ \alpha_f^5 \end{aligned}$$

$$\mathcal{N}_{K_h/\mathbb{Q}}(\mathcal{N}_{K_f/K_h}(r)) = 107163550929037295847348389498692485850896012667565290092782148406412605008460133 7$$

which is 270-bit long.

Example 6. With the same parameters p and extension degree 6, and polynomial $h(t) = t^2 + 1$ for the first extension in the ExtendedTNFS construction, we will define f and g with the JLSV1 method. We will obtain a norm of a given target bounded by $O(q^{5/6})$ (of approximately 424 bits) instead of $O(q)$ (where q is 508-bit long) in the classical JLSV1 construction.

We again take $h(x) = x^2 + 1$ and this time, we run the JLSV1 method as if we were about to compute DL in $\mathbb{F}_{p^3}$. We obtain

$$\begin{aligned} t &= \lceil \sqrt{p} \rceil = 5604991216398 \\ f(x) &= x^3 - 5604991216398x^2 - 56049912164401x - 1 \\ g(x) &= 5604991216398x^3 - 799279460045x^2 - 17614253109239x - 5604991216398 \end{aligned}$$

The finite fields involved in NFS and the finite field representation are the same as in Example 5, with the polynomials above. We take the same target as in the previous Example 5.

$$\begin{aligned} t_0 &= (327950288419716939937510 5 + 8209755913602112920808122\ t) \\ &= +(3868374359445757647591444 + 450939028378094990902013 9\ t)x \\ &= +(1624005243269389961317773 8 + 642770498858150816216245 5\ t)x^2 \end{aligned}$$

We compute a $\mathbb{F}_p$-Reduced Row Echelon Form of the matrix whose rows are made of T, xT, x^2T. The first row of the matrix corresponds to the element

$$(2377518149026844237417539 6 + 1302583989458420310547511 5\ t) + (437238451865494339607366 5 + t)x$$

We divide it by its leading term in $\mathbb{F}_{p^2}$ and we obtain this monic element in x:

$$(a_{00} + a_{01}t) + x = (8765047734261939339069 72 + 1060008532813898381347542 5\ t) + x\ .$$

We reduce the matrix $\begin{bmatrix} p & 0 & 0 \\ 0 & p & 0 \\ a_{00} & a_{01} & 1 \end{bmatrix}$ and obtain

$$\begin{bmatrix} -11286664433880042 & -47936093180625833 & -26104410756684150 \\ 85396065575201237 & -6489687287584757 & -35085391830929374 \\ -76070112599590337 & 107796889266873264 & -141335974330113951 \end{bmatrix}$$

The norm of the element whose coefficients are given by the first row is

$$\begin{aligned} \mathcal{N}_{K_h/\mathbb{Q}}(\mathcal{N}_{K_f/K_h}(r)) = {} & 1307054747806890613682763127695153658399231913764887236702073 69\backslash \\ & 3680492343370559098181272118105652782675693205157765673775302 77 \end{aligned}$$

of 416 bits. We obtained an element of norm 413 bits in Example 4.

4.3 Exploiting all the subgroups of $\mathbb{F}_{p^n}$

Given an element in the cyclotomic subgroup of $\mathbb{F}_{p^n}$, we can generalize the compact representation we obtained in Sections 4.1 and 4.2 to any n. We can represent any element with $\varphi(n)$ non-zero coefficients plus a monic leading term. Since in Algorithm 4 we do not need a one-to-one correspondence between the given elements of the cyclotomic subgroup on one hand, and their representation with only $\varphi(n)$ non-zero non-one coefficients on the other hand, we can use a Gröbner basis computation even if we do not expect a solution at all times. If no such compact representation is found, one test for the next t in $g^t T_0$.

The idea is to list all the distinct subfields $\mathbb{F}_{p^d}$ of $\mathbb{F}_{p^n}$, to compute a polynomial basis for each of them, and to allow a degree of freedom for the coefficients to be $\varphi(d)$ for each subfield $\mathbb{F}_{p^d}$. However the Gröbner basis computation becomes very costly even for $\mathbb{F}_{p^{30}}$, where we need to handle $n - \varphi(n) - 1 = 21$ variables.

5 Faster Initial Splitting in Small Characteristic Finite Fields

In small characteristic, the Function Field Sieve is used. The given target T in $\mathbb{F}_{p^n}$ is lifted in $\mathbb{F}_p[x]$ (usually $\mathbb{F}_2[x]$, $\mathbb{F}_3[x]$, or $\mathbb{F}_r[x]$ with r a small prime power in pairing target groups.) For each randomized target, the preimage as in Algorithm 1 is computed with the Waterloo algorithm that outputs two polynomials $u(x)$, $v(x)$ of degree $\leq \lfloor n/2 \rfloor$ s.t. $T \equiv u(x)/v(x) \bmod I(x)$, where I is a polynomial of degree n that defines the extension field.

In this case, we are only focused on reducing the degree of the target, not the coefficient size. The target coefficients are already very small (in $\mathbb{F}_2$ or $\mathbb{F}_3$ or a small extension $\mathbb{F}_r = \mathbb{F}_{2^4}$ or $\mathbb{F}_q = \mathbb{F}_{3^6}$ for example). We directly use the building block of Algorithm 4 that computes a reduced row echelon form of the matrix. Our idea is extremely simple and provides a very good speed-up. Given a target T in the cyclotomic subgroup of $\mathbb{F}_{p^n}^*$, we take the largest divisor d of n, $1 < d < n$, and we compute T' of degree $n - d$ instead of $n - 1$ and such that $\log T' \equiv \log T \bmod \ell$. When n is even, we simply take $d = n/2$ and compute a new target T' of degree $n/2$ instead of $n - 1$. This is clearly better than the initial splitting step of the Waterloo algorithm: we need to find *one* B_1-smooth polynomial $T'(x)$ of degree $n/2$ instead of *two* polynomials $C(x), D(x)$ of degree $n/2$ that are both B_1-smooth at the same time. This provides a very good speed-up in the asymptotic complexity formula as well as in practice.

5.1 Using implicitly the largest subfield with the QPA

In the recent discrete logarithm records in small characteristic using the QPA, the finite field is a tower of two extensions. The first extension is $\mathbb{F}_{p^l}$ and the second one is $\mathbb{F}_{p^{lk}} = \mathbb{F}_{p^n}$, such that $p^l \approx k$. The target is represented by a polynomial of degree $k - 1$ over $\mathbb{F}_{p^l}$. We need to adapt our algorithm to this representation. Our technique can cancel $d - 1$ coefficients in the base field, that become $\lfloor (d-1)/l \rfloor$ coefficients in $\mathbb{F}_{p^l}$. Let d be the largest proper divisor of $n = kl$. The subfield $\mathbb{F}_{p^d}$ is explicit with the representation $\mathbb{F}_{p^n}$ and implicit with the representation $\mathbb{F}_{p^{lk}}$. We first pre-compute a polynomial basis $[1, U, U^2, \ldots, U^{d-1}]$ of this subfield of $\mathbb{F}_{p^{lk}}$, then again we compute a row-echelon form E of the $d \times k$ matrix A whose rows are made of the coefficients of $U^i T$, for $0 \leq i \leq d - 1$. The key idea again is that each row should be a $\mathbb{F}_p$-linear combination of the other rows, so that each row is made of a $\mathbb{F}_p$-linear combination of the coefficients of a $\mathbb{F}_{p^d}$-multiple of the target T, and the logarithm is unchanged modulo ℓ. To do so, we modify the Gaussian elimination algorithm that computes the row-echelon form of E and allow only $\mathbb{F}_p$-linear combinations of the rows even if the coefficients are in $\mathbb{F}_{p^l}$. It means that we will cancel one coefficient in $\mathbb{F}_{p^l}$ every l rows (we cancel one coefficient in $\mathbb{F}_p$ every row). The generic idea is explained in Algorithm 5 and the technical details are illustrated through the three examples provided in the following sections.

Algorithm 5: Initial splitting in small characteristic with the subfield technique

Input: Finite field $\mathbb{F}_{p^n}$ of small characteristic (e.g. $p = 2, 3$), with the representation $\mathbb{F}_{p^n} = \mathbb{F}_{p^{lk}} = \mathbb{F}_{p^l}[x]/(I(x))$, with $I(x)$ irreducible of degree k over $\mathbb{F}_{p^l}$
generator g (of the order ℓ subgroup of the cyclotomic subgroup)
target $T_0 \in \mathbb{F}_{p^{lk}}$
smoothness bound B_1
Output: t, $T = g^t T_0$, s.t. T is represented by a polynomial $T(x)$ which is B_1-smooth (w.r.t. its degree)

$d \leftarrow$ the largest divisor of n, $1 < d < n$
Compute $U(x) \in \mathbb{F}_{p^{lk}}$ s.t. $(1, U, U^2, \ldots, U^{d-1})$ is a polynomial basis of the subfield $\mathbb{F}_{p^d}$
repeat
  take $t \in \{1, \ldots, \ell - 1\}$ at random
  $T \leftarrow g^t T_0$ in $\mathbb{F}_{p^{lk}}$
  Define $A = \begin{bmatrix} T \\ UT \\ \vdots \\ U^{d-1}T \end{bmatrix}$ a $d \times k$ matrix whose coefficients are in $\mathbb{F}_{p^l}$
  $E \leftarrow$ ReducedRowEchelonForm(A)
  $T'(x) \leftarrow$ the polynomial of lowest degree made of the first row of E
until $T'(x)$ *is* B_1*-smooth*
return $t, T'(x)$

Remark 1. We can increase the number of tested elements at each round for a given t by a factor n for almost free. We run again a Gaussian elimination algorithm on the matrix E but in the reverse side, for instance from row 1 to row n if it was done from row n to row 1 the first time. The matrix is in

row-echelon form on the left-hand side and on the right-hand side (the upper right and lower left corners are filled with zeros).

$$E = \begin{bmatrix} * & \dots & * & 1 & 0 & \dots & 0 \\ 0 & \ddots & & & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & & & \ddots & 0 \\ 0 & \dots & 0 & * & \dots & * & 1 \end{bmatrix}$$

The i-th row represents an element $T_i = X^{i-1}T'_i$, where T'_i is of reduced degree $n-d$ (or $(n-d)/l$ over $\mathbb{F}_{p^l}$, according to the representation). Since X is in the factor basis (by construction, as all the degree 1 polynomials), its logarithm is known at this point (after the relation collection and linear algebra steps), hence the logarithm of any X^i is known. It remains to compute the discrete logarithm of T'_i of degree $n-d$.

In practice there are some technicalities: in the second Gaussian Elimination, if the leading coefficient is zero then two rows are swapped, and it cancels the previous Gaussian Elimination (computed at the other side of the matrix) for that row. We end up with a matrix which is in row-echelon form on the right and almost row-echelon form on the left (or vice-versa). Since each sequence of l rows produces a sequence of l polynomials, all of same degree, and since at the end we divide by the leading term in $\mathbb{F}_{p^l}$, swapping two rows whose difference of indices is less than l will not change the degree in X of the corresponding polynomial. In average, some rare polynomials will have a degree increased by one or two. This second Gaussian elimination increases the number of tests by a factor n (or k for $\mathbb{F}_{p^{kl}}$) at a very cheap cost.

5.2 Complexity analysis.

Cost of one iteration of the initial splitting. At first glance, we replace an extended Euclidean algorithm costing $O(n^2)$ by an echelon form (or Gaussian elimination) computation. The complexity of a reduced echelon form computation is $O(n^3)$ [21, §13.4.2]. This would be more expensive (though still polynomial in n). But if we compute a double-side row-echelon form as in Remark 1, then we get n distinct polynomials of the form X^iT' where T' is of degree $n-d$, and we can test each of them for smoothness. In other words, we perform two Gaussian eliminations on reverse side for n smoothness tests, meaning that the $O(n^3)$ cost is shared over n smoothness tests, making a $O(n^2)$ per smoothness test in average. The complexity is then the same as the Waterloo algorithm. We also replace two B-smooth tests by only one and that might save some time in practice. We present the theoretical costs in Table 3 from [22]. XGCD stands for extended Euclidean algorithm, SQF stands for SQuare-free Factorization, DDF stands for Distinct Degree Factorization and EDF stands for Equal Degree Factorization.

algorithm	XGCD(T, I)	Echelon Form	SQF	DDF	EDF
cost	$O(n^2)$	$O(n^3)$	$O(n^2)$	$O(n^3 \log p^l)$	$O(n^2 \log p^l)$
Waterloo	1 time	–	2 times	1 time/SQF factor	1 time/DDF factor
This work	–	1 time/n tests	1 time	1 time/SQF factor	1 time/DDF factor

Table 3: Cost of one iteration of the initial splitting step

Total cost of the initial splitting step. If we assume that $b = \log L_{p^n}[2/3, \gamma]$, then the probability that two given polynomials of degree $n/2$ are b-smooth at the same time (Waterloo algorithm) is $L_{p^n}[1/3, -1/(3\gamma)]$. The probability of one polynomial of degree $a \cdot n$, for a fixed number $0 < a \leq n$, to be b-smooth is $L_{p^n}[1/3, -a/(3\gamma)]$. Whenever $a < 1$ (i.e. n is not prime and $a = \frac{n-d}{n}$ where $d \mid n$), the asymptotic complexity of our algorithm is better. But this complexity does not take into account the advantage of the initial splitting of the Waterloo algorithm: the gain disappears in the $o(1)$ (this fact was analyzed in [5, §4.1.2]). If n is even then $a = 1/2$, there is one polynomial of degree $n/2$ to test for smoothness instead of two polynomials of degree $n/2$ and obviously the saving is very important. But what if $a = 4/5$? Is there a crossing-point of the running-times of the Waterloo and this algorithm? The asymptotic complexity formula does not allow to see it. There is still a cross-over point in practice that

we can compute with the exact formulas. Example 6.3 deals with $\mathrm{GF}(3^{5\cdot 479})$ where $a = 4/5$ and illustrates well this cross-over.

6 Practical examples in small characteristic

6.1 Adj–Menezes–Oliveira–Rodríguez-Henríquez record in $\mathrm{GF}(3^{6\cdot 509})$

Adj, Menezes, Oliveira and Rodríguez-Henríquez estimated in [2] the cost to compute discrete logarithms in the 4841-bit finite field $\mathrm{GF}(3^{6\cdot 509})$. The elements are represented by polynomials of degree at most 508 whose coefficients are in $\mathbb{F}_{3^6}$. The initial splitting of the Waterloo algorithm outputs two polynomials of degree 254. The probability that two independent and relatively prime polynomials of degree 254 over $\mathbb{F}_{3^6}$ are simultaneously b-smooth is $(1-1/3^6)\Pr^2_{3^6}(254, b)$ [20], where $\Pr_q(N, b)$ is the probability that a polynomial of degree N over $\mathbb{F}_q$ is b-smooth. The term $(1-1/3^6)$ is negligible in practice for the values we are looking at.

Our algorithm 5 outputs *one* polynomial of degree 254. The probability that it is b-smooth is $\Pr_{3^6}(n, b)$, i.e. the square root of the previous one. For the same probability, we can take a much smaller b. We list in Tab. 4a in Appendix A the values of b to obtain a probability between 2^{-40} and 2^{-20}. For instance, if we allow 2^{30} trials, then we can set $b = 28$ with our algorithm, instead of $b = 43$ previously: we have $\Pr^2_{3^6}(254, 43) = 2^{-30.1}$ and we only need to take $b = 28$ to get the same probability with this work: $\Pr_{3^6}(254, 28) = 2^{-29.6}$. This will provide a good practical speed-up of the descent phase: much less elements need to be "reduced": this reduces the initial width of the tree, and they are of much smaller degree: this reduces the the depth of the descent tree.

The finite field is represented as a first extension $\mathbb{F}_{p^l} = \mathbb{F}_3[y]/(y^6 + 2y^4 + y^2 + 2y + 2)$, then a second extension $\mathbb{F}_{3^{6\cdot 509}} = \mathbb{F}_{3^6}[X]/(I(X))$, where $I(X)$ is the degree 509 irreducible factor of $h_1X^q - h_0$, where $h_1 = X^2 + y^{424}X$ and $h_0 = y^{316}X + y^{135}$. As a proof of concept, we computed a 30-smooth initial splitting of the target $T_0 = \sum_{i=0}^{508}(y^{\lfloor \pi(3^6)^{i+1} \rfloor} \bmod 3^6)X^i$. We found that $G^{47233}T_0 = uvX^{230}T'$ where $u = 1 \in \mathbb{F}_{3^6}$, $v \in \mathbb{F}_{3^{3\cdot 509}}$ and T' is of degree 255 and 30-smooth. The equality $(G^{47233}T_0)^{\frac{p^k-1}{r}} = (uvX^{230}T')^{\frac{p^k-1}{r}}$ is satisfied. The explicit value of T' is given in Appendix B. The whole computation took less than 6 days on 48 cores Intel Xeon E5-2609 2.40GHz (274 core-days). This is obviously overshot, but this was done with a non-optimized Magma implementation.

6.2 Computing discrete logarithms in $\mathbb{F}_{2^{512}}$ and $\mathbb{F}_{2^{1024}}$

In [24,25, §3.6] discrete logarithms in $\mathbb{F}_{2^{512}}$ and $\mathbb{F}_{2^{1024}}$ need to be computed modulo the full multiplicative group order $2^n - 1$. As pointed to us by R. Granger, our technique can be used to compute discrete logarithms in $\mathbb{F}_{2^{1024}}$. Our algorithm provides a decomposition of the target as the product uS where u is an element in the largest proper subfield $\mathbb{F}_{2^{512}}$, and S is an element of $\mathbb{F}_{2^{1024}}$ of degree 512 intead of 1023. The discrete logarithm of the subfield cofactor u can be obtained by a discrete logarithm computation in $\mathbb{F}_{2^{512}}$. More generally, our technique is useful when discrete logarithms in nested finite fields such as $\mathbb{F}_{2^{2^i}}$ are computed recursively.

6.3 Joux–Pierrot record in $\mathrm{GF}(3^{5\cdot 479})$

Joux and Pierrot announced a discrete logarithm record computation in $\mathrm{GF}(3^{5\cdot 479})$ in [32] (then published in [33]). They defined a first degree 5 extension $\mathrm{GF}(3^5) = \mathrm{GF}(3)[y]/(y^5 - y + 1)$ then a degree 479 extension on top of it. The irreducible degree 479 polynomial $I(X)$ was a divisor of $Xh_1(X^q) - h_0(X^q)$, where $h_0 = X^2 + y^{111}X$ and $h_1 = yX + 1$. Given a target $T \in \mathbb{F}_{3^{5\cdot 479}}$, the initial splitting outputs two polynomials $u(X), v(X) \in \mathbb{F}_{3^5}[X]$ of degree $\lfloor 478/2 \rfloor = 239$. Our method outputs one polynomial of degree $\lfloor \frac{4}{5}479 \rfloor = 383$. This example is interesting because the smoothness probabilities are very close. We computed the exact values with Drmota–Panario's formulas, and give them in table 4b of Appendix A. We obtain $\Pr^2_{3^5}(239, 50) = 2^{-20.96}$ (Waterloo) and $\Pr_{3^5}(383, 50) = 2^{-22.96}$ (Algorithm 5), i.e. our algorithm would be four times slower; $\Pr^2_{3^5}(239, 40) = 2^{-30.61}$ and $\Pr_{3^5}(383, 40) = 2^{-32.34}$; $\Pr^2_{3^5}(239, 30) = 2^{-48.46}$ and $\Pr_{3^5}(383, 30) = 2^{-49.34}$; and the cross-over point is for $b = 24$: in this case, we have $\Pr^2_{3^5}(239, 24) = 2^{-67.96}$ and $\Pr_{3^5}(383, 24) = 2^{-67.59}$ which is slightly larger.

The probabilities would advise to use the classical Initial Splitting with the extended GCD algorithm. We remark that this algorithm would output two b-smooth polynomials of degree $(n-1)/2$. Each would factor into approximately $\lceil n/b \rceil$ irreducible polynomials of degree at most b. Each such factor is sent as an input to the second step (descent step). If we use Algorithm 5, the initial splitting will outputs one polynomial of degree 383 that factors into approximately $\lceil 383/b \rceil$ polynomials of degree at most b, each of them sent as input to the second step, that is, the descent step is called 20% time less, and that would reduce the total width of the descent tree accordingly. Since the descent is the most costly part, and in particular, the memory size required is huge, this remark would need to be taken into consideration for a practical implementation.

As a proof of concept of our algorithm, we implemented in Magma our algorithm, took the same parameters, generator and target as in [32] and found a 50-smooth decomposition for the target given by the 471-th row of the matrix computed for $g^{23940}T_0$ in 1239 core-hours (22.12 hours over 56 cores) on an Intel Xeon E5-2609 2.40GHz (compared to 5000 core-hours announced in [32]). The value is given in appendix C. In our technique, we compute $g^t T_0 = uvT'$ where $u \in \mathbb{F}_{3^5}$ (this is the leading term of the polynomial), $v \in \mathbb{F}_{3^{479}}$ and T' which is 50-smooth. The discrete logarithm of u can be tabulated however it remain quite hard to compute the discrete logarithm of v. Our technique is useful if it is easy (or not required) to compute discrete logarithms in the subfields.

Acknowledgments. Many thanks to Francisco Rodríguez-Henríquez, Frederik Vercauteren, Robert Granger and Thorsten Kleinjung, François Morain, Pierrick Gaudry, and Luca De Feo. All these very fruitful discussions started at the ECC 2015 conference, the CATREL workshop and the Asiacrypt 2015 conference.

References

1. G. Adj, A. Menezes, T. Oliveira, and F. Rodríguez-Henríquez. Computing discrete logarithms in $\mathbb{F}_{3^{6\cdot 137}}$ and $\mathbb{F}_{3^{6\cdot 163}}$ using Magma. In Ç. K. Koç, S. Mesnager, and E. Savas, editors, *Arithmetic of Finite Fields (WAIFI 2014)*, volume 9061 of *LNCS*, pages 3–22. Springer, Heidelberg, 2014.
2. G. Adj, A. Menezes, T. Oliveira, and F. Rodríguez-Henríquez. Weakness of $\mathbb{F}_{3^{6\cdot 509}}$ for discrete logarithm cryptography. In Z. Cao and F. Zhang, editors, *PAIRING 2013*, volume 8365 of *LNCS*, pages 20–44. Springer, Heidelberg, Nov. 2014.
3. L. Adleman. The function field sieve. In L. M. Adleman and M.-D. Huang, editors, *Algorithmic Number Theory (ANTS-I)*, volume 877 of *LNCS*, pages 141–154. Springer, Heidelberg, 1994.
4. L. M. Adleman and M.-D. A. Huang. Function field sieve method for discrete logarithms over finite fields. *Information and Computation*, 151(1/2):5–16, 1999.
5. R. Barbulescu. *Algorithmes de logarithmes discrets dans les corps finis*. PhD thesis, Université de Lorraine, 2013.
6. R. Barbulescu. An appendix for a recent paper of Kim. Cryptology ePrint Archive, Report 2015/1076, 2015. `http://eprint.iacr.org/2015/1076`.
7. R. Barbulescu, C. Bouvier, J. Detrey, P. Gaudry, H. Jeljeli, E. Thomé, M. Videau, and P. Zimmermann. Discrete logarithm in GF(2^{809}) with ffs, April 2013. NMBRTHRY archives, item 004534.
8. R. Barbulescu, C. Bouvier, J. Detrey, P. Gaudry, H. Jeljeli, E. Thomé, M. Videau, and P. Zimmermann. Discrete logarithm in GF(2^{809}) with FFS. In H. Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 221–238. Springer, Heidelberg, Mar. 2014.
9. R. Barbulescu, P. Gaudry, A. Guillevic, and F. Morain. Improving NFS for the discrete logarithm problem in non-prime finite fields. In E. Oswald and M. Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 129–155. Springer, Heidelberg, Apr. 2015.
10. R. Barbulescu, P. Gaudry, A. Joux, and E. Thomé. A heuristic quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic. In P. Q. Nguyen and E. Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 1–16. Springer, Heidelberg, May 2014.
11. R. Barbulescu, P. Gaudry, and T. Kleinjung. The tower number field sieve. In T. Iwata and J. H. Cheon, editors, *ASIACRYPT 2015, Part II*, volume 9453 of *LNCS*, pages 31–55. Springer, Heidelberg, Nov. / Dec. 2015.
12. P. S. L. M. Barreto, H. Y. Kim, B. Lynn, and M. Scott. Efficient algorithms for pairing-based cryptosystems. In M. Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 354–368. Springer, Heidelberg, Aug. 2002.
13. I. F. Blake, R. Fuji-Hara, R. C. Mullin, and S. A. Vanstone. Computing logarithms in finite fields of characteristic two. *SIAM Journal on Algebraic Discrete Methods*, 5(2):276–285, 1984.
14. I. F. Blake, R. C. Mullin, and S. A. Vanstone. Computing logarithms in GF(2^n). In G. R. Blakley and D. Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 73–82. Springer, Heidelberg, Aug. 1984.

15. C. Bouvier, P. Gaudry, L. Imbert, H. Jeljeli, and E. Thomé. Discrete logarithms in GF(p) — 180 digits. NMBRTHRY archives, item 004703, June 2014. https://listserv.nodak.edu/cgi-bin/wa.exe?A2=NMBRTHRY;615d922a.1406.
16. A. Commeine and I. Semaev. An algorithm to solve the discrete logarithm problem with the number field sieve. In M. Yung, Y. Dodis, A. Kiayias, and T. Malkin, editors, *PKC 2006*, volume 3958 of *LNCS*, pages 174–190. Springer, Heidelberg, Apr. 2006.
17. D. Coppersmith. Fast evaluation of logarithms in fields of characteristic two. *IEEE Transactions on Information Theory*, 30(4):587–594, 1984.
18. D. Coppersmith, A. M. Odlyzko, and R. Schroeppel. Discrete logarithms in GF(p). *Algorithmica*, 1(1):1–15, 1986.
19. W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.
20. M. Drmota and D. Panario. A rigorous proof of the Waterloo algorithm for the discrete logarithm problem. *Designs, Codes and Cryptography*, 26(1):229–241, June 2002.
21. J.-G. Dumas and C. Pernet. *Handbook of finite fields*, chapter Computational linear algebra over finite fields, pages 520–535. CRC Press Taylor & Francis Group, 2013.
22. P. Flajolet, X. Gourdon, and D. Panario. The complete analysis of a polynomial factorization algorithm over finite fields. *Journal of Algorithms*, 40:37–81, 2001.
23. D. M. Gordon. Discrete logarithms in GF(p) using the number field sieve. *SIAM Journal on Discrete Mathematics*, 6(1):124–138, 1993.
24. R. Granger, P. Jovanovic, B. Mennink, and S. Neves. Improved masking for tweakable blockciphers with applications to authenticated encryption. Cryptology ePrint Archive, Report 2015/999, 2015. http://eprint.iacr.org/2015/999.
25. R. Granger, P. Jovanovic, B. Mennink, and S. Neves. Improved masking for tweakable blockciphers with applications to authenticated encryption. In M. Fischlin and J. Coron, editors, *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I*, volume 9665 of *Lecture Notes in Computer Science*, pages 263–293. Springer, 2016.
26. R. Granger, T. Kleinjung, and J. Zumbrägel. Breaking '128-bit secure' supersingular binary curves - (or how to solve discrete logarithms in $\mathbb{F}_{2^{4\cdot 1223}}$ and $\mathbb{F}_{2^{12\cdot 367}}$). In J. A. Garay and R. Gennaro, editors, *CRYPTO 2014, Part II*, volume 8617 of *LNCS*, pages 126–145. Springer, Heidelberg, Aug. 2014.
27. A. Guillevic. Computing individual discrete logarithms faster in GF(p^n) with the NFS-DL algorithm. In T. Iwata and J. H. Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 149–173. Springer, Heidelberg, Nov. / Dec. 2015.
28. J. Jeong and T. Kim. Extended tower number field sieve with application to finite fields of arbitrary composite extension degree. Cryptology ePrint Archive, Report 2016/526, 2016. http://eprint.iacr.org/.
29. A. Joux and R. Lercier. The function field sieve is quite special. In C. Fieker and D. R. Kohel, editors, *Algorithmic Number Theory (ANTS-V)*, volume 2369 of *LNCS*, pages 431–445. Springer, Heidelberg, 2002.
30. A. Joux and R. Lercier. Improvements to the general number field sieve for discrete logarithms in prime fields. A comparison with the Gaussian integer method. *Math. Comp.*, 72(242):953–967, 2003.
31. A. Joux, R. Lercier, N. Smart, and F. Vercauteren. The number field sieve in the medium prime case. In C. Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 326–344. Springer, Heidelberg, Aug. 2006.
32. A. Joux and C. Pierrot. Discrete logarithm record in characteristic 3, GF($3^{5\cdot 479}$) a 3796-bit field. Number Theory list, item 004745, Sep 2014. https://listserv.nodak.edu/cgi-bin/wa.exe?A2=NMBRTHRY;1ff78abb.1409.
33. A. Joux and C. Pierrot. Improving the polynomial time precomputation of frobenius representation discrete logarithm algorithms - simplified setting for small characteristic finite fields. In P. Sarkar and T. Iwata, editors, *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 378–397. Springer, Heidelberg, Dec. 2014.
34. A. Joux and C. Pierrot. The special number field sieve in $\mathbb{F}_{p^n}$ - application to pairing-friendly constructions. In Z. Cao and F. Zhang, editors, *PAIRING 2013*, volume 8365 of *LNCS*, pages 45–61. Springer, Heidelberg, Nov. 2014.
35. M. Kalkbrener. An upper bound on the number of monomials in determinants of sparse matrices with symbolic entries. *Mathematica Pannonica*, 8:73–82, 1997.
36. T. Kim. Extended tower number field sieve: A new complexity for medium prime case. Cryptology ePrint Archive, Report 2015/1027, 2015. http://eprint.iacr.org/2015/1027.
37. T. Kim and R. Barbulescu. Extended Tower Number Field Sieve: A New Complexity for Medium Prime Case. In M. Robshaw and J. Katz, editors, *CRYPTO 2016*, LNCS, Santa Barbara, August 2016. Springer. to appear, preprint available at https://hal.archives-ouvertes.fr/hal-01281966.
38. T. Kim and R. Barbulescu. Extended Tower Number Field Sieve: A New Complexity for Medium Prime Case. preprint, Mar. 2016.
39. T. Kleinjung. Discrete logarithms in GF(2^{1279}). Number Theory list, item 004751, October 2014. https://listserv.nodak.edu/cgi-bin/wa.exe?A2=NMBRTHRY;256db68e.1410.

40. T. Kleinjung. Discrete logarithms in GF(p) — 768 bits, January 2016. NMBRTHRY archives, item 004917.
41. B. A. LaMacchia and A. M. Odlyzko. Computation of discrete logarithms in prime fields. *Des. Codes Cryptography*, 1(1):47–62, 1991.
42. A. Lenstra, J. Lenstra, H.W., and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982.
43. D. Matyukhin. Effective version of the number field sieve for discrete logarithms in the field GF(p^k) (in Russian). *Trudy po Discretnoi Matematike*, 9:121–151, 2006. `http://m.mathnet.ru/php/archive.phtml?wshow=paper&jrnid=tdm&paperid=144&option_lang=eng`.
44. M. A. Morrison and J. Brillhart. A method of factoring and the factorization of F_7. *Math. Comp.*, 29(129):183–205, 1975. `http://www.ams.org/journals/mcom/1975-29-129/S0025-5718-1975-0371800-5/S0025-5718-1975-0371800-5.pdf`.
45. A. M. Odlyzko. Discrete logarithms in finite fields and their cryptographic significance. In T. Beth, N. Cot, and I. Ingemarsson, editors, *EUROCRYPT'84*, volume 209 of *LNCS*, pages 224–314. Springer, Heidelberg, Apr. 1985.
46. C. Pomerance. Analysis and comparison of some integer factoring algorithms. In H. W. J. Lenstra and R. Tijdeman, editors, *Computational methods in number theory, part I*, volume 154 of *Mathematical Centre Tracts*, pages 89–139. Mathematisch Centrum, Amsterdam, 1982. available in pdf at `http://oai.cwi.nl/oai/asset/19571/19571A.pdf`.
47. J. Z. Robert Granger, Thorsten Kleinjung. Discrete logarithms in the jacobian of a genus 2 supersingular curve over GF(2^{367}). Number Theory list, item 004665, January 2014. (DL in GF(2^{4404})), `https://listserv.nodak.edu/cgi-bin/wa.exe?A2=NMBRTHRY;23651c2.1401`.
48. P. Sarkar and S. Singh. A simple method for obtaining relations among factor basis elements for special hyperelliptic curves. Cryptology ePrint Archive, Report 2015/179, 2015. `http://eprint.iacr.org/2015/179`.
49. P. Sarkar and S. Singh. A general polynomial selection method and new asymptotic complexities for the tower number field sieve algorithm. Cryptology ePrint Archive, Report 2016/485, 2016. `http://eprint.iacr.org/`.
50. P. Sarkar and S. Singh. A generalisation of the conjugation method for polynomial selection for the extended tower number field sieve algorithm. Cryptology ePrint Archive, Report 2016/537, 2016. `http://eprint.iacr.org/`.
51. P. Sarkar and S. Singh. Tower number field sieve variant of a recent polynomial selection method. Cryptology ePrint Archive, Report 2016/401, 2016. `http://eprint.iacr.org/`.

A Smooth polynomials over a finite field

Odlyzko computed the asymptotic complexity of the Waterloo algorithm to be $L_{2^n}[1/2, \sqrt{2\log 2}]$ in [45], using the following heuristic: the probability that two independent polynomials of degree m are b-smooth is the same as the square of the probability that a polynomial of degree m is b-smooth. Drmota and Panario gave a non-heuristic proof in [20], and proved Theorem 2: the heuristic is true up to a factor $(1 - 1/q)$.

Definition 1. *Let $N_q(b; n)$ denote the number of monic polynomials over $\mathbb{F}_q$ of degree n which are b-smooth, i.e., all irreducible factors have degree smaller or equal to b. let $N_q(b; n_1, n_2)$ denote the number of coprime pairs of monic polynomials over $\mathbb{F}_q$ of degrees n_1 and n_2, respectively, which are b-smooth.*

Theorem 2 ([20, Theorem 1]). *Let $\delta > 0$ be given. Then we have, uniformly for $b, n_1, n_2 \to \infty$ with $n_1^\delta \le b \le n_1^{1-\delta}$ and $n_2^\delta \le b \le n_2^{1-\delta}$,*

$$N_q(b; n_1, n_2) \sim \left(1 - \frac{1}{q}\right) N_q(b; n_1) N_q(b; n_2) \ .$$

Corollary 1 ([20, Theorem 1]). *Let $\delta > 0$ be given. Then we have, uniformly for $b, n_1, n_2 \to \infty$ with $n_1^\delta \le b \le n_1^{1-\delta}$ and $n_2^\delta \le b \le n_2^{1-\delta}$,*

$$\Pr_q(b; n_1, n_2) \sim \left(1 - \frac{1}{q}\right) \Pr_q(b; n_1) \Pr_q(b; n_2) \ .$$

The asymptotic complexity computation needs the estimation of the probability that a polynomial over $\mathbb{F}_q$ of degree m factors entirely into polynomials of degree at most b, i.e. is b-smooth. Odlyzko gave the following estimation in [45, eq. (4.5) p. 14].

$$\Pr_q(b, n)^{-1} = \exp\left((1 + o(1))\frac{n}{b}\log_e \frac{n}{b}\right) \text{ for } n^{1/100} \le b \le n^{99/100} \tag{6}$$

Then in combination with Odlyzko's formulas we computed the table of the probabilities for GF($3^{6\cdot 509}$) and GF($3^{5\cdot 479}$).

b	$\mathrm{Pr}^2_{3^6}(254, b)$	b	$\mathrm{Pr}_{3^6}(254, b)$
36	$2^{-40.1}$	22	$2^{-42.3}$
37	$2^{-38.4}$	23	$2^{-39.6}$
38	$2^{-36.8}$	24	$2^{-37.2}$
39	$2^{-35.3}$	25	$2^{-35.1}$
40	$2^{-33.9}$		
41	$2^{-32.5}$	26	$2^{-33.1}$
42	$2^{-31.3}$	27	$2^{-31.3}$
43	$2^{-30.1}$	28	$2^{-29.6}$
44	$2^{-28.9}$		
45	$2^{-27.9}$	29	$2^{-28.1}$
46	$2^{-26.9}$		
47	$2^{-25.9}$	30	$2^{-26.6}$
48	$2^{-25.0}$	31	$2^{-25.3}$
49	$2^{-24.1}$	32	$2^{-24.1}$
50	$2^{-23.3}$	33	$2^{-23.0}$
51	$2^{-22.5}$		
52	$2^{-21.8}$	34	$2^{-21.9}$
53	$2^{-21.1}$	35	$2^{-21.0}$
54	$2^{-20.4}$	36	$2^{-20.1}$
55	$2^{-19.7}$		
56	$2^{-19.1}$	37	$2^{-19.2}$
57	$2^{-18.5}$		
58	$2^{-18.0}$	38	$2^{-18.4}$
59	$2^{-17.4}$	39	$2^{-17.6}$
60	$2^{-16.9}$	40	$2^{-16.9}$
61	$2^{-16.4}$	41	$2^{-16.3}$
62	$2^{-15.9}$	42	$2^{-15.6}$

(a) Probabilities for $\mathrm{GF}(3^{6 \cdot 509})$

b	$\mathrm{Pr}^2_{3^5}(239, b)$	$\mathrm{Pr}_{3^5}(383, b)$
24	$2^{-67.96}$	$2^{-67.59}$
25	$2^{-63.95}$	$2^{-63.86}$
26	$2^{-60.30}$	$2^{-60.45}$
27	$2^{-56.95}$	$2^{-57.32}$
28	$2^{-53.89}$	$2^{-54.44}$
29	$2^{-51.07}$	$2^{-51.79}$
30	$2^{-48.46}$	$2^{-49.34}$
31	$2^{-46.06}$	$2^{-47.06}$
32	$2^{-43.83}$	$2^{-44.95}$
33	$2^{-41.76}$	$2^{-42.99}$
34	$2^{-39.83}$	$2^{-41.16}$
35	$2^{-38.03}$	$2^{-39.44}$
36	$2^{-36.35}$	$2^{-37.84}$
37	$2^{-34.77}$	$2^{-36.34}$
38	$2^{-33.30}$	$2^{-34.92}$
39	$2^{-31.91}$	$2^{-33.60}$
40	$2^{-30.61}$	$2^{-32.34}$
41	$2^{-29.39}$	$2^{-31.16}$
42	$2^{-28.23}$	$2^{-30.05}$
43	$2^{-27.14}$	$2^{-28.99}$
44	$2^{-26.11}$	$2^{-27.99}$
45	$2^{-25.13}$	$2^{-27.04}$
46	$2^{-24.21}$	$2^{-26.14}$
47	$2^{-23.33}$	$2^{-25.29}$
48	$2^{-22.50}$	$2^{-24.47}$
49	$2^{-21.71}$	$2^{-23.70}$
50	$2^{-20.96}$	$2^{-22.96}$

(b) For $\mathrm{GF}(3^{5 \cdot 479})$

Table 4: Smoothness probabilities of polynomials over finite fields.

B Initial splitting in GF($3^{6\cdot509}$)

The generator is $G = X+y^2$. We computed a top-down left followed by bottom-up right $\mathbb{F}_3$-linear Gaussian elimination on the matrix whose rows correspond to U^iT, where $\{1, U, \ldots, U^{3\cdot509-1}\}$ is a polynomial basis of $\mathbb{F}_{3^{3\cdot509}}$ which is the largest proper subfield of $\mathbb{F}_{3^{6\cdot509}}$. About 2/3 of the 1529 rows of the matrix gave distinct polynomials of degree 254 or 255. Since the number of rows of our matrix is $3\cdot509 \equiv 3 \bmod 6$, then half the polynomials have degree 254 and half degree 255 in average. We want to compute a 32-smooth decomposition. Since each g^tT_0 produces about 2^{10} polynomials to test and $\Pr_{3^6}(254, 32) = 2^{-24.1}$, we need to test about $2^{14.1}$ different t to get one 32-smooth decomposition. We run our algorithm over 48 cores, each of them running 2^{10} t, so that we had $t \in \{0, \ldots, 48\cdot 2^{10} - 1 = 49151 \approx 2^{15.6}\}$, making $2^{25.6}$ candidates. We found that the 1384-th row of the two-side row-echelon form computed for $T = G^{47233}T_0$ gives $T = uvX^{230}T'$, where T' is 30-smooth, $u \in \mathbb{F}_{3^6}$ and $v \in \mathbb{F}_{3^{3\cdot509}}$. The whole computation took less than 6 days on 48 cores Intel Xeon E5-2609 2.40GHz (274 core-days). This is obviously overshot, but this was done with a non-optimized Magma implementation.

b	$\Pr_{3^6}(254, b)\cdot 2^{25.6}$	#	t	row	$\deg T'$	X^i	leading coeff $\in \mathbb{F}_{3^6}$
30	$2^{-1} = 0.5$	1	47233	1384	255	X^{230}	1
31	$2^{0.3} = 1.2$	1	24628	265	254	X^{44}	u^{405}
32	$2^{1.5} = 2.8$	1	44408	1036	255	X^{172}	1
33	$2^{2.6} = 6.0$	1	22847	7	255	X^0	u^{443}
34	$2^{3.7} = 13.0$	9					
35	$2^{4.6} = 24.3$	10					
36	$2^{5.5} = 45.3$	20					
37	$2^{6.4} = 84.4$	37					
38	$2^{7.2} = 147$	50					
39	$2^{8.0} = 256$	110					
40	$2^{8.7} = 416$	170					

Table 5: Results: b-smooth polynomials in $\mathbb{F}_{3^{6\cdot509}}$ found for $t \in \{0, \ldots, 49151 \approx 2^{15.6}\}$

```
// found a 30-smooth decomposition: T = g^47233*T0 = lc*X^230*poly, row = 1384, leading coeff=1:
poly:=X^255+u^295*X^254+u^477*X^253+u^214*X^252+u^557*X^251+u^50*X^250+u^273*X^249+u^51*X^248+u^725*X^247+u^619*X^246
+u^374*X^245+u^66*X^244+u^591*X^243+u^410*X^242+u^132*X^241+u^619*X^240+u^546*X^239+u^484*X^238+u^618*X^237+u^638*X^236
+u^399*X^235+u^598*X^234+u^701*X^233+u^647*X^232+u^663*X^231+u^78*X^230+u^43*X^229+u^520*X^228+u^151*X^227+u^726*X^226
+u^238*X^225+u^321*X^224+u^415*X^223+u^170*X^222+u^63*X^221+u^615*X^220+u^634*X^219+u^10*X^218+u^126*X^217+u^160*X^216
+u^579*X^215+u^425*X^214+u^93*X^213+u^436*X^212+u^191*X^211+u^573*X^210+u^725*X^209+u^336*X^208+u^19*X^207+u^580*X^206
+u^563*X^205+u^45*X^204+u^330*X^203+u^349*X^202+u^50*X^201+u^166*X^200+u^260*X^199+u^670*X^198+u^44*X^197+u^477*X^196
+u^476*X^195+u^492*X^194+u^71*X^193+X^192+u^277*X^191+u^60*X^190+u^103*X^189+u^403*X^188+u^112*X^187+u^213*X^186
+u^314*X^185+u^404*X^184+u^193*X^183+u^189*X^182+u^727*X^181+u^610*X^180+u^178*X^179+u^432*X^178+u^495*X^177+u^285*X^176
+u^577*X^175+u^444*X^174+u^715*X^173+u^416*X^172+u^644*X^171+u^629*X^170+u^468*X^169+u^293*X^168+u^724*X^167+u^502*X^166
+u^697*X^165+u^272*X^164+u^10*X^163+u^575*X^162+u^411*X^161+u^49*X^160+u^182*X^159+u^285*X^158+u^143*X^157+u^380*X^156
+u^594*X^155+u^349*X^154+u^141*X^153+u^683*X^152+u^102*X^151+u^523*X^150+u^410*X^149+u^300*X^148+u^450*X^147+u^16*X^146
+u^423*X^145+u^159*X^144+u^722*X^143+u^379*X^142+u^465*X^141+u^621*X^140+u^104*X^139+u^293*X^138+u^137*X^137+u^551*X^136
+u^627*X^135+u^158*X^134+u^426*X^133+u^150*X^132+u^484*X^131+u^677*X^130+u^120*X^129+u^75*X^128+u^223*X^127+u^186*X^126
+u^566*X^125+u^258*X^124+u^405*X^123+u^344*X^122+u^82*X^121+u^445*X^120+u^79*X^119+u^215*X^118+u^65*X^117+u^191*X^116
+u^251*X^115+u^169*X^114+u^282*X^113+u^510*X^112+u^617*X^111+u^116*X^110+u^639*X^109+u^552*X^108+u^468*X^107+u^340*X^106
+u^514*X^105+u^527*X^104+u^581*X^103+u^280*X^102+u^360*X^101+u^442*X^100+u^197*X^99+u^696*X^98+u^129*X^97+u^74*X^96
+u^720*X^95+u^235*X^94+u^479*X^93+u^382*X^92+u^308*X^91+u^547*X^90+u^716*X^89+u^35*X^88+u^117*X^87+u^548*X^86+u^94*X^85
+u^133*X^84+u^200*X^83+u^156*X^82+u^212*X^81+u^270*X^80+u^608*X^79+u^418*X^78+u^197*X^77+u^656*X^76+u^255*X^75+u^75*X^74
+u^651*X^73+u^462*X^72+u^285*X^71+u^24*X^70+u^306*X^69+u^396*X^68+u^10*X^67+u^87*X^66+u^656*X^65+u^672*X^64+u^500*X^63
+u^243*X^62+u^700*X^61+u^216*X^60+u^58*X^59+u^471*X^58+u^395*X^57+u^14*X^56+u^184*X^55+u^582*X^54+u^440*X^53+u^131*X^52
+u^137*X^51+u^386*X^50+u^422*X^49+u^79*X^48+u^371*X^47+u^680*X^46+u^321*X^45+u^34*X^44+u^258*X^43+u^23*X^42+u^525*X^41
+u^47*X^40+u^258*X^39+u^246*X^38+u^594*X^37+u^658*X^36+u^490*X^35+u^306*X^34+u^300*X^33+u^61*X^32+u^692*X^31+u^632*X^30
+u^198*X^29+u^485*X^28+u^594*X^27+u^252*X^26+u^342*X^25+u^112*X^24+u^92*X^23+u^455*X^22+u^559*X^21+u^241*X^20+u^656*X^19
+u^45*X^18+u^366*X^17+u^677*X^16+u^61*X^15+u^42*X^14+u^363*X^13+u^681*X^12+u^457*X^11+u^610*X^10+u^573*X^9+u^624*X^8
+u^584*X^7+u^102*X^6+u^692*X^5+u^273*X^4+u^500*X^3+u^700*X^2+u^232*X+u^445;
factors := [
<X, 230>,
<X+u^224, 1>,
<X+u^440, 1>,
<X^2+u^544*X+u^205, 1>,
<X^3+u^141*X^2+u^567*X+u^486, 1>,
<X^16+u^374*X^15+u^543*X^14+u^477*X^13+u^40*X^12+u^9*X^11+u^149*X^10+u^193*X^9+u^76*X^8+u^579*X^7+u^580*X^6+u^521*X^5
 +u^513*X^4+u^22*X^3+u^431*X^2+u^166*X+u^95, 1>,
<X^20+u^559*X^19+u^230*X^18+u^283*X^17+u^118*X^16+u^529*X^15+u^603*X^14+u^228*X^13+u^618*X^12+u^321*X^11+u^335*X^10
 +u^299*X^9+u^86*X^8+u^138*X^7+u^163*X^6+u^262*X^5+u^571*X^4+u^135*X^3+u^544*X^2+u^361*X+u^21, 1>,
<X^21+u^438*X^20+u^94*X^19+u^474*X^18+u^625*X^17+u^207*X^16+u^146*X^15+u^458*X^14+u^525*X^13+u^590*X^12+u^187*X^11
```

```
 +u^94*X^10+u^376*X^9+u^64*X^8+u^544*X^7+u^278*X^6+u^29*X^5+u^612*X^4+u^480*X^3+u^181*X^2+u^52*X+u^63, 1>,
<X^23+u^294*X^22+u^523*X^21+u^5*X^20+u^306*X^19+u^332*X^18+u^467*X^17+u^194*X^16+u^727*X^15+u^335*X^14+u^281*X^13
 +u^472*X^12+u^129*X^11+u^719*X^10+u^582*X^9+u^240*X^8+u^483*X^7+u^102*X^6+u^444*X^5+u^236*X^4+u^212*X^3+u^491*X^2+u^491*X
 +u^641, 1>,
<X^27+u^178*X^26+u^652*X^25+u^717*X^24+u^381*X^23+u^578*X^22+u^523*X^21+u^234*X^20+u^642*X^19+u^306*X^18+u^291*X^17
 +u^73*X^16+u^243*X^15+u^396*X^14+u^723*X^13+u^458*X^12+u^215*X^11+u^700*X^10+u^335*X^9+u^493*X^8+u^642*X^7+u^317*X^6
 +u^727*X^5+u^252*X^4+u^538*X^3+u^87*X^2+u^406*X+u^701, 1>,
<X^27+u^649*X^26+u^405*X^25+u^529*X^24+u^24*X^23+u^257*X^22+u^304*X^21+u^7*X^20+u^324*X^19+u^683*X^18+u^349*X^17
 +u^294*X^16+u^522*X^15+u^456*X^14+u^546*X^13+u^221*X^12+u^601*X^11+u^238*X^10+u^379*X^9+u^12*X^8+u^576*X^7+u^587*X^6
 +u^201*X^5+u^2*X^4+u^103*X^3+u^376*X^2+u^339*X+u^274, 1>,
<X^28+u^37*X^27+u^214*X^26+u^473*X^25+u^249*X^24+u^497*X^23+u^553*X^22+u^485*X^21+u^282*X^20+u^13*X^19+u^308*X^18
 +u^356*X^17+u^190*X^16+u^308*X^15+u^400*X^14+u^504*X^13+u^339*X^12+u^96*X^11+u^532*X^10+u^161*X^9+u^572*X^8+u^116*X^7
 +u^686*X^6+u^420*X^5+u^457*X^4+u^571*X^3+u^230*X^2+u^306*X+u^611, 1>,
<X^28+u^296*X^27+u^311*X^26+u^446*X^25+u^220*X^24+u^217*X^23+u^492*X^22+u^159*X^21+u^408*X^20+u^539*X^19+u*X^18+u^109*X^17
 +u^111*X^16+u^179*X^15+u^354*X^14+u^86*X^13+u^587*X^12+u^569*X^11+u^268*X^10+u^647*X^9+u^134*X^8+u^237*X^7+u^173*X^6
 +u^107*X^5+u^388*X^4+u^58*X^3+u^209*X^2+u^244*X+u^104, 1>,
<X^28+u^642*X^27+u^403*X^26+u^24*X^25+u^412*X^24+u^462*X^23+u^434*X^22+u^229*X^21+u^643*X^20+u^721*X^19+u^524*X^18
 +u^663*X^17+u^469*X^16+u^402*X^15+u^210*X^14+u^183*X^13+u^147*X^12+u^606*X^11+u^410*X^10+u^403*X^9+u^163*X^8+u^81*X^7
 +u^526*X^6+u^627*X^5+u*X^3+u^328*X^2+u^611*X+u^314, 1>,
<X^30+u^94*X^29+u^220*X^28+u^4*X^27+u^14*X^26+u^585*X^25+u^528*X^24+u^448*X^23+u^505*X^22+u^307*X^21+u^412*X^20+u^620*X^19
 +u^449*X^18+u^248*X^17+u^72*X^16+u^499*X^15+u^667*X^14+u^410*X^13+u^268*X^12+u^563*X^11+u^420*X^10+u^377*X^9+u^218*X^8
 +u^698*X^7+u^711*X^6+u^613*X^5+u^30*X^4+u^571*X^3+u^386*X^2+u^600*X+u^634, 1>];
```

C Initial splitting in GF($3^{5\cdot 479}$)

We iterated from $T_0, gT_0, g^2T_0, \ldots$ to $g^{56\cdot 1024-1}T_0$ (each of them producing 479 different polynomials, hence a space search of $2^{24.7}$) over 56 cores (Intel Xeon E5-2609 2.40GHz) for a total time of 1239 hours (22.12 hours per core). We obtained 188 polynomials that are 50-smooth up to 60-smooth, among them 20.7% of the polynomials are of degree 384, 78.2% of degree 383, and 1.1% of degree 382.

b	$\Pr_{3^5}(383,b)$	$\Pr_{3^5}(383,b)\cdot 2^{24.7}$	#	t	row	$\deg T'$	X^i	leading coeff $\in \mathbb{F}_{3^5}$
50	$2^{-22.96}$	$2^{1.74}=3.34$	1	23940	471	383	X^{94}	a^{69}
51	$2^{-22.26}$	$2^{2.44}=5.42$	2	26518	186	384	X^{36}	a^{69}
				53584	38	383	X^{7}	a^{230}
52	$2^{-21.59}$	$2^{3.11}=8.63$	0					
53	$2^{-20.95}$	$2^{3.75}=13.45$	4					
54	$2^{-20.33}$	$2^{4.37}=20.67$	10					
55	$2^{-19.74}$	$2^{4.96}=31.12$	8					
56	$2^{-19.18}$	$2^{5.52}=45.88$	12					
57	$2^{-18.64}$	$2^{6.06}=66.71$	18					
58	$2^{-18.13}$	$2^{6.57}=95.00$	26					
59	$2^{-17.63}$	$2^{7.07}=134.36$	42					
60	$2^{-17.16}$	$2^{7.54}=186.10$	65					

Table 6: Results: b-smooth polynomials in $\mathbb{F}_{3^{5\cdot 479}}$ found for $t \in \{0,\ldots,57343 \approx 2^{15.8}\}$

The 471-th row of the matrix corresponding to $T = g^{23940}T_0$ gives $T = uvT'$ where T' is 50-smooth.

```
// found a 50-smooth decomposition: T = g^23940*T0 = lc*X^94*poly, row = 471, leading coeff=a^69:
poly := X^383+a^191*X^382+a^207*X^381+a^119*X^380+a^214*X^379+a^49*X^378+a^199*X^377+a^78*X^376+a^217*X^375+a^152*X^374
+a^104*X^373+a^18*X^372+a^218*X^371+a^170*X^370+a^196*X^369+a^32*X^368+a^107*X^367+a^116*X^366+a^10*X^365+a^68*X^364
+a^34*X^363+a^46*X^362+2*X^361+a^150*X^360+a^77*X^359+a^48*X^358+a^95*X^357+a^70*X^356+a^8*X^355+a^217*X^354+a^203*X^353
+a^115*X^352+a^184*X^351+a^216*X^350+a^31*X^349+a^26*X^348+a^99*X^347+a^220*X^346+a^114*X^345+a^99*X^344+a^217*X^343
+a^196*X^342+a^27*X^341+a^201*X^340+a^173*X^339+a^156*X^338+a^79*X^337+a^33*X^336+a^160*X^334+a^23*X^333+2*X^332
+a^27*X^331+a^63*X^330+a^238*X^329+a^137*X^328+a^122*X^327+a^118*X^326+a^43*X^325+a^90*X^324+a^75*X^323+a^149*X^322
+a^76*X^321+a^138*X^320+a^179*X^319+a^126*X^318+a^33*X^317+a*X^316+a^81*X^315+a^189*X^314+a^163*X^313+a^99*X^312
+a^202*X^311+a^219*X^310+a^65*X^309+a^235*X^308+a^86*X^307+a^109*X^306+a^66*X^305+a^114*X^304+a^221*X^303+a^107*X^302
+a^122*X^301+a^179*X^300+a^55*X^299+a^4*X^298+a^25*X^297+a^159*X^296+a^219*X^295+a^60*X^294+a^199*X^293+a^208*X^292
+a^101*X^291+a^17*X^290+a^92*X^289+a^163*X^288+a^201*X^287+a^5*X^286+a^203*X^285+a^52*X^284+a^135*X^283+a^172*X^282
+a^20*X^281+a^110*X^280+a^90*X^279+a^38*X^278+a^13*X^277+a^167*X^276+a^101*X^275+a^69*X^274+a^34*X^273+a^53*X^272
+a^14*X^271+a^190*X^270+a^156*X^269+a^123*X^268+a^185*X^267+a^206*X^266+a^6*X^265+a^58*X^264+a^218*X^263+a^109*X^262
+a^241*X^261+a^8*X^260+a^163*X^259+a^90*X^258+a^218*X^257+a^132*X^256+a^189*X^255+a^171*X^254+a^159*X^253+a^228*X^252
+a^211*X^251+a^131*X^250+a^122*X^249+a^163*X^248+a^223*X^247+a^114*X^246+a^147*X^245+a^58*X^244+a^98*X^243+a^154*X^242
+a^205*X^241+a^217*X^239+a^80*X^238+a^103*X^237+a^96*X^236+a^152*X^235+a^22*X^234+a^58*X^233+a^127*X^232+a^73*X^231
+a^142*X^230+a^234*X^229+a^18*X^228+a^164*X^227+a^160*X^226+a^95*X^225+a^99*X^224+a^172*X^223+a^95*X^222+a^148*X^221
+a^180*X^220+a^71*X^219+a^207*X^218+a^64*X^217+a^136*X^216+a^141*X^215+a^127*X^214+a^22*X^213+a^15*X^212+a^131*X^211
+a^108*X^210+a^93*X^209+a^177*X^208+a^225*X^207+a^241*X^206+a^34*X^205+a^63*X^204+a^119*X^203+a^221*X^202+a^213*X^201
+a^136*X^200+a^184*X^199+a^170*X^198+a^193*X^197+a^228*X^196+a^133*X^195+a^166*X^194+a^35*X^193+a^139*X^192+a^212*X^191
+a^225*X^190+a^105*X^189+a^130*X^188+a^31*X^187+a^194*X^186+a^92*X^185+a^38*X^184+a^198*X^183+a^117*X^182+a^239*X^181
```

```
+a^149*X^180+a^225*X^179+a^170*X^178+a^225*X^177+a^231*X^176+a^195*X^175+a^160*X^174+a^35*X^173+a^15*X^172+a^108*X^171
+a^71*X^170+a^201*X^169+a^27*X^168+a^153*X^167+a^120*X^166+a^55*X^165+a^108*X^164+a^203*X^163+a^177*X^162+a^239*X^161
+a^66*X^160+a^50*X^159+a^63*X^158+a^225*X^157+a^225*X^156+a^108*X^155+a^112*X^154+a^142*X^153+a^227*X^152+a^52*X^151
+a^18*X^150+a^6*X^149+a^202*X^148+a^161*X^147+a^197*X^146+a^104*X^145+a^155*X^144+a^166*X^143+a^219*X^142+a^223*X^141
+a^52*X^140+a^36*X^139+a^171*X^138+a^84*X^137+a^20*X^136+a^68*X^135+a^170*X^134+a^12*X^133+a^181*X^132+a^212*X^131
+a^45*X^130+a^39*X^129+a^63*X^128+a^65*X^127+a^163*X^126+a^131*X^125+a^148*X^124+a^134*X^123+a^202*X^122+a^122*X^121
+a^29*X^120+a^196*X^119+a^236*X^118+a^155*X^117+a^6*X^116+a^124*X^115+a^228*X^114+a^205*X^113+a^194*X^112+a^78*X^111
+a^36*X^110+a^159*X^109+a^91*X^108+a^143*X^107+a^233*X^106+a^232*X^105+a^87*X^104+a^66*X^103+a^207*X^102+a^26*X^101
+a^129*X^100+a^136*X^99+a^177*X^98+a^162*X^97+a^206*X^96+a^17*X^95+a^222*X^94+a^90*X^93+a^108*X^92+a^151*X^91+a^162*X^90
+a^223*X^89+a^123*X^88+a^100*X^87+a^209*X^86+a^25*X^85+a^191*X^84+a^125*X^83+a^91*X^82+a^235*X^81+a^92*X^80+a^127*X^79
+a^201*X^78+a^122*X^77+a^69*X^76+a^234*X^75+a*X^74+a^137*X^73+a^235*X^72+a^42*X^71+a^222*X^70+a^230*X^69+a^30*X^68
+a^203*X^67+a^118*X^66+a^12*X^65+a^123*X^64+a^213*X^63+a^241*X^62+a^15*X^61+a^6*X^60+a^55*X^59+a^217*X^58+a^162*X^57
+a^152*X^56+a^222*X^55+a^144*X^54+a*X^53+a^145*X^52+a^88*X^51+a^176*X^50+a^238*X^49+a^112*X^48+a^208*X^47+a^78*X^46
+a^117*X^45+a^57*X^44+a^168*X^43+a^48*X^42+a^38*X^41+a^210*X^40+a^79*X^39+a^78*X^38+a^155*X^37+a^170*X^36+a^3*X^35
+a^104*X^34+a^198*X^33+a^42*X^32+a^99*X^31+a^16*X^30+a^232*X^29+a^218*X^28+a^231*X^27+a^100*X^26+a^139*X^25+a^17*X^24
+a^212*X^23+a^234*X^22+a^72*X^21+a^37*X^20+a^158*X^19+a^116*X^18+a^186*X^17+a^182*X^16+a^215*X^15+a^148*X^14+a^4*X^13
+a^175*X^12+a^14*X^11+a^158*X^10+a^181*X^9+a^63*X^8+a^58*X^7+a^77*X^6+a^128*X^5+a^176*X^4+a^144*X^3+a^64*X^2+a^204*X+a^78;
factors := [
<X, 94>,
<X^2+a^40*X+a^9, 1>,
<X^2+a^67*X+a^192, 1>,
<X^11+a^93*X^10+a^17*X^9+a^73*X^8+a^66*X^7+a^152*X^6+a^153*X^5+a^143*X^4+a*X^3+a^229*X^2+a^212*X+a^87, 1>,
<X^16+a^100*X^15+a^136*X^14+a^174*X^13+a^222*X^12+a^172*X^11+a^72*X^10+a^39*X^9+a^84*X^8+a^85*X^7+a^126*X^6+a^181*X^5
 +a^235*X^4+a^54*X^3+a^3*X^2+a^66*X+a^158, 1>,
<X^17+a^193*X^16+a^163*X^15+a^32*X^14+a^89*X^13+a^41*X^12+a^66*X^11+a^112*X^10+a^127*X^9+a^184*X^8+a^231*X^7+a^95*X^6
 +a^43*X^5+a^100*X^4+a^224*X^3+a^213*X^2+a^110*X+a^241, 1>,
<X^18+a^13*X^17+a^87*X^16+a^105*X^15+a^219*X^14+a^65*X^13+a^120*X^12+a^233*X^11+a^107*X^10+a^74*X^9+a^123*X^8+a^63*X^7
 +a^85*X^6+a^162*X^5+a^17*X^4+a^117*X^3+a^138*X^2+a^138*X+a^96,
 1>,
<X^18+a^153*X^17+a^63*X^16+a^151*X^15+a^62*X^14+a^238*X^13+a^237*X^12+a^189*X^11+a^220*X^10+a^55*X^9+a^92*X^8+a^95*X^7
 +a^95*X^6+a^72*X^5+a^151*X^4+a^93*X^3+a^130*X^2+a^203*X+a^118,
 1>,
<X^20+a^8*X^19+a^164*X^18+a^43*X^17+a^67*X^16+a^198*X^15+a^3*X^14+a^28*X^13+a^153*X^12+a^145*X^11+a^76*X^10+a^128*X^9
 +a^75*X^8+a^16*X^7+a^210*X^6+a^222*X^5+a^182*X^4+a^117*X^3+a^5*X^2+a^238*X+a^26, 1>,
<X^26+a^236*X^25+a^20*X^24+a^66*X^23+a^226*X^22+a^56*X^21+a^69*X^20+a^222*X^19+a^105*X^18+a^84*X^17+a^109*X^16+a^147*X^15
 +a^199*X^14+a^40*X^13+a^75*X^12+a^37*X^11+a^227*X^10+a^171*X^9+a^130*X^8+a^172*X^7+a^160*X^6+a^90*X^5+a^41*X^4+a^145*X^3
 +a^207*X^2+a^74*X+a^93, 1>,
<X^32+a^125*X^31+a^230*X^30+a^187*X^29+a^231*X^28+a^107*X^27+a^115*X^26+a^43*X^25+a^9*X^24+a^141*X^23+a^210*X^22+a^17*X^21
 +a^8*X^20+a^139*X^19+a^63*X^18+a^190*X^17+a^199*X^16+a^180*X^15+a^58*X^14+a^125*X^13+a^196*X^12+a^209*X^11+a^48*X^10
 +a^241*X^9+a^43*X^8+a^94*X^7+a^24*X^6+a^35*X^5+a^208*X^4+a^120*X^3+a^35*X^2+a^67*X+a^158, 1>,
<X^36+a^80*X^35+a^104*X^34+a^14*X^33+a^114*X^32+a^174*X^31+a^174*X^30+a^230*X^29+a^49*X^28+a^15*X^27+a^101*X^26+a^95*X^25
 +a^53*X^24+a^53*X^23+a^187*X^22+a^139*X^21+a^169*X^20+a^146*X^19+a^186*X^18+a^36*X^17+a^100*X^16+a^123*X^15+a^173*X^14
 +a^86*X^13+a^120*X^12+a^95*X^11+a^235*X^10+a^195*X^9+a^224*X^8+a^232*X^7+a^167*X^6+a^12*X^5+a^2*X^4+a^37*X^3+a^115*X^2
 +a^136*X+a^175, 1>,
<X^36+a^119*X^35+a^119*X^34+a^15*X^33+a^11*X^32+a^166*X^31+a^161*X^30+a^26*X^29+a^217*X^28+a^63*X^27+a^79*X^26+a^207*X^25
 +a^141*X^24+a^54*X^23+a^238*X^22+a^158*X^21+a^174*X^20+a^204*X^19+a^229*X^18+a^181*X^17+a^181*X^16+a^197*X^15+a^58*X^14
 +a^187*X^13+a^9*X^12+a^68*X^11+a^81*X^10+a^202*X^9+a^182*X^8+a^164*X^7+a^35*X^6+a^21*X^5+a^10*X^4+a^12*X^3+a^92*X^2
 +a^58*X+a^96, 1>,
<X^49+a^175*X^48+a^109*X^47+a^162*X^46+a^175*X^45+a^144*X^44+a^166*X^43+a^20*X^42+a^234*X^41+a^238*X^40+a^31*X^39
 +a^153*X^38+a^130*X^37+a^91*X^36+a^90*X^35+a^24*X^34+a^107*X^33+a^92*X^32+a^73*X^31+a^210*X^30+a^86*X^29+a^232*X^27
 +a^219*X^26+a^154*X^25+a^123*X^24+a^160*X^23+a^201*X^22+a^225*X^21+a^53*X^20+a^199*X^19+a^132*X^18+a^187*X^17+a^202*X^16
 +a^237*X^15+a^217*X^14+a^105*X^13+a^198*X^12+a^74*X^11+a^221*X^10+a^233*X^9+a^64*X^8+a^166*X^7+a^5*X^6+a^138*X^5+a^98*X^4
 +a^114*X^3+a^190*X^2+a^166*X+a^149, 1>,
<X^50+a^78*X^48+a^70*X^47+a^201*X^46+a^87*X^45+a^26*X^44+a^133*X^43+a^229*X^42+a^116*X^41+a^189*X^40+a^8*X^39+a^135*X^38
 +a^90*X^37+a^98*X^36+a^222*X^35+a^42*X^34+a^186*X^33+a^203*X^32+a^19*X^31+a^25*X^30+a^89*X^29+a^238*X^28+a^181*X^27
 +a^197*X^26+a^8*X^25+a^168*X^24+a^27*X^23+a^94*X^22+a^209*X^21+a^159*X^20+a^197*X^19+a^47*X^18+a^143*X^17+a^115*X^16
 +a^101*X^15+a^113*X^14+a^30*X^13+a^185*X^12+a^142*X^11+a^108*X^10+a^129*X^9+a^198*X^8+a^96*X^7+a^184*X^6+a^223*X^5
 +a^24*X^4+a^81*X^3+a^115*X^2+a^59*X+a^52, 1>,
<X^50+a^31*X^49+a^65*X^48+a^90*X^47+a^47*X^46+a^128*X^45+a^184*X^44+a^21*X^43+a^90*X^42+a^160*X^41+a^55*X^40+a^37*X^39
 +a^131*X^38+a^175*X^37+2*X^36+a^75*X^35+a^46*X^34+a^22*X^33+a^215*X^32+a^139*X^31+a^68*X^30+a^141*X^29+a^190*X^28
 +a^80*X^27+a^161*X^26+a^215*X^25+a^77*X^24+a^169*X^23+a^106*X^22+a^49*X^21+a^215*X^20+a^182*X^19+a^132*X^18+a^61*X^17
 +a^45*X^16+a^38*X^15+a^87*X^14+a^53*X^13+a^165*X^12+a^19*X^11+a^170*X^10+a^235*X^9+a^205*X^8+a^61*X^7+a^210*X^6+a^163*X^5
 +a^31*X^4+a^224*X^3+a^217*X^2+a^65*X+a^122, 1>];
```

D Polynomial selection algorithms

For completeness, we recall here the main polynomial selection methods available in the literature: the JLSV_1 method from [31, §2.3], the JLSV_2 method from [31, §3.2], the generalized Joux–Lercier method from [9, §3.2] and [43, §2] the Conjugation method from [9, §3.3], and the Joux–Pierrot method [34] dedicated to finite fields whose characteristic p is of special form, given by a polynomial of degree at least 2.

Algorithm 6: Polynomial selection with the JLSV_1 method

Input: p prime and n integer
Output: f, g, φ with $f, g \in \mathbb{Z}[x]$ irreducible and $\varphi = \gcd(f \bmod p, g \bmod p)$ in $\mathbb{F}_p[x]$ irreducible of degree n

Select $f_1(x), f_0(x)$, two polynomials with small integer coefficients, $\deg f_1 < \deg f_0 = n$
repeat
  choose $y \leq \lceil \sqrt{p} \rceil$
until $f = f_0 + y f_1$ *is irreducible in* $\mathbb{F}_p[x]$
$(u, v) \leftarrow$ a rational reconstruction of y modulo p
$g \leftarrow v f_0 + u f_1$
return $(f, g, \varphi = f \bmod p)$

Algorithm 7: Polynomial selection with the generalized Joux–Lercier method (gJL)

Input: p prime, n integer and $d \geq n$ integer
Output: f, g, φ with $f, g \in \mathbb{Z}[x]$ irreducible and $\varphi = \gcd(f \bmod p, g \bmod p)$ in $\mathbb{F}_p[x]$ irreducible of degree n

Choose a polynomial $f(x)$ of degree $D+1$ with small integer coefficients which has a monic irreducible factor $\varphi(x) = \varphi_0 + \varphi_1 x + \cdots + x^n$ of degree n modulo p
Reduce the following matrix using LLL

$$M = \begin{bmatrix} p & & & & \\ & \ddots & & & \\ & & p & & \\ \varphi_0 & \varphi_1 & \cdots & 1 & \\ & \ddots & \ddots & & \ddots \\ & & \varphi_0 & \varphi_1 & \cdots & 1 \end{bmatrix} \begin{matrix} \left.\vphantom{\begin{matrix}a\\a\\a\\a\end{matrix}}\right\} \deg \varphi = n \\ \left.\vphantom{\begin{matrix}a\\a\\a\\a\end{matrix}}\right\} D + 1 - n \end{matrix} \quad \text{to get LLL}(M) = \begin{bmatrix} g_0 & g_1 & \cdots & g_D & \\ & & & & \\ & & * & & \\ & & & & \end{bmatrix}.$$

return $(f, g = g_0 + g_1 x + \cdots + g_D x^D, \varphi)$

Algorithm 8: Polynomial selection with the Conjugation method

Input: p prime and n integer
Output: f, g, φ with $f, g \in \mathbb{Z}[x]$ irreducible and $\varphi = \gcd(f \bmod p, g \bmod p)$ in $\mathbb{F}_p[x]$ irreducible of degree n

repeat
  Select $g_1(x), g_0(x)$, two polynomials with small integer coefficients, $\deg g_1 < \deg g_0 = n$
  Select $P_y(Y)$ a quadratic, monic, irreducible polynomial over $\mathbb{Z}$ with small coefficients
until $P_y(Y)$ *has a root* y *in* $\mathbb{F}_p$ *and* $\varphi(x) = g_0(x) + y g_1(x)$ *is irreducible in* $\mathbb{F}_p[x]$
$f \leftarrow \mathrm{Res}_Y(P_y(Y), g_0(x) + Y g_1(x))$
$(u, v) \leftarrow$ a rational reconstruction of y
$g \leftarrow v g_0 + u g_1$
return (f, g, φ)

Algorithm 9: Polynomial selection with the JLSV_2 method

Input: p prime, n integer and $D \geq n$ integer
Output: f, g, φ with $f, g \in \mathbb{Z}[x]$ irreducible and $\varphi = \gcd(f \bmod p, g \bmod p)$ in $\mathbb{F}_p[x]$ irreducible of degree n

Choose a monic polynomial $g_0(x)$ of degree n with small integer coefficients ;
Choose an integer $W \approx p^{1/(D+1)}$, but slightly larger, and set $g(x) = g_0(x + W) = c_0 + c_1 x + \cdots + x^n$;
Reduce the rows of the following matrix using LLL

$$M = \begin{bmatrix} p & & & & \\ & \ddots & & & \\ & & p & & \\ c_0 & c_1 & \cdots & 1 & \\ & \ddots & \ddots & & \ddots \\ & & c_0 & c_1 & \cdots & 1 \end{bmatrix} \begin{matrix} \left.\vphantom{\begin{matrix}a\\a\\a\\a\end{matrix}}\right\} \deg \varphi = n \\ \left.\vphantom{\begin{matrix}a\\a\\a\\a\end{matrix}}\right\} D + 1 - n \end{matrix}, \text{ to get LLL}(M) = \begin{bmatrix} f_0 & f_1 & \cdots & f_D & \\ & & & & \\ & & * & & \\ & & & & \end{bmatrix}.$$

return $(f = f_D x^D + \cdots + f_0, g, \varphi = g \bmod p)$

Algorithm 10: Polynomial selection with the Joux–Pierrot method

Input: p prime, $p = P(x_0)$ where P is a polynomial of degree ≥ 2, and n integer
Output: f, g, φ with $f, g \in \mathbb{Z}[x]$ irreducible and $\varphi = \gcd(f \bmod p, g \bmod p)$ in $\mathbb{F}_p[x]$ irreducible of degree n

repeat
  Choose $g(x) = x^n + R(x) - x_0$ with R of small degree and tiny coefficients
  $f(x) \leftarrow P(x^n + R(x))$ `// where` P `is the polynomial s.t.` $p = P(x_0)$
until *f and g are irreducible*
return $(f, g, \varphi = g)$

E Pomerance's Early Abort Strategy

We write in this section a proof for Lemma 2 and Lemma 3 that state the complexity of Algorithms 2 and 3. The proofs are taken from Pomerance [46] and Barbulescu [5, Chapter 4] and adapted to the notations of Algorithms 2 and 3.

Assuming that $B_1 = L_q[\alpha, \gamma]$, the first B_1^θ-smoothness test costs $L_q[\frac{\alpha}{2}, \sqrt{2\alpha\gamma\theta} + o(1)]$, where $B_1^\theta = L_q[\alpha, \gamma\theta]$. The second B_1-smoothness test costs $L_q[\frac{\alpha}{2}, \sqrt{2\alpha\gamma}]$. The very subtle point is how to set the exponent b that will rule the early abort test, so that the actual B_1-smooth elements pass the test, while most of the other numbers are discarded at this point. The first step is to estimate the number of integers m that can be written $m = m_0 m_1$, where m_0 is B_1^θ-smooth, and m_1 is smaller than $q^{e(1-b)}$. Let $0 < \theta, b < 1$ given and let

$$M_1 = \left\{ \begin{array}{l} m = m_0 m_1 \leq q^e \text{ s.t. } m_0 \text{ is the } B_1^\theta\text{-smooth part of } m, \\ \text{all the prime factors of } m_1 \text{ are greater than } B_1^\theta, \\ \text{and } m_1 \leq q^{e(1-b)} \end{array} \right\} \tag{7}$$

and

$$M_2 = \left\{ \begin{array}{l} m = m_0 m_1 \leq q^e \text{ s.t. } m_0 \text{ is the } B_1^\theta\text{-smooth part of } m, \\ \text{all the prime factors of } m_1 \text{ are greater than } B_1^\theta, \\ m_1 \leq q^{e(1-b)} \text{ and } m_1 \text{ is } B_1\text{-smooth} \end{array} \right\} \tag{8}$$

In order to count the number of integers in these two sets, Pomerance denotes by $\psi(x, y, z)$ the number of positive integers less than x whose prime factors are in the interval $[z, y]$ (we have $x > y > z$), and Barbulescu denotes by $\Psi(x, y, z)$ the set of such integers, that is $\psi(x, y, z) = \#\Psi(x, y, z)$.

We will need to estimate the probability of an integer $m \leq q^e$ to fall in M_1, then the probability of an integer in M_1 to fall in M_2, i.e. to be B_1-smooth. Theorem 3 will be needed.

Theorem 3. *[46, Theorem 2.2 p. 95] Let $\varepsilon > 0$ be arbitrary. If $u = \frac{\log x}{\log y}$ and if $z < y^{1 - \frac{1}{\log u}}$, then*

$$\psi(x, y, z) = x \cdot u^{-u + o(u)}$$

uniformly for $(\log x)^{1+\varepsilon} < y < \exp\big((\log x)^{1-\varepsilon}\big)$, $x \geq 10$.

Lemma 5. *Let $0 < \gamma, b, \theta < 1$. Let M_1 denote the set of $m \in [1, q^e]$ s.t. m has a divisor m_0 which is B_1^θ-smooth, and $m/m_0 \leq q^{e(1-b)}$. Let M_2 denote the set of m with the same properties and also m is B_1-smooth. Then*

$$\#M_1 = q^e L_q \left[1 - \alpha, -(1-\alpha)\frac{eb}{\gamma\theta}\right]^{1+o(1)}$$

$$\#M_2 = q^e L_q \left[1 - \alpha, -(1-\alpha)\left(\frac{eb}{\gamma\theta} + \frac{e(1-b)}{\gamma}\right)\right]^{1+o(1)}$$

Proof. This is quite straightforward to see that

$$M_1 = \{m = m_0 m_1,\ m \leq q^e,\ m_1 \leq q^{e(1-b)},\ m_0 \text{ is } B_1^\theta\text{-smooth},\ p \mid m_1 \Rightarrow B_1^\theta < p \leq q^{e(1-b)}\}$$

and the set of such m_0 is larger than $\Psi(q^{eb}, B_1^\theta, 1)$ (since we do not strictly have $m_0 \leq q^{eb}$), and the set of such m_1 is $\Psi(q^{e(1-b)}, q^{e(1-b)}, B_1^\theta)$, so that

$$\#M_1 \geq \psi(q^{eb}, B_1^\theta, 1)\psi(q^{e(1-b)}, q^{e(1-b)}, B_1^\theta) .$$

In the same way, for M_2 the set of m_0 is the same and the set of m_1 is $\Psi(q^{e(1-b)}, B_1, B_1^\theta)$ so that

$$\#M_2 \geq \psi(q^{eb}, B_1^\theta, 1)\psi(q^{e(1-b)}, B_1, B_1^\theta) \ .$$

Then we need to compute

1. $\psi(q^{eb}, B_1^\theta, 1)$
2. $\psi(q^{e(1-b)}, B_1, B_1^\theta)$
3. $\psi(q^{e(1-b)}, q^{e(1-b)}, B_1^\theta)$

This is done with Theorem 3 [46, Theorem 2.2 and 3.3], and we obtain, where again $B_1 = L_q[\alpha, \gamma]$:

1. $\psi(q^{eb}, B_1^\theta, 1) = q^{eb} L_q\left[(1-\alpha), -(1-\alpha)\frac{eb}{\gamma\theta}\right]$
2. $\psi(q^{e(1-b)}, B_1, B_1^\theta) = \psi(q^{e(1-b)}, B_1, 1) = q^{e(1-b)} L_q\left[(1-\alpha), -(1-\alpha)\frac{e(1-b)}{\gamma}\right]$
3. $\psi(q^{e(1-b)}, q^{e(1-b)}, B_1^\theta) = q^{e(1-b)} L_q^{o(1)}$

We obtain the lower bounds for $\#M_1$ and $\#M_2$:

$$\#M_1 \geq q^{eb} q^{e(1-b)} L_q\left[(1-\alpha), -(1-\alpha)\frac{eb}{\gamma\theta}\right]^{1+o(1)}$$

and

$$\#M_2 \geq q^{e} L_q\left[1-\alpha, -(1-\alpha)\left(\frac{eb}{\gamma\theta} + \frac{e(1-b)}{\gamma}\right)\right]^{1+o(1)}$$

The proof of the upper bound is much more tedious.

E.1 Optimal parameters with one early abort test

The cost of the B_1^θ-smoothness test is $L_q[\frac{\alpha}{2}, \sqrt{2\alpha\gamma\theta} + o(1)]$. The probability that an integer $\leq q^e$ passes the test is $\#M_1/q^e$, thus we need to perform this test $q^e/\#M_1$ times before getting an integer m that will pass the test. So in the inner loop of Algorithm 2, the B_1^θ-smoothness test is done $q^e/\#M_1 = L_q\left[(1-\alpha), (1-\alpha)\frac{eb}{\gamma\theta}\right]^{1+o(1)}$ times.

The second test, of B_1-smoothness, is done for each integer that passed the first test, i.e. for each integer in M_1, until a B_1-smooth integer is found. The probability of success is $\#M_2/\#M_1$. This test is performed $\#M_1/\#M_2$ times, i.e. $L_q\left[1-\alpha, (1-\alpha)\frac{e(1-b)}{\gamma}\right]^{1+o(1)}$.

The overall cost of Algorithm 2 is

$$\left(\frac{q^e}{\#M_1}\texttt{ECM}(B_1^\theta) + \texttt{ECM}(B_1)\right)\frac{\#M_1}{\#M_2} \tag{9}$$

which is equal to

$$\left(L_q\left[1-\alpha, (1-\alpha)\frac{eb}{\gamma\theta}\right] L_q\left[\frac{\alpha}{2}, \sqrt{2\alpha\gamma\theta}\right] + L_q\left[\frac{\alpha}{2}, \sqrt{2\alpha\gamma}\right]\right) L_q\left[1-\alpha, (1-\alpha)\frac{e(1-b)}{\gamma}\right] \tag{10}$$

First we minimize the dominating parameter. The asymptotic complexity will be dominated by a term in $L_q[\max(1-\alpha, \alpha/2), \cdot]$ and this value is minimized for $1-\alpha = \alpha/2$, which gives the usual $\alpha = 2/3$ and $B_1 = L_q[2/3, \gamma]$. We obtain a complexity of

$$\left(L_q\left[\frac{1}{3}, \frac{eb}{3\gamma\theta} + \sqrt{4\gamma\theta/3}\right] + L_q\left[\frac{1}{3}, \sqrt{4\gamma/3}\right]\right) L_q\left[\frac{1}{3}, \frac{e(1-b)}{3\gamma}\right] \tag{11}$$

At this point we can minimize θ in the first $L_q[\frac{1}{3}, \cdot]$ on the left in (11). We obtain[6] $\theta = ((eb)^2/3)^{1/3}/\gamma$, and the minimal value of $\frac{eb}{3\gamma\theta} + \sqrt{\frac{4\gamma\theta}{3}}$ is $(3eb)^{1/3}$. We obtain

$$\left(L_q\left[\frac{1}{3}, (3eb)^{1/3}\right] + L_q\left[\frac{1}{3}, \sqrt{\frac{4\gamma}{3}}\right]\right) L_q\left[\frac{1}{3}, \frac{e(1-b)}{3\gamma}\right] \tag{12}$$

[6] One computes the derivative of the function $h_{u,v}(x) = u\sqrt{x} + \frac{v}{x}$: this is $h'_{u,v}(x) = \frac{u}{2\sqrt{x}} - \frac{v}{x^2}$ and find that the minimum of h for $x > 0$ is $h_{u,v}((\frac{2v}{u})^{2/3}) = 3(\frac{u^2 v}{4})^{1/3}$. With $u = (4\gamma/3)^{1/2}$ and $v = (eb)/(3\gamma)$, we obtain the minimum: $h((\frac{(eb)^2}{3})^{1/3}/\gamma) = (3eb)^{1/3}$.

Then we set $(3eb)^{1/3} = \sqrt{\frac{4\gamma}{3}}$ to minimize the left part of (12). We obtain $b = \frac{1}{3e}\left(\frac{4}{3}\gamma\right)^{3/2}$ and

$$L_q\left[\frac{1}{3}, \sqrt{\frac{4\gamma}{3}} + \frac{e(1-b)}{3\gamma}\right] = L_q\left[\frac{1}{3}, \left(\frac{4}{3}\right)^{1/2}\frac{23}{27}\sqrt{\gamma} + \frac{e}{3\gamma}\right] \tag{13}$$

Finally we minimize γ and obtain $\gamma = (e^2/3)^{1/3}(27/23)^{2/3}$, and the complexity is

$$L_q\left[\frac{1}{3}, (3e)^{1/3}\left(\frac{23}{27}\right)^{2/3}\right] . \tag{14}$$

We can check that $\gamma_{|e=1} = 0.7715845696$ and the corresponding value in the complexity is 1.296034212, as in [5, p. 49].

$$\begin{array}{lll} \gamma = (e^2/3)^{1/3}(27/23)^{2/3} & \approx_{|e=1} & 0.7715845696 \\ b = \frac{1}{3e}\left(\frac{4}{3}\gamma\right)^{3/2} & = 8/23 \approx & 0.3478260870 \\ \theta = ((eb)^2/3)^{1/3}/\gamma & = 4/9 \approx & 0.4444444444 \end{array}$$

We remark that b and θ do not depend on e.

e	1/2	2/3	1	3/2	2
special-q bound $L_q[\frac{2}{3}, \gamma]$					
$\gamma = (e^2/3)^{1/3}$ without early abort	0.4367	0.5291	0.6933	0.9085	1.1006
$\gamma = (e^2/3)^{1/3}(27/23)^{2/3}$ with early abort	0.4860	0.58883	0.7715	1.0110	1.2248
Expected running time $L_q[\frac{1}{3}, c]$					
$c = (3e)^{1/3}$ without early abort	1.1447	1.2599	1.4422	1.6510	1.8171
$c = (3e)^{1/3}(23/27)^{2/3}$ with early abort	1.0287	1.1322	1.2960	1.4836	1.6329

Table 7: New complexities of the boot step with the early abort strategy with one test, and $\theta = 4/9$, $b = 8/23$. The new complexity is $L_q[1/3, c]$ where $c = (3e)^{1/3}(23/27)^{2/3}$ in the last row of the table.

E.2 Optimal parameters with several early abort tests

It can be seen somehow as a strategy analog to the factorization of polynomials over finite fields, where after the square-free factorization, the sequence of factors of degree i are computed, for all $i = 1, 2, \ldots$. Here we compute the sequence of factors of increasing size, from $B_1^{\theta_0}$ to $B_1^{\theta_k} = B_1$ (there will be at most $k+1$ ECM tests per candidate). If at some point the remaining part is too large, then the smoothness test is aborted. The major difference for integers is their sub-exponential factorization cost, whereas the complete factorization of a polynomial over a finite field can be done in polynomial time (see [22]).

Cost: asymptotic running-time. In order to have coherent notations and indices with the ($k = 0$)-test (no early-abort, Algorithm 1) and the ($k = 1$)-test strategy (Algorithm 2), we assume that

1. if there is no early abort strategy, then $\mathbf{b} = [1]$ and $\boldsymbol{\theta} = [1]$, and one ECM test is performed;
2. if there is one early abort test, then $\mathbf{b} = [b, 1-b]$ and $\boldsymbol{\theta} = [\theta, 1]$;
3. if there are k tests, then $\boldsymbol{\theta} = [\theta_0, \ldots, \theta_{k-1}, \theta_k = 1]$ $\mathbf{b} = [b_0, \ldots, b_{k-1}]$ and by convention, we take $b_k = 1 - \sum_{i=0}^{k-1} b_i$.

In the algorithm, the size of m_i is decreasing, and we have

$$m_i \leq q^{e(1-\Sigma_b)} = q^{e(1-b_0-\ldots-b_{i-1})} .$$

We also have

$$m_0 \leq q^e \text{ and } m_{k+1} = 1 .$$

At each step, the factors of size between $B_1^{\theta_{i-1}}$ and $B_1^{\theta_i}$ are computed with ECM (where $B_1^{\theta_{-1}} = 1$), and removed from m_i. These factors are noted s_i in the algorithm. For the running-time analysis, we will need to consider m as the product of all its prime factors smaller than $B_1^{\theta_i}$: this is $\mathbf{s}_i = \prod_{j=0}^{i} s_j$ and the remaining part $m_i \leq q^{e(1-\sum_{j=0}^{i-1} b_i)}$ such that all its prime factors are greater than $B_1^{\theta_i}$. We can now define properly the sets M_i.

$$M_1 = \{m \leq q^e,\ m = s_0 m_1,\ s_0 \text{ is } B_1^{\theta_0}\text{-smooth},\ m_1 = m/s_0 \leq q^{e(1-b_0)}\}$$

$$M_i = \left\{ \begin{array}{l} m \leq q^e \text{ s.t. } m \in M_{i-1},\ m = \mathbf{s}_{i-1} m_i \text{ where } \mathbf{s}_{i-1} = \prod_{j=0}^{i-1} s_j, \\ \mathbf{s}_{i-1} \text{ is } B_1^{\theta_{i-1}}\text{-smooth, and } m_i = m/\mathbf{s}_{i-1} \leq q^{e(1-b_0-\ldots-b_{i-1})} \end{array} \right\}$$

$$M_{k+1} = \{m \leq q^e \text{ s.t. } m \in M_k,\ m \text{ is } B_1\text{-smooth, where } B_1 = B_1^{\theta_k} \text{ since } \theta_k = 1\}$$

The cost of the algorithm will be

$$\left(\cdots\left(\left(\texttt{ECM}(B_1^{\theta_0})\frac{q^e}{\#M_1} + \texttt{ECM}(B_1^{\theta_1})\right)\frac{\#M_1}{\#M_2} + \texttt{ECM}(B_1^{\theta_2})\right)\ldots\right)\frac{\#M_{k-1}}{\#M_k} + \texttt{ECM}(B_1^{\theta_k})\frac{\#M_k}{\#M_{k+1}}$$

When expending the parentheses and multiplying by $\frac{\#M_i}{\#M_{i+1}}$ one obtains

$$\frac{q^e}{\#M_{k+1}}\texttt{ECM}(B_1^{\theta_0}) + \frac{\#M_1}{\#M_{k+1}}\texttt{ECM}(B_1^{\theta_1}) + \ldots\frac{\#M_k}{\#M_{k+1}}\texttt{ECM}(B_1^{\theta_k}) = \frac{1}{\#M_{k+1}}\sum_{i=0}^{k}\#M_i\texttt{ECM}(B_1^{\theta_i})$$

where $M_0 = \{0 < m \leq q^e\}$ so that $\#M_0 = q^e$ by convention.

Then Barbulescu obtains [5, §4.4] the following theorem, that we rewrites with the notations used in this paper.

Theorem 4. *[5, Theorem 4.4.1] Let $0 < \alpha < 1$ and $\gamma > 0$ be two real numbers and let $B_1 = L_q[\alpha, \gamma]$. Let M_i be the set of integers $m \leq q^e$ that succeed the $(i-1)$-th test, for $1 \leq i \leq k$ and let M_{k+1} be the set of integers $\leq q^e$ that pass all the tests. Then we have:*

1. $\#M_{k+1} \geq q^e L_q\left[1-\alpha, -(1-\alpha)\left(\frac{eb_0}{\theta_0\gamma} + \ldots + \frac{eb_{k-1}}{\theta_{k-1}\gamma} + \frac{e(1-(b_0+\ldots+b_{k-1}))}{\gamma}\right)\right]^{1+o(1)}$;
2. $\forall 1 \leq i \leq k$, $\#M_i \leq q^e L_q\left[1-\alpha, -(1-\alpha)\left(\frac{eb_0}{\theta_0\gamma} + \ldots + \frac{eb_{i-1}}{\theta_{i-1}\gamma}\right)\right]^{1+o(1)}$.

We can now compute the expected running time of Algorithm 3. This computation was already done in [5, Ch 4], we only adapt it here to the notations of our Algorithm 3. We have

$$\frac{\#M_i}{\#M_{k+1}} = L_q\left[1-\alpha, (1-\alpha)\left(\frac{eb_i}{\theta_i\gamma} + \ldots + \frac{eb_{k-1}}{\theta_{k-1}\gamma} + \frac{e(1-(b_0+\ldots+b_{k-1}))}{\gamma}\right)\right]^{1+o(1)}$$

We also have

$$\texttt{ECM}(B_1^{\theta_i}) = L_q\left[\frac{\alpha}{2}, \sqrt{2\alpha\gamma\theta_i} + o(1)\right] \text{ and } \texttt{ECM}(B_1) = L_q\left[\frac{\alpha}{2}, \sqrt{2\alpha\gamma} + o(1)\right]$$

so that

$$\frac{\#M_i}{\#M_{k+1}}\texttt{ECM}(B_1^{\theta_i}) = L_q\left[1-\alpha, (1-\alpha)\left(\frac{eb_i}{\theta_i\gamma} + \ldots + \frac{eb_{k-1}}{\theta_{k-1}\gamma} + \frac{e(1-(b_0+\ldots+b_{k-1}))}{\gamma}\right)\right] L_q\left[\frac{\alpha}{2}, \sqrt{2\alpha\gamma\theta_i}\right]$$

Again we optimize α by setting $\alpha = \frac{2}{3}$, so that $B_1 = L_q\left[\frac{2}{3}, \gamma\right]$ and

$$\frac{\#M_i}{\#M_{k+1}}\texttt{ECM}(B_1^{\theta_i}) = L_q\left[\frac{1}{3}, \frac{eb_i}{3\theta_i\gamma} + \ldots + \frac{eb_{k-1}}{3\theta_{k-1}\gamma} + \frac{e(1-(b_0+\ldots+b_{k-1}))}{3\gamma} + \sqrt{\frac{4\gamma\theta_i}{3}}\right]^{1+o(1)}$$

Hence we have

$$\text{Total time} = \sum_{i=0}^{k} \frac{\#M_i}{\#M_{k+1}} \texttt{ECM}(B_1^{\theta_i}) = L_q\left[\frac{1}{3}, \max(f_0, \ldots, f_{k-1}, f_k)\right]$$

where

$$f_i = \frac{eb_i}{3\theta_i\gamma} + \ldots + \frac{eb_{k-1}}{3\theta_{k-1}\gamma} + \frac{e(1-(b_0+\ldots+b_{k-1}))}{3\gamma} + \sqrt{\frac{4\gamma\theta_i}{3}}$$

and

$$f_k = \frac{e(1-(b_0+\ldots+b_{k-1}))}{3\gamma} + \sqrt{\frac{4\gamma}{3}}$$

(where $\theta_k = 1$). For each f_i where the $\theta_{i+1}, \ldots, \theta_k$ and $b_1, \ldots, b_{k-1}$ take fixed values, Barbulescu minimizes f_i by minimizing [7] θ_i in $\frac{eb_i}{3\theta_i\gamma} + \sqrt{\frac{4}{3}\theta_i\gamma}$:

$$\theta_i = \left(\frac{(eb_i)^2}{3\gamma^3}\right)^{1/3} \text{ and the minimum is } \left.\frac{eb_i}{3\theta_i\gamma} + \sqrt{\frac{4}{3}\theta_i\gamma}\right|_{\theta_i \min} = (3eb_i)^{1/3}\ , \tag{15}$$

so that

$$f_i = \frac{eb_{i+1}}{3\theta_{i+1}\gamma} + \ldots + \frac{eb_{k-1}}{3\theta_{k-1}\gamma} + \frac{e(1-(b_0+\ldots+b_{k-1}))}{3\gamma} + (3eb_i)^{1/3} \tag{16}$$

Then to optimize the total time, Barbulescu solves recursively $f_i = f_{i+1}$, starting at $i+1 = k$ and decreasing i.

$$\begin{aligned} f_i = f_{i+1} &\Leftrightarrow \tfrac{eb_i}{3\theta_i\gamma} + \sqrt{\tfrac{4}{3}\theta_i\gamma} = \sqrt{\tfrac{4}{3}\theta_{i+1}\gamma} \Leftrightarrow (3eb_i)^{1/3} = \sqrt{\tfrac{4}{3}\theta_{i+1}\gamma} \\ &\Leftrightarrow b_i = \tfrac{1}{3e}\left(\tfrac{4}{3}\theta_{i+1}\gamma\right)^{3/2} \end{aligned}$$

So we can compute the numerical sequence of $\{\theta_i, b_i\}_{0\le i\le k}$ defined by

$$\begin{cases} \theta_k = 1 \\ b_i = \frac{1}{3e}\left(\frac{4}{3}\theta_{i+1}\gamma\right)^{3/2} & \text{for all } 0 \le i \le k-1 \\ \theta_i = \left(\frac{(eb_i)^2}{3\gamma^3}\right)^{1/3} & cf\ (15) \end{cases}$$

and we can show by simple induction that for $0 \le i \le k-1$,

$$\begin{cases} \theta_i = \left(\frac{4}{9}\right)^{k-i} \\ b_i = \frac{1}{3e}\left(\frac{4}{3}\gamma\right)^{3/2}\left(\frac{2}{3}\right)^{3(k-(i+1))} \end{cases}$$

We check that we have $\theta_k = 1$ which is coherent with our convention.

At this point, we have $f_i = f_j$ for all $0 \le i, j \le k$ then we only need to compute the value of f_k which is simpler. We need to compute the sum $\frac{e}{3\gamma}\sum_{i=0}^{k-1} b_i$:

$$\frac{e}{3\gamma}\sum_{i=0}^{k-1} b_i = \frac{\sqrt{\gamma}}{9}\left(\frac{4}{3}\right)^{3/2}\frac{1}{1-(2/3)^3}\left(1-\left(\frac{2}{3}\right)^{3k}\right)$$

so that

$$f_k = \frac{e}{3\gamma} - \frac{\sqrt{\gamma}}{9}\left(\frac{4}{3}\right)^{3/2}\frac{1}{1-(2/3)^3}\left(1-\left(\frac{2}{3}\right)^{3k}\right) + \sqrt{\frac{4}{3}\gamma} = A\sqrt{\gamma} + \frac{B}{\gamma}$$

and we minimize γ in f_k for a fixed k.

If $k = 1$ we get, as expected according to (13)

$$f_k = \frac{e}{3\gamma} - \frac{\sqrt{\gamma}}{9}\left(\frac{4}{3}\right)^{3/2} + \sqrt{\frac{4}{3}}\sqrt{\gamma} = \frac{e}{3\gamma} + \frac{23}{27}\left(\frac{4}{3}\right)^{1/2}\sqrt{\gamma}$$

[7] one time more with the following formula: the minimum of the function $h_{a,b}(x) = a\sqrt{x} + b/x$ for $x > 0$ is $x_{\min} = \left(\frac{2b}{a}\right)^{2/3}$ and $h(x_{\min}) = 3\left(\frac{a^2 b}{4}\right)^{1/3}$.

We finally obtain the formula

$$f_k = \frac{e}{3\gamma} + \sqrt{\gamma}\frac{2}{\sqrt{3}}\left(1 - \frac{4}{19}\left(1 - \left(\frac{2}{3}\right)^{3k}\right)\right) \tag{17}$$

and the optimal value of γ which minimizes f_k is

$$\gamma_{\min} = (e^2/3)^{1/3}\left(1 - \frac{4}{19}\left(1 - \left(\frac{2}{3}\right)^{3k}\right)\right)^{-2/3} \tag{18}$$

and f_k takes the minimal value

$$f_k|_{\gamma=\gamma_{\min}} = (3e)^{1/3}\left(1 - \frac{4}{19}\left(1 - \left(\frac{2}{3}\right)^{3k}\right)\right)^{2/3} \tag{19}$$

We found the sames values as in [5, Table 4.1] for $e = 1$.
Finally, we replace γ by the above expression in b_i and obtain that

$$b_i = \left(\frac{2}{3}\right)^{3(k-i)}\left(1 - \frac{4}{19}\left(1 - \left(\frac{2}{3}\right)^{3k}\right)\right)^{-1}$$

.

i	1	2	3	4	5	6
$k = 1$, b_i	0.348					
$k = 2$, b_i	0.109	0.367				
$k = 3$, b_i	0.033	0.110	0.373			
$k = 4$, b_i	0.010	0.033	0.111	0.375		
$k = 5$, b_i	0.003	0.010	0.033	0.111	0.375	
$k = 6$, b_i	0.0009	0.003	0.010	0.033	0.111	0.375

Table 8: Values of b_i