Course Project
EECS 475
Fall 2015
Implementing an Elliptic Curve Cryptosystem
Part 2

November 8, 2015

1 Implementation Using Uberzahl

Uberzahl, from the German "zahl" meaning number, is an arbitrary precision (multiprecision) integer class previously written by 475 GSIs. Thanks to operator overloading, for many cases, you can use these just like your familiar basic types:

1. Print with `<<`
2. Combine two uberzahls with `+, -, *, /, %`
3. Negate with unary `-`
4. Compare two uberzahls with `>, <, >=, <=, ==, !=`
5. If you wish to do bit operations on them, you can use `|, &, ^, >>, <<`. Output for the first 3 are always positive.

If you decide to do something with random uberzahls, there are two ways to get random bits. The member function `random(int n)` will set the current uberzahl to a random number with exactly n bits (no leading 0s). The global function `random(uberzahl min, uberzahl max)` will return a random uberzahl in the range [min,max).

When hardcoding constants for uberzahls, such as `x+1` or `x==0`, the easiest way to make it recognize the constants as a type uberzahl is using strings (e.g. `x+"1"` and `x=="0"`).

If you have further questions about uberzahl, feel free to ask them on Piazza.

For the assignment you will need to find a computer or network with a full installation of the g++ compiler. CAEN machines running Linux are your best bet.

You then need to download five files from the course CTools web site:

```
ec.cpp
ec_ops.h
ec_ops.cpp
uberzahl.h
uberzahl.cpp
```

These files already contain a number of the include statements, definitions, and functions you will need to use uberzahl in implementing an elliptic curve cryptosystem. The file `ec_ops.h` contains a number parameter definitions for some of the constants mentioned in the previous sections and three class definition headers for the classes `Zp, ECpoint`, and `ECsystem`. `Zp` is the class for doing arithmetic in GF(p). Notice that the parameters of type `uberzahl` are multiprecision integers, meaning they may be so long that they must be stored in several words of memory. `ECpoint` is the class for defining an elliptic curve point (essentially a boolean indicating whether it is the point at infinity, and, if not, a pair (x, y) where x and y are both `Zp` objects). `ECsystem` is the class for elliptic curve cryptosystems; it supports encryption and decryption. The file `ec_ops.cpp` has definitions for the methods used by these classes. You will have to take some time to figure out what these methods do. Some are, for example, definitions of overloaded operators so you can add `Zp` values, use them in standard input and output, etc.

The file `ec.cpp` is the only file that is not complete. The definitions of seven functions have been left blank. You will have to write these parts. The methods you will have to implement are the following.

1. For class `Zp` you will implement a method `inverse()` that uses the Extended Euclidean Algorithm to return the inverse mod PRIME.
2. For class `Zp` you will implement a method `power()` that computes a power mod PRIME using repeated squaring.
3. For class `ECpoint` you will define an operation + (clearly overloading) that adds another point.
4. For class `ECpoint` you will implement a method `repeatSum` that adds a point p to itself v times.
5. For the class `ECpoint` you will implement a method `pointDecompress` for decompressing a compressed point.
6. For class `ECsystem` you will implement the `encrypt` operation whose parameters are the public key (a point on the elliptic curve), the private key to be used as the random exponent, and two plaintext blocks.
7. For class `ECsystem` you will implement the `decrypt` operation.

The strategy you should pursue is to implement these methods one at a time in the order given. You may want to rewrite `main` temporarily to test your code.

To compile your program, you will type

```
g++ ec.cpp ec_ops.cpp uberzahl.cpp
```

You will not need to write a lot of code to complete this assignment, but there are some conceptual problems you will have to overcome, so you should not put this off to the final few days. If you are not an experienced C++ programmer, you may need to consult the GSI and professor on some of the finer points of object oriented programming.