Chapter 3: Elliptic Curve Cryptography (ECC)

3.1 Introduction

An **elliptic curve** is defined by an equation in two variables with coefficients. For cryptography, the variables and coefficients are restricted to elements in a **finite field**, which results in the definition of a **finite abelian group**. The principal attraction of ECC, compared to RSA, is that it appears to offer equal security for a far smaller key size, thereby reducing processing overhead.

3.2 Elliptic Curves over Real Numbers

Elliptic curves are not ellipses. They are so named because they are described by cubic equations. In general, cubic equations for elliptic curves take the following form, known as a **Weierstrass equation**:

$$y^2 + axy + by = x^3 + cx^2 + dx + e$$

where a, b, c, d, e are real numbers and x and y take on values in the real numbers. For our purpose, it is sufficient to limit ourselves to equations of the form

$$y^2 = x^3 + ax + b \qquad (3.1)$$

Such equations are said to be cubic, or of degree 3, because the highest exponent they contain is a 3. Also included in the definition of an elliptic curve is a single element denoted *O* and called the point at infinity or the zero point. To plot such a curve, we need to compute

$$y = \sqrt{x^3 + ax + b}$$

For given values of a and b, the plot consists of positive and negative values of y for each value of x. Thus, each curve is symmetric about y = 0 as shown in Figure 3.1.

Now, consider the set of points E(a, b) consisting of all of the points (x, y) that satisfy Equation (3.1) together with the element *O*. Using a different value of the pair (a, b) results in a different set E(a, b). Using this terminology, the two curves in Figure 3.1 depict the sets E(-1, 0) and E(1, 1), respectively.

3.2.1 GEOMETRIC DESCRIPTION OF ADDITION

It can be shown that a group can be defined based on the set E(a, b) for specific values of a and b in Equation (3.1), provided the following condition is met:

$$4a^3 + 27b^2 \neq 0 \qquad (3.2)$$

To define the group, we must define an operation, called addition and denoted by +, for the set E(a, b), where a and b satisfy Equation (3.2). In geometric terms, the rules for addition can be stated as follows: If three points on an elliptic curve lie on a straight line, their sum is *O*. From this definition, we can define the rules of addition over an elliptic curve.

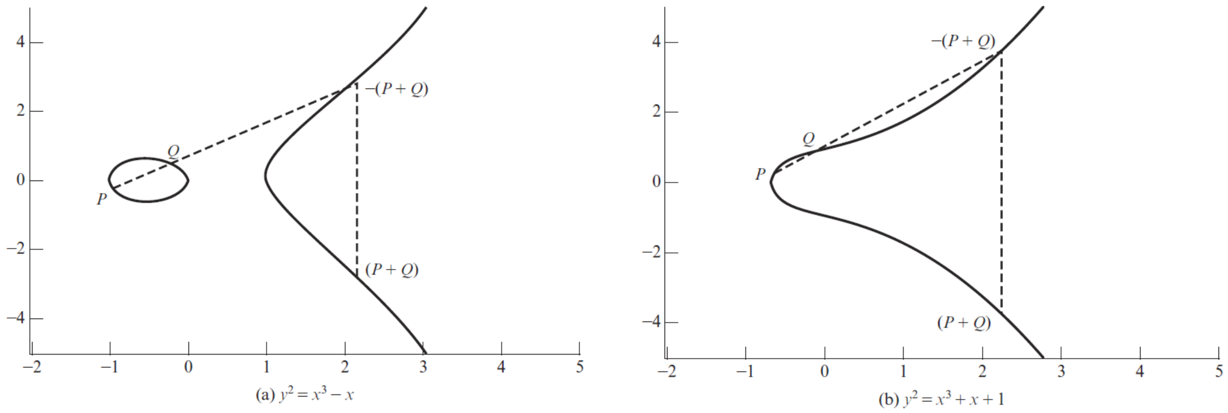


Figure 3.1 Example of Elliptic Curves

1. *O* serves as the additive identity. Thus *O* = -*O*; for any point *P* on the elliptic curve, *P* + *O* = *P*. In what follows, we assume *P* ≠ *O* and *Q* ≠ *O*.
2. The negative of a point *P* is the point with the same *x* coordinate but the negative of the *y* coordinate; that is, if *P* = *(x, y)*, then -*P* = *(x, -y)*. Note that these two points can be joined by a vertical line. Note that *P* + *(-P)* = *P* - *P* = *O*.
3. To add two points *P* and *Q* with different *x* coordinates, draw a straight line between them and find the third point of intersection *R*. It is easily seen that there is a unique point *R* that is the point of intersection (unless the line is tangent to the curve at either *P* or *Q*, in which case we take *R* = *P* or *R* = *Q*, respectively). To form a group structure, we need to define addition on these three points: *P* + *Q* = -*R*. That is, we define *P* + *Q* to be the mirror image (with respect to the *x* axis) of the third point of intersection. Figure 3.1 illustrates this construction.
4. The geometric interpretation of the preceding item also applies to two points, *P* and -*P*, with the same *x* coordinate. The points are joined by a vertical line, which can be viewed as also intersecting the curve at the infinity point. We therefore have *P* + *(-P)* = *O*, which is consistent with item (2).
5. To double a point *Q*, draw the tangent line and find the other point of intersection *S*. Then *Q* + *Q* = *2Q* = -*S*.

3.2.2 ALGEBRAIC DESCRIPTION OF ADDITION

In this subsection, we present some results that enable calculation of additions over elliptic curves. For two distinct points, $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$, that are not negatives of each other, the slope of the line *l* that joins them is $\Delta = (y_Q - y_P)/(x_Q - x_P)$. There is exactly one other point where *l* intersects the elliptic curve, and that is the negative of the sum of P and Q. After some algebraic manipulation, we can express the sum *R* = *P* + *Q* as

$$x_R = \Delta^2 - x_P - x_Q$$

$$y_R = -y_P + \Delta(x_P - x_R) \qquad (3.3)$$

We also need to be able to add a point to itself: *P* + *P* = *2P* = *R*. When $y_P \neq 0$, the expressions are

$$x_R = \left(\frac{3x_P^2 + a}{2y_P}\right)^2 - 2x_P$$

$$y_R = \left(\frac{3x_P^2 + a}{2y_P}\right)(x_P - x_R) - y_P \qquad 3.4$$

3.3 Elliptic Curves over Zp

For a prime curve over Zp, we use a cubic equation in which the variables and coefficients all take on values in the set of integers from 0 through p - 1 and in which calculations are performed modulo p. prime curves are best for software applications, because the extended bit-fiddling operations needed by binary curves are not required;

For elliptic curves over Zp, as with real numbers, we limit ourselves to equations of the form of Equation (3.1), but in this case with coefficients and variables limited to Zp:

$$y^2 \bmod p = (x^3 + ax + b) \bmod p \qquad (3.5)$$

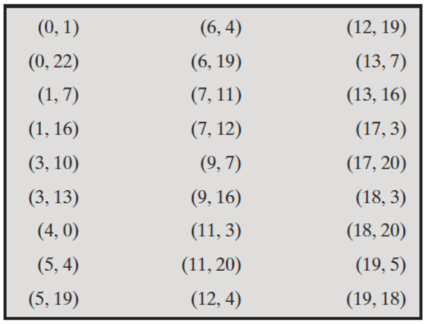
Now consider the set Ep(a, b) consisting of all pairs of integers (x, y) that satisfy Equation (3.5), together with a point at infinity *O*. The coefficients a and b and the variables x and y are all elements of Zp.

For example, let p = 23 and consider the elliptic curve $y^2 = x^3 + x + 1$. In this case, a = b = 1. The figure 3.1b shows a continuous curve with all of the real points that satisfy the equation. For the set $E_{23}(1, 1)$, we are only interested in the nonnegative integers in the quadrant from (0, 0) through (p - 1, p - 1) that satisfy the equation mod p. Table 3.1 lists the points (other than *O*) that are part of $E_{23}(1, 1)$. Note that the points, with one exception, are symmetric about y = 11.5.

It can be shown that a finite abelian group can be defined based on the set Ep(a, b) provided that $(x^3 + ax + b)$ mod p has no repeated factors. This is equivalent to the condition

$$(4a^3 + 27b^2) \bmod p \neq 0 \bmod p \qquad (3.2)$$

Table 3.1: Points (other than *O*) on the Elliptic Curve $E_{23}(1, 1)$



(0, 1)	(6, 4)	(12, 19)
(0, 22)	(6, 19)	(13, 7)
(1, 7)	(7, 11)	(13, 16)
(1, 16)	(7, 12)	(17, 3)
(3, 10)	(9, 7)	(17, 20)
(3, 13)	(9, 16)	(18, 3)
(4, 0)	(11, 3)	(18, 20)
(5, 4)	(11, 20)	(19, 5)
(5, 19)	(12, 4)	(19, 18)

The rules for addition over Ep(a, b), correspond to the algebraic technique described for elliptic curves defined over real numbers. For all points *P, Q* ∈ Ep(a, b):

1. *P + O = P*.
2. If *P = (x_P, y_P)*, then *P + (x_P, -y_P) = O*. The point *(x_P, -y_P)* is the negative of *P*, denoted as -*P*. For example, in $E_{23}(1, 1)$, for *P = (13, 7)*, we have -*P = (13, -7)*. But -7 mod 23 = 16. Therefore, -*P = (13, 16)*, which is also in $E_{23}(1, 1)$.
3. If *P = (x_p, y_p)* and *Q = (x_Q, y_Q)* with *P ≠ -Q*, then *R = P + Q = (x_R, y_R)* is determined by the following rules:

$$x_R = (\lambda^2 - x_P - x_Q)\ mod\ p$$

$$y_R = (\lambda(x_P - x_R) - y_P)\ mod\ p$$

where

$$\lambda = \begin{cases} \left(\dfrac{y_Q - y_P}{x_Q - x_P}\right) mod\ p & if\ P \neq Q \\ \left(\dfrac{3x_P^2 + a}{2y_P}\right) mod\ p & if\ P = Q \end{cases}$$

4. Multiplication is defined as repeated addition; for example, *4P = P + P + P + P*.

For example, let *P = (3, 10)* and *Q = (9, 7)* in $E_{23}(1, 1)$. Then

$$\lambda = \left(\frac{7-10}{9-3}\right) mod\ 23 = \left(\frac{-3}{6}\right) mod\ 23 = \left(\frac{-1}{2}\right) mod\ 23 = 11$$

$$x_R = (11^2 - 3 - 9)\ mod\ 23 = 109\ mod\ 23 = 17$$

$$y_R = (11(3 - 17) - 10)\ mod\ 23 = -164\ mod\ 23 = 20$$

So *P + Q = (17, 20)*. To find *2P*

$$\lambda = \left(\frac{3(3^2)+1}{2 \times 10}\right) mod\ 23 = \left(\frac{5}{20}\right) mod\ 23 = \left(\frac{1}{4}\right) mod\ 23 = 6$$

The last step in the preceding equation involves taking the multiplicative inverse of 4 in Z_{23}. This can be done using the extended Euclidean algorithm. To confirm, note that (6 * 4) mod 23 = 24 mod 23 = 1

$$x_R = (6^2 - 3 - 3)\ mod\ 23 = 30\ mod\ 23 = 7$$

$$y_R = (6(3 - 7) - 10)\ mod\ 23 = -34\ mod\ 23 = 12$$

and *2P = (7, 12).*

3.4 Elliptic Curves over GF(2^m)

For a binary curve defined over GF(2^m), the variables and coefficients all take on values in GF(2^m) and in calculations are performed over GF(2^m). Binary curves are best for hardware applications, where it takes remarkably few logic gates to create a powerful, fast cryptosystem.

3.5 ELLIPTIC CURVE CRYPTOGRAPHY

Consider the equation Q = kP where Q, P ∈ E_P(a, b) and k < p. It is relatively easy to calculate Q given k and P, but it is hard to determine k given Q and P. This is called the discrete logarithm problem for elliptic curves.

We give an example taken from the Certicom Web site (www.certicom.com). Consider the group E_{23}(9,17). This is the group defined by the equation y^2 mod 23 = (x^3 + 9x + 17) mod 23. What is the discrete logarithm k of Q = (4, 5) to the base P = (16, 5)? The brute-force method is to compute multiples of P until Q is found. Thus,

$$P = (16,5);\ 2P = (20, 20);\ 3P = (14, 14);\ 4P = (19, 20);\ 5P = (13, 10);$$

$$6P = (7, 3);\ 7P = (8, 7);\ 8P = (12, 17);\ 9P = (4, 5)$$

Because 9P = (4, 5) = Q, the discrete logarithm Q = (4, 5) to the base P = (16, 5) is k = 9. In a real application, k would be so large as to make the bruteforce approach infeasible.

3.5.1 Analog of Diffie–Hellman Key Exchange

Key exchange using elliptic curves can be done in the following manner. First pick a large integer q, which is a prime number p, and elliptic curve parameters a and b for Equation (3.5). This defines the elliptic group of points E_q(a, b). Next, pick a base point G = (x_1, y_1) in E_p(a, b) whose order is a very large value n. The order n of a point G on an elliptic curve is the smallest positive integer n such that nG = 0 and G are parameters of the cryptosystem known to all participants. A key exchange between users A and B can be accomplished as follows

1. A selects an integer n_A less than n. This is A's private key. A then generates a public key P_A = n_A * G; the public key is a point in E_q(a, b).

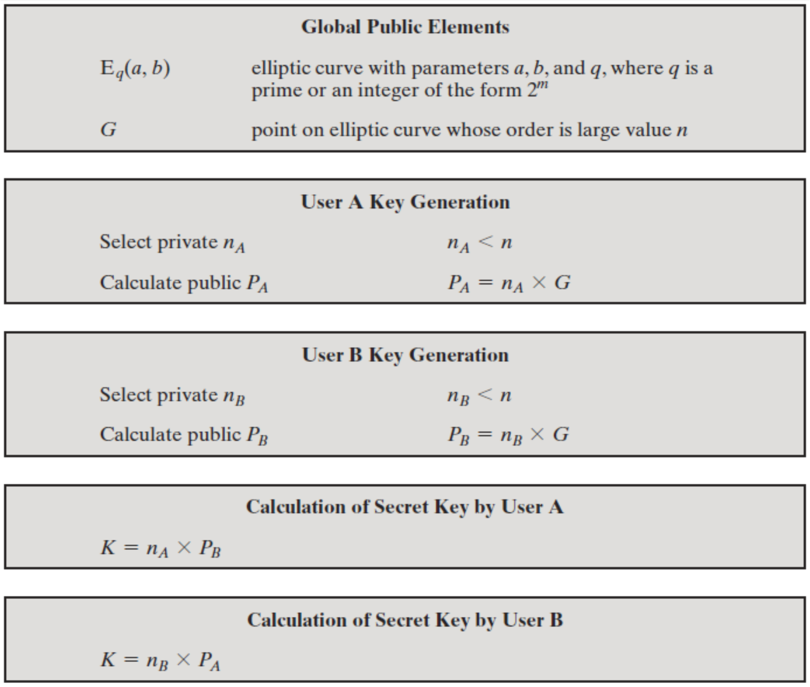
2. B similarly selects a private key n_B and computes a public key P_B.

3. A generates the secret key k = n_A * P_B. B generates the secret key k = n_B * P_A.

The two calculations in step 3 produce the same result because

$$n_A * P_B = n_A * (n_B * G) = n_B * (n_A * G) = n_B * PA$$

To break this scheme, an attacker would need to be able to compute k given G and kG, which is assumed to be hard.



Global Public Elements

$E_q(a, b)$	elliptic curve with parameters a, b, and q, where q is a prime or an integer of the form 2^m
G	point on elliptic curve whose order is large value n

User A Key Generation

Select private n_A	$n_A < n$
Calculate public P_A	$P_A = n_A \times G$

User B Key Generation

Select private n_B	$n_B < n$
Calculate public P_B	$P_B = n_B \times G$

Calculation of Secret Key by User A

$K = n_A \times P_B$

Calculation of Secret Key by User B

$K = n_B \times P_A$

As an example, take p = 211; $E_p(0, -4)$, which is equivalent to the curve $y^2 = x^3 - 4$; and G = (2, 2). One can calculate that 240G = *O*. A’s private key is n_A = 121, so A’s public key is P_A = 121(2, 2) = (115, 48). B’s private key is n_B = 203, so B’s public key is 203(2, 3) = (130, 203). The shared secret key is 121(130, 203) = 203(115, 48) = (161, 69).

Note that the secret key is a pair of numbers. If this key is to be used as a session key for conventional encryption, then a single number must be generated. We could simply use the x coordinates or some simple function of the x coordinate.

3.5.2 Elliptic Curve Encryption/Decryption

The first task in this system is to encode the plaintext message m to be sent as an (x, y) point P_m. It is the point P_m that will be encrypted as a ciphertext and subsequently decrypted. Note that we cannot simply encode the message as the x or y coordinate of a point, because not all such coordinates are in $E_q(a, b)$; for example, see Table 3.1. Again, there are several approaches to this encoding, which we will not address here, but suffice it to say that there are relatively straightforward techniques that can be used.

As with the key exchange system, an encryption/decryption system requires a point G and an elliptic group $E_q(a, b)$ as parameters. Each user A selects a private key n_A and generates a public key $P_A = n_A * G$.

To encrypt and send a message P_m to B, A chooses a random positive integer k and produces the ciphertext C_m consisting of the pair of points:

$$C_m = \{kG, P_m + kP_B\}$$

Note that A has used B’s public key P_B. To decrypt the ciphertext, B multiplies the first point in the pair by B’s private key and subtracts the result from the second point:

$P_m + kP_B - n_B(kG) = P_m + k(n_BG) - n_B(kG) = P_m$

A has masked the message P_m by adding kP_B to it. Nobody but A knows the value of k, so even though P_b is a public key, nobody can remove the mask kP_B. However, A also includes a “clue,” which is enough to remove the mask if one knows the private key n_B. For an attacker to recover the message, the attacker would have to compute k given G and kG, which is assumed to be hard.

Let us consider a simple example. The global public elements are q = 257; $E_q(a, b) = E_{257}(0, -4)$, which is equivalent to the curve $y^2 = x^3 - 4$; and G =(2, 2). Bob’s private key is $n_B = 101$, and his public key is $P_B = n_BG = 101(2, 2) =(197, 167)$. Alice wishes to send a message to Bob that is encoded in the elliptic point $P_m = (112, 26)$. Alice chooses random integer k = 41 and computes kG =41(2, 2) = (136, 128), $kP_B = 41(197, 167) = (68, 84)$ and $P_m + kP_B = (112, 26)+ (68, 84) = (246, 174)$. Alice sends the ciphertext $C_m = (C1, C2) = \{(136, 128), (246, 174)\}$ to Bob. Bob receives the ciphertext and computes $C_2 - n_BC_1 = (246, 174) - 101(136, 128) = (246, 174) - (68, 84) = (112, 26)$.

3.5.3 ELLIPTIC CURVE DIGITAL SIGNATURE ALGORITHM

First we give a brief overview of the process involved in ECDSA. In essence, four elements are involved.

1. All those participating in the digital signature scheme use the same global domain parameters, which define an elliptic curve and a point of origin on the curve.

2. A signer must first generate a public, private key pair. For the private key, the signer selects a random or pseudorandom number. Using that random number and the point of origin, the signer computes another point on the elliptic curve. This is the signer’s public key.

3. A hash value is generated for the message to be signed. Using the private key, the domain parameters, and the hash value, a signature is generated. The signature consists of two integers, r and s.

4. To verify the signature, the verifier uses as input the signer’s public key, the domain parameters, and the integer s. The output is a value v that is compared to r. The signature is verified if v = r.

Let us examine each of these four elements in turn.

Global Domain Parameters

The global domain parameters for ECDSA are the following:

q a prime number

a, b integers that specify the elliptic curve equation defined over Z_q with the equation $y^2 = x^3 + ax + b$

G a base point represented by $G = (x_g, y_g)$ on the elliptic curve equation

n order of point G; that is, n is the smallest positive integer such that $nG = O$. This is also the number of points on the curve.

Key Generation

Each signer must generate a pair of keys, one private and one public. The signer, let us call him Bob, generates the two keys using the following steps:

1. Select a random integer d, $d \in [1, n - 1]$
2. Compute $Q = dG$. This is a point in $E_q(a, b)$
3. Bob's public key is Q and private key is d.

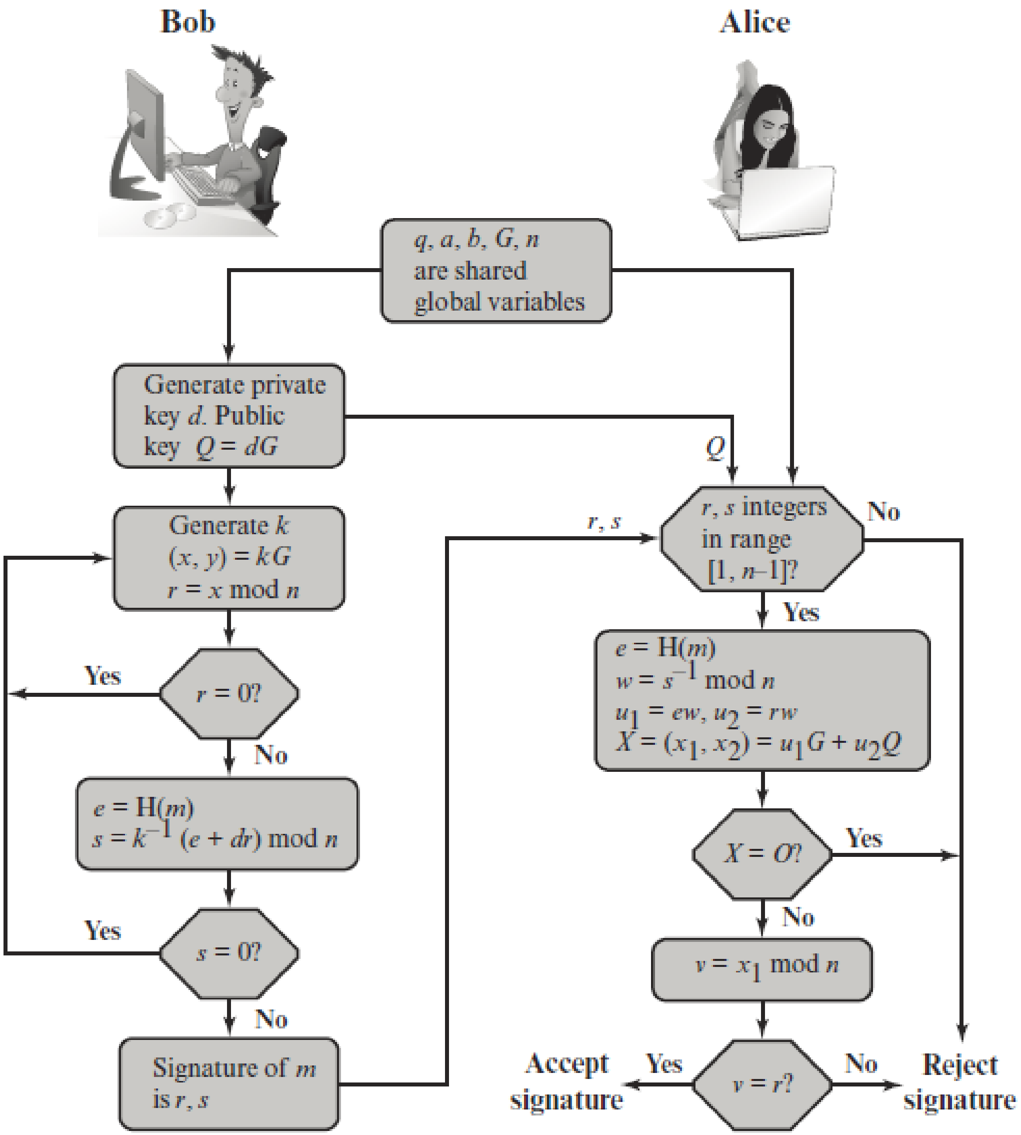
Digital Signature Generation and Authentication

With the public domain parameters and a private key in hand, Bob generates a digital signature of 320 bytes for message m using the following steps:

1. Select a random or pseudorandom integer k, $k \in [1, n - 1]$
2. Compute point $P = (x, y) = kG$ and $r = x \bmod n$. If $r = 0$ then goto step 1
3. Compute $t = k^{-1} \bmod n$
4. Compute $e = H(m)$, where H is one of the SHA-2 or SHA-3 hash functions.
5. Compute $s = k^{-1}(e + dr) \bmod n$. If $s = O$ then goto step 1
6. The signature of message m is the pair (r, s).

Alice knows the public domain parameters and Bob's public key. Alice is presented with Bob's message and digital signature and verifies the signature using the following steps:

1. Verify that r and s are integers in the range 1 through n - 1
2. Using SHA, compute the 160-bit hash value $e = H(m)$
3. Compute $w = s^{-1} \bmod n$
4. Compute $u_1 = ew$ and $u_2 = rw$
5. Compute the point $X = (x_1, y_1) = u_1G + u_2Q$
6. If $X = O$, reject the signature else compute $v = x_1 \bmod n$
7. Accept Bob's signature if and only if $v = r$


Bob
Alice
q, a, b, G, n are shared global variables
Generate private key d. Public key $Q = dG$
Q
Generate k
$(x, y) = kG$
$r = x \bmod n$
$r = 0$?
Yes
No
$e = H(m)$
$s = k^{-1}(e + dr) \bmod n$
$s = 0$?
Yes
No
Signature of m is r, s
r, s
r, s integers in range $[1, n-1]$?
No
Yes
$e = H(m)$
$w = s^{-1} \bmod n$
$u_1 = ew$, $u_2 = rw$
$X = (x_1, x_2) = u_1G + u_2Q$
$X = O$?
Yes
No
$v = x_1 \bmod n$
$v = r$?
Accept signature
Yes
No
Reject signature