MATH 3CY3: Cryptography, Assignment 3

Authors: Steven Gonder, Pesara Amarasekera

An elliptic curve represents the solutions to the equation $Y^2 = X^3 + AX + B$, along with a point at infinity, O, and the condition that the discriminant $D = 4a^3 - 27b^2$ be non-zero. While cryptography in some form has existed for millenia, it wasn't until 1985 that Neal Koblitz and Victor Miller independently proposed using elliptic curves (Hankerson et al, 2006, pg. 1). On the other hand Elliptic curves have been studied quite extensively in the 19th century. The problem of integer factorization has been of vast intrigue in the cryptographic community, thus Lenstra's elliptic curve factorization algorithm, introduced in the early 1980's, helped popularize Elliptic Curves in the community. Elliptic curves are frequently used as a finite-field cryptosystem, based around an elliptic curve variation of the discrete log problem. These elliptic curve cryptosystems eventually gained popularity as an alternative to RSA cryptosystems, because elliptic curve cryptography offered similar security to RSA at much smaller key length sizes (Hankerson et al, 2006, pg. 18-19), thereby making asymmetric cryptography more performant and an ideal candidate for securing information.

The conventional discrete logarithm problem in a finite field F^*_p is the difficulty of solving for integer x in $h \equiv g^x \pmod p$, with integers (h, g, p) known and p being prime. The analogous Elliptic Curve Discrete Logarithm Problem is the difficulty of determining an integer quantity n of point additions that solves $Q = nP = (P_1 + P_2 + \dots + P_{n-1} + P_n)$, with (Q, P) known and being points on the elliptic curve (Hoffstein et al, 2014, pg. 310-311).

A crucial operation on an elliptic curve is the aforementioned point addition. Given two points on a curve, the algorithm draws a chord between the two points, and extends that chord until it intersects a third point on the curve. This third intersection is possible as the elliptic curve is of degree 3, and in the case that the chord is vertical, we define O, the point at infinity, as being on the curve as well. The final step of the point addition algorithm: the third point of intersection is reflected across the x-axis.

When n is known, an nP multiplication is trivial to compute or verify as there exist algorithms for determining it efficiently, such as the "double-and-add" method for fast repeated point addition (Hoffstein et al, 2014, pg. 312-313). However, ECDLP is difficult to solve when n is secret, because brute-forcing DLPs iteratively should generate seemingly arbitrary values or points. Additionally, there are no known algorithms that can solve the general case of ECDLP faster than roughly square root of p steps, in $E(F_p)$ (Hoffstein et al, 2014, pg. 316).

Now that we have some background on elliptic curve cryptosystems, we will use the relatively simple Elliptic Diffie-Hellman Key Exchange to run through the process and provide a numerical example.

Alice and Bob choose the same elliptic curve $E(F_p)$, hopefully one without a trivial weakness or "backdoor." They also agree to choose the same point P on the curve. Both the curve and the point P are public. Alice and Bob then each decide on their own unique private keys – some integers n_A, n_B -- and keep these keys secret.

Alice computes $Q_A = n_A P$ and sends to Bob. Bob computes $Q_B = n_B P$ and sends to Alice. Alice and Bob then multiply their own private key with what they received from each other.

$n_A Q_B = n_A(n_B P) = n_A(P n_B) = (n_A P) n_B = Q_A n_B = n_B Q_A$, which shows that both Alice and Bob can determine $(n_A n_B)$. The application of this shared product is that $(n_A n_B)P$ represents a shared, secret point on the curve, and Alice and Bob discard the y-coordinate and keep the x-coordinate as the relevant secret detail (Hoffman et al, 2014, pg. 316-317).

Now for a concrete example: suppose that Alice and Bob choose the curve $y^2 \equiv x^3 + 4x + 4 \pmod{101}$, and agree publically on the point P = (2, 11).

In the programming language Haskell, we confirm that the point does indeed reside on the curve. For simplicity we are using integer points on elliptic curves, rather than rationals more generally.

```
type XCoordinate = Integer
type YCoordinate = Integer
type Coefficient = Integer
type Fp = Integer

data Point = Point { x :: XCoordinate, y :: YCoordinate }
   deriving Show
data EllipticCurve =
    EllipticCurve { a :: Coefficient, b :: Coefficient, p :: Point }

isEllipticCurvePoint :: EllipticCurve -> Fp -> Bool
isEllipticCurvePoint (EllipticCurve a b (Point x y)) fp =
    y^2 `mod` fp == (x^3 + a*x + b) `mod` fp
 && 4*a^3 - 27*b^2 /= 0
```

```
*Main> isEllipticCurvePoint (EllipticCurve 4 4 (Point 2 11)) 101
True
```

Alice now chooses her private key $n_A = 4$, and Bob chooses his private key $n_B = 16$. Alice computes 4P and sends this point to Bob, and Bob

computes 16P and sends this point to Alice. For the sake of simplicity, the private keys are powers of two, so we can use the following point doubling algorithm (Paar et al, 2010, pg. 244-245) without additional overhead.

```
-- Brute force for simplicity
modInv :: Integer -> Fp -> Integer
modInv n fp = let modInv' n fp m = if n*m `mod` fp == 1
                                   then m
                                   else modInv' n fp (m+1)
              in modInv' n fp 0

pointDoubling :: EllipticCurve -> Fp -> Point
pointDoubling (EllipticCurve a b (Point x1 y1)) fp =
    let Point x2 y2 = Point x1 y1
        slope = (3*x1^2 + a) * (modInv (2*y1) fp)
        x3 = (slope^2 - x1 - x2) `mod` fp
        y3 = (slope*(x1 - x3) - y1) `mod` fp
    in Point x3 y3
```

(Note that for public applications of cryptography, using branching conditions can create vulnerabilities and we would often use branchless computation instead)

Remembering that we are given the point P = 1P to start with, Alice by repeated doubling computes:

```
*Main> pointDoubling (EllipticCurve 4 4 (Point 2 11)) 101
Point {x = 80, y = 70}
*Main> pointDoubling (EllipticCurve 4 4 (Point 80 70)) 101
Point {x = 41, y = 45}
```

And sends the point A = (41, 45) to Bob.

Bob by repeated doubling computes:

```
*Main> pointDoubling (EllipticCurve 4 4 (Point 2 11)) 101
Point {x = 80, y = 70}
*Main> pointDoubling (EllipticCurve 4 4 (Point 80 70)) 101
Point {x = 41, y = 45}
*Main> pointDoubling (EllipticCurve 4 4 (Point 41 45)) 101
Point {x = 75, y = 10}
*Main> pointDoubling (EllipticCurve 4 4 (Point 75 10)) 101
Point {x = 16, y = 86}
```

And sends the point B = (16, 86) to Alice.

Alice multiplies computes 4B = 64P:

```
*Main> pointDoubling (EllipticCurve 4 4 (Point 16 86)) 101
Point {x = 26, y = 3}
*Main> pointDoubling (EllipticCurve 4 4 (Point 26 3)) 101
Point {x = 53, y = 44}
```

Bob computes 16A = 64P:

```
*Main> pointDoubling (EllipticCurve 4 4 (Point 41 45)) 101
Point {x = 75, y = 10}
*Main> pointDoubling (EllipticCurve 4 4 (Point 75 10)) 101
Point {x = 16, y = 86}
*Main> pointDoubling (EllipticCurve 4 4 (Point 16 86)) 101
Point {x = 26, y = 3}
*Main> pointDoubling (EllipticCurve 4 4 (Point 26 3)) 101
Point {x = 53, y = 44}
```

Both Alice and Bob arrive at the secret point (53, 44), and discard the y-coordinate, retaining the x-coordinate as the secret value.

The eavesdropper without knowing n_A or n_B would have to determine how many repeated point additions are needed to reach (53, 44) from (2, 11). With carefully selected elliptic curves, the ECDLP has no known efficient classical algorithms to do so.

Long-term security of elliptic curve cryptography

Currently used Elliptic curve cryptography schemes are susceptible to attacks by a theoretical Quantum Computer (however such a Quantum computer has not been built yet). This is because Shor's algorithm can be used to break the DLP in a feasible amount of time (rendering most cryptographic schemes that rely on the intractability of the DLP

to be useless) (Hoffstein et al, 2014, pg. 497). However, the implementation of a practical quantum computer for performing large enough factorizations is still a decade or so away. This leads towards the development of post-quantum cryptographic schemes, and quantum cryptography itself.

Programs and Companies that use Elliptic Curve Cryptography

Elliptic Curve Cryptography is quite popular these days. Most notably, in the present day, ECC is used in Bitcoin. ECC and RSA took different routes to be accepted. While RSA was initially patented, ECC was not. This led companies to apply for patents giving improvements on ECC, most notably Certicom. However not all the improvements patented could be considered significant, and due to the many patents that were made, there has been uncertainty as to which versions of ECC are free and which must be licensed. (Hoffstein et al, 2014, pg. 321). It should be noted the use of ECC is shrouded by the fact that the difficulty of the problems that underlie the scheme are not well known, as opposed to RSA. And in a way the theory underlying ECC is more complex than that of RSA, thus it can be the case that there can be more vulnerabilities for using ECC than that are known, simply because it hasn't necessarily been subjected to as much scrutiny.

Bibliography

Hankerson, D., Menezes, A. J., & Vanstone, S. (2006). *Guide to elliptic curve cryptography*. Springer Science & Business Media. https://books.google.ca/books?hl=en&lr=&id=V5oACAAAQBAJ&oi=fnd&pg=PR12&dq=elliptic+curve+history&ots=kFsxQi4G-D&sig=mfkynwgUAvRvUriuG9YaMs2zHrE&redir_esc=y#v=onepage&q=elliptic%20curve%20history&f=false

Hoffstein, J., Pipher, J., & Silverman, J. H. (2014). *An introduction to mathematical cryptography* (2nd ed.). Springer.

Paar, C., & Pelzl, J. (2010). *Understanding cryptography: A textbook for students and practitioners* (2nd ed.). Springer Science & Business Media. https://books.google.ca/books?hl=en&lr=&id=f24wFELSzkoC&oi=fnd&pg=PR2&dq=understanding+cryptography&ots=YY231AevRr&sig=d4JDzG3FUWl42vxMl4WS3q-rPBw&redir_esc=y#v=onepage&q=understanding%20cryptography&f=false