Course Project
EECS 475
Fall 2015
Implementing an Elliptic Curve Cryptosystem
Part 1

November 7, 2015

1 Introduction

For this project you will use the Uberzahl library to implement a variant of the Elliptic Curve Cryptosystem called the Menezies-Vanstone Elliptic Curve Cryptosystem. The due date for the project is 5 pm, Thursday December 3.

From class you know that the elliptic curve $E_p(a, b)$ is the set of points (x, y) satisfying the equation

$$y^2 = x^3 + ax + b,$$

where x, y, a, b are in the field GF(p); in addition, $E_p(a, b)$ contains one other point, the *point at infinity* $\mathcal{O}$. (The equation above is correct as long as p is not a power of 2 or 3. We also require that $4a^3 + 27b^2 \neq 0$ for the definitions below to work.)

If $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$ are two points on $E_p(a, b)$ (and neither is $\mathcal{O}$), the the sum of these points $R = P + Q$ is the point $R = (x_R, y_R)$ defined as follows.

1. If $P \neq Q$ and $x_P \neq x_Q$, define

$$\begin{aligned} \lambda &= (y_Q - y_P)(x_Q - x_P)^{-1} \\ x_R &= \lambda^2 - x_P - x_Q \\ y_R &= -y_P + \lambda(x_P - x_R) \end{aligned}$$

2. If $P = Q$ and $2y_p \neq 0$, define

$$\begin{aligned} \lambda &= (3x_P^2 + a)(2y_P)^{-1} \\ x_R &= \lambda^2 - 2x_P \\ y_R &= -y_P + \lambda(x_P - x_R) \end{aligned}$$

3. In other cases, $P + Q = \mathcal{O}$.

If either P or Q (or both) is $\mathcal{O}$, we define $P + \mathcal{O} = \mathcal{O} + P = P$.

As we have seen, an elliptic curve with the binary operation + is a group. It may not be a cyclic group, but a deep result (that we will not prove in this course) states that it is either cyclic or is the Cartesian product of two cyclic groups. The elliptic curve group you will use for this assignment is cyclic.

Most researchers believe that the discrete log problem is hard for cyclic elliptic curve groups (at least for some parameters p, a, b) and, hence, they are suitable for cryptographic applications such a public key cryptography, key distribution, and digital signatures.

Suppose $E_p(a, b)$ is a cyclic group with generator $G = (x_G, y_G)$ and $|E_p(a, b)| = N$. Since we use additive notation, rather than the usual multiplicative notation, we use multiples of points P on the curve, rather than powers. That is, for any nonnegative integer X, we define

$$X \cdot P = \underbrace{P + P + \cdots + P}_{X \text{ addends}}.$$

Suppose $R = (x_R, y_R)$ is a point on $E_p(a, b)$. Then so is $-R = (x_R, p - y_R)$, since $p - y_R \equiv -y_R \pmod{p}$ and $y_P^2 \equiv (-y_P)^2 \pmod{p}$. There are no other values for y other than these two such that (x_R, y) is a point on the curve. Now the sum of y_R and $p - y_R$ is p, an odd number, so one of these values is even and the other is odd. This gives us a method to compress the representation of point on an elliptic curve. Rather than storing a pair (x_R, y_R), let b_R be just the bit ($y_R \bmod 2$) and store the *compressed point* $\gamma(R) = x_R b_R$, the concatenation of the binary representation of x_R with the bit b_R. (For some purposes we also view this as the integer $2 \cdot X_R + b_R$.) $\gamma(R)$ uses slightly more than half number of bits needed to explicitly store the coordinates of R.

Can we recover $R = (x_R, y_R)$ from $\gamma(R) = x_R b_R$ in a reasonable amount of time? Since

$$y_R^2 = x_R^3 + a x_R + b$$

from the elliptic curve equation, this means we must be able to find square roots efficiently in $GF(p)$. It turns out that if p is a prime and $p \equiv 3 \pmod{4}$, computing square roots is easy. In this case, if $z \in GF(p)$ is a *quadratic residue* (i.e., has square roots), one of its square roots is

$$z^{(p+1)/4} \bmod p$$

and the other is

$$p - z^{(p+1)/4} \bmod p.$$

One of these is even and the other is odd, so take whichever one has b_R as its low order bit. This allows us to compute the *decompression function* $\lambda(x_R b_R) = (x_R, y_R)$.

There are three algorithms associated with the Elliptic Curve Cryptosystem: the key generation algorithm, which is a randomized algorithm generating a private key/public key pair; the encryption algorithm, which is a randomized algorithm for computing a ciphertext C from a plaintext M using the public key; and the decryption algorithm for obtaining the plaintext M for a ciphertext C using the private key. In this version, we will assume that $p < 2^k$ but that $2^k - p$ is very small. Here $k + 1$ will be the block length.

Key Generation Algorithm

1. Generate a random integer Y, $0 \le Y \le N - 1$.
2. Compute $P = Y \cdot G = (x_P, y_P)$ using repeated squaring. (Since we are using additive notation here and below, we are really using repeated doubling.)

Return: Public key $(E_p(a, b), G, P)$ and private key $(E_p(a, b), G, Y)$.

Encryption Algorithm

Given: plaintext block M (a bit string of length $2k$) and the public key. Divide M into two blocks M_0 and M_1 of length k: $M = M_1 M_2$.

1. Generate a random integer X, $0 \leq X \leq N-1$.
2. Compute $Q = X \cdot G = (x_Q, y_Q)$ and $R = X \cdot P = (X \cdot Y) \cdot G = (x_R, y_R)$ using repeated squaring.
3. Set $C_0 = M_0 \cdot x_R \bmod p$, $C_1 = M_1 \cdot y_R \bmod p$, and $C_2 = \gamma(Q)$.

Return: Ciphertext $C = (C_0, C_1, C_2)$.

Thus, the total number of bits in the ciphertext is 769 $(256 + 256 + 257)$. Notice that this definition differs from the usual El Gamal encryption algorithm where M would be an element of the elliptive curve group. It overcomes the difficulty with that not every bit string of the appropriate length is actually a group element. The reason the definition above works is that R looks random, so we use it in the same way we would use two randomly chosen chosen integers in the range $0 \leq x < p$. Another version of the Elliptic Curve Cryptosystem (which you will *not* use in this project) would return ciphertext (C_0, C_1) where

$$\begin{aligned} C_0 &= \gamma(Q) \\ C_1 &= M \cdot \gamma(R) \bmod p \end{aligned}$$

Decryption Algorithm

Given: Ciphertext (C_0, C_1, C_2) and the private key.

1. Compute $R = Y \cdot \lambda(C_2) = (x_R, y_R)$ using the square root algorithm and repeated squaring.
2. Set $M_0 = C_0 \cdot x_R^{-1} \bmod p$ and $M_1 = C_1 \cdot y_R^{-1} \bmod p$.

Return: Plaintext $M = M_0 M_1$.

Fields and Curves Used for this Assignment.

The elliptic curve you will be using is called w-256-mers. It is one of the curves investigated by Joppe W. Bos, Craig Costello, Patrick Longa and Michael Naehrig in a paper titled "Selecting Elliptic Curves for Cryptography: An Efficiency and Security Analysis." This is Report 2014/130 in the Cryptology ePrint Archive at: http://eprint.iacr.org/2014/130.

You will be working over the field GF(p), where the prime $p = 2^{256} - 189$. As remarked in class, we usually pick prime powers that are nearly a power of 2, and p is very close to 2^{256}. The binary representation of $2^{256} - 1$ is 256 consecutive 1's. Subtracting 188 from this simply flips a few of the lower order bits.) In hexadecimal, p is

FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFF43.

The probability that a randomly chosen 256-bit string represents an integer less than p is $p/2^{256} = 1 - 189/2^{256}$, which is very close to 1. Thus, p is a very convenient modulus to use with 256-bit blocks. In decimal, p is

115792089237316195423570985008687907853269984665640564039457584007913129639747

The elliptic curve is $y^2 = x^3 + ax + b$, where $a = -3$ and b is

25581

in hexadecimal, or

152961

in decimal. The number of points on the curve is

FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF E43C8275 EA265C60 20AB2029 4751A825

in hexadecimal, or

115792089237316195423570985008687907853233080465625507841270369819257950283813

in decimal. This is a prime number so the group is cyclic.

The generator we use is (x_G, y_G), where x_G is

004F734B 6790D3E5 247FE3DD B4D61F50 24BA2AC0 4BBC1182 3A5D9895 1DC8B48B

and y_G is

10406987 8EEB9C02 D4972630 BF469384 7658942D 8784174F 7C2281E3 F4A8FCC4

in hexadecimal, or x_G is

140376651850118021374176667111932786297688265892421886295983839501296907403

and y_G is

735081212748856139538012186185644182601047321917539197020010157368044217875 6

in decimal.