An introduction to elliptic curve cryptography: an explicit approach

Carlos Castano-Bernard
Facultad de Ciencias
Universidad de Colima
Bernal Diaz del Castillo 340
28045 Colima
Mexico
ccastanobernard@gmail.com

Abstract

This note is a concise introduction to some basic aspects of elliptic curve cryptography. It is based on lecture notes of a summer course at undergraduate level. Our approach is computational and uses the computer package PARI/GP *as a tool to describe explicitly the underlying arithmetical objects used in the cryptographic algorithms.*

1 Introduction

It has become clear the need of adequate technologies that may mirror the traditional paper-based transactions in the realm of digital communications in public networks. It is well-known that trillions of dollars in funds and securities are transferred daily electronically. Standards for public key infrastructures such as e.g. the American National Standard X9.62-1998 for the elliptic curve digital signature algorithm (and other standards) have been created for that purpose. More precisely, the public key infrastructures allow us to create digital signatures and establish secret keys secretly, to be used in symmetric-key cryptosystems and guarantee

- *data integrity*,
- *authentication* of sender, and
- *non-repudiation* of the message by the sender.

The underlying mathematics of the first standards of public key infrastructures is modular arithmetic. The key ideas were developed in the 1970's by Whitfield Diffie, Martin Hellman and Taher ElGamal, among others. But by the mid 1980's a new approach to public-key cryptography emerged. This was discovered independently by Neal Koblitz and Victor S. Miller and is based on the arithmetic of elliptic curves over finite fields.

The purpose of this note is to provide a concise introduction to some basic elements of public key infrastructures based on the arithmetic of elliptic curves. These public key infrastructures provide commercial grade security with a much smaller key size than the ones based on modular arithmetic. It is hoped that this note will encourage the reader with an eye on actual implementations to learn more about this topic by looking at the above mentioned standard, as well as the standard IEEE P1363.

In this note we first introduce some basic definitions and results from the theory of finite fields. Then we describe the Diffie-Hellman key exchange protocol based on the multiplicative group of a general finite field, and also discuss the digital signature algorithm based on the integers modulo a prime p. We introduce elliptic curves by writing down the general form of a Weiestraß equation and the explicit expressions of its group law. Then we focus on some basic facts of the theory of elliptic curves over a finite field. This is followed by a discussion of the Diffie-Hellman key exchange protocol and the digital signature algorithm and the elliptic curve versions of them. The main part of this note concludes with some remarks on security and also domain parameter generation. Also included at the end of this note is an appendix containing some basic results from the arithmetic theory of elliptic curves, as well as a brief description on how to get PARI/GP [4].

Our approach in this note is computational and uses the free open-source computer algebra system PARI/GP as a tool to describe explicitly the underlying arithmetical objects used in the cryptographic algorithms. PARI/GP consists of two components. The first component is a highly optimized C library whose primary focus is number theory known as PARI. The second component is an easy-to-use interactive command line interface that gives access to PARI known as GP. The latter may be used as a tool to quickly acquire a good working knowledge of the basics of cryptographic protocols, including those based on elliptic curves. It is hoped that the discussions of the GP sessions will encourage the reader to write his / her own GP routines.

It is worth noting that the reader may use the GP2C package for translating the GP routines into the C programming language; the GP2C-compiled routines will typically run three to four times faster. Moreover, the use of PARI as a C library can be very powerful for dealing with large objects—large enough to be of cryptographic use. Indeed, Bill Allombert (CNRS / Université Bordeaux 1) has been developing parallel PARI. The experimental GIT branch BILL-PAREVAL adds support for the multi-threading technologies POSIX Threads (Pthreads), which runs on a single machine and the Message Passing Interface (MPI), which runs on as many machines as you want. For further details the reader may consult the slides of Allombert's talk

`http://pari.math.u-bordeaux1.fr/Events/PARI2012/talks/pareval.pdf`

The above slides include a discussion of parallel SEA. The SEA algorithm is one of the most efficient algorithms for counting points of elliptic curves defined over the field of p elements, where p is a prime number. So the above may be used for efficiently producing commercial-grade elliptic curve domain parameters. The enhancements may also be used to make massively parallel computation of elliptic curve discrete logarithms to test the strength of the domain parameters.

For an open-source implementation of public key infrastructures the reader is encouraged to consider OpenSSL:

`http://www.openssl.org/`

This system implements cryptographic protocols for SSL (Secure Socket Layer) and TLS (Transport Layer Security) and it is one of the few open-source projects involved with validation under the

Federal Information Processing Standard (FIPS) 140-2 of the U.S. government. The reader having in mind real-life applications is strongly encouraged to install it and use it. We must warn the reader that, mostly due to patent issues, the elliptic curve handshake capabilities of OpenSSL are disabled by default. However, the reader is encouraged to learn about these patent issues and then decide if he / she wishes to enable some of them before compiling the package.

2 Preliminaries

2.1 Finite fields from an explicit point of view

Let E be a finite field. It is well-known that $|E| = p^d$, where d is a positive integer. In order to make explicit calculations in a given finite field E it suffices to determine an irreducible, monic polynomial $p(X) \in F[X]$ of degree d. More precisely, consider the canonical ring homomorphism

$$\begin{aligned} F[X] &\longrightarrow F[X]/I \\ f(X) &\longmapsto \overline{f(X)} \end{aligned} \tag{1}$$

where $\overline{f(X)} := f(X) + I$ and $I = p(X)F[X]$ is the ideal generated by $p(X)$ (as a $F[X]$-module). The irreducibility of $p(X)$ implies that the ideal I is maximal. Indeed, recall that all ideals of $F[X]$ are generated by one element. So if J is an ideal of $F[X]$ such that $I \subsetneq J \subsetneq F[X]$, then $J = p_1(X)F[X]$ and $p(X) = p_1(X)p_2(X)$ non-trivially, where $p_2(X) \in F[X]$. Hence I is maximal and the quotient ring $F[X]/I$ is a field, which we denote E. We will show that $|E| = p^d$, where $d = \partial p(X)$ is the degree of $p(X)$. Let α be the image of X in $F[X]/I$ under the canonical map. We claim that the set

$$\mathcal{B} = \{1, \alpha, \alpha^2, \ldots, \alpha^{d-1}\} \tag{2}$$

is a basis of E, where we regard E as an F-vector space. To see this first we pick an element $\overline{f(X)}$ of the quotient ring $F[X]/I$ and use the fact that there are polynomials $q(X), r(X) \in F[X]$ such that

$$f(X) = p(X)q(X) + r(X), \tag{3}$$

where either $r(X) = 0$ or $\partial r(X) < \partial p(X)$. Hence

$$\overline{f(X)} = \overline{p(X)q(X) + r(X)} = \overline{p(X)q(X)} + \overline{r(X)} = \overline{r(X)}. \tag{4}$$

Thus $\overline{r(X)}$ is a linear combination of elements of $\mathcal{B}$ with coefficients in F. It remains to prove that $\mathcal{B}$ is linearly independent over F. To see this note that a non-trivial linear relation among elements of $\mathcal{B}$ over F yields a polynomial of $p_1(X) \in F[X]$ of degree $1 < d_1 < d$, and thus an ideal $J = p_1(X)F[X]$ such that $I \subsetneq J \subsetneq F[X]$, which certainly contradicts the maximality of I. Therefore $\mathcal{B}$ is linearly independent and our claim follows. So $\mathcal{B}$ is a basis for E over F and thus E has p^d elements.

The above furnishes an explicit approach to the arithmetic of E. More precisely, we may identify each point $x \in E$ with the unique polynomial $f(X) \in F[X]$ of degree $\partial f(X) < d$ such that $x = \overline{f(X)}$, and $f(X)$ may in turn be identified in a natural way with a (unique) point of F^d. Using this we may transfer to F^d the sum and product operations of E, as follows. First, for each $x_i \in E$

we write $x_i = \overline{f(X)}$, where $f_i(X) \in F[X]$ such that $\partial f_i(X) < d$ and $i = 1, 2$. Then we attach to $x_1 + x_2 \in E$ the point $f_1(X) + f_2(X) \in F^d$, and attach to $x_1 x_2 \in E$ the point $r(f_1(X)f_2(X)) \in F^d$, where $r(f(X))$ denotes r in Equation 3.

We shall find useful to explicitly compute $g(X) \in F[X]$ such that $y = \overline{g(X)} \in F[X]/I$ is the inverse of a given non-zero element $x = \overline{f(X)} \in F[X]/I$. This may be accomplished via Euclid's algorithm. The assumption $\overline{f(X)} \neq \overline{0}$ is equivalent to $(f(X), p(X)) = 1$. So there exist (effectively computable) $g(X), h(X) \in F[X]$ such that

$$f(X)g(X) + p(X)h(X) = 1. \tag{5}$$

Taking the class modulo I of left-hand side of this equation yields

$$\overline{f(X)g(X) + p(X)h(X)} = \overline{f(X)g(X)} = \overline{f(X)}\,\overline{g(X)}.$$

Now taking the class modulo I of the right-hand side of Equation 5 together with the above implies that $y = \overline{g(X)}$ is such that $xy = 1$. It is not difficult to see that such $g(X)$ may be chosen so that $\partial g(X) < \partial p(X)$. Thus taking the inverse of $x \in E^\times$ may be expressed in terms of the points of F^d via the identification of E with F^d we have used before.

A well-known result from algebra says that any finite subgroup U of the multiplicative group $K^\times$ of an arbitrary field K is cyclic. In particular, if E is a finite field then there is a generator g of $E^\times$. Given $x \in E^\times$, the *discrete logarithm a of x with respect to g* is the element $a \in \{0, 1, 2, p-2\}$ such that $x = g^a$. This generalizes in a natural way to any finite cyclic group C of order n with a given generator g. (Of course, if we want to have a uniquely defined by $x = g^a$, then we must insist that $a \in \{0, 1, 2, n-1\}$.) There are several algorithms to compute discrete logarithms, such as the baby-step giant step algorithm, the index calculus algorithm, and others. The efficiency of these algorithms will depend on the nature of the cyclic group. It turns out that for cyclic groups C of prime order coming from suitable elliptic curves over a large enough field E, the problem of computing discrete logarithms is in general considerably much harder than the corresponding problem for $K^\times$, where K is a random field such that $|K|$ has the same order-of-magnitude of $|E|$. As we shall see, this kind of considerations will guide us in our security assessments of some protocols that are used in public key infrastructures.

We shall illustrate some of the above with an explicit example coming from a finite field E. Let $p = 2$, i.e. F is the field that consists of two elements, and suppose we are interested in determining an extension E of F of degree $d = 9$, i.e. $|E| = 2^9$. Now we need to find an irreducible polynomial $p(X)$ over F. In order to compute efficiently the arithmetic of E we want $p(X)$ to have as little weight as possible. (Here the *weight* of a polynomial is the number of its nonzero coefficients.) There are tables of irreducible polynomials over finite fields. See for instance Hansen and Mullen [2, p. 6]. But in order to make our description more self-contained, we shall describe a simple method to compute such polynomials. Let us consider the cyclotomic polynomial $\phi_k(X) \in \mathbf{Z}[X]$ of order k. That is, $\phi_k(X)$ is the minimum polynomial over $\mathbf{Q}$ of a generator ζ of the group μ_k of k-th roots of unity in a fixed algebraic closure $\mathbf{Q}^{al}$ of $\mathbf{Q}$. For $k = l$ prime it is just

$$\phi_l(X) = 1 + X + X^2 + \cdots + X^{l-1},$$

Now let $\overline{\phi_k(X)} \in F[X]$ denote the polynomial obtained by applying the canonical map $\mathbf{Z} \longrightarrow F$ to each of the coefficients of $\phi_k(X)$. The new polynomial $\overline{\phi_k(X)}$ is known as the *reduction of* $\phi_k(X)$

modulo 2. Now we shall single out some non-trivial factorizations of $\overline{\phi_l(X)}$ (where l is assumed prime) with the help of the following PARI/GP script.

```
forprime(l=3,80,
f=polcyclo(l)*Mod(1,2);
L=lift(factor(f));
if(length(L~)>3,print(l," ",L))
);
```

The output of the above script (in a slightly edited form so that the lines are not too long) is:

```
31 [x^5 + x^2 + 1, 1;
    x^5 + x^3 + 1, 1;
    x^5 + x^3 + x^2 + x + 1, 1;
    x^5 + x^4 + x^2 + x + 1, 1;
    x^5 + x^4 + x^3 + x + 1, 1;
    x^5 + x^4 + x^3 + x^2 + 1, 1]
73 [x^9 + x + 1, 1;
    x^9 + x^4 + x^2 + x + 1, 1;
    x^9 + x^6 + x^3 + x + 1, 1;
    x^9 + x^6 + x^5 + x^2 + 1, 1;
    x^9 + x^7 + x^4 + x^3 + 1, 1;
    x^9 + x^8 + 1, 1;
    x^9 + x^8 + x^6 + x^3 + 1, 1;
    x^9 + x^8 + x^7 + x^5 + 1, 1]
```

The above script may be interpreted as follows. In the PARI/GP script language is is known that

- `forprime(l=a,b,P(l))` means *compute* $P(l)$ *for each prime* l *contained in the set* $\{a, a+1, \dots, b\}$;
- `polcyclo(l)` means *compute the cyclotomic polynomial* $\phi_l(X)$,
- `Mod(a,p)` represents *the class of* a *modulo* p, where a, p may be either both integers (or both polynomials in one variable);
- multiplying by `Mod(1,p)` represents *taking its reduction modulo* p;
- `lift(P)` means *taking the obvious section of the reduction map*, where P denotes a class modulo m (and here m is either an integer or a polynomial);
- `factor(f)` means *compute the primary factorization* of f and shows it as a matrix whose $(i, 1)$-entry is the i-th irreducible divisor of f and $(i, 2)$-entry is the corresponding multiplicity.

So the above script outputs the factorization of the reduction of $\phi_l(X)$ modulo 2 whenever it has more than 3 irreducible factors, for each l ranging in the set of primes contained in $\{3, 4, 5, \dots, 80\}$. Among these irreducible polynomials let us pick one of degree 9 and lowest weight, e.g. $p(X) = X^9 + X + 1 \in F[X]$. This is represented in PARI/GP by the expression

```
p = Mod(1,2)*(x^9 + x + 1)
```

Now we may explicitly do arithmetic in the field E of $|E| = 2^9 = 512$ elements. Let us obtain a generator g of the multiplicative group $E^\times$. So we consider a random element g of $E^\times$, say $g := \overline{X^2 + X + 1}$. (Recall that we defined the field as $F[X]/I$, where $I = p(X)F[X]$.) Let us find out if g generates the cyclic group $E^\times$. The element g in PARI/GP may be represented as

```
g = Mod(x^2 + x + 1, p);
```

Clearly it suffices to check that $g^k \neq 1$ for all non-trivial divisors k of $|E^\times| = p^d - 1$. Note that $2^9 - 1 = 511$, which has prime decomposition $511 = 7 \cdot 73$, so the non-trivial divisors k of $|E^\times|$ are just $k = 7$ and $k = 73$. For $k = 7$ we compute g^k, which turns out to be

```
Mod(Mod(1, 2)*x^8 + Mod(1, 2)*x^7 + Mod(1, 2),
    Mod(1, 2)*x^9 + Mod(1, 2)*x + Mod(1, 2))
```

and similarly for $k = 73$ we compute g^k, which turns out to be

```
Mod(Mod(1, 2)*x^8 + Mod(1, 2)*x^7 + Mod(1, 2)*x^6 + Mod(1, 2)*x^4 +
    Mod(1, 2)*x + Mod(1, 2),
    Mod(1, 2)*x^9 + Mod(1, 2)*x + Mod(1, 2))
```

As none of these expressions represent the identity element of the group $E^\times$, it follows that the element $g = \overline{1 + X + X^2}$ is indeed a generator of $E^\times$.

Remark 1 *Finite fields E over the field F of two elements are known in computer science as binary fields, as their elements may be represented as binary expressions (i.e. sequences using 0 and 1 only) of length d. The arithmetic of E may be implemented in hardware quite efficiently. These fields are of fundamental importance in many areas of pure and applied mathematics. If the size of the field is small, we produce a table of discrete logarithms a and the powers $x = g^a$. The table may be used to compute the product x_1x_2 of given elements $x_1, x_2 \in E$ simply by looking up x_i in the table to obtain its discrete logarithm a_i, for $i = 1, 2$, and then finding the entry corresponding to the discrete logarithm $a_1 + a_2$.*

2.2 Basic aspects of elliptic curves

Here we review some basic facts on the arithmetic of elliptic curves. For further details see the Appendix, below, and Washington's book [7]. Let K denote either a finite field or the field of rational numbers and let A be an elliptic curve defined over K. A well-known result from the theory of algebraic curves says that A has a Weierstrass model

$$Y^2Z + a_1XYZ + a_3YZ^2 = X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3,$$

for suitable a_1, a_2,a_3,a_4, and a_6 in K. If the characteristic of K is neither 2 nor 3 then after a change of coordinates if necessary, it is possible to find for A a Weierstrass model of the form

$$Y^2 = X^3 - 27c_4X - 54c_6,$$

for suitable c_4, and c_6 in K. (See the Appendix for the precise relation between the coefficients of the former equation and the coefficients of the latter.)

Remark 2 *Elliptic curves over fields of characteristic* 2 *are of fundamental importance in certain cryptographic technologies that use the fact that the bytes are binary strings. Such elliptic curves generally do not admit a Weierstrass model of the form* $Y^2 = X^3 - 27c_4X - 54c_6$.

Another well-known result from the theory of algebraic curves says that for each field extension K' of K the set $A(K')$ has a natural group structure. This may be given in terms of rational functions of the coordinates with coefficients in K. We shall recall these expressions below. As is customary, we express the group law using additive notation. The identity element is $(0:1:0)$ and we denote it O. Let $P \in W(K')$ such that $P \neq O$. Then we may write $P = (x_P, y_P)$ and

$$-P = (x_P, -y_P - a_1x_P - a_3) \in W(K').$$

If $Q = (x_Q, y_Q) \in W(K')$ is such that $P + Q \neq O$ then $P + Q = R$, where $R = (x_R, y_R) \in W(K')$ with coordinates

$$\begin{aligned} x_R &= \lambda^2 + a_1\lambda - a_2 - x_P - x_Q, \\ y_R &= -(\lambda + a_1)x_R - \nu - a_3, \end{aligned}$$

where

$$\lambda = \begin{cases} \frac{y_Q - y_P}{x_Q - x_P}, & \text{if } P \neq Q, \\ \frac{3x_P^2 + 2a_2x_P + a_4 - a_1y_P}{2y_P + a_1x_P + a_3}, & \text{if } P = Q, \end{cases}$$

and

$$\nu = \begin{cases} \frac{y_Px_Q - y_Qx_P}{x_Q - x_P}, & \text{if } P \neq Q, \\ \frac{-x_P^3 + a_4x_P + 2a_6 - a_3y_P}{2y_P + a_1x_P + a_3}, & \text{if } P = Q. \end{cases}$$

This corresponds to the famous chord-and-tangent construction, which is depicted in Figure 1. (See Silverman [5, pp. 58–59] for further details.) Note that these expressions turn the curve A into an algebraic group over K, as the expressions for group law are all defined over K. We denote $[n]P$ the n-fold addition of $P \in A(K)$, for each $n \in \mathbb{N}$ and extend the definition of $[n]P$ to all $n \in \mathbb{Z}$ in the obvious way.

Now let us consider the following example of an elliptic curve defined over $\mathbb{Q}$. Suppose A is the elliptic curve labeled **37a1** in Cremona's Tables [1]. This is the elliptic curve in the first isogeny class among the isomorphism classes of elliptic curves defined over $\mathbb{Q}$ and conductor[1] 37. We may bring A in and perform explicit computations on it with the help of PARI/GP [4]. First we define A as **37a1** via the *initiate elliptic curve object* command

```
A = ellinit("37a1");
```

The Weierstrass equation is $[0, 0, 1, -1, 0]$, i.e. in affine form

$$Y^2 + Y = X^3 - X.$$

By inspection we may see that $P = (0, 0)$ lies on $A(\mathbb{Q})$ and some elementary properties of torsion points show us that P is non-torsion. In fact, it turns out that this point actually generates the

[1]The interested reader may find a definition of conductor of an elliptic curve in Silverman's book [6]. Here it suffices to say that the conductor measures the "arithmetic complexity" of an elliptic curve defined over $\mathbb{Q}$ and has precisely the same prime divisors as the minimal discriminant Δ_A of A

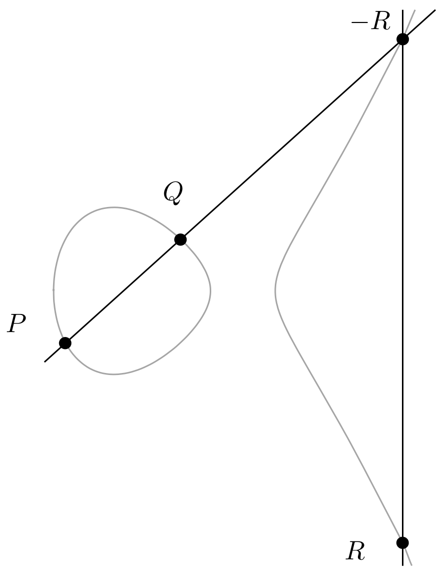

Figure 1: Chord-and-tangent construction

whole Mordell-Weil group $A(\mathbb{Q})$. Define the *naïve height* $h(P)$ of a point $P \in A(\mathbb{Q})$ as $h(P) =$ *max*$\{|x|, |y|, |z|\}$, where $P = (x : y : z) \in \mathbb{P}^2$ and x, y, z are integers in lowest possible terms $(x, y, z) = 1$, We may obtain a table for the naïve height $[k]P$ with k ranging from, say $k \in \{1, \ldots, 39, 40\}$ by invoking the command

```
naiveheight(x)=max(abs(numerator(x)), denominator(x));
for(k=1,40,print(k," ",height(ellpow(A,[0,0],k)[1])));
```

The corresponding output is

```
1 1
2 1
3 1
4 2
5 4
6 6
7 9
8 25
9 49
10 161
11 529
12 1357
13 3741
14 18526
15 98596
16 480106
```

```
17 2337841
18 13608721
19 67387681
20 683916417
21 6941055969
22 51678803961
23 384768368209
24 5677664356225
25 61935294530404
26 997454379905326
27 16063784753682169
28 213822353304561757
29 3148929681285740316
30 79799551268268089761
31 2237394491744632911601
32 53139223644814624290821
33 1262082793174195430038441
34 41998153797159031581158401
35 1063198259901027900600665796
36 54202648602164057575419038802
37 2763291877248901877407461697249
38 102993803538933982914320107718801
39 3838799532815709794201672388387649
40 292736325329248127651484680640160000
```

The reader may perhaps note above that the last figures lie approximately on a parabola. In fact, if the reader extends the range of k in the above computation, the parabolic shape is more evident. This is not an accident. Actually, this is a special case of a general property of non-torsion points $P \in A(K)$, where $K = \mathbb{Q}$ (or more generally any finite extension of $\mathbb{Q}$). It is well-known that the limit

$$\hat{h}(P) := \lim_{k\to\infty} \frac{\log h([k]P)}{k^2}$$

exists and does not depend on the choice of Weierstrass equation for A, and it is known as the *canonical height of* $P \in A(\mathbb{Q})$. (It is easy to see that graph of the map $k \mapsto \hat{h}([k]P)$ actually is a parabola as k ranges through $\mathbb{Z}$.) This is the basis of some heuristic considerations on the difficulty to solve via index calculus the discrete logarithm problem for suitable cyclic subgroups of $\bar{A}(F)$, where $\bar{A}$ is the reduction of A modulo a prime p and F denotes a field of p elements.

Let us study the structure of $\bar{A}(F)$ for the elliptic curve A we have discussed above, i.e. **37a1** in the notation of Cremona's Tables. Let us pick a prime different from $\Delta_A = 37$, say $p = 13$, so that the reduction $\bar{A}$ of A modulo p is non-singular. Thus $\bar{A}$ is an elliptic curve. (See Appendix for some details.) The reduced curve $\bar{A}$ may be defined in GP via the commands

```
p = 13;
a = ellinit([0, 0, 1, -1, 0]*Mod(1,p));
```

Now we can list the elements of its points P defined over F (different from Q_A) by invoking just counting the points of A with affine coordinates, as in the following script.

```
for(j=1,p,for(k=1,p,
P=[j,k]*Mod(1,p);
if(ellisoncurve(a,P),print(centerlift(P)))
));
```

Here `ellisoncurve(a,P)` is 1 when $P \in A(F)$ and 0, otherwise. The output of the script is

```
[1, -1]
[1, 0]
[2, 2]
[2, -3]
[5, 6]
[6, 1]
[6, -2]
[-3, 1]
[-3, -2]
[-2, 4]
[-2, -5]
[-1, -1]
[-1, 0]
[0, -1]
[0, 0]
```

In order to have the complete list of points of $\bar{A}(F)$ we have to add to the above list the point $(0 : 1 : 0)$, which corresponds to the identity element O_A of the Mordell-Weil group of A. So the finite group $A(F)$ consists of 16 points. We may obtain them in a different way by making a list of the multiples of the reduction $\bar{P}$ of the point $P = (0, 0)$ of the discussion on the structure of $A(\mathbb{Q})$. So we invoke

```
for(k=0,17,print(k," ",ellpow(a,[0,0],k)));
```

The output of the above script is

```
0 [0]
1 [0, 0]
2 [Mod(1, 13), Mod(0, 13)]
3 [Mod(12, 13), Mod(12, 13)]
4 [Mod(2, 13), Mod(10, 13)]
5 [Mod(10, 13), Mod(1, 13)]
6 [Mod(6, 13), Mod(1, 13)]
7 [Mod(11, 13), Mod(8, 13)]
8 [Mod(5, 13), Mod(6, 13)]
9 [Mod(11, 13), Mod(4, 13)]
10 [Mod(6, 13), Mod(11, 13)]
11 [Mod(10, 13), Mod(11, 13)]
```

```
12 [Mod(2, 13), Mod(2, 13)]
13 [Mod(12, 13), Mod(0, 13)]
14 [Mod(1, 13), Mod(12, 13)]
15 [Mod(0, 13), Mod(12, 13)]
16 [0]
17 [0, 0]
```

We may see that from $k = 16$ on the list repeats itself. So we have obtained the same 16 points (expressed using a different notation). This means that $\bar{P}$ is a generator of $\bar{A}(F)$.

3 Cryptographic algorithms

3.1 The Diffie-Hellman key exchange

The Diffie-Hellman key exchange is an algorithm that involves two parties, say Alice and Bob, that produces a shared secret key, given certain common public data known as the domain parameters. The shared secret key can then be used to encrypt subsequent communications using a symmetric key cipher. The domain parameters in essence consist of

- a group G,
- an element $g \in G$ that generates a non-trivial subgroup C, and
- the order $n = |C|$ of the cyclic group C.

This is regarded as public information. In this setting the Diffie-Hellman key exchange may be described the protocol in which

1. Alice picks secretly $\alpha \in \{1, 2, \ldots, n-1\}$ and computes her public key $a = g^\alpha$,
2. Bob picks secretly $\beta \in \{1, 2, \ldots, n-1\}$ and computes his public key $b = g^\beta$,
3. Alice copies Bob's public key b and computes b^α,
4. Bob copies Alice's public key a and computes a^β.

The fact that G is a group implies that $b^\alpha = a^\beta$. So Alice and Bob have arrived at common secret information $Q := b^\alpha = a^\beta$ over a public channel. Let us illustrate this algorithm with a simple example. Consider the multiplicative group $E^\times$ of the field E of 2^9 elements and let g be the class of the polynomial $f(X) = 1 + X + X^2$ modulo the irreducible $p(X) = 1 + X + X^9 \in F[X]$. Here F is a field of 2 elements, as in discussed by the end of Subsection 2.1 (above). For the computations we shall use GP.

Step 1. Alice picks, say, $\alpha = 324$ and keeps this information in a secret place. Then she invokes the commands

```
alpha = 324
a = g^alpha
```

and gets

```
Mod(Mod(1, 2)*x^6 + Mod(1, 2)*x^4 + Mod(1, 2)*x^2 + Mod(1, 2)*x +
    Mod(1, 2), Mod(1, 2)*x^9 + Mod(1, 2)*x + Mod(1, 2))
```

She publishes the output.

Step 2. Similarly, Bob picks, say, $\beta = 215$ and keeps this information in a secret place. Then he invokes the commands

```
beta = 215
b = g^beta
```

and gets

```
Mod(Mod(1, 2)*x^8 + Mod(1, 2)*x^6 + Mod(1, 2)*x^2,
    Mod(1, 2)*x^9 + Mod(1, 2)*x + Mod(1, 2))
```

He publishes the output.

Step 3. Alice copies Bob's public key and invokes

```
Q = b^alpha
```

This yields

```
Mod(Mod(1, 2)*x^6 + Mod(1, 2)*x^3 + Mod(1, 2)*x,
    Mod(1, 2)*x^9 + Mod(1, 2)*x + Mod(1, 2))
```

Step 4. Similarly, Bob copies Alice's public key and invokes

```
Q = a^beta
```

This yields

```
Mod(Mod(1, 2)*x^6 + Mod(1, 2)*x^3 + Mod(1, 2)*x,
    Mod(1, 2)*x^9 + Mod(1, 2)*x + Mod(1, 2))
```

We may see that the shared secret keys are indeed the same.

Remark 3 *The main security concern here is the possible use of the index calculus algorithm to efficiently solve the associated discrete logarithm problem. (See Subsection 4.1 for a brief discussion about some heuristic considerations that seem to suggest that the elliptic curve domain parameters are considerably less vulnerable than the above domain parameters, if the corresponding n's are roughly of the same size.)*

Alice and Bob also need to agree on which stream cipher to use. Let us suppose that they decided to use the stream cipher based on RC4 provided by OpenSSL. Then they also have to agree on a rule on how to associate to Q a sequence of ASCII characters, e.g. first invoke the command `Vec(lift(Q))` to convert Q into a binary sequence

```
[1, 0, 0, 1, 0, 1, 0],
```

and then complete the binary sequence with zeros (say, on the left-hand side of it) to get a sequence of length divisible by 8, which may be converted to ASCII. In this case it turns out to be just one character:

```
J
```

Both parties may save the sequence of characters thus obtained in a file, say, `shared.txt`. If Alice wants to send Bob `message.txt` first she encrypts the message by invoking the command

```
openssl rc4 -pass file:shared.txt -in message.txt -out message.rc4 -e
```

and sends Bob only `message.rc4`. Then Bob receives the message and invokes the command

```
openssl rc4 -pass file:shared.txt -in message.rc4 -out message.txt -d
```

and decrypts the message.

3.2 Digital signature algorithm

A digital signature algorithm is an electronic avatar of the traditional written signature. It is used to prove to a third party that the information was actually signed by the claimed sender. A special feature of digital signatures (as opposed to written ones) is that they can also be used to verify the integrity of information. Here we shall describe the digital signature algorithm based on modular arithmetic. First we choose domain parameters consisting of the cyclic group $F^\times$, where F is a finite field of prime order p, and a generator g of $F^\times$. As above, the domain parameters are in essence public knowledge. Assume that the message sender, say, Alice has produced her secret key α and has also computed her public key $a = g^\alpha$. (This is just as in the first step of the Diffie-Hellman key exchange protocol discussed above.) Without loss of generality we may assume that the message m is an integer. To sign m Alice chooses another integer $k \in \{1, 2, \ldots, p-2\}$ at random. and computes $r \in \{1, 2, \ldots, p-1\}$ and $s \in \{0, 2, \ldots, p-2\}$ such that

$$r \equiv g^k \pmod p$$

and

$$m \equiv \alpha r + ks \pmod{p-1}.$$

If the latter congruence is not soluble, then we randomly pick another $k \in \{1, 2, \ldots, p-2\}$ and attempt again to find the desired r and s. After successfully obtaining r and s, Alice sends Bob m with the pair (r, s) appended to it. Then Bob receives this information and he needs only verify the relation

$$g^m \equiv a^r r^s \pmod p$$

to check the authenticity of m.

Let us illustrate this algorithm with a simple example. Put $p = 691$. (Of course, in real life applications we would chose a huge prime number p.) It turns out that the class of 2 in F is a generator of the group $F^\times$. Let $F^\times$ with $p = 691$ and $g = 2$ be our domain parameters. Then in GP we invoke

```
p = 691;
g = Mod(2, p);
```

Then Alice produces randomly her secret key, say $\alpha = 155$ and enters it:

```
alpha = 155
```

Then computes her public key by invoking

```
a = g^alpha;
```

That produces the output

```
Mod(246, 691)
```

To sign a message, say $m = 10$, she does as follows. First she generates a another random integer k, subject only to the condition $\gcd(k, p-1) = 1$, say $k = 649$, and then invokes the commands

```
m = 10
r = lift(g^k)
s = lift( (m - alpha*r)*Mod(k, p - 1)^-1 )
```

This yields $(r, s) = (566, 120)$. Now she may send Bob the message with the signature `[r, s]` attached to it. Bob receives the message and its attachment and then decides if it is OK by invoking the command

```
if(g^m == a^r * r^s, print("OK"), print("NOT OK"));
```

3.3 The elliptic curve digital signature algorithm

Now we shall describe the elliptic curve version of the digital signature algorithm. For this purpose we need to define the elliptic curve domain parameters. These consist essentially of an elliptic curve $\bar{A}$ defined over a finite field F, a point $g \in \bar{A}(F)$ of order n, where n is a prime number. As above, these domain parameters are basically public knowledge. The secret key of the sender, say, Alice is a random choice of $\alpha \in \{1, 2, \ldots n-1\}$, and her public key is the point $a = [\alpha]g$. (This is analogous to the first step of the Diffie-Hellman key exchange protocol discussed above.) Again, without loss of generality we may assume that the message m is an integer. In order to sign m Alice needs to chose another integer $k \in \{1, 2, \ldots, p-2\}$ and then compute

$$(x_1, y_1) = [k]g \tag{6}$$

and

$$r = x_1 \pmod{n}.$$

If it turns out that if $r = 0$, then she picks another k at random and goes to Line 6, else

$$s = k^{-1}(m + \alpha r) \pmod{n}$$

If $s = 0$, then she picks another k at random and goes to Line 6. Once she has successfully obtained r and s, she sends m with (r, s) appended to Bob. To verify the signature he needs only to compute

$$\begin{array}{lcl} u_1 & = & ms^{-1} \pmod{n} \\ u_2 & = & rs^{-1} \pmod{n} \\ (x_1, y_1) & = & [u_1]g + [u_2]a \end{array}$$

and the signature is authentic if $r = x_1 \pmod{n}$, and not authentic otherwise.

A measure of the security provided by the elliptic curve domain parameters is given in essence by the size of n. As n is the order of the subgroup of $\bar{A}(F)$ generated by g, we may compute it by first computing $c := |\bar{A}(F)|$ and then computing g^d for certain $d|c$. But if $|F|$ is large, it is not feasible to compute c by merely counting the points $(x, y) \in F \times F$ that satisfy the affine version of its Weierstrass equation (and the point O). Fortunately there are some efficient algorithms that may be used to explicitly compute c when $|F|$ is rather large quite quickly. These algorithms are based on some advanced results coming from arithmetic algebraic geometry. (See the Appendix for further information.) One of these efficient algorithms is implemented in the PARI/GP function `ellap()`. Let us illustrate how this function may be used to compute n with a simple example. Suppose A is the elliptic curve with Weierstrass equation $[0, 0, 1, -1, 0]$. This is elliptic curve **37a1** we discussed before. Let us put $p = 691$, but we hope that the interested reader will pick much larger primes. By basic properties of the trace of Frobenius (see the Appendix) it turns out that c may be computed via

```
A = ellinit([0, 0, 1, -1, 0]);
p = 691;
c = p + 1 - ellap(A, p);
```

Now let us consider the reduction $\bar{A}$ of A modulo $p = 691$

```
Reduced_A = ellinit([0, 0, 1, -1, 0]*Mod(1,p));
```

The point $g = (0, 0)$ generates $A(\mathbb{Q})$. So from a heuristic point of view it is natural to consider $\bar{g}$ in our first attempt. We may compute the order of $\bar{g}$ with the help of

```
m=0;
g = [0, 0];
for(j=1, c, P=ellpow(Reduced_A, g, j); if(P == [0], m = m+1));
n = c/m
```

As the reader may see, $n = 178$ and the point $\bar{g} = (\bar{0}, \bar{0}) \in \bar{A}(F)$ has order $178 = 2 \cdot 89$. Replacing $\bar{g}$ by $[2]\bar{g}$

```
g = elladd(Reduced_A, [0, 0], [0, 0])
```

we get a point of $\bar{A}(F)$ of prime order $n = 89$. Let us put

```
n = n/2
```

Let us define as our domain parameters the elliptic curve $\bar{A}$, the point $g = (\bar{1}, \bar{0})$, and the integer n.

Now we will illustrate the elliptic curve signature algorithm with a simple example based on the above domain parameters. Suppose Alice generates a secret key, say, $\alpha = 55$. Then she computes her public key

```
Public_Key = ellpow(Reduced_A, g, alpha)
```

Now suppose that her message is $m = 10$. The signature of m (with respect to her secret key) may be produced by choosing another random integer $k \in \{1, 2, \ldots, n - 1\}$, say $k = 26$, and then invoking

```
P = ellpow(Reduced_A, g, k);
r = lift(Mod(lift(P[1]), n));
s = lift( (m + alpha*r)*Mod(k, n)^-1 );
```

It turns out that $(r, s) = (69, 6)$, so we may proceed to the next step. Now Bob receives the message m and the signature (r, s) attached to it. He verifies the signature in two steps as follows. The first step is

```
u1 = lift(m*Mod(s, n)^-1);
u2 = lift(r*Mod(s, n)^-1);
Q1 = ellpow(Reduced_A, g, u1);
Q2 = ellpow(Reduced_A, Public_Key, u2);
Q = elladd(Reduced_A, Q1, Q2);
```

For our special case it turns out that Q is the point

```
[Mod(69, 691), Mod(495, 691)]
```

The second step is just

```
if(Mod(r, n) == Mod(lift(Q[1]), n), print("OK"), print("NOT OK"));
```

The reader may invoke the above sequence of commands and see that the output is indeed `OK`.

4 Concluding remarks

4.1 On the index calculus attack applied to elliptic curves

The index calculus algorithm is one of most efficient algorithms to compute the discrete logarithms. This is a probabilistic algorithm which requires a certain relatively small set of prime numbers known as a *factor base* as input. To solve the discrete logarithm problem for the group $E^\times$, where E is a finite field, it is possible to get a suitably small factor base. But if A is an elliptic curve defined over $\mathbb{Q}$ and $\bar{A}$ is its reduction modulo p, the quadratic nature of the height function on $A(\mathbb{Q})$ (briefly discussed by the end of Subsection 2.2.) implies in particular that there are not many points $\bar{P} \in \bar{A}(F)$ that come from points $P \in A(\mathbb{Q})$ having small height $\hat{h}(P)$. This justifies from a heuristic point of view that there is not much hope about the possibility of applying successfully the index calculus algorithm to solve discrete logarithms for cyclic subgroups $C \subset \bar{A}(F)$ of prime order, provided $\bar{A}$ is a "general type" elliptic curve defined over a finite field F. Further details may be found in Victor Miller's paper [3].

4.2 On the generation of elliptic curve domain parameters

In order to obtain commercial grade domain parameters it is required that the order c of the point $g \in \bar{A}(F)$ is a prime number of 163 bits or more, with the field of definition F for A of roughly the same size. It is also required that A is not vulnerable to certain well-known attacks (i.e. the Weil-descent, the MOV, and the additive group attacks). To get an elliptic curve that meets all these requirements one has to test a huge number of elliptic curves generated at randomly. So one needs a fast algorithm to compute the trace of Frobenius at p. The Schoof-Elkies-Atkin algorithm is an efficient algorithm for that purpose. In fact, this algorithm is implemented in `ellap()` of PARI/GP [4]. There are other algorithms to compute the trace of Frobenius for elliptic curves over some specific types of fields, such as Satoh's algorithm for elliptic curves over finite fields of characteristic 2, the so-called binary case. The interested reader may look at the implementation by Kim-Ee Yeoh of Satoh's point counting in

`http://pages.cs.wisc.edu/~yeoh/nt/satoh-fgh.gp`

The above is an implementation that uses the GP scripting language.

5 Appendix

5.1 Finite fields

Suppose E is a finite field. So E is of positive characteristic p (for some prime p) and may thus be regarded as a finite dimensional vector space over a finite field F of p elements. If d is its dimension then the cardinality $|E|$ of E is $|E| = |F^d| = p^d$. Given any prime number p and any positive integer d there is a field F such that $|E| = p^d$. Indeed, if F^{al} is a fixed algebraic closure of a finite field F of p elements, we may cut out a field E by taking the fixed field of the d-fold composition φ^d of the *Frobenius automorphism*

$$\begin{aligned} F^{al} &\longrightarrow F^{al} \\ x &\longmapsto x^p \end{aligned} \tag{7}$$

In particular, the elements of E are precisely the roots of the polynomial $f(X) = X^{p^d} - X$ of degree p^d; since the derivative $f'(X) = -1$ we have $f(X)$ is separable and thus $|E| = p^d$.

Proposition 4 *With the above notation, the map $F \mapsto d$ defines an isomorphism from the lattice of finite subfields of F^{al} and the set of positive integers, the latter equipped with divisibility as order relation.*

Proof. We have just shown that $F \mapsto d$ defines a surjective map onto the set of positive integers. Our next step is to show that it is injective. Suppose K is a subfield of F^{al} such that $|E| = p^d$. For each x in the multiplicative group $K^\times = K - \{0\}$ we have $x^{p^d-1} = 1$. So $x^{p^d} = x$ for all $x \in K$. Therefore every element $x \in K$ is a root of the above polynomial $f(X) = X^{p^d} - X$ and we may conclude that $K = E$. The rest follow from Galois theory and the fact that ϕ generates the Galois group of E over the field F. The details are left as an easy exercise to the reader. ■

5.2 Elliptic curves

Let K be either a field of characteristic 0 or a field of positive characteristic p such that the Frobenius map $x \mapsto x^p$ defines an automorphism of K. An *elliptic curve A over K* is a complete non-singular projective curve A/K of genus 1 with a specified point in $A(K)$. It is customary to denote the specified point O_A. Now let n be a positive integer. The Riemann-Roch theorem implies that the

K-vector space of functions $x \in K(A)$ with at worst a pole of order n at O has dimension n. In particular, there is a function $x \in K(A)$ with a double pole at O_A and no other poles. Similarly, it is easy to see that there is $y \in K(A)$ with a triple pole at O_A and no other poles. Moreover, there must be an algebraic relation of the form

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6, \tag{8}$$

where a_1, a_2,a_3,a_4, and a_6 are suitable elements of K. It is a classical result that

$$P \mapsto \begin{cases} (0:1:0), & \text{if } P = O_A \\ (x(P):y(P):1), & \text{if } P \neq O_A \end{cases}$$

defines an embedding over K from A into the projective plane $\mathbf{P}^2$. The image of the embedding is the locus defined by the *homogeneous Weierstrass equation*

$$Y^2Z + a_1XYZ + a_3YZ^2 = X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3.$$

The *discriminant* $\Delta \in K$ of the Weierstrass equation is defined by the formulæ

$$\begin{array}{rcl} \Delta & = & \frac{1}{1728}(c_4^3 - c_6^2) \\ c_4 & = & b_2^2 - 24b_4 \\ c_6 & = & -b_2^3 + 36b_2b_4 - 216b_6 \\ b_2 & = & a_1^2 + 4a_2 \\ b_4 & = & a_1a_3 + 2a_4 \\ b_6 & = & a_3^2 + 4a_6 \end{array}$$

and it may be shown that $\Delta \neq 0$. Conversely, if we have a Weierstrass equation with coefficients a_1, a_2,a_3,a_4, and a_6 in K such that $\Delta \neq 0$, the curve defined by it in the projective plane $\mathbf{P}^2$ is an elliptic curve A defined over K. We usually denote a Weierstrass equation by $[a_0, a_1, a_2, a_3, a_4, a_6]$ and denote $W = W(a_1, a_2, a_3, a_4, a_6)$ the curve defined by it in $\mathbf{P}^2$. Given Weierstrass equations $W(a_1, a_2, a_3, a_4, a_6)$ and $W' = W(a'_1, a'_2, a'_3, a'_4, a'_6)$ of A there exist $u \in k^\times$ and $r, s, t \in k$ such that

$$\begin{array}{rcl} ua'_1 & = & a_1 + 2s, \\ u^2a'_2 & = & a_2 - sa_1 + 3r - s^2, \\ u^3a'_3 & = & a_3 - ra_1 + 2t, \\ u^4a'_4 & = & a_4 - sa_3 + 2ra_2 - (t + rs)a_1 + 3r^2 - 2st, \\ u^6a'_6 & = & a_6 + ra_4 + 3r + r^2a_2 + r^3 - ta_3 - t^2 - rta_1. \end{array} \tag{9}$$

Given an elliptic curve A defined over $\mathbb{Q}$ described by a Weierstrass equation $[a_0, a_1, a_2, a_3, a_4, a_6]$ we may use the changes of coordinates defined by Equations 9 if necessary to have all its coefficients in $\mathbb{Z}$. Then, given a prime number p we may define the *reduction of* W *modulo* p as the projective curve $\overline{W}$ defined by the Weierstrass equation $[\bar{a}_0, \bar{a}_1, \bar{a}_2, \bar{a}_3, \bar{a}_4, \bar{a}_6]$. Note that $\overline{W}$ becomes a singular curve (and thus not an elliptic curve) if and only if $p|\Delta$. So in order to increase the number of primes p such that the reduced curve $\overline{W}$ is an elliptic curve let us make further changes of coordinates 9 if necessary to have the absolute value $|\Delta|$ of the discriminant Δ of the Weierstrass equation as small as possible. We call such discriminant the *minimal discriminant of* A and denote it Δ_A. The resulting equation is known as a *minimal Weierstrass equation* for $A/\mathbb{Q}$. We say that A has bad reduction at a prime p

if the discriminant of a minimal Weierstrass equation of A is divisible by p. So we may define the *reduction map* $A \longrightarrow \bar{A}$ by $P \mapsto \bar{P}$, where $\bar{P}$ is obtained by reducing (suitably chosen) projective coordinates of P. It is not difficult to see that this is a group homomorphism.

As usual, if K' is a field extension of K, we let $A(K')$ be the locus of points P on A defined over K'. Using again the Riemann-Roch theorem it may be shown that for each algebraic field extension K'/K the canonical map

$$\begin{aligned} E(K') &\longrightarrow \text{Pic}^0(A/K') \\ P &\longmapsto P - O \end{aligned}$$

is actually a bijection. Here if Pic$^0(A/K^{al})$ denotes the free group generated by the set $A(K^{al})$ such that their degree is 0 modulo the divisors (f) of functions $f \in K^{al}(A)$, then Pic$^0(A/K')$ denotes the classes of Pic$^0(A/K^{al})$ defined over K'. This gives the set $A(K')$ a group structure known as the *Mordell-Weil group* of A/K'. By using still once more the Riemann-Roch theorem we may find explicit expressions for the coordinates of the group law.

Suppose A is an elliptic curve over $\mathbb{Q}$. Denote $\bar{A}$ its reduction modulo a prime p. The Frobenius endomorphism $\phi \in$ End(A) defined by $(x, y) \mapsto (x^p, y^p)$ induces a semi-simple operator on the *Tate module* $T_\ell A$, for a prime $\ell \neq p$, and this operator has characteristic polynomial

$$\xi(X) = X^2 - a_A(p)X + p,$$

where $a_A(p) = p + 1 - |\bar{A}(F)|$, and F denotes the integers modulo p. Here, if $A[\ell^k]$ denotes the ℓ^k-torsion of $A(\mathbb{Q}^{al})$ then taking the inverse limit over k defines

$$T_\ell A := \varprojlim A[\ell^k]$$

The integer $a_A(p)$ is known as the *trace of Frobenius*. As mentioned in the above section, the Schoof-Elkies-Atkin algorithm is an efficient algorithm to compute $a_A(p)$, which is implemented in the PARI/GP function `ellap()`.

5.3 Installing PARI/GP

The computer algebra system PARI/GP [4] is free software that runs on most common operating systems. It is designed for fast computations in number theory, but also contains many other useful functions to compute, such as matrices, polynomials, power series, and also transcendental functions. It is distributed under the GNU General Public License and is available in the most popular GNU/Linux distributions. To install Pari/GP, together with the J. E. Cremona elliptic curve data package on a computer running Fedora it suffices to invoke the command

```
yum install pari-gp pari-elldata
```

Alternatively, the reader may consider downloading the PARI/GP source code and compiling it. This may be done with the help of an `ANSI C` or a `C++` compiler such as the GNU Compiler Collection. The PARI/GP kernel may take advantage of optimizations this compiler provides. It is useful to build PARI with the GNU MP library, which is a library for arbitrary precision arithmetic and the GNU readline library, which provides line editing under GP.

Acknowledgements: The author wishes to heartily thank José Antonio Vallejo for his kind invitation and generosity, and also the Facultad the Ciencias of the Universidad de San Luis Potosí for their full financial support and an excellent working environment.

References

[1] J. E. Cremona, *Elliptic curves of conductor* $\leq 129,999$, http://www.maths.nott.ac.uk/personal/jec/ftp/data/.

[2] T. Hansen and G. L. Mullen, *Primitive polynomials over finite fields*, Math. Comp. **59** (1992), no. 200, 639–643.

[3] V. S. Miller, *Use of elliptic curves in cryptography*, Lecture notes in computer sciences; 218 on Advances in cryptology—CRYPTO 85 (New York, NY, USA), Springer-Verlag New York, Inc., 1986, pp. 417–426.

[4] The PARI Group, Bordeaux, *PARI/GP, Version 2.3.5*, http://pari.math.u-bordeaux.fr.

[5] J. H. Silverman, *The arithmetic of elliptic curves*, Graduate Texts in Mathematics, vol. 106, Springer-Verlag, New York, 1992, Corrected reprint of the 1986 original.

[6] ———, *Advanced topics in the arithmetic of elliptic curves*, Springer-Verlag, New York, 1994.

[7] L.C. Washington, *Elliptic curves: Number theory and cryptography*, Discrete Mathematics and Its Applications Series, CRC PressINC, 2008.