**Exp 1: The programs should XOR each character in this string with 0 and display the result.**

def main():

str\_ = "Hello World"

str1 = [''] \* len(str\_)

length = len(str\_)

for i in range(length):

str1[i] = chr(ord(str\_[i]) ^ 0)

print(str1[i], end='')

print()

**EXP 2:The program should AND or and XOR each character in this string with 127 and display the result.**

def main():

str\_ = "Hello World"

str1 = [''] \* 11 # Create a list to hold characters

str2 = list(str\_) # Convert string to list of characters

str3 = [''] \* 11 # Create a list to hold characters

length = len(str\_)

# First loop: Copy characters with bitwise AND operation

for i in range(length):

str1[i] = chr(ord(str\_[i]) & 127) # Perform bitwise AND with 127

print(str1[i], end='') # Print character without newline

print() # Newline after first loop

# Second loop: Copy characters with bitwise XOR operation

for i in range(length):

str3[i] = chr(ord(str2[i]) ^ 127) # Perform bitwise XOR with 127

print(str3[i], end='') # Print character without newline

print() # Newline after second loop

if \_\_name\_\_ == "\_\_main\_\_":

**main()**

if \_\_name\_\_ == "\_\_main\_\_":

main()

**OUTPUT** :

Hello World

EXP 3:Write a PYTHON program to perform encryption and decryption using the following algorithms:

1. Ceaser Cipher
2. Substitution Cipher
3. Hill Cipher

**A.Ceaser Cipher**

def encrypt(text, key):

encrypted = ""

for char in text:

c = ord(char)

if char.isupper():

c = c + (key % 26)

if c > ord('Z'):

c = c - 26

elif char.islower():

c = c + (key % 26)

if c > ord('z'):

c = c - 26

encrypted += chr(c)

return encrypted

def decrypt(text, key):

decrypted = ""

for char in text:

c = ord(char)

if char.isupper():

c = c - (key % 26)

if c < ord('A'):

c = c + 26

elif char.islower():

c = c - (key % 26)

if c < ord('a'):

c = c + 26

decrypted += chr(c)

return decrypted

def main():

str\_input = input("Enter any String: ")

key = int(input("\nEnter the Key: "))

encrypted = encrypt(str\_input, key)

print("\nEncrypted String is: " + encrypted)

decrypted = decrypt(encrypted, key)

print("\nDecrypted String is: " + decrypted)

print("\n")

if \_\_name\_\_ == "\_\_main\_\_":

main()

### Output:

Enterany String: HelloWorld Enter the Key: 5

Encrypted String is: MjqqtBtwqi Decrypted Stringis: HelloWord

**B.Substitution Cipher**

def substitution\_cipher():

a = "abcdefghijklmnopqrstuvwxyz"

b = "zyxwvutsrqponmlkjihgfedcba"

str\_input = input("Enter any string: ")

decrypt = ""

for c in str\_input:

j = a.index(c)

decrypt += b[j]

print("The encrypted data is:", decrypt)

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

substitution\_cipher()

Output:

Enter any string: aceho

The encrypted data is: zxvsl

***D.HillCipher***

*import numpy as np*

*class HillCipher:*

*def \_\_init\_\_(self, key\_matrix, message):*

*self.a = np.array(key\_matrix, dtype=float)*

*self.mes = np.array([[ord(c) - 97] for c in message], dtype=float)*

*self.res = np.zeros((3, 1), dtype=float)*

*self.decrypt = np.zeros((3, 1), dtype=float)*

*def encrypt(self):*

*# Matrix multiplication*

*self.res = np.dot(self.a, self.mes)*

*# Encrypt to a string*

*encrypted = ''.join([chr(int(self.res[i][0] % 26) + 97) for i in range(3)])*

*print("\nEncrypted string is: " + encrypted)*

*return encrypted*

*def decrypt\_message(self):*

*self.inverse()*

*# Matrix multiplication*

*self.decrypt = np.dot(self.b, self.res)*

*# Decrypt to a string*

*decrypted = ''.join([chr(int(self.decrypt[i][0] % 26) + 97) for i in range(3)])*

*print("\nDecrypted string is: " + decrypted)*

*return decrypted*

*def inverse(self):*

*# Calculate the inverse of a*

*self.b = np.linalg.inv(self.a)*

*print("\nInverse Matrix is:")*

*print(self.b)*

*def get\_key\_mes():*

*print("Enter 3x3 matrix for key (It should be invertible):")*

*key\_matrix = [[float(input()) for \_ in range(3)] for \_ in range(3)]*

*print("\nEnter a 3 letter string: ")*

*msg = input()*

*return key\_matrix, msg*

*key\_matrix, message = get\_key\_mes()*

*cipher = HillCipher(key\_matrix, message)*

*cipher.encrypt()*

*cipher.decrypt\_message()*

**EXP 4:Write a Java program to implement the DES algorithm logic.**

from Crypto.Cipher import DES

from secrets import token\_bytes

def pad(text):

"""Pads the input text to ensure it's a multiple of 8 bytes."""

while len(text) % 8 != 0:

text += ' '

return text

def generate\_key():

"""Generates a random 8-byte key for DES."""

return token\_bytes(8)

def encrypt(plain\_text, key):

"""Encrypts the plaintext using DES encryption."""

des = DES.new(key, DES.MODE\_ECB)

padded\_text = pad(plain\_text).encode('utf-8')

cipher\_text = des.encrypt(padded\_text)

return cipher\_text

def decrypt(cipher\_text, key):

"""Decrypts the ciphertext using DES decryption."""

des = DES.new(key, DES.MODE\_ECB)

decrypted\_text = des.decrypt(cipher\_text).decode('utf-8').strip()

return decrypted\_text

Enterthestring:WelcomeString To Encrypt: OUTPUT:

Welcome

Encrypted Value : BPQMwc0wKvg Decrypted Value: Welcome

**EXP 5-Write a program to implement the BlowFish algorithm logic.**

from Crypto.Cipher import Blowfish

from Crypto.Random import get\_random\_bytes

from struct import pack

BLOCK\_SIZE = Blowfish.block\_size # Block size for Blowfish (8 bytes)

def pad(text):

"""Pads the input text to ensure it's a multiple of 8 bytes."""

plen = BLOCK\_SIZE - len(text) % BLOCK\_SIZE

padding = [plen]\*plen

padding = pack('b'\*plen, \*padding)

return text + padding

def unpad(text):

"""Removes the padding from the decrypted text."""

plen = text[-1]

return text[:-plen]

def generate\_key():

"""Generates a random Blowfish key (4 to 56 bytes)."""

return get\_random\_bytes(16) # Example: Generate a 16-byte key

def encrypt(plain\_text, key):

"""Encrypts the plaintext using Blowfish encryption."""

cipher = Blowfish.new(key, Blowfish.MODE\_ECB)

padded\_text = pad(plain\_text.encode('utf-8'))

encrypted\_text = cipher.encrypt(padded\_text)

return encrypted\_text

def decrypt(cipher\_text, key):

"""Decrypts the ciphertext using Blowfish decryption."""

cipher = Blowfish.new(key, Blowfish.MODE\_ECB)

decrypted\_padded\_text = cipher.decrypt(cipher\_text)

decrypted\_text = unpad(decrypted\_padded\_text).decode('utf-8')

return decrypted\_text

**Write a PYTHON program to implement the Rijndael algorithm logic**

from Crypto.Cipher import AES

from Crypto.Random import get\_random\_bytes

from Crypto.Util.Padding import pad, unpad

def generate\_key(key\_size=32):

"""

Generates a random key for AES.

The key\_size can be 16 (AES-128), 24 (AES-192), or 32 (AES-256) bytes.

"""

return get\_random\_bytes(key\_size)

def encrypt(plain\_text, key):

"""

Encrypts the plaintext using AES encryption with CBC mode.

:param plain\_text: The plaintext string to encrypt.

:param key: The secret key for encryption (16, 24, or 32 bytes).

:return: A tuple containing the initialization vector (IV) and the ciphertext.

"""

cipher = AES.new(key, AES.MODE\_CBC)

padded\_text = pad(plain\_text.encode('utf-8'), AES.block\_size)

cipher\_text = cipher.encrypt(padded\_text)

return cipher.iv, cipher\_text

def decrypt(cipher\_text, key, iv):

"""

Decrypts the ciphertext using AES decryption with CBC mode.

:param cipher\_text: The encrypted data.

:param key: The secret key used for encryption.

:param iv: The initialization vector used during encryption.

:return: The decrypted plaintext string.

"""

cipher = AES.new(key, AES.MODE\_CBC, iv)

decrypted\_padded\_text = cipher.decrypt(cipher\_text)

decrypted\_text = unpad(decrypted\_padded\_text, AES.block\_size).decode('utf-8')

return decrypted\_text

IMPLEMENT RC4 LOGIC, encryptthetext“Hello world” using BlowFish.

### PROGRAM:

import javax.crypto.Cipher; import javax.crypto.KeyGenerator; import javax.crypto.SecretKey; import javax.swing.JOptionPane; public class BlowFishCipher {

public static void main(String[] args) throws Exception {

// create a key generator based upon the Blowfish cipher KeyGeneratorkeygenerator = KeyGenerator.getInstance("Blowfish");

// create a key

// create a cipher based upon Blowfish Cipher cipher

= Cipher.getInstance("Blowfish");

// initialise cipher to with secret key cipher.init(Cipher.ENCRYPT\_MODE, secretkey);

// get the text to encrypt

String inputText = JOptionPane.showInputDialog("Input your message: "); // encrypt message byte[] encrypted = cipher.doFinal(inputText.getBytes());

//re-initialisetheciphertobeindecryptmode cipher.init(Cipher.DECRYPT\_MODE, secretkey);

// decrypt message

byte[] decrypted = cipher.doFinal(encrypted);

// and display the results

JOptionPane.showMessageDialog(JOptionPane.getRootFrame(), "\nEncrypted text:"+ new String(encrypted)+"\n"+"\nDecryptedtext:"+ new String(decrypted));

System.exit(0);

} }

### OUTPUT:

Input your message: Helloworld Encrypted text: 3ooo&&(\*&\*4r4 Decrypted text: Hello world

### Week 8

Write a Java program to implement RSA Algoithm.

### PROGRAM:

importjava.io.BufferedReader; import java.io.InputStreamReader; import java.math.\*;

import java.util.Random; import java.util.Scanner; public class RSA{

static Scanner sc = new Scanner(System.in); public static void main(String[] args){

// TODO code application logic here System.out.print("Enter a Prime number: ");

BigIntegerp= sc.nextBigInteger();// Here'soneprimenumber.. System.out.print("Enter another prime number: "); BigInteger q = sc.nextBigInteger(); // ..andanother.

BigInteger n = p.multiply(q);

BigInteger n2 = p.subtract(BigInteger.ONE).multiply(q.subtract(BigInteger.ONE)); BigInteger e

= generateE(n2);

BigInteger d = e.modInverse(n2); // Here's the multiplicative inverse

System.out.println("Encryptionkeysare:"+e+","+ n); System.out.println("Decryption keys are: " + d + ", " + n);

}

public static BigIntegergenerateE(BigIntegerfiofn)

{

int y, intGCD;

BigInteger e; BigInteger gcd; Random x = new Random(); do {

y = x.nextInt(fiofn.intValue()-1); String z = Integer.toString(y); e= new BigInteger(z);

gcd = fiofn.gcd(e); intGCD = gcd.intValue();

}

while(y <= 2 ||intGCD != 1); return e;

}

}

### OUTPUT:

Enter a Prime number: 5

Enteranotherprimenumber:11 Encryption keys are: 33, 55

Decryption keys are: 17, 55

### Week 9

Implement the Diffie-Hellman Key Exchange mechanism

import java.math.BigInteger; import java.security.KeyFactory; import java.security.KeyPair;

import java.security.KeyPairGenerator; import java.security.SecureRandom;

import javax.crypto.spec.DHParameterSpec; import javax.crypto.s pec.DHPublicKeySpec; public class DiffeHellman{

public final static int pValue = 47;

public final static int gValue = 71; public final static int XaValue = 9; publicfinalstaticint XbValue= 14;

public static void main(String[] args) throws Exception

{ // TODO code application logic here

BigInteger p = new BigInteger(Integer.toString(pValue)); BigInteger g = new BigInteger(Integer.toString(gValue)); BigIntegerXa = new BigInteger(Integer.toString(XaValue))

; BigIntegerXb = new BigInteger(Integer.toString(XbValue)); createKey(); intbitLength = 512; // 512 bits SecureRandomrnd = new SecureRandom();

p = BigInteger.probablePrime(bitLength, rnd); g = BigInteger.probablePrime(bitLength, rnd);

createSpecificKey(p, g);

}

public static void createKey() throws Exception { KeyPairGeneratorkpg = KeyPairGenerator.getInstance("DiffieHellman"); kpg.initialize(512);

KeyPairkp = kpg.generateKeyPair(); KeyFactorykfactory = KeyFactory.getInstance("DiffieHellman"); DHPublicKeySpeckspec = (DHPublicKeySpec)

kfactory.getKeySpec(kp.getPublic().DHPublicKeySpec.class); System.out.println("Public key is: " +kspec);

}

public static void createSpecificKey(BigInteger p, BigInteger g) throws Exception

{

KeyPairGeneratorkpg = KeyPairGenerator.getInstance("DiffieHellman"); DHParameterSpecparam = new DHParameterSpec(p, g); kpg.initialize(param);

KeyPairkp = kpg.generateKeyPair();

KeyFactorykfactory = KeyFactory.getInstance("DiffieHellman"); DHPublicKeySpeckspec = (DHPublicKeySpec) kfactory.getKeySpec(kp.getPublic(), DHPublicKeySpec.class);

System.out.println("\nPublic key is : " +kspec);

}

}

### OUTPUT:

Public key is: javax.crypto.s pec.DHP ublicKeySpec @5afd 29 Public key is: javax.crypto.s pec. DHPubl icKeyS pec @9971a

### Week 10

Calculate the message digest of a text using the SHA-1 algorithm in JAVA.

### PROGRAM:

import java.security.\*; public class SHA1 {

public static void main(String[] a) { try

{

MessageDigest md = MessageDigest.getInstance("SHA1");

System.out.printl n("Message digest object info: "); System.out.printl n(" Algorithm = " +md.getAlgorithm()); System.out.printl n(" Provider = " +md.getProvi der());

System.out.println(" ToString = " +md.toString()); String input = ""; md.update(input.getBytes()); byte[] output = md.digest(); System.out.println();

System.out.println("SHA1(\""+input+"\") = " +bytesToHex(output));

input = "abc"; md.update(input.getBytes()); output = md.digest(); System.out.println();

System.out.println("SHA1(\""+input+"\") = " +bytesToHex(output));

input = "abcdefghijklmnopqrstuvwxyz"; md.update(input.getBytes()); output = md.digest();

System.out.println();

System.out.println("SHA1(\"" +input+"\") = " +bytesToHex(output)); System.out.println

}

catch (Exception e) {

System.out.println("Exception: " +e);

}

}

public static String bytesToHex(byte[] b) {

char hexDigit[] = {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C', 'D', 'E', 'F'};

StringBufferbuf=new StringBuffer(); for (int j=0; j<b.length;j++)

{ buf.append(hexDigit[(b[j] >> 4) & 0x0f]); buf.append(hexDigit[b[j] & 0x0f]);

}

returnbuf.toString(); }

}

### OUTPUT:

Message digest object info: Algorithm = SHA1 Provider = SUN version 1.6 ToString = SHA1 Message Digest from SUN, <initialized> SHA1("") = DA39A3EE5E6B4B0D3255BFEF95601890AFD80709 SHA1("abc") = A9993E364706816ABA3E25717850C26C9CD0D89D

SHA1("abcdefghijklmnopqrstuvwxyz")=32D10 C7 B8 CF96570 CA04CE37F2A19 D8424 0D3 A89

### Week 11

Calculate the message digest of a text using the SHA-1 algorithm in JAVA.

### PROGRAM:

import java.security.\*; public class MD5 {

public static void main(String[] a) {

// TODO code application logic here try {

MessageDigest md = MessageDigest.getInstance("MD5"); System.out.println("Message digest object info: "); System.out.println(" Algorithm = " +md.getAlgorithm()); System.out.println(" Provider = " +md.getProvider()); System.out.println(" ToString = " +md.toString());

String input = ""; md.update(input.getBytes()); byte[] output = md.digest(); System.out.println();

System.out.println("MD5(\""+input+"\") = " +bytesToHex(output));

input = "abc"; md.update(input.getBytes output = md.digest(); System.out.println();

System.out.println("MD5(\""+input+"\") = " +bytesToHex(output));

input = "abcdefghijklmnopqrstuvwxyz"; md.update(input.getBytes()); output = md.digest();

System.out.println(); System.out.println("MD5(\"" +input+"\") = "

+bytesTo Hex(output)); System.out.println("");

}

catch (Exception e)

{ System.out.println("Exception: " +e); }

}

public static String bytesToHex(byte[] b) {

char hexDigit[] = {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C', 'D', 'E', 'F'};

StringBufferbuf = new StringBuffer(); for (int j=0; j<b.l ength;j++)

{ buf.append(hexDigit[(b[j] >> 4) & 0x0f]); buf.append(hexDigit[b[j] & 0x0f]); } return buf.toString(); } }

### OUTPUT:

Message digest object info:

Algorithm = MD5 Provider = SUN version 1.6

ToString=MD5MessageDigestfromSUN,<initialized>MD5("")= D41D8CD98F00B204E9800998ECF8427E MD5("abc") =

900150983CD24FB0D6963F7D28E17F72 MD5("abcdefghijklmnopqrstuvwxyz")

= C3FCD3D76192E4007DFB496CCA67E13B

* 1. Write a java program to implement Diffie Hellman Key Exchange

**PROGRAM**

class Diffie\_Hellman

{

public static void main(String args[])

{

Scanner sc=new Scanner(System.in); System.out.println("Enter modulo(p)"); int p=sc.nextInt();

System.out.println("Enter primitive root of "+p); int g=sc.nextInt();

System.out.println("Choose 1st secret no(Alice)"); int a=sc.nextInt();

System.out.println("Choose 2nd secret no(BOB)"); int b=sc.nextInt();

int A = (int)Math.pow(g,a)%p; int B = (int)Math.pow(g,b)%p;

int S\_A = (int)Math.pow(B,a)%p; int S\_B =(int)Math.pow(A,b)%p;

if(S\_A==S\_B)

{

System.out.println("ALice and Bob can communicate with each other!!!"); System.out.println("They share a secret no = "+S\_A);

}

else

{

}

}

}

System.out.println("ALice and Bob cannot communicate with each other!!!");

Calculate the message digest of a text using the MD5 algorithm in JAVA.

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

import java.math.BigInteger;

public class MD5HashExample {

public static String getMD5(String input) {

try {

// Create an instance of the MessageDigest class with MD5 algorithm

MessageDigest md = MessageDigest.getInstance("MD5");

// Pass the input string's bytes to the digest method

byte[] messageDigest = md.digest(input.getBytes());

// Convert the byte array into a BigInteger

BigInteger no = new BigInteger(1, messageDigest);

// Convert the BigInteger into a hexadecimal string

String hashtext = no.toString(16);

// While loop to ensure it fills in leading zeros to make it 32 characters

while (hashtext.length() < 32) {

hashtext = "0" + hashtext;

}

return hashtext;

} catch (NoSuchAlgorithmException e) {

throw new RuntimeException(e);

}

}

public static void main(String[] args) {

// Example input text

String input = "Hello MD5!";

// Get the MD5 hash of the input text

String md5Hash = getMD5(input);

// Print the MD5 hash

System.out.println("Original Text: " + input);

System.out.println("MD5 Hash: " + md5Hash);

}

}

IMPLEMENTATION OF HASH FUNCTION

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

public class HashFunctionExample {

// Function to calculate hash using a given algorithm

public static String calculateHash(String algorithm, String input) {

try {

// Create MessageDigest instance for the given algorithm (MD5, SHA-1, SHA-256, etc.)

MessageDigest md = MessageDigest.getInstance(algorithm);

// Convert the input string to bytes and compute the hash

byte[] hashBytes = md.digest(input.getBytes());

// Convert byte array into hexadecimal string

StringBuilder hexString = new StringBuilder();

for (byte b : hashBytes) {

// Convert byte to hexadecimal (00 to ff)

String hex = Integer.toHexString(0xff & b);

if (hex.length() == 1) hexString.append('0'); // Pad with leading zero if necessary

hexString.append(hex);

}

// Return the final hexadecimal hash value

return hexString.toString();

} catch (NoSuchAlgorithmException e) {

throw new RuntimeException("Hash algorithm not found: " + algorithm, e);

}

}

public static void main(String[] args) {

// Input text to be hashed

String inputText = "Hello, World!";

// Calculate and print the hash for different algorithms

System.out.println("Input Text: " + inputText);

System.out.println("MD5 Hash: " + calculateHash("MD5", inputText));

System.out.println("SHA-1 Hash: " + calculateHash("SHA-1", inputText));

System.out.println("SHA-256 Hash: " + calculateHash("SHA-256", inputText));

}

}