# New Discrete Logarithm Computation for the Medium Prime Case Using the Function Field Sieve

Madhurima Mukhopadhyay, Palash Sarkar, Shashank Singh, Emmanuel Thomé

# New Discrete Logarithm Computation for the Medium Prime Case Using the Function Field Sieve

Madhurima Mukhopadhyay[1], Palash Sarkar[1], Shashank Singh[2], and Emmanuel Thomé[3]

[1]Applied Statistics Unit, Indian Statistical Institute, 203 B. T. Road, Kolkata, India 700108
[2]Electrical Engineering & Computer Science, Indian Institute of Science Education and Research Bhopal, Bhopal 462066, Madhya Pradesh, India
[3] Université de Lorraine, CNRS, INRIA, Nancy, France
{madhurima_r,palash}@isical.ac.in, shashank@iiserb.ac.in, Emmanuel.Thome@inria.fr

February 4, 2020

**Abstract**

The present work reports progress in discrete logarithm computation for the general medium prime case using the function field sieve algorithm. A new record discrete logarithm computation over a 1051-bit field having a 22-bit characteristic was performed. This computation builds on and implements previously known techniques. Analysis indicates that the relation collection and descent steps are within reach for fields with 32-bit characteristic and moderate extension degrees. It is the linear algebra step which will dominate the computation time for any discrete logarithm computation over such fields.
**Keywords:** finite field, discrete logarithm, function field sieve.
**MSC (2010):** 11Y16, 94A60.

## 1 Introduction

Let $p$ be a prime and $n$ be a positive integer. Let $\mathbb{F}_{p^n}$ be the finite field consisting of $p^n$ elements. Let $\mathfrak{g}$ be a generator of the cyclic group of all non-zero elements of $\mathbb{F}_{p^n}$. The discrete logarithm problem in $\mathbb{F}_{p^n}$ is the following. Given a non-zero element $\mathfrak{h}$ of $\mathbb{F}_{p^n}$, find $i$ such that $\mathfrak{h}^i = \mathfrak{g}$. The computational difficulty of the discrete logarithm problem forms the basis for the security of cryptographic primitives such as the Diffie-Hellman key agreement scheme [6] and the Digital Signature Algorithm [18].

Let $Q = p^n$ and $L_Q(a, c)$ with $0 < a < 1$ and $c > 0$ denote the sub-exponential expression

$$L_Q(a, c) = \exp\left((c + o(1))(\ln Q)^a (\ln \ln Q)^{1-a}\right).$$

Writing $p = L_Q(a, c)$ leads to several cases: $a > 2/3$ is called the large characteristic case, $a = 2/3$ is called the boundary case, $1/3 \leq a < 2/3$ is called the medium prime case while $a < 1/3$ is called the small characteristic case.

There are two major sub-exponential time algorithms for solving the discrete logarithm problem over a finite field, namely the number field sieve (NFS) [9] and the function field sieve (FFS) [2, 3, 15]. Both algorithms have seen substantial improvements over time and several variants of these algorithms are presently known. In the present state of the art, the NFS is generally used for large characteristic fields while the FFS is used for small to medium characteristic fields. In this work, we will consider discrete logarithm computation for a medium characteristic prime using the FFS.

For the small characteristic case, there has been a tremendous amount of progress in the FFS algorithm [8, 7, 5, 14, 4, 10, 1]. A quasi-polynomial time algorithm has been obtained in [4]. Record discrete logarithm computations have been made over large fields such as $\mathbb{F}_{2^{9234}}$ [11] and $\mathbb{F}_{2^{30750}}$ [12].

Table 1: A comparison of the difficulty of computing discrete logarithms for the medium prime case using the function field sieve algorithm.

| Ref | $\lceil \log_2 p \rceil$ | $n$ | $\lceil \log_2 p^n \rceil$ | $\lceil \log_2 \#\mathbb{B} \rceil$ | $\Lambda$ |
|---|---|---|---|---|---|
| JL [16] | 17 | 25 | 401 | 18 | 3.79 |
| SS [19] | 16 | 37 | 592 | 17 | 0.11 |
| SS [19] | 19 | 40 | 728 | 20 | 0.08 |
| This work | 22 | 50 | 1051 | 23 | 0.07 |

Table 2: A comparison of the difficulty of computing discrete logarithms for the medium prime case using the function field sieve algorithm for Kummer extensions, i.e., for fields $\mathbb{F}_{p^n}$ satisfying $n \mid (p-1)$.

| Ref | $\lceil \log_2 p \rceil$ | $n$ | $\lceil \log_2 p^n \rceil$ | $\lceil \log_2 \#\mathbb{B} \rceil$ | $\Lambda$ |
|---|---|---|---|---|---|
| JL [16] | 19 | 30 | 556 | 18 | 4.29 |
| Joux [14] | 25 | 47 | 1175 | 20 | 0.77 |
| Joux [14] | 25 | 57 | 1425 | 20 | 0.13 |

There has also been progress in discrete logarithm computation using the FFS for the medium prime case. This progress, however, has not been as remarkable as in the small characteristic case. We briefly summarise the previous works. Important simplification of the FFS was made by Joux and Lercier [16]. The next work was by Joux [14] who introduced the important idea of pinpointing. Later work by Sarkar and Singh [19] performed a detailed asymptotic analysis.

All three of the works [16, 14, 19] reported discrete logarithm computations. These are summarised in Tables 1 and 2. Table 1 compares the various discrete logarithm computations in the medium prime case using the FFS algorithm. For Kummer extensions, the condition $n \mid (p-1)$ holds. Comparison of previously performed discrete logarithm computations for Kummer extensions are shown in Table 2. In Tables 1 and 2, $\#\mathbb{B}$ is the size of the factor basis. The parameter $\Lambda$ is a measure of the feasibility of 2-1 descent. The lower the value of $\Lambda$, the more difficult it is to carry out a 2-1 descent. We provide the definition of $\Lambda$ and explain its connection to the difficulty of 2-1 descent later.

The present work represents progress in the discrete logarithm computation for the medium prime case using the FFS algorithm. The challenge was to perform a larger discrete logarithm computation for a medium prime case field than what has been reported earlier. To keep the problem general, we decided not to work with Kummer extensions. We chose a 1051-bit field having a 22-bit characteristic and extension degree 50 as our target. While the size of this field is smaller than the 1175-bit and the 1425-bit fields considered by Joux [14], the Kummer extension property of the latter two fields make the discrete logarithm computation much easier than the field considered in this work. In particular, for the fields considered in [14], 20-bit factor bases suffice whereas in our case a 23-bit factor basis is required. Also, the 2-1 descent for the field considered in this work is more difficult than those considered in [14]. This is indicated by the value of $\Lambda$ in Tables 1 and 2. More details on the 2-1 descent are provided later.

For our computation, the main techniques that were used are from [14, 19]. Applying these techniques to a larger field, on the other hand, created complications, especially in the descent step. This required building on and implementing the alternating walk and branching technique. We considered the feasibility of using the FFS algorithm to solve a discrete logarithm challenge for a field having a 32-bit characteristic and extension degree 17. For this challenge, the relation collection and the descent phases are well within reach. The linear algebra step, on the other hand, will require much more time. The ability to solve the discrete logarithm challenge depends on the feasibility of the linear algebra step.

## 2 Function Field Sieve in the Medium Prime Case

We provide a brief description of the FFS algorithm for the medium prime case. For details we refer to [16, 14, 19]. We will assume that the extension degree $n$ is greater than 1.

**Representation of $\mathbb{F}_{p^n}$:** The first task is to choose a convenient representation for $\mathbb{F}_{p^n}$.

Choose $n_1, n_2$ and $k$ to be positive integers less than $n$. Let $g_1(X) = X^{-n_1}$ and $g_2(X)$ be a polynomial over $\mathbb{F}_p$ of degree $n_2$. Write $X^k - g_2(g_1(X)) = f_1(X)/X^{n_1 n_2}$, where $f_1(X)$ is a polynomial of degree $n_1 n_2 + k$. The idea is to choose $g_2(X)$ to be a polynomial over $\mathbb{F}_p$ of degree $n_2$ such that $f_1(X)$ has an irreducible factor $f(X)$ of degree $n$. Since the degree of $f_1(X)$ is $n_1 n_2 + k$, $n \leq n_1 n_2 + k$. If $n = n_1 n_2 + k$, then experiments show that it is possible to choose $g_2(X)$ such that $f_1(X)$ itself is irreducible. In this case, we take $f(X)$ to be equal to $f_1(X)$.

The field $\mathbb{F}_{p^n}$ is represented as $\mathbb{F}_p[x]/(f(x))$. Let $y = g_1(x) = x^{-n_1}$. Then $x^k - g_2(y) = f_1(x)/x^{n_1 n_2}$. Since $f(x) \mid f_1(x)$, the relation $x^k = g_2(y)$ holds in $\mathbb{F}_{p^n}$. For $k = 1$, this method was described by Joux [14]. In the following, we will assume $k = 1$, $y = g_1(x) = x^{-n_1}$ and $x = g_2(y)$ since these are the choices we use in our discrete logarithm computation. For other variants of choosing the field representation, we refer to [14, 19].

**Choice of generator:** The non-zero elements of $\mathbb{F}_{p^n}$ forms a cyclic group under multiplication. Discrete logarithms are computed with respect to some generator. The actual choice of a generator is not important. It usually turns out that one of the polynomials $x + a$ is a generator for some $a \in \mathbb{F}_p$.

**Factor basis:** The factor basis is $\mathbb{B} = \{(x + a_i), (y + b_j) : a_i, b_j \in \mathbb{F}_p\}$. There are $2p$ elements in $\mathbb{B}$.

For certain extension fields, by using the action of Frobenius it is possible to reduce the size of the factor basis by a factor of $n$. Joux [14] showed this for Kummer extensions, i.e., for fields $\mathbb{F}_{p^n}$ such that $n \mid (p-1)$. An advantage of Kummer extensions is that the size of the factor basis reduces by a factor of $n$, making it possible to perform discrete logarithm computations on larger fields. This can be seen from Tables 1 and 2. The condition $n \mid (p-1)$, however, is very special and is unlikely to hold for general $p$ and $n$. Due to this reason, we do not consider this condition in our work.

**Modulus of discrete logarithms:** The requirement is to compute the discrete logarithm modulo $p^n - 1$. In practice, $p^n - 1$ is factored and the FFS algorithm is used to compute the discrete logarithm modulo the large prime factors. Let $M = (p^n - 1)/(p - 1)$. The discrete logarithms of elements of $\mathbb{F}_p$ are equal to 0 modulo $M$ and so in the computation of discrete logarithm modulo a large prime factor of $M$, one may ignore the discrete logarithms of the elements of $\mathbb{F}_p$. See [19] for an explanation.

## 2.1 Relation Collection

For arbitrary elements $a, b, c \in \mathbb{F}_p$, consider the expression $(x + a)y + (bx + c) = xy + ay + bx + c$. Using $y = g_1(x) = x^{-n_1}$ and $x = g_2(y)$, this expression can be written in two different ways as follows.

$$xg_1(x) + ag_1(x) + bx + c = (x + a + bx^{n_1+1} + cx^{n_1})/x^{n_1} = h_1(x)/x^{n_1}; \quad (1)$$

$$yg_2(y) + ay + bg_2(y) + c = h_2(y). \quad (2)$$

Note that $h_1(x)$ is a polynomial of degree $n_1 + 1$ and $h_2(y)$ is a polynomial of degree $n_2 + 1$. Over $\mathbb{F}_{p^n}$, we have $h_1(x)/x^{n_1} = h_2(y)$. Suppose that both $h_1(x)$ and $h_2(y)$ are smooth polynomials, i.e., $h_1(x) = b\,\Pi_{\alpha_i}(x + \alpha_i)$ and $h_2(y) = d_1\Pi_{\beta_j}(y + \beta_j)$ for some $d_1 \in \mathbb{F}_p$. Then, over $\mathbb{F}_{p^n}$, we have the relation

$$h_1(x)/x^{n_1} = b\,\Pi_{\alpha_i}(x + \alpha_i)/x^{n_1} = d_1\Pi_{\beta_j}(y + \beta_j) = h_2(y). \quad (3)$$

This gives the following linear equation among the discrete logarithms of the elements of the factor basis.

$$-n_1 \log x + \sum_{\alpha_i} \log(x + \alpha_i) = \sum_{\beta_j} \log(y + \beta_j) \bmod M.$$

Each such linear equation involves $n_1 + n_2 + 1$ terms.

The factor basis contains $2p$ elements. To be able to solve the system of linear equations arising from linear equations of the above type, a little more than $2p$ relations are required. The free parameters are

$a$, $b$ and $c$ giving rise to $p^3$ expressions of the type $xy + ay + bx + c$. Heuristically, we may assume that the $p^3$ expressions give rise to $p^3/((n_1+1)!(n_2+1)!)$ linear equations. So, for the relation collection phase to succeed, the following condition has to hold.

$$\frac{p^3}{(n_1+1)!(n_2+1)!} \quad > \quad 2p. \tag{4}$$

**Pinpointing:** The idea of pinpointing was introduced by Joux [14] to speed up relation collection. Suppose that for some choice of $a, b$ and $c$, the polynomial $h_1(x)$ turns out to be smooth, i.e.,

$$x + a + bx^{n_1+1} + cx^{n_1} \quad = \quad b\Pi_{\alpha_i}(x + \alpha_i). \tag{5}$$

Using the transformation $x \mapsto tx$, for $t \in \mathbb{F}_p \setminus \{0, 1\}$, the right side of (5) remains smooth, while the left hand side corresponds to the expression obtained from $a' = a$, $b' = bt^{n_1+1}$ and $c' = ct^{n_1}$. So, once a smooth $h_1(x)$ is obtained, by varying $t$ over all elements of $\mathbb{F}_p$, it is possible to obtain $p-2$ smooth $h_1(x)$'s without any further smoothness checking.

In [14], it was shown that the number of trials required for obtaining a single relation which has both sides smooth is

$$\frac{(n_1+1)!(n_2+1)!}{p-1} + \min\left((n_1+1)!, (n_2+1)!\right). \tag{6}$$

It was shown in [19] that the idea of smoothness checking can be combined with a sieving procedure which entirely avoids smoothness checking. Our implementation uses the sieving based pinpointing algorithm.

### 2.2 Linear Algebra

The relation collection phase produces a little more than $2p$ linear equations involving the discrete logarithms of the elements of the factor basis. Each equation has $n_1 + n_2 + 1$ terms. Additionally, we include the linear equation $\log y = -n_1 \log x$ to account for the relation $y = x^{-n_1}$ between $x$ and $y$.

The obtained system of linear equations is sparse. Techniques from sparse linear algebra are used to solve the linear system. The standard methods for solving a sparse linear system is either the Lanczos algorithm or the block Wiedemann algorithm. For a factor basis of size $B$, the cost of both the methods is $O(B^2)$. The second method is preferable as it can be parallelised. We have used the block Wiedemann algorithm implemented in the CADO-NFS [21] to complete the linear algebra step.

The system of inhomogenous linear equations is given by a matrix $\mathbf{M}$ and a coefficient vector $\mathbf{b}$. Before attempting to solve the system, a filtering step is applied. The goal of the filtering step is reduce the size of the matrix and/or make it more sparse. The basic filtering that we applied was to remove duplicate rows and empty columns of the matrix $\mathbf{M}$. It is possible to apply other more sophisticated filtering methods.

It is usually assumed that the relation collection step provides relations involving all elements of the factor basis. There is no proof that this will necessarily be the case. Our computation shows that this assumption need not be true, i.e., there could be elements of the factor basis which are not part of any relation. We provide more details later.

### 2.3 Individual Descent

Let $\Pi(x)$ be the target element whose discrete logarithm is to be computed. Typically, $\Pi(x)$ will be a polynomial of degree $n-1$. After the linear algebra step, assume that we have computed the discrete logarithms of all linear polynomials of the form $x+\alpha_i$ and $y+\beta_j$. So, the goal is to be able to express $\Pi(x)$ as a rational function where both the numerator and the denominator are products of linear polynomials. This procedure is called descent.

The entire descent is not done in a single step. The target polynomial is successively descended to lower degree polynomials until finally descent to linear polynomials become possible. For the initial

descent, a simple randomisation strategy usually works. Suppose we wish to descend from an irreducible polynomial $\phi(x)$. Choose a random polynomial $D(x)$ whose factors are of lower degree than that of $\phi(x)$. Let $N(x) = \phi(x)D(x) \bmod f(x)$. If the factors of $N(x)$ are also of degrees lower than that of $\phi(x)$, then since $\phi(x) = N(x)/D(x)$, we have a descent from $\phi(x)$ to lower degree factors of $N(x)$ and $D(x)$. If these factors are not linear, then they would require to be further descended. More systematic techniques for descent are known. A method based on computing the kernel of a matrix has been described in [16, 13]. Another method has been used in [19] and we provide further details in the next section.

The descent becomes more difficult as the degree of the polynomials become close to 1 with the 2-1 descent (i.e., descent from quadratic to linear polynomials) being the most difficult. A heuristic argument has been used to show that the probability of a successful 2-1 descent in a single trial is $1/((n_1 - 1)!(n_2 + 1)!)$ [14, 19]. The work [19] provides the probability of a successful $d$-$(d-1)$ descent (i.e., descent from a degree $d$ polynomial to polynomials of degrees at most $d$) for $d \geq 2$. In an asymptotic setting the effect of $d$-$(d-1)$ descent on the overall time for solving discrete logarithm has been analysed in [19]. It has been shown that for $d > 2$, the asymptotic cost of $d$-$(d-1)$ descent is always lower than the asymptotic cost of relation collection. On the other hand, for $d = 2$, there are situations where the asymptotic cost of 2-1 descent is more than the asymptotic costs of the other two phases.

Following the methods of [16, 14, 19], using a single degree of freedom, the heuristic probability of success for a 2-1 descent is $1/((n_1-1)!(n_2+1)!)$. So, the number of trials required for a single 2-1 descent is about

$$(n_1 - 1)!(n_2 + 1)!. \tag{7}$$

With a single degree of freedom, the number of trials that can be made is $p$. Let

$$\Lambda \quad = \quad \frac{p}{(n_1 - 1)!(n_2 + 1)!}. \tag{8}$$

It has been suggested in [16, 14, 19] that for a 2-1 descent to be possible, $\Lambda \geq 1$ has to hold. Experiments show that while $\Lambda \geq 1$ makes the descent easy, it may be possible to perform a 2-1 descent even when $\Lambda < 1$. The parameter $\Lambda$ does, however, play a role in determining the ease of descent. The higher the value of $\Lambda$, the easier is a 2-1 descent, while for lower values of $\Lambda$, a direct 2-1 descent may not be possible and one would have to use walk and/or branching techniques (as explained in the next section).

### 2.4 Final Discrete Logarithm Computation

The linear algebra step provides the discrete logarithm of the elements of the factor basis elements modulo the large prime divisors of $p^n - 1$. So, once the descent step is completed, it is possible to compute the discrete logarithm of the target element modulo the large prime divisors of $p^n - 1$. The discrete logarithm of the target element modulo the smaller factors of $p^n - 1$ are computed using the Pollard rho and the Pohlig-Hellman algorithm. Finally, all the discrete logarithms are combined using the Chinese remainder theorem to obtain the discrete logarithm of the target element modulo $p^n - 1$.

## 3 Sieving Using Partial Smoothness-cum-Divisibility

The technique of smoothness-cum-divisibility and a sieving method based on it was introduced in [19]. Here we provide a brief account of this method based on the relations $y = g_1(x) = x^{-n_1}$ and $x = g_2(y)$. (This description is somewhat different from the one in [19] which was based on $y = g_1(x) = x^{n_1}$ and $x = g_2(y)$.)

Let $\phi(x)$ be a polynomial of degree $d \geq 0$. Let $T(x, y)$ be a bivariate polynomial and let $F(x)$ and $C(y)$ be such that $T(x, g_1(x)) = F(x)/x^{n_1}$ and $C(y) = T(g_2(y), y)$. The polynomial $T(x, y)$ is said to be *good for* $\phi(x)$ if $\phi(x)$ divides $F(x)$ and both $G(x) = F(x)/\phi(x)$ and $C(y)$ are smooth, i.e., can be factored into linear polynomials. Note that $F(x) = \mathrm{Res}_y(T(x, y), x^{n_1}y - 1)$, and $C(y) = \mathrm{Res}_x(T(x, y), x - g_2(y))$.

Suppose $T(x,y)$ is a monic polynomial which has a total of $\rho+1$ monomials. We assume $d<\rho$. Let the degrees of $F(x)$ and $C(y)$ be $\rho_1$ and $\rho_2$ respectively. The degree of $G(x)$ is $\rho_1-d$. Let $e=\rho-d$ which represents the degree of freedom. Let us write

$$F(x) \;=\; \phi(x)(x-a_1)\cdots(x-a_e)H(x) \tag{9}$$

for some polynomial $H(x)$ of degree $h=\rho_1-d-e$. So, $G(x)=(x-a_1)\cdots(x-a_e)H(x)$. If we can find $a_1,\ldots,a_e\in\mathbb{F}_p$ such that $F(x)$ can be written as in (9), then we are able to ensure that $F(x)$ is divisible by $\phi(x)$ and partial smoothness of $G(x)$. By trying various values of $a_1,\ldots,a_e$, the smoothness of $H(x)$ and the corresponding $C(y)$ has to be ensured. A sieving based method for implementing this idea has been described in [19].

The partial smoothness-cum-divisibility technique is useful for both relation collection and the descent step. In the context of relation collection, we set $T(x,y)=xy+ay+bx+c$ and $\phi(x)=1$ so that $\rho=3$ and $d=0$ leading to $e=\rho-d=3$. The resulting sieving technique can be combined with pinpointing. We refer to [19] for further details.

For application to the descent step, the 2-1 descent is described in details in [19] since it is for such descent that the partial smoothness-cum-divisibility technique was applied. Here we describe how the technique can be used for $d$-$(d-1)$ descent for $d\geq 2$. Let $\phi(x)$ be a polynomial of degree $d$ and the goal is to descend to polynomials of degrees less than $d$. Let $\rho=d+1$ so that $e=\rho-d=1$, providing a single degree of freedom. With $e=1$, we have

$$F(x) \;=\; \phi(x)(x-\alpha)H(x) \tag{10}$$

where degree of $H(x)$ is $h$. The $\rho$ undetermined coefficients of $T(x,y)$ appears in $F(x)$. As before, let $G(x)=(x-\alpha)H(x)$.

Consider $\alpha$ to be a symbolic variable. From (10) and using a method described in [19], it is possible to symbolically solve for the coefficients of $H(x)$ and $F(x)$ in terms of $\alpha$. The symbolic computation is a one-time task.

We provide an example of the symbolic computation. Suppose $n=49=n_1n_2$ with $n_1=n_2=7$, $T(x,y)=xy+ax+by+c$ where $a,b$ and $c$ are undetermined elements of $\mathbb{F}_p$. Then $F(x)=T(x,x^{n_1})$ is a polynomial of degree $\rho_1=8$. Let $\phi(x)=q_0+q_1x+q_2x^2$, where $q_0,q_1,q_2\in\mathbb{F}_p$. So, $d=2$, $e=1$ and from (10), the degree $h$ of $H(x)$ is 5. Symbolic computation using SAGE [20] provides $H(x)=h_0+h_1x+\cdots+h_4x^4+x^5$ where $h_0,\ldots,h_4$ are as follows.

$$\begin{aligned}
h_0 &= -(\alpha^5q_1^5-\alpha^4q_0q_1^4+\alpha^3q_0^2q_1^3-\alpha^2q_0^3q_1^2+\alpha q_0^4q_1-q_0^5+(3\alpha^5q_0^2q_1-\alpha^4q_0^3)q_2^2\\
&\quad -(4\alpha^5q_0q_1^3-3\alpha^4q_0^2q_1^2+2\alpha^3q_0^3q_1-\alpha^2q_0^4)q_2)/\Delta(\alpha);\\
h_1 &= q_0^2q_2^3-\alpha^4q_1^5+\alpha^3q_0q_1^4-\alpha^2q_0^2q_1^3+\alpha q_0^3q_1^2-q_0^4q_1\\
&\quad -(3\alpha^5q_0q_1^2+\alpha^4q_0^2q_1)q_2^2+(\alpha^5q_1^4+3\alpha^4q_0q_1^3-2\alpha^3q_0^2q_1^2+\alpha^2q_0^3q_1)q_2)/\Delta(\alpha);\\
h_2 &= (2\alpha^5q_0q_1q_2^3-\alpha^3q_1^5+\alpha^2q_0q_1^4-\alpha q_0^2q_1^3+q_0^3q_1^2-(\alpha^5q_1^3+2\alpha^4q_0q_1^2+2\alpha^3q_0^2q_1-\alpha^2q_0^3)q_2^2\\
&\quad +(\alpha^4q_1^4+3\alpha^3q_0q_1^3-2\alpha^2q_0^2q_1^2+\alpha q_0^3q_1-q_0^4)q_2)/\Delta(\alpha);\\
h_3 &= -(\alpha^5q_0q_2^4+\alpha^2q_1^5-\alpha q_0q_1^4+q_0^2q_1^3-(\alpha^5q_1^2+\alpha^4q_0q_1+\alpha^3q_0^2)q_2^3\\
&\quad +(\alpha^4q_1^3+2\alpha^3q_0q_1^2+2\alpha^2q_0^2q_1)q_2^2-(\alpha^3q_1^4+3\alpha^2q_0q_1^3-2\alpha q_0^2q_1^2+2q_0^3q_1)q_2)/\Delta(\alpha);\\
h_4 &= -(\alpha^5q_1q_2^4+\alpha q_1^5-q_0q_1^4-(\alpha^4q_1^2+\alpha^3q_0q_1)q_2^3+(\alpha^3q_1^3+2\alpha^2q_0q_1^2+\alpha q_0^2q_1-q_0^3)q_2^2\\
&\quad -(\alpha^2q_1^4+3\alpha q_0q_1^3-3q_0^2q_1^2)q_2)/\Delta(\alpha),
\end{aligned}$$

where

$$\begin{aligned}
\Delta(\alpha) &= \alpha^5q_2^5-q_1^5-(\alpha^4q_1+\alpha^3q_0)q_2^4+(\alpha^3q_1^2+2\alpha^2q_0q_1+\alpha q_0^2)q_2^3\\
&\quad -(\alpha^2q_1^3+3\alpha q_0q_1^2+3q_0^2q_1)q_2^2+(\alpha q_1^4+4q_0q_1^3)q_2.
\end{aligned}$$

Once the polynomial $\phi(x)$ in (10) is fixed, the coefficients of $H(x)$ and $G(x)$ are functions of $\alpha$. For each possible value of $\alpha$, denote the corresponding $H(x)$ and $G(x)$ as $H_\alpha(x)$ and $G_\alpha(x)$ respectively. Next, for

each possible value of $\alpha \in \mathbb{F}_p$, compute the coefficients of $H_\alpha(x)$ and hence obtain $G_\alpha(x) = (x-\alpha)H_\alpha(x)$. Store all the $G_\alpha(x)$'s in a list $\mathcal{L}$. After $G_\alpha(x)$ has been added to $\mathcal{L}$ for all $\alpha \in \mathbb{F}_p$, sort $\mathcal{L}$. If a $G(x)$ occurs $h-d+2$ or more times in the list, then at least $h-d+2$ roots of $G(x)$ have been encountered in the sieving process. The remaining factor of $G(x)$ has degree at most $d-1$ and so $G(x)$ is $(d-1)$-smooth. For such a $G(x)$, construct the corresponding $F(x)$ as $\phi(x)G(x)$. Using Proposition 1 of [19], obtain $C(y)$ and check whether $C(y)$ is also $(d-1)$ smooth. If it turns out that $C(y)$ is indeed $(d-1)$ smooth, then we have

$$\phi(x) \quad = \quad \frac{x^{n_1}C(y)}{G(x)}$$

where both $C(y)$ and $G(x)$ are $(d-1)$-smooth. So, it has been possible to descend from the polynomial $\phi(x)$ of degree $d$ to the polynomials $C(y)$ and $G(x)$ which are $(d-1)$-smooth. Note that for $d > 2$, while aiming for $d$-$(d-1)$ descent, it might be possible to reach smaller degrees if in the sieving process, a $G(x)$ appears more than $h+d-2$ times. The above description is for descending from a polynomial $\phi(x)$. A similar method works for descending from a polynomial $\psi(y)$.

Suppose the above method is not successful, i.e., the sieving procedure does not result in a suitable $G(x)$ and $C(y)$. At this point, there are several ways to proceed.

**The $x$-$x$ walk:** Suppose the sieving procedure results in a $G(x)$ which has $h-d+1$ linear factors. Then the other factor of $G(x)$ is of degree $d$. Let this factor be $\phi_1(x)$. Further, suppose that $C(y)$ turns out to be $(d-1)$-smooth. Then, in effect, we have moved from a $\phi(x)$ of degree $d$ to the polynomial $\phi_1(x)$ also of degree $d$. Descent may now be attempted from $\phi_1(x)$. Similarly, one may need to move from a polynomial $\psi(y)$ of degree $d$ to a polynomial $\psi_1(y)$ also of degree $d$ and try to descend from $\psi_1(y)$. Such a method is called the walk technique.

**The $x$-$y$ walk:** Suppose the sieving procedure results in a desirable $G(x)$, i.e., one that has at least $h-d+2$ linear factors. On the other hand, suppose that the corresponding $C(y)$ turns out to be $d$-smooth, instead of being $(d-1)$-smooth. For each factor $\psi(y)$ of $C(y)$ of degree $d$, one may try to descend to lower degree polynomials.

Similarly, one may define the $y$-$y$ and the $y$-$x$ walks. It is possible that neither $x$-$x$ nor $x$-$y$ walks succeed for a polynomial $\phi(x)$ of degree $d$. Then the following strategies can be tried.

1. Move from a single degree $d$ polynomial in $x$ to two degree $d$ polynomials in $x$.
2. Move from a single degree $d$ polynomial in $x$ to one degree $d$ polynomials in $x$ and one degree $d$ polynomial in $y$.

Analogous strategies hold for moving from a degree $d$ polynomial $\psi(y)$ in $y$. Since this strategy moves from a single degree $d$ polynomial to two degree $d$ polynomials, it is called a branching strategy.

The walk and branching strategies were briefly mentioned in [14]. Detailed discussion of these strategies in the context of 2-1 descent is given in [19]. The computations in [19] used these techniques only for 2-1 descent. For the present computation, we needed the walk technique for both 3-2 and 2-1 descent.

For the $x$-$y$ walk, an important implementation issue is to avoid cycling. Suppose the $x$-$y$ walk starts from $\phi(x)$. It is possible that after a number of steps, the walk again enters $\phi(x)$. This is called cycling. In the presence of cycling, the descent fails. By suitably using randomisation, it is usually possible to avoid such cycling. Alternatively, cycle detection algorithms may be used to detect the presence of cycling and abort.

## 4 A Concrete Discrete Logarithm Computation

In this section, we present the details of an actual discrete logarithm computation. The prepatory phase, relation collection and the descent steps were done using Magma V2.21-10 on four servers. Out of the four

servers, three have the same configuration with each of these three servers consisting of Intel(R) Xeon(R) E7-4890 @ 2.80 GHz (60 physical cores and 120 logical cores) and the fourth server consists of Intel Xeon E7-8890 @ 2.50 GHz (72 physical cores and 144 logical cores). These servers are shared resources and were simultaneously utilised by other users to run heavy simulation programs. We were never able to obtain exclusive access to the servers. The linear algebra phase was run on a cluster of 16 dual-socket Intel(R) Xeon(R) Gold 6130 CPU @ 2.10GHz connected with Intel 100 Gbps Omni-Path.

In our computation, we chose $p = 2111023$ and $n = 50$. Note that $\lceil \log_2(p) \rceil = 22$ and $\lceil \log_2(p^n) \rceil = 1051$. So, the discrete logarithm computation is over a 1051-bit field having a 22-bit characteristic.

**Preparatory phase:** We chose $n_1 = n_2 = 7$. Experimentally we obtained $g_1(x) = x^{-7}$ and $g_2(x) = x^7 + 1224488$ such that $x - g_2(g_1(x)) = f(x)/x^{49}$ where $f(x) = x^{50} + 886535x^{49} + 2111022$. The polynomial $f(x)$ is irreducible over $\mathbb{F}_p$ and we represented $\mathbb{F}_{p^n}$ as $\mathbb{F}_p[x]/(f(x))$. Under this representation, $x + 11$ turned out to be a primitive element and was taken as the base of our discrete logarithm computation.

The factorization of $(p^n - 1)/(p - 1)$ is the following.

$$\begin{aligned}\frac{p^n-1}{p-1} &= 2^5 \cdot 3^4 \cdot 11 \cdot 31 \cdot 83 \cdot 101 \cdot 131 \cdot 157 \cdot 251 \cdot 6361 \cdot 12241 \cdot 131939 \cdot 839532251 \cdot 896407381 \cdot \\ &\quad 3943088101 \cdot 164534375651 \cdot 3062950366849991 \cdot 36244934276573651 \cdot \\ &\quad 752902385776306150901 \cdot p_1 \cdot p_2 \cdot p_3\end{aligned}$$

where

$$\begin{aligned}p_1 &= 204692161033930730108568803278296327232200 1; \\ p_2 &= 553059810014751323913181174167985322787847067 51; \\ p_3 &= 923983173059841394501412330899341649381951887 56\backslash \\ &\quad 2818470664758881417032041985737711765745606292778672234895 1.\end{aligned}$$

Note $\lceil \log_2(p_1) \rceil = 141$, $\lceil \log_2(p_1) \rceil = 156$ and $\lceil \log_2(p_1) \rceil = 353$.

Based on the choices of $g_1(x)$ and $g_2(x)$, we have $y = g_1(x) = x^{-7}$ and $x = g_2(y) = y^7 + 1224488$. The factor basis was set to be $\mathbb{B} = \{(x + a_i), (y + b_j) : a_i, b_j \in \mathbb{F}_p\}$.

**Relation collection:** The relation collection was done using sieving based on partial smoothness-cum-divisibility technique combined with pinpointing. The computation is highly parallelisable. It was distributed on four servers with 90 processes per server. The total time required for the relation collection phase was about 25 hours. Assuming that our jobs were allocated about 75% of the server time, a rough estimate of the number of core-years required for relation collection is 0.53 core-years.

A total of $2p + 100 = 4222146$ relations were generated among the elements of the factor basis which includes the relation $y = x^{-7}$. Except for $(y + 1849709)$, all elements of the factor basis were involved in at least one relation.

The fact that none of the $2p + 100$ relations involved $(y + 1849709)$ seemed peculiar to us. So, we decided to investigate this further. For this we considered applying the partial smoothness-cum-divisibility technique for relation collection from the $y$-side. The starting point of this technique is to write

$$C(y) = (y - \alpha_1)(y - \alpha_2)(y - \alpha_3)H_1(y).$$

We set $(y - \alpha_1) = (y + 1849709)$. This is to ensure that any relation obtained from $C(y)$ and the corresponding $F(x)$ will necessarily involve $(y + 1849709)$. There are two degrees of freedom given by $\alpha_1$ and $\alpha_2$. This allows trying $p^2$ options. We were surprised to find that no relation could be obtained. It was possible to ensure that $C(y)$ is smooth. However, in each such case, it turned out that the corresponding $F(x)$ is not smooth. This suggests that it may indeed be the case that there is no relation among the factor basis elements which involves $(y + 1849709)$. Conventionally, it is assumed that the relation collection phase will provide relations involving all elements of the factor basis. This particular example suggests that this may not be true.

Since we were unable to obtain any relation involving $(y + 1849709)$, we decided to proceed without this element. The resulting matrix for the linear algebra stage consisted of $2p + 100$ relations involving $2p - 1$ unknowns.

**Linear algebra:** The linear algebra step was performed for the three primes $p_1$, $p_2$ and $p_3$. The block Wiedemann algorithm implemented in the CADO-NFS software was used to complete the linear algebra step. For the largest prime $p_3$, the Krylov step took about 1.6 core years, Lingen required negligible time and Mksol required about 0.25 core years. The time requirements for the two smaller sized primes were lesser. In terms of space, for $p_1$ and $p_2$ about 29GB each was required and for $p_3$ about 53GB was required. We used $n = 4$ distinct sequences for the block Wiedemann algorithm, so that we were able to simultaneously use 16 nodes as 4 groups of 4 nodes, each group working on one sequence.

After the linear algebra step, the discrete logarithms of all the elements in the factor basis other than $(y + 1849709)$ were obtained modulo $p_1$, $p_2$ and $p_3$.

**Individual logarithm - descent to factor basis elements:** As the target for the individual discrete logarithm computation we chose the following element derived from the digits of the real number $\pi$. The function 'Normalize' mentioned below makes the input polynomial monic by multiplying with the inverse of the leading coefficient.

$$\Pi(x) \quad = \quad \text{Normalize}\left(\sum_{0}^{(n-1)} \lfloor \pi \cdot p^{i+1} \mod p \rfloor x^i\right).$$

Explicitly $\Pi(x)$ is given by the following degree 49 polynomial.

$$\begin{aligned}
\Pi(x) \quad = \quad & x^{49} + 308380x^{48} + 467398x^{47} + 934029x^{46} + 37835x^{45} + 2003442x^{44} + 174801x^{43} \\
& +1414683x^{42} + 733114x^{41} + 1077558x^{40} + 1049867x^{39} + 1848765x^{38} + 1653554x^{37} \\
& +949244x^{36} + 1627181x^{35} + 1592837x^{34} + 652981x^{33} + 1601022x^{32} + 635134x^{31} \\
& +900855x^{30} + 413911x^{29} + 74385x^{28} + 2057944x^{27} + 930210x^{26} + 310181x^{25} + \\
& +118528x^{24} + 1515849x^{23} + 93830x^{22} + 393848x^{21} + 644073x^{20} + 1018627x^{19} \\
& +1654544x^{18} + 611872x^{17} + 1491385x^{16} + 1797395x^{15} + 1833421x^{14} \\
& +1711611x^{13} + 406154x^{12} + 1588768x^{11} + 530413x^{10} + 1458736x^{9} \\
& +696502x^{8} + 496320x^{7} + 196737x^{6} + 535254x^{5} + 194167x^{4} \\
& +977109x^{3} + 1911333x^{2} + 1037166x + 1347394.
\end{aligned}$$

For the initial descent a simple randomisation strategy was utilised which led to

$$\Pi(x) \quad = \quad \frac{N(x)}{D(x)}$$

where $N(x)$ and $D(x)$ are as follows.

$$\begin{aligned}
N(x) \quad = \quad & (x + 1424244)(x^3 + 237998x^2 + 42029x + 734901)(x^3 + 299760x^2 + 1210894x + 1086517) \\
& (x^4 + 1182727x^3 + 563430x^2 + 1055902x + 1247639) \\
& (x^4 + 1251838x^3 + 723661x^2 + 1707546x + 110202) \\
& (x^5 + 221654x^4 + 445454x^3 + 650438x^2 + 1275751x + 124811) \\
& (x^5 + 665157x^4 + 337641x^3 + 1409401x^2 + 1379166x + 322114) \\
& (x^5 + 927040x^4 + 199439x^3 + 342445x^2 + 1316050x + 1494757) \\
& (x^6 + 61134x^5 + 1695168x^4 + 2017581x^3 + 293438x^2 + 766784x + 1054073) \\
& (x^6 + 1565656x^5 + 129255x^4 + 419731x^3 + 1556013x^2 + 2087232x + 207329) \\
& (x^7 + 1746884x^6 + 469847x^5 + 382378x^4 + 425150x^3 + 944772x^2 + 530084x + 1756060), \\
D(x) \quad = \quad & (x + 3541)(x + 87748)(x + 110850)(x + 119667)(x + 241035)(x + 305058) \\
& (x + 395128)(x + 399638)(x + 422176)(x + 578119)(x + 582549)(x + 586316) \\
& (x + 662109)(x + 770637)(x + 772129)(x + 775849)(x + 800910)(x + 865556) \\
& (x + 902073)(x + 971438)(x + 1011431)(x + 1052833)(x + 1060253)(x + 1062580) \\
& (x + 1103078)(x + 1132166)(x + 1147933)(x + 1174406)(x + 1176644)(x + 1189750) \\
& (x + 1231997)(x + 1248106)(x + 1289845)(x + 1297742)(x + 1347100)(x + 1418494) \\
& (x + 1433230)(x + 1528574)(x + 1579870)(x + 1596791)(x + 1660898)(x + 1741103) \\
& (x + 1805530)(x + 1849061)(x + 1912058)(x + 2041324).
\end{aligned}$$

Note that $N(x)$ is 7-smooth while $D(x)$ is smooth. This decomposition required about 30 minutes using 50 processes.

In the next step, the goal was to reduce to quadratic polynomials using successive $d$-$(d-1)$ descent for polynomials of degree $d > 2$. This strategy mostly succeeded, except for three cubic polynomials. For these polynomials, we had to resort to alternating walk, i.e., move from degree 3 polynomials in $x$ to degree 3 polynomials in $y$, as explained earlier. Using such a walk, we were able to descend to quadratic and linear polynomials. The total number of quadratic polynomials that were generated was 1212, of which 673 were quadratic polynomials in $x$ and 539 were quadratic polynomials in $y$. The total time required for descending to quadratic polynomials was about 1481 minutes using 50 processes.

Finally we performed the 2-1 descent on all the quadratic polynomials. This was the most time consuming of all the descent steps. We used 50 processes on each of the four servers to perform either direct 2-1 descent or to apply alternating walk and/or branching. The entire 2-1 descent step was automated. The total time for all the 2-1 descents required about 10 days. Assuming that our jobs were allocated about 75% of the server time, a rough estimate of the number of core-years required for all the 2-1 descents is about 4.1 core-years.

*Remark:* The descent to quadratic polynomials resulted in a total of 1212 quadratic polynomials. For comparison, we mention the number of quadratic polynomials obtained in previous computations. In [19], the numbers of quadratic polynomials were 92 and 59 for the 592-bit and the 728-bit cases respectively. In [14], the number of quadratic polynomials for the 1125-bit case was 278; the number of quadratic polynomials for the 1425-bit case was not reported.

We note that the descent is a random procedure. So, different runs of the descent procedure may lead to different number of quadratic polynomials. Since the 2-1 descent is the most time consuming of all the descent steps, it would be worthwhile to try and minimise the number of quadratic polynomials that arise from the upper levels of the descent. There is, however, no known method for such minimisation.

**Individual logarithm - final discrete logarithm computation:** After the completion of the 2-1 descent, the target polynomial $\Pi(x)$ was expressed as a ratio $N_1(x)/D_1(x)$, where $N_1(x)$ is a product of 147126 linear polynomials in $x$ and 127149 linear polynomials in $y$, and $D_1(x)$ is a product of 147164 linear polynomials in $x$ and 126001 linear polynomials in $y$.

Recall that the relation collection and linear algebra steps were not able to compute the discrete logarithm of $(y + 1849709)$. Fortunately, this element does not appear among the linear factors of $N_1(x)$ and $D_1(x)$.

The discrete logarithms modulo $p_1$, $p_2$ and $p_3$ of the linear factors of $N_1(x)$ and $D_1(x)$ had already been obtained after the completion of the linear algebra step. Consequently, after the descent step we were able to obtain the discrete logarithm of $\Pi(x)$ modulo $p_1$, $p_2$ and $p_3$.

We used Pollard rho and Pohlig-Hellman to compute the discrete logarithm of $\Pi(x)$ modulo the smaller factors of $p^n - 1$. The final discrete logarithm of $\Pi(x)$ to base $(x + 11)$ was computed using the Chinese Remainder Theorem. This value is given below.

$$\begin{aligned}\log_{x+11}(\Pi(x)) \;=\; & 1323496538911863968895271989039865754003499138979788669347646690861304065811174\backslash \\ & 1258084796458565453948561234741578427747721119813849133097458001127822232655615\backslash \\ & 0735099096613330104434651232074005278625612674879570628049934937631130006839219\backslash \\ & 5452506485478263044561377117997258194255748683503064110129248778733465564209650 1.\end{aligned}$$

A short Magma program to verify the discrete logarithm is given in the Appendix A.

## 5 Unsolved DLP Challenge for the Medium Prime Case

In [17], the following has been stated.

> "For powers of very small primes and for large prime fields the function-field sieve and the number-field sieve are highly optimized; for intermediate fields algorithms with the same asymptotic behavior exist but the actual running times are slower."

To encourage research for intermediate size fields, the following challenge has been proposed in [17]. Solve DLP in $\mathbb{F}_{p^{17}} = \mathbb{F}_p[x]/(x^{17} - 2)$ where $p = 2^{32} - 27897$.

This problem can be tackled using the FFS for the medium prime case. For this, we set $n_1 = n_2 = 4$ so that $n = n_1 n_2 + 1 = 17$. We estimate the costs of the relation collection, linear algebra and the descent steps.

*Cost of linear algebra:* For this problem, $n \mid (p-1)$ which will allow the factor basis to be reduced by a factor of 17. So, the size of the factor basis will be about $2^{29}$ and hence, linear algebra step will require about $2^{58}$ operations in $\mathbb{F}_p$.

*Cost of relation collection:* Using $n_1 = n_2 = 4$, from (6), the number of trials required to obtain a single relation is about $2^{6.9}$. So, the total number of trials required to obtain about $2^{29}$ relations is about $2^{35.9}$. Further, the feasibility condition given by (4) holds.

*Cost of 2-1 descent:* Using $n_1 = n_2 = 4$, from (7), the number of trials required to obtain a single 2-1 descent is about $2^{9.5}$. We would expect a few thousand quadratic polynomials would be required to be descended. The feasibility condition given by (8) holds.

So, based on the above analysis, we see that while the relation collection and the descent steps are well within reach, the major cost of performing the computation required for solving the challenge lies in the linear algebra computation.

*Remark:* The above estimates are rough. For one thing, we have estimated that the time required for the linear algebra step on a factor basis of size $N$ is about $N^2$. Secondly, the time for individual operations for the linear algebra step and the times for individual trials of the relation collection step and the 2-1 descent step have been assumed to be equal. To obtain more precise estimates, these issues would be required to be taken into consideration. Nevertheless, even with the rough estimates, the main conclusion that for this particular DLP computation it is the linear algebra step which will be the main challenge, remains valid.

**Larger extension degree:** The extension degree suggested in the above problem is 17 which is quite low. Let us consider a higher value of $n$. Suppose $n_1 = n_2 = 8$ and $n = n_1 n_2 + 1 = 65$. As in the above problem, assume that $p$ is a 32-bit prime. Also, let us not make the assumption that $n \mid (p-1)$ holds. So, the factor basis will have about $2p$ elements. From (6), the number of trials required to obtain a single relation is about $2^{18}$ and to obtain about $2^{33}$ relations, about $2^{51}$ trials would be required. From (7), the number of trials required to obtain a single 2-1 descent is about $2^{31}$. The feasibility condition (4) and (8) both hold. Since the size of the factor basis is about $2^{33}$, the linear algebra will require about $2^{66}$ $\mathbb{F}_p$-operations. So, it is the linear algebra step which will be the major challenge in any such discrete logarithm computation.

## 6 Conclusion

In this paper, we have reported the computation of discrete logarithm in a 1051-bit field having a 22-bit characteristic. For the general medium prime case (i.e., fields for which the condition $n \mid (p-1)$ does not hold), the computation reported here is the current discrete logarithm record computation. The techniques used in this work can be extended to solve relation collection and descent phases for 32-bit primes and moderate extension degrees. It is the linear algebra phase that will require the maximum time for performing any record discrete logarithm computation for such fields.

## Acknowledgement

The linear algebra part of the reported calculation was carried out on the French Grid'5000/SILECS experimental testbed, supported by a scientific interest group hosted by Inria and including CNRS,

RENATER and several universities as well as other organizations. Some of the computational resources used on the Grid'5000/SILECS testbed were also funded by the CPER cyberentreprises project.

## References

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

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

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

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

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

[6] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Trans. Information Theory*, 22(6):644–654, 1976.

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

[8] Faruk Göloglu, Robert Granger, Gary McGuire, and Jens Zumbrägel. Solving a 6120-bit DLP on a desktop computer. In Tanja Lange, Kristin E. Lauter, and Petr Lisonek, editors, *Selected Areas in Cryptography - SAC 2013 - 20th International Conference, Burnaby, BC, Canada, August 14-16, 2013, Revised Selected Papers*, volume 8282 of *Lecture Notes in Computer Science*, pages 136–152. Springer, 2013.

[9] Daniel M. Gordon. Discrete logarithms in GF($p$) using the number field sieve. *SIAM J. Discrete Math.*, 6(1):124–138, 1993.

[10] Robert Granger, Thorsten Kleinjung, and Jens Zumbrägel. Breaking '128-bit secure' supersingular binary curves – (or how to solve discrete logarithms in $\mathbb{F}_{2^{4\cdot 1223}}$ and $\mathbb{F}_{2^{12\cdot 367}}$). In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part II*, volume 8617 of *Lecture Notes in Computer Science*, pages 126–145. Springer, 2014.

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

[12] Robert Granger, Thorsten Kleinjung, and Jens Zumbrägel. Discrete logarithms in $GF(2^{30750})$. *NMBRTHRY list*, January 2019.

[13] Antoine Joux. *Algorithmic Cryptanalysis*. Cryptography and Network Security. Chapman & Hall/CRC, 2009.

[14] Antoine Joux. Faster index calculus for the medium prime case application to 1175-bit and 1425-bit finite fields. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques, Eurocrypt*, pages 177–193. Springer, 2013.

[15] Antoine Joux and Reynald Lercier. The function field sieve is quite special. In Claus Fieker and David R. Kohel, editors, *ANTS*, volume 2369 of *Lecture Notes in Computer Science*, pages 431–445. Springer, 2002.

[16] Antoine Joux and Reynald Lercier. The function field sieve in the medium prime case. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques, Eurocrypt*, pages 254–270. Springer, 2006.

[17] Tanja Lange. Digital signature: DSA with medium fields. https://www.mysterytwisterc3.org/images/challenges/mtc3-lange-01-dsasig-en.pdf, 2011.

[18] National Institute of Standards and Technology. Digital Signature Algorithm. https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf, 2013.

[19] Palash Sarkar and Shashank Singh. Fine tuning the function field sieve algorithm for the medium prime case. *IEEE Transactions on Information Theory*, 62(4):2233–2253, 2016.

[20] W. A. Stein et al. *Sage Mathematics Software*. The Sage Development Team, 2013. http://www.sagemath.org.

[21] The CADO-NFS Development Team. CADO-NFS, an implementation of the number field sieve algorithm. Development version, August 2019.

# A Magma Script to Verify the Computation

```
n := 50;
p := 2111023;
assert(IsPrime(p));
Fp := GF(p);
FpX<X> := PolynomialRing(Fp);
fX := X^50 + 886535*X^49 + 2111022;
assert IsIrreducible(fX);
Fpnx<x> := ext<Fp|fX>;

RR:=RealField();
pi := Normalize(&+[(Floor(Pi(RR)*p^(i+1)) mod p)*X^i : i in [0..n-1]]);

log := 13234965389118639688952719890398657540034991389797886693476466 90\
       86130406581117412580847964585654539485612347415784277477211198 13\
       84913309745800112782223265561507350990966133301044346512320740 05\
       27862561267487957062804993493763113000683921954525064854782630 44\
       56137711799725819425574868350306411012924877873346556420965 01;
base := x+11;
target := Fpnx!Eltseq(pi);

printf "base := %o\n",base;
printf "target := %o\n",target;
printf "log := %o\n", log;
printf "base^log eq target = %o\n",base^log eq target;
```