**Code:Python code for implementing Caesar Cipher**

#A python program to illustrate Caesar Cipher Technique

def encrypt(text,s):

result = ""

# traverse text

for i in range(len(text)):

char = text[i]

# Encrypt uppercase characters

if (char.isupper()):

result += chr((ord(char) + s-65) % 26 + 65)

# Encrypt lowercase characters

else:

result += chr((ord(char) + s - 97) % 26 + 97)

return result

#check the above function

text=input(" Enter the text to encrypt ")

s = 3

print("Text : " + text)

str(s)

print( "Cipher: " + encrypt(text,s))

**Code: Python code for implementing Railfence Cipher**

string=input("enter a string")

def RailFence(txt):

result=""

for i in range(len(string)):

if(i%2==0):

result+=string[i]

for i in range(len(string)):

if(i%2!=0):

result += string[i]

return result print(RailFence(string))

**Code: Java code for implementing Railfence Cipher**

public class railfence {

public static void main(String args[])

{

String input = "ismile";

String output = "";

intlen = input.length(),flag = 0;

System.out.println("Input String : " + input);

for(int i=0;i<len;i+=2) {

output += input.charAt(i);

}

for(int i=1;i<len;i+=2) {

output += input.charAt(i);

}

System.out.println("Ciphered Text : "+output);

}

}

**Code: Python code for implementing RSA Algorithm**

fromCrypto.PublicKey import RSA

fromCrypto.Cipher import PKCS1\_OAEP

importbinascii

keyPair = RSA.generate(1024)

pubKey = keyPair.publickey()

print(f"Public key: (n={hex(pubKey.n)}, e={hex(pubKey.e)})")

pubKeyPEM = pubKey.exportKey()

print(pubKeyPEM.decode('ascii'))

print(f"Private key: (n={hex(pubKey.n)}, d={hex(keyPair.d)})")

privKeyPEM = keyPair.exportKey()

print(privKeyPEM.decode('ascii'))

#encryption

msg = 'Ismile Academy'

encryptor = PKCS1\_OAEP.new(pubKey)

encrypted = encryptor.encrypt(msg)

print("Encrypted:", binascii.hexlify(encrypted))

**Code: Python code for implementing MD5 Algorithm**

importhashlib

result = hashlib.md5(b'Ismile')

result1 = hashlib.md5(b'Esmile')

# printing the equivalent byte value.

print("The byte equivalent of hash is : ", end ="")

print(result.digest())

print("The byte equivalent of hash is : ", end ="")

print(result1.digest())

**Code: Python code for implementing SHA Algorithm**

importhashlib

str = input(" Enter the value to encode ")

result = hashlib.sha1(str.encode())

print("The hexadecima equivalent if SHA1 is : ")

print(result.hexdigest()

**Code: Python code for implementing SHA Algorithm**

fromCrypto.Signature import PKCS1\_v1\_5

fromCrypto.Hash import SHA256

fromCrypto.PublicKey import RSA

from Crypto import Random

defgenerate\_signature(private\_key, message):

# Load the private key

key = RSA.importKey(private\_key)

# Generate SHA-256 hash of the message

hashed\_message = SHA256.new(message.encode('utf-8'))

# Create a signature using the private key

signer = PKCS1\_v1\_5.new(key)

signature = signer.sign(hashed\_message)

return signature

defverify\_signature(public\_key, message, signature):

# Load the public key

key = RSA.importKey(public\_key)

# Generate SHA-256 hash of the message

hashed\_message = SHA256.new(message.encode('utf-8'))

# Verify the signature using the public key

verifier = PKCS1\_v1\_5.new(key)

returnverifier.verify(hashed\_message, signature)

# Generate RSA key pair

random\_generator = Random.new().read

key\_pair = RSA.generate(2048, random\_generator)

# Extract public and private keys

public\_key = key\_pair.publickey().export\_key()

private\_key = key\_pair.export\_key()

# Example usage

message = "Hello, World!"

# Generate a digital signature

signature = generate\_signature(private\_key, message)

print("Generated Signature:", signature)

# Verify the digital signature

is\_valid = verify\_signature(public\_key, message, signature)

print("Signature Verification Result:", is\_valid)

**Code: Python code for implementing Diffie-Hellman Algorithm**

from random import randint

if \_\_name\_\_ == '\_\_main\_\_':

P = 23

G = 9

print('The Value of P is :%d'%(P))

print('The Value of G is :%d'%(G))

a = 4

print('Secret Number for Alice is :%d'%(a))

x = int(pow(G,a,P))

b = 6

print('Secret Number for Bob is :%d'%(b))

y = int(pow(G,b,P))

ka = int(pow(y,a,P))

kb = int(pow(x,b,P))

print('Secret key for the Alice is : %d'%(ka))

print('Secret Key for the Bob is : %d'%(kb))