Solving a 6120-bit DLP on a Desktop Computer*

Faruk Göloğlu, Robert Granger, Gary McGuire, and Jens Zumbrägel

Complex & Adaptive Systems Laboratory and
School of Mathematical Sciences
University College Dublin, Ireland
{farukgologlu,robbiegranger}@gmail.com, {gary.mcguire,jens.zumbragel}@ucd.ie

Abstract. In this paper we show how some recent ideas regarding the discrete logarithm problem (DLP) in finite fields of small characteristic may be applied to compute logarithms in some very large fields extremely efficiently. In particular, we demonstrate a practical DLP break in the finite field of 2^{6120} elements, using just a single core-month.

Keywords: Discrete logarithm problem, binary finite fields

1 Introduction

The understanding of the hardness of the DLP in the multiplicative group of finite extension fields could be said to be undergoing a mini-revolution. It began with Joux's 2012 paper in which he introduced a method of relation generation dubbed 'pinpointing', which reduces the time required to obtain the logarithms of the elements of the factor base [8]. For medium-sized base fields, this technique has heuristic complexity as low as $L_{q^n}(1/3, 2/3^{2/3}) \approx L_{q^n}(1/3, 0.961)$, significantly improving upon the previous best by Joux and Lercier [12] $L_{q^n}(1/3, 3^{1/3}) \approx L_{q^n}(1/3, 1.442)$. To demonstrate the practicality of this approach, Joux solved two example DLPs in fields of bitlength 1175 and 1425 respectively, both with prime base fields.

Soon afterwards Göloğlu, Granger, McGuire, and Zumbrägel showed that in the context of binary fields (and more generally small characteristic fields), finding relations for the factor base can be *polynomial time* in the size of the field [3]. By extending the basic idea to eliminate degree two elements during the descent phase, for medium-sized base fields an heuristic complexity as low as $L_{q^n}(1/3, (2/3)^{2/3}) \approx L_{q^n}(1/3, 0.763)$ was achieved; this approach was demonstrated via the solution of the DLP in the field $\mathbb{F}_{2^{1971}}$ [5], and in the field $\mathbb{F}_{2^{3164}}$.

After the initial publication of [3], Joux released a preprint [9] detailing an algorithm for solving the discrete logarithm problem for fields of the form $\mathbb{F}_{q^{2n}}$, with $q = p^k$ and $n \approx q$, which was used in the solving of a DLP in $\mathbb{F}_{2^{1778}}$ [10], and later in $\mathbb{F}_{2^{4080}}$ [11]. This algorithm has heuristic complexity $L(1/4 + o(1))$, and also has an heuristic polynomial time relation generation method, similar in principle to that in [3]. While the degree two element elimination in [3] is arguably superior, for other small degrees, Joux's elimination method is faster, resulting in the stated complexity.

In this paper we explain in detail how these new ideas may be combined to compute discrete logarithms in some large finite fields very efficiently. Indeed, we explain the details of the algorithms used in the world record discrete logarithm computation in the finite field $\mathbb{F}_{2^{6120}}$ [4]. We emphasise that this work is but an initial foray into the behaviour and performance of the new techniques, and we expect many more developments both in terms of our algorithmic understanding, and larger computations, in due course.

* Research supported by the Claude Shannon Institute, Science Foundation Ireland Grant 06/MI/006. The fourth author was in addition supported by SFI Grant 08/IN.1/I1950.

The remainder of the paper is organised as follows. Section 2 explains the algorithm in detail. Section 3 concentrates on the practical issues regarding the computation. Finally, Section 4 gives the details of a discrete logarithm computation in $\mathbb{F}_{2^{6120}}$.

2 The algorithm

The following describes the index calculus method that we use for our discrete logarithm computation.

2.1 Setup

Let k and k' be positive integers, $\ell := kk'$, $q := 2^\ell$, and $n := 2^k - 1$. We construct the finite field $\mathbb{F}_{q^n}$ of bit length $\ell n = kk'(2^k - 1)$ in which we solve the DLP as follows.[1]

We consider $\mathbb{F}_q$ as an extension of $\mathbb{F}_{2^k}$ of degree k'. Then we choose $\gamma \in \mathbb{F}_q$ such that the polynomial $X^n + \gamma$ is irreducible over $\mathbb{F}_q$ and define $\mathbb{F}_{q^n}$ as the Kummer extension

$$\mathbb{F}_q(x) \cong \mathbb{F}_q[X]/\langle X^n + \gamma \rangle ,$$

where x is the root of the polynomial $X^n + \gamma$ in $\mathbb{F}_{q^n}$. Note that a Kummer extension of degree n over $\mathbb{F}_q$ exists if and only if $n \mid q-1$. Throughout the paper, the upper case letters $X, W, \ldots$ are used for indeterminates and the lower case letters $x, w, \ldots$ are reserved for the roots of polynomials.

The following table displays the bit length ℓn of the finite field $\mathbb{F}_{q^n}$ for various choices of the numbers k and k'.

$k' \setminus k$	6	7	8	9
3	1134	2667	6120	13797
4	1512	3556	8160	18396
5	1890	4445	10200	22995
6	2268	5334	12240	27594

In Section 4, we will give the details of the discrete logarithm computation when $\ell n = 6120$. The algorithm we explain in this section can be applied in principle to any of the above parameters. However, for a very fast degree 2 elimination, some of the above parameters (including 6120) are more suitable.

2.2 Factor base and automorphisms

The factor base we use consists of the elements in $\mathbb{F}_{q^n}$ which have degree 1 in the polynomial representation over $\mathbb{F}_q$, i.e., we consider the set $\{x+a \mid a \in \mathbb{F}_q\}$. As noted in [12, 8, 3] factor base preserving automorphisms of $\mathbb{F}_{q^n}$ can be used to drastically reduce the number of variables involved in the linear algebra step. Indeed, the map $\sigma := \mathrm{Frob}^k : \alpha \to \alpha^{2^k}$ satisfies $\sigma(x) = \gamma x$ with $\gamma \in \mathbb{F}_q$, and thus preserves the factor base. Furthermore, for $\varphi := \sigma^{k'} = \mathrm{Frob}^\ell : \alpha \to \alpha^q$ we have $\varphi(x) = \mu x$ with $\mu \in \mathbb{F}_{2^k}$ a primitive n-th root of unity, and thus we find

$$(x + a)^{2^{j\ell+ik}} = \sigma^{k'j+i}(x + a) = \sigma^i(\varphi^j(x + a)) = \sigma^i(\mu^j x + a) = \mu^j \gamma^{e_i} x + a^{2^{ki}} ,$$

[1] Our choice of representation of the finite field $\mathbb{F}_{q^n}$ will be advantageous for our method to solve the discrete log problem. Note that it is a computationally easy problem to switch between two different representations of a finite field [15].

where $e_0 = 0$ and $e_i = 2^k e_{i-1} + 1$ for $1 \le i \le k'$; thus it follows

$$\log\left(x + \frac{a^{2^{ki}}}{\mu^j \gamma^{e_i}}\right) = 2^{j\ell+ik} \log(x+a)$$

for all $0 \le j < n$ and $0 \le i < k'$.

The automorphism σ generates a group of order $k'n$, which acts on the set of q factor base elements, thus dividing the factor base into about N orbits, where $N \approx \frac{q}{k'n} \approx \frac{1}{k'} 2^{\ell-k}$ is the number of variables to consider.

2.3 Relation generation

In order to generate relations between the factor base elements we use a method based on Case $n = 2^k - 1$ in [3]. We consider polynomials of the form

$$F_B(X) := X^{2^k+1} + BX + B\,,$$

which have been studied by Bluher [1] and Helleseth/Kholosha [7]. We recall in particular the following result of Bluher [1] (see also [7, 3]):

Theorem 1. *The number of elements $B \in \mathbb{F}_q^*$ such that the polynomial $F_B(X)$ splits completely over $\mathbb{F}_q$ equals*

$$\frac{2^{\ell-k}-1}{2^{2k}-1} \quad \textit{if } k' \textit{ is odd}\,, \qquad \frac{2^{\ell-k}-2^d}{2^{2k}-1} \quad \textit{if } k' \textit{ is even}\,.$$

Let $B \in \mathbb{F}_q^*$ be an element such that $F_B(X)$ splits and denote its roots by μ_i, where $i = 1, \ldots, 2^k+1$. For arbitrary $a, b \in \mathbb{F}_q$ (with $a^{2^k} \neq b$) there exists $c \in \mathbb{F}_q$ with $(a^{2^k}+b)^{2^k+1} = B\,(ab+c)^{2^k}$ and we then find that

$$f(X) := F\Big(\frac{ab+c}{a^{2^k}+b}\,X + a\Big) = X^{2^k+1} + aX^{2^k} + bX + c$$

and that $f(X)$ also splits over $\mathbb{F}_q$, with roots $\nu_i := \frac{ab+c}{a^{2^k}+b}\,\mu_i + a$.

Now by definition of $\mathbb{F}_{q^n}$ we have $x^n = \gamma$ and thus $x^{2^k} = \gamma x$, where $\gamma \in \mathbb{F}_q$. Hence in $\mathbb{F}_{q^n}$ there holds

$$f(x) = \gamma x^2 + a\gamma x + bx + c = \gamma(x^2 + (a + \tfrac{b}{\gamma})x + \tfrac{c}{\gamma}) = \gamma g(x)\,,$$

where $g(X) := X^2 + (a + \frac{b}{\gamma})X + \frac{c}{\gamma}$. Hence, if this polynomial also splits, say $g(X) = (X+\xi_1)(X+\xi_2)$, which occurs with probability $\frac{1}{2}$, then we find a relation of factor base elements, namely

$$\prod_{i=1}^{2^k+1} (x+\nu_i) = \gamma(x+\xi_1)(x+\xi_2)\,.$$

Such a relation corresponds to a linear relation among the discrete logarithms of the factor base elements. Once we have found more than N relations we can solve the discrete logarithms of the factor base elements by means of linear algebra (see Subsection 3.3).

2.4 Individual logs

After the logarithms of the factor base elements have been found, a general individual discrete logarithm can be computed, as is common, by a descent strategy. The basic idea of this method is trying to write an element, given by its polynomial representation over $\mathbb{F}_q$, as a product in $\mathbb{F}_{q^n}$ of factors represented by lower degree polynomials. By applying this principle repeatedly a descent tree is constructed, and we can eventually express a given target element by a product of factor base elements. This will enable us to obtain the discrete logarithm of the target element easily.

While for larger degree polynomials it is computationally relatively easy to find an expression involving lower degree polynomials by a standard approach, this method becomes increasingly less efficient as the degree becomes smaller. In addition, the number of small degree polynomials in the descent tree grows significantly with lower degree. We therefore propose new methods for degree 2 elimination and small degree descent, which are inspired by the recent works [3] and [9] respectively.

Degree 2 elimination Given a polynomial $Q(X) := X^2 + q_1X + q_0 \in \mathbb{F}_q[X]$ we aim at expressing the corresponding finite field element $Q(x) \in \mathbb{F}_{q^n}$ as a product of factor base elements.

Our basic approach is to find $a, b, c \in \mathbb{F}_q$ such that, up to a multiplicative constant in $\mathbb{F}_q$, $Q(x) = x^2+q_1x+q_0$ equals $x^{2^{k+1}}+ax^{2^k}+bx+c$ where the polynomial $X^{2^{k+1}}+aX^{2^k}+bX+c$ splits into linear factors (compare with [3, Sec. 5]).

As $x^n = \gamma$ holds, we have $x^{2^{k+1}} + ax^{2^k} + bx + c = \gamma(x^2 + (a + \frac{b}{\gamma})x + \frac{c}{\gamma})$ and comparing coefficients we find $\gamma q_0 = c$ and $\gamma q_1 = \gamma a + b$. Now letting $B \in \mathbb{F}_q^*$ be an element satisfying the splitting property of Theorem 1 and combining the previous equations with $\left(a^{2^k} + b\right)^{2^k+1} = B\,(ab + c)^{2^k}$ we arrive at the condition

$$(a^{2^k} + \gamma a + \gamma q_1)^{2^k+1} + B(\gamma a^2 + \gamma q_1 a + \gamma q_0)^{2^k} = 0\,.$$

Considering $\mathbb{F}_q$ as a degree k' extension over $\mathbb{F}_{2^k}$ this equation gives a quadratic system in the k' components of a, which can be solved very fast by a Gröbner basis method.

Heuristically, for each of the above B's the probability of success of this method, i.e., when an $a \in \mathbb{F}_q$ as above exists, is $\frac{1}{2}$. Note that if $k' = 3$ there is just one single B in the context of Theorem 1, and so this direct method fails in half of the cases. However, this issue can be resolved under certain circumstances, e.g., if $k = 8$, as will be explained in Subsection 4.4.

Small degree descent The following describes the Gröbner basis descent of Joux [9] applied in the context of the polynomials $F_B(X) = X^{2^k+1} + BX + B$ of Theorem 1. Let $g(X)$ and $h(X)$ be polynomials over $\mathbb{F}_q$ of (low) degree δ. We substitute X by the rational function $\frac{g(X)}{h(X)}$ and thus find that the polynomial

$$P(X) := g(X)^{2^k+1} + g(X)\,h(X)^{2^k} + h(X)^{2^k+1}$$

factors into polynomials of degree at most δ. Since $x^{2^k} = \gamma x$ holds in $\mathbb{F}_{q^n}$ the element $P(x)$ can also be represented by a polynomial of degree 2δ.

Now given a polynomial $Q(X) \in \mathbb{F}_q[X]$ of degree 2δ or $2\delta - 1$ to be eliminated we consider the equation $P(x) = Q(x)$ (or $P(x) = Q(x)(x + a)$), which results as above in a quadratic system in variables over $\mathbb{F}_{2^k}$ representing the coefficients of $g(X)$ and $h(X)$ in $\mathbb{F}_q$. By solving this system with a Gröbner basis algorithm we can perform the descent.

Large degree descent This part of the descent is somewhat classical (see [12] for example), but includes the degree balancing technique described in [3, Sec. 4], which makes the descent far more rapid when the base field $\mathbb{F}_q$ is a degree k' extension of a non-prime field. In the finite field $\mathbb{F}_{q^n}$ we let $y := x^{2^k}$ and $\bar{x} := x^{2^{k-a}}$ for some suitably chosen integer $1 < a < k$. Then $y = \bar{x}^{2^a}$ and $\bar{x} = (\frac{y}{\gamma})^{2^{k-a}}$ holds. Now for given $Q(X) \in \mathbb{F}_q[X]$ of degree d representing $Q(y)$ we consider the lattice

$$L := \left\{(w_0, w_1) : Q(X) \mid (\tfrac{X}{\gamma})^{2^{k-a}} w_0(X) + w_1(X)\right\} \subseteq \mathbb{F}_q[X]^2 \,.$$

By Gaussian lattice reduction we find a basis (u_0, u_1), (v_0, v_1) of L of degree $\approx \frac{d}{2}$ and can thus generate lattice elements $(w_0, w_1) = r(u_0, u_1) + s(v_0, v_1)$ of low degree. In $\mathbb{F}_{q^n}$ we then consider the equation

$$\bar{x}w_0(\bar{x}^{2^a}) + w_1(\bar{x}^{2^a}) = \bar{x}w_0(y) + w_1(y) = (\tfrac{y}{\gamma})^{2^{k-a}} w_0(y) + w_1(y) \,,$$

where the right-hand side is divisible by $Q(y)$ by construction, and a is chosen so as to make the degrees of both sides as close as possible. The descent is successful whenever a lattice element (w_0, w_1) is found such that the involved polynomials $Xw_0(X^{2^a}) + w_1(X^{2^a})$ and $\frac{1}{Q(x)}(X^{2^{k-a}} w_0(X) + \gamma^{2^{k-a}} w_1(X))$ are $(d-1)$-smooth, i.e., have only factors of degree less than d.

3 Practical considerations

3.1 Factorisation of the group order

The factorisation of the group order $|\mathbb{F}_{q^n}^*| = 2^{\ell n} - 1$ is of interest for several reasons. First it indicates the difficulty of solving the associated DLP using the Pohlig-Hellman algorithm. It is also required for proving that some element is a generator of the group. Finally, we need it to determine the small factors, for which we apply Pollard's rho method, and the large factors for the index calculus method.

However, the number $2^{\ell n} - 1$ cannot always be completely factored in a reasonable time. In this case it is vital to know at least all the small prime factors of the group order. We remark that the factorisation problem of the number $2^{\ell n} - 1$ can be slightly simplified by using the identity

$$2^{\ell n} - 1 = \prod_{d \mid \ell n} \Phi_d(2) \,,$$

where $\Phi_d \in \mathbb{Z}[x]$ denotes the d-th cyclotomic polynomial and d runs through all divisors of ℓn.

3.2 Pohlig-Hellman and Pollard's rho method

In order to compute a discrete logarithm in a group G of order m we can use any factorisation of $m = m_1 \cdot \ldots \cdot m_r$ into pairwise coprime factors m_i and compute the discrete log modulo each factor. Indeed, if we are to compute $z = \log_\alpha \beta$ it suffices to compute $\log_{\alpha^{c_i}} \beta^{c_i}$ with $c_i = m/m_i$, which determines $z \bmod m_i$. With the information of $z \bmod m_i$ for all i one easily determines $z \pmod{m}$ by the Chinese Remainder Theorem.

For the small prime (power) factors of m we use Pollard's rho method to compute the discrete logarithm modulo each factor. Regarding the large factors of m we find it most efficient to combine them into a single product m_*, so that in the linear algebra step of the index calculus method we work over the ring $\mathbb{Z}_{m_*}$. Note that each iteration of the Lanczos method that we use for the linear algebra problem requires the inversion of a random element in $\mathbb{Z}_{m_*}$; this is the reason why we actually have to separate the small factors of the group order from the large ones.

3.3 Linear algebra

The relation generation phase of the index calculus method produces linear relations among the logarithms of the factor base elements. As the factor base logs are also related by the automorphism group as explained in Subsection 2.2 the number N of variables is reduced and the linear relations will have coefficients being powers of 2. Once $M > N$ relations have been generated we have to find a nonzero solution vector for the linear system. To ensure that the matrix is of maximal rank $N-1$ we generate $M \approx N + 100$ relations. As noted earlier the number of variables N is expected to be about $\frac{2^\ell}{k'(2^k-1)} \approx \frac{2^{\ell-k}}{k'}$.

We let B be the $M \times N$ matrix of the relations' coefficients, which is a matrix of constant row-weight $2^k + 3$. We have to find a nonzero vector v of length N such that $Bv = 0$ modulo m_*, the product of the large prime factors of the group order m.

A common approach in index calculus algorithms is to reduce the matrix size at this stage by using a structured Gaussian elimination (SGE) method. In our case, however, the matrix is not extremely sparse while its size is quite moderate, hence the expected benefit from SGE would be minimal and we refrained from this step.

We use the iterative Lanczos method [14, 13] to solve the linear algebra problem, which we briefly describe here. Let $A = B^t B$, which is a symmetric $N \times N$ matrix. We let $v \in \mathbb{Z}_{m^*}^N$ be random, $w = Av$, and find a vector $x \in \mathbb{Z}_{m^*}^N$ such that $Ax = w$ holds (since $A(x-v) = 0$ we have thus found a kernel element). We compute the following iteration

$$w_0 = w\,, \qquad v_0 = Aw_0\,, \qquad w_1 = v_0 - \frac{(v_0, v_0)}{(v_0, w_0)} w_0$$
$$v_i = Aw_i\,, \qquad w_{i+1} = v_i - \frac{(v_i, v_i)}{(v_i, w_i)} w_i - \frac{(v_i, v_{i-1})}{(v_{i-1}, w_{i-1})} w_{i-1}$$

and stop once $(v_j, w_j) = 0$; if $w_j \neq 0$ the algorithm fails, otherwise we find the solution vector

$$x = \sum_{i=0}^{j-1} \frac{(w, w_i)}{(v_i, w_i)} w_i\,.$$

Performing the above iteration consists essentially of several matrix-vector products, scalar-vector multiplications, and vector-vector inner products. As the matrix is sparse and consists of entries being powers of 2 the matrix-vector products can be carried out quite efficiently. Therefore, the scalar multiplications and inner products consume a significant part of the computation time. We have used a way to reduce the number of inner products per iteration, as was suggested recently [16].

Indeed, using the A-orthogonality $(v_i, w_j) = w_i^t A w_j = 0$ for $i \neq j$ we find that

$$(v_i, v_{i-1}) = (v_i, w_i) \quad \text{and} \quad (w, w_{i+1}) = -\frac{(v_i, v_i)}{(v_i, w_i)} w_i - \frac{(v_i, v_{i-1})}{(v_{i-1}, w_{i-1})} w_{i-1}\,.$$

Now at each iteration, given w_i we compute the matrix-vector product Bw_i and the inner product $a_i := (v_i, w_i) = (Bw_i, Bw_i)$, as well as $v_i = Aw_i = B^t(Bw_i)$ and $b_i := (v_i, v_i) = (Aw_i, Aw_i)$. We then have the simplified iteration

$$w_0 = w\,, \quad w_1 = v_0 - \frac{b_0}{a_0} w_0\,, \quad w_{i+1} = v_i - \frac{b_i}{a_i} w_i - \frac{a_i}{a_{i-1}} w_{i-1}$$

and the solution vector $x = \sum_{i=0}^{j-1} \frac{c_i}{a_i} w_i$, where $c_i := (w, w_i)$ can be computed by the iteration

$$c_0 = (w, w)\,, \quad c_1 = a_0 - \frac{b_0}{a_0} c_0\,, \quad c_{i+1} = -\frac{b_i}{a_i} c_i - \frac{a_i}{a_{i-1}} c_{i-1}\,.$$

We see that each iteration requires two matrix-vector products, three scalar multiplications, and two inner products.

3.4 Target element

In order to set ourselves a DLP challenge we construct the "random" target element $\beta \in \mathbb{F}_{q^n}$ using the binary digits expansion of the mathematical constant π. More precisely, considering the q-ary expansion

$$\pi = 3 + \sum_{k=1}^{\infty} c_i \, q^{-i} \quad \text{with} \quad c_i \in S_q := \{0, 1, \ldots, q-1\}$$

we use a bijection between the sets S_q and $\mathbb{F}_q$, which is defined by the mappings $\varphi_{2^k} : \mathbb{F}_{2^k} \to S_{2^k}$, $\sum_{i=0}^{k-1} a_i t_i \mapsto \sum_{i=0}^{k-1} a_i 2^i$ and $\varphi : \mathbb{F}_q \to S_q$, $\sum_{j=0}^{2} b_j w^j \mapsto \sum_{j=0}^{2} \varphi_{2^k}(b_j) 2^{kj}$, and construct this way the target element

$$\beta_\pi := \sum_{i=0}^{n-1} \varphi^{-1}(c_i) \, x^i \in \mathbb{F}_{q^n} \,.$$

4 Discrete logarithms in $\mathbb{F}_{2^{6120}}$

In this section we document the breaking of DLP in the case $k = 8$ and $k' = 3$, i.e., in $\mathbb{F}_{2^{6120}}$. The salient features of the computation are:

- The relation generation for degree 1 elements took 60 seconds.
- The corresponding linear algebra took 60.5 core-hours.
- In contrast to [11, 9], we computed the logarithm of degree 2 irreducibles as they arise; each took on average 0.03 seconds.
- The descent was designed so as to significantly reduce the number of bottleneck (degree 6) eliminations. As a result, the individual logarithm phase took just under 689 core-hours.

4.1 Setup

We first defined $\mathbb{F}_{2^8}$ using the irreducible polynomial $T^8 + T^4 + T^3 + T + 1$. Letting t be a root of this polynomial, we defined $\mathbb{F}_{2^{24}}/\mathbb{F}_{2^8}$ using the irreducible polynomial $W^3 + t$. Letting w be a root of this polynomial, we finally defined $\mathbb{F}_{2^{6120}}/\mathbb{F}_{2^{24}}$ using the irreducible polynomial $X^{255} + w + 1$, where we denote a root of this polynomial by x.

We chose as a generator $g = x + w$, which provably has order $2^{6120} - 1$, since $2^{6120} - 1 =$

$$\begin{aligned}
&3^3 \cdot 5^2 \cdot 7 \cdot 11 \cdot 13 \cdot 17^2 \cdot 19 \cdot 31 \cdot 37 \cdot 41 \cdot 61 \cdot 73 \cdot 103 \cdot 109 \cdot 137 \cdot 151 \cdot 181 \cdot 241 \cdot 307 \cdot 331 \cdot 409 \\
&\cdot 433 \cdot 613 \cdot 631 \cdot 919 \cdot 953 \cdot 1021 \cdot 1321 \cdot 1361 \cdot 1531 \cdot 2143 \cdot 2857 \cdot 3061 \cdot 4421 \cdot 6121 \cdot 6529 \cdot 8161 \\
&\cdot 11119 \cdot 12241 \cdot 13669 \cdot 16831 \cdot 23311 \cdot 26317 \cdot 36721 \cdot 38737 \cdot 43691 \cdot 51001 \cdot 54001 \cdot 61681 \cdot 70381 \\
&\cdot 106591 \cdot 123931 \cdot 131071 \cdot 354689 \cdot 550801 \cdot 949111 \cdot 2582029 \cdot 3696481 \cdot 4260133 \cdot 12717361 \\
&\cdot 15571321 \cdot 18837001 \cdot 23650061 \cdot 29247661 \cdot 40932193 \cdot 318194713 \cdot 965133181 \cdot 1326700741 \\
&\cdot 2949879781 \cdot 4562284561 \cdot 26159806891 \cdot 168692292721 \cdot 611787251461 \cdot 1392971637361 \\
&\cdot 1467129352609 \cdot 2879347902817 \cdot 15455023589221 \cdot 27439122228481 \cdot 253190737566001 \\
&\cdot 418562986357561 \cdot 737539985835313 \cdot 2109936092650831 \cdot 12458723489217613 \\
&\cdot 171664686650370481 \cdot 238495197879143209 \cdot 469775495062434961 \cdot 7226904352843746841 \\
&\cdot 9520972806333758431 \cdot 26831423036065352611 \cdot 51366149455494753931 \\
&\cdot 1230412270786066204321 \cdot 8088220746627020943841 \cdot 75582488424179347083438319 \\
&\cdot 5702451577639775545838643151 \cdot 42515530888344717190444481725601 \\
&\cdot 630894905395143528221826310327361 \cdot 33141833204828142196706150379164851 \\
&\cdot 358689400191468213568189014966376501 \cdot 24710462787135943791475548268920478656481
\end{aligned}$$

$\cdot\, 13854772173181680651901626546855984966582610663321$
$\cdot\, 24887469843822698594826280167758390788291264092494689636443895296 1$
$\cdot\, 112993216763723572293509811422126922278036735923954184346074567839714750801$
$\cdot\, 271144413760016349789555736217593092904995730811145325345053050025782557917644038 4$
$\quad 4101991735185701$
$\cdot\, 394116258262400061951449171583971188086227174396632099228669947330253614155926578 6$
$\quad 93253171536241612096128866834227 21$
$\cdot\, 597590455727045321517345142296769037017630646981106180102454234286272212356398990 4$
$\quad 56648167908702377833056103529473 61 \cdot P_{463},$

where P_{463} is the 463 digit prime (proven with Magma [2] V2.16-12) $\Phi_{6120}(2)$, the 6120-th cyclotomic polynomial evaluated at 2.

As usual, the target element was set to be β_π as explained in Subsection 3.4.

4.2 Relation generation

Our factor base is simply the set of degree 1 elements of $\mathbb{F}_{2^{6120}}/\mathbb{F}_{2^{24}}$. As detailed in Subsection 2.2, quotienting out by the action of the 8-th power of Frobenius produces 21932 distinct orbits. To obtain relations, as explained in Subsection 2.3, we make essential use of the single polynomial $X^{257} + X + 1$, which splits completely over $\mathbb{F}_{2^{24}}$. In particular, letting $y := x^{256}$ so that $x = \frac{y}{w+1}$, the $\mathbb{F}_{2^{6120}}$ element $xy + ay + bx + c$ corresponds to $X^{257} + aX^{256} + bX + c$ on the one hand, and $\frac{X^2}{w+1} + aX + \frac{bX}{w+1} + c$ on the other. The first of these transforms to $X^{257} + X + 1$ if and only if $(a^{256} + b)^{257} = (ab + c)^{256}$. So for randomly chosen (a, b) we compute c and check whether the corresponding quadratic splits. If it does - which occurs with probability 1/2 - we obtain a relation. Thanks to the simplicity of this approach, we collected 22932 relations and wrote these to a matrix in 60 seconds using C++/NTL [17].

4.3 Linear algebra

We took as our modulus the product of the largest 35 factors listed above, which has bitlength 5121. We ran a parallelised C/GMP [6] implementation of Lanczos on 4 of the Intel (Westmere) Xeon E5650 hex-core processors of ICHEC's SGI Altix ICE 8200EX Stokes cluster. This took 60.5 core-hours (just over 2.5 hours wall time).

4.4 Individual logarithm

Using C++/NTL we first used continued fractions to express β_π as a ratio of two 27-smooth polynomials, which took 10 core-hours, and then we applied the three different descent strategies as explained in Subsection 2.4.

We used the large degree descent strategy to express all of the featured polynomials using polynomials of degree 6 or less. This took a further 495 core-hours. While we could have performed this part of the descent more efficiently, as noted above we opted to find expressions which resulted in a relatively small number of degree 6 polynomials - which are the bottleneck eliminations for the subsequent descent - namely 326.

For degrees 6 down to 3 we used the analogue of Joux's small degree elimination method, based on the same polynomial that we used for relation generation, i.e., $X^{257} +$

$X+1$, rather than the polynomial $X^{256}+X$ that was used in [11], since the resulting performance was slightly better.

For degree 2 elimination we try to equate $Q(x)=x^2+q_1x+q_0$ with $x^{257}+ax^{256}+bx+c$, where $(a^{256}+b)^{257}=(ab+c)^{256}$. If this fails we apply the following strategy, making use of the fact that $\mathbb{F}_q$ can also be viewed as a field extension over $\mathbb{F}_{2^6}$. We consider $y=x^{256}$ and $\bar{x}=x^4$, so that $y=\bar{x}^{64}$ and $\bar{x}=(\frac{y}{\gamma})^4$ holds, and apply the large degree descent method to $\bar{Q}(X):=Q(\frac{X}{\gamma})$ (note that $\bar{Q}(y)=Q(x)$). Considering the lattice L we construct a basis of the form $(X+u_0,u_1)$, $(v_0,X+v_1)$, where $u_0,u_1,v_0,v_1\in\mathbb{F}_q$. Then for $s\in\mathbb{F}_q$ we have lattice elements $(X+u_0+sv_0,sX+u_1+sv_1)\in L$. Now for each $B\in\mathbb{F}_q^*$ such that $X^{65}+BX+B$ splits we solve for $s\in\mathbb{F}_q$ satisfying

$$(v_0s^2+(u_0+v_1)s+u_1)^{64}=B\,(s^{64}+v_0s+u_0)^{65}\,,$$

which can be expressed as a quadratic system in the $\mathbb{F}_{2^6}$-components of s, and thus solved by a Gröbner basis computation over $\mathbb{F}_{2^6}$. We then have an equation

$$\bar{x}^{65}+a\bar{x}^{64}+b\bar{x}+c=\tfrac{1}{\gamma^4}(y^5+by^4+a\gamma^4y+c\gamma^4)$$

with $a=s$, $b=\gamma s+q_1$, and $c=\frac{q_0}{\gamma}$, where the left-hand side polynomial splits, while the right-hand side polynomial contains $\bar{Q}(X)$. The polynomial $X^5+bX^4+a\gamma^4X+c\gamma^4=\bar{Q}(X)R(X)$ has the property that $R(X)$ factors always into a linear factor and a quadratic polynomial $Q'(X)$. Now if $Q'(X)$ is resolvable by the direct method, we have successfully eliminated the original polynomial $Q(X)$. The number of B such that $X^{65}+BX+B$ splits over $\mathbb{F}_q$ equals 64, according to Theorem 1, and by experiment, for each one the success probability to find a resolvable polynomial $Q'(X)$ is about $\frac{1}{3}$.

For convenience we coded the eliminations of polynomials of degrees 6 down to 2 in Magma [2] V2.16-12, using Faugere's F4 algorithm. The total time for this part was just over 183.5 core-hours on a 2GHz AMD Opteron computer.

For the logarithm modulo the cofactor of our modulus we used either linear search or Pollard's rho method, which took 20 minutes in total in C++/NTL. Thus the total time for the descent was just under 689 hours.

Finally, we found that $\beta_\pi=g^{\log}$, with $\log=$

13858759836397869262547571128312317100923636150389699236649593170451770028012717802223489
4098617581360131441835074256363730624426814293233474272521598166126957928116825443110965
4042538379388085954041110352380271077721788229392818734034519997318151400734817665137153
5844927931455679735244624686031794675012447568947440627494235603593650167405093344890920
1029834522226732247771897083223217282051573645013603613042367782716361877817938374393824
3130190736247863876184140375416811202840446593831929074368525263920877243047754516312718
2525096811145140050273340438176967525528912734663935009822157084440038078851633249658388
2522436381918008200167032186350245107751346979596314696153666716168951481948091060066730
1847667581377739443038754298308672054639181442568439117307472651461541934380416278336617
3977505716123634609623656687525127784306232997304447548656106220435690856847147127938378
1038538818884463796989906076079843248127252020839705886436071213650575186707456948584072
3789169429253691408684171964795734810327114810217291628659735881740963899133056076778580
3399636173490553715036202472051577266078120885550543433105576657001421187560294063357576
3850457503079087074376585304470520411320246292255375711457573555286060236699317039454479
3267182811289614232751427875694256905328332833440496355213025960008971925120366952988072
9403296453095969137708720454634896013276009554410598019825524549320241283159389198478815
2417957691939817112366182063687529915365150361180214451234387656883256149355994405051149
5859691630753070266479560356836715895464485399551327261120349386559612918562034222476803
8702907847352095116033447252547507168067262366158729272032960618251204431219435715613920
1340952037872975243254476081554937002122953415949407262137232099852298394838422907643191
39767329023834418304604097585991592853653044569714531766804497370964833241561850418.

4.5 Total running time

The total running time is $689+60.5 = 749.5$ core-hours. Note that most of the computation (all except the linear algebra part) was performed on a personal computer. On a modern quad-core PC, the total running time would be around a week.

References

1. Antonia W. Bluher. On $x^{q+1} + ax + b$. *Finite Fields and Their Applications*, 10(3):285–305, 2004.
2. Wieb Bosma, John Cannon, and Catherine Playoust. The Magma algebra system. I. The user language. *J. Symbolic Comput.*, 24(3-4):235–265, 1997.
3. Faruk Göloğlu, 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 $\mathbb{F}_{2^{3164}}$. To appear in Advances in Cryptology—CRYPTO 2013.
4. Faruk Göloğlu, Robert Granger, Gary McGuire, and Jens Zumbrägel. Discrete Logarithms in $GF(2^{6120})$. NMBRTHRY list, https://listserv.nodak.edu/cgi-bin/wa.exe?A2=NMBRTHRY;fe9605d9.1304, April 11th, 2013.
5. Faruk Göloğlu, Robert Granger, Gary McGuire, and Jens Zumbrägel. Discrete Logarithms in $GF(2^{1971})$. NMBRTHRY list, https://listserv.nodak.edu/cgi-bin/wa.exe?A2=NMBRTHRY;f7755cbe.1302, February 19th, 2013.
6. Torbjörn Granlund and the GMP development team. *GNU MP: The GNU Multiple Precision Arithmetic Library*, 5.0.5 edition, 2012. http://gmplib.org/.
7. Tor Helleseth and Alexander Kholosha. $x^{2^l+1} + x + a$ and related affine polynomials over $\mathrm{GF}(2^k)$. *Cryptogr. Commun.*, 2(1):85–109, 2010.
8. Antoine Joux. Faster index calculus for the medium prime case. Application to 1175-bit and 1425-bit finite fields. Cryptology ePrint Archive, Report 2012/720, 2012. http://eprint.iacr.org/.
9. Antoine Joux. A new index calculus algorithm with complexity $L(1/4 + o(1))$ in very small characteristic. Cryptology ePrint Archive, Report 2013/095, 2013. http://eprint.iacr.org/.
10. Antoine Joux. Discrete Logarithms in $GF(2^{1778})$. NMBRTHRY list, https://listserv.nodak.edu/cgi-bin/wa.exe?A2=NMBRTHRY;7d4dd9a6.1302, February 11th, 2013.
11. Antoine Joux. Discrete Logarithms in $GF(2^{4080})$. NMBRTHRY list, https://listserv.nodak.edu/cgi-bin/wa.exe?A2=NMBRTHRY;71e65785.1303, March 22nd, 2013.
12. Antoine Joux and Reynald Lercier. The function field sieve in the medium prime case. In *Advances in cryptology—EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Comput. Sci.*, pages 254–270. Springer, 2006.
13. Brian A. LaMacchia and Andrew M. Odlyzko. Solving large sparse linear systems over finite fields. In *Advances in cryptology—CRYPTO '90*, volume 537 of *Lecture Notes in Comput. Sci.*, pages 109–133. Springer, Berlin, 1991.
14. Cornelius Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *J. Research Nat. Bur. Standards*, 45:255–282, 1950.
15. Hendrik W. Lenstra, Jr. Finding isomorphisms between finite fields. *Math. Comp.*, 56(193):329–347, 1991.
16. Ilya Popovyan. Efficient parallelization of lanczos type algorithms. Cryptology ePrint Archive, Report 2011/416, 2011. http://eprint.iacr.org/.
17. Victor Shoup. *NTL: A library for doing number theory*, 5.5.2 edition, 2009. http://www.shoup.net/ntl/.