Elliptic Curves

- Used for public key cryptography
  - Better than RSA
  - 256 bit EC key has same security as 3072 bit RSA key
  - More complex
  - Relies on Trapdoor Function
  - RSA relies on factoring large numbers, DH on discrete log problem but ECC is diff.

~~Trelis slo~~

Weierstrass Eqn.

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

- We take operation (point addition) that takes 2 points on same curve & produces third point on curve. Taking set of points on elliptic curve defines Abelian group operation.

- We can understand scalar multiplication of point as repeated point addition of same point.

Date:

$$Q = 2|P| = P + P$$

↳ Trapdoor function that ECC relies on.

↳ Finding 'n' such that

$Q = [n]P$, Given Q & P.

Point Addition (P+Q)

↳ Take elliptic curve & mark two points P, Q along curve with their x|y coordinates.

↳ Draw straight line that passes through both points.

↳ Now continue until line intersects your curve a third time. Mark this new intersection 'R'.

↳ Take R & reflect it along y direction to produce $R' = R(x, -y)$

↳ Result of point addition is

$$P + Q = R'$$

What if we want to add same points together P+P?

Date:

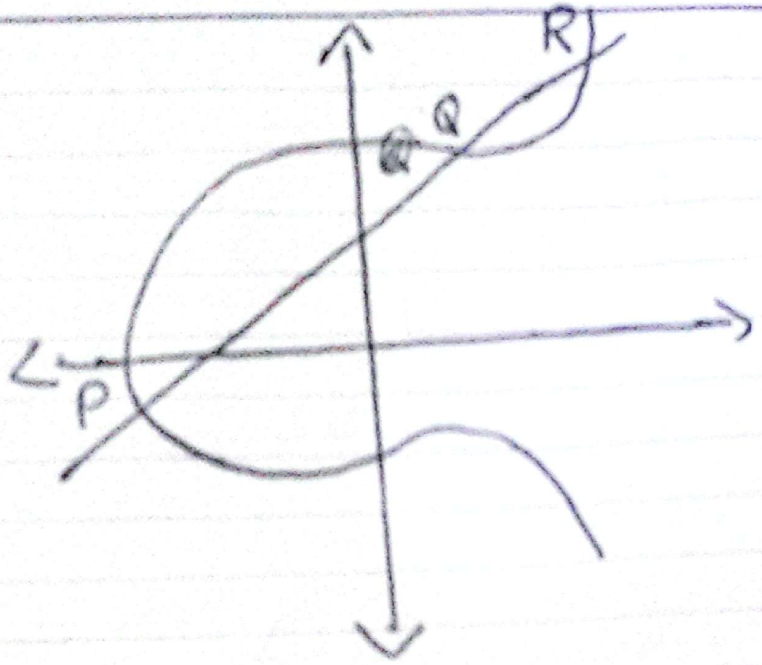
Su Mo Tu We Th Fr Sa

↳ Pick unique line by calculating tangent line to curve at point.
↳ Calculate tangent line at point P.
↳ continue until it intersects with curve point R. Reflect this point as before

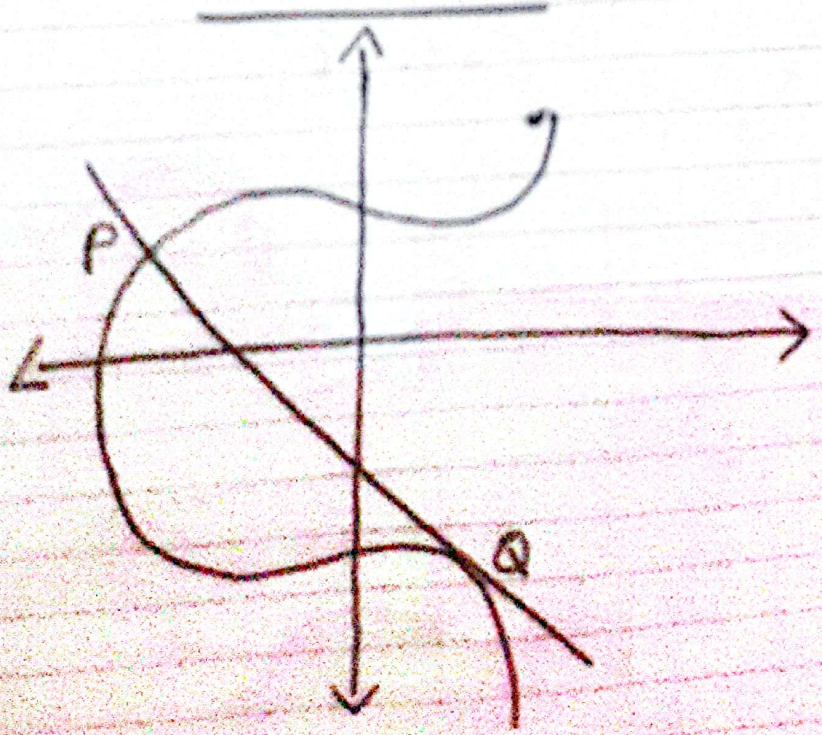
$$P+P = R' = R(x, -y)$$

If no third intersection?
↳ Say line intersects with point O which is single point located at end of every vertical line at infinity.
↳ Point Addition for elliptic curve is defined in 2D space, with additional point located to infinity

Date



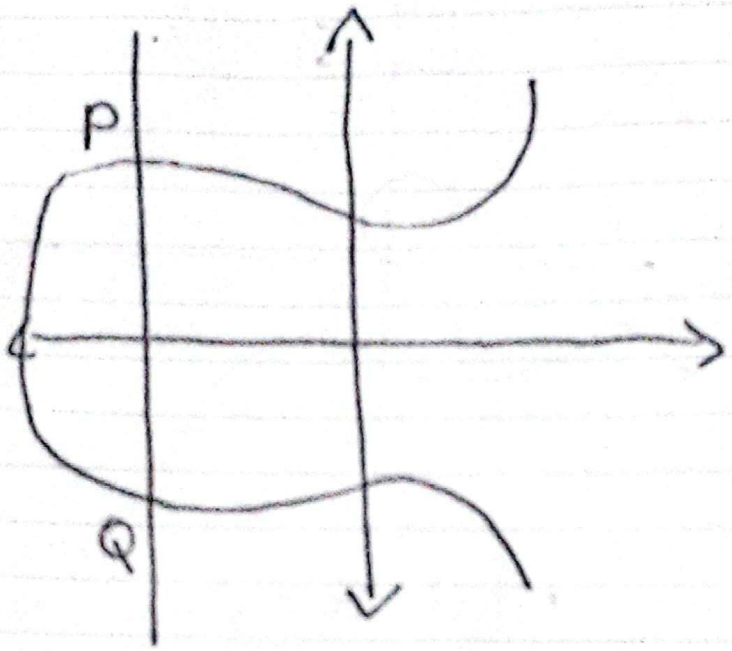
P + Q + R = 0



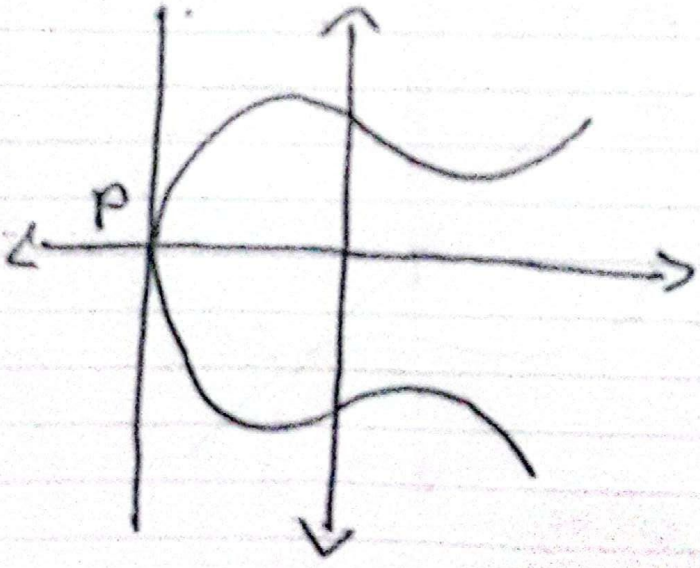
P + Q + Q = 0

Date:

Su Mo Tu We Th Fr Sa



P + Q + O = O



P + P + O = O

Date: Su Mo Tu We Th Fr Sa

Point 'O' acts as infinity operator of group: $P+O=P$ & $P+(-P)=0$

Elliptic curve

↳ An Elliptic curve E is set of solution to Wierstrass eqn.

$$E: Y^2 = X^3 + aX + b$$

together with point at infinty O. The constant a|b must satisfy relationship

$$4a^3 + 27b^2 \neq 0$$

to ensure there are no singularities on curve

Let E be E.C, point Addition has following properties

① $P+O = O+P = P$
② $P+(-P) = 0$
③ $(P+Q)+R = P+(Q+R)$
④ $P+Q = Q+P$

In ECC, we study elliptic curves over finite field F_p. This means we look at curve modulo the characteristic 'p' & an elliptic

Date:

curve will no longer be a curve, but collection of points whose x, y coordinates are integers in F_p.

Point Negation

↳ Consider E.C of form

$E: y^2 = x^3 + ax + b$, which satisfies $a, b \in F_p$ & $4a^3 + 27b^2 \neq 0$.

↳ We no longer think of E.C as geometric object, but rather a set of points defined by

$E(F_p) = \{(x, y) : x, y \in F_p$ satisfying $y^2 = x^3 + ax + b\} \cup 0$.

Chall

$E: y^2 = x^3 + 497x + 7768 \mod 9739$

Using curve above, & point $P(8025, 6936)$, find $Q(x, y)$ such that $P + Q = 0$.

Soln

Date:

Su Mo Tu We Th Fr Sa

10

p = 9739
x = 8025
y = 6936.

Now,

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

Consider y is -y

Since

$(-y)^2 = (-1)^2 \cdot y^2 = y^2$

Eqn becomes

$(-y)^2 = y^2 \equiv x^3 + ax + b \pmod p$

In Modulus Arithmetic

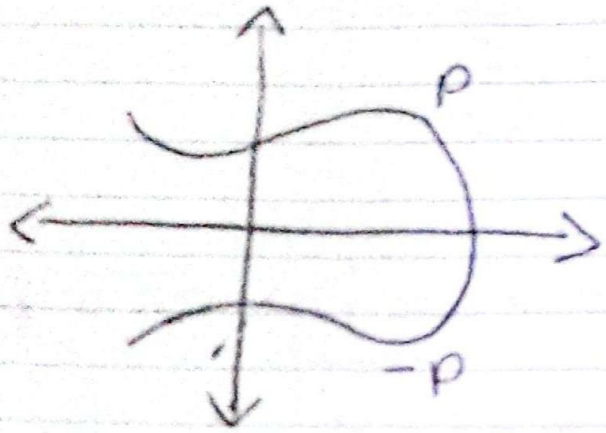
$-y \bmod p = p - y$

$-Qy = +y \bmod p$

In elliptic curve:

↳ The opposite of point 'P' = (x, y) is another point.

Q = (x, -y)



2n F_p finite field.

$p = 9739$

a) To negate numbers 'y' modulo p, we do,

$$\boxed{-y \bmod p = p - y}$$

↓ Given,

$P = (8045, 6936)$

x is same 8045

To negate y modulo 9739.

$$-Q_y = -6936 \bmod 9739$$
$$= 9739 - 6936$$
$$= 2803$$

$Q = (8045, 2803)$ //

Point Addition

↳ Efficient way to calculate two Points Algo

↓

P + Q

↳ ① If P = 0, then P + Q = Q.

② Otherwise, if Q = 0, then P + Q = P

③ Otherwise, write $P = (x_1, y_1)$ & $Q = (x_2, y_2)$

④ If $x_1 = x_2$ & $y_1 = -y_2$, then P + Q = 0

⑤ Otherwise,

↳ If $P \neq Q$: $\lambda = \frac{(y_2 - y_1)}{(x_2 - x_1)}$

↳ If $P = Q$: $\lambda = \frac{(3x_1^2 + a)}{2y_1}$

⑥ $x_3 = \lambda^2 - x_1 - x_2$

⑦ $y_3 = \lambda(x_1 - x_3) - y_1$

⑧ $\boxed{P + Q = (x_3, y_3)}$

In above algo, we r working with finite field, so above calc should be done modulo p, & we donot divide instead we multiply by modular inverse of numbers

Eg: $5^{-1} \equiv 9 \bmod 11$

So dividing by 5 modulo 11 is same as ~~apptym~~ multiplying by 9 modulo 11.

#Chall (work with following EC & prime

$E: y^2 = x^3 + 497x + 1768 \bmod 9739$

using above curve & points

P(493, 5564), Q = (1539, 4742), R = (4403, 5202), find the point S(x, y) = P + P + Q + R by implementing above algorithm.

Sol:

Since $P \neq Q$.

Date:

ECDSA = Elliptic Curve Digital Signature Algorithm

$$\lambda = (y_2 - y_1)(x_2 - x_1)^{-1} \bmod p$$

Case 2 $P = Q$

$$\lambda = (3x_1^2 + a)(2y_1)^{-1} \bmod p$$

$$\therefore x_3 = \lambda^2 - x_1 - x_2 \bmod p$$

$$\therefore y_3 = \lambda(x_1 - x_3) - y_1 \bmod p$$

Here in our case,

$a = 497$

$b = 1768$.

$$S(x, y) = P + P + Q + R$$

$$A = P + P$$
$$B = Q + B$$
$$C = B + R$$

Scalar Multiplication

→ Scalar multiplication of two points is defined by repeated addition.

$3[P] = P + P + P$

Date:

Su Mo Tu We Th Fr Sa

Double & Add Algorithm for Scalar Multiplication.

Input: $P \in E(F_p)$ & an integer $n > 0$.
Output: $Q = nP \in E(F_p)$.

1) Set $Q = P$ and $R = O$.
2) Loop while $n > 0$.
   - ↳ If $n \equiv 1 \bmod 2$, set $R = R + Q$
   - ↳ set $Q = [2]Q$ & $n = [n/2]$.
   - ↳ If $n > 0$, continue with loop at step 2
3) Return the point R, which equals $[n]P$.

Chall of Scalar Multiplication

$E: Y^2 = x^3 + 497x + 1768 \bmod 9739$.

Using above curve, & points $P = (2339, 2213)$, And point $Q = (x, y) = [7863]P$.

Sol:

$a = 497$
$b = 1768$

Date

n = 7863,

ECDLP (Elliptic curve Discrete Log Problem) is finding integer n such that $Q = [n]P$

① A & B agree on curve E, a prime 'p' & generator point 'G' which generates a subgroup $H = \langle G \rangle$ of prime order q.

② A generates ~~rand~~ secret random int n_A & calculates $Q_A = [n_A]G$.

③ B generates secret random int n_B & calculates $Q_B = [n_B]G$.

④ A sends B Q_A & B sends A Q_B. Due to ECDLP, onlooker Eve is unable to calculate $n_{A/B}$

⑤ A then calculates $[n_A]Q_B$, & B calculates $[n_B]Q_A$

⑥ Due to scalar Multiplication property

$$S = [n_A]Q_B = [n_B]Q_A$$

⑦ A & B use S as their shared secret.

More than a College

Date:

Su Mo Tu We Th Fr Sa

#chall

using curve, prime & generator,

$E: y^2 \equiv x^3 + 497x + 1768 \pmod{9739}$

$G = (1804, 5368)$.

Calculate shared secret after A sends ya $Q_A = (815, 3190)$ with your secret int $n_B = 1829$.

Sol

A ~~B~~ A

$S = [n_B] Q_B$

n_B we need Q_B

$S = Q_B = [n_B] Q_A$ #

We can make sharing of secret in ECC more efficient by not sharing all parameters

- Given either value of 'y'
- Each x has two possible y values.

Date:

Su Mo Tu We Th Fr Sa

###

↳ You can just send x+1 bit instead of full (x, y).

Chall

↳ we have used prime $p \equiv 3 \bmod 4$, which will help you find y & y^2.

E: $y^2 = x^3 + 497x + 1768 \bmod 9739$,
G: (1804, 5368)

Calculate shared secret after A sends you $x(Q_A) = 4726$, with your secret integer $n_B = 6534$.

↳ Using, IV & Encrypted flag. We can confirm it's CBC encryption. So from IV we can just ~~write~~ get the flag.

So,

We need to find:

$S = Q_A(n_B)$ using only

$x(Q_A) = 4726$,

Date: Su Mo Tu We Th Fr Sa

we can recover now y

or, $y^2 = x^3 + 497x + 1768 \bmod 9739$.

put $x = 4726$,

Get y^2

Since $p \equiv 3 \pmod 4$, we got shortcut formula to compute square root modulo p

$$y = \pm (y^2)^{(p+1)/4} \bmod p$$

which will give 2 values of y.

once you have y,

$Q_A = (4726, 0)$

Just find s#

Double & Add (naive scalar multiplication) ~~has~~ is vulnerable to side channel Attacks.

Date:

LadderLeak Attack

↳ SideChannel Attack for ECDSA

↳ To protect against this lot of work has been done to make scalar Multiplication of points on E.C to run on constant time

↳ This is done on or based by Montogomery Ladder.

Montgomery's binary algo in Group E(Fp)

Input: $P \in E(F_p)$ & n-bit integers.

$K = \sum 2^i k_i$ where $k_{n-1} = 1$

Output: $[K]P \in E(F_p)$

① Set (R_0, R_1) to $(P, [2]P)$

② For $i = n-2$ down to 0 do.

a ↳ If $k_i = 0$ then.

Set (R_0, R_1) to $([2]R_0, R_1 + R_0)$

else.

Set (R_0, R_1) to $(R_0 + R_1, 2[R_1])$

③ Return R_0

More than a College

Affine coordinate represent point in elliptic curve as

$$P = (x, y)$$

#Chall.

$E: y^2 = x^3 + 486662x^2 + x \bmod 2^{255} - 19$

Using above curve, & generator point with $6 \cdot x = 9$, find x-coordinate of point.

$Q = [0x1337c0decafe] G$

by above algorithm

Sol:

The curve is in form of montogomery than wierstrass

So,

#Addition formula for Montogomery curve (Affine)

Input: $P, Q \in E(F_p)$ with $P \neq Q$
Output: $R = (P + Q) \in E(F_p)$

$(x_1, y_1), (x_2, y_2) = P, Q$

$$\alpha = \frac{(y_2 - y_1)}{(x_2 - x_1)}$$

Date:

Su Mo Tu We Th Fr Sa

$$x_3 = B\alpha^2 - A - x_1 - x_2$$
$$y_3 = \alpha(x_1 - x_3) - y_1$$
$$R = (x_3, y_3)$$

Doubling Formula for Montgomery Curve (Affine)

Input: $P \in E(F_p)$
Output: $R = [2]P \in E(F_p)$

$$(x_1, y_1) = P$$
$$\alpha = \frac{3x_1^2 + 2Ax_1 + 1}{2By_1}$$
$$x_3 = B\alpha^2 - A - 2x_1$$
$$y_3 = \alpha(x_1 - x_3) - y_1$$
$$R = (x_3, y_3)$$

Date:

ECDLP / ECDH.

~~ECDH~~ chall

#Smooth criminal (chall)

In source.py

4) ① A picks secret n (random int)
② Alice computes $Q_A = [n]G$.
③ B has Q_B
④ shared secret = $[n]Q_B$

we have,

In output.txt

Q_A (A pub key), Q_B (B pub key), G (generator point), curve parameters (p, a, b), Encrypted flag & IV.

we need,

Alice secret (n) so we can, get $S = [n]Q_B$ → Decrypt flag.

So find n such that

$$[n]G = Q_A$$

→ Every E.C has group order, the total number of points on curve. Call it 'N'.

In code, there is ECDH setup.

curve order notation = # E.

Date:

Note: If we are given p, a, b in crypto chall always check is curve order.

In code we are given,

```
#Define curve
p = 3107 .... 03
a = 2
b = 3
```

If we find is curve order which means total number of points on curve over finite field Fp, including point at infinity. (# E)
i.e

E = (number of valid (x, y) pairs) + 1.

In sage math find its order.

E.order()

① If order is prime → good.

② If order is smooth meaning all prime factors of order of E are below or small we can do Pohling-Hellman Attack which is in our case.

Date:

Su Mo Tu We Th Fr Sa

Pohling-Hellman

$$\#G = n = \prod_i p_i^{e_i}$$

↳ works when group order is smooth meaning it factors into small or median primes.

↳ ① we wan n such that

$$A = nG$$

② Instead of solving Discrete log in full group of order #G, do it modulo each prime power $p_i^{e_i}$.

$$n \bmod p_i^{e_i}$$

③ once you know n mod $p_i^{e_i}$ for all factors, combine them using CRT.

$$n \bmod \#E$$

to get full discrete log n.

Date:

Curveball (Chall)

↳ CVE 2020 0601

↳ ~~EC~~

Normally ECC : $Q = d * G$

where d is secret.

In ECC ~~cert~~ ~~attr~~, it should check all,

1) $G' = G_{standard}$

2) $Q' = Q_{trusted}$

3) curve name is trusted

↳ In code:

g = Point(x, y, Curve=P256)

↳ It doesn't verify (x, y)

↳ In ECC public key is just repeated point addition:

$Q = G + G + G + \ldots + G = d \cdot G$

↳ We can supply our own G.

Date:

Su Mo Tu We Th Fr Sa

↳ No we need to find point "G'" & int "d' such that.

$$d' \cdot G' = Q\text{-bing}$$

Note: ~~Code needs~~ To we need to make seever think we are www.bing.com to get flag.

To find it, we can pick any $d' = K$ ~~to~~. then work backward

$$d' \times G' = Q\text{-bing}$$
$$2 \times G' = Q\text{-bing}$$
$$G' = \frac{Q\text{-bing}}{2}$$
$$G' = 2^{-1} \times Q\text{bing}$$

Date

Prosign 3 (chall).

- uses ECDSA
- we have two option:
  1. sign-time → signs time block in m:s.
  2. verify → verifies signature.
     - If valid & msg = unlock we get flag
- we need valid signature for 'unlock'

In code originally n = g.order()
but inside sign-time

```
m,n = int(now.strftime),
      int(now.strftime)
```

- So now n = current seconds
- Then in code does randrange(1,n)
- It means ECDSA nonce 'k' is between [1, seconds] = [1, 59]
- <u>ECDSA Eqn</u>

$$r = (kG)_x \bmod n.$$

$$s = k^{-1}(h + rd) \bmod n$$

Date:

where
d = private key.
h = ~~hash~~ hash
k = nonce
G = generator point
n = order of G

Rearrange.

$$s = k^{-1}(h + \gamma d) \bmod n.$$

Multiply both by k

$$sk = h + \gamma d \bmod n.$$

$$\gamma d = sk - h \bmod n.$$

Multiply both by γ^{-1}

$$\boxed{d = (sk - h)\gamma^{-1} \bmod n}$$

Like that we can decrypt private key.

Date

Su Mo Tu We Th Fr Sa

Steps

1) Call sign-time once
2) get msg, r, s.
3) Compute $h = SHA1(msg)$
4) for k in 1 .. 60:
   compute candidate private key
   $d = (sk - h) r^{-1} \bmod n$.
5) verify candidate.
   a) check if $d * G = pubkey$
   b) If yes → we got private key.
6) Sign "unlock" using that private key.
7) send "verify".
8) Get flag.

Date

Moving problems (chall)

- we have generator point G
  - A public key $P_1 = n_a G$
  - B public key $P_2 = n_b G$
  - AES enc flag

$$S = n_a n_B G$$

key = SHA1 (x-coordinate of S) [:16]

- we need to recover A private key n_a

$$P_1 = n_a G$$

- Vuln is curve has weak embedding degree which means small factor of order of group.
- Embedding degree 'k' is smallest integer such that

$$l \mid (p^k - 1)$$

Date:

Su Mo Tu We Th Fr Sa

where l = order of generator point (Ord(G)).

```
k = 1
while (p^k - 1) % l != 0:
    k += 1
print("Embedding degree: ", k)
```

↳ we get E.D as 2 which is small & we can perform MOV Attack.

MOV Attack

↳ Menezes-Okamoto-Vanstone

↳ Reduces

ECDLP on E(Fp)

to:

$DLP = F_{p^k}^*$

using weil pairing.

Date: Su Mo Tu We Th Fr Sa

Weil pairing.

$e_l : E[l] \times [l] = \mu_l \subset F_{p^k}$

Steps of chall Moving problems.

1) Define curve p, a, b, O, G, P1, P2
2) Find order of G.(l)
3) Find Embedding Degree
4) Move to Extension field:
   Since k=2,
   F_{p^2}
   $F_k \cdot \langle 2 \rangle = G \subset F(p^k)$
5) Now pairing points to
   $F^* p^2$

Date:

5) Find Independent Point T such that.

i) Order = l
ii) Linearly ~~deperd~~ independent from G. Because Weil pairing must be nontrivial. (non-significant)

6) Apply Weil Pairing

$$\alpha = e(G, T)$$
$$\beta = e(P, T)$$

7) Use Bilinearity.
b) we know.

$$P = n_a G$$

So,

$$e(P, T) = e(n_a G, T)$$

By Bilinearity

$$= e(G, T)^{n_a}$$

So, $\boxed{\beta = \alpha^{n_a}.}$

This is finite DLP

8) Solve finite Field DLP

Date

Get $\boxed{n_a = \log_\alpha(\beta)}$

we get n_a

Now just as before #

$\boxed{\begin{aligned} S &= n_a P_z \\ key &= SHA1(S_x)[:16] \#\end{aligned}}$