Improving NFS for the Discrete Logarithm Problem in Non-prime Finite Fields

Razvan Barbulescu[1,2,3(✉)], Pierrick Gaudry[1,3,4], Aurore Guillevic[1,2], and François Morain[1,2,3]

[1] Institut National de Recherche en Informatique et en Automatique (INRIA), Paris, France
razvan.barbaud@imj-prg.fr
[2] École Polytechnique/LIX, Palaiseau, France
{guillevic,morain}@lix.polytechnique.fr
[3] Centre National de la Recherche Scientifique (CNRS), Paris, France
[4] Université de Lorraine, Nancy, France
pierrick.gaudry@loria.fr

Abstract. The aim of this work is to investigate the hardness of the discrete logarithm problem in fields $\mathrm{GF}(p^n)$ where n is a small integer greater than 1. Though less studied than the small characteristic case or the prime field case, the difficulty of this problem is at the heart of security evaluations for torus-based and pairing-based cryptography. The best known method for solving this problem is the Number Field Sieve (NFS). A key ingredient in this algorithm is the ability to find good polynomials that define the extension fields used in NFS. We design two new methods for this task, modifying the asymptotic complexity and paving the way for record-breaking computations. We exemplify these results with the computation of discrete logarithms over a field $\mathrm{GF}(p^2)$ whose cardinality is 180 digits (595 bits) long.

1 Introduction

The security of cryptographic protocols relies on hard problems like integer factorization or discrete logarithm (DLP) computations in a finite group. The difficulty of the latter depends on the chosen group. While no subexponential methods for DLP instances are known for some groups (including elliptic curves), finite fields are vulnerable to variants of the Number Field Sieve (NFS) algorithm.

Getting more insight about the theoretical and the practical behaviour of NFS for non-prime fields is important in cryptography. Indeed, although cryptosystems based on discrete logarithms in non-prime finite fields are not as widely deployed as for prime fields, they can be found in two areas: torus-based and pairing-based cryptography.

Torus-based cryptography, and in particular its most popular avatars LUC [32], XTR [21] and CEILIDH [29], provides an efficient way to build a cryptosystem working in a subgroup of the multiplicative group of a finite field $\mathbb{F}_{p^n}$ where

E. Oswald and M. Fischlin (Eds.): EUROCRYPT 2015, Part I, LNCS 9056, pp. 129–155, 2015.
DOI: 10.1007/978-3-662-46800-5_6

$n > 1$ is a small integer. The size of the considered prime-order subgroup must be large enough to resist Pollard's rho attacks, and p^n must be large enough to resist NFS attacks.

In pairing-based cryptography, the security relies on the difficulty of the discrete logarithm problem in elliptic curves, and in finite fields of the form $\mathbb{F}_{q^n}$, where n is small (mostly $n \leq 20$). The table [12, Tab. 1.1] lists the available choices of elliptic curve and finite field sizes to balance the security on both sides. Due to recent progress in attacks on discrete logarithms in finite fields of small characteristic, it is also common to assume that q is prime, so that NFS is also to be considered as an attack to be resisted.

Expressing the complexity of subexponential methods is done using the L-function. If $\alpha \in [0, 1]$ and $c > 0$ are two constants, we set

$$L_Q(\alpha, c) = \exp\left((c + o(1))(\log Q)^{\alpha}(\log\log Q)^{1-\alpha}\right),$$

and simply write $L_Q(\alpha)$ if the constant c is not made explicit.

It was proven in [16] that the complexity of DLP in the medium an large characteristic case is $L_Q(1/3, c)$. This ended proving that the complexity of DLP for every finite field of cardinality Q is $L_Q(1/3, c)$. The constant c depends on the size of the characteristic p with respect to Q. It is customary to write $p = L_Q(\alpha, c)$ and to classify the different cases as follows: p is said to be *small* if $\alpha < 1/3$, *medium* if $1/3 < \alpha < 2/3$ and *large* if $2/3 < \alpha$. In this article, we target non-small characteristic finite fields, for which the state-of-art algorithm is the Number Field Sieve (NFS) and where the quasi polynomial time algorithm [2] does not apply. Schirokauer [30] studied the family of fields $\mathbb{F}_{p^n}$ for which n is constant when p goes to infinity, and obtained the same complexity as in the prime case $L_Q(1/3, \sqrt[3]{64/9})$. The variants of the algorithm by Joux, Lercier, Smart and Vercauteren [16] have complexity $L_Q(1/3, \sqrt[3]{64/9})$ for fields of large characteristic, and $L_Q(1/3, \sqrt[3]{128/9})$ in medium characteristic. A Coppersmith-like variant [3] has a complexity of $L_Q(1/3, \sqrt[3]{(92 + 26\sqrt{13})/27})$ in large characteristic, and $L_Q(1/3, \sqrt[3]{2^{13}/3^6})$ in medium characteristic. The situation is more complex in the boundary case, i.e. when $p = L_Q(2/3)$, having a complexity $L_Q(1/3, c)$ with c varying between $16/9$ and $\sqrt[3]{2^{13}/3^6}$ [3]. Finally, if the characteristic p has a special form, i.e. can be written as $p = P(m)$, for an integer $m \approx p^{1/\deg P}$ and a polynomial $P \in \mathbb{Z}[x]$ of small degree and with small coefficients, then a faster variant of NFS is available [18].

In practice, the boundary between the medium and large characteristic cases is determined by direct timing of each variant of NFS, for each pair $(\lfloor \log p \rfloor, n)$. A series of record computations were realized using the medium characteristic variant [17, Table 8.], but to our knowledge no computations have been done in non-prime fields using the large characteristic variant of NFS.

In this article, we propose two new methods to select polynomials for NFS in the context of discrete logarithms in non-prime finite fields: the Generalized Joux–Lercier (GJL) method and the Conjugation (Conj) method.

We prove the following result in Section 4.

Theorem 1. *Let $\mathbb{F}_Q$, with $Q = p^n$ be a finite field of characteristic p. Assuming the usual heuristics on smoothness of algebraic numbers, made in the analysis of the Number Field Sieve algorithm, we get the following time complexities for computing a discrete logarithm in $\mathbb{F}_Q$, depending on the polynomial selection method and the size of p compared to Q:*

1. *(Large prime case). Assuming $p \geq L_Q\left(2/3, \sqrt[3]{8/3}\right)$, NFS with the Generalized Joux–Lercier method has complexity*
$$L_Q\left(1/3, \sqrt[3]{64/9}\right).$$
2. *(Medium prime case). Assuming $p = L_Q(\alpha)$ for $1/3 < \alpha < 2/3$, NFS with the Conjugation method has complexity*
$$L_Q\left(1/3, \sqrt[3]{96/9}\right).$$
3. *(Boundary case). Assuming $p = L_Q(2/3, 12^{1/3})^{1+o(1)}$, NFS with the Conjugation method has complexity*
$$L_Q\left(1/3, \sqrt[3]{48/9}\right).$$

This improves on the previously known complexities in the medium characteristic case, and in the boundary case where p is around $L_Q(2/3)$.

When the characteristic is large, we improved on the known method of the polynomial selection (Proposition 5), but our gain is hidden in the $1 + o(1)$ exponent of the complexity formula. It led us to do a more precise analysis for sizes around the limit of what seems feasible with current and near future technology, and for extension degrees between 2 and 6. This suggests that our polynomial selection methods behave better than expected for small extension degrees.

In order to illustrate this, we implemented our methods and ran a discrete logarithm computation in a finite field of the form $\mathbb{F}_{p^2}$, where p has 90 decimal digits. With the Conjugation method, we were able to perform this in a time that is smaller than what is needed to factor an integer of 180 decimal digits.

Outline. The paper is organized as follows. In Section 2 we make a short presentation of the number field sieve. In Section 3 we present two new methods of polynomial selection for NFS. We measure their asymptotic complexity, thus proving Theorem 1, and derive non-asymptotic cost estimates for small degree extensions in Section 4. After discussing some further improvements in Section 5, we detail a record computation in $\mathbb{F}_{p^2}$ obtained with our algorithm in Section 6.

2 Sketch of the Number Field Sieve

Let us sketch the variant of NFS, the state-of-art algorithm used to compute discrete logarithms in any finite field $\mathbb{F}_{p^n}$ with non-small characteristic.

In the first stage, called *polynomial selection*, two polynomials f, g in $\mathbb{Z}[x]$ are constructed (we assume that $\deg f \geq \deg g$), such that their reductions modulo p have a common irreducible factor $\varphi \in \mathbb{F}_p[x]$ which is irreducible of degree n. This polynomial φ defines $\mathbb{F}_{p^n}$ over $\mathbb{F}_p$, and does not impact the complexity of NFS. We will use it in the final part of the algorithm, the individual logarithm stage (see below), to embed the input element, whose discrete logarithm is requested, into our representation $\mathbb{F}_{p^n} = \mathbb{F}_p[x]/\langle\varphi\rangle$.

Essentially, we impose no additional comdition on f and g. During this presentation we will consider the number fields of f and g, meaning that the two polynomials are irreducible, but everything works the same if we replace them by their irreducible factors over $\mathbb{Z}$ which are divisible by φ modulo p. Far more important is the fact that we do not make the classical assumption that f and g are monic. Indeed, the NFS algorithm can be modified to work with non-monic polynomials by adding factors of the leading coefficient to the factor base and by implementing carefully the decompostion of algebraic numbers into prime ideals. This was known in the folklore of NFS for a long time now and was also written more or less explicitly in the literature of NFS [16,20] where non-monic polynomials are used. A recent description of this technicalities is made in Section 6.5 of [4]. CADO-NFS accepts non-monic polynomials as an input, which allowed us to do experiments with our polynomials.

Let α and β be algebraic numbers such that $f(\alpha) = 0$ and $g(\beta) = 0$ and let m be a root of φ in $\mathbb{F}_{p^n}$, allowing us to write $\mathbb{F}_{p^n} = \mathbb{F}_p(m)$. Let K_f and K_g be the number fields associated with f and g respectively, and $\mathcal{O}_f$ and $\mathcal{O}_g$ their rings of integers. In the algorithm, we consider elements of $\mathbb{Z}(\alpha)$ and $\mathbb{Z}(\beta)$ as polynomials in α and β respectively, which are in general not integers. However, the only denominators that can occur are divisors of the leading coefficients of the polynomials f and g that we denote respectively by $l(f)$ and $l(g)$.

For the second stage of NFS, called *relation collection* or *sieving*, a smoothness bound B is chosen and we consider the two factor bases

$$\mathcal{F}_f = \left\{ \begin{array}{c} \text{prime ideals } \mathfrak{q} \text{ in } \mathcal{O}_f \text{ of norm less than } B \\ \text{or above prime factors of } l(f) \end{array} \right\},$$

$$\mathcal{F}_g = \left\{ \begin{array}{c} \text{prime ideals } \mathfrak{q} \text{ in } \mathcal{O}_g \text{ of norm less than } B \\ \text{or above prime factors of } l(g) \end{array} \right\}.$$

An integer is B-smooth if all its prime factors are less than B. For any polynomial $\phi(x) \in \mathbb{Z}[x]$, the algebraic number $\phi(\alpha)$ (resp. $\phi(\beta)$) in K_f (resp. K_g) is B-smooth if $\mathrm{Res}(f, \phi)$ (resp. $\mathrm{Res}(g, \phi)$) is the product of a B-smooth integer and of a product of factors of $l(f)$ (resp. $l(g)$). In that case, due to the equation

$$N\left(\phi(\alpha)\right) = \pm l(f)^{-\deg \phi} \mathrm{Res}(f, \phi),$$

the fractional ideal $\phi(\alpha)\mathcal{O}_f$ decomposes into a product of ideals of $\mathcal{F}_f$, with positive or negative exponents.

In the sieve stage, one collects $\#\mathcal{F}_f + \#\mathcal{F}_g$ polynomials $\phi(x) \in \mathbb{Z}[x]$ with coprime coefficients and degree at most $t - 1$, for a parameter $t \geq 2$ to be

chosen, such that $\mathrm{Res}(f,\phi)$ and $\mathrm{Res}(g,\phi)$ are B-smooth. Since both $\phi(\alpha)$ and $\phi(\beta)$ are B-smooth, we get *relations* of the form:

$$\begin{cases} \phi(\alpha)\mathcal{O}_f = \prod_{\mathfrak{q}\in\mathcal{F}_f} \mathfrak{q}^{\mathrm{val}_\mathfrak{q}(\phi(\alpha))} \\ \phi(\beta)\mathcal{O}_g = \prod_{\mathfrak{r}\in\mathcal{F}_g} \mathfrak{r}^{\mathrm{val}_\mathfrak{r}(\phi(\beta))}. \end{cases}$$

Each relation allows us to write a linear equation among the (virtual) logarithms of the ideals in the factor base. In this article we hush up the technical details related to virtual logarithms and Schirokauer maps, but the reader can find a detailed description in [5,16,31].

The norm of $\phi(\alpha)$ (resp. of $\phi(\beta)$) is the product of the norms of the ideals in the right hand side and will be bounded by the size of the finite field $(p^n)^{\max(\deg f,\deg g)}$ (this is a very crude estimate; refining it is the heart of the complexity analysis of Section 4); therefore the number of ideals involved in a relation is less than $\log_2(p^n)^{O(1)}$. One can also remark that the ideals that can occur in a relation have degrees that are at most equal to the degree of ϕ, that is $t-1$. Therefore, it makes sense to include in $\mathcal{F}_f$ and $\mathcal{F}_g$ only the ideals of degree at most $t-1$ (for a theoretical analysis of NFS one can maintain the variant without restrictions on the degree of the ideals in the factor base).

In order to estimate the probability that a random polynomial ϕ with given degree and size of coefficients gives a relation, we make the common heuristic that the integer $\mathrm{Res}(\phi,f)\cdot\mathrm{Res}(\phi,g)$ has the same probability of B-smoothness as a random integer of the same size. Therefore, reducing the expected size of this product of norms is the main criterion when selecting the polynomials f and g.

In the *linear algebra* stage, we consider the homogeneous linear system obtained after the sieve. We make the usual heuristic that this system has a space of solutions of dimension one. Since the system is sparse (at most $(\log_2(p^n))^{O(1)}$ non-zero entries per row), an iterative algorithm like (block) Wiedemann [10,33] is used to compute a non-zero solution in quasi-quadratic time. This gives the (virtual) logarithms of all the factor base elements.

In principle, the coefficient ring of the matrix is $\mathbb{Z}/(p^n-1)\mathbb{Z}$, but it is enough to solve it modulo each prime divisor ℓ of p^n-1 and then to recombine the results using the Pohlig–Hellman algorithm [27]. Since one can use Pollard's method [28] for small primes ℓ, we can suppose that ℓ is larger than $L_{p^n}(1/3)$. Since ℓ is large, we may assume that ℓ is coprime to $\mathrm{Disc}(f)$, $\mathrm{Disc}(g)$, the class numbers of K_f and K_g, and the orders of the roots of unity in K_f and K_g. These assumptions greatly simplify the theory, but again, we do not elaborate on the mathematical aspects of the algorithm since the improvements that we discuss in this article do not affect them.

In the last stage of the algorithm, called *individual logarithm*, the discrete logarithm of any element $z=\sum_{i=0}^{n-1} z_i m^i$ of $\mathbb{F}_{p^n}$ in the finite field is computed. For this, we associate z with the algebraic number $\overline{z}=\sum_{i=0}^{n-1} z_i\alpha^i$ in K_f and check whether the corresponding principal ideal factors into prime ideals of norms bounded by a second bound B' larger than B. We also ask the prime ideals to

be of degree at most $t-1$. If $\overline{z}$ does not satisfy these smoothness assumptions, then we replace z by z^e for a randomly chosen integer e and try again. This allows us to obtain a linear equation similar to those of the linear system, in which one of the unknowns is $\log z$. The second step of the individual logarithm stage consists in obtaining relations between a prime ideal and prime ideals of smaller norm, until all the ideals involved are in $\mathcal{F}_f$ or $\mathcal{F}_g$. This allows us to backtrack and obtain $\log z$.

3 New Methods for Selecting Polynomials

In this section, we propose two new methods to select the polynomials f and g, in the case of finite fields that are low degree extensions of prime fields. For any polynomial g, we denote by $||g||_\infty$ the maximum absolute value of its coefficients. The first method is an extension to non-prime fields of the method used by Joux and Lercier [15] for $\mathbb{F}_p$. In the second one, we use rational reconstruction and the existence of some square roots in $\mathbb{F}_p$.

All the constructions that follow use either LLL reduction [22] or simple rational reconstruction (also known as the continued fraction method), see for example [8]. It allows us to write any residue a modulo a prime p as

$$a \equiv u/v \mod p$$

with $u, v \leq c\sqrt{p}$ for some constant c. If one computes the rational reconstruction using LLL in dimension two, then one can show that it always succeeds if c is a large enough explicit constant. In practice we might want two different rational reconstructions $a \equiv u_1/v_1 \equiv u_2/v_2 \mod p$. In this case we cannot make any proof on the size of u_2 and v_2, but they can be small.

For ease of reading, f is supposed to have degree greater than or equal to that of g. Although the polynomial $\varphi(x)$ that defines $\mathbb{F}_{p^n}$ as $\mathbb{F}_p[X]/\langle\varphi(x)\rangle$ does not occur explicitly in the computations, we sometimes give it along with the pair (f, g).

3.1 State of the Art

Joux, Lercier, Smart and Vercauteren [16] introduced two methods of polynomial selection in non-prime fields which are the only option for their respective range of application: medium characteristic and, respectively, large characteristic finite fields.

JLSV$_1$. In Algorithm 1 we recall the method introduced in [16, §2.3]. This method is best suited to the medium characteristic case. It produces two polynomials f and g of the same degree n, which have coefficients of size $O(\sqrt{p})$ each.

Example 1. Take $p = 1000001447$, $n = 4$, and $a = 44723 \geq \lceil\sqrt{p}\rceil$. One has $f = (x^4 - 6x^2 + 1) - 44723(x^3 - x)$ and $g = 22360(x^4 - 6x^2 + 1) - 4833(x^3 - x)$ with $u/v = 4833/22360$ a rational reconstruction of a modulo p.

Algorithm 1. Polynomial selection with the first method in [16] (JLSV_1)

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 $a \geq \lceil \sqrt{p} \rceil$;
until $f = f_0 + af_1$ *is irreducible in* $\mathbb{F}_p[x]$;
$(u, v) \leftarrow$ a rational reconstruction of a modulo p ;
$g \leftarrow vf_0 + uf_1$;
return $(f, g, \varphi = g \bmod p)$

JLSV_2. In Algorithm 2 we reproduce the method described in [16, §3.2]. We denote by $\mathrm{LLL}(M)$ the matrix obtained by applying the LLL algorithm to the rows of a matrix M with integer coefficients.

Algorithm 2. Polynomial selection with the second method in [16] (JLSV_2)

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 some 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

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

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

Proposition 1. *The coefficients of the polynomial f in Algorithm 2 have approximate size $Q^{1/(D+1)}$.*

Proof. By construction, $|c_0| \approx W^n \approx Q^{1/(D+1)}$. Since c was chose so that $c_n = 1$, we get $\det(M) = p^n$. The first row of $\mathrm{LLL}(M)$ gives a polynomial f of degree at most D which is divisible by g modulo p. The coefficients of f are of approximate size $(\det M)^{1/(D+1)}$. It is $= p^{n/(D+1)} = Q^{1/(D+1)}$ if we assume that the dimension $D + 1$ stays small.

We can have $\deg(f) = D$ for any value of $D \geq n$. We may want to take $D = n$ for some real-life cases, so that f and g are of degree n; moreover we take $\varphi \equiv g \bmod p$. We give such an example here.

Example 2. Consider again the case of $p = 1000001447$ and $n = 4$ this time. We take g_0 to be a polynomial of degree four and small coefficients, for example $g_0 = x^4 + x^3 + x^2 + x + 1$. We use $W_0 = 64 \geq p^{1/(D+1)}$ and we set

$$g = g_0(x + W_0) = x^4 + 257x^3 + 24769x^2 + 1060993x + 17043521.$$

We construct the lattice of integer polynomials of degree at most 4 which are divisible by g modulo p. Since $D = n$ we are here in a particular case where the lattice corresponds to the line of multiples of g by elements of $\mathbb{F}_p$. We obtain

$$f = 7791770x^4 + 2481996x^3 - 5928141x^2 + 1465261x + 3462017\ .$$

Note that f and g have coefficients of size $p^{4/5} = Q^{1/5}$.

3.2 The Generalized Joux–Lercier (GJL) Method[1]

Joux and Lercier proposed a method in [15] to select polynomials in the context of prime fields. In Algorithm 3 we generalize their method so that it applies to any finite field.

Algorithm 3. 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

1 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 ;
2 Reduce the following matrix using LLL

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

3 **return** $(f, g = g_0 + g_1 x + \cdots + g_d x^d, \varphi)$

[1] Recently, (February 2015), D. Matyukhin informed us that he proposed the same polynomial selection method for his algorithm of discrete logarithm [24], published in Russian.

In the prime field case, where $n = 1$, GJL goes as follows. One starts with a polynomial f of degree $d + 1$ with small coefficients such that f admits a factor φ of degree one, or equivalently a root m modulo p. Then, a matrix M is constructed whose rows correspond to the lattice of polynomials in $\mathbb{Z}[x]$ of degree at most d, that also admit m as a root modulo p. We reduce this matrix with LLL and obtain g from the first row of LLL(M). Note that one can transform M into M' in Equation (1) by linear combinations on the rows. Hence, LLL(M) = LLL(M'), so GJL and the original method of Joux–Lercier obtain the same polynomial g.

$$M = \begin{bmatrix} p & 0 & \cdots & 0 \\ -m & 1 & 0 & 0 \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & -m & 1 \end{bmatrix}, \quad M' = \begin{bmatrix} p & 0 & \cdots & 0 \\ -m & 1 & 0 & 0 \\ \vdots & \ddots & \ddots & 0 \\ -m^d & \cdots & 0 & 1 \end{bmatrix}, \tag{1}$$

The following result is proved in the same way as Proposition 1.

Proposition 2. *The coefficients of the polynomial g in Algorithm 3 have approximate size $Q^{1/(d+1)}$.*

Remark 1. By considering a smaller matrix, it is possible to produce a polynomial g whose degree is smaller than $d = \deg f - 1$. This does not seem to be a good idea. Indeed, the size of the coefficients of g would be the same as the coefficients of a polynomial obtained starting with a polynomial f with coefficients of the same size but of a smaller degree (d or less).

Example 3. We keep $p = 1000001447$ and $n = 4$ (see Ex. 2). We choose $d = n = 4$, $f = x^5 + x^4 - 7x^3 + 4x^2 + x + 1$ of degree 5. Then f factors modulo p and has a degree four factor $\varphi = x^4 + 234892989x^3 + 208762833x^2 + 670387270x + 109760434$. We construct the lattice of polynomials of degree at most 4 which are divisible by φ modulo p. Reducing it, we obtain

$$g = 8117108x^4 + 1234709x^3 + 9836008x^2 - 1577146x + 7720480 \ .$$

The polynomial f has coefficients of size $O(1)$; g of size $O(p^{4/5})$. More precisely, $\log_2 p = 30$ and $\log_2 ||g||_\infty = 24 = 4/5 \log_2 p$.

3.3 The Conjugation Method

Roughly speaking, the aim of polynomial selection is to produce two polynomials f, g such that the norm of f, resp. g evaluated at some algebraic integer $a - b\alpha$ (or $\phi(\alpha)$) is minimal, with respect to the size of a and b (see Sec. 2). This means, the degrees of f and g need to stay small but also the size of their coefficients. The previous method GJL was focused on minimizing the degrees of both f and g to e.g. n and $n+1$. Here we set f with a higher degree: $2n$ instead of $n+1$, so that the size of the coefficients of g is always $O(\sqrt{p})$. We cannot achieve a so small

coefficient size for g with the GJL method. We show in Sec. 4.4 that despite the higher degree of f, this method is better in practice than any other one when $n = 2$, 3 and p is more than 660 bits, and in particular for cryptographic sizes.

Let us give the idea of our method. We first carefully select f of degree $2n$, irreducible over $\mathbb{Z}$, and so that it factors into two irreducible conjugate polynomials φ and $\overline{\varphi}$ over some quadratic extension of $\mathbb{Q}$. If we can embed this quadratic extension in $\mathbb{F}_p$, we end up with two irreducible factors of f modulo p. Because of our judicious choice of f with two conjugate factors, we obtain g whose coefficients have size $O(\sqrt{p})$ using rational reconstruction. We give a first example in Ex. 4 to clarify what we have in mind.

We start with two examples, and then give the general construction.

Example 4. We target $\mathbb{F}_{p^4}$ with $p = 1000010633$ and $n = 4$. We try integers $a = -2, -3, \ldots$ until a is a square but not a fourth power in $\mathbb{F}_p$. We find $a = -9$. We set $f = x^8 - a = x^8 + 9$ which is irreducible over $\mathbb{Z}$. Observe that by construction, f has two degree 4 conjugate irreducible factors $f = (x^4 - \sqrt{a})(x^4 + \sqrt{a})$. We set $\varphi = x^4 - \sqrt{a}$ which, due to the choice of a, belongs to $\mathbb{F}_p[x]$ and is irreducible. We continue by computing a rational reconstruction (u, v) of $\sqrt{a}$ modulo p: $u \cdot v^{-1} \equiv \sqrt{a} \mod p$; here $u = -58281$ and $v = 24952$. Finally we set $g = vx^4 - u = 24952x^4 + 58281$ of norm $||g||_\infty \sim \sqrt{p}$. Note that we respect the condition $\varphi = \gcd(f \mod p, g \mod p)$.

In the previous example, the prime p was given and we searched for a parameter a, or equivalently for a polynomial $f = x^8 - a$. It turns out that some polynomials f have very good sieving properties and we would like to use them for many primes p. In this case, we can reverse the process and start with a given f, while expecting to succeed in only a fraction of the cases.

Example 5. We want to use $f = x^4 + 1$. We target $\mathbb{F}_{p^2}$ for $p \equiv 7 \mod 8$ prime. Note that $f(x) = (x^2 + \sqrt{2}x + 1)(x^2 - \sqrt{2}x + 1)$ over $\mathbb{Q}(\sqrt{2})$. Since 2 is a square modulo p, we take $\varphi = x^2 + \sqrt{2}x + 1 \in \mathbb{F}_p[x]$ and note that φ is a factor of f modulo p. Now, by rational reconstruction of $\sqrt{2}$ in $\mathbb{F}_p$, we can obtain two integers $u, v \in \mathbb{Z}$ such that $\frac{u}{v} \equiv \sqrt{2} \mod p$, and u and v have size similar to $\sqrt{p}$. We define $g = vx^2 + ux + v$. Then f and g share a common irreducible factor of degree 2 modulo p, and satisfy the degree and size properties given in Prop. 3.

Although the technique in the second example is more interesting in practice, it is the construction in the first example that can be made general, as given in Algorithm 4. Under reasonable assumptions, this algorithm terminates and finds pairs of polynomials f and g with the claimed degree and size properties for any extension field $\mathbb{F}_{p^n}$.

Proposition 3. *The polynomials (f, g) returned by Algorithm 4 satisfy the following properties*

1. *f and g have integer coefficients and degrees $2n$ and n respectively;*
2. *the coefficients of f have size $O(1)$ and the coefficients of g are bounded by $O(\sqrt{p})$;*

Algorithm 4. Polynomial selection with the Conjugation method (Conj)

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 $\mu(x)$ a quadratic, monic, irreducible polynomial over $\mathbb{Z}$ with small coefficients ;
until $\mu(x)$ *has a root* λ *in* $\mathbb{F}_p$ *and* $\varphi = g_0 + \lambda g_1$ *is irreducible in* $\mathbb{F}_p[x]$;
$(u, v) \leftarrow$ a rational reconstruction of λ ;
$f \leftarrow \mathrm{Res}_Y(\mu(Y), g_0(x) + Y g_1(x))$;
$g \leftarrow v g_0 + u g_1$;
return (f, g, φ)

3. f and g have a common irreducible factor φ of degree n over $\mathbb{F}_p$.

Proof. The polynomial f is the resultant of two bivariate polynomials with integer coefficients. Using classical properties of the resultant, f can be seen as the product of the polynomial $g_0(x) + Y g_1(x)$ evaluated in Y at the two roots of $\mu(Y)$, therefore its degree is $2n$. Since all the coefficients of the polynomials involved in the definition of f have size $O(1)$, and the degree n is assumed to be "small", then the coefficients of f are also $O(1)$. For the size of the coefficients of g, it follows from the output of the rational reconstruction of λ in $\mathbb{F}_p$, which is expected to have sizes in $O(\sqrt{p})$. The polynomials f and g are suitable for NFS in $\mathbb{F}_{p^n}$, because both are divisible by $\varphi = g_0 + \lambda g_1$ modulo p, and by construction φ is irreducible of degree n.

In the example above (Ex. 5), for $\mathbb{F}_{p^2}$ with $p \equiv 7 \bmod 8$, Algorithm 4 was applied with $g_1 = x$, $g_0 = x^2 + 1$ and $\mu = x^2 - 2$. One can check that $f = \mathrm{Res}_Y(Y^2 - 2, (x^2 + 1) + Yx) = x^4 + 1$.

In the following section, an asymptotic analysis shows that there are cases where this Conjugation method is more interesting than JLSV_1 and that GJL is competitive with JLSV_2; furthermore, the new methods are also well-suited for small degree extensions that can be reached with current implementations.

4 Complexity Analysis

In this section we prove Theorem 1 that we stated in the Introduction.

4.1 Preliminaries

As introduced in Section 2, the parameter t denotes the number of terms of the polynomials in the sieving domain, i.e. $\deg \phi = t - 1$, and B is the smoothness bound. We call E the square root of the number of sieved polynomials, i.e. the

coefficients of ϕ belong to the interval $[-E^{2/t}, E^{2/t}]$. Kalkbrener's bound [19, Corollary 2] states that, for any polynomials f and ϕ,

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

where $\kappa(n,m) = \binom{n+m}{n}\binom{n+m-1}{n}$. For two polynomials f and g we write $\kappa(f,g)$ for $\kappa(\deg f, \deg g)$. Hence, we obtain a bound on the product of the norms:

$$|\operatorname{Res}(f,\phi)\operatorname{Res}(g,\phi)| \leq \kappa(f,\phi)\kappa(g,\phi)||\phi||_\infty^{\deg f+\deg g}(||f||_\infty||g||_\infty)^{\deg \phi}. \quad (2)$$

A simple upper bound for $\kappa(n,m)$ is $(n+m)!$:

$$\begin{aligned}|\operatorname{Res}(f,\phi)\operatorname{Res}(g,\phi)| \leq \\ &\leq (\deg f + \deg\phi)!(\deg g + \deg\phi)!(||f||_\infty||g||_\infty)^t||\phi||_\infty^{\deg f+\deg g} \\ &\leq (\deg f + t - 1)!(\deg g + t - 1)!(||f||_\infty||g||_\infty)^{t-1}E^{2(\deg f+\deg g)/t}.\end{aligned}$$

In what follows we respect make sure that the degrees of our polynomials satisfy: $\deg f + \deg g + t = O(1)\max\left((\log Q)^{1/3}, n\right)$. Writing $p = L_Q(\alpha)$ with $\alpha > 1/3$, we obtain $n = O(1)(\log Q)^{1-\alpha}/(\log\log Q)^{1-\alpha}$. Then, we have $(\deg f + t-1)!(\deg g+t-1)! = L_Q(\max(1-\alpha, 1/3))$, whereas our estimates for the right hand side will have a size $L_Q(2/3)$. This allows us to use the estimation

$$\max_{\phi \text{ in sieving domain}} |\operatorname{Res}(f,\phi)\operatorname{Res}(g,\phi)| \approx (||f||_\infty||g||_\infty)^{t-1}E^{2(\deg f+\deg g)/t}. \quad (3)$$

Since the number of sieved polynomials is E^2, the cost of the sieve is $E^{2+o(1)}$. Independently of the choice of the polynomials f and g, the cardinality of the factor base is $B^{1+o(1)}$, and using the (block) Wiedemann algorithm [10,33], the cost of the linear algebra is $B^{2+o(1)}$. Hence, we set $E = B$ and, since we expect an algorithm of complexity $L_Q(1/3)$, the two are equal to $L_Q(1/3, \beta)$

$$E = B = L_Q(1/3, \beta)$$

for a constant β to be found.

We make the common heuristic that the product of the resultants of the polynomials ϕ in the sieving domain with f and g has the same probability to be B-smooth as a random integer of the same size; we denote this probability by $\mathcal{P}$. Since the cost of the sieve is $B\mathcal{P}^{-1}$ and, at the same time $E^{2+o(1)}$, we find the equation

$$B = \mathcal{P}^{-1}. \quad (4)$$

4.2 The Generalized Joux–Lercier Method

We are now ready to prove the first part of Theorem 1. Using GJL, one constructs two polynomials f and g such that, for a parameter $d \geq n$, we have $\deg f = d+1$, $\deg g = d$, $||g||_\infty \approx Q^{1/(d+1)}$ and $||f||_\infty$ of size $O(1)$.

The GJL polynomials have the same degree and coefficient size as those obtained in [15] for prime fields. Hence, we make the same choices for parameter

t, i.e. we sieve on linear polynomials. Also, we take d of roughly the same size, i.e. $d = \delta\left((\log Q)/(\log\log Q)\right)^{1/3}$, which we inject in Equation (3):

$$\begin{aligned}\max_\phi\left(|\operatorname{Res}(f,\phi)\operatorname{Res}(g,\phi)|\right) &\approx ||f||_\infty ||g||_\infty E^{\deg f+\deg g},\\ &\approx Q^{1/(d+1)}E^{2d+1}.\end{aligned}$$

With the L-notation, we obtain

$$|\operatorname{Res}(f,\phi)\operatorname{Res}(g,\phi)| \leq L_Q\left(2/3, \delta\beta + \frac{2}{\delta}\right).$$

Using the Canfield–Erdös–Pomerance theorem [7], we obtain

$$\mathcal{P} = 1/L_Q\left(1/3, \frac{\delta}{3} + \frac{2}{3\beta\delta}\right).$$

The equality $\mathcal{P}^{-1} = B$ imposes

$$\beta = \frac{\delta}{3} + \frac{2}{3\beta\delta}.$$

The optimal value of δ is the one which minimizes the expression on the right hand side, so we take $\delta = \sqrt{2/\beta}$ and we obtain $\beta = 2/3\sqrt{2/\beta}$, or equivalently $\beta = \sqrt[3]{8/9}$. Since the complexity of NFS is $(E^2 + B^2)^{1+o(1)} = L_Q(1/3, 2\beta)$, we obtain the complexity given in Theorem 1.

The range of application of GJL is determined by the condition $n \leq d$. This is true for all fields of large characteristic. In the boundary case, since $d = \delta/2\left(\frac{\log Q}{\log\log Q}\right)^{1/3}$ with $\delta = \sqrt{2/\beta} = \sqrt[3]{3}$, the GJL method applies only to those fields $\mathbb{F}_{p^n}$ such that

$$p \geq L_Q\left(2/3, \sqrt[3]{8/3}\right).$$

This concludes the proof of the first part of Theorem 1.

4.3 The Conjugation Method

Recall that the Conjugation method allows us to construct two polynomials f and g such that $\deg f = 2n$, $\deg g = n$, $||g||_\infty \approx Q^{1/(2n)}$ and $||f||_\infty = O(1)$. When introduced in Equation (3), these values give

$$|\operatorname{Res}(\phi, f)\operatorname{Res}(\phi, g)| \leq \left(E^{6n/t}Q^{(t-1)/2n}\right)^{1+o(1)}. \tag{5}$$

We study first the case of medium characteristic and then the boundary case between medium and large characteristic.

The Case of Medium Characteristic. The Conj polynomials are similar to the ones in [16] obtained using JLSV_1, so we choose parameters of the same type as in [16] and set

$$t = c_t n \left(\frac{\log Q}{\log\log Q}\right)^{-1/3}.$$

Using $E = B = L_Q(1/3, \beta)$ in Equation (5), the product of the two resultants has norm $L_Q(2/3, 6\beta/c_t + c_t/2)$. Due to the Canfield–Erdös–Pomerance theorem, the probability that a polynomial ϕ in the sieving domain has B-smooth resultants is

$$\mathcal{P} = 1/L_Q\left(1/3, \frac{2}{c_t} + \frac{c_t}{6\beta}\right).$$

We choose $c_t = 2\sqrt{3\beta}$ in order to optimize this probability:

$$\mathcal{P} = 1/L_Q\left(1/3, 2/\sqrt{3\beta}\right).$$

From the condition $\mathcal{P}^{-1} = B$, we have $\beta = \sqrt[3]{4/3}$, and we obtain the second result in Theorem 1.

The Boundary Case. For every constant $c_p > 0$, we consider the family of finite fields $\mathbb{F}_{p^n}$ such that

$$p = L_{p^n}(2/3, c_p)^{1+o(1)}.$$

We will take parameter t (Section 2) to be a constant in this analysis. Then the probability that a polynomial ϕ in the sieving domain has B-smooth resultants is

$$\mathcal{P} = 1/L_Q\left(1/3, \frac{2}{c_p t} + \frac{c_p(t-1)}{6\beta}\right).$$

The condition $\mathcal{P}^{-1} = B$ leads to $\frac{2}{c_p t} + \frac{c_p(t-1)}{6\beta} = \beta$, or equivalently to $\beta = \frac{1}{c_p t} + \sqrt{\frac{1}{(c_p t)^2} + \frac{1}{6}c_p(t-1)}$.

This completes the proof of the following result.

Proposition 4. *When* $\log Q$ *goes to infinity and* p *satisfies the condition* $p = L_Q(2/3, c_p)$*, the complexity of NFS with the Conjugation method is:*

$$L_Q\left(1/3, \frac{2}{c_p t} + \sqrt{\frac{4}{(c_p t)^2} + \frac{2}{3}c_p(t-1)}\right).$$

In Figure 1 we have plotted the complexities of Proposition 4, together with GJL and the Multiple number field sieve variant of [3].[2] There are some ranges of the parameter c_p where the Conjugation method is the fastest and a range where the GJL method is optimal. The best case for NFS with Conjugation polynomials corresponds to $c_p = 12^{1/3} \approx 2.29$ and $t = 2$. In this case we get the third result in Theorem 1.

[2] Thanks to Cécile Pierrot who pointed to us that this figure was inexact in an earlier variant of the manuscript.

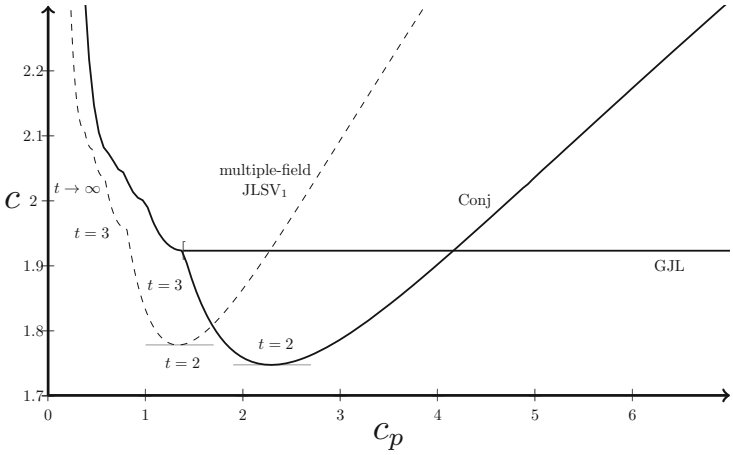

Fig. 1. The complexity of NFS for fields $\mathbb{F}_{p^n}$ with $p = L_{p^n}(2/3, c_p)$ is $L_{p^n}(1/3, c)$

4.4 Non-asymptotic Comparisons for Small Extension Degrees

Let us make a practical (as opposed to asymptotic) analysis of the four methods in our arsenal:

- the methods of Joux, Lercier, Smart and Vercauteren: JLSV$_1$ and JLSV$_2$.
- the generalized Joux–Lercier method, GJL; to indicate the value of the parameter d we sometimes write GJL-$(d+1, d)$.
- the Conjugation method, Conj.

We do not include Schirokauer's variant in our study since it is very different in nature, requiring to sieve on polynomials with coefficients in small degree extensions of $\mathbb{Q}$.

The complexity analysis that was presented earlier in this section gives hints, but does not allow us to choose the best method. For example, if one wants to compute discrete logarithms in finite fields $\mathbb{F}_Q$ of constant bitsize, i.e. $\log_2 Q \approx$ const, then JLSV$_1$ and Conj are competitive when p is smaller (medium prime case), whereas JLSV$_2$ and GJL are better when p is larger (large characteristic case). Also, we expect the choice $t = 2$ to be optimal when p is large, whereas we might consider sieving on non-linear polynomials, i.e. $t \geq 3$, for smaller values of p.

Table 1 summarizes the properties of the polynomials obtained with each method.

Although JLSV$_2$ was the state-of-art for the non-prime fields of large characteristic, it is now beaten either by GJL or by JLSV$_1$:

Table 1. Summary of the sizes of the norms product corresponding to various methods. Here $\mathbb{F}_Q$, $Q = p^n$, is the target field, d and D are parameters in the polynomial selection stage and E is the sieve parameter.

method	$\deg f$	$\deg g$	$\|\|f\|\|_\infty$	$\|\|g\|\|_\infty$	product of norms
Conj	$2n$	n	$O(1)$	$Q^{1/(2n)}$	$E^{6n/t}Q^{(t-1)/(2n)}$
GJL	$d+1$	$d \geq n$	$O(1)$	$Q^{1/(d+1)}$	$E^{2(2d+1)/t}Q^{(t-1)/(d+1)}$
JLSV_1	n	n	$Q^{1/(2n)}$	$Q^{1/(2n)}$	$E^{4n/t}Q^{(t-1)/n}$
JLSV_2	$D \geq n$	n	$Q^{1/(D+1)}$	$Q^{1/(D+1)}$	$E^{2(D+n)/t}Q^{2(t-1)/(D+1)}$

Proposition 5. *Let n, t, D be integers with $n, t \geq 2$ and $D \geq n$, and let E and Q be positive real numbers. Then the quantity $(E^{n+D})^{2/t}(Q^{2/(D+1)})^{t-1}$ is either larger than $(E^{2n})^{2/t}(Q^{1/n})^{t-1}$ or we can select $d \geq n$ so that it is larger than $(E^{2d+1})^{2/t}(Q^{1/(d+1)})^{t-1}$.*

Proof. **Case $D \geq 2n$.** We set $d = \lfloor D/2 \rfloor$ and we use GJL with this parameter. On the one hand we have $2(\lfloor D/2 \rfloor + 1) \geq D + 1$, so $(Q^{2/(D+1)})^{t-1} \geq (Q^{1/(d+1)})^{t-1}$. On the other hand, we have $n+D \geq 1+(D+1) \geq 1+(2\lfloor D/2 \rfloor+1)$, so $(E^{n+D})^{2/t} \geq (E^{2d+1})^{2/t}$.

Case $n \leq D \leq 2n-1$. On the one hand we have $D+n \geq 2n$, so $(E^{n+D})^{2/t} \geq (E^{2n})^{2/t}$. On the other hand, $2/(D+1) \geq 1/n$, so $(Q^{2/(D+1)})^{t-1} \geq (Q^{1/n})^{t-1}$.

In order to compare the remaining candidates we need to plug numerical values into Equation (2). The parameter E is hard to determine, and depends on the polynomials which are used. For example, better polynomials allow us to sieve less and hence to use a smaller value of E. Luckily, the difference between the various values of E are not very large and, when one method is considerably better than another, an approximate value of E is enough for the comparison. Another bias we are aware of is that for norm products of similar sizes, a method that provides norms that are well-balanced should be better than if the norm on one side is much larger than the norm on the other side. Therefore, when the differences between methods are small, we cannot decide by looking only at the size of the norm product. Table 2 lists the values of E with respect to the cardinality Q of the target field, obtained from the default parameters of the CADO factoring software [1] up to $\log_{10} Q = 220$ and extrapolated afterwards. But these values should not be taken too seriously in order to derive security parameters. The goal is only to investigate the relative performances of the various methods of polynomial selection for sizes where it is currently too costly to do experiments.

We considered several values of parameters d and $t = \deg \phi + 1$. As the asymptotic analysis predicts for the case of large characteristic (small degree), our results showed that the choice $t = 2$ is optimal for $n = 2, 3, 4, 5$. Since, in addition, the comparison goes in a similar manner for each value of parameter t, we focus on the case $t = 2$. We make an exception to this rule for $n = 6$, where the choice $t = 3$ is optimal for some ranges of $\log_{10} Q$.

Table 2. Practical values of E for Q from 100 to 300 decimal digits(dd)

Q(dd)	100	120	140	160	180	200	220	240	260	280	300
Q(bits)	333	399	466	532	598	665	731	798	864	931	997
E(bits)	20.9	22.7	24.3	25.8	27.2	28.5	29.7	30.9	31.9	33.0	34.0

In Table 3 we expand the formula of Equation (3) for $t = 2$ and $n = 2, 3, 4, 5, 6$. Note also that we list several values for the parameter d of GJL since it cannot be fixed by asymptotic arguments. Using the remark that the quotient $\log Q / \log E$ belongs to the interval $[15, 30]$, we discard some choices, and mark them with an $\otimes$. For example, JLSV$_1$ beats GJL-$(3, 2)$ only when $\log Q / \log E \leq 6$, which is outside of our range of interest.

4.5 Final Results

The Case $n = 2$. We draw the curves corresponding to the results in Table 3, since $t = 2$ is optimal in this case. From Figure 2, the best choice is to use Conj polynomials.

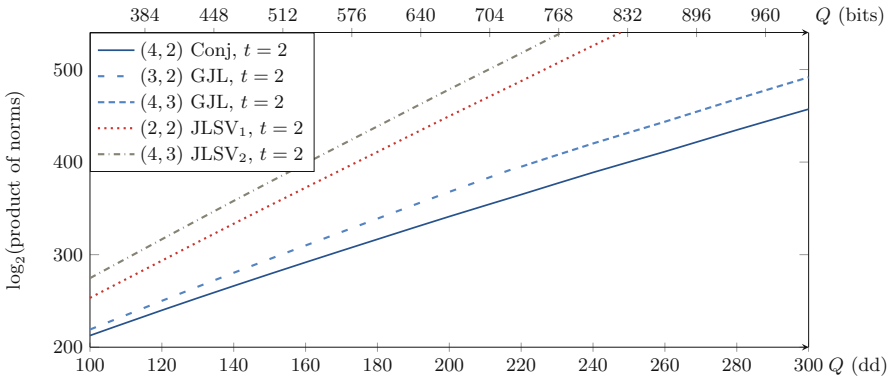

Fig. 2. Polynomials for $\mathbb{F}_{p^2}$

The Case $n = 3$. Again, it is optimal to sieve on linear polynomials. For smaller values, i.e. Q of less than 220 decimal digits, we use GJL-$(4, 3)$ polynomials, whereas for larger fields we switch to Conj, as exemplified in Figure 3.

The Case $n = 4$. We sieve on linear polynomials as before ($t = 2$). Here JLSV$_1$ and GJL$-(5, 4)$ offer similar polynomials, see Figure 4.

The Case $n = 5$. Several methods give polynomials with a norm product of roughly the same size: Conj with $t = 3$ and $t = 4$, GJL$-(6, 5)$ with $t = 2$ and $t = 3$ and, finally JLSV$_1$ with $t = 2$. All this is demonstrated in Figure 5.

Table 3. Size of the product of norms for various methods and associated parameters, when one sieves on linear polynomials ($t = 2$). We discard ($\otimes$) the methods which offer sizes of norms product which are clearly not competitive compared to some other one, assuming that $15 \leq \log Q / \log E \leq 30$ (Tab. 2).

n	method	$(\deg g, \deg f)$	$\|\|f\|\|_\infty$	$\|\|g\|\|_\infty$	$E^{\deg f+\deg g}\|\|f\|\|_\infty\|\|g\|\|_\infty$	discard
2	GJL $-(3,2)$	$(3,2)$	$O(1)$	$Q^{1/3}$	$E^5Q^{1/3}$	
	GJL $-(4,3)$	$(4,3)$		$Q^{1/4}$	$E^7Q^{1/4}$	$\otimes$
	Conj	$(4,2)$		$Q^{1/4}$	$E^6Q^{1/4}$	
	JLSV$_1$	$(2,2)$	$Q^{1/4}$	$Q^{1/4}$	$E^4Q^{1/2}$	$\otimes$

n	method	$(\deg f, \deg g)$	$\|\|f\|\|_\infty$	$\|\|g\|\|_\infty$	$E^{\deg f+\deg g}\|\|f\|\|_\infty\|\|g\|\|_\infty$	discard
3	GJL $-(4,3)$	$(4,3)$	$O(1)$	$Q^{1/4}$	$E^7Q^{1/4}$	
	GJL $-(5,4)$	$(5,4)$		$Q^{1/5}$	$E^9Q^{1/5}$	$\otimes$
	Conj	$(6,3)$		$Q^{1/6}$	$E^9Q^{1/6}$	
	JLSV$_1$	$(3,3)$	$Q^{1/6}$	$Q^{1/6}$	$E^6Q^{1/3}$	$\otimes$

n	method	$(\deg f, \deg g)$	$\|\|f\|\|_\infty$	$\|\|g\|\|_\infty$	$E^{\deg f+\deg g}\|\|f\|\|_\infty\|\|g\|\|_\infty$	discard
4	GJL $-(5,4)$	$(5,4)$	$O(1)$	$Q^{1/5}$	$E^9Q^{1/5}$	
	GJL $-(6,5)$	$(6,5)$		$Q^{1/6}$	$E^{11}Q^{1/6}$	$\otimes$
	Conj	$(8,4)$		$Q^{1/8}$	$E^{12}Q^{1/8}$	$\otimes$
	JLSV$_1$	$(4,4)$	$Q^{1/8}$	$Q^{1/8}$	$E^8Q^{1/4}$	

n	method	$(\deg f, \deg g)$	$\|\|f\|\|_\infty$	$\|\|g\|\|_\infty$	$E^{\deg f+\deg g}\|\|f\|\|_\infty\|\|g\|\|_\infty$	discard
5	GJL $-(6,5)$	$(6,5)$	$O(1)$	$Q^{1/6}$	$E^{11}Q^{1/6}$	
	GJL $-(7,6)$	$(7,6)$		$Q^{1/7}$	$E^{13}Q^{1/7}$	$\otimes$
	Conj	$(10,5)$		$Q^{1/10}$	$E^{15}Q^{1/10}$	$\otimes$
	JLSV$_1$	$(5,5)$	$Q^{1/10}$	$Q^{1/10}$	$E^{10}Q^{1/5}$	

n	method	$(\deg f, \deg g)$	$\|\|f\|\|_\infty$	$\|\|g\|\|_\infty$	$E^{\deg f+\deg g}\|\|f\|\|_\infty\|\|g\|\|_\infty$	discard
6	GJL $-(7,6)$	$(7,6)$	$O(1)$	$Q^{1/7}$	$E^{13}Q^{1/7}$	
	GJL $-(8,7)$	$(8,7)$		$Q^{1/8}$	$E^{15}Q^{1/8}$	$\otimes$
	Conj	$(12,6)$		$Q^{1/12}$	$E^{18}Q^{1/12}$	$\otimes$
	JLSV$_1$	$(6,6)$	$Q^{1/12}$	$Q^{1/12}$	$E^{12}Q^{1/6}$	

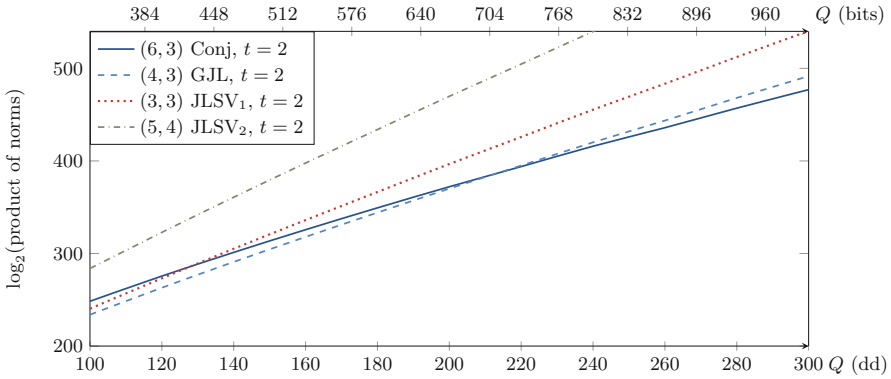

Fig. 3. Polynomials for $\mathbb{F}_{p^3}$

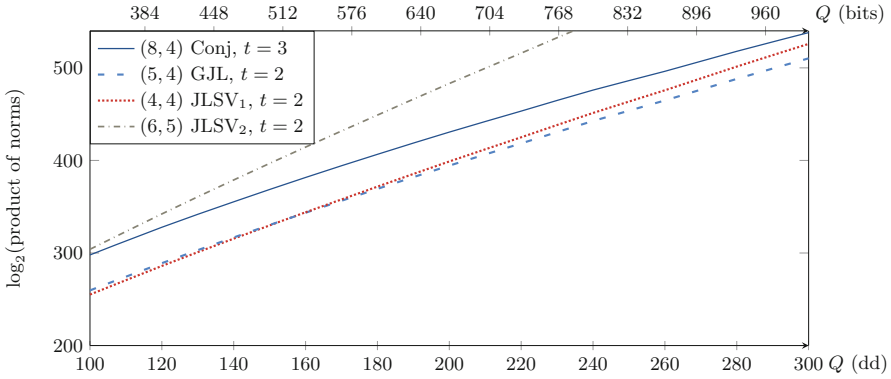

Fig. 4. Polynomials for $\mathbb{F}_{p^4}$

The Case $n = 6$. When Q is less than 180 decimal digits, the choice is between GJL $-(7, 6)$ with $t = 3$ and Conj with $t = 4$. When Q is larger than 260 decimal digits, the best two methods are Conj with $t = 4$ and JLSV$_1$ with $t = 2$. Between the two ranges, one needs to consider the three methods listed before. See Figure 6.

5 Additional Improvements

5.1 Improving the Root Properties

In both GJL and Conjugation methods, it is possible to obtain more than one polynomial g for a given f by taking another choice for the rational reconstruction, or another vector in the reduced lattice. Hence, we can assume that one has obtained two distinct reduced polynomials g_1 and $g_2 \in \mathbb{Z}[x]$ such that φ

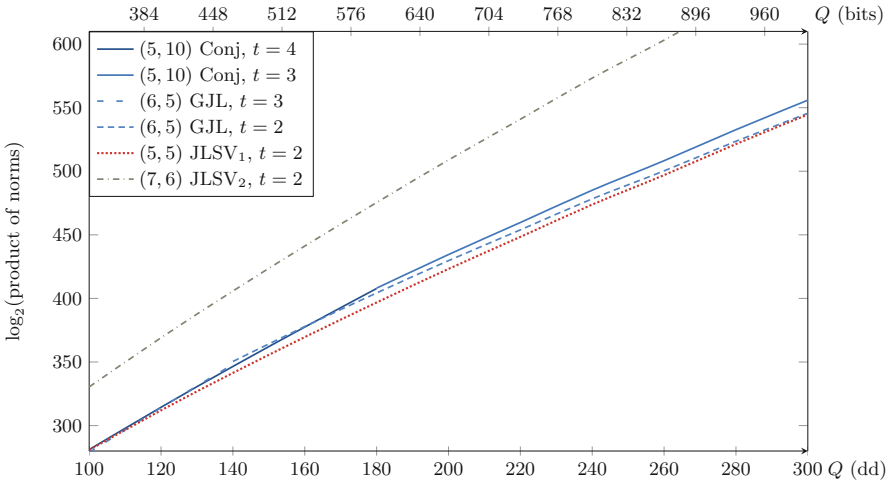

Fig. 5. Polynomials for $\mathbb{F}_{p^5}$

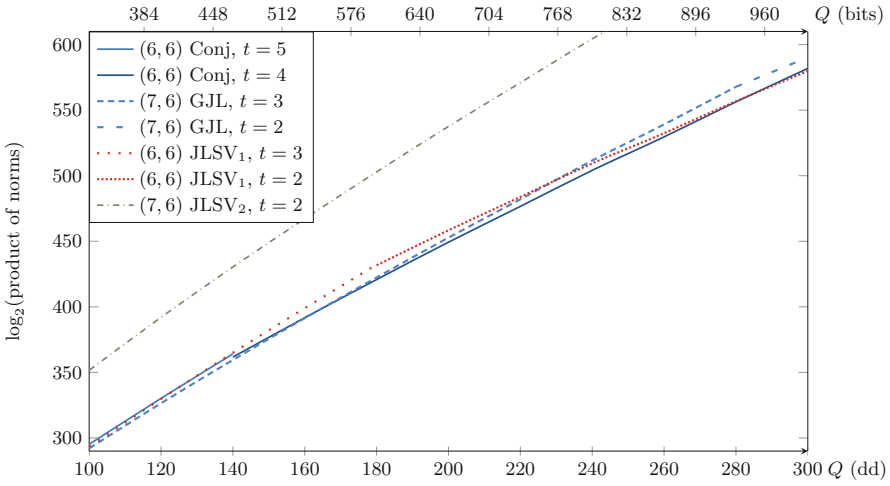

Fig. 6. Polynomials for $\mathbb{F}_{p^6}$

divides both g_1 and g_2 in $\mathbb{F}_p[x]$. Any linear combination $g = \lambda_1 g_1 + \lambda_2 g_2$ for small integers λ_1 and λ_2 is then suitable for running the NFS algorithm.

The Murphy $\mathbb{E}$ value as explained in [25, Sec. 5.2.1, Eq. 5.7 p. 86] is a good criterion to choose between all these g polynomials. In our experiments we searched for $g = \lambda_1 g_1 + \lambda_2 g_2$ with $|\lambda_i| < 200$ and such that $\mathbb{E}(f, g)$ is maximal. In practice we obtain g with $\alpha(g) \leq -1.5$ and $\mathbb{E}(f, g)$ improved by 2% up to 30 %.

5.2 Coppersmith's Multiple-field Variant

In [9], Coppersmith introduced a variant of NFS in which more than two polynomials are used. This can be applied to essentially all polynomial selection methods and in particular to the ones mentioned in this article. The base-m method, which applies only to prime fields, was analyzed by Matyukhin [23]. Recently, it was shown that JLSV_1 and JLSV_2 can successfully be adapted to use multiple fields, as demonstrated by [3].

Another important example is the Conjugation method. In Sec. 5.1, we noted that the same polynomial f can be paired with any of the two polynomials g_1 and g_2, and that we have $\gcd(f \bmod p, g_1 \bmod p, g_2 \bmod p) = \varphi$. This fact can be used to derive a multiple-field variant. It was remarked and analyzed in [26] and results in a complexity of $L_Q(1/3, c)$ with $c = \left(8(9+4\sqrt{6})/15\right)^{1/3} \approx 2.156$, in the medium characteristic case.

We have also analyzed a multiple-field variant of the Generalized Joux–Lercier method. This provides only a marginal improvement in the theoretical complexity and is not competitive with other methods [3], and therefore we do not include this analysis here.

5.3 Taking Advantage of Automorphisms

Joux, Lercier, Smart and Vercauteren [16, Section4.3] proposed to speed up computations in NFS using number field automorphisms. Given a field K and an irreducible polynomial $f \in K[x]$ without multiple roots, a K-automorphism is a rational fraction $A \in K(x)$ such that, for some rational fraction $D(x) \in K(x)$, we have $f(A(x)) = D(x)f(x)$. Using the language of Galois theory, a K-automorphism is an automorphism of the extension $\left(K[x]/\langle f\rangle\right)/K$. Hence, the automorphisms form a group whose order divides $\deg f$.

It is possible to push further the idea in [16] so that one can use automorphisms of both polynomials f and g. An example is given in our record computation described in Section 6 where we used two reciprocal polynomials: this saves a factor of two in the sieve and a factor of four in the linear algebra.

When a method to select polynomials gives the choice of the first polynomial f, we can select f in the family of our preference, making possible for example to have automorphisms. The only obstacle is that we cannot find polynomials with an automorphism of order n, as required by the results in [16] if $\deg f$ is not a multiple of n.

But in fact some methods allow us to have automorphisms for both polynomials. Indeed, the literature, e.g. [11], offers examples of polynomials $g_0, g_1 \in \mathbb{Q}[x]$ and rational fractions $A(x) \in \mathbb{Q}(x)$ such that, for any number field K and any parameter $a \in K$, the polynomial $g_0 + ag_1$ admits A as a K-automorphism:

$$\exists D_a(x) \in K(x), \qquad g_0(A(x)) + ag_1(A(x)) = D_a(x)\left(g_0(x) + ag_1(x)\right).$$

Example 6. For $g_0 = x^3 - 3x - 1$ and $g_1 = x^2 + x$, the rational fraction $A = -(1+1/x)$ is an automorphism, for any value of parameter a. Indeed, $g_0(A(x)) + ag_1(A(x))$ is $(x^3 + ax^2 + (a-3)x - 1 = (g_0(x) + ag_1(x))/x^3$, so $D(x) = 1/x^3$.

We do not study the question of finding such families. Let us instead make a list of the cases where one or both polynomials can admit automorphisms.

- JLSV$_1$ allows both polynomials to be in a family of type $g_0 + ag_1$, with g_0 and g_1 fixed; we can have automorphisms on both sides.
- JLSV$_2$ allows g to be selected with good properties; since $\deg g = n$, g can have $\mathbb{Q}$-automorphisms of order n.
- GJL allows f to be selected in the family of our choice; when $\deg f$ is divisible by n, f can have $\mathbb{Q}$-automorphisms of order n.
- Conj allows us to have $\mathbb{Q}$-automorphisms for both polynomials, for the values of n where families as above can be found. On the one hand, g is chosen in the family $\{g_0 + ag_1\}$, of automorphism $A(x)$. On the other hand, let ω be an algebraic number, root of an irreducible degree two polynomial $\mu \in \mathbb{Q}[x]$, such that $f = (g_0 + \omega g_1)(g_0 + \overline{\omega} g_1)$. Let $D_\omega \in \mathbb{Q}(\omega)(x)$ be such that $g_0(A(x)) + \omega g_1(A(x)) = D_\omega(x)(g_0(x) + \omega g_1(x))$. Then, by conjugation in $\mathbb{Q}(\omega)$ we have $g_0(A(x)) + \overline{\omega} g_1(A(x)) = \overline{D_\omega}(x)(g_0(x) + \overline{\omega} g_1(x))$. When we multiply, we get

$$\begin{aligned} f(A(x)) &= (g_0(A(x)) + \omega g_1(A(x))) \cdot (g_0(A(x)) + \overline{\omega} g_1(A(x))) \\ &= (D_\omega(x)(g_0(x) + \omega g_1(x))) \cdot \left(\overline{D_\omega}(x)(g_0(x) + \overline{\omega} g_1(x))\right) \\ &= \left(D_\omega(x)\overline{D_\omega}(x)\right) f(x). \end{aligned}$$

  By noting that $D_\omega(x)\overline{D_\omega}(x)$ belongs to $\mathbb{Q}(x)$, we conclude that A is a $\mathbb{Q}$-automorphism for f.

6 Record Computations

6.1 Setup

In order to test how our ideas perform in practice, we did several medium-sized practical experiments in fields of the form $\mathbb{F}_{p^2}$. We have decided to choose a prime number p of 90 decimal digits so that $\mathbb{F}_{p^2}$ has size 180 digits, the current record-size for $\mathbb{F}_p$. This corresponds to a 600-bit field. To demonstrate that our approach is not specific to a particular form of the prime, we took the first 90 decimal digits of π. Our prime number p is the next prime such that $p \equiv 7 \bmod 8$ and both $p+1$ and $p-1$ have a large prime factor: $p = \lfloor \pi \cdot 10^{89} \rfloor + 14905741$.

$$\begin{aligned} p &= 314159265358979323846264338327950288419716939937510582097494459230781640628620899877709223 \\ \ell &= 392699081698724154807830422909937860524646174921888227621868074038477050785776124847136 53 \\ p - 1 &= 6 \cdot h_0 \text{ with } h_0 \text{ a 89 digit prime} \\ p + 1 &= 8 \cdot \ell \end{aligned}$$

We solved the discrete logarithm problem in the order ℓ subgroup. We imposed p to be congruent to -1 modulo 8, so that the polynomial $f(x) = x^4+1$ could be used. The Conjugation method yields a polynomial g of degree 2 and negative discriminant:

$$\begin{aligned} f &= x^4 + 1 \\ g &= 448225077249286433565160965828828303618362474\ x^2 \\ &\quad - 29606109908476368046927513730655796265782462 3\ \ x \\ &\quad + 448225077249286433565160965828828303618362474\ . \end{aligned}$$

Since p is 90 digits long, the coefficients of g have 45 digits. The polynomials f and g have the irreducible factor

$$\varphi = t^2 + 10778151309582301866698988310224439480941229764389534\backslash \\ 90974106325080494553766987846916995 93\, t + 1$$

in common modulo p, and $\mathbb{F}_{p^2}$ will be taken as $\mathbb{F}_p[X]/\langle\varphi\rangle$.

This choice of polynomials has several practical advantages. Both f and g are reciprocal polynomials, so that $x \mapsto 1/x$ is an automorphism in the sense of Subsection 5.3. This provides a speed-up by a factor of 2 for the relation collection and a factor of 4 in the linear algebra, as explained below. Furthermore, the polynomial f corresponds to a number field K_f with unit rank 1, and a fundamental unit is given by the fundamental unit of the subfield $\mathbb{Q}(\sqrt{2})$. By construction, 2 is a square modulo p, so that $\sqrt{2}$ belongs to $\mathbb{F}_p$. Then, the image in $\mathbb{F}_{p^2}$ of the fundamental unit of K_f is actually in $\mathbb{F}_p$ and its discrete logarithm is 0 modulo ℓ. Since the polynomial g corresponds to a number field with unit rank 0, we do not need Schirokauer maps for this case. Generalizations of this interesting fact will be explained elsewhere.

6.2 Collecting Relations

The relation collection step was then done using the sieving software of CADO-NFS [1]. More precisely, we used the special-q technique for ideals q on the g-side, since it produces norms that are larger than on the f-side. We sieved all the special-qs between $120,000,000$ and $160,000,000$, keeping only one in each pair of conjugates under the action $x \mapsto 1/x$. Indeed, if $\phi = a - bx$ gives a relation for a special-q, then $b - ax$ yields a relation for the conjugate ideal of this special-q. In total, we computed about 34M relations.

The main parameters in the sieve were the following: we sieved all primes below 80M on the f-side, and below 120M on the g-side, and we allowed two large primes less than 2^{29} on each side. The search space for each special-q was set to $2^{15} \times 2^{14}$ (the parameter I in CADO-NFS was set to 15).

The total CPU time for this relation collection step is equivalent to 157 days on one core of an Intel Xeon E5-2650 at 2 GHz. This was run in parallel on a few nodes, each with 16 cores, so that the elapsed time for this step was a few days, and could easily be made arbitrary small with enough nodes.

Table 4. Comparison of running time for integer factorization (NFS-IF), discrete logarithm in prime field (NFS-DL(p)) and in quadratic field (NFS-DL(p^2)) of same global size 180 dd

Algorithm	relation collection	linear algebra	total
NFS-IF	5 years	5.5 months	5.5 years
NFS-DL(p)	50 years	80 years	130 years
NFS-DL(p^2)	157 days	18 days (GPU)	0.5 years

6.3 Linear Algebra

The filtering step was run as usual, but we modified it to take into account the Galois action on the ideals: we selected a representative ideal in each orbit under the action $x \mapsto 1/x$, and rewrote all the relations in terms of these representatives only. Indeed, it can be shown that the corresponding virtual logarithms are opposite modulo ℓ; this amounts just to keep track of sign-change, that has to be reminded when combining two relations during the filtering, and when preparing the sparse matrix for the sparse linear algebra step. Since we keep only half of the columns in the matrix, and assuming a quadratic cost for the linear algebra step, the $x \mapsto 1/x$ automorphism saves a factor of 4, as claimed. The output of the filtering step was a matrix with about 2.7M rows and columns, having on average 86 non-zero entries per row.

Thanks to our choice of f and g, it was not necessary to add columns with Schirokauer maps. We used Jeljeli's implementation of Block Wiedemann's algorithm for GPUs [13,14]. We used two sequences in parallel, on two independent NVidia GTX 680 graphic cards. The total running time for this step is equivalent to around 18.2 days on a single NVidia GTX 680 graphic card.

At the end of the linear algebra we know the virtual logarithms of almost all prime ideals of degree one above primes of at most 28 bits, and of some of those above primes of 29 bits. At this point we could test that the logs on the f-side were correct.

6.4 Computing Individual Logarithms

The last step is that of computing some individual logarithms. We used $G = t+2$ as a generator for $\mathbb{F}_{p^2}$ and the following "random" element:

$$s = \lfloor(\pi(2^{298})/8)\rfloor t + \lfloor(\gamma \cdot 2^{298})\rfloor.$$

We started by looking for an integer e such that $z = s^e$, seen as an element of the number field of f, is smooth. After a few dozen of core-hours, we found a value of e such that $z = z_1/z_2$ with z_1 and z_2 splitting completely into prime ideals of at most 65 bits. With the lattice-sieving software of CADO-NFS, we then performed a "special-$\mathfrak{q}$ descent" for each of these prime ideals. The total time for descending all the prime ideals was a few minutes. Finally, we found

$$\log_G s \equiv 276214243617912804300337349268306605403758173819414418610\backslash 98322785683188853924304990580 12 \bmod \ell.$$

7 Conclusions

The present article contains new estimates for the complexity of solving DLP over non-prime finite fields. We have discovered several places in the $(\log p, n)$ plane where more methods battle to be the best ones. We have also analyzed the complexity of sieving on a domain of non-linear polynomials, and this shows the way for more algorithmic problems, so that this could be a routine problem for subsequent records.

From a practical point of view, we have demonstrated that a clever use of algebraic properties of fields occurring in DLP computations, such as finding polynomials defining number fields with automorphisms and/or Galois properties, gives a significant practical speed-up. This study will be continued elsewhere.

We gather some figures for the factorization of an 180 decimal digit composite number; the time needed for solving DLP on $\mathbb{F}_p$ with p of 180 decimal digits taken from [6] and our computations for $\mathbb{F}_{p^2}$ with p of 90 decimal digits.

Considering the relation collection phase only, we see that for the same object size, a DLP over $\mathbb{F}_{p^2}$ is much easier than the corresponding factoring of an integer. This tends to contradict the usual rule-of-thumb: *The discrete logarithm problem in large characteristic finite fields is at least as hard as factoring an integer of the same size as the cardinality of the finite field.*

References

1. Bai, S., Filbois, A., Gaudry, P., Kruppa, A., Morain, F., Thomé, E., Zimmermann, P., et al.: Crible algébrique: Distribution, optimisation - NFS (2009). downloadable at http://cado-nfs.gforge.inria.fr/
2. Barbulescu, R., Gaudry, P., Joux, A., Thomé, E.: A Heuristic Quasi-Polynomial Algorithm for Discrete Logarithm in Finite Fields of Small Characteristic. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 1–16. Springer, Heidelberg (2014)
3. Barbulescu, R., Pierrot, C.: The multiple number field sieve for medium- and high-characteristic finite fields. LMS Journal of Computation and Mathematics 17, 230–246 (2014). http://journals.cambridge.org/article_S1461157014000369
4. Barbulescu, R.: Algorithmes de logarithmes discrets dans les corps finis. Ph.D. thesis, Université de Lorraine (2013)
5. Barbulescu, R., Gaudry, P., Guillevic, A., Morain, F.: Improvements to the number field sieve for non-prime finite fields. preprint available at http://hal.inria.fr/hal-01052449
6. Bouvier, C., Gaudry, P., Imbert, L., Jeljeli, H., Thomé, E.: Discrete logarithms in GF(p) – 180 digits (2014), announcement available at the NMBRTHRY archives, item 004703
7. Canfield, E.R., Erdös, P., Pomerance, C.: On a problem of Oppenheim concerning "factorisatio numerorum". J. Number Theory **17**(1), 1–28 (1983)

8. Collins, G.E., Encarnación, M.J.: Efficient rational number reconstruction. Journal of Symbolic Computation **20**(3), 287–297 (1995)
9. Coppersmith, D.: Modifications to the number field sieve. J. of Cryptology **6**(3), 169–180 (1993)
10. Coppersmith, D.: Solving homogeneous linear equations over GF(2) via block Wiedemann algorithm. Math. Comp. **62**(205), 333–350 (1994)
11. Foster, K.: HT90 and "simplest" number fields. Illinois J. Math. **55**(4), 1621–1655 (2011)
12. Freeman, D., Scott, M., Teske, E.: A taxonomy of pairing-friendly elliptic curves. J. of Cryptology **23**(2), 224–280 (2010)
13. Jeljeli, H.: Accelerating iterative SpMV for discrete logarithm problem using GPUs (2014). http://hal.inria.fr/hal-00734975/, preprint, to appear in WAIFI 2014
14. Jeljeli, H.: An implementation of the Block-Wiedemann algorithm on NVIDIA-GPUs using the Residue Number System (RNS) arithmetic (2014). available from http://www.loria.fr/~hjeljeli/
15. Joux, A., Lercier, R.: Improvements to the general number field for discrete logarithms in prime fields. Math. Comp. **72**(242), 953–967 (2003)
16. Joux, A., Lercier, R., Smart, N.P., Vercauteren, F.: The Number Field Sieve in the Medium Prime Case. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 326–344. Springer, Heidelberg (2006)
17. Joux, A., Lercier, R., et al.: Algorithmes pour résoudre le problème du logarithme discret dans les corps finis. Nouvelles Méthodes Mathématiques en Cryptographie, volume Fascicule Journées Annuelles, p. 23 (2007)
18. Joux, A., Pierrot, C.: The Special Number Field Sieve in $\mathbb{F}_{p^n}$. In: Cao, Z., Zhang, F. (eds.) Pairing 2013. LNCS, vol. 8365, pp. 45–61. Springer, Heidelberg (2014)
19. Kalkbrener, M.: An upper bound on the number of monomials in determinants of sparse matrices with symbolic entries. Mathematica Pannonica **73**, 82 (1997)
20. Kleinjung, T.: On polynomial selection for the general number field sieve. Mathematics of Computation **75**(256), 2037–2047 (2006)
21. Lenstra, A.K., Verheul, E.R.: The XTR Public Key System. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 1–19. Springer, Heidelberg (2000)
22. Lenstra, A.K., Lenstra, H.W., Lovász, L.: Factoring polynomials with rational coefficients. Mathematische Annalen **261**(4), 515–534 (1982)
23. Matyukhin, D.V.: On asymptotic complexity of computing discrete logarithms over GF(p). Discrete Mathematics and Applications **13**(1), 27–50 (2003)
24. Matyukhin, D.: Effective version of the number field sieve for discrete logarithms in the field GF(p^k). Trudy po Discretnoi Matematike **9**, 121–151 (2006) (in Russian). http://m.mathnet.ru/php/archive.phtml?wshow=paper&jrnid=tdm&paperid=144&option_lang=eng
25. Murphy, B.A.: Polynomial selection for the number field sieve integer factorisation algorithm. Ph.D. thesis, Australian National Univers (1999)
26. Pierrot, C.: The multiple number field sieve with conjugation method (August 2014). preprint available at https://eprint.iacr.org/2014/641
27. Pohlig, S., Hellman, M.: An improved algorithm for computing logarithms over GF(p) and his cryptographic significance. IEEE Trans. Inform. Theory **24**(1), 106–110 (1978)
28. Pollard, J.M.: Monte Carlo methods for index computation (mod p). Math. Comp. **32**(143), 918–924 (1978)
29. Rubin, K., Silverberg, A.: Torus-Based Cryptography. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 349–365. Springer, Heidelberg (2003)

30. Schirokauer, O.: Using number fields to compute logarithms in finite fields. Math. Comp. **69**(231), 1267–1283 (2000)
31. Schirokauer, O.: Virtual logarithms. J. Algorithms **57**, 140–147 (2005)
32. Smith, P., Skinner, C.: A public-key cryptosystem and a digital signature system based on the Lucas function analogue to discrete logarithms. In: Pieprzyk, J., Safavi-Naini, R. (eds.) Advances in Cryptology - ASIACRYPT 1994. LNCS, vol. 917, pp. 357–364. Springer, Heidelberg (1994)
33. Wiedemann, D.: Solving sparse linear equations over finite fields. IEEE Trans. Inform. Theory **32**(1), 54–62 (1986)