Generating a public key (can be shared, like a checking account number ) from a private key( known to only you, like ATM code ) is a hopping process , across points on an elliptic curve.

Elliptic curve eqn. is y^2 = x^3 + ax + b, for bitcoin math a=0, b=7, so eqn. is y^2=x^3 + 7

The hopping process goes as below

1. Start from a point called GENERATOR POINT. This is a universal constant. In figure (1), draw a tangent from the point, this will intersect the curve only at one point (property of elliptic curve). From the intersecting point draw a perpendicular line across x-axis, reaching point A1.

A1 is given by the following eqn , where G is the generator point, and in crypto math A1 is called double of G , and is represented as A1=2G, G.x, and G.y are co ordinates of G.

slope = (3\*G.x^2 + a) / 2\*G.y

**A1.x** = slope^2 - 2\*G.x

**A1.y** = slope \*(G.x – A1.x) - G.y

1. Now connect A1, and G, it will intersect the curve only at one point, from that point , draw a perp. to reach point A2. In Crypto terms A2=A1+G=3G ( Figure II )

A2 is given as follows

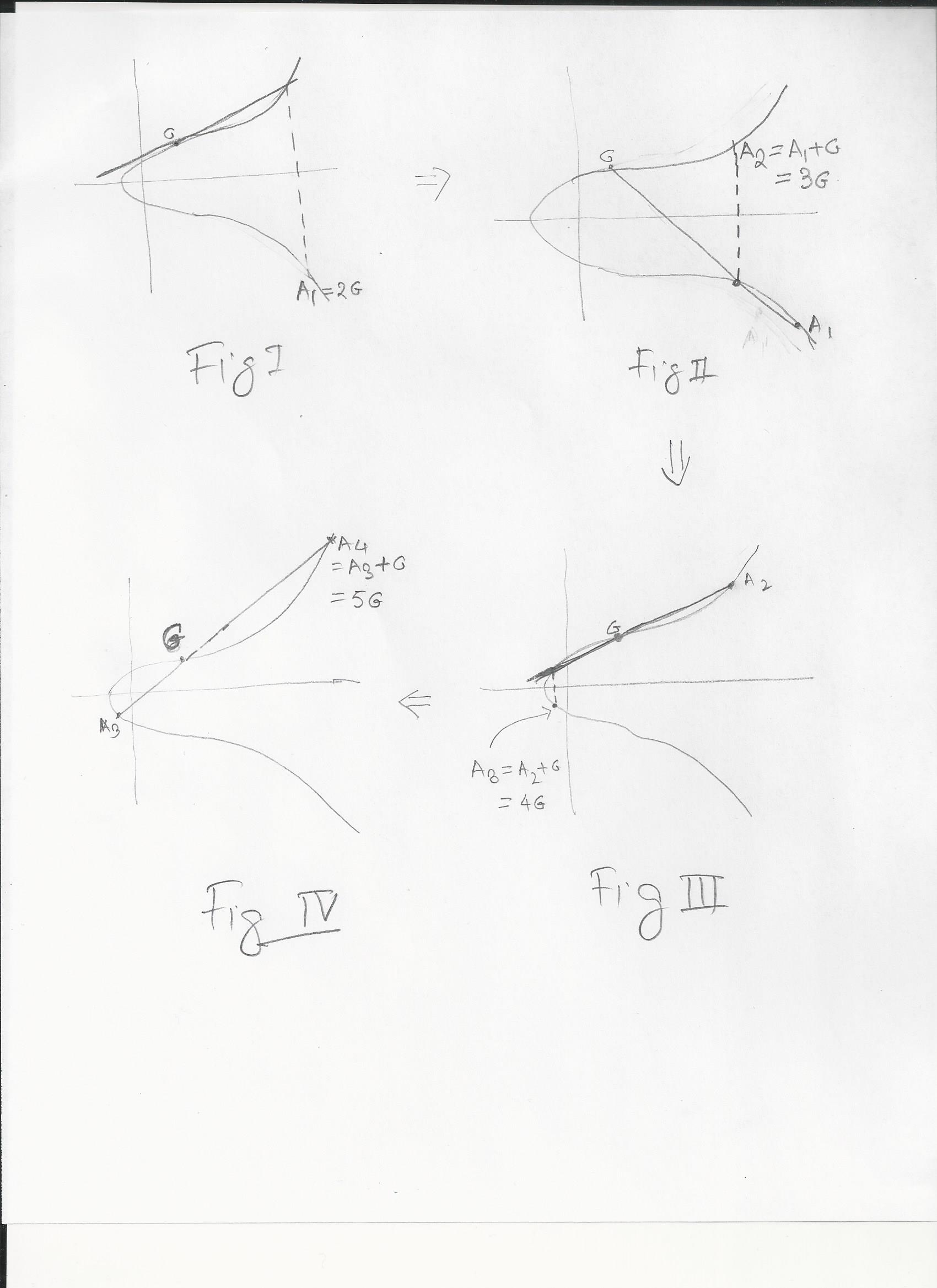
slope = (A1.y - G.y) / (A1.x – G.x)

**A2.x** = slope^2 – A1.x - G.x

**A2.y** = slope \*(G.x – A2.x) – G.y

1. Keep repeating the process with each point A(i-1) , and G, each time arrive at a new point A(i) so that A(i) = A(i-1) + G
2. After an extremely large number of hops like above, you will arrive at a final point on the curve. This point is your public key, and the number of iterations is your private key , which you choose. You can understand that these operations are so opaque that after very large number of hops it is impossible to back out the private key from the public key, even though you may know the generator point and the public key. Only way is brute force, and it may take trillions of years.

So the final conclusion is public key = privatekey \* Genarator point . This is known as scalar multiplication ( ie multiplying a point by a scalar private key ) in elliptic curve math.



**Now Some specifics and some hiccups:**

Strength of the above cryptograhical process stems from the fact that generator point and private key( i.e., # of hops ) are very large numbers. Generator point is

( 0x79BE667EF9DCBBAC55A06295CE870B07029BFCDB2DCE28D959F2815B16F81798,

0x483ADA7726A3C4655DA4FBFC0E1108A8FD17B448A68554199C47D08FFB10D4B8 )

Private keys ( which you choose ) look like 0xd8a8bb5aa721409deb930e8c2278b444d1bdb0f0a8a6e8cb97ec0ea9167175c5

You can see that all are 256 bit integers.

Therefore all the above computations will very soon generator arithmetic overflow, very quickly.

So plain algebraic computation will not work**. Here comes modulo arithmetic in rescue.** so to understand elliptic curve cryptography we need to understand basics of modulo- algebra. Next topic