δ **1-st Part: Elliptic Curve Cryptography - ECC**

$Z_p^*=\{1, 2, \ldots, p-1\}$

Elliptic-curve cryptography (ECC) is an approach to public-key cryptography based on the algebraic structure of elliptic curves over finite fields.
ECC requires smaller keys compared to non-ECC cryptography to provide equivalent security.
For example, to achieve the same security ensured by ECC having private key of 256 bit length, it is required to use over 3000 bit private key length for RSA cryptosystem and others.

Elliptic curves are applicable for key agreement, digital signatures, pseudo-random generators and other tasks.
Indirectly, they can be used for encryption by combining the key agreement with a symmetric encryption scheme AES-128, 192, 256.

Elliptic Curve Digital Signature Algorithm - Bitcoin Wiki **(ECDSA)**
https://en.bitcoin.it/wiki/Elliptic_Curve_Digital_Signature_AlgorithmFeb 10, 2015
Elliptic Curve Digital Signature Algorithm or **ECDSA** is a cryptographic algorithm used by Bitcoin, Ethereum and other blockchain methods to ensure that funds can only be spent by their owner.
https://en.wikipedia.org/wiki/Elliptic-curve_cryptography

Finite Field denoted by $\boldsymbol{F_p}$ (or rarely $\boldsymbol{Z_p}$), when: $\boldsymbol{p}$ is prime.

$\boldsymbol{F_p}=\{0, 1, 2, 3, \ldots, \boldsymbol{p}-1\}$; $+_{\mathbf{mod}\ p}$, $-_{\mathbf{mod}\ p}$, $\bullet_{\mathbf{mod}\ p}$, $:_{\mathbf{mod}\ p}$.
Cyclic Group: $\boldsymbol{Z_p}^* = \{1, 2, 3, \ldots, \boldsymbol{p}-1\}$; $\bullet_{\mathbf{mod}\ p}$, $:_{\mathbf{mod}\ p}$.

For example, if $\boldsymbol{p}=11$, then one of the generetors is $\boldsymbol{g}=2$.
$\boldsymbol{Z_{11}}^* = \{1, 2, 3, \ldots, 10\}$

$x \in Z_{10}$				$a \in Z_{11}^*$
0				1
1				2
2				4
3				8
4				5
5				10
6				9
7				7
8				3
9				6

The main function used in cryptography was Discrete Exponent Function - DEF:
$\mathrm{DEF}(x) = g^x \bmod p = a$.

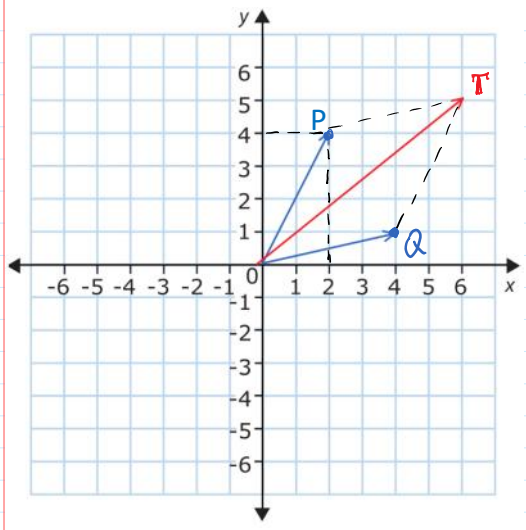
x	0	1	2	3	4	5	6	7	8	9	10
$2^x \bmod p$	1	2	4	8	5	10	9	7	3	6	1

Discrete Exponent Function - $\mathrm{DEF}_g(x)=g^x \bmod p$
x is in $\boldsymbol{Z_{p-1}} = \boldsymbol{Z_{10}} = \{0, 1, 2, \ldots, 9\}$;
$\mathrm{DEF}(x)$ is in $\boldsymbol{Z_p}^* = \boldsymbol{Z_{11}}^* = \{1, 2, 3, \ldots, 10\}$;
DEF: $\boldsymbol{Z_{p-1}} \rightarrow \boldsymbol{Z_p}^*$.

For illustration of 1-to-1 mapping of $\mathbf{DEF}_7(x)$ we perform the following step-by-step computations.

	$x \in Z_{10}$	$a \in Z_{11}^*$
$7^0 = 1 \bmod 11$	0	1
$7^1 = 7 \bmod 11$	1	2
$7^2 = 5 \bmod 11$	2	3
$7^3 = 2 \bmod 11$	3	4
$7^4 = 3 \bmod 11$	4	5
$7^5 = 10 \bmod 11$	5	6
$7^6 = 4 \bmod 11$	6	7
$7^7 = 6 \bmod 11$	7	8
$7^8 = 9 \bmod 11$	8	9
$7^9 = 8 \bmod 11$	9	10

Coordinate systems XOY in subsequent examples are defined in the plane of real numbers.

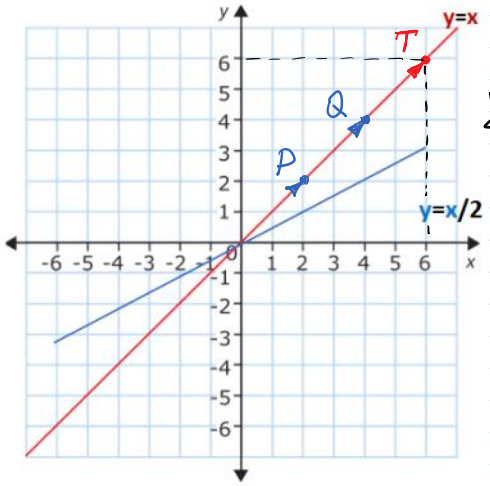


$P(x_P, y_P) = (2,4)$
$Q(x_Q, y_Q) = (4,1)$
$P+Q = (2+4, 4+1)$
$T = P+Q = (6,5)$

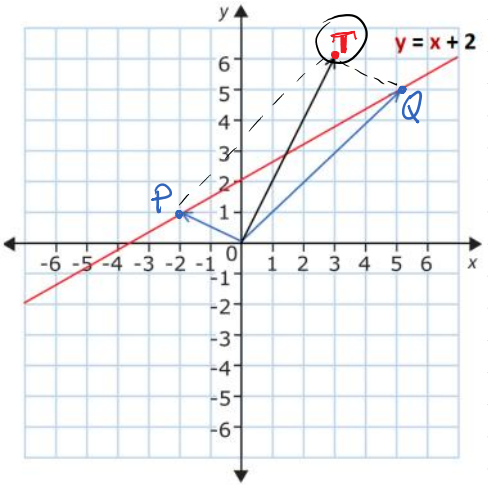
$T = P(x_P, y_P) \boxplus Q(x_Q, y_Q) =$
$= T(x_P + x_Q, y_P + y_Q) = T(x_T, y_T)$

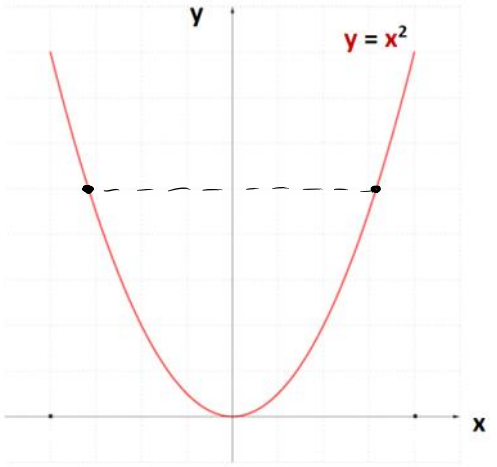
$x_T = x_P + x_Q$
$y_T = y_P + y_Q$

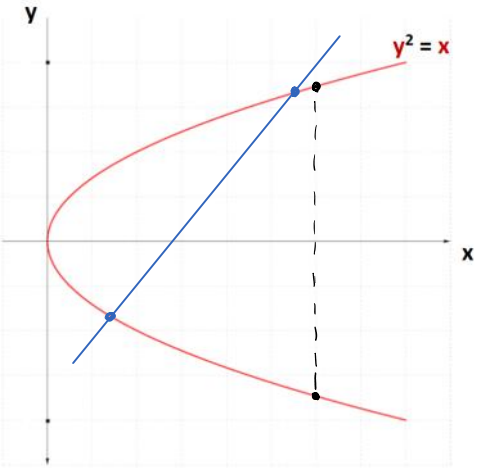
$T_2 = P + P = 2P =$

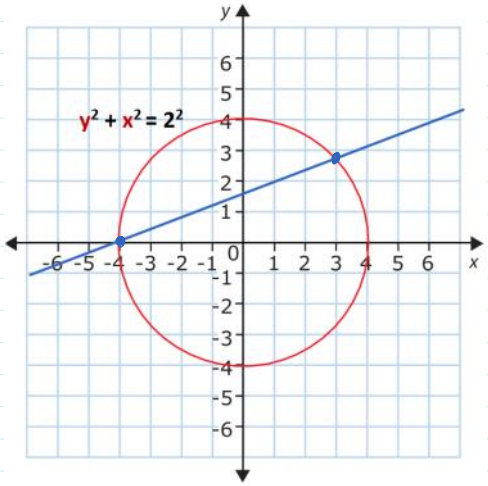


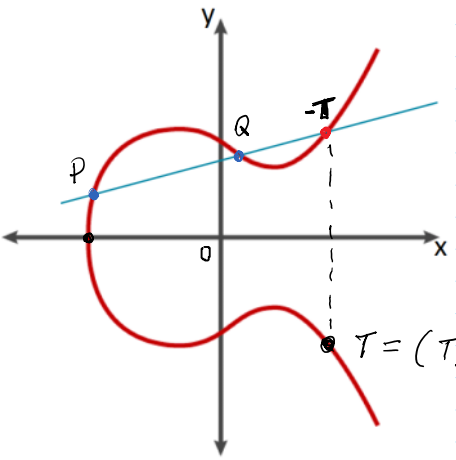
$x_T = 2+4 = 6$
$y_T = 2+4 = 6$
$|T| =$
$= \sqrt{6^2 + 6^2}$











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

Operations are performed in field F_p

$P \boxplus Q = T \in EC$

$T \boxplus (-T) = O$

$\alpha = \frac{\Delta}{x_2 - x_1}$

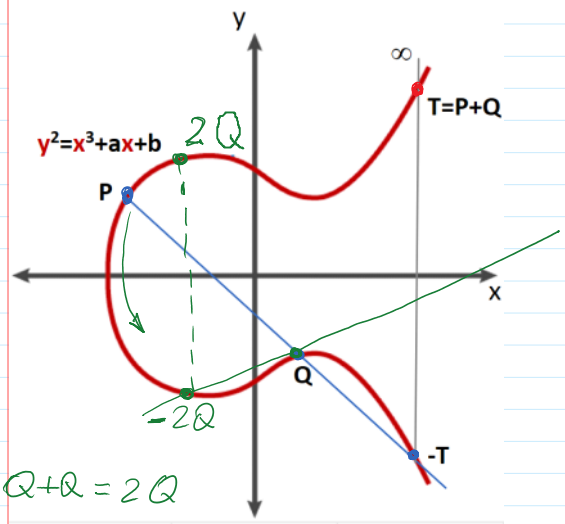
Elliptic curve ha a property that if line crosses two points, then there is a third crossing point in the curve.

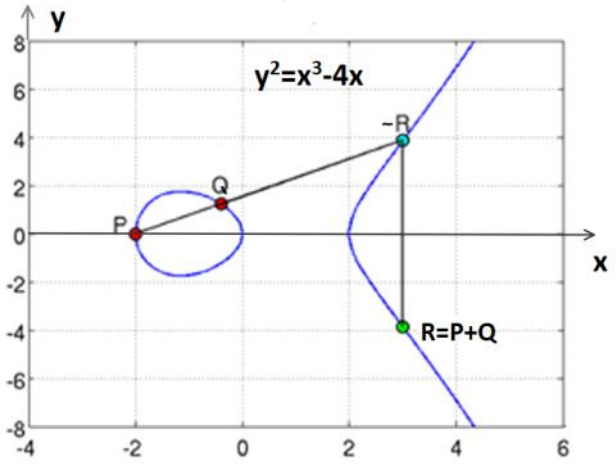
Points in the plane or plane curve we denote by the capital letters, e.g. **A**, **G**, **P**, **Q**, etc.
Numbers-scalars we denote by the lowercase letters, e.g., **a**, **g**, **x**, **y**, **z**, etc.

⊞

Addition of points P and Q in EC: **P⊞Q = T**

$P(x_P, y_P) \boxplus Q(x_Q, y_Q) = T(x_T, y_T)$





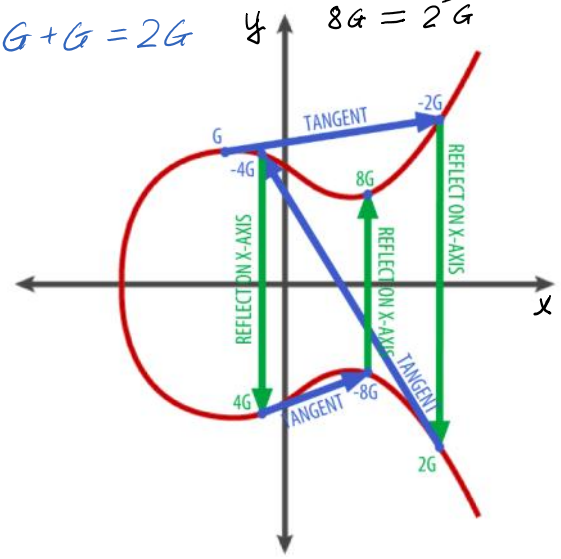
$5-5 \bmod 10 = 0$ $\quad T-T = "0" \rightarrow T+(-T) = "0" \equiv \infty$

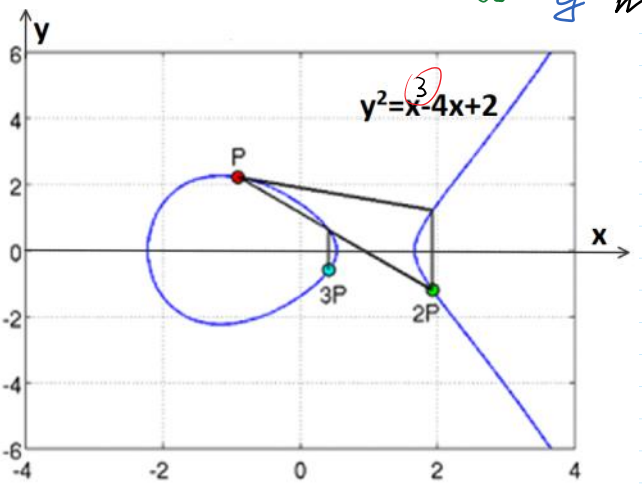
$7+0 \bmod 10 = 7$ $\quad T+\infty = T$

When z is large, $z \sim 2^{256} \rightarrow |z| = 256$ bits:

Doubling of points allows effectively compute point $A = zG$ $\quad a = g^x \bmod p$

$G+G = 2G$ $\quad 8G = 2^3 G$





ECDSA animacija

Signing and Verifying Ethereum Signatures – Yos Riady · Software Craftsman

https://medium.com/coinmonks/elliptic-curve-cryptography-6de8fc748b8b

For current cryptographic purposes, an *elliptic curve* is a plane curve over a finite field $\boldsymbol{F_p}=\{0, 1, 2, 3, \ldots, \boldsymbol{p}-1\}$, (rather than the real numbers) $\boldsymbol{p}$-is prime having 256 bit length and is of order $2^{257} \sim 10^{80}$.
Which consists of the points satisfying the equation over $\boldsymbol{F_p}$

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

along with a distinguished point at infinity, denoted by $\boldsymbol{0}$ (∞).
Finite field is an algebraic structure, where 4 algebraic operations: $+_{\bmod p}$, $-_{\bmod p}$, $\bullet_{\bmod p}$, $:_{\bmod p}$ are defined except the division by 0 excluded.

Elliptic Curve Group (ECG)
Number of points **N** of Elliptic Curve with coordinates (x, y) is an order of ECG.
Addition operation $\boxplus$ of points in ECG: let points $\boldsymbol{P}(x_P, y_P)$ and $\boldsymbol{Q}(x_Q, y_Q)$ are in EC with coordinates (x_P, y_P) and (x_Q, y_Q) then $\boldsymbol{P} \boxplus \boldsymbol{Q} = \boldsymbol{T}$ with coordinates (x_T, y_T) in EC.
Neutral element is group zero $\boldsymbol{0}$ at the infinity (∞) of [XOY] plane.
Multiplication of any EC point $\boldsymbol{G}$ by scalar in $\boldsymbol{F_p}$ $\boldsymbol{z}$: $\boldsymbol{T}=\boldsymbol{z}*\boldsymbol{G}$; $\boldsymbol{T}=\boldsymbol{G} \boxplus \boldsymbol{G} \boxplus \boldsymbol{G} \boxplus \ldots \boxplus \boldsymbol{G}$; $\boldsymbol{z}$–times.
Generator–Base Point $\boldsymbol{G}$: ECG={ $\boldsymbol{i}*\boldsymbol{G}$; $\boldsymbol{i}$=1,2,…,**N**}; $\boldsymbol{N}*\boldsymbol{G}$=0 and $\boldsymbol{q}*\boldsymbol{G} \neq 0$ if $\boldsymbol{q}<\boldsymbol{N}$.

Because this curve is defined over a finite field of prime order instead of over the real numbers, it looks like a pattern of dots scattered in two dimensions, which makes it difficult to visualize. However, the math is identical to that of an elliptic curve over real numbers. As an example, Elliptic curve cryptography: visualizing an elliptic curve over F(p), with p=17 shows the same elliptic curve over a much smaller finite field of prime order 17, showing a pattern of dots on a grid. The secp256k1 bitcoin elliptic curve can be thought of as a much more complex pattern of dots on a unfathomably large grid.

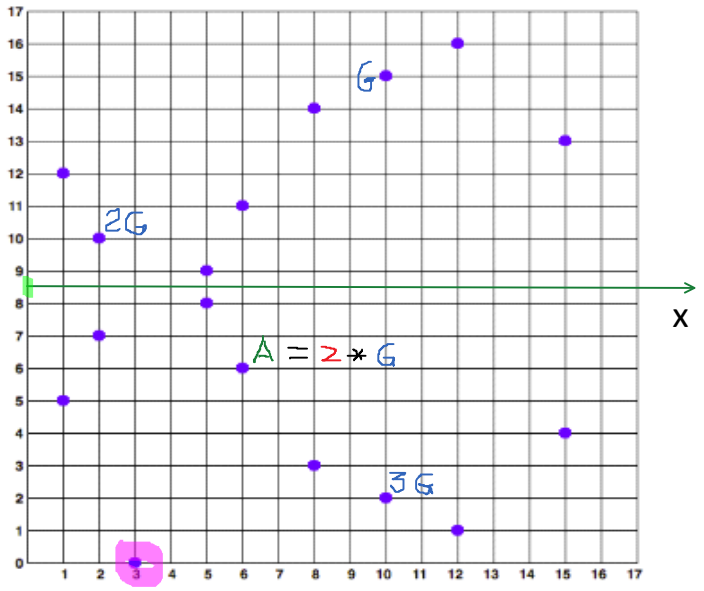


Figure 3. Elliptic curve cryptography: visualizing an elliptic curve over F(p), with p=17

$F_p = Z_p = \{0, 1, 2, 3, \ldots, p-1\}$
When p = 17, then
$F_{17} = Z_{17} = \{0, 1, 2, 3, \ldots, 16\}$

$DEF: g^x \bmod p = a;$

$DEF(x+z) = DEF(x) \cdot DEF(y) \bmod p$

$g^{x+y} \bmod p = g^x \cdot g^y \bmod p$

$ECDEF: z * G = A = (x_A, y_A);$

$ECMul(z * G) = z * G = G \boxplus G \boxplus G \boxplus \ldots, \boxplus G$ (z-times)

$ECDEF((x+y) * G) = ECDEF(x * G) \boxplus ECDEF(y * G) = T$

$P \quad \boxplus \quad Q \quad = T$

EC-Property: $(x_1+x_2) * \mathbf{G} = x_1 * \mathbf{G} \boxplus x_2 * \mathbf{G} =$

$= \mathbf{P} \boxplus \mathbf{Q}$

z - is a random number of 256 bit length and is a private key **PrK** in Elliptic Curve Cryptography - ECC

A = ***z*** * ***G*** - is apoint of Elliptic Curve (EC) and is a public key **PuK** in Elliptic Curve Cryptography - ECC

Elliptic Curve Cryptosystem - ECC

ElGamal Cryptosystem (CS) Schnorr Cryptosystem	Elliptic Curve Cryptosystem (CS)
PP=(strongprime ***p***, generator ***g***);	**PP**=(EC **secp256k1**; BasePoint-Generator ***G***; prime ***p***; param. ***a***, ***b***);

p=255996887; g=22;	Parameters a, b defines EC equation $y^2=x^3+ax+b \bmod p$ over F_p.
PrK=x; >> x=randi(p-1);	PrK_{ECC}=z; >> z=randi(p-1);
PuK=$a=g^x \bmod p$.	$PuK_{ECC}=A=z*G$.
Alice A: x=1975596; a=210649132;	Alice A: z=…..; $A=(x_A, y_A)$;

Till this place

Let us consider abstract EC defined in XOY and expressed by the equation:

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

EC points are computed by choosing coordinate x and computing coordinate y^2.
To compute coordinate y it is needed to extract root square of y^2.

$$y = \pm\sqrt{y^2} \bmod p.$$

Notice that from y^2 we obtain 2 points in EC, namely with coordinates y and $-y$ no matter computations are performed with integers **mod** p or with real numbers.
Notice also that since EC is symmetric with respect to x-axis, the points y and $-y$ are symmetric in EC.
Since all arithmetic operations are computed **mod** p then according to the definition of negative points in F_p points y and $-y$ must satisfy the condition

$$y + (-y) = 0 \bmod p.$$

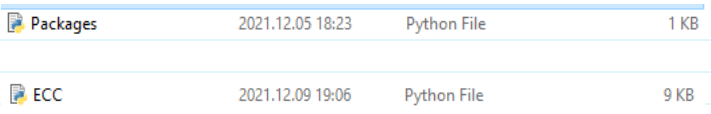
Then evidently

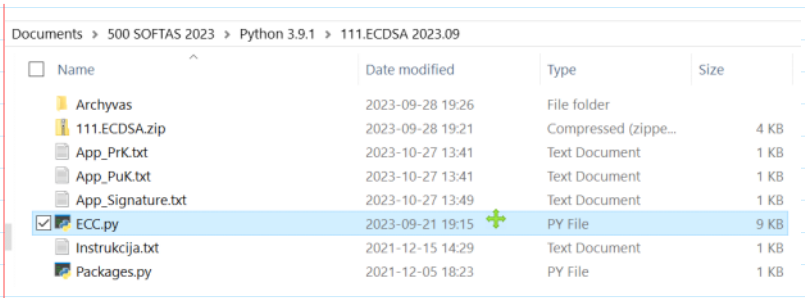
$$y^2 = (-y)^2 \bmod p.$$

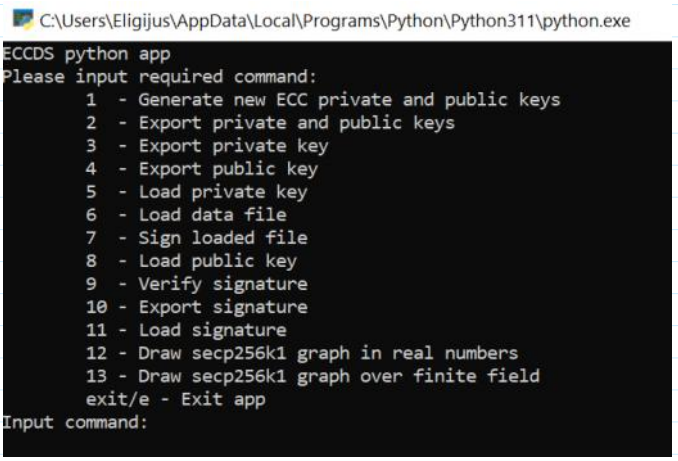
For example:
-2 mod 11 = 9
2^2 mod 11 = 4 & 9^2 mod 11 = 4
>> mod(9^2,11)
ans = 4

$-2 \bmod 11 = 9 \bmod 11$
$(+2 + (-2)) \bmod 11 = (2+9) \bmod 11 = 11 \bmod 11 = 0$

ECDSA using **secp256k1** software:
1.Install Python 3.9.1.
2.Launch script Packages for joining a libraries.
3.Launch file ECC.
4.If window is escaping, then open hiden windows in icon near the Start icon.

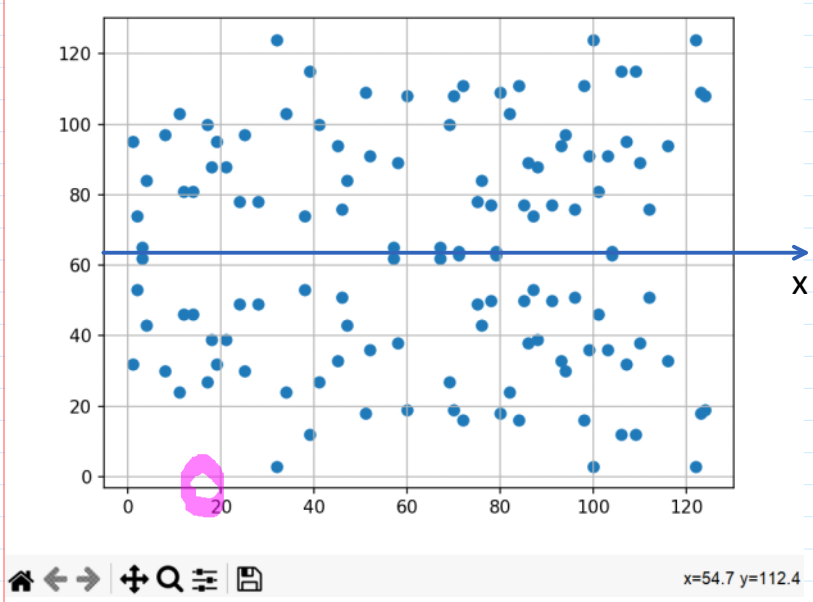




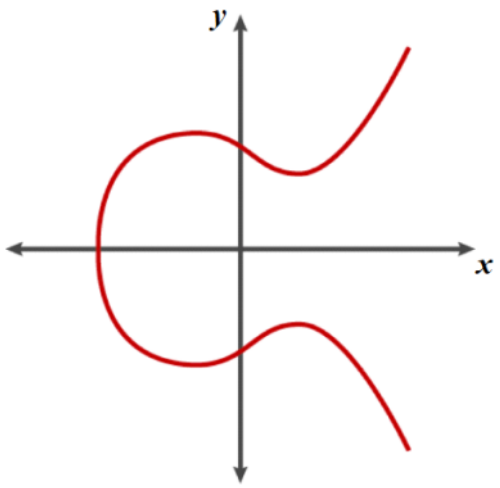


PrK: has 256 bit length or 64 hexadecimal digits:
0x4f30dcb4235b9d52d0cd9aef9eb11b9c75b07ef25fc2249491b071f48803e5d3

PuK: has 512 bit length or 128 hexadecimal digits:
0x0beffa8fb3b2665089333456bc98b3c003181d7b7c23750c1dde4fa070608442638d7391846bda26aad95accba5810fc326611e804506f95e93ffa006d15b0f0



The positive and negative coordinates ***y*** and ***-y*** in EC in the real numbers plane XOY are presented in Fig.
The positive and negative numbers for ***p***=11 are presented in table .



y mod 11			(-***y***) mod 11
1	odd	even	-1=10
2	even	odd	-2=9
3	odd	even	-3=8
4	even	odd	-4=7
5	odd	even	-5=6
6	even	odd	-6=5
7	odd	even	-7=4
8	even	odd	-8=3
9	odd	even	-9=2
10	even	odd	-10=1

Notice that performing operations **mod** ***p*** if ***y*** is odd then -***y*** is even and vice versa.
This property allows us to reduce bit representation of $\mathbf{PuK_{ECC}}=A=z*G=(x_A, y_A)$;
In normal representation of $\mathbf{PuK_{ECC}}$ it is needed to store 2 coordinates (x_A, y_A) every of them having 256 bits.
For $\mathbf{PuK_{ECC}}$ it is required to assign 512 bits in total.
Instead of that we can store only x_A coordinate with an additional information either coordinate y_A is odd or even.
The even coordinate y_A is encoded by prefix 02 and odd coordinate y_A is encoded by prefix 03.
It is a compressed form of $\mathbf{PuK_{ECC}}$.
If $\mathbf{PuK_{ECC}}$ is presented in uncompressed form than it is encoded by prefix 04.
Imagine, for example, that having generator G we are computing $\mathbf{PuK_{ECC}}=A=z*G=(x_A, y_A)$ when z=8.
Please ignore that after this explanation since it is crasy to use such a small z. It is a gift for adversary
To provide a search procedure.
Then $\mathbf{PuK_{ECC}}$ is represented by point $8G$ as depicted in Fig. So we obtain a concrete point in EC being either even or odd.
The coordinate y_A of this point can be computed by having only coordinate x_A using formulas presented above and having prefix either 02 or 03.