CRYPTOGRAPHY BASICS.

NAME: SNEHA MAGANAHALLI RAJENDRANATH.

ROLL NO: CS21M522

ASSIGNMENT NO: 3

What is RSA?

- RSA is a public-key cryptosystem used for secure data transmissio
- It will use public-key for encryption
- It will use private key for decryption

Example:

The below example shows how RSA works for small numbers. The program should work in the same way for big numbers

let

$\boxed{\begin{matrix} p = 5 \\ q = 13 \end{matrix}}$ } p, q are two large prime numbers. Therefore they are relatively prime to each other.

$n = p \times q$

$= 5 \times 13$

$\boxed{n = 65}$ ---- This is the data range. i.e; one can encrypt the numbers from 0 - 64 i.e (0 - (n-1))

$\phi(n) = (p-1)(q-1)$

$= (5-1)(13-1)$

$= 4 \times 12$

$\boxed{\phi(n) = 48}$

Choose public key.

let

$\boxed{e = 5}$ ... let public key be 5.

calculate private key d.

$d = e^{-1} \bmod \phi(n)$

.... d is inverse mod $\phi(n)$

$d = 5^{-1} \bmod 48$

$\boxed{d = 29}$ ——

$5 \times 29 = 145$
$145 \bmod 48 = 1$.
∴ 29 is inverse of 5 mod 48

$d_p = d \bmod (p-1)$
$d_q = d \bmod (q-1)$

∴ $d_p = 29 \bmod 4$
$\boxed{d_p = 1}$

∴ $d_q = 29 \bmod 12$
$\boxed{d_q = 5}$

Encryption side :-

plain text : 7.

Encryption formula : $y = x^e \bmod n$

$= y = 7^5 \bmod 65$

$= 37$

Send 37 as the ciphr text.

Decryption side :-

$$x_p = y^{d_p} \bmod p$$

$$x_p = 37^1 \bmod 5$$

$$\boxed{x_p = 2}$$

$$x_q = y^{d_q} \bmod q$$

$$= 37^5 \bmod 13$$

$$\boxed{x_q = 7}$$

$$X = (R \times X$$

Inverse of $q \bmod p$:

$$inv\_q\_mod\_p = 13^{-1} \bmod 5 = \underline{2}$$

$13 \times 2 = 26$

$26 \bmod 5$

$= 1$

$\therefore$ 26 is inverse of 13 mod 5

Inverse of $p \bmod q$:

$$inv\_p\_mod\_q = 5^{-1} \bmod 13 = \underline{8}$$

$5 \times 8 = 40$

$40 \bmod 13$

$= 1$

$\therefore$ 8 is inverse of 5 mod 13.

$$\begin{pmatrix} X \\ \text{Plain text} \end{pmatrix} = \left(\left[x_p \times V \times inv\_v\_mod\_p \right] + \left[x_q \times p \times inv\_p\_mod\_q \right] \right) \bmod n$$

$$X = (2 \times 13 \times 2 + 7 \times 5 \times 8) \bmod 65$$

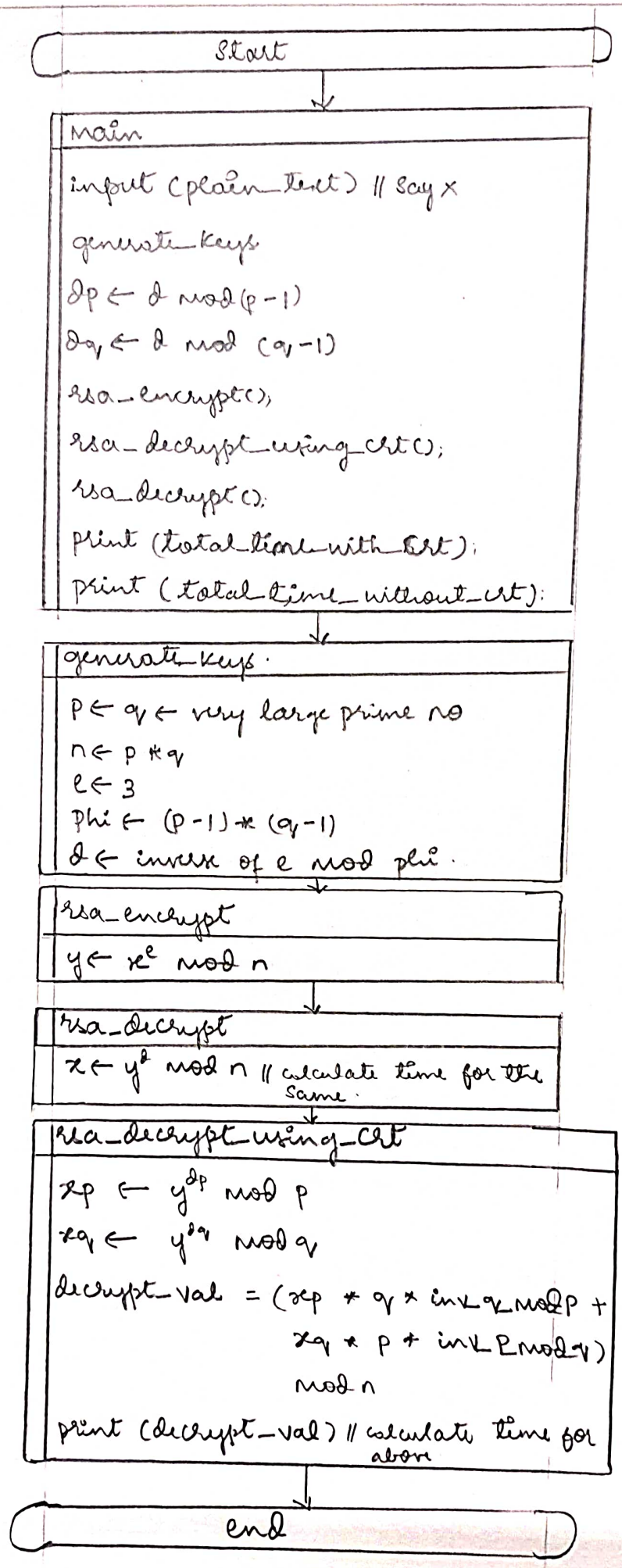
$$X = (52 + 280) \bmod 65$$

$$X = 332 \bmod 65$$

$$\boxed{X = 7}$$

$\therefore$ Decryption is successfull.

Flow chart


Start
main
input (plain_text) // say x
generate_keys
dp ← d mod (p-1)
dq ← d mod (q-1)
rsa_encrypt();
rsa_decrypt_using_crt();
rsa_decrypt();
print (total_time_with_crt);
print (total_time_without_crt);
generate_keys:
p ← q ← very large prime no
n ← p * q
e ← 3
phi ← (p-1) * (q-1)
d ← inverse of e mod phi.
rsa_encrypt
y ← x^e mod n
rsa_decrypt
x ← y^d mod n // calculate time for the same.
rsa_decrypt_using_crt
xp ← y^dp mod p
xq ← y^dq mod q
decrypt_val = (xp * q * invk_q_mod_p + xq * p * invk_p_mod_q) mod n
print (decrypt_val) // calculate time for above
end

Algorithm:

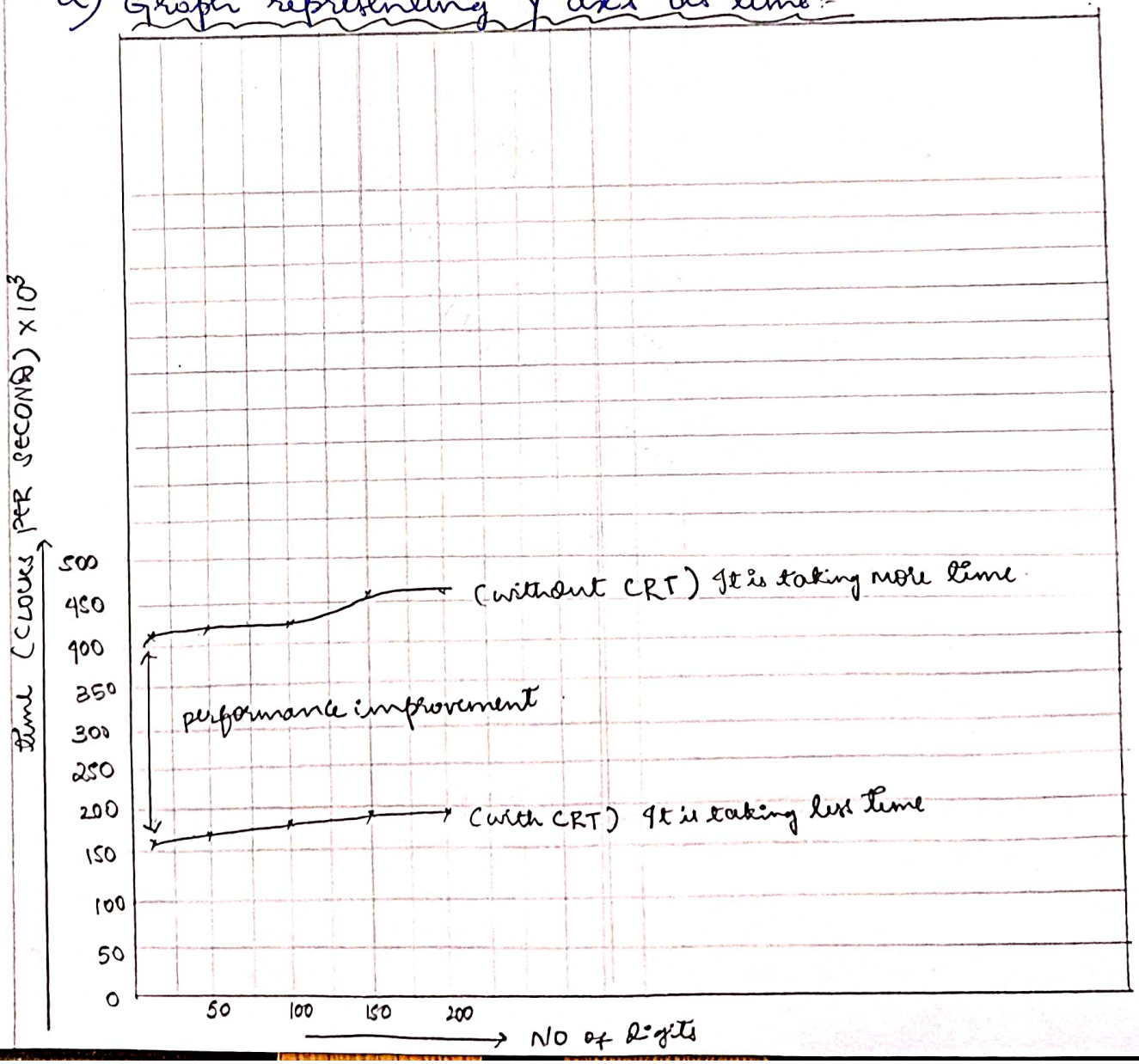
```
Void generate_keys()
{
    p ← large_prime_no;
    q ← large_prime_no;
    n ← p * q
    e ← 3
    phi ← (p-1) * (q-1)
    d ← inverse of e mod phi
}

Void rsa_encrypt()
{
    y ← x^e mod n
}

Void rsa_decrypt()
{   start ← clock();
    x ← y^d mod n
    end ← clock();
    total_t ← (end - start) / CLOCKS_PER_SECOND;
}

Void rsa_decrypt_using_crt()
{
    start ← clock();
    xp ← y^dp mod p
    xq ← y^dq mod q
    end ← clock
    total_with_crt ← (end - start) / CLOCKS_PER_SECOND;
    inv_p_mod_q ← inverse p mod q
    inv_q_mod_p ← inverse q mod p
    start = clock();
    decrypt_val ← [ ( xp * q * inv_q_mod_p ) +
                    ( xq * p * inv_p_mod_q ) ] mod n
    end = clock();
    print (decrypt_val);
    total_with_crt = total_with_crt + ( (end - start) / CLOCKS_PER_SECOND );
}
```

```
int main()
{
    input (x)
    generate_keys();
    dp = d mod (p-1)
    dq = d mod (q-1)
    rsa_encrypt();
    rsa_decrypt_using_crt();
    rsa_decrypt();
    print ( total_with_crt ); // time taken using CRT
    print ( total_t) // time taken without using CRT
    return 0;
}
```

Graph.

No of digits X axis	Clocks per second Y axis.	
	Using CRT	without using CRT
10	0.000159	0.000417
50	0.000161	0.000418
100	0.000169	0.000418
150	0.000192	0.000467
~~200~~ 193	0.000197	0.000467

a) Graph representing Y axis as time:-



By looking also at the graph, the X-Axis, we can see that there is a vertical jump (159 → 417)(X axis) The vertical jump is consistent as the number of digits increases.

Conclusion :-

By using chinese Remainder theorm, there is 2-3 times improvement in the performance.

∴ CRT decryption is two to three times more efficient than normal decryption

b) Graph by Representing X Axis as time.