Elliptic Curve Public Key Cryptography

Why?

- ECC offers greater security for a given key size.

Elliptic Curve Public Key Cryptography

Why?

- ECC offers greater security for a given key size.
- The smaller key size also makes possible much more compact implementations for a given level of security, which means faster cryptographic operations, running on smaller chips or more compact software.

Elliptic Curve Public Key Cryptography

Why?

- ECC offers greater security for a given key size.
- The smaller key size also makes possible much more compact implementations for a given level of security, which means faster cryptographic operations, running on smaller chips or more compact software.
- There are extremely efficient, compact hardware implementations available for ECC exponentiation operations, offering potential reductions in implementation footprint even beyond those due to the smaller key length alone.

Elliptic Curve Public Key Cryptography

Why?

Date	Minimum of Strength	Symmetric Algorithms	Factoring Modulus	Discrete Logarithm		Elliptic Curve	Hash (A)	Hash (B)
				Key	Group			
(Legacy)	80	2TDEA*	1024	160	1024	160	SHA-1**	
2016 - 2030	112	3TDEA	2048	224	2048	224	SHA-224 SHA-512/224 SHA3-224	
2016 - 2030 & beyond	128	AES-128	3072	256	3072	256	SHA-256 SHA-512/256 SHA3-256	SHA-1
2016 - 2030 & beyond	192	AES-192	7680	384	7680	384	SHA-384 SHA3-384	SHA-224 SHA-512/224
2016 - 2030 & beyond	256	AES-256	15360	512	15360	512	SHA-512 SHA3-512	SHA-256 SHA-512/256 SHA-384 SHA-512 SHA3-512

Elliptic Curve Public Key Cryptography

Group:

A set of objects and an operation on pairs of those objects from which a third object is generated. Group must be closed, invertible, the operation must be associative, there must be an identity element.

Example: integers 0-9 and addition modulo 10

closed: sum of 2 numbers from 0-9 modulo 10 is an integer from 0 to 9.

identity: 0 since $0+x = x$ for any x

invertible: $x + (10-x) = 0 \bmod 10$ for any x
so $(10-x)$ is the inverse of x

associative: $(x+y)+z = x+(y+z)$

Elliptic Curve Public Key Cryptography

Finite Field:

If p is any prime number and n is any positive integer, then there exists a finite field of size p^n .

There are no other finite fields.

Example: $p = 7$, $n = 1$, operation = *

	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6
2	0	2	4	6	1	3	5
3	0	3	6	2	5	1	4
4	0	4	1	5	2	6	3
5	0	5	3	1	6	4	2
6	0	6	5	4	3	2	1

If $G = \{ g^0, g^1, g^2, g^3, g^4, g^5 \}$ is a group, then $g^6 = g^0$, and G is cyclic

generator = 3

3, 2, 6, 4, 5, 1, 3

inverse:

1:1, 2:4, 3:5, 4:2, 5:3, 6:6

Elliptic Curve Public Key Cryptography

Elliptic curve:

General form:

$$qy^2 = rx^3 + ax + b + sx^2 + ty$$

We want the set of solutions (x,y) to the equation

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

where $4a^3 + 27b^2 \neq 0$ (reason given later)

Elliptic Curve Public Key Cryptography

Elliptic curve:

General form:

$$qy^2 = rx^3 + ax + b + sx^2 + ty$$

We want the set of solutions (x,y) to the equation

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

where $4a^3 + 27b^2 \neq 0$ (reason given later)

Example: (p=7, n=2)

$y^2 = x^3 + x + 6$ (4+27*36 = 976 = 3 mod 7)

there are 49 integer points on this "curve"

(1,1), (1,6), (2,3), (2,4), (3,1), (3,6), (4,2), (4,5), (6,2), (6,5) …

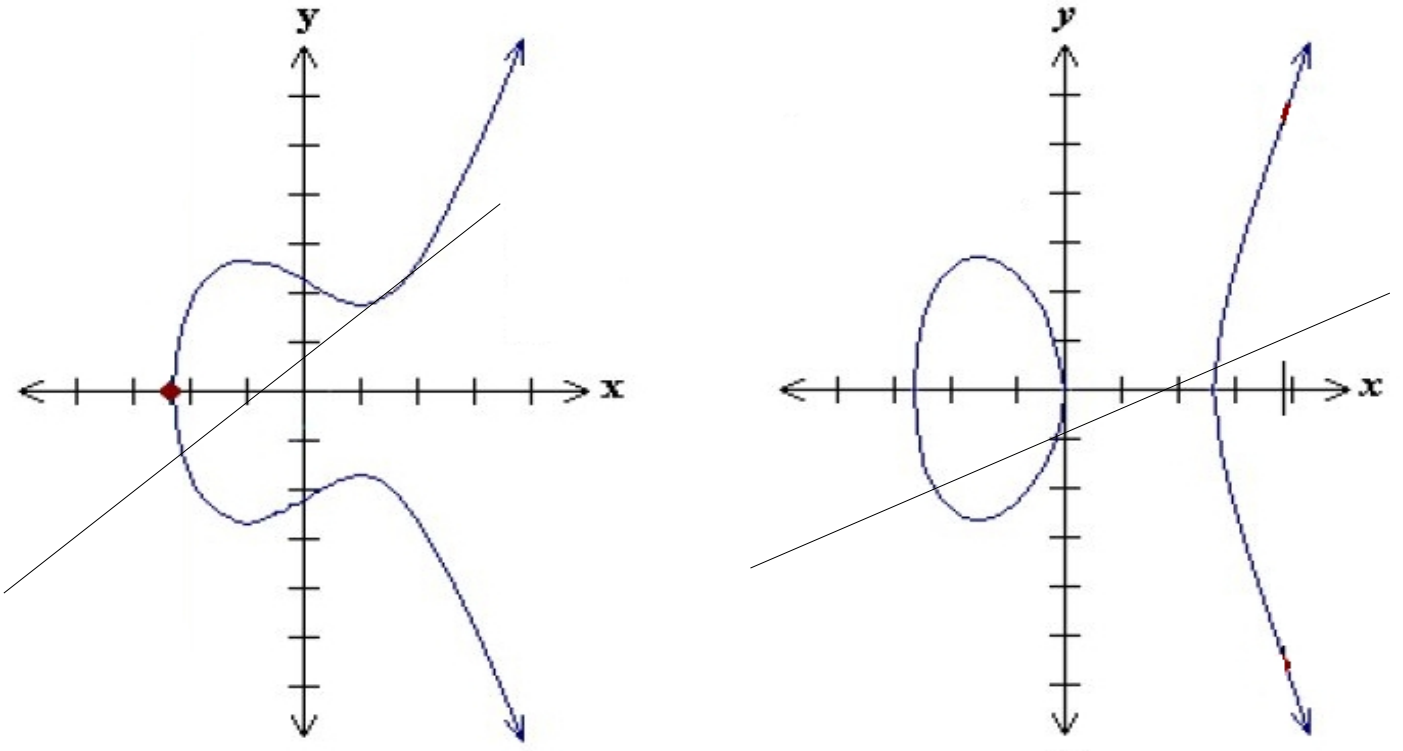
Note: (3,1), (2,3), (6,2) are on the same line

y = -2x +7, y = x/3 are the same line

(7 mod 7=0, 1/3 is the inverse of 3 which is -2)

Elliptic Curve Public Key Cryptography

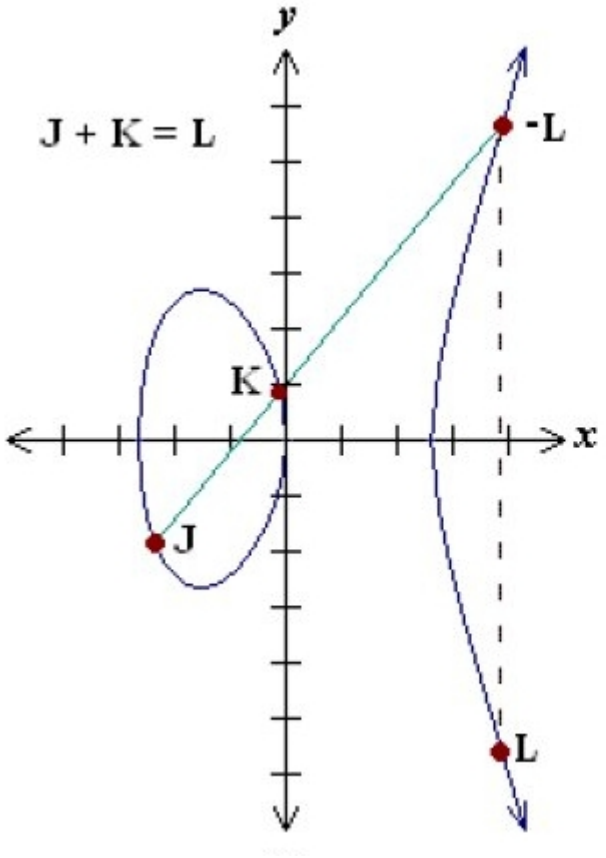
Shape of the curves and incident lines:

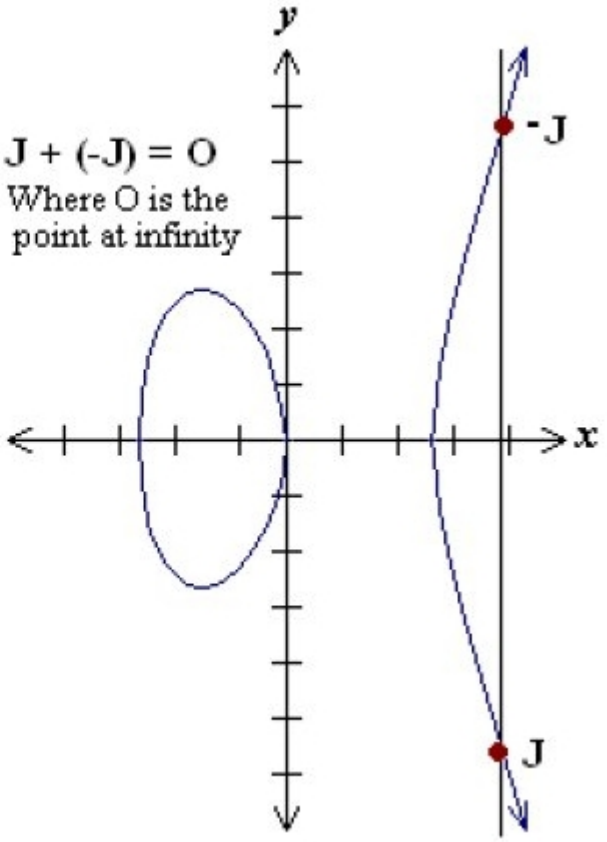


The curve is intersected by lines in 0, 1, 2, or 3 places
Touching in 1 place, a line is tangent to the curve
If (x,y) is on the curve, so is (x,-y)
Restriction ensures right side/left side do not meet at origin
Any two points generate a third point on the curve

Elliptic Curve Public Key Cryptography

Geometric explanation of addition:

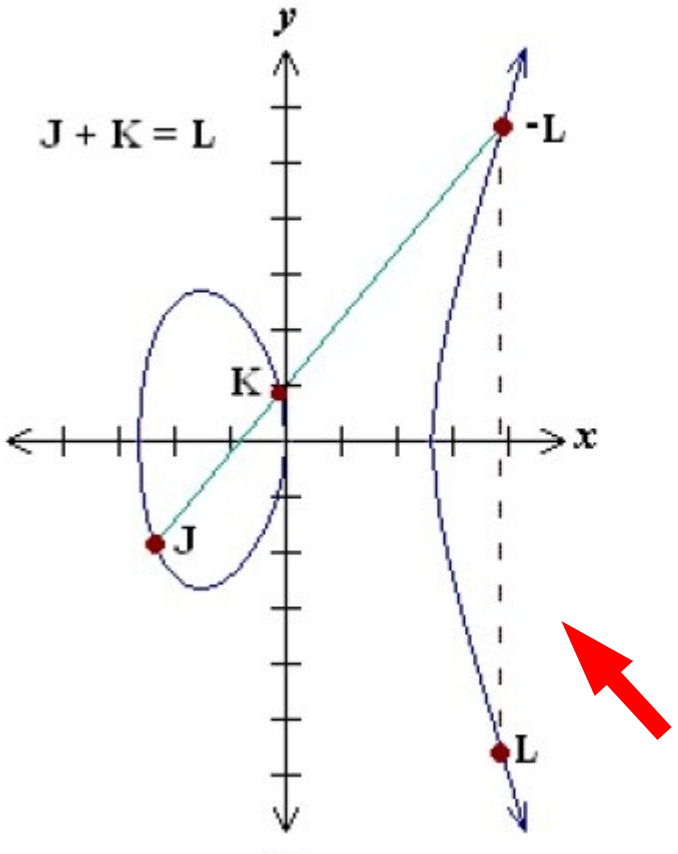


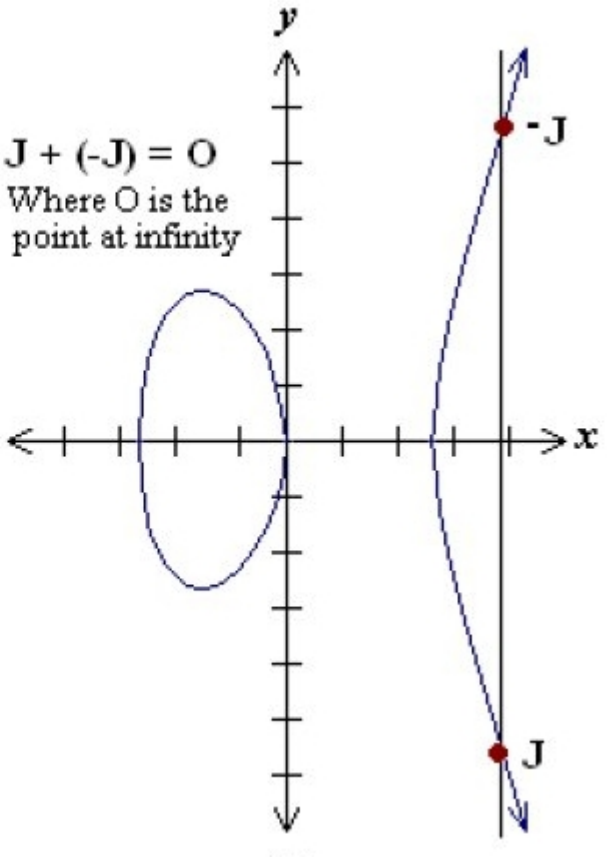


The negative of a point is its reflection in the x axis

Elliptic Curve Public Key Cryptography

Geometric explanation of addition:





If J and K are distinct and J ≠ -K

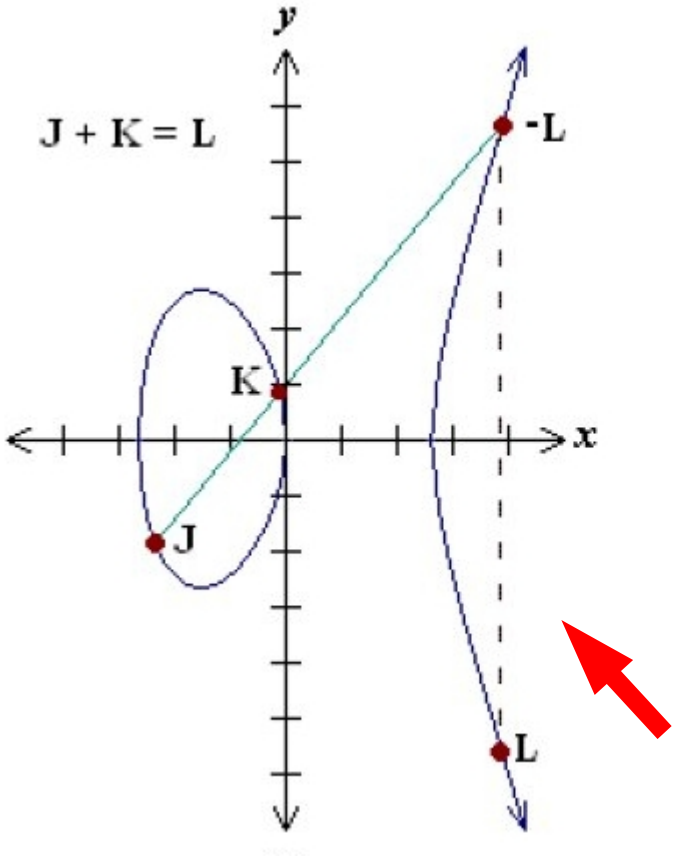
Draw a line though J and K – it intersects the curve at -L

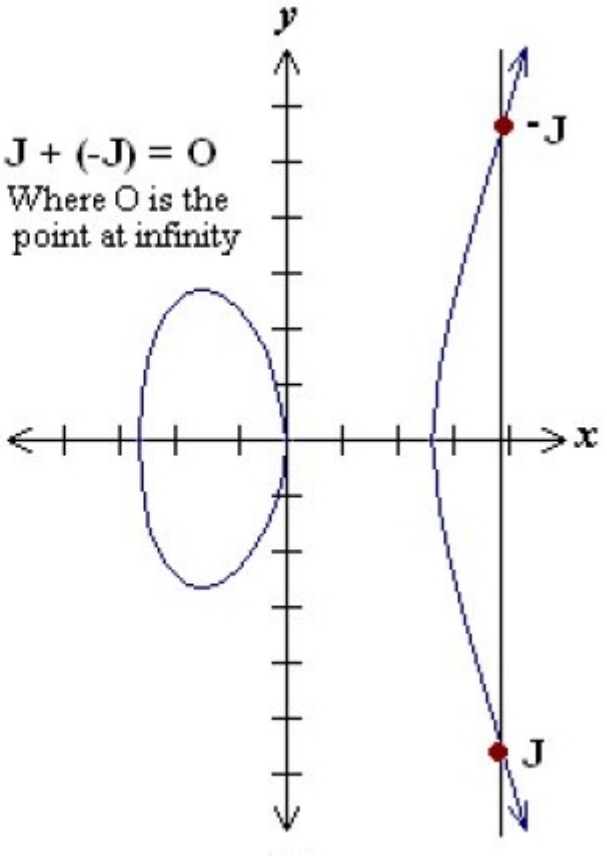
If J and K coordinates are rational, so are L, and -L

But the rationals get messier as addition is applied repeatedly

Elliptic Curve Public Key Cryptography

Geometric explanation of addition:



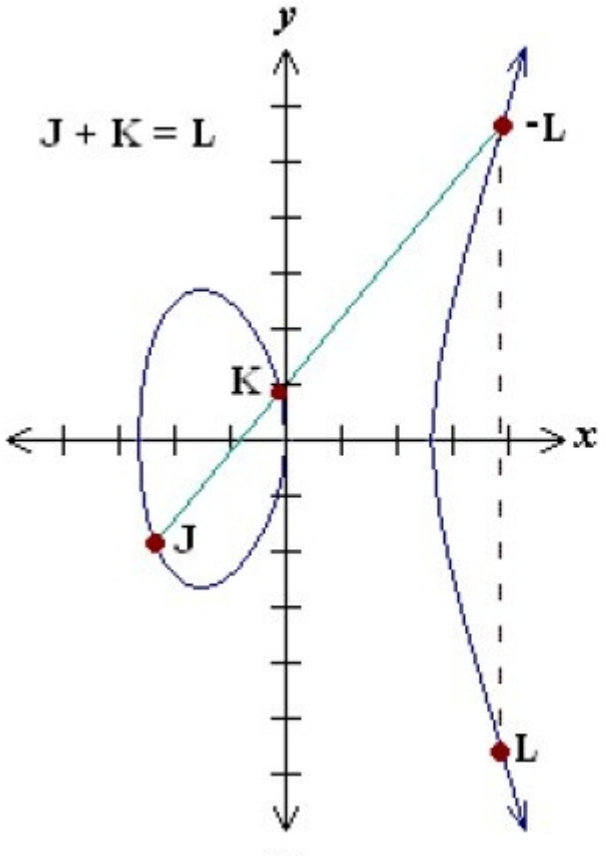


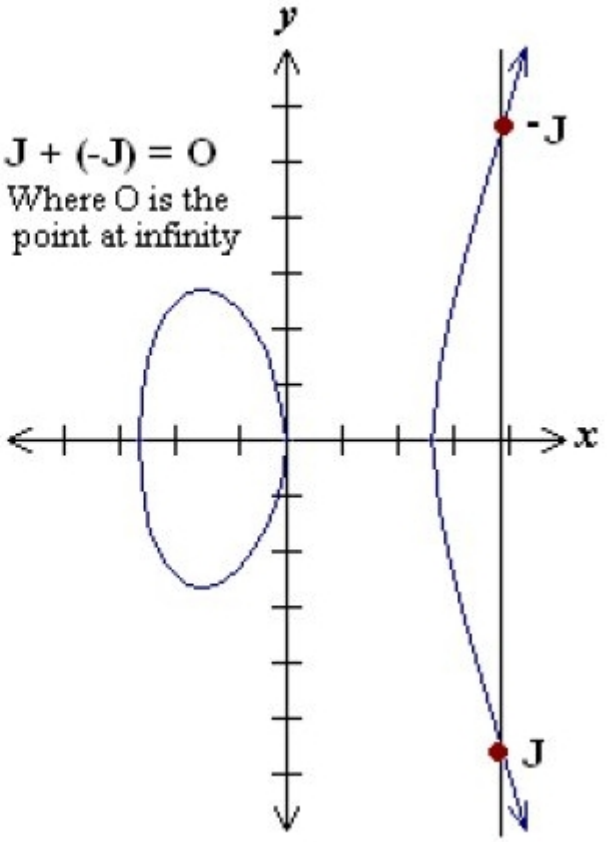
Curve: $y^2 = x^3 - 7x$

(-2.35,-1.86) + (-0.1,0.836) = (3.89,5.62), L=(3.89,-5.62)

Elliptic Curve Public Key Cryptography

Geometric explanation of addition:

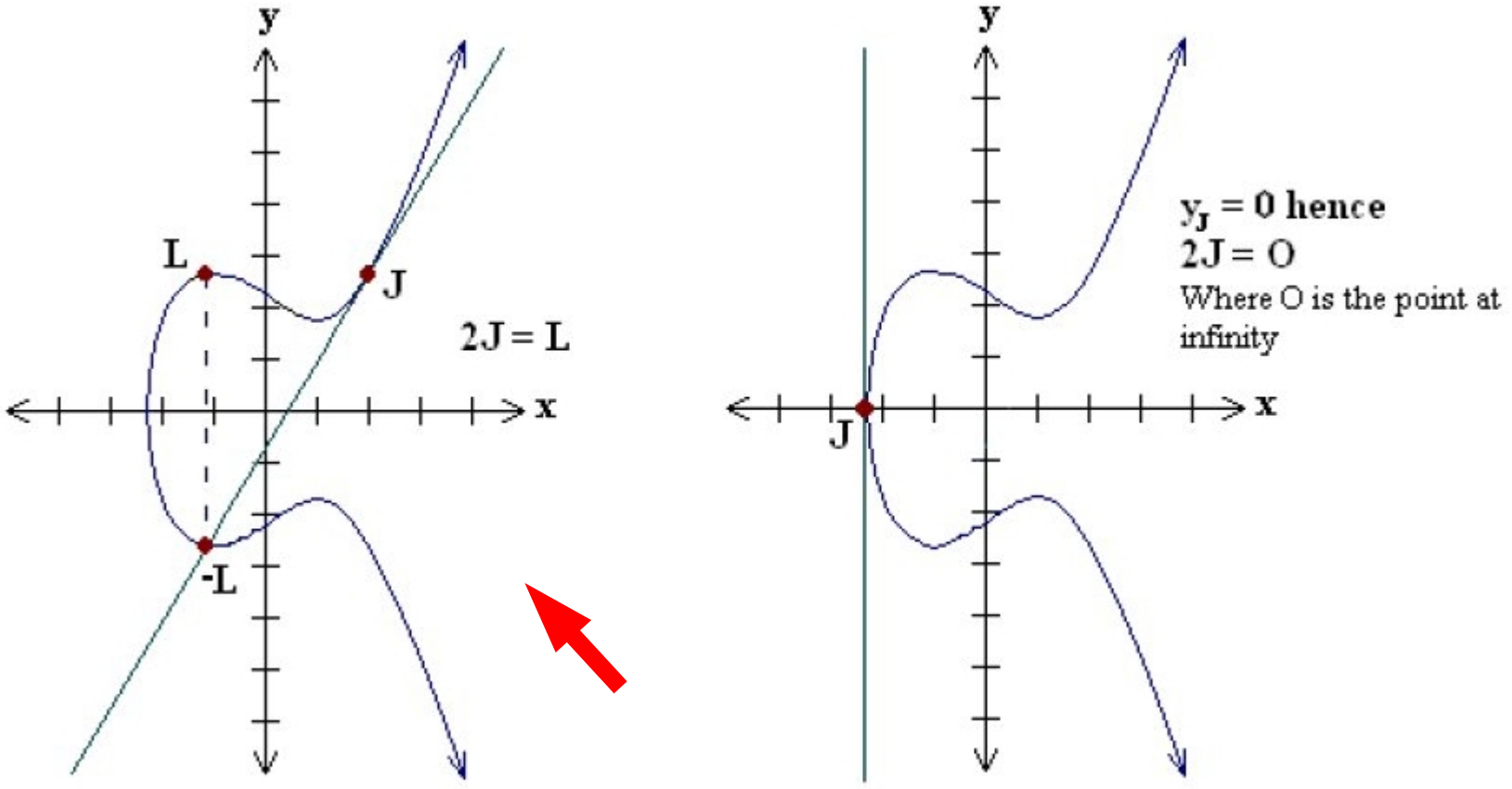




If J = -K the line though J and K does not hit a point on the curve – that is why O is the point at infinity

Elliptic Curve Public Key Cryptography

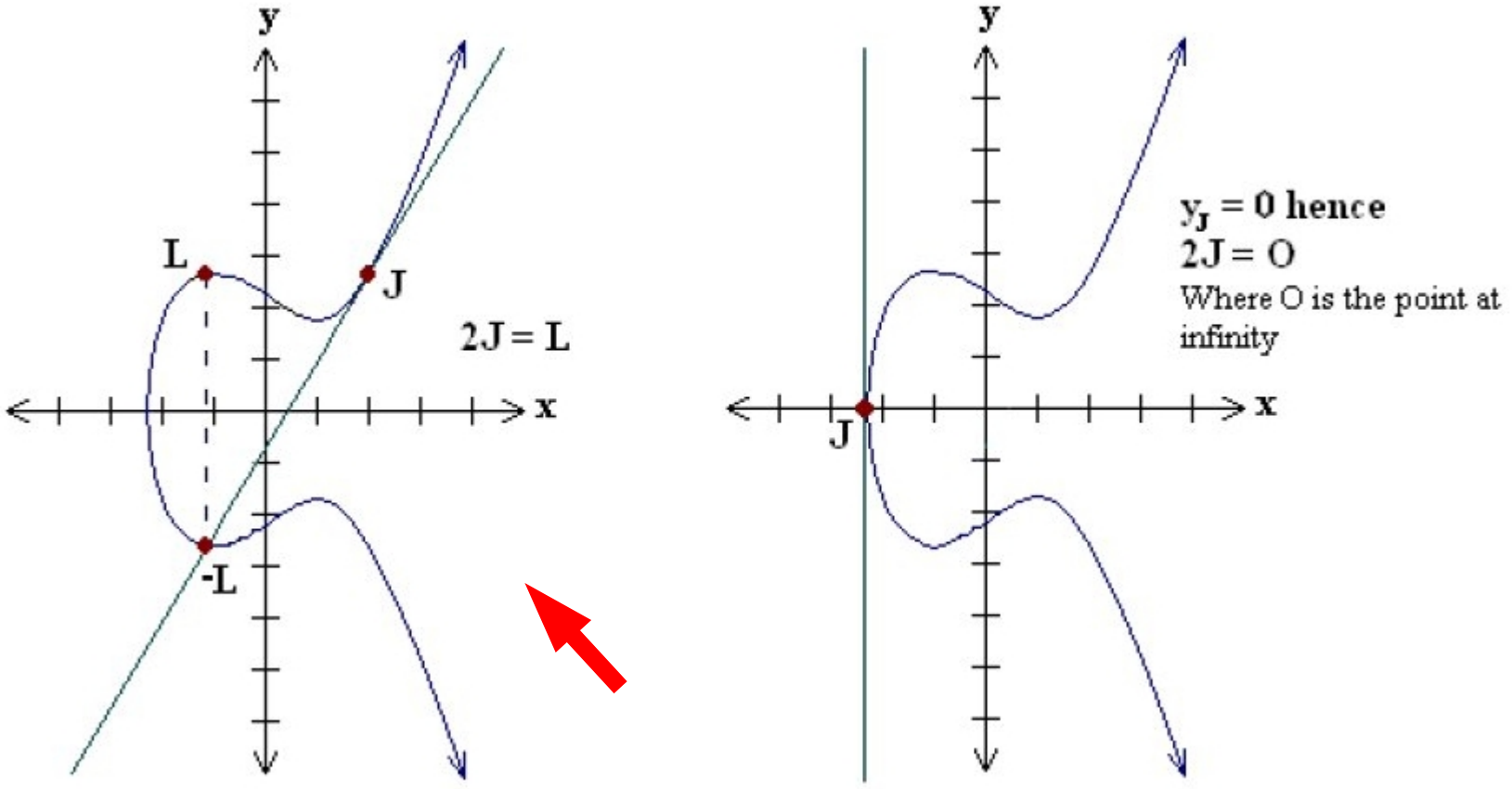
Geometric explanation of addition:



If J = K, vertical not 0, draw a line tangent to the curve at J and reflect its intersection with curve about x axis

Elliptic Curve Public Key Cryptography

Geometric explanation of addition:

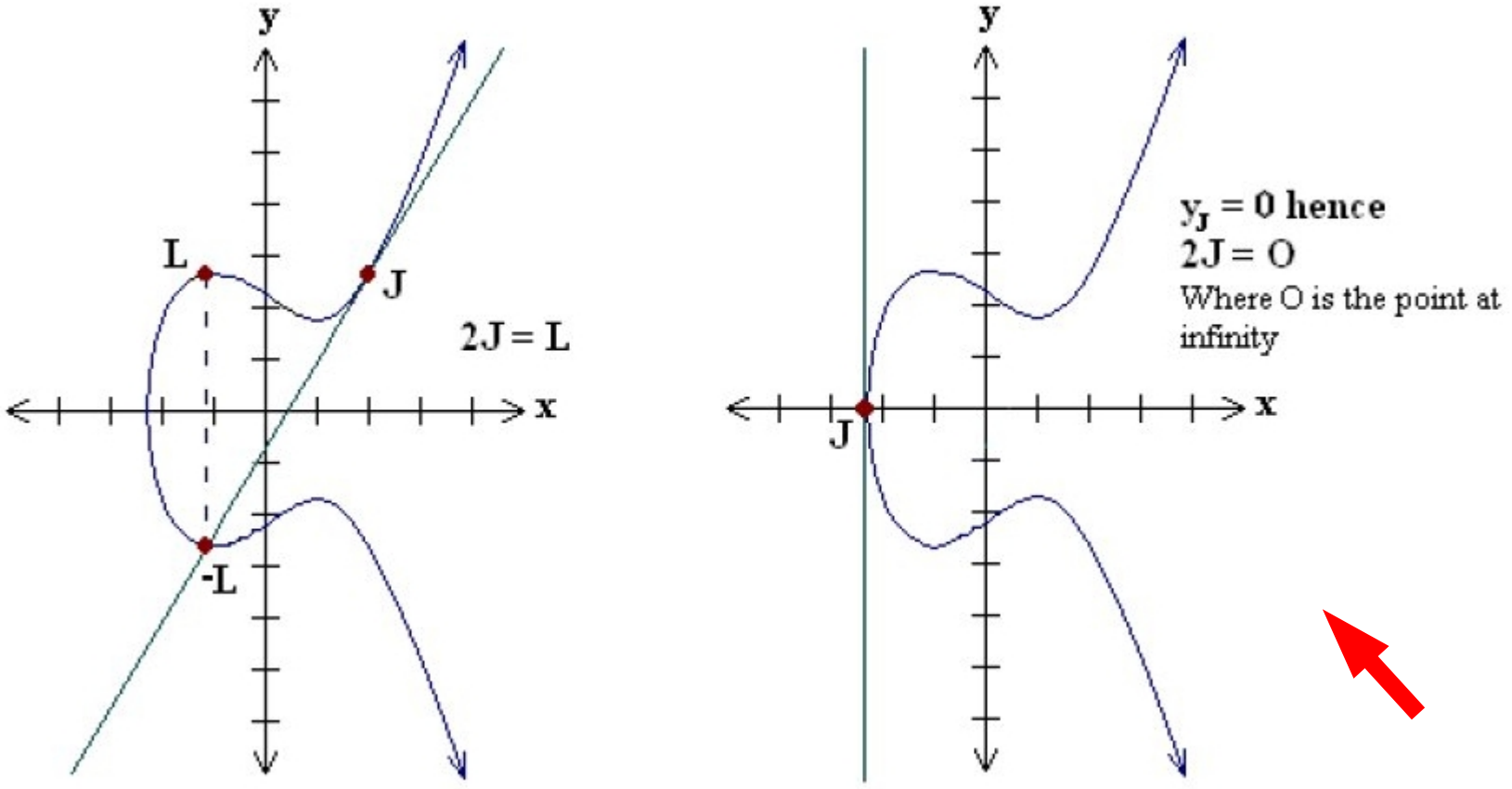


Curve: $y^2 = x^3 - 3x + 5$

(2, 2.65) + (2, 2.65) = (-1.11, -2.64), L=(-1.11, 2.64)

Elliptic Curve Public Key Cryptography

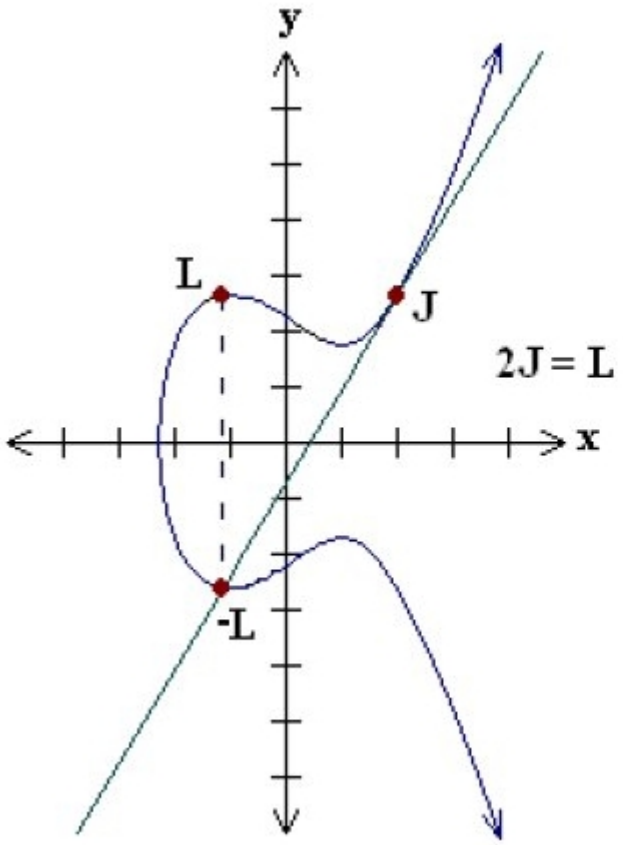
Geometric explanation of addition:

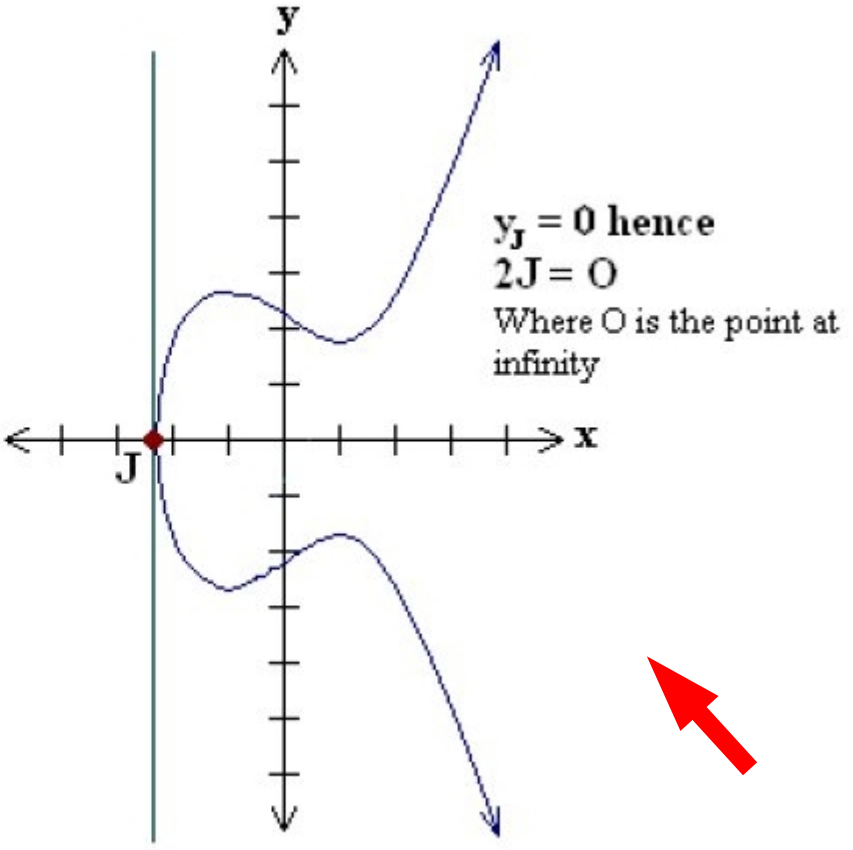


If J = K, vertical coordinate is 0, draw a line tangent to the curve at J it "intersects" at O so J + J = O.

Elliptic Curve Public Key Cryptography

Geometric explanation of addition:





$J + J + J = J + O = J$

$J + J + J + J + J = O + J = J$

$J + J + J + J = J + J = O$

$J + J + J + J + J + J = J + J = O$

Elliptic Curve Public Key Cryptography

Other properties of addition:

$P + Q = Q + P$

$P + O = P$

$O + O = O$

$P + (Q + R) = (P + Q) + R$

$P + (-P) = O$

Elliptic Curve Public Key Cryptography

Rigorously define the addition of two points *J* and *K*:

If J = -K then J + K = O (then $x_J = x_K$)

If J = K then J + J = L = (x_L, y_L)

where $x_L = m^2 - x_J - x_J$

$y_L = -m^3 + m(x_J + x_J) - y_J + mx_j$

$m = (3(x_J)^2 + a)/(2y_J)$

Otherwise J + K = L = (x_L, y_L)

where $x_L = m^2 - x_J - x_K$

$y_L = -m^3 + m(x_J + x_K) - y_J + mx_j$

$m = (y_J - y_K)/(x_J - x_K)$

Elliptic Curve Public Key Cryptography

Rigorously define the addition of two points *J* and *K*:

Example:

$y^2 = x^3 + 22x + 25 \bmod 47$

Points (1,1) and (25,28) satisfy this equation, the latter point obtained by doubling (1,1) – please check

Compute *m*:

$m = -27/-24 = 27/24$

the inverse of 24 mod 47 is 2 (so 1/24 mod 47 is 2)

thus $m = 54 \bmod 47 = 7$

Compute coordinates of next point:

$x_L = 7*7 - 1 - 25 = 23 \bmod 47$

$y_L = -7*7*7 + 7*(1+25) - 1 + 7*1 = -343+182-1+7$

$= -155 \bmod 47 = 33 \bmod 47$

Next point = (23,33)

Elliptic Curve Public Key Cryptography

Cyclic subgroup:

For any point G on the elliptic curve the set

$$\{ O,\ G, G+G, G+G+G, G+G+G+G, \ldots \}$$

is a cyclic subgroup of the points that are solutions to the elliptic curve – hence multiplying a point G by a scalar k, as in $kG = Q$, results in another solution Q.

Elliptic curve discrete logarithm problem:

Given G and Q, it is computationally infeasible to obtain k, if k is sufficiently large.

k is the discrete logarithm of Q to the base G.

Elliptic Curve Public Key Cryptography

Use in cryptography:

Do not use real numbers, use integers

$$y^2 \bmod p = x^3 + ax + b \bmod p$$

where a and b are no greater than p
the number of points so defined is p (0 to p-1)

Restriction:

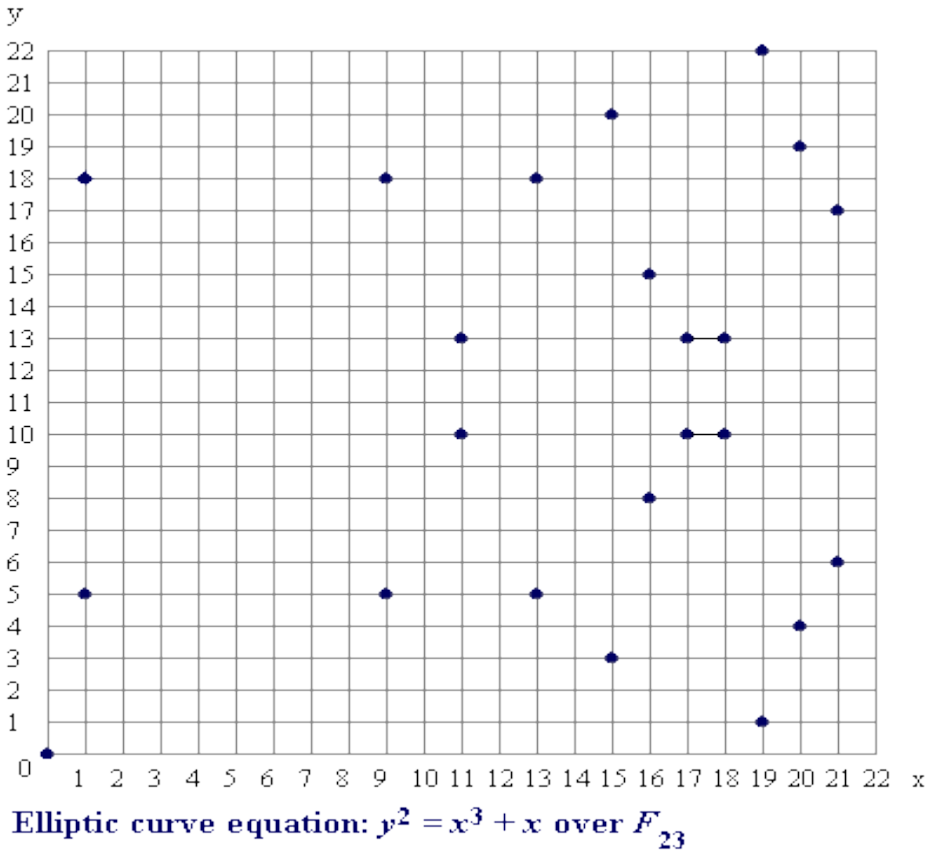
If $4a^3 + 27b^2 \bmod p \neq 0$ then the elliptic curve can be used to form a group – then there are no repeating factors

Elliptic Curve Public Key Cryptography

Example:

Set $p = 23$, $y^2 \bmod p = x^3 + 1x + 0 \bmod p$.
Observe $4a^3 + 27b^2 \bmod p = 4 \neq 0$
Choose G = (9,5) (on curve: 25 mod 23 = 729+9 mod 23)

The 23 points on this curve:
(0,0) (1,5) (1,18) (9,5) (9,18)
(11,10) (11,13) (13,5) (13,18)
(15,3) (15,20) (16,8) (16,15)
(17,10) (17,13) (18,10) (18,13)
(19,1) (19,22) (20,4) (20,19)
(21,6) (21,17)



Elliptic curve equation: $y^2 = x^3 + x$ over F_{23}

From: ECC tutorial
www.certicom.com

Elliptic Curve Public Key Cryptography

Cryptosystem parameters:

The number of integers to use to express points is a prime number p.

The public key is a point in the curve and the private key is a random number (the k from before).

The public key is obtained by multiplying the private key with the generator point G in the curve.

The number n is the smallest positive integer such that $n\mathrm{G} = \mathrm{O}$, n had better be prime.

The number of points on the elliptic curve divided by n is the parameter h.

Elliptic Curve Public Key Cryptography

EC Diffie-Hellman algorithm for key generation:

Sender and receiver agree on parameters p, a, b, G, n, h

Private keys are random integers d_S, and d_R, less than n

Public keys are $e_S = d_S\mathrm{G}$ and $e_R = d_R\mathrm{G}$

Algorithm for computing a shared secret:

Sender computes $\mathrm{K} = (x_\mathrm{K}, y_\mathrm{K}) = d_S e_R$

Receiver computes $\mathrm{L} = (x_\mathrm{L}, y_\mathrm{L}) = d_R e_S$

We know $d_S e_R = d_S d_R \mathrm{G} = d_R d_S \mathrm{G} = d_R e_S$

Hence K = L and $x_\mathrm{K} = x_\mathrm{L}$

The shared secret is x_K

Elliptic Curve Public Key Cryptography

EC Digital Signature Algorithm:

Sender and receiver agree on parameters p, a, b, G, n, h

Sender's private key is a random integer d_s, less than n

Sender's public key is $e_s = d_s\text{G}$.

Algorithm for computing a digital signature:

Compute $X = \text{HASH}(m)$

do {

Select a random integer k from $[1, n-1]$

Let $(x_1, y_1) = k\text{G}$

Compute $r = x_1 \bmod n$

Compute $s = k^{-1}(X + d_s r) \bmod n$

} while ($r == 0 \,||\, s == 0$)

(r,s) is the signature that is sent to the receiver

Elliptic Curve Public Key Cryptography

Algorithm for verifying a digital signature:

Receiver gets the sender's public key e_s and (r,s) and m.

Make sure r and s are between 1 and n-1.

Compute $X = \text{HASH}(m)$

Compute $w = s^{-1} \bmod n$

Compute $u_1 = Xw \bmod n$

Compute $u_2 = rw \bmod n$

Compute $(x_1,y_1) = u_1\text{G} + u_2 e_s$

Verify signature if $x_1 = r \bmod n$

Elliptic Curve Public Key Cryptography

What's so good about ECC?

- Smaller keys, ciphertexts and signatures
- Very fast key generation
- Fast signatures
- Moderately fast encryption and decryption
- Signatures can be computed in two stages, allowing latency much lower than inverse throughput
- Good protocols for authenticated key exchange
- Good US government support
- Special curves with bilinear pairings allow new-fangled cryptography
- Binary curves are really fast in hardware

Elliptic Curve Public Key Cryptography

What's so bad about ECC?

Complicated and tricky to implement securely,
particularly true for the standard curves

Standards aren't state-of-the-art, particularly ECDSA
which is kind of a hack compared to Schnorr signatures

Signing with a broken random number generator
compromises the key

Still has some patent problems, especially for binary curves

Newer algorithms might have some unknown weaknesses

Binary curves are slightly scary

Don't use DUAL_EC_DRBG, since it has a back door

Elliptic Curve Public Key Cryptography

What's so good about RSA?

- Relatively fast, very simple encryption and verification
- Easier to implement than ECC
- Easier to understand
- Signing and decryption are similar
- Encryption and verification are similar
- Widely deployed, good industry support

Elliptic Curve Public Key Cryptography

What's so bad about RSA?

- Very slow key generation
- Slow signing and decryption, which are slightly tricky to implement securely
- Two-part key is vulnerable to GCD/CRT attack if poorly implemented