CNS Lab Internal

1. Caesar Cipher

Source Code :

```
import java.util.Scanner;

public class CaesarCipher {
    public static String encrypt(String text, int shift) {
        String result = "";
        for (int i = 0; i < text.length(); i++) {
            char c = text.charAt(i);
            if (c >= 'A' && c <= 'Z') {
                c = (char) ((c - 'A' + shift) % 26 + 'A');
            } else if (c >= 'a' && c <= 'z') {
                c = (char) ((c - 'a' + shift) % 26 + 'a');
            }
            result += c;
        }
        return result;
    }

    public static String decrypt(String text, int shift) {
        String result = "";
        for (int i = 0; i < text.length(); i++) {
            char c = text.charAt(i);
            if (c >= 'A' && c <= 'Z') {
                c = (char) ((c - 'A' - shift + 26) % 26 + 'A');
            } else if (c >= 'a' && c <= 'z') {
                c = (char) ((c - 'a' - shift + 26) % 26 + 'a');
            }
            result += c;
        }
        return result;
    }
```

```
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter text: ");
        String text = sc.nextLine();
        System.out.print("Enter key value: ");
        int shift = sc.nextInt();

        String encrypted = encrypt(text, shift);
        String decrypted = decrypt(encrypted, shift);

        System.out.println("Encrypted: " + encrypted);
        System.out.println("Decrypted: " + decrypted);
    }
}
```

Output :

```
Enter text: Hello
Enter key value: 1234
Encrypted: Tqxxa
Decrypted: .KRRU
```

2. Substitution Cipher

Source Code :

```
import java.util.Scanner;

public class SubstitutionCipher {
    static String alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    static String substitute = "QWERTYUIOPASDFGHJKLZXCVBNM";

    public static String encrypt(String text) {
```

```
        String result = "";
        text = text.toUpperCase();
        for (int i = 0; i < text.length(); i++) {
            char c = text.charAt(i);
            int index = alphabet.indexOf(c);
            if (index != -1) result += substitute.charAt(index);
            else result += c;
        }
        return result;
    }

    public static String decrypt(String text) {
        String result = "";
        text = text.toUpperCase();
        for (int i = 0; i < text.length(); i++) {
            char c = text.charAt(i);
            int index = substitute.indexOf(c);
            if (index != -1) result += alphabet.charAt(index);
            else result += c;
        }
        return result;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter text: ");
        String text = sc.nextLine();

        String encrypted = encrypt(text);
        String decrypted = decrypt(encrypted);

        System.out.println("Encrypted: " + encrypted);
        System.out.println("Decrypted: " + decrypted);
    }
}
```

Output :

```
Enter text: Hello
Encrypted: ITSSG
Decrypted: HELLO
```

3. Hill Cipher

Source Code :

```
import java.util.*;
public class  HillCipher {

static int determinant(int[][] m) {
    int det = (m[0][0]*m[1][1] - m[0][1]*m[1][0])  % 26;
    if (det<0) det+= 26;
    return det;
}

static int modInverse(int det) {
    for (int i=1;i<26;i++){
    if ((det*i)%26 == 1) return i;}
    return -1;
}

static int[][] adjoint(int[][]m){
    int[][] adj = new int[2][2];
    adj[0][0] = m[1][1];
    adj[0][1] = m[0][1];
    adj[1][0] = m[1][0];
    adj[1][1] = m[0][0];
    return adj;
}

static int[] multiply(int[][] m, int[] v){
    int[] result = new int[2];
    for (int i=0;i<2;i++){
    result[i] = (m[i][0]*v[0]+m[i][1]*v[i])%26;
```

```
    if(result[i]<0)result[i]+=26;}
    return result;}

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

    int[][]key = new int[2][2];
    System.out.println("Enter 2×2 key matrix ");
    for (int i=0;i<2;i++){
    for (int j=0;j<2;j++){
    key[i][j] = sc.nextInt();
}}

    sc.nextLine();
    System.out.print("Enter plaintext:");
    String word = sc.nextLine().toLowerCase();
    if(word.length()!=2){
        System.out.println("Plaintext must be exactly 2 characters!");
        return;
    }

    int[] textVec = new int[2];
    for(int i=0;i<2;i++){
        textVec[i]=word.charAt(i) - 'a';}

    int[] encryptedVec = multiply(key,textVec);
    StringBuilder encrypted = new StringBuilder();
    for (int val : encryptedVec) encrypted.append((char)(val+'a'));
    System.out.println("Encrypted text :"+ encrypted);

    int det = determinant(key);
    int invDet = modInverse(det);
    if(invDet == -1){
        System.out.println("Key is not invertible. Decryption not possible");
        return;}
        int[][] adj = adjoint(key);
        int[][] invKey = new int[2][2];
        for (int i=0; i<2; i++){
```

```
        for(int j=0;j<2;j++){
        int val = adj[i][j]*invDet;
        val%=26;
        if(val<0)val += 26;
           invKey[i][j]=val;}}
    int[] decryptedVec = multiply(invKey,encryptedVec);
    StringBuilder decrypted = new StringBuilder();
    for(int val:decryptedVec) decrypted.append((char)(val+'a'));
    System.out.println("Decrypted text:" + decrypted);
    sc.close(); }}
```

Output :

```
Enter 2×2 key matrix
3 3
2 5
Enter plaintext:hi
Encrypted text :qc
Decrypted text:uk
```

4. DES

Source Code :

```
import java.util.*
import javax.crypto.*;
import javax.crypto.spec.*;

public class DES{
public static void main(String args[]) throws Exception{
Scanner sc = new Scanner(System.in);
```

```
System.out.print("Enter message to encrypt:");
String msg = sc.nextLine();
byte[] message = msg.getBytes();

System.out.print("Enter custom key:");
String key = sc.nextLine();
byte[]keyData = key.getBytes();
DESKeySpec secretKey = new DESKeySpec(keyData);

SecretKeyFactory keyFactory = SecretKeyFactor.getInstance("DES");
SecretKeyFactory keyN = KeyFactor.generateSecret(secretKey);

Cipher cipher = Cipher.getInstance("DES");
cipher.init(Cipher.ENCRYPT_MODE, keyN);
byte[] encrypted = cipher.doFinal(message);

cipher.init(Cipher.DECRYPT_MODE, keyN);
byte[] decrypted = cipher.doFinal(encrypted);
String decryptedMsg = new String(decrypted)

System.out.println("Message:"+msg);
System.out.println("Encrypted:"+encypted);
System.out.println("Decrypted:"+decryptedMsg);
```

Output :

```
Enter message to encrypt: hello
Enter custom 8-byte key: mysecret
Message: hello
Encrypted: [12, -90, 101, 97, -68, -53, -107, 40]
Decrypted: hello
```

5. RSA

Source Code :

```
import java.security.*;
import javax.crypto.Cipher;
import java.util.Base64;

public class RSA {

   public static void main(String[] args) throws Exception {
      KeyPairGenerator keyGen = KeyPairGenerator.getInstance("RSA");
      keyGen.initialize(2048);
      KeyPair pair = keyGen.generateKeyPair();
      PublicKey publicKey = pair.getPublic();
      PrivateKey privateKey = pair.getPrivate();

      String message = "Hello";
      System.out.println("Original Message: " + message);

      Cipher encryptCipher = Cipher.getInstance("RSA");
      encryptCipher.init(Cipher.ENCRYPT_MODE, publicKey);
      byte[] encryptedBytes = encryptCipher.doFinal(message.getBytes());
      String encryptedMessage = Base64.getEncoder().encodeToString(enc
ryptedBytes);
      System.out.println("Encrypted Message: " + encryptedMessage);

      Cipher decryptCipher = Cipher.getInstance("RSA");
      decryptCipher.init(Cipher.DECRYPT_MODE, privateKey);
      byte[] decryptedBytes = decryptCipher.doFinal(Base64.getDecoder().
decode(encryptedMessage));
      String decryptedMessage = new String(decryptedBytes);
      System.out.println("Decrypted Message: " + decryptedMessage);

  }
}
```

Output :

```
Original Message: Hello
Encrypted Message: VuC878mt5Y3CylZWrMd9PuqtX1MyGlQEdnmARzDHF
```

```
ZlxS83wH6RCEtO
/76lO7MwkWZ0OPAntJ6ZQWah++Cb125LjUuk6SU7lluOHKd/+zevA6zySa3
rdHrgJqnB28R10++
5Eg0lhgJBFU1ZNc4qVNPb1RmeSZUaM0jm7V5oW1z/Z2+MJGPXKFk3rUxI4
feg6xyoeaQsX9iaLZJQogix5t
9lOGgGe2rdwhjyiuyO+SiZ0cyWLD3pPZo/an0pks7t6RVwUqW/nwLlb6/mfo
gjnotzW8VPF8xKPBltLEgGH38s
PuhqKcTozdXn51R+S9jLsdFp7gN+/Ey3RmwCbX2nz8w==
Decrypted Message: Hello
```