TUTORIAL: ELLIPTIC CURVES OVER FINITE FIELDS
IN PARI/GP

BILL ALLOMBERT AND KARIM BELABAS

This is a mostly self-contained introduction to computations in curves over finite fields in PARI/GP. PARI/GP is free software computer algebra system available at http://pari.math.u-bordeaux.fr/download.html

In the following, ? denote the GP prompt and \ denote GP comments. A text version of the list of commands to enter is available at http://pari.math.u-bordeaux.fr/Events/PARI2017c/talks/ecc.gp

Explanation of the syntax is in the appendix.

It is recommended to use PARI/GP 2.7 or later.

Please do
? \l ecc.log

to enable logging of your input for future reference.

1. PRIME FINITE FIELDS

To create a random prime number:
? p=randomprime(2^100)
%1 = 792438309994299602682608069491

To create an element of $\mathbb{F}_p$:
? a=Mod(2,p)
? a^(p-1) \ powering
%3 = Mod(1,792438309994299602682608069491)

2. GENERAL FINITE FIELDS

To build an irreducible polynomial of degree $n$ of $\mathbb{F}_p$, use $\text{ffinit}(p,n)$.

? P=ffinit(13,2)
%4 = Mod(1,13)*x^2+Mod(1,13)*x+Mod(12,13)
? polisirreducible(P)
%5 = 1

To build an element of $\mathbb{F}_{p^n}$ (also work for $n = 1$) from its minimal polynomial:
? a=ffgen(P,'a)
The above can be abbreviated by $\text{ffgen}(p^n,'a)$

? a=ffgen(13^7,'a)

Basic operations:
2.1. Exercice. Compute a discrete logarithm in $F_{2^{127}}$ and see how much time it takes. Use # to activate the timer or ## to see the time of the last command.

3. Elliptic curves over finite fields

From a short Weierstrass model $y^2 = x^3 + a_4 x + a_6$:
\[
E_s = \text{ellinit}([a^4, a^6], a);
\]

From a long Weierstrass model $y^2 + a_1 xy + a_3 y = x^3 + a_2 x^2 + a_4 x + a_6$:
\[
E = \text{ellinit}([a, a^2, a^3, a^4, a^6], a);
\]

Basic functions:
\[
E.j \quad \text{j-invariant}
\]
Structure of the group $E(F_q)$:
\[
\text{ellcard}(E) \quad \text{cardinal of } E(F_q)
\]
\[
\text{ellgroup}(E) \quad \text{structure of } E(F_q)
\]
Above $[d_1, d_2]$ means $\mathbb{Z}/d_1 \mathbb{Z} \times \mathbb{Z}/d_2 \mathbb{Z}$, with $d_2 | d_1$.
\[
G = \text{ellgenerators}(E) \quad \text{generators of } E(F_q)
\]
A minimal generating set.
\[
P = \text{random}(E) \quad \text{random point on } E(F_q)
\]
\[
Q = \text{random}(E) \quad \text{another random point on } E(F_q)
\]
\[
\text{ellisoncurve}(E, P) \quad \text{check that the point is on the curve}
\]
\[
\text{elladd}(E, P, Q) \quad \text{P+Q in } E
\]
\[
\text{ellmul}(E, P, 100) \quad \text{100.P in } E
\]
\[
oP = \text{ellorder}(E,P) \quad \text{order of } P
\]
\[
oQ = \text{ellorder}(E,Q) \quad \text{order of } Q
\]
\[
o = \text{lcm}(oP,oQ); \quad \text{both } P \text{ and } Q \text{ are in } E(F_q)[o]
\]
\[
w = \text{ellweilpairing}(E,P,Q,o) \quad \text{Weil pairing of } P \text{ and } Q \text{ of order } o
\]
\[
forder(w)
\]
\[
nP = \text{ellmul}(E, P, \text{random}(o));
\]
\[
n = \text{elllog}(E,nP,P)
\]
\[
\text{ellmul}(E,P,n) == nP
\]
4. Application: Speed of Discrete Logarithm

We want to compare the speed of discrete logarithm over $\mathbb{F}_p$ and $E(\mathbb{F}_p)$, for $p$ of 30, 40 and 45 bits. For fairness, we need to use groups of nearly prime orders.

```plaintext
? until(isprime((p-1)/2), p=randomprime(2^30)); p
? g=ffprimroot(ffgen(p))^2; a=g^random(p);
? fflog(a,g)
? ##
```

5. Application: the MOV attack on the discrete logarithm problem

We build curves where the Weil pairing allows us to reduce the discrete logarithm problem on the curve to a discrete logarithm problem on a finite field. This is the idea behind the MOV attack of Menezes, Okamoto, and Vanstone.

5.1. Example over $\mathbb{F}_p$. We will use the curve $E : y^2 = x^3 + x$ over $\mathbb{F}_p$ where $p = 4n^2 + 1$ for some integer $n$, which has the property that $E(\mathbb{F}_p)$ is isomorphic to $\mathbb{Z}/2n\mathbb{Z} \times \mathbb{Z}/2n\mathbb{Z}$. We choose $n$ to be prime. To find suitable $n$ of 50 bits:

```plaintext
? until(isprime(p),n=randomprime(2^50);p=1+4*n^2); p
? a=ffgen(p); E=ellinit([1,0],a);
? ellgroup(E)
? [P,Q] = ellgenerators(E); \ \% name P and Q the generators.
```

We use the Weil pairing with the second generator to solve the discrete logarithm in the group generated by the first one:

```plaintext
? e = random(2*n)
? R = ellmul(E,P,e);
? wR = ellweilpairing(E,Q,R,2*n);
? wP = ellweilpairing(E,Q,P,2*n);
? default(parisize,"32M");
? fflog(wR,wP,2*n)
? ##
```

5.2. Example over $\mathbb{F}_{p^2}$. We will use the curve $E : y^2 = x^3 + x$ and a prime $p \equiv 3 \pmod{4}$, so that $E$ is supersingular, of order $p+1$. To solve a discrete logarithm problem in $E(\mathbb{F}_p)$, we embed it in $E(\mathbb{F}_{p^2})$ which isomorphic to $\mathbb{Z}/(p+1)\mathbb{Z} \times \mathbb{Z}/(p+1)\mathbb{Z}$, so we can use the Weil pairing in $E(\mathbb{F}_{p^2})$. We choose $(p+1)/4$ to be prime.
? until(p%4==3 && isprime((p+1)/4),p=randomprime(2^52));
? a=ffgen(p^2,'a);
? E=ellinit([1,0],p); \ E(F_p)
? ellgroup(E)
? [P] = ellgenerators(E)
? E2=ellinit([1,0],a); \ E(F_p^2)
? [m,m]=ellgroup(E2)
? [P1,Q] = ellgenerators(E2)
? ellweilpairing(E2,P,Q,m);
? e = random(m)
? R = ellmul(E,P,e);
? wR = ellweilpairing(E2,Q,R,m);
? wP = ellweilpairing(E2,Q,P,m);
? fflog(wR,wP,m)
? ##
? elllog(E,R,P,m)
? ##

5.3. Exercises.

- Do the same for a supersingular curve over $F_{341}$.
- Write a function that compute the group structure of $E(F_q)$ using the Weil pairing. See the syntax for functions in the appendix.
- Implement the Pollard rho algorithm to compute discrete logarithm on $E(F_q)$.

6. To know more about PARI/GP

See our website http://pari.math.u-bordeaux.fr or attend the next workshop Atelier PARI/GP.
Cheat sheet

APPENDIX A. BASIC INPUT

From a terminal, typing `gp` starts the interpreter.

1+1 basic operation
1+1; basic operation, no printout
quit or \q exit the interpreter
? online help
??function extended online help about function
# start timer
read("file") or \r file load file in interpreter

Line by line evaluation. Enclose multi-line statements between {...}.

APPENDIX B. BASIC COMMANDS

/*...*/ comment
a = 1 assignment
[u,v,d] = gcdext(5,8) simultaneous assignment (d = 1, u = −3, v = 2)
a == 1 equality test
%,%5 last result, 5-th result in history
||,&&,! boolean operators (or, and, not)
i++,i-- increase i by 1, −1, 2
v[i] i-th component of vector v
print(x, ":", y) outputs the value of x, a colon, then y

APPENDIX C. CONSTRUCTORS

random([100,200]) a uniform integer in [100,200]
vector(5, i, 2*i+1) \{2i + 1: 1 \leq i \leq 5\} = \{3, 5, 7, 9, 11\}
Mod(2,N) \bar{2} in \mathbb{Z}/N\mathbb{Z}
g = ffgen(t^2+Mod(1,3)) \bar{t} in \mathbb{F}_3[t]/(t^2 + 1) = \mathbb{F}_9

APPENDIX D. FLOW CONTROL

for(i = 1, 10, print(i)) /* 1,2,...,10 */
i = 8; while(!isprime(i), i++)
until(isprime(p), p = random(1000))

if (isprime(p), print("yes")
{ if (isprime(p), /*then*/ print("yes")
, /*else*/ print("no")); }

APPENDIX E. USER FUNCTIONS

f(x,y) =
{ my(z = x + y); /* local variable */
  if (z < 0, return (-1));
  return (1);
}
