**XOR Encryption**

import java.util.Scanner;

public class XOREncryption {

public static String encryptDecrypt(String inputString) {

char xorKey = 'P';

String outputString = "";

int len = inputString.length();

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

outputString = outputString + Character.toString((char)(inputString.charAt(i)^ xorKey));

}

System.out.println(outputString);

return outputString;

}

public static void main(String[] args) {

Scanner s = new Scanner(System.in);

System.out.println("Enter plain text: ");

String sampleString = s.nextLine();

System.out.println("Encrypted String: ");

String encryptedString = encryptDecrypt(sampleString);

System.out.println("Decrypted String: ");

encryptDecrypt(encryptedString);

}

}

**Output**

Enter plain text:

Hello

Encrypted String:

5<<?

Decrypted String:

Hello

**Caesar Cipher**

import java.util.Scanner;

public class CaesarCipher {

public static String encrypt(String str, int key) {

StringBuilder result = new StringBuilder();

key = key % 26;

for (char c : str.toCharArray()) {

if (Character.isLetter(c)) {

char base = Character.isUpperCase(c) ? 'A' : 'a';

c = (char) (base + (c - base + key + 26) % 26);

}

result.append(c);

}

return result.toString();

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter any string: ");

String str = sc.nextLine();

System.out.print("Enter the key: ");

int key = sc.nextInt();

String encrypted = encrypt(str, key);

System.out.println("\nEncrypted String: " + encrypted);

String decrypted = encrypt(encrypted, -key);

System.out.println("Decrypted String: " + decrypted);

sc.close();

}

}

**Output**

Enter any string:

Hello

Enter the key:

24

Encrypted String is: Fcjjm

Decrypted String is: Hello

**AES**

import java.util.\*;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

import java.util.Base64;

import javax.crypto.Cipher;

import javax.crypto.spec.SecretKeySpec;

public class AES {

private static SecretKeySpec secretKey;

public static void setKey(String myKey) {

try {

MessageDigest sha = MessageDigest.getInstance("SHA-1");

byte[] key = Arrays.copyOf(sha.digest(myKey.getBytes()), 16);

secretKey = new SecretKeySpec(key, "AES");

} catch (NoSuchAlgorithmException e) {

e.printStackTrace();

}

}

public static String encrypt(String strToEncrypt, String secret) {

try {

setKey(secret);

Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");

cipher.init(Cipher.ENCRYPT\_MODE, secretKey);

return Base64.getEncoder().encodeToString(cipher.doFinal(strToEncrypt.getBytes()));

} catch (Exception e) {

System.out.println("Error while encrypting: " + e);

return null;

}

}

public static String decrypt(String strToDecrypt, String secret) {

try {

setKey(secret);

Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");

cipher.init(Cipher.DECRYPT\_MODE, secretKey);

return new String(cipher.doFinal(Base64.getDecoder().decode(strToDecrypt)));

} catch (Exception e) {

System.out.println("Error while decrypting: " + e);

return null;

}

}

public static void main(String[] args) {

Scanner scn = new Scanner(System.in);

System.out.print("Enter the secret key: ");

String secretKey = scn.nextLine();

System.out.print("Enter the original message: ");

String originalString = scn.nextLine();

String encryptedString = encrypt(originalString, secretKey);

String decryptedString = decrypt(encryptedString, secretKey);

System.out.println("\nMessage Encryption Using AES Algorithm\n--------");

System.out.println("Original Message: " + originalString);

System.out.println("Encrypted Message: " + encryptedString);

System.out.println("Decrypted Message: " + decryptedString);

scn.close();

}

}

**Output**

Enter the secret key: secret

Enter the original message: Welcome to CSE DEPT

Message Encryption Using AES Algorithm

--------

Original Message: Welcome to CSE DEPT

Encrypted Message: OMH1q4EBwCsTrdpgUItKFnZua5n6f7fBsXWRuhYAPds=

Decrypted Message: Welcome to CSE DEPT

**Blowfish**

import java.nio.charset.StandardCharsets;

import java.security.InvalidKeyException;

import java.security.NoSuchAlgorithmException;

import java.util.Base64;

import javax.crypto.\*;

import javax.crypto.spec.SecretKeySpec;

public class BlowFish {

private static SecretKeySpec getKey(String key) {

return new SecretKeySpec(key.getBytes(StandardCharsets.UTF\_8), "Blowfish");

}

public static String encrypt(String password, String key) throws Exception {

Cipher cipher = Cipher.getInstance("Blowfish");

cipher.init(Cipher.ENCRYPT\_MODE, getKey(key));

return Base64.getEncoder().encodeToString(cipher.doFinal(password.getBytes(StandardCharsets.UTF\_8)));

}

public static String decrypt(String encryptedText, String key) throws Exception {

Cipher cipher = Cipher.getInstance("Blowfish");

cipher.init(Cipher.DECRYPT\_MODE, getKey(key));

return new String(cipher.doFinal(Base64.getDecoder().decode(encryptedText)), StandardCharsets.UTF\_8);

}

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

final String password = "Malla Reddy University";

final String key = "CSE";

System.out.println("Password: " + password);

String encryptedText = encrypt(password, key);

System.out.println("Encrypted text: " + encryptedText);

System.out.println("Decrypted text: " + decrypt(encryptedText, key));

}

}

**Output**

Password: Malla Reddy University

Encrypted text: SBEvS2jP6Ui5mMhSf6bYYkuZOhSwnf3y

Decrypted text: Malla Reddy University

**RSA**

import java.math.BigInteger;

import java.util.Scanner;

public class RSA {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter the number to be encrypted and decrypted: ");

int msg = sc.nextInt();

System.out.print("Enter 1st prime number p: ");

int p = sc.nextInt();

System.out.print("Enter 2nd prime number q: ");

int q = sc.nextInt();

int n = p \* q;

int z = (p - 1) \* (q - 1);

System.out.println("The value of z = " + z);

int e = findE(z);

int d = findD(e, z);

System.out.println("The value of e = " + e);

System.out.println("The value of d = " + d);

BigInteger C = BigInteger.valueOf(msg).pow(e).mod(BigInteger.valueOf(n));

System.out.println("Encrypted message: " + C);

BigInteger msgback = C.pow(d).mod(BigInteger.valueOf(n));

System.out.println("Decrypted message: " + msgback);

}

private static int findE(int z) {

for (int e = 2; e < z; e++) {

if (gcd(e, z) == 1) return e;

}

return 2;

}

private static int findD(int e, int z) {

for (int i = 0; i <= 9; i++) {

int x = 1 + (i \* z);

if (x % e == 0){

return x / e;

}

}

return 1;

}

private static int gcd(int a, int b) {

return (b == 0) ? a : gcd(b, a % b);

}

}

**Output**

Enter the number to be encrypted and decrypted: 21

Enter 1st prime number p: 11

Enter 2nd prime number q: 13

The value of z = 120

The value of e = 7

The value of d = 103

Encrypted message: 109

Decrypted message: 21