The Discrete Logarithm Problem

René Schoof

Abstract

For large prime numbers p, computing discrete logarithms of elements of the multiplicative group $(\mathbf{Z}/p\mathbf{Z})^*$ is at present a very difficult problem. The security of certain cryptosystems is based on the difficulty of this computation. In this expository paper we discuss several generalizations of the discrete logarithm problem and we describe various algorithms to compute discrete logarithms.

1 Introduction

For a prime number p, the multiplicative group $(\mathbf{Z}/p\mathbf{Z})^*$ is cyclic of order $p-1$. Generators of $(\mathbf{Z}/p\mathbf{Z})^*$ are called *primitive roots mod* p. Let p be a prime and let g denote a primitive root modulo p. Then for every $x \in (\mathbf{Z}/p\mathbf{Z})^*$ we have

$$x = g^a,$$

for some integer a. This integer is called the *discrete logarithm of* x and is denoted by $\log x$. Since it depends on the primitive root g, one often writes $\log_g x$ rather than $\log x$. Since the discrete logarithm is only unique modulo $p-1$, we view it as an element of the additive group $\mathbf{Z}/(p-1)\mathbf{Z}$. Just as for the usual logarithm, we have for $x, y \in (\mathbf{Z}/p\mathbf{Z})^*$ that

$$\log xy \;=\; \log x + \log y.$$

Here is an explicit example. Let $p = 10000000259$. Since $(p-1)/2$ is prime, it is easy to see that $g = 2$ is a primitive root modulo p. We have

R. Schoof
Università di Roma "Tor Vergata", Dipartimento di Matematica, Via della Ricerca Scientifica, I-00133 Roma, ITALY
e-mail: schoof@mat.uniroma2.it

$$\begin{aligned}\log 3 &= 9635867242,\\ \log 5 &= \ \ 227891530,\\ \log 7 &= 1803320787,\\ &\vdots\end{aligned}$$

illustrating the fact that there is no simple minded formula for $\log x$ in terms of x. Indeed, the discrete logarithms of the first few primes appear like random numbers in the interval from 0 to $p-1$.

Given a prime number p, a primitive root $g \in (\mathbf{Z}/p\mathbf{Z})^*$ and an exponent a in $\mathbf{Z}/(p-1)\mathbf{Z}$, the element $x = g^a$ can be computed efficiently by repeated squarings and multiplications. On the other hand, given a large prime number p and a primitive root $g \in (\mathbf{Z}/p\mathbf{Z})^*$, there are at present no good methods to compute the discrete logarithm of a given element $x \in (\mathbf{Z}/p\mathbf{Z})^*$. In other words, computing the exponent $a \in \mathbf{Z}/(p-1)\mathbf{Z}$ for which $x = g^a$, is a very difficult problem. In particular, there is no polynomial time algorithm known to perform this calculation. There do, however, exist subexponential algorithms.

Designing good algorithms to compute discrete logarithms is a problem that is of interest in itself. It is also relevant for applications in cryptography. The security of the Diffie-Hellmann key exchange protocol [6] relies on the assumption that computing discrete logarithms is very hard. More precisely, the working hypothesis of this protocol is that given a prime number p, a primitive root g modulo p and elements x and y in $(\mathbf{Z}/p\mathbf{Z})^*$, but not their discrete logarithms a and b, it is very hard to compute the element g^{ab}.

In this expository paper we describe some methods for computing discrete logarithms. We do not pretend to present the best known ones, but merely try to give the main ideas behind the most commonly used algorithms. In section 2 we describe a natural generalization of the discrete logarithm problem and we discuss the baby-step-giant-step method due to Dan Shanks and a probabilistic method due to John Pollard. Section 3 is dedicated to the subexponential index calculus algorithm. In section 4 we discuss the discrete logarithm problem for multiplicative groups of finite fields of relatively small characteristic. In particular, we describe recent work of Antoine Joux and others. Finally in section 5, we discuss the discrete logarithm problem for groups of points of elliptic curves over finite fields. This is relevant for applications in cryptography.

I thank Hendrik Lenstra for several useful comments on an earlier version of this paper.

2 Exponential algorithms

Following ideas in [11, section 7], we consider the following general problem. It is a natural extension of the discrete logarithm problem.

Problem. Given a finite set S, a finite abelian group A and a group homomorphism

$$f : \mathbf{Z}^S \longrightarrow A,$$

determine the kernel of f.

In applications, the group A is the multiplicative group of a finite field or the multiplicative group $(\mathbf{Z}/n\mathbf{Z})^*$ of the finite ring $\mathbf{Z}/n\mathbf{Z}$. It can also be an ideal class group of a number field, or the group of points of an elliptic curve or of an abelian variety over a finite field. In general, we assume that A is given to us in such a way that we can efficiently compose elements, calculate inverses and test for equality. Usually we even require that every element in A has a unique, easily computable "reduced" representative. But in general we do not suppose that we know the structure or even the cardinality of A. In most applications we know an upper bound for $\#A$. Since $\mathrm{Hom}(\mathbf{Z}^S, A)$ is naturally isomorphic to A^S, the map f can be specified by giving $\#S$ elements in A indexed by $s \in S$. The kernel of f is a free group of the same rank as $\mathbf{Z}^S$. It can be described by giving $\#S$ generators.

If S consists of one element, then f is determined by $f(1) = x \in A$. Determining the kernel of f is the same problem as determining the order of the element $x \in A$. An algorithm that solves this problem for the group $A = (\mathbf{Z}/n\mathbf{Z})^*$ can be used to factor the integer n. See [13, Lemma 5]. If $\#S = 2$ and f is therefore given by two elements $x, y \in A$, determining the kernel of f is the same as finding all relations between x and y that are of the form $x^k y^m = 1$ with $(k,m) \in \mathbf{Z}^2$. In particular, if A is a cyclic group of order m generated by x, so that $y = x^a$ for some $a \in \mathbf{Z}$, then the kernel of f is the subgroup generated by $(m,0)$ and $(a,-1) \in \mathbf{Z}^2$. Determining $\ker f$ is therefore the same problem as computing the 'discrete logarithm' a of y.

In general, the difficulty in computing the kernel of a group homomorphism $f : \mathbf{Z}^S \longrightarrow A$ does not depend on the group structure of the finite abelian group A, but rather on the way it is presented. For instance, if A is the additive group $\mathbf{Z}/n\mathbf{Z}$ of integers modulo n, presented in the usual way, the problem is very easy. It can be solved using at most $O(\#S)$ gcd computations in the ring $\mathbf{Z}$, which can be efficiently calculated by means of the Euclidean algorithm. Indeed, when $\#S = 1$ the kernel of $f : \mathbf{Z} \longrightarrow \mathbf{Z}/n\mathbf{Z}$ is generated by n/d, where $d = \gcd(f(1), n)$. For larger S one proceeds inductively, dealing in a similar way with one element of S at the time. On the other hand, if A is the cyclic multplicative group $\mathbf{F}_p^*$ or the group $E(\mathbf{F}_p)$ of an elliptic curve over $\mathbf{F}_p$ for some large p, there are no methods known to compute the kernel of a homomorphism $f : \mathbf{Z}^S \longrightarrow A$ that are of a comparable efficiency.

Let $f : \mathbf{Z}^S \longrightarrow A$ be a homomorphism. Since both A and S are finite sets, the problem of determining the kernel of f is a finite issue. The straightforward naive algorithm to solve it, runs as follows. First assume that $\#S = 1$. In other words, we are given an element in $x \in A$ and we wish to compute its order. This can be done by

computing the powers $1, x, x^2, \ldots$ of x until x^i is equal to the neutral element $1 \in A$. The exponent i is then the order of x and generates $\ker f$. When S is larger and f is determined by elements $x, y, z, \ldots \in A$, we first list the elements of the subgroup H generated by x as above. Next we list all elements in the cosets $y^i H$ for $i = 0, 1, 2, \ldots$ The smallest exponent i for which $y^i H = H$ gives rise to a relation of the form $y^i = x^j$ and hence to an element in the kernel of f. Next one puts $H = \langle x, y \rangle$ and lists all elements of the cosets $z^i H$ ... etc. This method uses $O(\#A \#S)$ operations in A. Since eventually all elements of A may be listed, the amount of memory required is $O(\#A)$.

If we can compute a proper non-trivial subgroup B of A, then the problem of computing the kernel K of $f : \mathbf{Z}^S \longrightarrow A$ can be reduced to *two* similar problems involving the subgroup B and the quotient group $B' = A/B$. More precisely, writing π for the canonical map $A \longrightarrow B'$ and K' for the kernel of the composite map $\pi \cdot f : \mathbf{Z}^S \longrightarrow A \longrightarrow B$, we have the following commutative diagram with exact rows and columns

$$\begin{array}{ccccccccc}
 & & 0 & & 0 & & & & \\
 & & \downarrow & & \downarrow & & & & \\
 & & \ker f & \xrightarrow{\cong} & K & & & & \\
 & & \downarrow & & \downarrow & & & & \\
0 & \longrightarrow & K' & \longrightarrow & \mathbf{Z}^S & \xrightarrow{\pi \cdot f} & B' & & \\
 & & \downarrow{\scriptstyle f} & & \downarrow{\scriptstyle f} & & \| & & \\
0 & \longrightarrow & B & \longrightarrow & A & \xrightarrow{\pi} & B' & \longrightarrow & 0.
\end{array}$$

The group K' is isomorphic to $\mathbf{Z}^{S'}$ for some finite set S' having the same cardinality as S. The map f maps K' to B. Since the kernel of the homomorphism $f : K' \longrightarrow B$, is isomorphic to K, we can compute K by first computing the kernel K' of $\pi \cdot f : \mathbf{Z}^S \longrightarrow B'$ and then the kernel of $f : K' \longrightarrow B$.

The groups B and B' are smaller than A. Since B is contained in A, one can efficiently compose elements, calculate inverses and test for equality in B. Therefore, if one is also able to do this in the group $B' = A/B$, then it is usually a good idea to make this reduction. This observation is due to Pohlig-Hellmann [16]. It applies for instance, if one knows a section $j : B' \longrightarrow A$ of π so that $A \cong B \times B'$. In this case equality tests in B' can be performed in A. Another example is the case when A is cyclic and we know a proper divisor d of the order of $n = \#A$. Then we can take $B = dA$ and compute in $B' = A/dA$ exploiting the isomorphism $B' \cong (n/d)A$ given by multiplication by n/d. It follows that computing the kernel of $f : \mathbf{Z}^S \longrightarrow A$ is relatively easy when all prime divisors of $\#A$ are small. Therefore, in this paper one should keep in mind groups A, for which $\#A$ is divisible by at least one large prime number.

Next we describe a more efficient algorithm to compute the kernel of a homomorphism $f : \mathbf{Z}^S \longrightarrow A$. It is the baby-step-giant-step algorithm due to Dan Shanks [20]. It is deterministic and uses $O(\#S\sqrt{\#A})$ operations and equality tests in the group A. It also requires the storage of $O(\sqrt{\#A})$ elements of A. We explain the algorithm in

the case $\#S = 1$. For larger S, the idea remains the same, but, as in our description above of the naive algorithm, the details are more cumbersome to write down [3]. Any homomorphism $f : \mathbf{Z} \longrightarrow A$ is determined by the element $x = f(1)$ of A. Let a be the integer part of $\sqrt{\#A} + 1$. We first make baby-steps. This means that we make a list of the elements x^i for $0 \leq i < a$. If for some i in this range, x^i is the neutral element of A, we are done: the smallest such i is the order of x and generates the kernel of f. If this is not the case, we make giant steps: we put $y = x^a$ and compute y^j for $1 \leq j \leq a$. Each time we check whether y^j is in the list that we made. In order to be able to do this efficiently, we assume that the elements in A are presented in some unique "reduced" way and that the list of the elements x^i for $0 \leq i \leq a$ is sorted with respect to this presentation. Since the order of x is at most $\#A < a^2$, it is bound to happen that for some j, the element y^j is in the list. If it does, we have $x^i = y^j = x^{aj}$ for some $i = 0, 1, \ldots, a$. The first value of j for which this happens has the property that $aj - i$ is the order of x and hence generates the kernel of f.

There are also probabilistic algorithms to compute the kernel of $f : \mathbf{Z}^S \longrightarrow A$. They have the same running time as the baby-step-giant-step algorithm. The advantage of the probabilistic algorithms is, that they do not require making lists of size $\sqrt{\#A}$. We describe the so-called ρ-algorithm, due to John Pollard [17]. Once again we explain the algorithm only in the case $\#S = 1$. In this case, we put $x = f(1)$ and make a random, or rather pseudorandom, walk by evaluating elements of the form x^{n_i} for $i = 0, 1, 2, \ldots$ with $1 = n_0 < n_1 < n_2 \ldots$. This means that for each i, the next element $x^{n_{i+1}}$ is computed in a pseudorandom fashion from the element $x^{n_i} \in A$. By the birthday paradox, one expects that $x^{n_i} = x^{n_j}$ for two distinct values of i, j that are $O(\sqrt{\#A})$. Moreover, this can be detected efficiently using the cycle detection algorithm, attributed to R.W. Floyd by Donald Knuth [9, p.7]. The order of x divides $n_i - n_j$. In practice the quotient is small, so that the order of x can be computed easily.

3 Index calculus

Index calculus is a method to compute discrete logarithms and, more generally, to determine kernels of homomorphisms $f : \mathbf{Z}^S \longrightarrow A$, that applies when A is the multiplicative group of a finite field. In this section we assume that $A = \mathbf{F}_p^*$ for some large prime p. In the next section we consider the case where A is the multiplicative group of a finite field of small characteristic.

Before considering the map f, we do a precomputation and use the index calculus algorithm to compute the kernel of

$$h : \mathbf{Z}^T \longrightarrow \mathbf{F}_p^*,$$

where T is the set of the primes $l \leq X$ for some bound $X < p$ and h is the homomorphism that, for any prime $l \leq X$, maps the l-th basis vector of $\mathbf{Z}^T$ to $l \pmod p$. The kernel of h consists of the vectors $(x_l)_{l \in T} \in \mathbf{Z}^T$ that satisfy

$$\prod_{l\in T} l^{x_l} = 1 \text{ in } \mathbf{Z}_p^*$$

and hence

$$\sum_{l\in T} x_l \log l \equiv 0 \pmod{p-1},$$

where $\log l$ denotes the discrete logarithm of l with respect to any fixed primitive root in $\mathbf{F}_p^*$. The algorithm to determine $\ker h$ runs as follows. Pick random exponents $e(l) \geq 0$ with $\sum_{l\in T} e(l)$ bounded by some power of $\log p$, that is sufficiently large in the sense that the products $\prod_{l\in T} l^{e(l)}$ exceed p. Then check whether the remainder modulo p of $\prod_{l\in T} l^{e(l)}$ is "X-smooth". In other words, check whether its factorization in the ring $\mathbf{Z}$ is of the form $\prod_{l\in T} l^{f(l)}$. If it is, we obtain the relation

$$\prod_{l\in T} l^{e(l)} = \prod_{l\in T} l^{f(l)}, \qquad \text{in } \mathbf{F}_p^*.$$

It follows that the vector $(e(l)-f(l))_{l\in S}$ is in the kernel of h:

$$\sum_{l\in T} (e(l)-f(l)) \log l = 0, \qquad \text{in } \mathbf{Z}/(p-1)\mathbf{Z}.$$

Repeating this procedure, we occasionally find that $\prod_{l\in T} l^{e(l)}$ is X-smooth, hence obtain a non-trivial relation and thus a non-zero vector in the kernel of h. Once we have obtained a bit more than $\#T$ vectors, it is reasonable to expect that the vectors that we found, generate the kernel.

It remains to choose the value of X. If X is very small with respect to p, there are very few X-smooth numbers in the set $\{1,2,\ldots,p-1\}$. Since the remainders of the products $\prod_{l\in T} l^{e(l)}$ appear to be distributed randomly in this set, it is difficult to obtain relations and the algorithm may be time consuming. On the other hand, if X is very large, it is much easier to find X-smooth numbers and vectors in $\ker h$. However, since we need more than $\#T$ relations, we need to find many more of them and the algorithm may also be time consuming.

The optimal value of X is somewhere in the middle. It depends on the probability that a random natural number less than p is X-smooth. Writing $X = p^{1/u}$ for some $u > 1$, this probability is roughly u^{-u}. See [4]. A back of an envelope computation shows that the optimal value for u is approximately $u = 2\sqrt{\log p/\log\log p}$. With this choice of u, computing the kernel of f involves

$$\exp(2\sqrt{\log p \log\log p})$$

elementary operations with numbers that have $O(\log p)$ digits. Therefore this is a subexponential algorithm.

With this choice of u, the set of primes $l \leq X$ almost certainly generates $\mathbf{F}_p^*$, so that f is surjective. Therefore the induced map

$$\overline{h} : (\mathbf{Z}/(p-1)\mathbf{Z})^T \longrightarrow \mathbf{F}_p^*$$

is also surjective and hence split. This means that the kernel of $\overline{h}$ is the zero set of a single linear equation $\sum_{l\in T} a_l X_l \equiv 0 \pmod{p-1}$, with coefficients a_l equal to $\log l$ with respect to the primitive root g that is given by $g = \prod_{l\in T} l^{y_l}$. Here $(y_l)_{l\in T}$ is any vector for which one has $\sum_{l\in T} a_l y_l = 1$ in $\mathbf{Z}/(p-1)/\mathbf{Z}$. The equation can be computed efficiently using linear algebra over the ring $\mathbf{Z}/(p-1)\mathbf{Z}$. This completes the description of the precomputation.

In order to explain how to determine the kernel of the given homomorphism

$$f : \mathbf{Z}^S \longrightarrow \mathbf{F}_p^*,$$

we consider first the case that $\#S = 1$. In this case f is determined by the element $x = f(1) \in \mathbf{F}_p^*$ and the kernel of f is generated by the order of x in the group $\mathbf{F}_p^*$. To compute the kernel, pick random products $x\prod_{l\in T} l^{e(l)}$ with T as above and check whether the factorization in $\mathbf{Z}$ of the remainder modulo p is of the form $\prod_{l\in T} l^{f(l)}$. If this happens, we obtain the relation

$$x\prod_{l\in T} l^{e(l)} = \prod_{l\in T} l^{f(l)}, \qquad \text{in } \mathbf{F}_p^*.$$

This implies that

$$\log x = \sum_{l\in T} (f(l) - e(l)) \log l, \quad \text{in } \mathbf{Z}/(p-1)\mathbf{Z}.$$

Since we already have computed $\log l$ for every $l \in T$, we can now evaluate $\log x$. The order of x in the group $\mathbf{F}_p^*$ is equal to the order of $\log x$ in the additive group $\mathbf{Z}/(p-1)\mathbf{Z}$. It is therefore equal to $p-1$ divided by $\gcd(p-1, \log x)$.

The method for $\#S > 1$ is based on this. One computes the discrete logarithm of $f(s)$ for each element $s \in S$. Composing $f : \mathbf{Z}^S \longrightarrow \mathbf{F}_p^*$ with the discrete logarithm gives a homomorphism from $\mathbf{Z}^S$ to the additive group $\mathbf{Z}/(p-1)\mathbf{Z}$. As remarked above, determining the kernel of such a homomorphism is easy and can be done by means of linear algebra over $\mathbf{Z}$.

4 Finite fields

Recently there has been great progress in solving the discrete logarithm problem for finite fields of small characteristic. Indeed, in [1,5,7,8] algorithms are described that almost run in polynomial time. As in the previous section, the algorithms proceed by first computing the logarithms of a set of elements –*the factor base*– that are, in some sense, small. Next one uses this to solve the problem of computing the discrete logarithm of an individual element that is not in the factor base. Here we describe the first phase following Antoine Joux and his collaborators [1]. For the second phase we refer to the papers mentioned above for more details.

As a typical example of the method, we discuss the case of a finite field of $Q = q^{2k}$ elements, where q is a prime power and k is of the same order of magnitude as q. See [1] for a precise description of the range of finite fields for which the algorithm is effective. Let $\mathbf{F}_{q^2}$ denote the subfield of q^2 elements of $\mathbf{F}_Q$. Note that if k and q are approximately equal, q^2 is small with respect to $Q = q^{2k}$. Therefore, making a list of *all* elements of $\mathbf{F}_{q^2}$ can be done in time polynomial in $\log Q$. As a consequence, computing discrete logarithms of elements in $\mathbf{F}_{q^2}^*$ can be done in time polynomial in $\log Q$ as well. Therefore, the Pohlig-Hellmann argument of section 2 reduces the problem of computing discrete logarithms in the group $\mathbf{F}_Q^*$ to the problem of computing discrete logarithms in the group $A = \mathbf{F}_Q^*/\mathbf{F}_{q^2}^*$.

We assume that the field with $Q = q^{2k}$ elements is represented as $\mathbf{F}_{q^2}[X]/(\phi(X))$, where $\phi(X)$ is an irreducible degree k polynomial in $\mathbf{F}_{q^2}[X]$. In order to have an efficient algorithm, the polynomial $\phi(X)$ in $\mathbf{F}_Q = \mathbf{F}_{q^2}[X]/(\phi(X))$ is supposed to have a special shape: we want that

$$X^q \equiv r(X) \bmod \phi(X),$$

for some rational function $r(X) \in \mathbf{F}_{q^2}(X)$ whose numerator and denominator have very small degrees. Examples are provided by the polynomials $\phi(X) = X^{q-1} - g$ or $X^{q+1} - g$, where g is a generator of the cyclic group $\mathbf{F}_{q^2}^*$. In the first case we have $r(X) = gX$ and in the second $r(X) = g/X$. See [23]. Numerical experiments suggest [1] that when k is close to q, one can find a rational function $r(X)$ with denominator and numerator of degree at most 2, for which $X^q - r(X)$ is divisible by an irreducible degree k polynomial $\phi(X)$. Since an algorithm of Lenstra [10] allows one to compute an isomorphism between any presentation of the finite field $\mathbf{F}_Q$ and $\mathbf{F}_{q^2}[X]/(\phi(X))$ in polynomial time, requiring $\phi(X)$ to have this special shape, is not a serious restriction.

In the first phase of the algorithm we compute the discrete logarithms of the elements in a factor base, which in this case consists of the images of all monic linear polynomials in $\mathbf{F}_{q^2}[X]$ in the group $A = \mathbf{F}_{q^2}[X]/(\phi(X))^*/\mathbf{F}_{q^2}^*$. Putting $T = \mathbf{F}_{q^2}$, this means that we compute the kernel of the homomorphism

$$h : \mathbf{Z}^T \longrightarrow A,$$

that maps the basisvector e_u corresponding to $u \in T$ to the image of $X - u$ in the group $A = (\mathbf{F}_{q^2}[X]/(\phi(X)))^*/\mathbf{F}_{q^2}^*$.

The identity

$$X^q - X = \prod_{u \in \mathbf{F}_q} (X - u)$$

implies that

$$r(X) - X = \prod_{u \in \mathbf{F}_q} (X - u), \qquad \text{in } \mathbf{F}_{q^2}[X]/(\phi(X)).$$

The denominator of the rational function $r(X) - X$ is equal to the one of $r(X)$. For the sake of exposition, we suppose that it factors into a product of linear polynomials

in $\mathbf{F}_{q^2}[X]$. Its numerator has degree at most 3 and may or may not factor into a product of linear polynomials in $\mathbf{F}_{q^2}[X]$. If it does, we obtain a multiplicative relation in the group A between the elements of our factor base. The relation gives then rise to an element in the kernel of the homomorphism $h : \mathbf{Z}^T \longrightarrow A$.

In order to get more relations, we apply automorphisms of the fraction field $\mathbf{F}_{q^2}(X)$ of $\mathbf{F}_{q^2}[X]$. The group $\mathrm{PGL}_2(\mathbf{F}_{q^2})$ acts on the right on $\mathbf{F}_{q^2}(X)$ as follows:

$$f^\sigma(X) = f(\frac{aX+b}{cX+d}), \qquad \text{for any } \sigma = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \mathrm{PGL}_2(\mathbf{F}_{q^2}).$$

Applying $\sigma \in \mathrm{PGL}_2(\mathbf{F}_{q^2})$ to the identity above, we obtain the equality

$$(X^\sigma)^q - X^\sigma = (X^q - X)^\sigma = \prod_{u \in \mathbf{F}_q} (X^\sigma - u), \qquad \text{in } \mathbf{F}_{q^2}(X).$$

The group $\mathrm{PGL}_2(\mathbf{F}_{q^2})$ acts via linear fractional transformations on the left on the projective line $\mathbf{P}_1$ and preserves the set of $\mathbf{F}_{q^2}$-points $\mathbf{P}_1(\mathbf{F}_{q^2})$. We view $\mathbf{F}_{q^2}$ as a subset of $\mathbf{P}_1(\mathbf{F}_{q^2})$. So we have $\mathbf{P}_1(\mathbf{F}_{q^2}) = \mathbf{F}_{q^2} \cup \{\infty\}$. For a function $f \in \mathbf{F}_{q^2}(X)$, a point $u \in \mathbf{P}_1(\mathbf{F}_{q^2})$ and an automorphism $\sigma \in \mathrm{PGL}_2(\mathbf{F}_{q^2})$, we have $f^\sigma(u) = f(\sigma(u))$.

The above identity for $\sigma = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \mathrm{PGL}_2(\mathbf{F}_{q^2})$ then becomes

$$(cX+d)(aX+b)^q - (aX+b)(cX+d)^q = \prod_{u \in \sigma^{-1}(\mathbf{P}_1(\mathbf{F}_q)) - \{\infty\}} (X-u).$$

It holds in the function field $\mathbf{F}_{q^2}(X)$ up to multiplication by some $\lambda \in \mathbf{F}_{q^2}^*$. Note that the set $\sigma^{-1}(\mathbf{P}_1(\mathbf{F}_q))$ consists of $q+1$ points and may or may not contain the point ∞. Since $X^q \equiv r(X)$ modulo $\phi(X)$, we have

$$(aX+b)^q \equiv \overline{a} r(X) + \overline{b} \quad \text{and} \quad (cX+d)^q \equiv \overline{c} r(X) + \overline{d}$$

in $\mathbf{F}_{q^2}[X]/(\phi(X))$. Here we put $\overline{t} = t^q$ for $t \in \mathbf{F}_{q^2}$. This leads to the following relation

$$(cX+d)(\overline{a} r(X) + \overline{b}) - (aX+b)(\overline{c} r(X) + \overline{d}) = \prod_{u \in \sigma^{-1}(\mathbf{P}_1(\mathbf{F}_q)) - \{\infty\}} (X-u). \qquad (*)$$

It holds in the ring $\mathbf{F}_{q^2}[X]/(\phi(X))$ up to multiplication by some $\lambda \in \mathbf{F}_{q^2}^*$. The denominator of the left hand side of this equation is equal to the one of $r(X)$. The numerator has degree at most 3 and it seems reasonable to expect that it varies randomly when we vary σ. Numerical experiments have confirmed this [1]. Under this assumption a positive proportion factors into a product of linear polynomials in $\mathbf{F}_{q^2}[X]$. Indeed, this proportion is approximately $1/6$. For other choices of $r(X)$, see [15]. A positive proportion of the relations $(*)$ are therefore multiplicative relations between the elements $X-u$ of the factor base in the group

$A = (\mathbf{F}_{q^2}[X]/(\phi(X)))^*/\mathbf{F}_{q^2}^*$. They give rise to elements in the kernel of the homomorphism $h : \mathbf{Z}^T \longrightarrow A$.

The question is how many independent multiplicative relations between the elements $X-u$ of the factor base we obtain in this way. The discrete logarithms of the elements $X-u$ of the factor base are a solution of the system of linear equations that we obtain from the relations $(*)$. There is at present no proof that we obtain sufficiently many relations for the linear system to have a unique solution over $\mathbf{Z}/M\mathbf{Z}$, where $M = \#A$. However, there are some heuristic arguments in this direction that seem to be confirmed by experiments [1].

The subgroup of $\mathrm{PGL}_2(\mathbf{F}_{q^2})$ that preserves the subset $\mathbf{P}_1(\mathbf{F}_q)$ of $\mathbf{P}_1(\mathbf{F}_{q^2})$, is equal to $\mathrm{PGL}_2(\mathbf{F}_q)$. Therefore, the set $\sigma^{-1}(\mathbf{P}_1(\mathbf{F}_q))$ and hence the right hand side of the relation $(*)$, depends only on the left coset $\sigma^{-1}\mathrm{PGL}_2(\mathbf{F}_q)$ rather than on σ^{-1} itself. The number of cosets is equal to $\#\mathrm{PGL}_2(\mathbf{F}_{q^2})/\#\mathrm{PGL}_2(\mathbf{F}_q) = q^3+q$. It follows that there are q^3+q different subsets $\sigma^{-1}(\mathbf{P}_1(\mathbf{F}_q))$ and hence q^3+q possibilities for the right hand sides of the relations $(*)$. For a positive proportion the left hand sides of $(*)$ factor into products of linear polynomials in $\mathbf{F}_{q^2}[X]$. Therefore one expects many more than q^2 relations between the elements $X-u$ of the factor base, at least when q is not very small. As a consequence we obtain many more than q^2 elements in the kernel of the homomorphism $h : \mathbf{Z}^T \longrightarrow A$.

The subsets $\sigma^{-1}(\mathbf{P}_1(\mathbf{F}_q))$ and therefore the right hand sides of the relations, are very different from one another. Indeed, it easy to see that any two distinct subsets $\sigma^{-1}(\mathbf{P}_1(\mathbf{F}_q))$ intersect in at most two points. Moreover, when σ runs over representatives of the cosets of $\mathrm{PGL}_2(\mathbf{F}_q)$ in $\mathrm{PGL}_2(\mathbf{F}_{q^2})$ and P runs over the points of $\mathbf{P}_1(\mathbf{F}_{q^2})$, the q^3+q by q^2+1 matrix $m_{\sigma,P}$ given by

$$m_{\sigma,P} = \begin{cases} 1, \text{ when } P \in \sigma^{-1}(\mathbf{P}_1(\mathbf{F}_q)); \\ 0, \text{ otherwise.} \end{cases}$$

has maximal rank q^2+1. See [1]. In fact, it is not difficult to show that its rows span a subgroup of index $q+1$ in $\mathbf{Z}^{q^2+1}$. The matrix of the homogeneous linear system we want to solve consists of the subset of rows of the matrix $m_{\sigma,P}$ for which the left hand side of $(*)$ factors completely, somewhat perturbed by the few non-zero coefficients that come from the left hand side of $(*)$. Since the matrix $m_{\sigma,P}$ has maximal rank, it is perhaps not unreasonable to expect that our linear system has a unique solution over $\mathbf{Z}/M\mathbf{Z}$, where $M = \#A$.

It was pointed out by D. Wan et al. [5] that there is a problem with linear polynomials $X-u$ that divide $(X^q - r(X))/\phi(X)$. Indeed, the relations that we find, not only hold modulo $\phi(X)$, but also modulo $X-u$. Typically the multiplicity of $X-u$ is the same on both sides of the relations $(*)$. This cancellation implies that the logarithm of $X-u$ does not appear in the linear system. Therefore it cannot be computed this way. For instance, in the case $\phi(X) = X^{q-1} - g$, we have $r(X) = gX$ and $X^q - gX = X\phi(X)$. In this case, the logarithm of X may not at all appear in the linear system. In this special case however, computing the logarithm of X is easy since its

order in the group A is $q-1$, which is very small. See the original papers [1,8] for ways to get around this problem in general.

5 Elliptic curves

Elliptic curve cryptography is based on the difficulty of solving the discrete logarithm problem in the finite group $E(\mathbf{F}_q)$ of points of an elliptic curve E over a finite field $\mathbf{F}_q$. More generally, determining the kernel of a homomorphism

$$f : \mathbf{Z}^S \longrightarrow E(\mathbf{F}_q),$$

is a difficult problem. Apart from some exceptional situations that we describe below, the only methods that are available at present, are the baby-step-giant-step method and the Pollard ρ-method that were discussed in section 2. Since $\#E(\mathbf{F}_q) \approx q$, both methods require $O(\sqrt{q})$ operations in the group $E(\mathbf{F}_q)$. This is much more than the number of operations required by the subexponential index calculus algorithm that was described in section 3. Therefore, in cryptographical systems based on elliptic curves, key sizes can be made smaller, so that encryption and decryption algorithms are faster.

Suppose that E is an elliptic curve over a finite field $\mathbf{F}_q$ given by a Weierstrass equation

$$Y^2 + a_1XY + a_3 = X^3 + a_2X^2 + a_4X + a_6,$$

with coefficients $a_i \in \mathbf{F}_q$. See [21] for the basic properties of elliptic curves. We let $E(\overline{\mathbf{F}}_q)$ denote the group of points on E with coordinates in an algebraic closure $\overline{\mathbf{F}}_q$ of $\mathbf{F}_q$. It is an infinite torsion group. The set $E(\mathbf{F}_q)$ of points on E with coordinates in $\mathbf{F}_q$ is a finite subgroup. For every natural number n, we let $E[n]$ denote the group of points on E that are annihilated by n. In other words, we have

$$E[n] = \{P \in E(\overline{\mathbf{F}}_q) : nP = 0\}.$$

If n is not divisible by the characteristic p of $\mathbf{F}_q$, the group $E[n]$ is isomorphic to $\mathbf{Z}/n\mathbf{Z} \times \mathbf{Z}/n\mathbf{Z}$. The *Weil pairing* is a bilinear, antisymmetric and non-degenerate pairing

$$e_n : E[n] \times E[n] \longrightarrow \mu_n.$$

Here μ_n denotes the subgroup of n-th roots of unity of $\overline{\mathbf{F}}_q^*$. The pairing e_n is Galois equivariant.

Suppose now that the group $E(\mathbf{F}_q)$ is cyclic of order n, coprime to p. Let $Q \in E[n]$ be a point of order n with the property that the subgroup it generates has trivial intersection with $E(\mathbf{F}_q)$. Then the map

$$g : E(\mathbf{F}_q) \longrightarrow \mu_n$$

given by $g(P) = e_n(P,Q)$ is an injective group homomorphism. It can be efficiently computed by means of an algorithm invented by Victor Miller [14]. In this way the kernel of $f : \mathbf{Z}^S \longrightarrow E(\mathbf{F}_q)$ can be calculated by computing the kernel of the composite homomorphism

$$\mathbf{Z}^S \xrightarrow{f} E(\mathbf{F}_q) \xrightarrow{g} \mu_n \hookrightarrow \mathbf{F}_{q^d}^*.$$

Here d is the order of q modulo n. This approach is due to Menezes, Okamoto and Vanstone [12]. It reduces the problem of computing the kernel of a homomorphism $\mathbf{Z}^S \longrightarrow E(\mathbf{F}_q)$ to a similar problem involving the multiplicative group $\mathbf{F}_{q^d}^*$ rather than $E(\mathbf{F}_q)$.

Since the Weil pairing is Galois equivariant, the field of definition of the point Q contains $\mathbf{F}_{q^d}$. Since d is typically very large, computing in the group $E(\mathbf{F}_{q^d})$ is very costly and this approach is usually not very successful. However, in certain special cases it can be very effective. An important example is provided by *supersingular* elliptic curves over prime fields $\mathbf{F}_p$. When $p \equiv 1 \pmod 4$, the group $E(\mathbf{F}_p)$ of a supersingular curve E is cyclic of order $p+1$. In this case μ_{p+1} is contained in an extension of $\mathbf{F}_p$ that has only degree $d = 2$. Therefore this method is very efficient. With small modifications it also works when $p \equiv 3 \pmod 4$ and more generally when the order of q modulo $n = \#E(\mathbf{F}_q)$ is small. In this situation, this use of the Weil pairing is an efficient way to compute discrete logarithms or, more generally, to compute the kernels of homomorphisms $f : \mathbf{Z}^S \longrightarrow E(\mathbf{F}_q)$.

An even faster algorithm was invented by I.A. Semaev [19] for elliptic curves E over prime fields $\mathbf{F}_p$, for which the group $E(\mathbf{F}_p)$ has order p. In this case an isomorphism

$$g : E[p] \longrightarrow \mathbf{Z}/p\mathbf{Z}$$

is constructed as follows. We fix a non-zero point Q in $E(\mathbf{F}_p)$. For $P \in E(\mathbf{F}_p)$ we put

$$g(P) = \frac{f_P'}{f_P}(Q).$$

Here f_P is a function on E whose divisor is equal to $p(Q-\infty)$. We let f_P' denote the function for which we have the following equality of Kähler differentials: $df_P = f_P' dX$. The map g can be efficiently computed by means of Miller's algorithm. In this way we can compute the kernel of $f : \mathbf{Z}^S \longrightarrow E(\mathbf{F}_p)$ by computing the kernel of

$$\mathbf{Z}^S \xrightarrow{f} E(\mathbf{F}_p) \xrightarrow{g} \mathbf{Z}/p\mathbf{Z}.$$

Similar related algorithms have been proposed by Satoh and Araki [18] and by Smart [22]. See Belding's paper [2] for a relation between the algorithms described in this section.

References

1. Barbulescu, R., Gaudry, P., Joux, A. and Thomé, E.: A quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic, In Nguyen, P., Oswald, E. (Eds) Eurocrypto 2014, LNCS **8441**, 1–16, Springer 2014.
2. Belding, J.V.: A Weil pairing on the p-torsion of ordinary elliptic curves over $K[\varepsilon]$, *J. of Number Theory*, **128** (2008), 1874–1888.
3. Buchmann, J., Jacobson, M. and Teske, E.: On some computational problems in finite abelian groups, *Math. Comp.* **66** (1997), 1663–1687.
4. Canfield, E.R., Pomerance C. and Erdős, P.: On a problem of Oppenheim concerning 'Factorisation Numerorum', *J. Number Theory* **17** (1981), 1–28.
5. Cheng, Q., Wan, D. and Zhuang, J: Traps to the BGJT-algorithm for discrete logarithms, *LMS Journal of Computation and Mathematics* **17** (2014), 218–229.
6. Diffie, W. and Hellman, M.: New directions in cryptography. *IEEE Transactions on Information Theory* **22** (1976), 587–594.
7. Göloğlu, F., Granger, R., McGuire, G. and Zumbrägel, J.: On the function field sieve and the impact of higher splitting probabilities. In Canetti, R. and Garay, J. editors, *Advances in Cryptology—CRYPTO 2013*, LNCS **8043**, 109–128. Springer 2013.
8. Granger, R., Kleinjung, T. and Zumbrägel, J.: On the discrete logarithm problem in finite fields of fixed characteristic, Cryptology ePrint Archive: Report 2015/685.
9. Knuth, Donald E.: *The Art of Computer Programming*, vol. II: Seminumerical Algorithms, Addison-Wesley 1969.
10. Lenstra, H.W.: Finding isomorphisms between finite fields, *Math. Comp.* **56** (1991), 329–347.
11. Lenstra, H.W. and Silverberg, A.: Roots of unity in orders, *Foundations of Computational Mathematics*, to appear.
12. Menezes, A., Okamoto, T., Vanstone, S. A.: Reducing elliptic curve logarithms to logarithms in a finite field, *IEEE Transactions on Information Theory* **39** (1993), 1639–1646.
13. Miller, G.: Riemann's Hypothesis and tests for primality *J. of Computer and System science* **13** (1976), 300–317.
14. Miller, V.: The Weil pairing, and its efficient calculation, *J. Cryptology* **17** (2004), 235–261.
15. Panario, D., Gourdon, X. and Flajolet, P.: An analytic approach to smooth polynomials over finite fields. In J. Buhler, editor, *Algorithmic Number Theory*, Proceedings of the ANTS-III conference, **1423**, 226–236. Springer 1998.
16. Pohlig, S. and Hellman, M.: An improved algorithm for computing logarithms over GF(p) and its cryptographic significance. *IEEE Trans. Inform. Theory* IT24 (1978), 106–110.
17. Pollard, J.: Monte Carlo methods for index computation mod p, *Mathematics of Computation*, **32** (1978), 918–924.
18. Satoh T. and Araki, K.: Fermat quotients and the polynomial time discrete log algorithm for anomalous elliptic curves, *Comment. Math. Univ. St. Paul.* (1998), 81–92.
19. Semaev, I.A.: Evaluation of discrete logarithms in a group of p-torsion points of an elliptic curve in characteristic p, *Math. Comp.* **67** (1998), 353–356.
20. Shanks, D.: Class number, a theory of factorization and genera. In *Proc. Symp. Pure Math.* **20** (1971), 415–440. AMS, Providence, R.I.
21. Silverman, J.H.: *The arithmetic of elliptic curves*, Graduate Texts in Mathematics **106**, 2^{nd} Ed. Springer–Verlag, 2009.
22. Smart, N.: The discrete logarithm problem on elliptic curves of trace one, *J. Cryptology* **12** (1999), 193–196.
23. Xiao, D., Zhuang, J. and Cheng, Q.: Factor base discrete logarithms in Kummer Extensions, Cryptology ePrint Archive: Report 2015/859.