CSE/AMS 547 Discrete Mathematics December 20, 2017

Project: Elliptic Curve Cryptography

Instructor: David Gu

Name: Rajesh Mohan
ID: 111498022

1 Introduction

Security plays a major role in today's exchange of information across the world. We see it everywhere, (albeit concealed) in our web browsers, smart devices and effectively the entire internet. Before the 1970s, the world relied on using a single shared key for encryption which is termed as symmetric encryption. A man in the middle can obtain hold of the key and the encrypted message and it was possible to decrypt the message using the key in his possession. In 1977, Ron Rivest, Adi Shamir and Leonard Adleman developed the RSA encryption system (named after their last names) that used the concept of a trapdoor function. The encryption used two random large prime numbers, multiplied them and resulted in a 256 bit number that was very difficult to split back to the original prime numbers. This in essence is a trapdoor function. It is also an asymmetric encryption as it uses two types of keys, private and public.

Come 2017, the 256-bit RSA key size is no longer secure due to the advancements in computing power over the years. A key size of 2048 or higher is required for a secure communication. This results in two problems- the size of a key affects the speed of transactions that occur in devices that have low computation power. The other problem is that the advancements in the coming years would again require a higher key size thereby looping back to the same two problems. This requires us to use an alternative solution that can have a smaller key size and maintain the same level of security.

Elliptic Curve Cryptography (ECC) enters the foray here. Although found in 1986 by two independent researchers Neal Koblitz and Victor S.Miller, ECC has been in use only since 2004. The popularity of RSA has negated the use for ECC but now it is coming into the lime-light because a 256 bit ECC key is equivalent to a 3072 bit RSA key. We can see the benefits already. But is it a good trapdoor function?

2 An Elliptic Curve

A general elliptic curve is defined by the following equation:

$$y^2 = x^3 + a.x + b \tag{2.1}$$

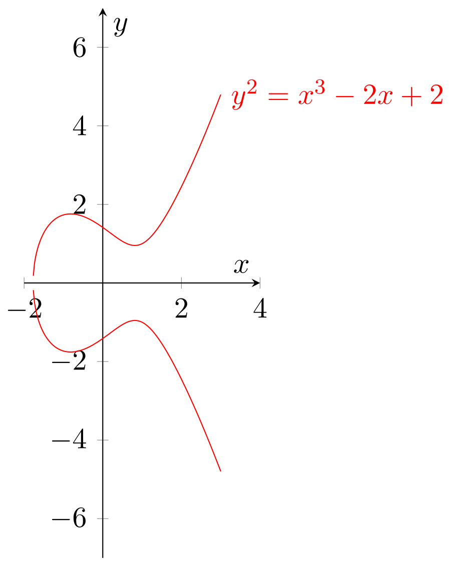

Figure 1 : An elliptic curve

For cryptographic purposes, we define the elliptic curve over a finite field especially over a prime field $\mathbb{Z}_p$ instead of over all real numbers. The elliptic curve over $\mathbb{Z}_p$, $p > 3$, is the set of all pairs $(x, y) \in \mathbb{Z}_p$ which fulfill

$$y^2 \equiv x^3 + a.x + b \mod p \tag{2.2}$$

together with an imaginary point of infinity ð, where

$$a, b \in \mathbb{Z} \text{ and } 4.a^3 + 27.b^2 \neq 0 \mod p$$

We can see that the curve is symmetric about the x-axis and has the roots as $\sqrt{{x_i}^3 + a.x_i + b}$ and $-\sqrt{{x_i}^3 + a.x_i + b}$ for a given y_i. Because the function values are modulo p, the curve will be bounded on the y-axis by p. Let us see what makes this a good trapdoor function for cryptography purposes. For this let us define the Point Addition and Point Doubling operations on the curve first.

2.1 Point Addition Operation

Given two points on the curve, say P and Q, we can find a third point by drawing a straight line through P and Q ($P \neq Q$) such that the line intersects the curve at a third point. Mirroring the obtained point about the x-axis gives us another point on the curve R. If we define this operation giving it a '+' notation, we get

$$P + Q = R \tag{2.3}$$

The figure shows the point addition operation for a curve over all real numbers.

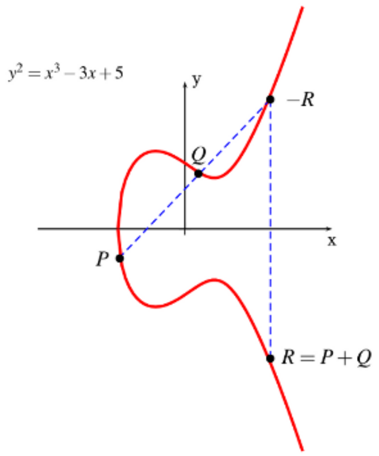

Figure 2: Point Addition

2.2 Point Doubling Operation

Given a point P on the curve, we can do the Point Addition operation on itself, effectively doing

$$R = P + P = 2P \tag{2.4}$$

This can be done by drawing a tangent to the curve at P and obtaining an intersection point and then mirroring it about the x-axis to obtain the point $2P$. This can be seen in the following figure. If we denote the coordinates of P, Q, R as $(x_1, y_1), (x_2, y_2), (x_3, y_3)$ respectively and the

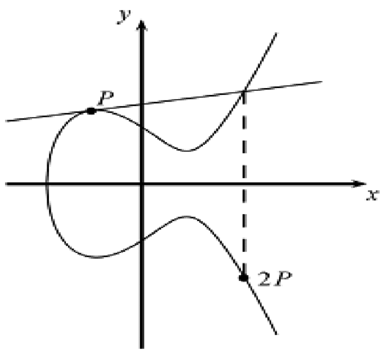

Figure 3: Point Doubling

slope as m, then

$$x_3 = m^2 - x_1 - x_2 \mod p \tag{2.5}$$
$$y_3 = m(x_1 - x_3) - y_1 \mod p \tag{2.6}$$

where

$$m = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} \mod p, & \text{if } P \neq Q \text{ (Point Addition)} \\ \frac{3x_1^2 + a}{2y_1} \mod p, & \text{if } P = Q \text{ (Point Doubling)} \end{cases} \tag{2.7}$$

The slope can be easily computed by taking the gradient of the curve at the point P in case of Point Doubling and it is the slope of the line through P and Q in the case of Point Addition.

Now, coming to the trapdoor feature of the curve, we define the infinite point $\eth$ on the curve as:

$$P + \eth = P \tag{2.8}$$

Essentially, when we continue this point addition operation from the initial point, also called the Generator point, we end up with $\eth$, the point at infinity. This is because from Eq 2.7, we can see that when the x coordinates of P and Q become equal, the slope becomes infinite thereby making x_3 and y_3 as infinite. Hence we define Eq 2.8 to make it circular so that when we add this infinite point to P, we get back P.

Example 1: If we define our elliptic curve E as follows:

$$E : y^2 \equiv x^3 + 2x + 2 \mod 17$$

If we set our generator point $P = (5, 1)$, then point doubling works as follows:

$$\begin{aligned}
&R = P + P = 2P = (x_3, y_3) \\
&(x_1, y_1) = (5, 1) \\
&(x_2, y_2) = (5, 1) \\
&m = \frac{3x_1^2 + a}{2y_1} = (2.1)^{-1}(3.5^2 + 2) \mod 17 \equiv 13 \mod 17 \\
&x_3 = m^2 + x_1 - x_2 = 13^2 - 5 - 5 = 159 \equiv 6 \mod 17 \\
&y_3 = m(x_1 - x_3) - y_1 = 13(5 - 6) - 1 \equiv 3 \mod 17 \\
&\therefore R = 2P = (6, 3)
\end{aligned}$$

Similarly, Point Addition of P and $2P$ gives $3P$ and so on. We keep performing Point Addition in this fashion until we get the original generator point P back. For the same example, if we calculate $18P$, we get $(5, 16)$ which upon adding with $(5, 1)$ results in $\eth$. (Note that the x-coordinates of the two points are the same).

We define the number of point additions required to get to $\eth$ as the number of points on the curve ($\#E$). In the example given above, $\#E = 19$. This number's bounds are defined by Hasse's theorem as follows:

$$p + 1 - 2\sqrt{p} \leq \#E \leq p + 1 + 2\sqrt{p} \tag{2.9}$$

3 The Trapdoor Feature

We have seen how to do a couple of operations on the elliptical curve. Having done that, we define the discrete logarithmic problem as follows:

Given an elliptic curve E, the generator point P and another point on the curve T, we need to find an integer d, where $1 \leq d \leq \#E$*, such that*

$$dP = P + P + ... + P \text{ (d-times)} = T \tag{2.10}$$

This d becomes the private-key for a person during exchange of communication. Why is this a good key? Because it is very difficult to determine the number of point additions required to get to T from P very similar to the prime factorization problem of RSA. The points P and T become the parameters of the public key. We need to choose a cryptographically secure elliptic curve to ensure that the attacker cannot solve the discrete logarithm problem easily. The standardized curves to use are specified by the National Institute of Standards and Technology (NIST) and they have recommended to use the following prime curves namely, p192, p224, p256, p384 and p521.

4 Elliptic Curve Diffie-Hellman Key Exchange (ECDH)

The Diffie-Hellman key exchange protocol uses the parameter of shared secret between two parties to generate a symmetric key to be exchanged between the two parties in a communication. Let us use the standard example of Alice and Bob to demonstrate the key sharing methodology.

As seen in Example 1, $\#E$ was 19 for the curve

$$E : y^2 \equiv x^3 + 2x + 2 \mod 17$$

The input parameters to ECDH are

$$\begin{aligned} a &= 2 \\ b &= 2 \\ p &= 17 \\ P &= (5, 1) \end{aligned}$$

For Alice, choose a number from 2 to $\#E - 1$, lets call it a_1. This becomes the private key for Alice, $k_{Pr(Alice)}$. Next, compute the public key $k_{Pub(Alice)}$ for Alice as

$$k_{Pub(Alice)} = a_1 P = A = (x_A, y_A)$$

Similarly assign a different number from 2 to $\#E - 1$ for Bob, lets call it b_1. This becomes the private key for Bob, $k_{Pr(Bob)}$. Next, compute the public key $k_{Pub(Bob)}$ for Bob as

$$k_{Pub(Bob)} = b_1 P = B = (x_B, y_B)$$

So now we have the following,

$$\begin{aligned} k_{Pr(Alice)} &= a_1 \\ k_{Pr(Bob)} &= b_1 \\ k_{Pub(Alice)} &= (x_A, y_A) \\ k_{Pub(Bob)} &= (x_B, y_B) \end{aligned}$$

Next, we can compute the shared secret for Alice and Bob as follows:

$$\begin{aligned} k_{AliceSS} &= k_{Pr(Alice)} * k_{Pub(Bob)} = a_1(b_1P) = a_1b_1P = (x_{AB}, y_{AB}) \\ k_{BobSS} &= k_{Pr(Bob)} * k_{Pub(Alice)} = b_1(a_1P) = b_1a_1P = (x_{AB}, y_{AB}) \end{aligned}$$

We can prove that $k_{AliceSS} = k_{BobSS}$ because the point multiplication operation is associative. We can remove y_{AB} and use only x_{AB} as the shared secret since the former can be computed using the curve E. This shared secret is then sent through a hash function to be used as a symmetric key for the session.

5 How secure is ECDH?

ECDH primarily relies on the curve being secure and as discussed before, we can use prime curves specified by the NIST. An attacker would only know the following information - the curve E, the modulo prime p, A, B, and the generator point P. The attacker will try to find the private keys of either Alice or Bob namely, a_1 and b_1 by trying to solve either of the following discrete logarithm problems:

$$a_1 = \log_p A$$

or

$$b_1 = \log_p B$$

The ECDH algorithm's security also relies on the size of the prime p used. In practice, at least a 256-bit prime must be used in order to guarantee that traditional crypto-breaking algorithms will not be able to tear down the security of the keys.

6 Conclusion

We have seen how Elliptic Curve Cryptography can be used to secure messages between two transacting users. The operations of Point Addition, Doubling and Multiplication particularly prove to be very useful in helping to setup the trapdoor feature of the curve. ECC has many advantages over RSA especially reduced key size given that the chosen curve is secure. Although not used by many in this day and age, ECC is being picked up slowly as a better alternative to RSA, especially in these ages where technology advances exponentially.

References

1. Understanding Cryptography: A Textbook for Students and Practitioners.
2. https://devcentral.f5.com/articles/real-cryptography-has-curves-making-the-case-for-ecc-20832

Appendix

1. ECDH Algorithm implemented in Python 3 (attached in the zip folder).