Outline

- Merkle Tree
- Elliptic Curve Cryptography (ECC)

Public key cryptography

- Recall
  - RSA algorithm
  - Diffie-Hellman key exchange algorithm

- What you need for a public key cryptographic system to work is a set of algorithms that is easy to process in one direction, but difficult to undo.

- These algorithms serve as **trap door** functions

- Finding a good Trapdoor Function is critical for a secure public key cryptographic system.

Motivation

- **Problem:**
  Asymmetric schemes like RSA and El Gamal require exponentiations in integer rings and fields with parameters of more than 1,000 bits.
  - High computational effort on CPUs with 32-bit or 64-bit arithmetic
  - Large parameter sizes critical for storage on small and embedded device
- **Motivation:**
  Smaller field sizes providing equivalent security are desirable
- **Solution:**
  Elliptic Curve Cryptography (ECC) uses a group of points (instead of integers) for cryptographic schemes with coefficient sizes of 160-256 bits, reducing significantly the computational effort.

What is an Elliptic Curve?

- An ***Elliptic Curve*** *E* is a curve given by an equation

$E : y^2 = f(x)$,

where f(x) is a square-free (no double roots) cubic (x^3) or a quartic polynomial (x^4).

After a change of variables, it takes a simpler form:

$$E : y^2 = x^3 + Ax + B$$

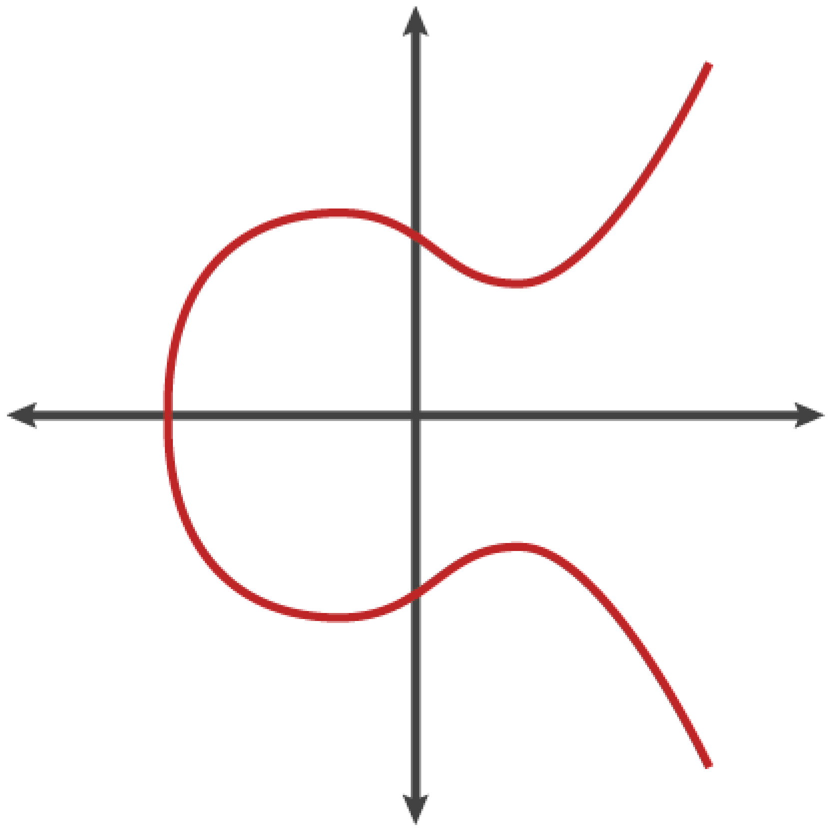
So, $y^2 = x^3$ is not an elliptic curve but $y^2 = x^3-1$ is

What exactly is an elliptic curve?

- Let $A \in \mathbb{R}$, $B \in \mathbb{R}$, be constants such that $4A^3 + 27B^2 \neq 0$. A *non-singular elliptic curve* is the set *E* of points $(x, y) \in \mathbb{R} \times \mathbb{R}$ described by the equation:

$$y^2 = x^3 + Ax + B$$

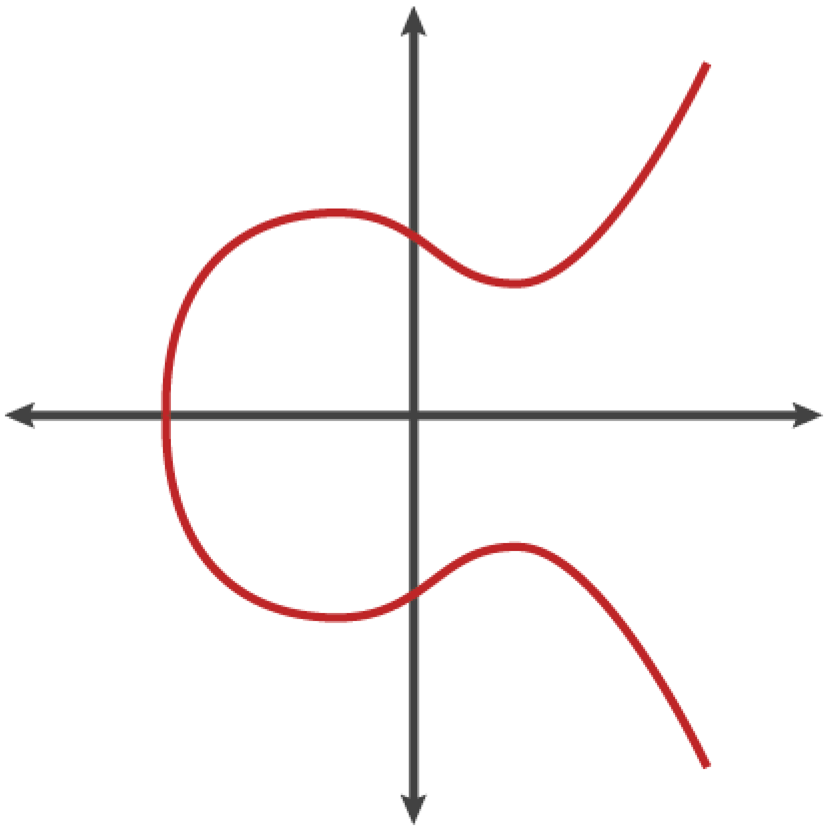
together with a special point *O* called *the point at infinity*.



Computations on Elliptic Curves

- In cryptography, we are interested in elliptic curves module a prime p:

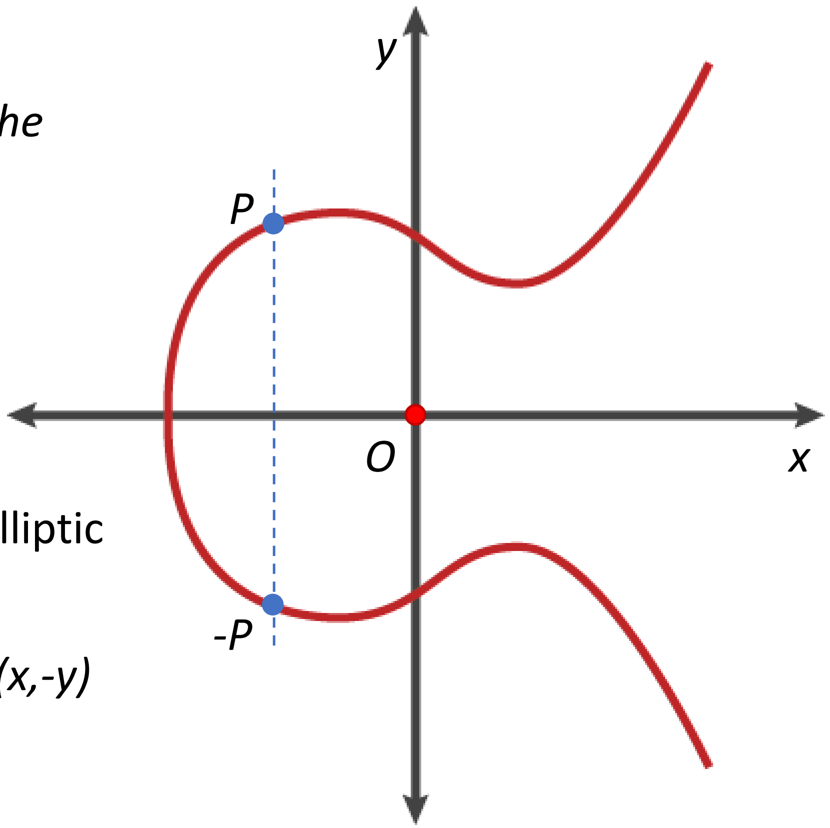
> ***Definition: Elliptic Curves over prime fields***
>
> *The elliptic curve over Z_p, $p>3$ is the set of all pairs $(x,y) \in Z_p$ which fulfill*
>
> $$y^2 = x^3 + Ax + B \bmod p$$
>
> *together with an imaginary point of infinity O, where $A,B \in Z_p$ fulfill the condition*
>
> $$4A^3+27B^2 \neq 0 \bmod p.$$



- Note that $Z_p = \{0,1,\ldots, p -1\}$ is a set of integers with modulo p arithmetic

Computations on Elliptic Curves

- Some special considerations are required to convert elliptic curves into a group of points
  - *In any group, a special element is required to allow for the identity operation, i.e., given $P \in E$: $P + O = P = O + P$*
  - *This identity point (which is not on the curve) is additionally added to the group definition*
  - *This (infinite) identity point is denoted by O*

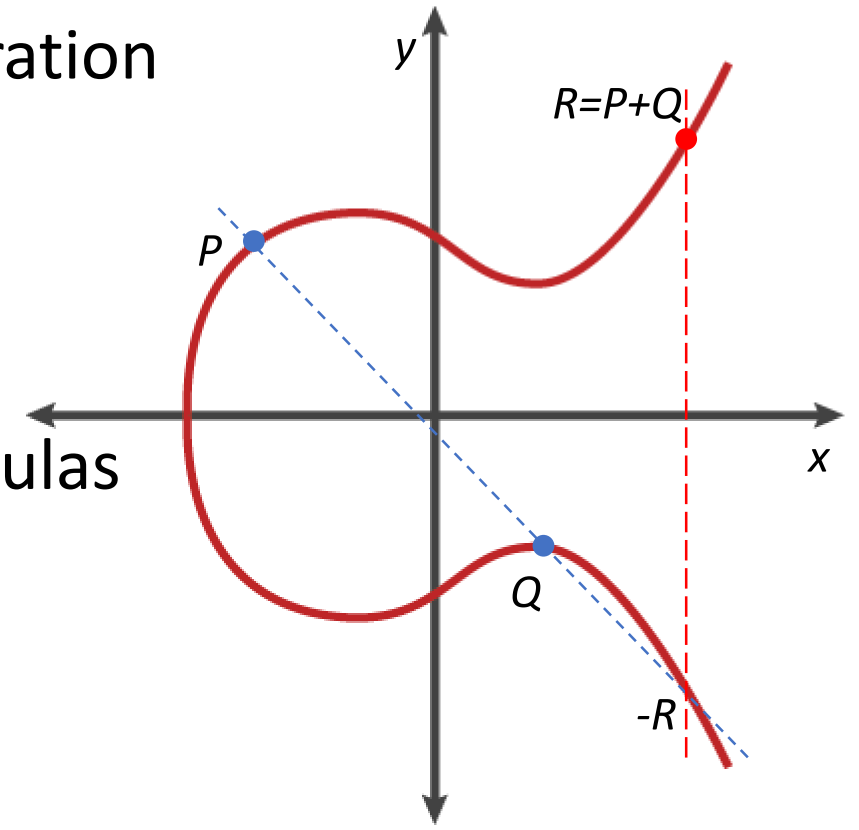


- Elliptic Curve are symmetric along the *x*-axis
  - Up to two solutions *y and -y* exist for each quadratic residue *x* of the elliptic curve
  - For each point *P =(x,y),* the inverse or negative point is defined as *-P =(x,-y)*

Computations on Elliptic Curves

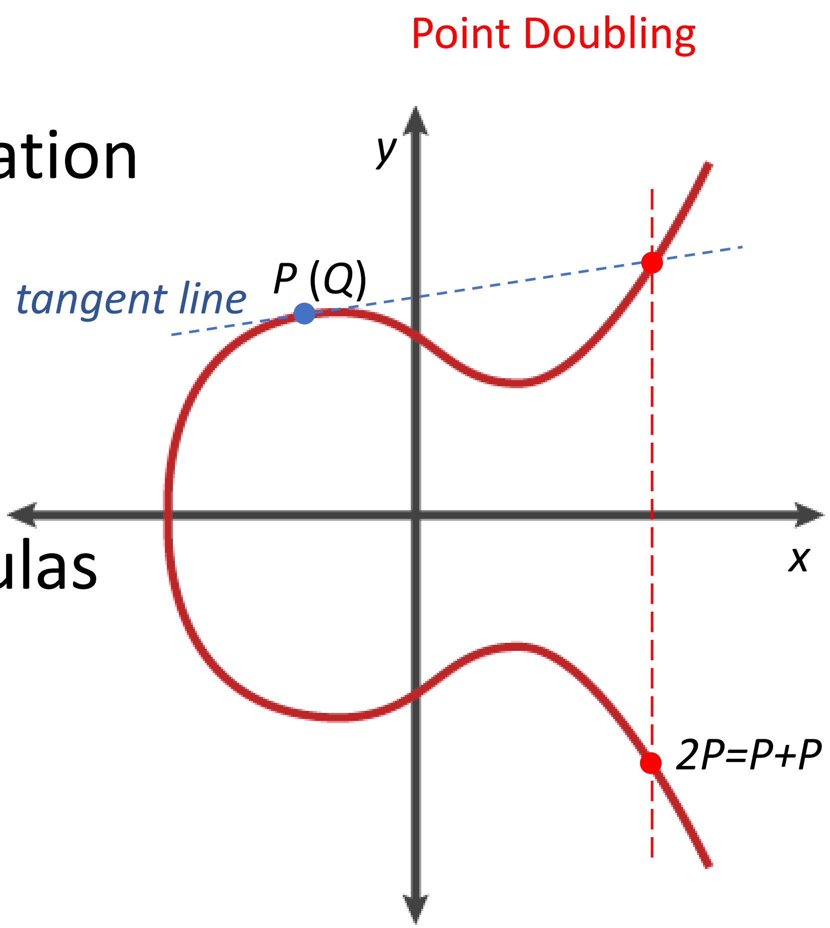
- Generating a *group of points* on elliptic curves based on point addition operation *P+Q = R, i.e.,* $(x_P, y_P)+(x_Q, y_Q) = (x_R, y_R)$
- Geometric Interpretation of point addition operation
  - *Draw straight line through P and Q; if P=Q use tangent line instead*
  - *Mirror third intersection point of drawn line with the elliptic curve along the x-axis*
- Elliptic Curve Point Addition and Doubling Formulas

Point Addition



Computations on Elliptic Curves

- Generating a *group of points* on elliptic curves based on point addition operation *P+Q = R, i.e.,* $(x_P,y_P)+(x_Q,y_Q) = (x_R,y_R)$
- Geometric Interpretation of point addition operation
  - *Draw straight line through P and Q; if P=Q use tangent line instead*
  - *Mirror third intersection point of drawn line with the elliptic curve along the x-axis*
- Elliptic Curve Point Addition and Doubling Formulas



Computations on Elliptic Curves

- Elliptic Curve Point Addition and Doubling Formulas

$x_3 = s^2 - x_1 - x_2 \bmod p$ and $y_3 = s(x_1 - x_3) - y_1 \bmod p$

where

$$s = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1} \bmod p & \text{; if P} \neq \text{Q (point addition)} \\ \dfrac{3x_1^2 + a}{2y_1} \bmod p & \text{; if P = Q (point doubling)} \end{cases}$$

In the special case where m is a prime, $\phi(m) = m - 1$ and a modular inverse is given by

$$a^{-1} \equiv a^{m-2} \pmod{m}.$$

[2]

This method is generally slower than the extended Euclidean algorithm,

Example: Given $E: y^2 = x^3+2x+2 \bmod 17$ and point $P=(5,1)$

Goal: Compute $2P = P+P = (5,1)+(5,1)= (x_3,y_3)$

$s = \dfrac{3x_1^2 + a}{2y_1} \pmod{17} = (2 \cdot 1)^{-1}(3 \cdot 5^2 + 2) \pmod{17} = 2^{-1} \cdot 9 \equiv 2^{17-2}\, 9 \pmod{17} \equiv 9 \cdot 9 \equiv 13 \pmod{17}$

$x_3 = s^2 - x_1 - x_2 = 13^2 - 5 - 5 = 159 \equiv 6 \pmod{17}$

$y_3 = s(x_1 - x_3) - y_1 = 13(5 - 6) - 1 = -14 \equiv 3 \pmod{17}$

Finally $2P = (5,1) + (5,1) = (6,3)$

[1] https://en.wikipedia.org/wiki/Modular_arithmetic
[2] https://en.wikipedia.org/wiki/Modular_multiplicative_inverse

Addition of Points on E

1. Commutativity. $P_1 + P_2 = P_2 + P_1$

2. Existence of identity. $P + O = P$

3. Existence of inverses. $P + (-P) = O$

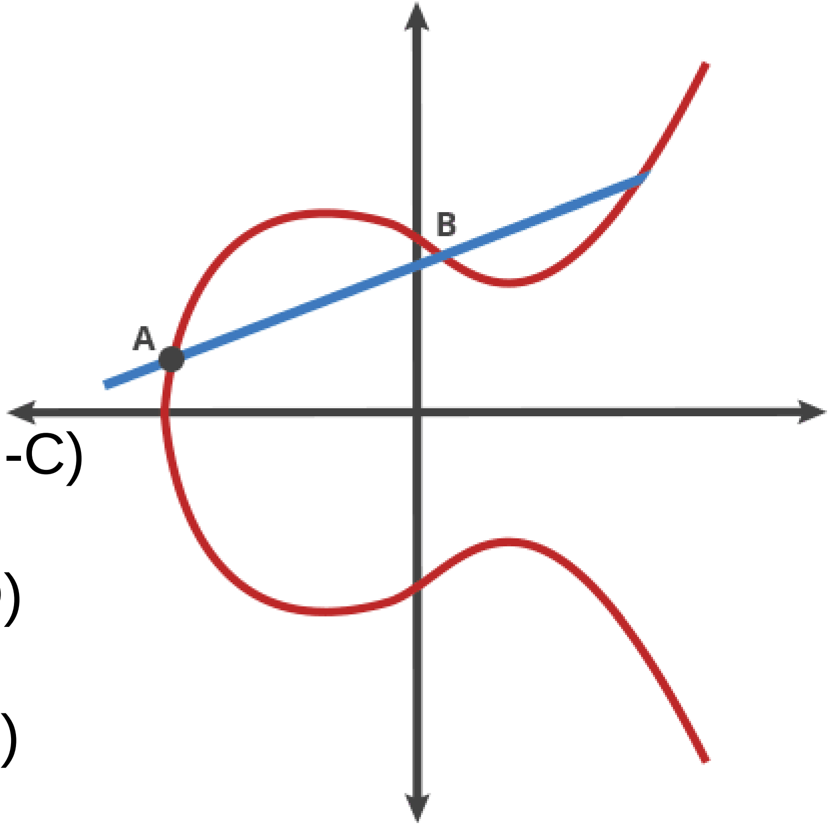
4. Associativity. $(P_1+P_2) + P_3 = P_1+(P_2+P_3)$

The point at infinity O, is the identity element.

- Online visual tool for elliptic curve: https://andrea.corbellini.name/ecc/interactive/reals-add.html

ECC’s trapdoor function

- We start with an arbitrary point on the curve. Next, we use the dot function to find a new point. Finally, we keep repeating the dot function to hop around the curve until we finally end up at our last point.

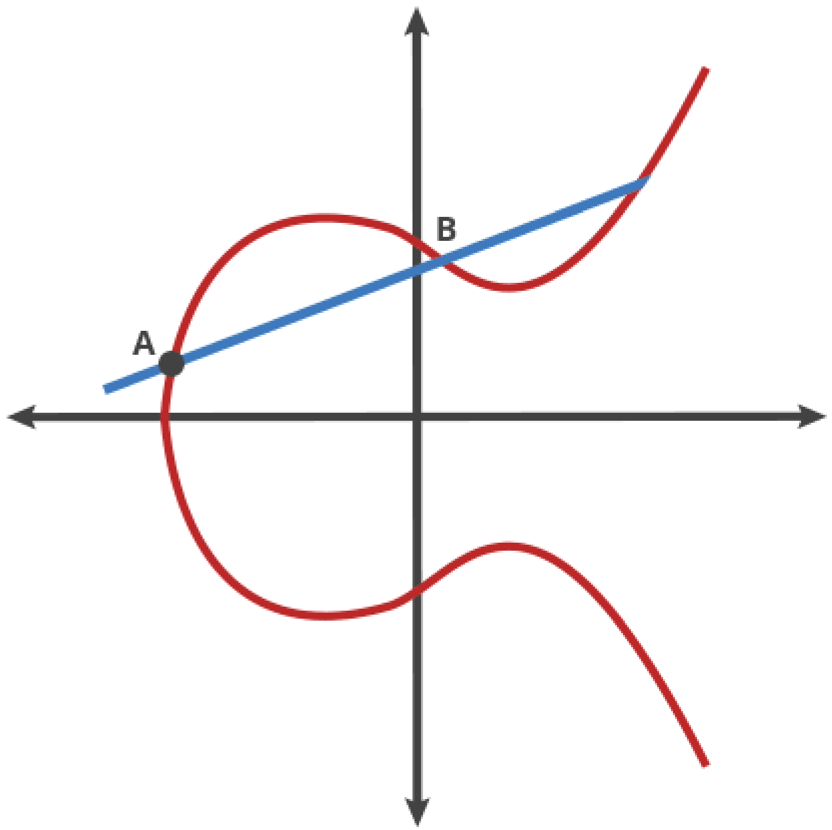


Starting at A:

- `A dot B = -C` (Draw a line from A to B and it intersects at -C)
- Reflect across the X-axis from -C to C
- `A dot C = -D` (Draw a line from A to C and it intersects -D)
- Reflect across the X-axis from -D to D
- `A dot D = -E` (Draw a line from A to D and it intersects -E)
- Reflect across the X-axis from -E to E

ECC's trapdoor function

- This is a great trapdoor function because if you know where the starting point (A) is and how many hops are required to get to the ending point (E).
- It's very easy to find the ending point.
- If all you know is where the starting point and ending point are, it's nearly *impossible* to find how many hops it took to get there.



Public Key: Starting Point A, Ending Point E
Private Key: Number of hops from A to E

Elliptic Curve Cryptography

Suppose that you are given two points ***P*** and ***Q*** in E($\mathbf{F}_p$).

The **<u>Elliptic Curve Discrete Logarithm Problem</u>** (ECDLP) is to find an integer ***m*** satisfying

$$Q = \overbrace{P + P + \cdots + P}^{m \text{ summands}} = mP.$$

- Cryptosystems are based on the idea that *m* is large and kept secret and attackers cannot compute it easily
- If *m* is known, an efficient method to compute the point multiplication *mP* is required to create a reasonable cryptosystem
- The extreme difficulty of the ECDLP yields highly efficient cryptosystems.

Elliptic Curve Diffie-Hellman Key Exchange

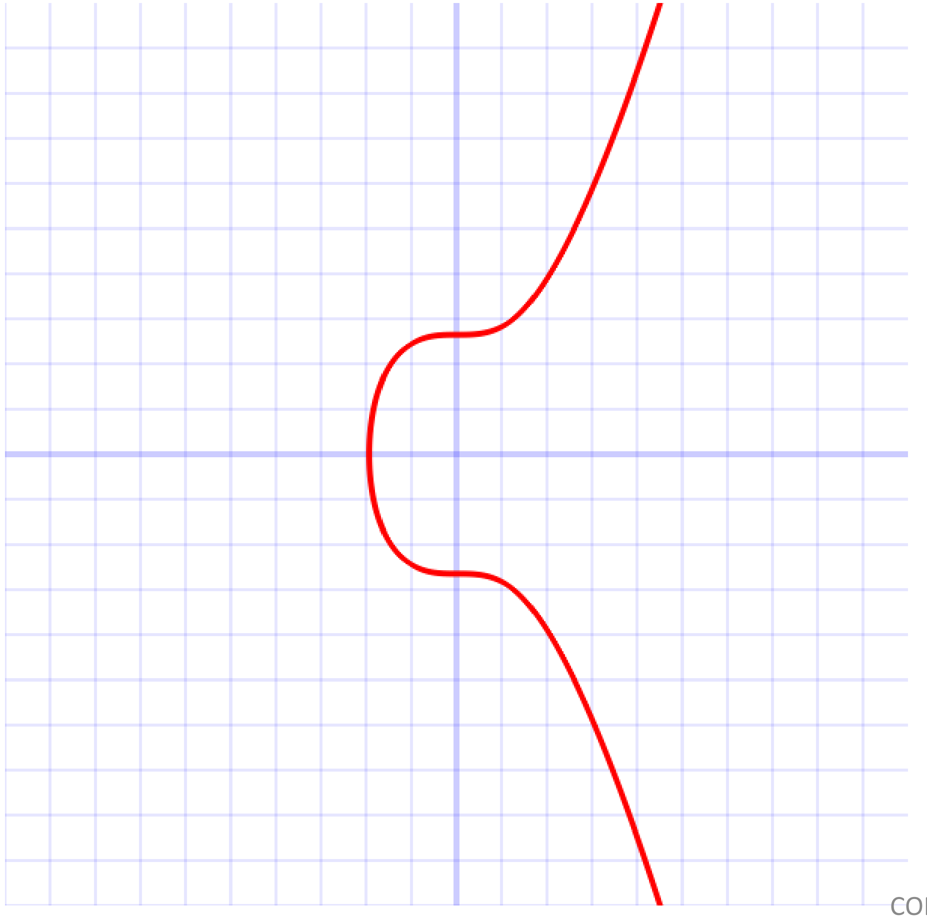
Public Knowledge: A group $\mathbf{E(F_p)}$ and a point $\mathbf{P}$ of order n.

BOB		ALICE
Choose secret $0 < \mathbf{b} < n$		Choose secret $0 < \mathbf{a} < n$
Compute $\mathbf{Q}_{Bob} = \mathbf{bP}$		Compute $\mathbf{Q}_{Alice} = \mathbf{aP}$
Send $\mathbf{Q}_{Bob}$	$\longrightarrow$	to Alice
to Bob	$\longleftarrow$	Send $\mathbf{Q}_{Alice}$
Compute $\mathbf{bQ}_{Alice}$		Compute $\mathbf{aQ}_{Bob}$

Bob and Alice have the shared value $\mathbf{bQ}_{Alice} = \mathbf{abP} = \mathbf{aQ}_{Bob}$

Curve secp256k1

- The curve used by Bitcoin is secp256k1
- The curve is $y^2 = x^3 + 7 \ mod \ p$, where *p* is a 256-bit prime number:



Note that because secp256k1 is actually defined over the field Z_p, its graph will in reality look like random scattered points, not anything like this.

https://en.bitcoin.it/wiki/Secp256k1

Curve secp256k1

- p = FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFFC2F (hex version)
- The base point is P = (x, y) where
  - x = 79BE667E F9DCBBAC 55A06295 CE870B07 029BFCDB 2DCE28D 59F2815B 16F81798
  - y = 483ADA77 26A3C465 5DA4FBFC 0E1108A8 FD17B448 A6855419 9C47D08F FB10D4B
- The order n of P is a 256-bit prime (s.t. nP = O), where
  - n = FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF BAAEDCE6 AF48A03B BFD25E8C D0364141

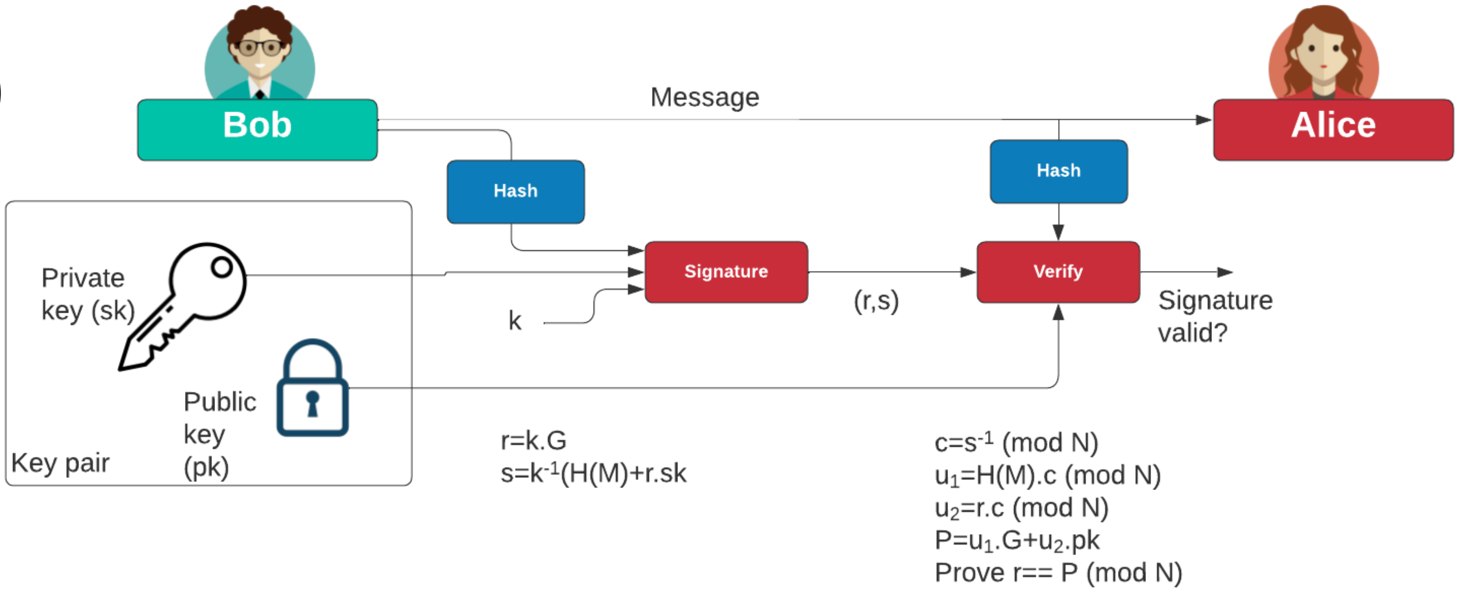
ECDSA in Bitcoin

- ECDSA - **E**lliptic **C**urve **D**igital **S**ignature **A**lgorithm

- Private Key: *k* in {1, …, n-1}
- Public Key:
  - $U = kP$
  - An elliptic curve (i.e., secp256k1)
  - *P*, elliptic curve base point
  - *n*, integer order of *P*, means that $n*P = O$, where *O* is the identity element.

Signature Generation

- To Sign message *m*
  1. Compute e = Hash(m)
  2. Pick a random j from {1, …, n-1}
  3. Compute $jP = (x, y)$, and $r = x \bmod n$
  4. Compute $s = j^{-1}(e + kr) \bmod n$
  5. Output (r, s) as the signature on m



Signature Verification

- Given message m, signature (r, s), public key U, verification consists of the following steps:
  1. Compute e = Hash(m)
  2. Compute $u = es^{-1} \bmod n$, and $v = rs^{-1} \bmod n$
  3. Compute $Q = uP + vU := (x, y)$

     // remember, Q is a point
  4. Accept if and only if $r = x \bmod n$
- *Proof:*

  $Q = uP + vU = es^{-1}P + rs^{-1}U = es^{-1}P + rs^{-1}kP = s^{-1}(e+rk)P, \;\; s^{-1} = j(e + kr)^{-1}$

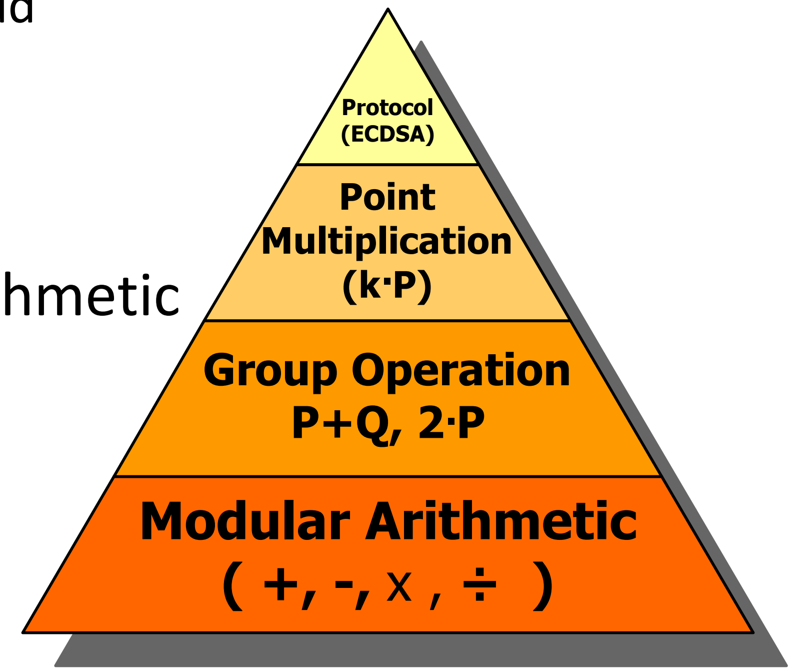
  So, $Q = jP := (x, y)$

  (We are calculating the same point $Q=jP$, just with a different set of equations.)

Implementations in Hardware and Software

- Elliptic curve computations usually regarded as consisting of four layers:
  - Basic modular arithmetic operations are computationally most expensive
  - Group operation implements point doubling and point addition
  - Point multiplication can be implemented using the Double-and-Add method
  - Upper layer protocols like ECDH and ECDSA

- Most efforts should go in optimizations of the modular arithmetic operations, such as
  - Modular addition and subtraction
  - Modular multiplication
  - Modular inversion



Summary

- Merkle Tree is based on Tree and Hash
- Merkle Tree Root is public for verification (integrity, membership)
- In blockchain, we build a Merkle Tree for transactions, and store the root in the block header
- The non-singular elliptic curve is the set of points and the point at infinity O
- The point at infinity O is the identity element
- Elliptic Curve Cryptography is based on elliptic curve logarithm problem
- In Bitcoin, we use Elliptic Curves Modulo p (secp256k1)
- Bitcoin uses Elliptic Curve Digital Signature Algorithm (ECDSA)