**Elliptic Curve Cryptography in Asymmetric Cryptography**

**Step 1:** Understanding the Elliptic Curve Equation

Elliptic curves used in cryptography are typically defined over a finite field (where ***p*** is a prime number). The general equation of an elliptic curve is:

***= + ax + b* mod *p***

* ***a*** and ***b*** are constants that define the specific curve,
* ***p*** is a prime number,
* ***x***, ***y*** are coordinates of points on the curve

Example of***= + ax + b*** graphed:

A graph of a function

AI-generated content may be incorrect.

**Step 2:** Point Addition on the Elliptic Curve

Given two points **P** (*x*1​, *y*1​) and **Q** (*x*2, *y*2) on the curve, we define the sum **R** (*x*3​, *y*3​)

**R** (*x*3​, *y*3​) = **P** + **Q**

**Note**: Point Addition doesn’t mean the addition of x and y coordinates of P and Q but merely the name of the given approach.

A graph of a line

AI-generated content may be incorrect.

**Case 1: Adding Two Distinct Points**

If **P** ≠ **Q**: the sum **R** = **P** + **Q** is found as follows:

1. Compute the slope:

​​ mod p

2. Compute the new coordinates:

mod *p*

mod *p*

**Example Calculation:**

*= +* 2*x +* 2 mod 17

Consider two points:

P (5,1) and Q (6,3)

1. Compute the slope:

mod 17 = mod 17 =

1. Compute *x*3:

mod 17

= 4 − 5 − 6 = −7

−7 mod 17 = 10

1. Compute *y*3​:

mod 17

= 2(-5) – 1 = -10 – 1 = -11

-11 mod 17 = 6

Thus, the sum of the two points is:

**P**+ **Q**= (10, 6)

**Case 2: Doubling a Point (P = Q)**

If **P** = **Q**, we use a different formula for the slope:

mod p

**Example Calculation (Doubling P (5, 1)):**

1. **Compute the slope:**

mod 17

= mod 17

= mod 17

= mod 17

Since mod17 (modular inverse of 2) is 9 (because 2×9 =18 ≡ 1 mod17),

mod 17

= 693 mod 17 = 12

1. **Compute x3:**

x3​ = - 5 - 5 mod 17

= 144 – 10 mod 17

= 134 mod 17 = 15

1. **Compute y3:**

y3 = 12(5 – 15) – 1 mod 17

= 12(-10) – 1 = -120 – 1 = -121

-121 mod 17 = 4

**Result:**

2P = (15, 4)

**Step 3:** Scalar Multiplication (Essential for Key Generation)

Scalar multiplication is the core operation in elliptic curve cryptography (ECC). It involves multiplying a point **P** by an integer ***k*** to get another point ***k*P**. This operation is the basis for generating public and private keys.

Scalar multiplication is achieved through **repeated point addition and doubling**:

*k*P = P + P + P + … (*k* times)

However, computing *k*P using simple addition is inefficient. Instead, we use the **double-and-add method**, which is similar to exponentiation by squaring.

**Double-and-Add Method**

To compute *k*P efficiently:

1. Convert k to binary.
2. Initialize **R** = **P** if the leftmost bit is 1, otherwise initialize **R** = **O** (the identity point).
3. Process each remaining bit:

* **Double** **R** (i.e., compute **2R**).
* If the bit is 1, **add** **P** to **R**.

1. Repeat until all bits of **k** are processed.

**Example Calculation:** Compute 7**P** for **P** = (5,1) **on** *= + 2x + 2 mod 17*

**Step 1:** Convert ***k*** = 7 to **Binary**

7 = 111

**Step 2:** Compute Using **Double**-and-**Add**

Since 7 = , we compute:

1. **2P** (**Point Doubling** of **P** (5, 1))

We already computed this in the previous step:

2P = (15, 4)

1. **4P** = 2(**2P**) (**Double 2P**):

Using the doubling formula for **P** = (15, 4):

mod 17

mod 17

mod 17

Finding mod 17, we get = 15, so:

mod 17

= 10155 mod 17 = 13

**Now,**

x3 = – 15 – 15 mod 17

= 169 – 30 mod 17

= 139 mod 17 = 3

y3 ​= 13 (15 − 3) – 4 mod 17

= 13 (12) – 4 mod 17

= 156 − 4 = 152 mod 17 = 6

**So,**

**4P** = (3, 6)

7P = 4P + 2P + P (Add 4P, 2P, P):

* 4P = (3, 6)
* 2P = (15, 4)
* P = (5, 1)

Compute 4P+2P:

* Use point addition formula with (3,6) and (15,4):

mod 17

mod 17

Finding mod 17, we get = 10, so:

mod 17

mod 17 = 14

mod 17

mod 17 = 8

y3 = 14(3 – 8) – 6 mod 17

= 14(-5) - 6 = -70 – 6 = -76 mod 17 = 13

**6P** = (8, 13)

Now compute **6P** + **P** = (8, 13) + (5,1):

mod17

mod 17

Since mod 17 = 6,

mod 17 = -72 mod 17 = 14

mod 17

mod17 = 13

mod 17

mod 17 = 7

Final Result:

**7P** = (13, 7)

Step 4: Key Generation in Elliptic Curve Cryptography (ECC)

Now that we understand scalar multiplication, we can generate **private** and **public keys** using an elliptic curve.

1. **Choosing Domain Parameters**

In ECC, the parameters of the curve are publicly known and include:

* A **prime number** *p*
* The **curve equation *= + ax + b* mod *p***
* A **base point** **G (x, y)** (a fixed point on the curve with high order).
* The **order** of **G** (the smallest integer ***n*** such that ***n*G** = **O**, the identity point).

For example, let’s continue using:

*= +* 2*x +* 2 mod 17

With a base point:

G = (5, 1)

and assume its order is **19** (meaning **19G** = **O**).

**CONTINUE TO NEXT PAGE**

**2. Selecting a Private Key**

A private key is a randomly chosen integer ***d*** from the range:

1 ***d*** ***n*** -1

Let’s choose:

***d*** = 7

**3. Computing the Public Key**

The public key **Q** is computed as:

**Q** = ***d*G**

We already computed 7G in the previous step:

**7G =** (13, 7)

So, the **public key is**:

**Q** = (13,7)

**Summary**

* The **private key** is ***d*** = 7
* The **public key** is **Q** = ***d*G** = (13, 7)
* The public key can be **shared**, while the private key must be **kept secret**.

Step 5: ECC Encryption and Decryption

Now that we have the key pair, we can use it for encryption and decryption in ECC. The ECC encryption scheme is based on the **Elliptic Curve ElGamal Encryption** method.

**Encryption Process**

To encrypt a message M, we represent it as a point P on the elliptic curve. The encryption process involves the following steps:

1. **Public Information**: The sender has access to:

* The recipient’s **public key** Q = dG
* The chosen **elliptic curve** and **base point** G

1. **Sender Chooses a Random Integer**

* **k** is a randomly selected integer from to n − 1.
* This ensures encryption is **unique each time**, even for the same message.

1. **Compute Two Points for Encryption:**
   * Compute the **ephemeral key**:

C1​=kG

* Compute the **masked message point**:

C2​=PM​+kQ

1. **Send the Ciphertext:**

* The encrypted message consists of the **two points (C1​, C2​).**

**Example: Encrypting a Message**

Let’s say our **message** is represented by the point:

PM​ = (6, 3)

We use the previously computed **public key**:

Q = (13, 7)

Step 1: Choose a Random k

Let’s choose:

k = 3

Step 2: Compute C1 ​

C1​ = kG = 3G

We compute 3G using the double-and-add method:

* 2G = (15,4) (from previous steps)
* 3G = 2G + G = (10,6) (using point addition)

So,

C1 ​= (10, 6)

Step 3: Compute C2

C2 ​= PM​ + kQ = (6, 3) + 3 (13, 7)

First, compute 3Q:

* 2Q = (8, 13) (from previous steps)
* 3Q = 2Q + Q = (3, 6)

So,

kQ = (3, 6)

Now,

C2 = (6, 3) + (3, 6)

Using point addition, we compute:

C2​ = (9, 16)

Thus, the **ciphertext** is:

(C1​, C2​) = ((10,6), (9,16))

**Decryption Process**

To decrypt, the recipient (who knows the private key ***d***) computes:

1. Compute dC1

Since C1 = kG, we compute:

dC1 ​= d(kG) = k(dG) = kQ

Using d=7 and C1 = (10,6), we know:

dC1​ = kQ = (3, 6)

1. Recover the Original Message

PM​ = C2 ​− dC1​

PM​ = (9,16) − (3,6)

Using point subtraction:

PM ​= (6, 3)

This matches our original message point, so **decryption is successful!**