Lab Tutorial (SageMath)

SageMath is a Python library with a lot of built-in Math functions. You are going to use SageMath in this course to verify your results.

How to use Sagemath

For this course, you have multiple options to use SageMath.

Option 1: You can use this link to use SageMath online without installing any software. This is the easiest option to start working with SageMath. The only downside is you cannot save your code. This should be more than enough for this course.

Option 2: You can use a cloud server for SageMath. This server has the feature to save your code if you create an account.

Option 3: Download SageMath locally on your PC. Running SageMath code will be fastest in here as it will not depend on a free online server to do run your code.

You are free to use whatever option you prefer but for this course, the code you will run does not require a lot of CPU power so the first 2 options are more than enough.

SageMath Functions

The basic arithmetic operations $(+, -, *, /)$ are exactly like in Python. Below is a list of helpful functions for SageMath that you will use in the course. You are also free to use Python features to help you with verifying your results. You can find some useful Python tutorials here.

Basic Functions

```
# Compute Modulo c = a mod b
c = a % b
OR
c = mod(a,b)

# Compute inverse modulo c= a^-1 mod b
c = inverse_mod(a,b)

# GCD of two integers c=gcd(a,b)
c = gcd(a,b)
```

```
# Extended Euclidean algorithm parameter of two integers
c = xgcd(a,b)


# Generate a list of bits for an integer
c = a.bits()

# Generate a list of digits of base b (b=2 is the .bits function) for an integer and
extend to size n (fill 0s)
c = a.digits(base=b,padto=n)


# Convert list back to Integer (Reverse of previous function)
a = Integer(c, base=b)

# Get number of bits of an Integer
c = a.nbits()

# Get number of digits of an Integer with base b (b=2 is the .nbits function)
c = a.ndigits()


# Create Field $F_p$
K = GF(p)

# Convert to hex
hex(a)
OR
a.hex()

# Random Integer
c = ZZ.random_element()

# Sometimes values will be printed as fraction or irrational value. If you want decim
al approximation of value, use this function
c = Numerical_approx(a)
```

Elliptic Curve Functions

```
# Create an Elliptic Curve over $F_p$ with equation: $y^2 = x^3 + ax + b$
E = EllipticCurve(GF(p),[a,b])

# Point at infinity.
# Note when printing point at infinity, sage will show (0:1:0)
O = E(0)

# Point with coordinates (x,y).
# Note when printing any point other than point at infinity, sage will show (x:y:1)
P = E(x,y)

# Point Addition R=P+Q: Adding two points P and Q
R = P + Q

# Point Doubling R=P+P=[2]P: Doubling a point P
R = 2*P

# Point Multiplication R=[s]P: Multiplying P by scalar s
R = s*P

# Point Inversion R=-P: P+R = point at infinity
R = -P

# Order of point P :  k*P = point at infinity
k = P.order()

# Number of bits of number k. Note: number of bits of k is ⌈log₂ k⌉.
k.nbits()
```