Elliptic Curve Cryptography (ECC)

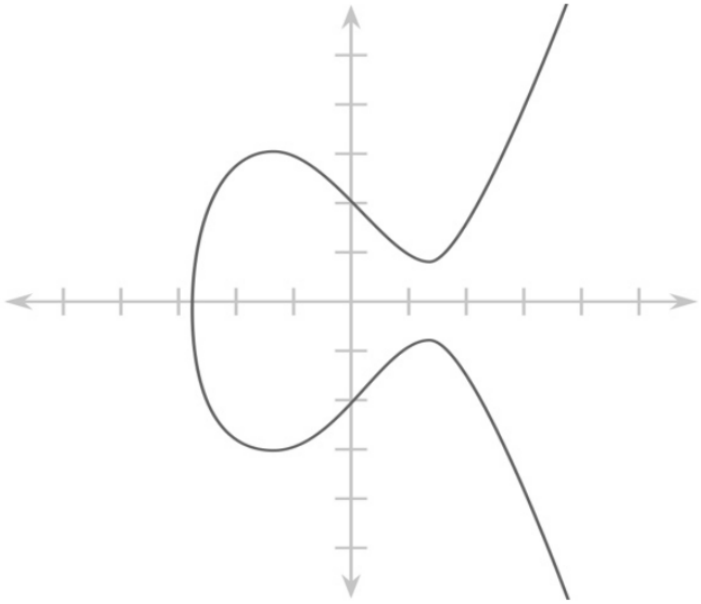
uses comparatively shorter keys for the equivalent level of security than other asymmetric cryptosystems

Algorithm Family	Cryptosystem(s)	80-bit	128-bit	192-bit	256-bit
Symmetric	AES	80	128	192	256
Discrete log	DH, DSA	1,024	3,072	7,680	15,360
Integer factorization	RSA	1,024	3,072	7,680	15,360
Elliptic curve	ECDH, ECDSA	160	256	384	512

DH: Diffie-Hellman
DSA: Digital Signature Algorithm
ECDH: Elliptic Curve Diffie-Hellman
ECDSA: Elliptic Curve Digital Signature Algorithm

maths

an elliptic curve is given by an equation of the form: $y^2 = x^3 + ax + b$



the curve is symmetric over the *x*-axis

this is because any value of *x* will yield two values of *y*

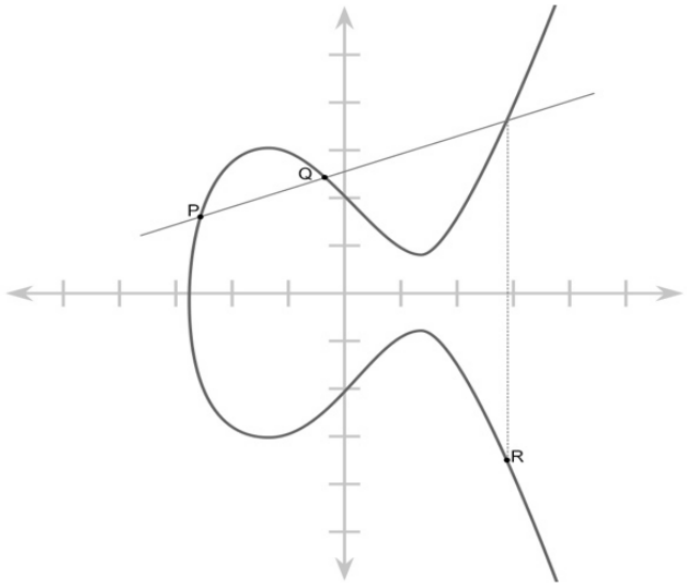
one positive and one negative (unless it's 0)

y can be computed as: $y = \pm\sqrt{x^3 + ax + b}$

note: any straight line drawn can intersect an elliptic curve at, at most, three points

point addition in an elliptic curve

the sum of two points on an elliptic curve is also a point on the elliptic curve



on the curve above, when points P and Q are added, the resulting sum is point R

i.e., $P+Q=R$

the resulting sum is the mirrored point of the third point of intersection of the straight line

the same point also can be added to itself

in that case, it is called doubling

ECC is based on doubling instead of using the addition of two unique points (more later)

finding the third point of intersection

starting with the equations for elliptic curves and straight lines:

$y^2=x^3+ax+b$

$y=mx+c$ (here, we use c instead so we don't confuse it with b in the elliptic curve equation)

a point that satisfies the following equation indicates where a straight-line intersects an elliptic curve:

$(mx+c)^2=x^3+ax+b$ (i.e., their y's are equivalent)

e.g., two given points:

$P=(x_{1,}y_1)$

$Q=(x_{2,}y_2)$

we can find the third point of intersection as:

$x_3=m^2-x_1-x_2$

$y_3=m(x_3-x_1)+y_1$

(where m is the slope of the straight line)

the slope m is given by:

$$m=\frac{y_2-y_1}{x_2-x_1}$$

(use this when two unique points are added – i.e., $P+Q$)

$$m = \frac{3x_1^2 + a}{2y_1}$$

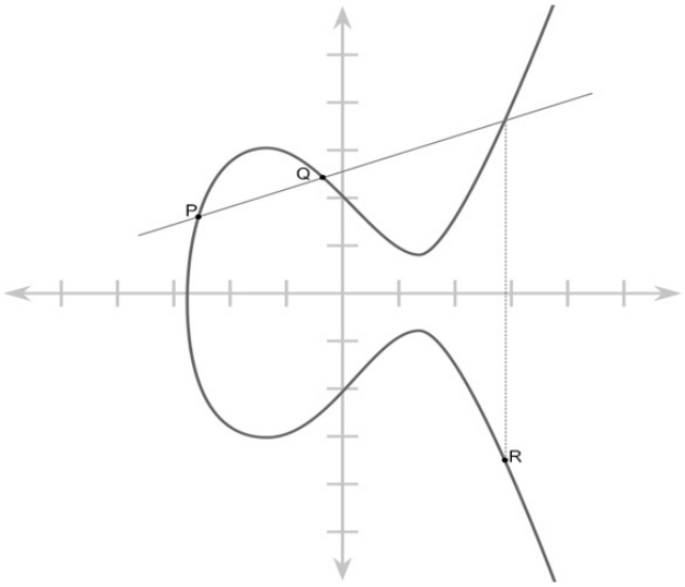
(use this when the same point is added to itself – i.e., doubling)

note that the values of *a* and *b* are predefined when an elliptic curve is defined

you could now pick *Q* and the third point to find another point on the elliptic curve
and keep going...until you get back to point *P*
these points are collectively called the **field** of the elliptic curve

ECC key generation
think about playing a game of pool (billiards)
hit a ball from *P* to *Q*
it will intersect the elliptic curve in one other place
where it does, it bounces to the mirrored portion on the elliptic curve
and keep hitting the ball doing the same thing…
suppose someone walks in at some point in the game
where was the ball last hit from?
how many times was it hit to get to where it currently is?
even if the person knows where the ball started and where it currently lies
it's hard to determine how many times the ball was hit
this is a trapdoor function!
keep hitting the ball
at some point, it will get back to its first position

e.g. (using the graph below):
P to Q → R
R to P → S (note: where it intersects the elliptic curve not at either point)
S to P → T (note: it's a line, but choose the direction that intersects the elliptic curve)



actual process

1. p, a large prime number is chosen
   - p defines the field in which the curve operates – the calculations stay within the field
   - because of this restrictive value, p, the actual equation used will be:
     $$y^2 = (x^3 + ax + b) \bmod p$$
2. a primitive element (starting point) on the curve is chosen
   - this point, A, becomes a generator
3. once we have this point, we add it to itself
   - we keep adding this point to the resulting sum, n number of times
   - these n repetitions of addition are known as hops
   - (this is like playing our game of billiards n times)
     $$2A = A + A$$
     $$3A = 2A + A$$
     $$4A = 3A + A$$
     …
     $$19A = 18A + A$$
     and so on…
4. at one point, the value of the resulting point starts repeating
   - this is when the cyclic group is complete
5. at any of the hops, we could stop and consider the obtained sum value to be the final point
   - well, obtained after n hops anyways
   - this final point is used as the public key
6. the number of hops, n, is used as the private key
   - it's hard to figure out how many hops were taken to get here (from the starting point)

OK, what about Alice and Bob?

- we use ECC a bit like Diffie-Hellman (i.e., to determine symmetric key(s)):
  - Alice and Bob agree on an elliptic curve function: $y^2 = (x^3 + ax + b) \bmod p$
    - i.e., a, b, and p are agreed upon
  - they also agree on the initial point/generator P
  - Alice picks a random integer i (her private key)
  - Alice generates her public key $Q_a = iP$ (i hops)
  - Bob picks a random integer j (his private key)
  - Bob generates his public key $Q_b = jP$ (j hops)
  - Alice and Bob exchange their public keys
  - Alice calculates iQ_b
  - Bob calculates jQ_a
    - proof: $iQ_b = i(jP) = j(iP) = jQ_a$
  - they use their shared secret to generate symmetric keys for exchanging messages

security of ECC

- depends on the simple computation of point "multiplication"
- but computation of the multiplicand is infeasible (given the original and resulting points)
- given P (initial point/generator) and Q (resulting point)
  - it is infeasible to calculate n (private key)
  - $Q = nP$