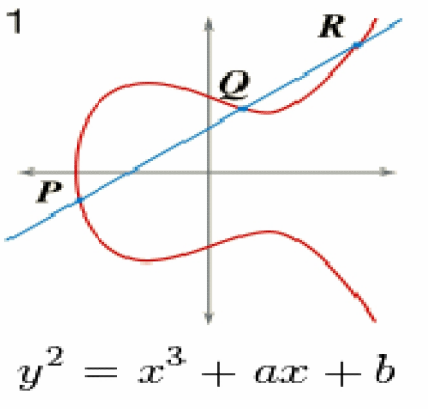
Elliptic Curve Cryptographic (ECC)

Elliptic Curve Cryptography (ECC) was discovered in 1985 by Victor Miller (IBM) and Neil Koblitz (University of Washington) as an alternative mechanism for implementing public-key cryptography. In mathematics, an **elliptic curve** is a plane algebraic **curve** defined by an equation of the form

The equation of an elliptic curve is given as,

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

Note that the curve coefficients have to fulfill one condition: $\mathbf{4a^3+27b^2 \neq 0.}$
This condition guarantees that the curve will not contain any singularities.



Few terms that will be used,

E -> Elliptic Curve

P -> Point on the curve

n -> Maximum limit (This should be a prime number)

So in theory, if you make a list of a lot of "y" coordinates (real numbers) and apply that equation to all of them, you should obtain a curve similar to shown.

One property of this curve:

This is probably the most important property of this curve, and it is that **if a line intersects two points in the curve it will always intersect a third.** This is very important; since that third point will be the representation of the public key, remember that all this hassle is to create public keys from a private key, that at this point are massive prime numbers.

Why Elliptic Curve?

ECC is a **public key** based, such as **RSA**, but it is sort of represented in an algebraic structure, **ECC** offers the same security than **RSA** but at a **smaller footprint**, also it's less cpu intensive so it's ideal for mobile devices and faster acting networks.

Symmetric Key Size (bits)	RSA and Diffie-Hellman Key Size (bits)	Elliptic Curve Key Size (bits)
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	521

Key Generation

Key generation is an important part where we have to generate both public key and private key. The sender will be encrypting the message with receiver's public key and the receiver will decrypt using its private key.

Now, we have to select a number **'d'** within the range of **'n'**.

Using the following equation we can generate the public key: **Q = d * P**

d = The random number that we have selected within the range of (**1 to n-1**). **P** is the point on the curve.

'Q' is the public key and **'d' is the private key.**

Encryption

Let 'm' be the message that we are sending. We have to represent this message on the curve. Consider *'m'* has the point *'M'* on the curve *'E'*. Randomly select 'k' from [1 – (n-1)].

Two cipher texts will be generated let it be **C1** and **C2**.

C1 = k*P

C2 = M + k*Q

C1 and C2 will be sent.

Decryption

We have to get back the message 'm' that was sent to us,

M = C2 – d * C1

M is the original message that we have sent.

Proof

How does we get back the message: **M = C2 – d * C1**

'M' can be represented as 'C2 – d * C1'

C2 – d * C1 = (M + k * Q) – d * (k * P) (C2 = M + k * Q and C1 = k * P)

= M + k * d * P – d * k *P (canceling out k * d * P)

= M (Original Message)