Name – Vivek Pusti
Rollno – 2193283
Class – CSE IS 3

Assignment – 1

Title – Chinese Remainder Theorem

Problem Statement – Implement a number theory such as Chinese remainder Theorem.

Objective – To study and implement Chinese remainder Theorem

Theory –
Chinese Reminder Theorem is used to solve set of congruent equations with one variable but different modulus, which are relatively prime

$$x = a_1 \bmod m_1$$
$$x \equiv a_2 \bmod m_2$$
$$x \equiv a_3 \bmod m_3$$
$$\vdots$$
$$x \equiv a_k \bmod m_k$$

The Chinese Reminder Theorem states that the above equation have unique solution if the moduli are relatively prime. Below are the steps needed to follow to solve set of congruent equation using Chinese Remainder theorem

Step 1: Find $M = m_1 \times m_2 \times m_3 \times \ldots m_k$ where M is common modulus.
Step 2: Find $M_1 = M/m_1$, $M_2 = M/m_2$, and so on
Step 3: Find multiplicative inverse for M_1, M_2 and so on
Step 4: Put the variables in the below equation

$$X = (a_1 \times M_1 \times M_1^{-1} + a_2 \times M_2 \times M_2^{-1} + a_3 \times M_3 \times M_3^{-1}) \bmod M$$

Example

$X = 4 \bmod 5$
$X = 6 \bmod 8$
$X = 8 \bmod 9$

Step 1: $M = 5 \times 8 \times 9 = 360$

Step 2: $M_1 = M/m_1 = 360/5 = 72$
$M_2 = M/m_2 = 360/8 = 45$
$M_3 = M/m_3 = 360/9 = 40$

Step 3: To find M_1 inverse, solve the GCD (m_1, M_1) using Extended Euclidean Algorithm
GCD(5, 72)

q	r1	r2	r	t1	t2	t
0	5	72	5	0	1	0
14	72	5	2	1	0	1
2	5	2	1	0	1	-2

The inverse value cannot be negative, so add modulus into it to make it positive.

M_1 inverse $= -2 + 5 = 3$

To find M_2 inverse, solve for GCD (m_2, M_2) using Extended Euclidean Algorithm.

GCD (8, 45)

q	r_1	r_2	r	t_1	t_2	t
0	8	45	8	0	1	0
5	45	8	5	1	0	1
1	8	5	3	0	1	-1
1	5	3	2	1	-1	2
1	3	2	1	-1	2	-3

M_2 inverse $= -3 + 8 = 5$

To find M_3 inverse, solve for GCD (m_3, M_3) using Extended Euclidean Algorithm.

GCD (9, 40)

q	r_1	r_2	r	t_1	t_2	t
0	9	40	9	0	1	0
4	40	9	4	1	0	1
2	9	4	1	0	1	-2

M_3 inverse $= -2 + 9 = 7$

Step 4: Put the values in the equation to solve x

$$X = (a_1 \times M_1 \times M_1^{-1} + a_2 \times M_2 \times M_2^{-1} + a_3 \times M_3 \times M_3^{-1}) \bmod M$$

$$X = (4 \times 72 \times 3 + 6 \times 45 \times 5 + 8 \times 40 \times 7) \bmod 360$$

$$X = (864 + 1350 + 2240) \bmod 360$$

$$X = 4454 \bmod 360$$

$$X = 134$$

Application - The Chinese Reminder Theorem has several application in cryptography. One is the solve the quadratic congruence and the other is to represent very large number in terms of list of small integers.

Conclusion.
We have studied and implemented Chinese Reminder Theorem.

Tadhaya
13/12/22

Assignment

Run | Debug | Stop | Share | Save | {} Beautify | Language: Python 3

main.py

```
#Chinese Remainder Theorem
# Name-Viveh Pusti

def findMinX(num, rem, k):
    x = 1;
    while(True):

        j = 0;
        while(j < k):
            if (x % num[j] != rem[j]):
                break;
            j += 1;
        if (j == k):
            return x;
        x += 1;

# Driver Code
num = [3, 4, 5];
rem = [2, 3, 1];
k = len(num);
print("x is", findMinX(num, rem, k));
```

input

```
is 11


.Program finished with exit code 0
ess ENTER to exit console.
```

Assignment - 2

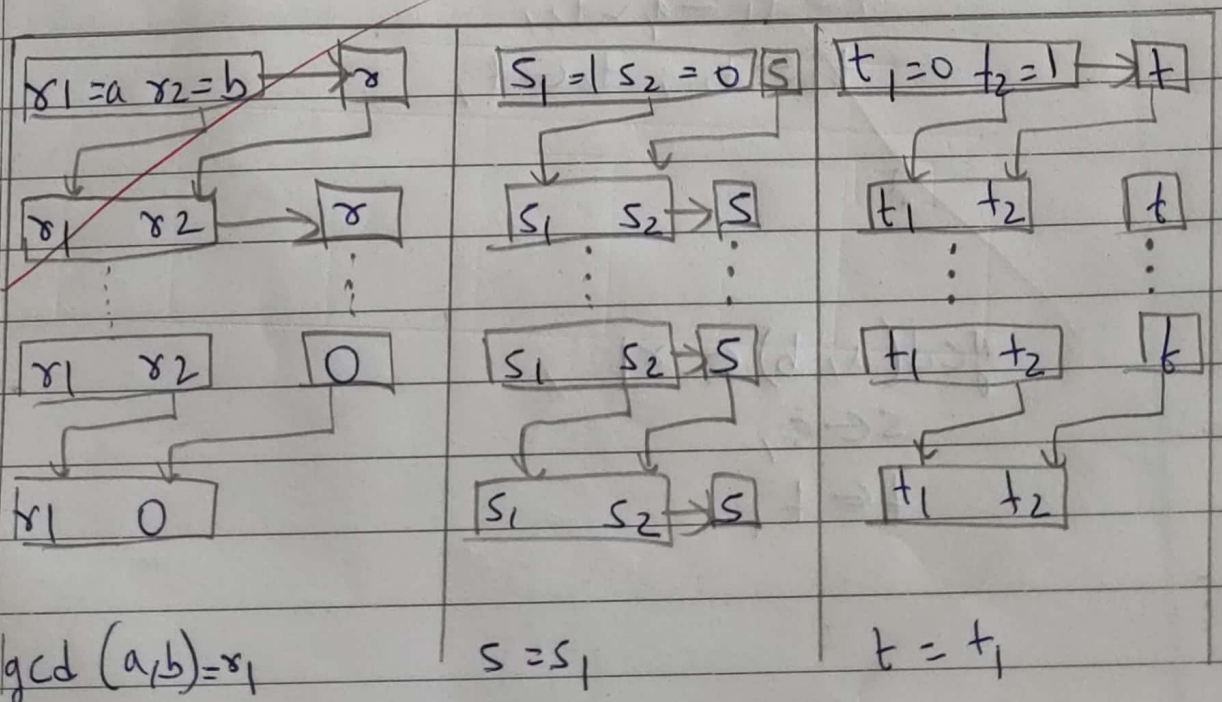
Title - Extended Euclidian Algorithm

Problem Statement - Implement Euclidian and Extended Euclidian algorithm to find out GCD and solve the inverse mod problem.

Objectives - To study and extended Euclidian algorithm.

Theory - The extended Euclidian algorithm is an extension to the Euclidian algorithm. Besides finding the greatest common divisor of integers a and b, as the Euclidian algorithm does, it also finds integers x and y.

$$ax + by = \gcd(a,b) \quad \text{or} \quad sa + tb = \gcd(a,b)$$



Algorithm:
Extend the algorithm to compute the integer coefficients x and y such that
gcd(a,b) = ax + by
Extended-Euclid(a,b)

$r_1 \leftarrow a$; $r_2 \leftarrow b$;
$s_1 \leftarrow 1$; $s_2 \leftarrow 0$;
$t_1 \leftarrow 0$; $t_2 \leftarrow 1$;

while ($r_2 > 0$)
{
$q \leftarrow r_1 / r_2$;
$r \leftarrow r_1 - q \times r_2$;
$r \leftarrow r_2$; $r_2 \leftarrow r$;
$s \leftarrow s_1 - q \times s_2$;
$s \leftarrow s_2$; $s_2 \leftarrow s$;
$t \leftarrow t_1 - q \times t_2$;
$t_1 \leftarrow t_2$; $t_2 \leftarrow t$;
}

$gcd(a,b) \leftarrow r_1$;
$s \leftarrow s$;
$t \leftarrow t_1$

Example :
GCD (161, 28)

q	r1	r2	r	s1	s2	s	t1	t2	t
5	161	28	21	1	0	1	0	1	-5
1	28	21	7	0	1	-1	1	-5	6
3	21	7	0	1	-1	4	-5	6	-23
	7	0		-1	4		6	-23	

Here GCD value are getting as 7. Value of s is -1 and value of t is 6.

$$ax + by = gcd(a, b)$$
$$161 \times (-1) + 28 \times (6) = 7$$

This satisfy the equation for Extended Euclidean Algorithm.

Conclusion - We have studied and implemented the Extended Euclidian Algorithm

rdkulkarni
13/10/22

main.py

```python
# Python program to demonstrate working of extended
# Euclidean Algorithm

# function for extended Euclidean Algorithm
# Name-Vivek Pusti

def gcdExtended(a, b):

    # Base Case
    if a == 0:
        return b, 0, 1

    gcd, x1, y1 = gcdExtended(b % a, a)

    # Update x and y using results of recursive
    # call
    x = y1 - (b//a) * x1
    y = x1

    return gcd, x, y


# Driver code
a, b = 35, 15
g, x, y = gcdExtended(a, b)
print("gcd(", a, ",", b, ") = ", g)

```

input

```
cd( 35 , 15 ) =  5


.Program finished with exit code 0
ess ENTER to exit console.
```

Assignment - 3

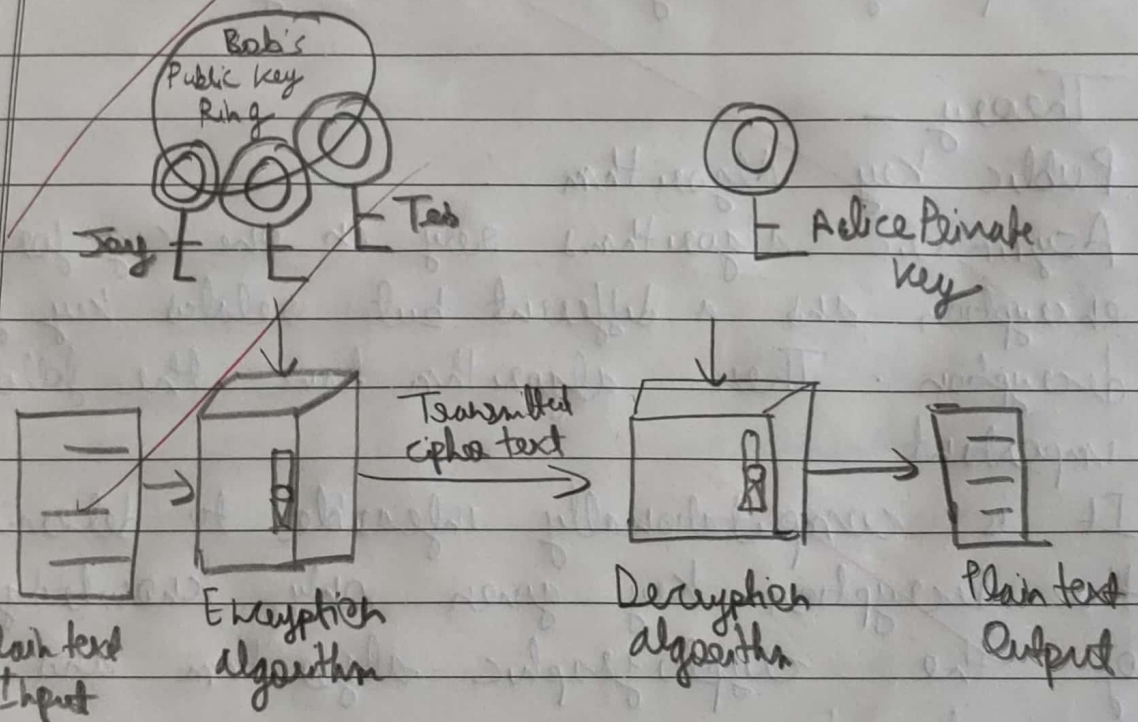
Title - RSA Algorithm

Problem Statement - Implemented RSA public key cryptosystem for key generation and cipher verification.

Objectives - To understand,
1. Public key algorithms,
2. RSA algorithm
3. Concept of Public key and Private key

Theory -
Public key Algorithm :
Asymmetric algorithms rely on one key for encryption and a different but related key for decryption. These algorithm have the following important
- It is computationally infeasible to determine the decryption key given only knowledge of the cryptographic algorithm and the encryption key.

A Public key encryption scheme has six ingredients:
Plaintext - This is readable message or data.
Encryption algorithm - The encryption algorithm performs various transformation on the plain text.
Public and Private key - This is a pair that have been selected so that if one is used for encryption
Cipher text - This is the scrambled message produced as output. It depends on the plaintext and the key.
Decryption algorithm - This algorithm accepts the cipher text



Public key cryptography

The RSA Algorithm

The scheme developed by Rivest, Shamir and Adleman make use of an expression with exponentials. Plain text is encrypted in blocks having a binary value less than some number n. That is the block size must be less than or equal to $\log_2(n)$. practice the block size is i bit, where $2^i < n \leq 2^{i+1}$.

$$C = M^e \bmod n$$
$$M = C^d \bmod n$$

Both sender and reciever must know the value of n. The sender know the value of e, and only the reciever know the value of d.

For this algorithm to be satisfactory for public key

- It is possible to find values of e, d, n.
- It is relatively easy to calculate $M^e \bmod n$ and $C^d \bmod n$ for all values of $M < n$.
- It is feasible to determine d given e and n.

Key Generation

Select p, q	p and q both prime, $q \neq p$
Calculate	$n = p * q$
Calculate	$\phi(n) = (p-1)(q-1)$
Select integer e	$\gcd(\phi(n), e) = 1;\ 1 < e < \phi(n)$
Calculate d	$d = e^{(-1)} \bmod \phi(n)$
Public key	$PU = \{e, n\}$
Private key	$PR = \{d, n\}$

Encryption

Plain txt	$M < n$
Cipher text	$C = M^e \bmod n$

Decryption

Cipher text	C
Plain text	$M = C^d \bmod n$

Example

P = 7, Q = 13, M = 10

Step 1 = $n = 3 \wedge 11 = 33$

Step 2 = $\theta(n) = 2 \times 10 = 20$

Step 3 = Select e, such that gcd $(\theta(n), e) = 1$

gcd(20, 3) = 1 so we select $e = 3$.

Step 4 = To calculate d, solve for gcd($\theta(n)$, e) using extended Euclid's algorithm and pick up the value of t.

q	r_1	r_2	r	t_1	t_2	t
6	20	3	2	0	1	-6
1	3	2	1	1	-6	7

d = 7

Step 5: For encryption

$C = M^e \bmod n$

$= 2 \wedge 3 \bmod 33$

$= 8 \bmod 33$

$= 8$

step 6: For decryption

$M = C^d \bmod n$

$= 8 \wedge 7 \bmod 33$

$= ((8 \wedge 2 \bmod 33)(8 \wedge 2 \bmod 33)(8 \wedge 2 \bmod 33)(8 \wedge 1 \bmod 33)) \bmod 33$

$= (31 \times 31 \times 31 \times 8) \bmod 33$

$(961 \bmod 33)(2488 \bmod 33) \bmod 33$

$68 \bmod 33$

= 2

Advantages
Easy to implement

Disadvantages
1. Anyone can announce the public key.

Algorithm
Start
Input two prime number p and q
Calculate $n = pq$
Calculate $\phi(n) = (p-1)(q-1)$
Input value of e
Determine d
Take input plain text
Encrypt the plaintext and show the output.
Stop.

Conclusion
We have studied and implemented the public key algorithm that is RSA Algorithm.

13/10/22

main.py

```python
# Python for RSA asymmetric cryptographic algorithm.
# Name- Vivek Pusti

import math
def gcd(a, h):
    temp = 0
    while(1):
        temp = a % h
        if (temp == 0):
            return h
        a = h
        h = temp
p = 3
q = 7
n = p*q
e = 2
phi = (p-1)*(q-1)

while (e < phi):
    if(gcd(e, phi) == 1):
        break
    else:
        e = e+1
k = 2
d = (1 + (k*phi))/e
msg = 12.0
print("Message data = ", msg)
c = pow(msg, e)
c = math.fmod(c, n)
print("Encrypted data = ", c)
m = pow(c, d)
m = math.fmod(m, n)
print("Original Message Sent = ", m)
```

input

```
Message data =  12.0
Encrypted data =  3.0
Original Message Sent =  12.0


...Program finished with exit code 0
Press ENTER to exit console.
```

Name - Vivek Pusti
Class - CSE IS 3

Assignment - 4

Problem statement - Implement Diffie Hellman key exchange algorithm for secret key generation and distribution of public key

Objectives:
To learn the basics of key management.
To study and implement Diffie Hellman key exchange algorithm

Theory:
The purpose of the algorithm is to enable two users to securely exchange a key that can then be used for subsequent encryption of messages. The algorithm itself is limited to the exchange of secret of secret values.

Algorithm:
There are two publicly known number a prime number q and an integer a that is a primitive root of q.
Suppose the user A and B wish to exchange a key. User A select a random integer $X_A < q$ and computes $Y_A = a^{X_A} \bmod q$. Similarly User B independently a random integer $X_B < q$ and computes $Y_B = a^{X_B} \bmod q$.
Each sides keeps the value X private and makes the values Y available publicly to the other sides.

User A computes the key K as $K = Y_B^{X_A} \bmod q$
User B computes the key K as $K = Y_A^{X_B} \bmod q$
These two calculation produce identical results

eg:
1) User Alice and Bob who wish to swap keys
Agree on prime $q = 353$ and $a = 3$
Select random secret key
A chooses $X_A = 97$, B chooses $X_B = 233$
Computes public key
$Y_A = 3^{97} \bmod 353 = 40$ (Alice)
$Y_B = 3^{233} \bmod 353 = 248$ (Bob)
Compute shared session key
$K_{AB} = Y_B^{X_A} \bmod 353 = 248^{97} = 160$ (Alice)
$K_{AB} = Y_A^{X_B} \bmod 353 = 40^{233} = 160$ (Bob)

Global Public Elements

q — prime number

α — $\alpha < q$ and α a primitive root of q

User A key Generation

Select private X_A — $X_A < q$

Calculate public Y_A — $Y_A = \alpha^{X_A} \bmod q$

User B key Generation

Select private X_B — $X_B < q$

Calculate public Y_B — $Y_B = \alpha^{X_B} \bmod q$

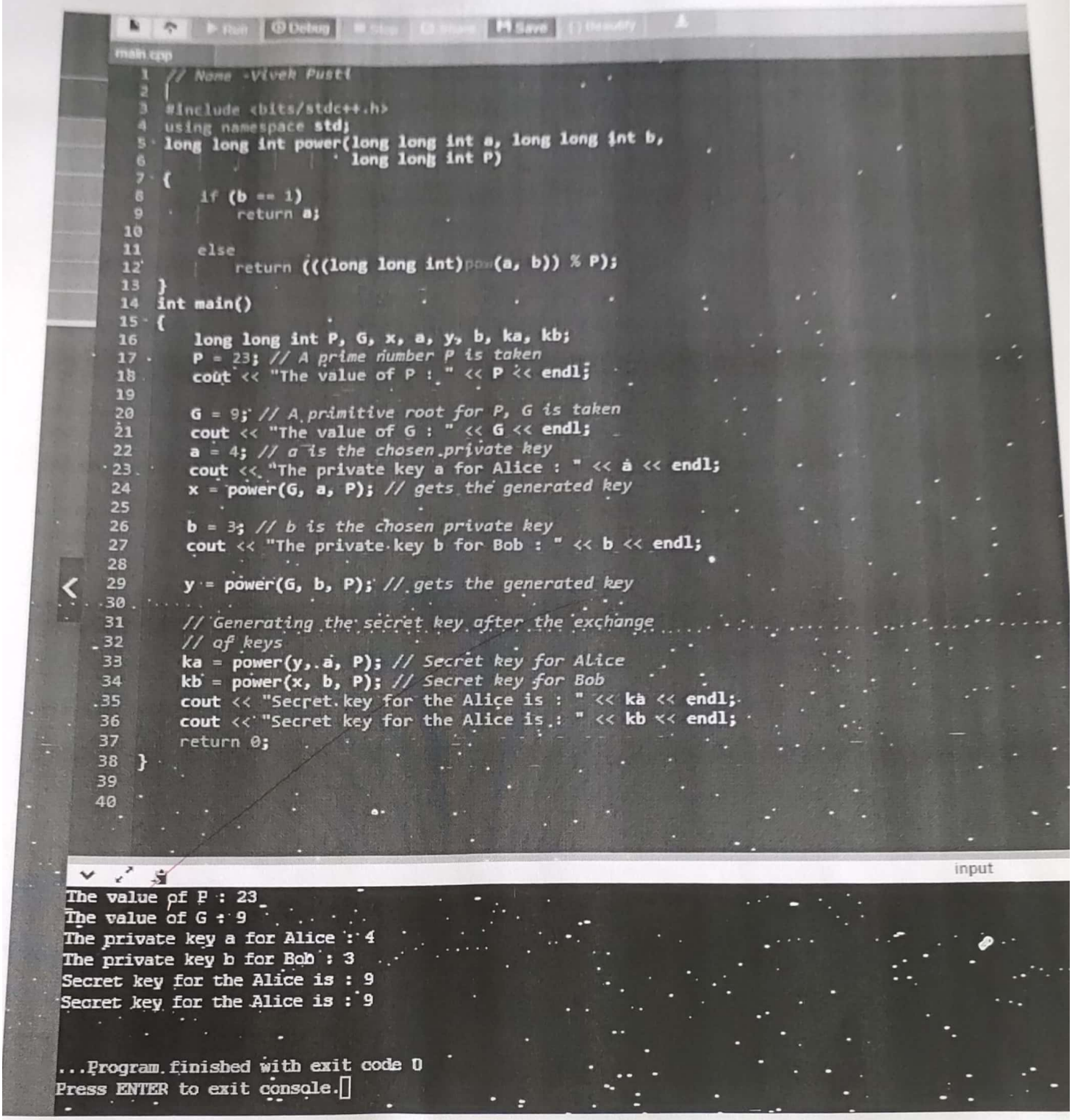
Calculation of Secret key by User A

$K = (Y_B)^{X_A} \bmod q$

Calculation of Secret key by User B

$K = (Y_A)^{X_B} \bmod q$

Conclusion - We have studied and implemented Diffie Hellman key exchange algorithm.



```
// Name -Vivek Pusti
#include <bits/stdc++.h>
using namespace std;
long long int power(long long int a, long long int b,
                    long long int P)
{
    if (b == 1)
        return a;

    else
        return (((long long int)pow(a, b)) % P);
}
int main()
{
    long long int P, G, x, a, y, b, ka, kb;
    P = 23; // A prime number P is taken
    cout << "The value of P : " << P << endl;

    G = 9; // A primitive root for P, G is taken
    cout << "The value of G : " << G << endl;
    a = 4; // a is the chosen private key
    cout << "The private key a for Alice : " << a << endl;
    x = power(G, a, P); // gets the generated key

    b = 3; // b is the chosen private key
    cout << "The private key b for Bob : " << b << endl;

    y = power(G, b, P); // gets the generated key

    // Generating the secret key after the exchange
    // of keys
    ka = power(y, a, P); // Secret key for Alice
    kb = power(x, b, P); // Secret key for Bob
    cout << "Secret key for the Alice is : " << ka << endl;
    cout << "Secret key for the Alice is : " << kb << endl;
    return 0;
}
```

```
The value of P : 23
The value of G : 9
The private key a for Alice : 4
The private key b for Bob : 3
Secret key for the Alice is : 9
Secret key for the Alice is : 9


...Program finished with exit code 0
Press ENTER to exit console.
```

Name - Vivek Purti Roll no - 2193283
Class - CSE IS3

Assignment - 6

AIM - Install and configure kali linux for pentesting.

Objectives:
To learn the basics of kali linux
To configure kali linux for penetration testing

Theory:

Kali linux is an open source, Debian-based Linux distribution aimed at advanced Penetration Testing and Security Auditing. It does this by providing common tools, to focus on the task that needs to be completed, not the surrounding activity. It is geared towards various information security tasks such as Penetration testing, security research, computer forensics and reverse engineering.

Kali linux is a multi platform solution, accessible and freely available to information security professionals and hobbyist.

Kali linux Installation

Different ways of installing Kali Linux:

- Bootable Device
- Hard disk installation
- Dual Boot
- Virtual-box

Kali - linux using Bootable USB:

Step1: Download Kali linux ISO. Download the ISO image file of Kali linux from its official website www.kali.org.

Step2: You should download any bootable image file processing tool, which allows you to create a bootable USB drive.

Step3: Plug your USB drive into your window PC and remember the drive name associated with it ("h:\")

Step4: Launch rufus and choose the Kali linux ISO file that you want to copy on the USB drive.

Match the name of the device selected, which corresponds to the name of the USB device you inserted.

Step 5: click the start button to overwrite the contents of USB device.

Step 6: Once the status bar is completed 100% close and safely remove USB drive from the PC. You can now use your USB drive to boot Kali Linux.

Step 7: Insert your bootable USB drive to your PC and start your PC. Press "Esc" drive startup menu.

Step 8: Press F9 to enter into Boot Device option. It will redirect you to the Boot Manager. These option may vary from drive to device.

Step 9: Select USB hard drive to boot from your bootable USB drive. It will drive you to the Kali Linux live Boot Menu.

Step 10: Select live system and press enter to boot Kali linux.

Conclusion - We have installed Kali linux on windows using bootable USB device.

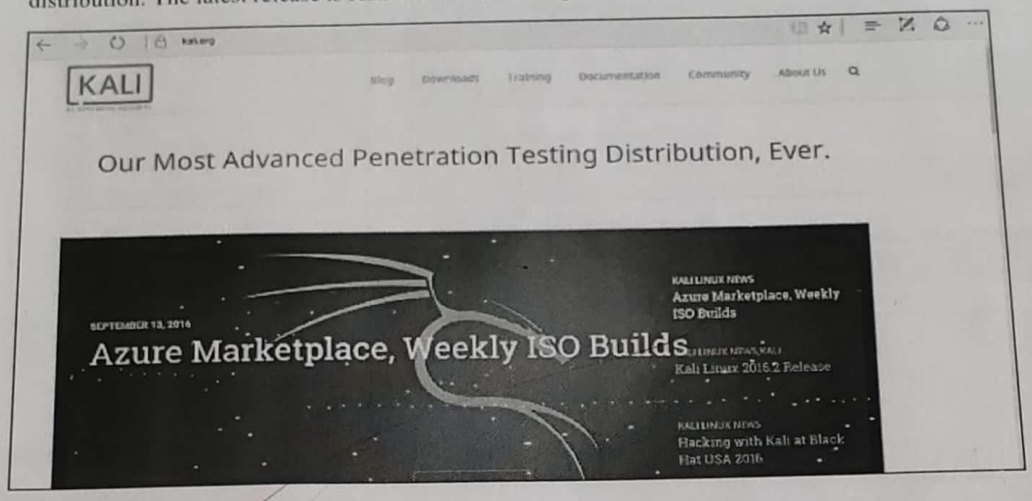
Name: Vivek Pusti

Roll no: 2193283

Is lab – Assignment 05 Kali Linux

Implementation:-

official webpage of kali Linux is https://www.kali.org. Backtrack was the old version of Kali Linux distribution. The latest release is Kali 2016.1 and it is updated very often.



Install Kali Linux

Step 1) Download the Kali Linux package from its official website: https://www.kali.org/downloads/

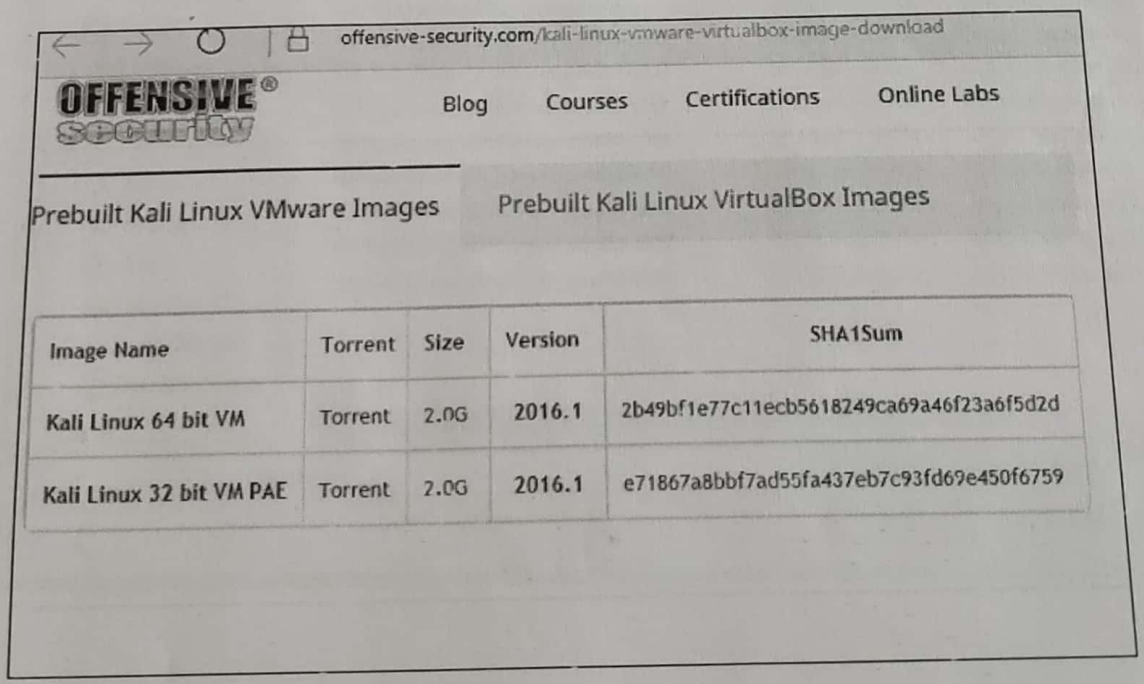
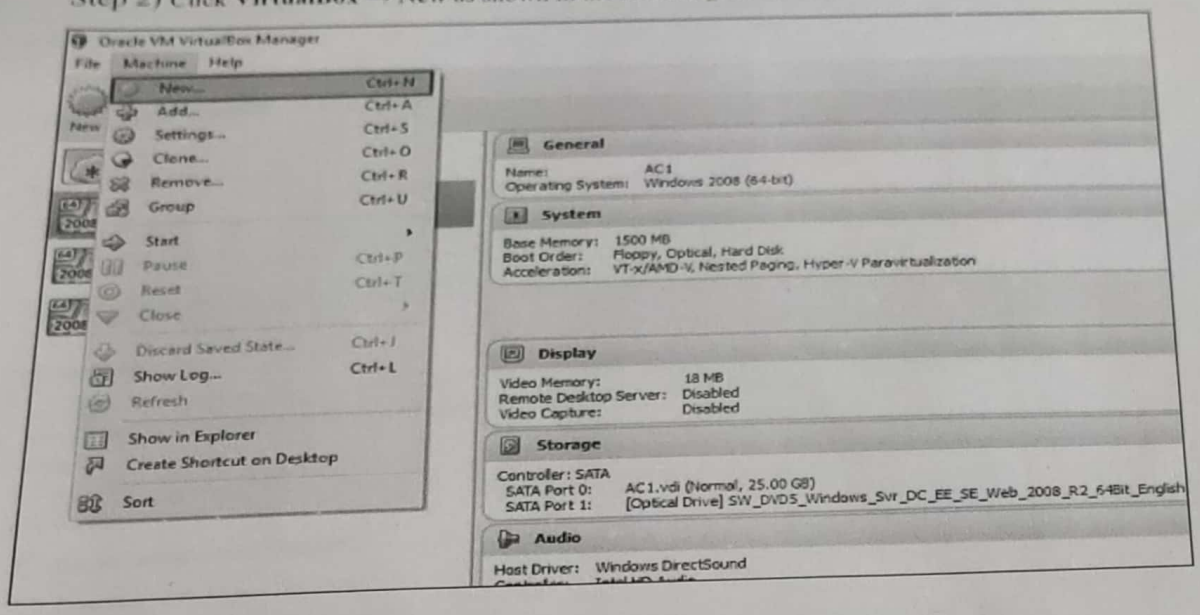
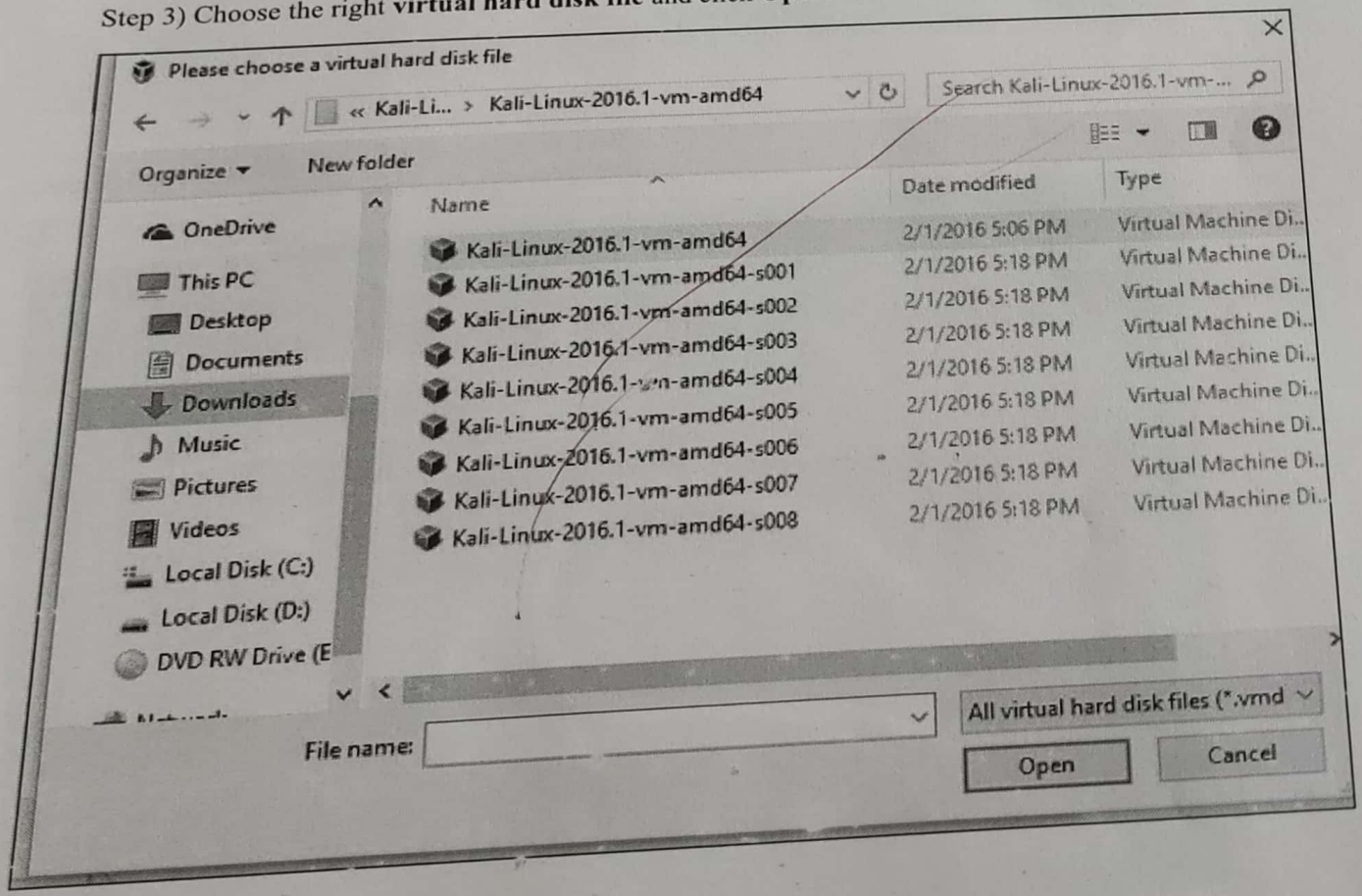


Image Name	Torrent	Size	Version	SHA1Sum
Kali Linux 64 bit VM	Torrent	2.0G	2016.1	2b49bf1e77c11ecb5618249ca69a46f23a6f5d2d
Kali Linux 32 bit VM PAE	Torrent	2.0G	2016.1	e71867a8bbf7ad55fa437eb7c93fd69e450f6759

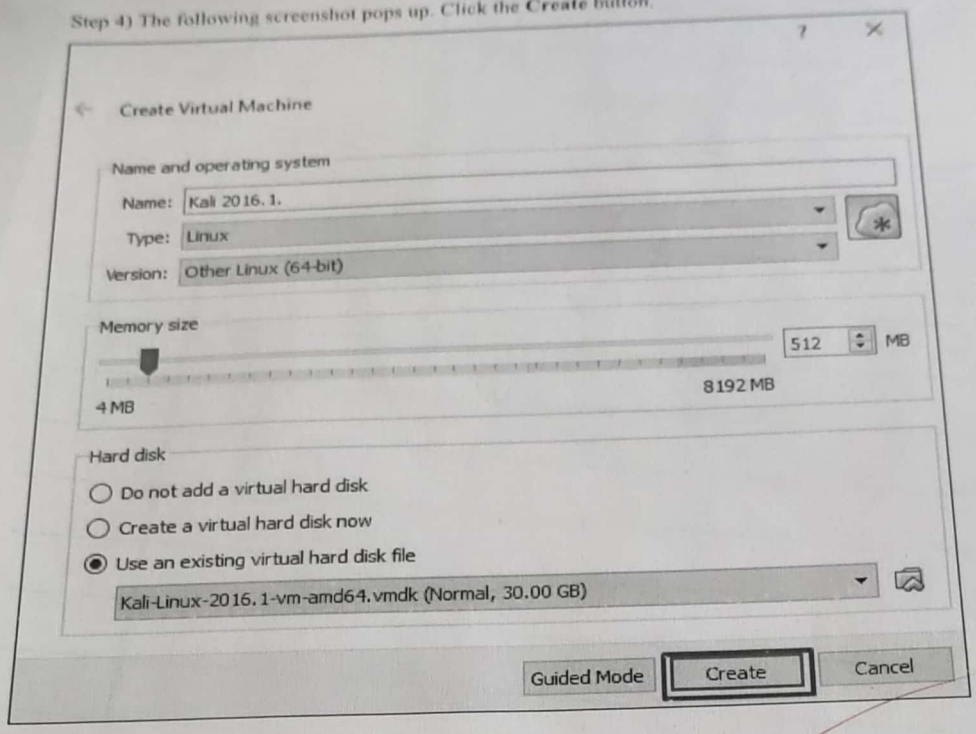
Step 2) Click **VirtualBox → New** as shown in the following screenshot.



Step 3) Choose the right **virtual hard disk file** and click **Open**.



Step 4) The following screenshot pops up. Click the **Create** button.



Step 5) Start Kali OS. The default username is **root** and the password is **toor**.