Name: Pradyumna Bhosale

Roll Number : 20141224

Experiment No: 1

1.Caesar Cipher

```cpp
#include <iostream>

using namespace std;

int main() {
    cout << "Caesar Cipher program for encryption\n\n";
    cout << "20141224 Pradyumna Bhosale Exp-1. A\n\n";
    string s, t;
    int key;

    cout << "Enter the key: ";
    cin >> key;

    cout << "Enter the message: ";
    cin.ignore(); // Clear the newline character left in the input buffer
    getline(cin, s); // Read a full line of input

    for (int i = 0; i < s.size(); i++) {
        char originalChar = s[i];
        t.push_back(((originalChar + key) + 128) % 128);
    }

    cout << "\n\nEncrypted message is " << t << '\n';

    return 0;
}
```

```
Caesar Cipher program for encryption

20141224 Pradyumna Bhosale Exp-1. A

Enter the key: 5
Enter the message: Pradyumna Bhosale@


Encrypted message is_Uwfi~zrsf%GmtxfqjE
```

// Caesar Cipher Decryption

```cpp
#include <iostream>

using namespace std;

int main() {
    cout << "Caesar Cipher program for decryption\n\n";
    cout << "20141224  Pradyumna Bhosale Exp-1. A\n\n";
    string s, t;
    int key;

    cout << "Enter the key: ";
    cin >> key;

    cout << "Enter the message to decrypt: ";
    cin.ignore(); // Clear the newline character left in the input buffer
    getline(cin, s); // Read a full line of input

    for (int i = 0; i < s.size(); i++) {
        char originalChar = s[i];
        t += ((originalChar - key) + 128) % 128;
    }

    cout << "\n\nDecrypted message is " << t << '\n';

    return 0;
}
```

```
Caesar Cipher program for decryption

20141224  Pradyumna Bhosale Exp-1. A

Enter the key: 5
Enter the message to decrypt: Uwfi~zrsf%GmtxfqjE


Decrypted message is_Pradyumna Bhosale@
```

Playfair Cipher Decryption

```
#include<bits/stdc++.h>
using namespace std;
int main(){
 cout<<"20141224 Pradyumna Bhosale  Exp-1. B\n\n";
 int i,j,k,n;
 cout<<"Enter the message"<<endl;
 string s,origin;
 getline(cin,origin);
 cout<<"Enter the key"<<endl;
 string key;
 cin>>key;
 for(i=0;i<origin.size();i++){
 if(origin[i]!=' ')
 s+= origin[i];
 }
 vector<vector<char> > a(5,vector<char>(5,' '));
 n=5;
 map<char,int> mp;
 k=0;
 int pi,pj;
 for(i=0;i<n;i++){
 for(j=0;j<n;j++){
 while(mp[key[k]]>0&&k<key.size()){
 k++;
 }
 if(k<key.size()){
 a[i][j]=key[k];
 mp[key[k]]++;
 pi=i;
 pj=j;
 }
 if(k==key.size())
 break;
 }
 if(k==key.size())
 break;
 }
 k=0;
 for(;i<n;i++){
 for(;j<n;j++){
 while(mp[char(k+'a')]>0&&k<26){
 k++;
 }
 if(char(k+'a')=='j'){
 j--;
 k++;
 continue;
```

```
}
if(k<26){
a[i][j]=char(k+'a');
mp[char(k+'a')]++;
}
}
j=0;
}
string ans;
if(s.size()%2==1)
s+="x";
for(i=0;i<s.size()-1;i++){
if(s[i]==s[i+1])
s[i+1]='x';
}
map<char,pair<int,int> > mp2;
for(i=0;i<n;i++){
for(j=0;j<n;j++){
mp2[a[i][j]] = make_pair(i,j);
}
}
for(i=0;i<s.size()-1;i+=2){
int y1 = mp2[s[i]].first;
int x1 = mp2[s[i]].second;
int y2 = mp2[s[i+1]].first;
int x2 = mp2[s[i+1]].second;
if(y1==y2){
ans+=a[y1][(x1+1)%5];
ans+=a[y1][(x2+1)%5];
}
else if(x1==x2){
ans+=a[(y1+1)%5][x1];
ans+=a[(y2+1)%5][x2];
} else {
ans+=a[y1][x2];
ans+=a[y2][x1];
} }
cout<<"Encrypted message is ";
cout<<ans<<'\n';
return 0;
}
```

```
20141224 Pradyumna Bhosale  Exp-1. B

Enter the message
helloworld
Enter the key
abc
Encrypted message is kcnvmymtoa
```

Playfair Cipher Decryption

```
#include<bits/stdc++.h>
using namespace std;
int main(){
 cout<<"Playfair cipher Decryption\n";
 cout<<"20141224 Pradyumna Bhosale Exp-1. B\n\n";
 cout<<"Enter the encrypted message\n";
 string s;
 cin>>s;
 int i,j,k,n;
 cout<<"Enter the key\n";
 string key;
 cin>>key;
 vector<vector<char> > a(5,vector<char>(5,' '));
 n=5;
 map<char,int> mp;
 k=0;
 int pi,pj;
 for(i=0;i<n;i++){
 for(j=0;j<n;j++){
 while(mp[key[k]]>0&&k<key.size()){
 k++;
 }
 if(k<key.size()){
 a[i][j]=key[k];
 mp[key[k]]++;
 pi=i;
 pj=j;
 }
 if(k==key.size())
 break;
}
 k=0;
 for(;i<n;i++){
 for(;j<n;j++){
 while(mp[char(k+'a')]>0&&k<26){
 k++;
 }
 if(char(k+'a')=='j'){
 j--;
 k++;
 continue;
 }
 if(k<26){
 a[i][j]=char(k+'a');
 mp[char(k+'a')]++;
 }
 }
```

```
 j=0;}
 string ans;
 map<char,pair<int,int> > mp2;
 for(i=0;i<n;i++){
 for(j=0;j<n;j++){
 mp2[a[i][j]] = make_pair(i,j);
 }
 }
 for(i=0;i<s.size()-1;i+=2){
 int y1 = mp2[s[i]].first;
 int x1 = mp2[s[i]].second;
 int y2 = mp2[s[i+1]].first;
 int x2 = mp2[s[i+1]].second;
if(y1==y2){
 ans+=a[y1][(x1-1)%5];
 ans+=a[y1][(x2-1)%5];
 }
 else if(x1==x2){
 ans+=a[(y1-1)%5][x1];
 ans+=a[(y2-1)%5][x2];
 }
 else {
 ans+=a[y1][x2];
 ans+=a[y2][x1];
 }
 }
 if(ans[ans.size()-1]=='x')
 ans[ans.size()-1]='\0';
 for(i=1;i<ans.size();i++){
 if(ans[i]=='x')
 ans[i]=ans[i-1];
 }
 cout<<"Decrypted message is ";
 cout<<ans<<'\n';
 return 0;
}
}
```

```
Playfair cipher Decryption
20141224 Pradyumna Bhosale Exp-1. B

Enter the encrypted message
kcnvmymtoa
Enter the key
abc
Decrypted message is_helloworld
```

Hill Cipher Encryption

```
#include<bits/stdc++.h>
using namespace std;
int main(){
 cout<<"Hill Cipher Encryption\n";
 cout<<"Pradyumna Bhosale 20141224 Exp1-c \n";
 int x,y,i,j,k,n;
 cout<<"Enter the size of key matrix\n";
 cin>>n;
 cout<<"Enter the key matrix\n";
 int a[n][n];
 for(i=0;i<n;i++){
 for(j=0;j<n;j++){
 cin>>a[i][j];
 } }
 cout<<"Enter the message to encrypt\n";
 string s;
 cin>>s;
 int temp = (n-s.size()%n)%n;
 for(i=0;i<temp;i++){
 s+='x';
 }
 k=0;
 string ans="";
 while(k<s.size()){
 for(i=0;i<n;i++){
 int sum = 0, temp = k;
 for(j=0;j<n;j++){
 sum += (a[i][j]%26*(s[temp++]-'a')%26)%26;
 sum = sum%26;
 }
 ans+=(sum+'a');
 }
 k+=n;
 }
 cout<<ans<<'\n';
 return 0;
}
```

```
Hill Cipher Encryption
Pradyumna Bhosale 20141224 Exp1-c
Enter the size of key matrix
2
Enter the key matrix
2 1
3 4
Enter the message to encrypt
pradyumnabhosale
vjdmqwlkbeczkcax
```

Hill Cipher Decryption

```
#include<bits/stdc++.h>
using namespace std;
int modInverse(int a, int m){
 a=a%m;
 for(int x=-m;x<m;x++)
 if((a*x)%m==1)
 return x;
}
void getCofactor(vector<vector<int> > &a, vector<vector<int> > &temp, int p,
int q, int n){
 int i=0,j=0;
 for(int row=0;row<n;row++){
 for(int col=0;col<n;col++){
 if(row!=p&&col!=q){
 temp[i][j++] = a[row][col];
 if (j==n-1){
 j=0;
 i++;
 } } } } }
int determinant(vector<vector<int> > &a, int n, int N){
 int D = 0;
 if(n==1)
 return a[0][0];
 vector<vector<int> > temp(N, vector<int>(N));
 int sign = 1;
 for(int f=0;f<n;f++){
getCofactor(a, temp, 0, f, n);
 D += sign * a[0][f] * determinant(temp, n - 1, N);
 sign = -sign;
 } return D; }
void adjoint(vector<vector<int> > &a,vector<vector<int> > &adj,int N){
 if(N == 1){
 adj[0][0] = 1;
 return;
 }
 int sign = 1;
 vector<vector<int> > temp(N, vector<int>(N));
 for(int i=0;i<N;i++){
 for(int j=0;j<N;j++){
 getCofactor(a, temp, i, j, N);
 sign = ((i+j)%2==0)? 1: -1;
 adj[j][i] = (sign)*(determinant(temp, N-1 , N));
 } }
}
bool inverse(vector<vector<int> > &a, vector<vector<int> > &inv, int N){
 int det = determinant(a, N, N);
 if(det == 0){
```

```
 cout << "Inverse does not exist";
 return false; }
 int invDet = modInverse(det,26);
 cout<<det%26<<' '<<invDet<<'\n';
 vector<vector<int> > adj(N, vector<int>(N));
adjoint(a, adj, N);
 for(int i=0;i<N;i++)
 for(int j=0;j<N;j++)
 inv[i][j] = (adj[i][j]*invDet)%26;
 return true;
}

int main(){
 cout<<"Hill Cipher Decryption \n";
 cout<<"Pradyumna Bhosale 20141224 Exp1-C\n";
 int x,y,i,j,k,n;
 cout<<"Enter the size of key matrix\n";
 cin>>n;
 cout<<"Enter the key matrix\n";

 vector<vector<int> > a(n, vector<int>(n));
 vector<vector<int> > adj(n, vector<int>(n));
 vector<vector<int> > inv(n, vector<int>(n));
 for(i=0;i<n;i++){
 for(j=0;j<n;j++){
 cin>>a[i][j];
 }
 }
 if(inverse(a,inv,n))
 cout<<"Inverse exist\n";
 cout<<"Enter the message to decrypt\n";
 string s;
 cin>>s;
 k=0;
 string ans;
while(k<s.size()){
 for(i=0;i<n;i++){
 int sum = 0;
 int temp = k;
 for(j=0;j<n;j++){
 sum += ((inv[i][j] + 26)%26*(s[temp++]-'a')%26)%26;
 sum = sum%26;
 }
 ans+=(sum+'a');
 } k+=n;
 }
 int f=ans.size()-1;
 while(ans[f]=='x')
```

```
 f--;
 cout<<"Decrypted message is ";
 for(i=0;i<=f;i++)
 { cout<<ans[i]; }
 return 0;
}
```

```
Hill Cipher Decryption
Pradyumna Bhosale 20141224 Exp1-C
Enter the size of key matrix
2
Enter the key matrix
2 1
3 4
5 21
Inverse exist
Enter the message to decrypt
vjdmqwlkbeczkcax
Decrypted message is pradyumnabhosale
```

Vigenere Cipher Encyption

```
#include<bits/stdc++.h>
using namespace std;
int main(){
 cout<<"Vigenere Cipher Encryption\n";
 cout<<"Pradyumna Bhosale 20141224 Exp1-D\n";
 int i,j,k,n;
 vector<vector<char> > a(26,vector<char>(26));
 k=0;
 n=26;
 for(i=0;i<n;i++){
 k=i;
 for(j=0;j<n;j++){
 a[i][j]='A'+k;
 k++;
 if(k==26)
 k=0;
 }
 }
 cout<<"Enter the message\n";
 string s;
 cin>>s;
 cout<<"Enter the key\n";
 string key;
 cin>>key;
 k=0;
 int mod = key.size();
 for(i=key.size();i<s.size();i++){
 key+=key[k%mod];
 k++;
}
 string encrypt;
 for(i=0;i<s.size();i++){
 encrypt+= a[s[i]-'A'][key[i]-'A'];
 }
 cout<<"Encrypted message: "<<encrypt<<'\n';
 return 0;
}
```

```
Vigenere Cipher Encryption
Pradyumna Bhosale 20141224 Exp1-D
Enter the message
HELLO
Enter the key
PAB
Encrypted message: WEMAO
```

Vigenere Cipher Decryption

```
#include<bits/stdc++.h>
using namespace std;
int main(){
 cout<<"Vigenere Cipher Decryption\n";
 cout<<"Pradyumna Bhosale 20141224 Exp1-D\n";
 int i,j,k,n;
 vector<vector<char> > a(26,vector<char>(26));
 k=0; n=26;
 for(i=0;i<n;i++){
 k=i;
 for(j=0;j<n;j++){
 a[i][j]='A'+k;
 k++;
 if(k==26) k=0;
 } }
 cout<<"Enter the encrypted message\n";
 string s;
 cin>>s;
 cout<<"Enter the key\n";
 string key;
 cin>>key;
 k=0;
 for(i=key.size();i<s.size();i++){
 key+=key[k]; k++;
 }
 string decrypt;
 for(i=0;i<s.size();i++){
 for(j=0;j<n;j++){
 if(a[j][key[i]-'A']==s[i]){
 decrypt += 'A'+j;
 break;
 } } }
 cout<<"Decrypted message: "<<decrypt<<'\n';
 return 0;
}
```

```
Vigenere Cipher Decryption
Pradyumna Bhosale 20141224 Exp1-D
Enter the encrypted message
WEMAO
Enter the key
PAB
Decrypted message: HELLO
```

Name: Pradyumna Bhosale

Roll Number : 20141224

Experiment No: 2

```
#include<bits/stdc++.h>
using namespace std;
int main(){
 cout<<"Rail Fense Encryption \n";
 cout<<"Pradyumna Bhosale 20141224\n";
 int t,n,m,i,j,k,sum=0;
 string s;
 cout<<"Enter the message : "<<'\n';
 cin>>s;
 cout<<"Enter key : "<<'\n';
 cin>>n;
 vector<vector<char>> a(n,vector<char>(s.size(),' '));
 j=0;
 int flag=0;
 for(i=0;i<s.size();i++){
 a[j][i] = s[i];
 if(j==n-1){
 flag=1;
 }
 else if(j==0)
 flag=0;
 if(flag==0){
 j++;
 }
else j--;
 }
 cout<<"Encrypted Message : \n";
 for(i=0;i<n;i++){
 for(j=0;j<s.size();j++){
 if(a[i][j]!=' ')
 cout<<a[i][j];
 }
 }
 cout<<'\n';
 return 0;
 }
```

```
Rail Fense Encryption
Pradyumna Bhosale 20141224
Enter the message :
pradyumnabhosale
Enter key :
3
Encrypted Message :
pyasrdunboaeamhl
```

Name: Pradyumna Bhosale

Roll Number : 20141224

Experiment No: 3

```
package Exp;

import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;
import java.util.Base64;

public class DESEncryptionExample {
    public static void main(String[] args) throws Exception {
        // Generate a DES key
        SecretKey secretKey = KeyGenerator.getInstance("DES").generateKey();

        // Create a DES cipher for encryption
        Cipher encryptionCipher = Cipher.getInstance("DES/ECB/PKCS5Padding");
        encryptionCipher.init(Cipher.ENCRYPT_MODE, secretKey);

        // Create a DES cipher for decryption
        Cipher decryptionCipher = Cipher.getInstance("DES/ECB/PKCS5Padding");
        decryptionCipher.init(Cipher.DECRYPT_MODE, secretKey);

        String plainText = "This is a secret message";

        // Encrypt the data
        byte[] encryptedBytes =
encryptionCipher.doFinal(plainText.getBytes("UTF-8"));
        String encryptedBase64 =
Base64.getEncoder().encodeToString(encryptedBytes);
        System.out.println("Encrypted: " + encryptedBase64);

        // Decrypt the data
        byte[] decryptedBytes =
decryptionCipher.doFinal(Base64.getDecoder().decode(encryptedBase64));
        String decryptedText = new String(decryptedBytes, "UTF-8");
        System.out.println("Decrypted: " + decryptedText);
    }
}
```

```
Encrypted: vPEg4gn4AYw5jW2TxOsN2n5dUuEeR4V2bG6y3BNBWwY=
Decrypted: This is a secret message
PS D:\prog\Java practise>
```

Name: Pradyumna Bhosale

Roll Number : 20141224

Experiment No: 4

```
package Exp;
import java.io.UnsupportedEncodingException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.Arrays;
import java.util.Base64;
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;


public class AES {
 private static SecretKeySpec secretKey;
 private static byte[] key;
 public static void setKey(String myKey) {
 MessageDigest sha = null;
 try {
 key = myKey.getBytes("UTF-8");
 sha = MessageDigest.getInstance("SHA-1");
 key = sha.digest(key);
 key = Arrays.copyOf(key, 16);
 secretKey = new SecretKeySpec(key, "AES");
 } catch (NoSuchAlgorithmException e) {
 e.printStackTrace();
 } catch (UnsupportedEncodingException 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("UTF-
8")));
 } catch (Exception 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.toString());
 }
 return null;
 }
 public static void main(String[] args) {
 final String secretKey = "Pradyumna Bhosale";
 System.out.println("Pradyumna Bhosale (20141224)");
 System.out.println();
 String originalString = " https://www.linkedin.com/in/pradyumna-bhosale-
a65a14206/";
 String encryptedString = AES.encrypt(originalString, secretKey);
 String decryptedString = AES.decrypt(encryptedString, secretKey);
 System.out.println("URL Encryption Using AES Algorithm\n------------");
 System.out.println("Original URL : " + originalString);
 System.out.println("Encrypted URL : " + encryptedString);
 System.out.println("Decrypted URL : " + decryptedString);
 }
}
```

Output:

```
Pradyumna Bhosale (20141224)

URL Encryption Using AES Algorithm
------------
Original URL :  https://www.linkedin.com/in/pradyumna-bhosale-a65a14206/
Encrypted URL : K4aKGvwJIuR8/6TXx47X84n731+qZZ/dJqWyyOevPjO996X3cwPuaQ8h02mQfzDJpikJp1lTBsLz5Hp8wgNPIw==
Decrypted URL :  https://www.linkedin.com/in/pradyumna-bhosale-a65a14206/
PS D:\prog\Java practise>
```

Name: Pradyumna Bhosale

Roll Number : 20141224

Experiment No: 5

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
 cout<<"20141224 Pradyumna Bhosale \n";
 cout<<"Implementation of RSA Algorithm \n";
 cout << "Enter two prime numbers\n";
 double num1, num2;
 cin >> num1 >> num2;
 double n = num1 * num2;
 double track;
 double phi = (num1 - 1) * (num2 - 1);
 double e = 7;
 while (e < phi)
 {
 track = __gcd((int)e, (int)phi);
 if (track == 1)
 break;
 else
 e++;
 }
 double d1 = 1 / e;
 double d = fmod(d1, phi);
 double message;
 cout<<"Enter message\n";
 cin>>message;
 double c = pow(message, e);
 double m = pow(c, d);
 c = fmod(c, n);
 c = pow(message, e);
  m = pow(c, d);
 c = fmod(c, n);
 m = fmod(m, n);
 cout << "Original Message = " << message;
 cout << "\n"<< "p = " << num1;
 cout << "\n"<< "q = " << num2;
 cout << "\n"<< "n = pq = " << n;
 cout << "\n"<< "phi = " << phi;
 cout << "\n"<< "e = " << e;
 cout << "\n"<< "d = " << d;
 cout << "\n"<< "Encrypted message = " << c;
```

```
 cout << "\n"<< "Decrypted message = " << m;
 return 0;
}
```

Output:

```
20141224 Pradyumna Bhosale
Implementation of RSA Algorithm
Enter two prime numbers
7  11
Enter message
5
Original Message = 5
p = 7
q = 11
n = pq = 77
phi = 60
e = 7
d = 0.142857
Encrypted message = 47
Decrypted message = 5
```

Name: Pradyumna Bhosale

Roll Number : 20141224

Experiment No: 6

Implementation Of Diffie Hellman Key exchange algorithm.

```
#include <bits/stdc++.h>
using namespace std;
#define ll long long int
ll power(ll a, ll b,ll P)
{
 if (b == 1)
 return a;
 else
 return (((ll)pow(a, b)) % P);
}
int main()
{
 cout<<"Implementation of Diffie-Hellman Algorithm\n";
 cout<<"20141224 Pradyumna Bhosale Exp-6\n";
 ll P, G, x, a, y, b, ka, kb;
 cout<<"Enter value of P\n";
 cin>>P;
 cout << "The value of P : " << P << endl;
 cout<<"Enter value of G\n";
 cin>>G;
 cout << "The value of G : " << G << endl;
 cout<<"Enter private key for alice\n";
 cin>>a;
 cout << "The private key a for Alice : " << a << endl;
 x = power(G, a, P);
 cout<<"Enter private key for b\n";
 cin>>b;
 cout << "The private key b for Bob : " << b << endl;
 y = power(G, b, P);
 ka = power(y, a, P);
 kb = power(x, b, P);
 cout << "Secret key for the Alice is : " << ka << endl;
 cout << "Secret key for the Bob is : " << kb << endl;
 return 0;
}
```

Output:

```
Implementation of Diffie-Hellman Algorithm
20141224 Pradyumna Bhosale Exp-6
Enter value of P
23
The value of P : 23
Enter value of G
9
The value of G : 9
Enter private key for alice
4
The private key a for Alice : 4
Enter private key for b
3
The private key b for Bob : 3
Secret key for the Alice is : 9
Secret key for the Bob is : 9
```

Name: Pradyumna Bhosale

Roll Number : 20141224

Experiment No: 8

```
package Exp;
import java.math.BigInteger;
import java.security.*;
public class ExpSHA1 {
public static String encryptThisString(String input)
{
try {
MessageDigest md = MessageDigest.getInstance("SHA-1");
byte[] messageDigest = md.digest(input.getBytes());
BigInteger no = new BigInteger(1, messageDigest);
String hashtext = no.toString(16);
while (hashtext.length() < 32) {
hashtext = "0" + hashtext;
}
return hashtext;
}
catch (NoSuchAlgorithmException e) {
throw new RuntimeException(e);
}
}
public static void main(String args[]) throws
NoSuchAlgorithmException
{
System.out.println("Pradyumna Bhosale 20141224");
System.out.println("HashCode Generated by SHA-1 is: ");
String s1 = "Pradyumna";
System.out.println("\n" + s1 + " : " + encryptThisString(s1));
}
}
```

Output:

```
Pradyumna Bhosale 20141224
HashCode Generated by SHA-1 is:

Pradyumna : 523040ed4681a1e425db5388e37ac74d679c23ab
PS D:\prog\Java practise>
```

Name: Pradyumna Bhosale

Roll Number : 20141224

Experiment No: 9

```
package Exp;
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.PrivateKey;
import java.security.Signature;
import java.util.Scanner;
public class DigitalSignature {
public static void main(String args[]) throws Exception {
Scanner sc = new Scanner(System.in);
System.out.println("20141224 Pradyumna Bhosale \nEnter some text ");
String msg = sc.nextLine();
KeyPairGenerator keyPairGen = KeyPairGenerator.getInstance("DSA");
keyPairGen.initialize(2048);
KeyPair pair = keyPairGen.generateKeyPair();
PrivateKey privKey = pair.getPrivate();
Signature sign = Signature.getInstance("SHA256withDSA");
sign.initSign(privKey);
byte[] bytes = "msg".getBytes();
sign.update(bytes);
byte[] signature = sign.sign();
System.out.println("\nDigital signature for given text: "+new
String(signature, "UTF8"));
System.out.println("Digital Signature is Verified");
}
}
```

```
20141224 Pradyumna Bhosale
Enter some text
pradyumna

Digital signature for given text: 0>0↔????_hpu*????/?tn??}M@?N?0↔????|?5??QL→?????>¶?%??w
Digital Signature is Verified
```

Experiment: 12

Title: Demonstrate various methods of Message Authentication.

Theory:

Message authentication is the process of verifying that a message or data has not been tampered with during transmission and that it indeed comes from the claimed source. Various methods of message authentication exist, each with its own strengths and use cases. Here are some commonly used methods:

1. **Message Authentication Code (MAC):**
    - A MAC is a specific code generated using a secret key that is shared between the sender and the recipient.
    - The sender calculates the MAC using a cryptographic algorithm, like HMAC (Hash-based MAC), which combines the message and the secret key.
    - The recipient recalculates the MAC using the received message and the same secret key. If the calculated MAC matches the received MAC, the message is authenticated.
2. **Digital Signatures:**
    - Digital signatures use asymmetric cryptography with a pair of public and private keys.
    - The sender signs the message with their private key to generate a digital signature.
    - The recipient uses the sender's public key to verify the signature. If it's valid, the message is authenticated.
3. **Public Key Infrastructure (PKI):**
    - PKI is a comprehensive system for managing digital keys and certificates.
    - Certificates issued by trusted certificate authorities (CAs) are used to verify the authenticity of messages and entities.
    - It is commonly used for secure web communication, email, and other applications.
4. **Hash Functions:**
    - Hash functions like SHA-256 are used to generate fixed-size hashes from variable-size messages.
    - The sender computes the hash of the message and sends both the message and the hash to the recipient.
    - The recipient calculates the hash of the received message and compares it to the received hash. If they match, the message is considered authentic.
5. **Time Stamping:**

- Time-stamping involves adding a timestamp to a message to prove when it was sent.
- Trusted time-stamping authorities, often referred to as Time Stamp Authorities (TSAs), are responsible for verifying the time of the message.
- Recipients can verify the timestamp to ensure the message hasn't been altered and was sent at the claimed time.

6. **Biometric Authentication:**
    - Biometric methods use unique physical or behavioral characteristics like fingerprints, iris scans, or voice recognition.
    - These methods are used for authenticating individuals rather than messages but can be used to verify the identity of a person sending a message.
7. **Message Authentication without a Key:**
    - This method relies on the inherent structure of the message to detect tampering.
    - For example, appending a checksum or a CRC (Cyclic Redundancy Check) to the message can help verify its integrity without using encryption.
8. **Blockchain Technology:**
    - Blockchain uses distributed ledger technology to provide a tamper-proof and transparent way of verifying the authenticity of data.
    - Once data is added to a blockchain, it's nearly impossible to alter without consensus from the network.
9. **Secure Sockets Layer (SSL) and Transport Layer Security (TLS):**
    - These protocols provide encryption and authentication for data in transit over the internet, ensuring that data exchanged between a client and a server remains confidential and untampered.

Implementation:

Digital Signature:

```
package Exp;
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.PrivateKey;
import java.security.Signature;
import java.util.Scanner;
public class DigitalSignature {
public static void main(String args[]) throws Exception {
Scanner sc = new Scanner(System.in);
System.out.println("20141224 Pradyumna Bhosale \nEnter some text ");
String msg = sc.nextLine();
KeyPairGenerator keyPairGen = KeyPairGenerator.getInstance("DSA");
```

```
keyPairGen.initialize(2048);
KeyPair pair = keyPairGen.generateKeyPair();
PrivateKey privKey = pair.getPrivate();
Signature sign = Signature.getInstance("SHA256withDSA");
sign.initSign(privKey);
byte[] bytes = "msg".getBytes();
sign.update(bytes);
byte[] signature = sign.sign();
System.out.println("\nDigital signature for given text: "+new
String(signature, "UTF8"));
System.out.println("Digital Signature is Verified");
}
}
```

```
20141224 Pradyumna Bhosale
Enter some text
pradyumna

Digital signature for given text: 0>0↔????_hpu*????/?tn??}M@?N?0↔????|?5??QL→?????>¶?%??w
Digital Signature is Verified
```

Hash Functions:

```
package Exp;
import java.math.BigInteger;
import java.security.*;
public class ExpSHA1 {
public static String encryptThisString(String input)
{
try {
MessageDigest md = MessageDigest.getInstance("SHA-1");
byte[] messageDigest = md.digest(input.getBytes());
BigInteger no = new BigInteger(1, messageDigest);
String hashtext = no.toString(16);
while (hashtext.length() < 32) {
hashtext = "0" + hashtext;
}
return hashtext;
}
catch (NoSuchAlgorithmException e) {
throw new RuntimeException(e);
}
}
```

```
public static void main(String args[]) throws
NoSuchAlgorithmException
{
System.out.println("Pradyumna Bhosale 20141224");
System.out.println("HashCode Generated by SHA-1 is: ");
String s1 = "Pradyumna";
System.out.println("\n" + s1 + " : " + encryptThisString(s1));
}
}
```

Output:

```
Pradyumna Bhosale 20141224
HashCode Generated by SHA-1 is:

Pradyumna : 523040ed4681a1e425db5388e37ac74d679c23ab
PS D:\prog\Java practise>
```

HMAC:

```
package Exp12;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import java.security.NoSuchAlgorithmException;
import java.security.InvalidKeyException;
import java.util.Base64;

public class HMACExample {

    public static void main(String[] args) {
        try {
            // Your secret key (should be kept secret)
            String secretKey = "MySecretKey";
            String message = "Hello, HMAC!";

            // Create a SecretKeySpec object from the secret key
            SecretKeySpec secretKeySpec = new
SecretKeySpec(secretKey.getBytes(), "HmacSHA256");

            // Initialize the MAC with the HmacSHA256 algorithm and the secret
key
            Mac mac = Mac.getInstance("HmacSHA256");
            mac.init(secretKeySpec);

            // Calculate the HMAC
            byte[] hmac = mac.doFinal(message.getBytes());
```

```
            // Encode the HMAC as a Base64 string for easy storage and
transmission
            String encodedHmac = Base64.getEncoder().encodeToString(hmac);

            System.out.println("Message: " + message);
            System.out.println("HMAC: " + encodedHmac);

            // To verify the HMAC, you can repeat the process with the
received HMAC and compare it to the calculated HMAC.
        } catch (NoSuchAlgorithmException | InvalidKeyException e) {
            e.printStackTrace();
        }
    }

}
```

```
Message: Hello, HMAC!
HMAC: k8alqpaLEZO8fYRjEoKRWvM0hRiadIQxKQ30eRakMAY=
PS D:\prog\Java practise>
```

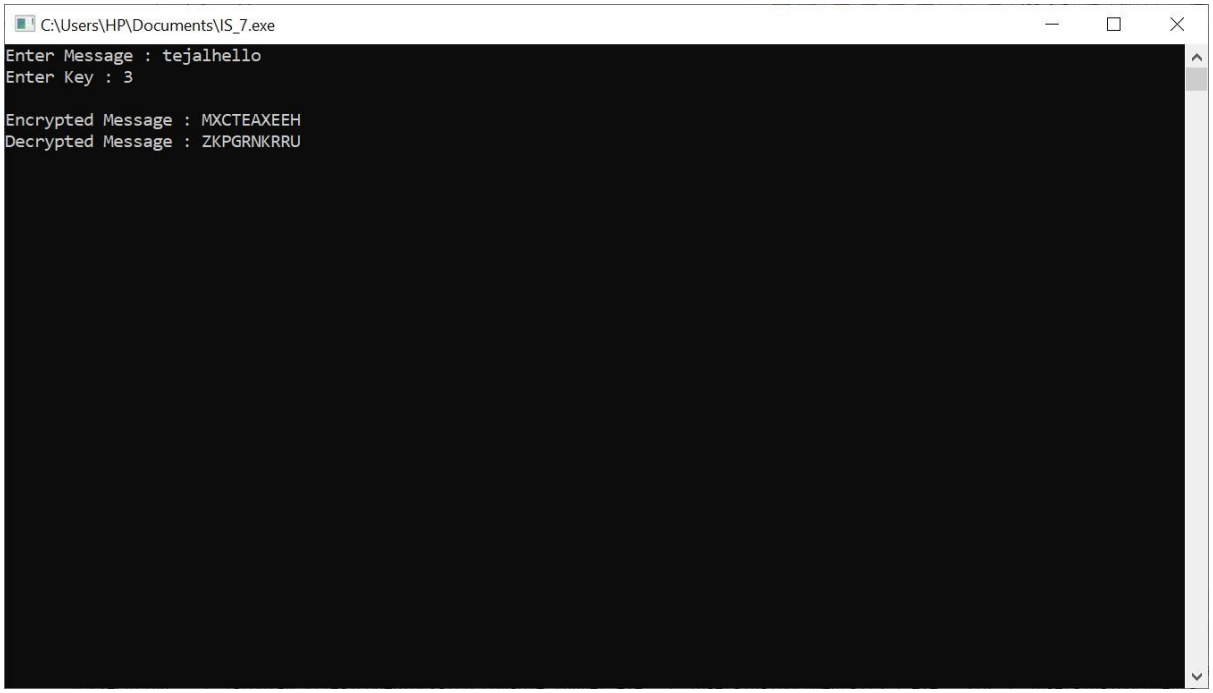
Experiment No 7

Implement and write advantages of Poly-alphabetic Cipher.

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main() {
    char msg[30],key[30],k[20],ct[20],pt[20];
int lenm,lenk,i,j;      //clrscr();

    printf("Enter Message : ");
gets(msg);      printf("Enter
Key : ");      gets(key);
lenm=strlen(msg);
lenk=strlen(key);
for(i=0;i<lenm;i++,j++)
    {
if(j==lenk)
        {
j=0;          }
k[i]=key[j];
    }
    for(i=0;i<lenm;i++)
    {
        ct[i]=((msg[i]+k[i])%26)+'A';
    }
    ct[i]='\0';
for(i=0;i<lenm;i++)
    {
        pt[i]=(((ct[i]-k[i])+26)%26)+'A';
```

```
    }
pt[i]='\0';
    printf("\nEncrypted Message : %s", ct);
printf("\nDecrypted Message : %s", pt);
getch();
}
```



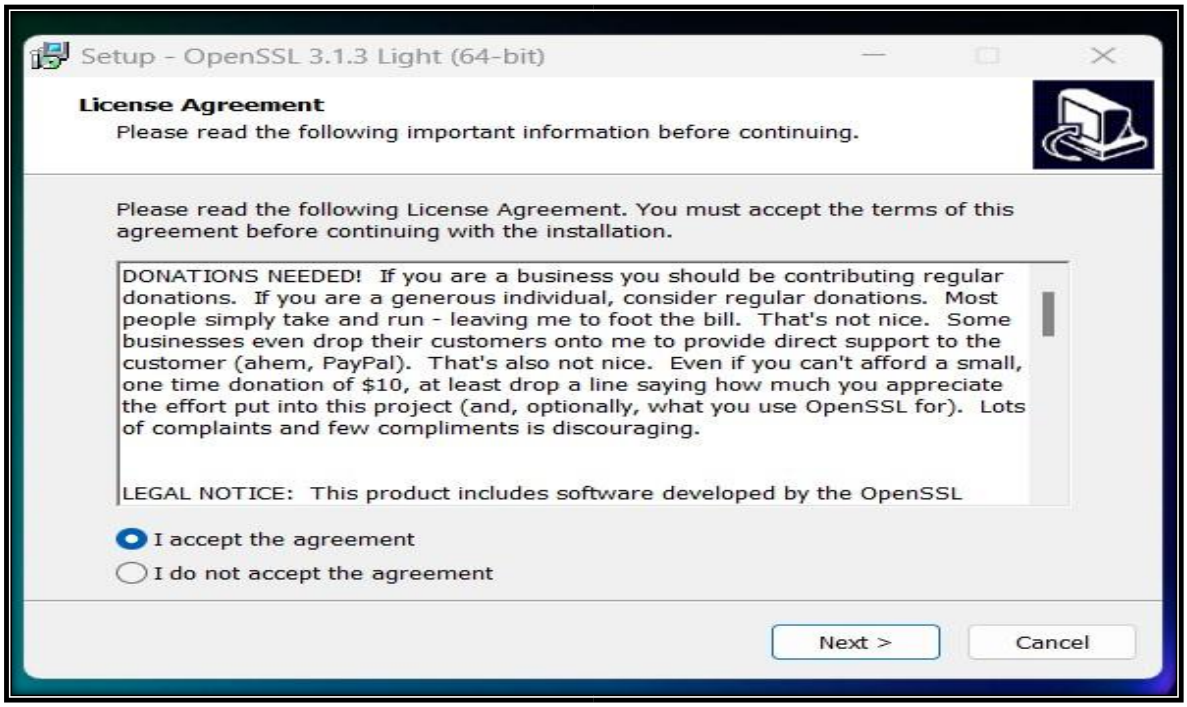
EXPERIMENT NO.10

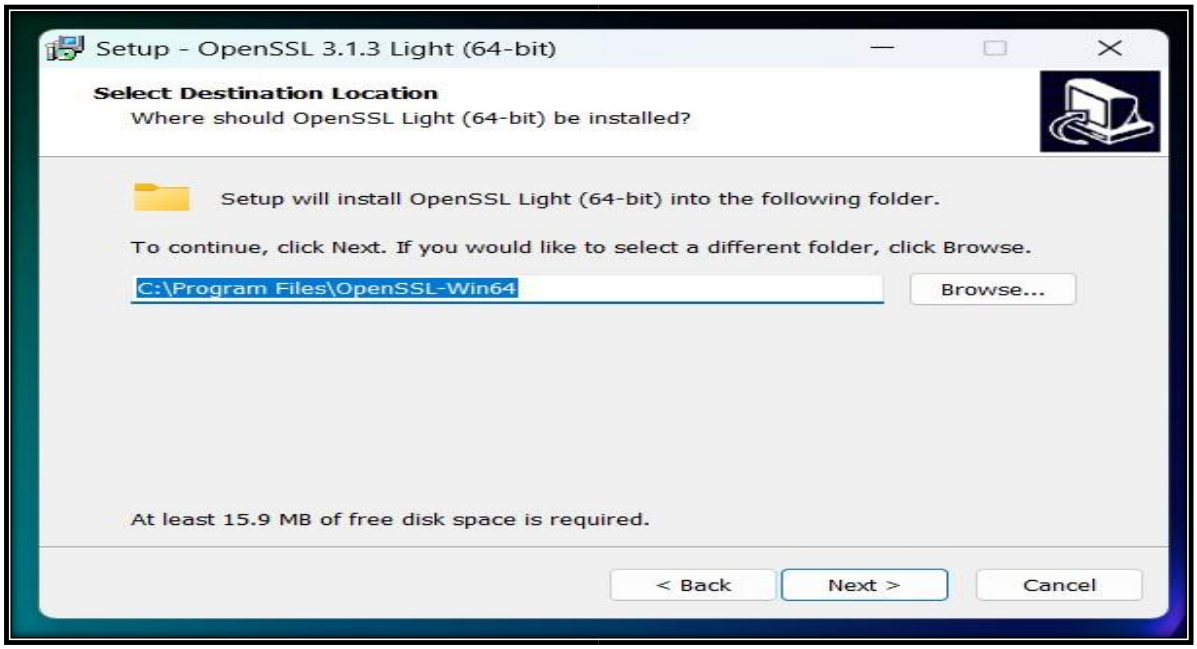
Perform a case study on roll of Private & Public Key.

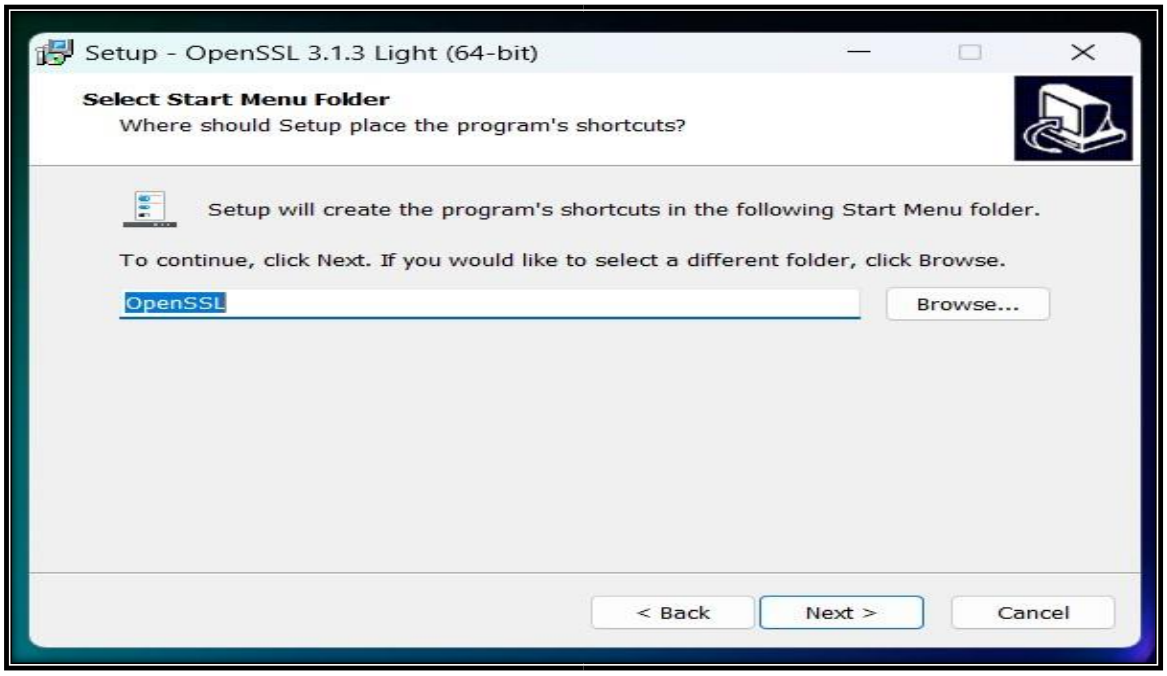
Title: Perform a case study on roll of Private & Public Key.

Objective: Students will be able to implement a case study on roll of Private & Public Key.

Step 1: Download the Openssl

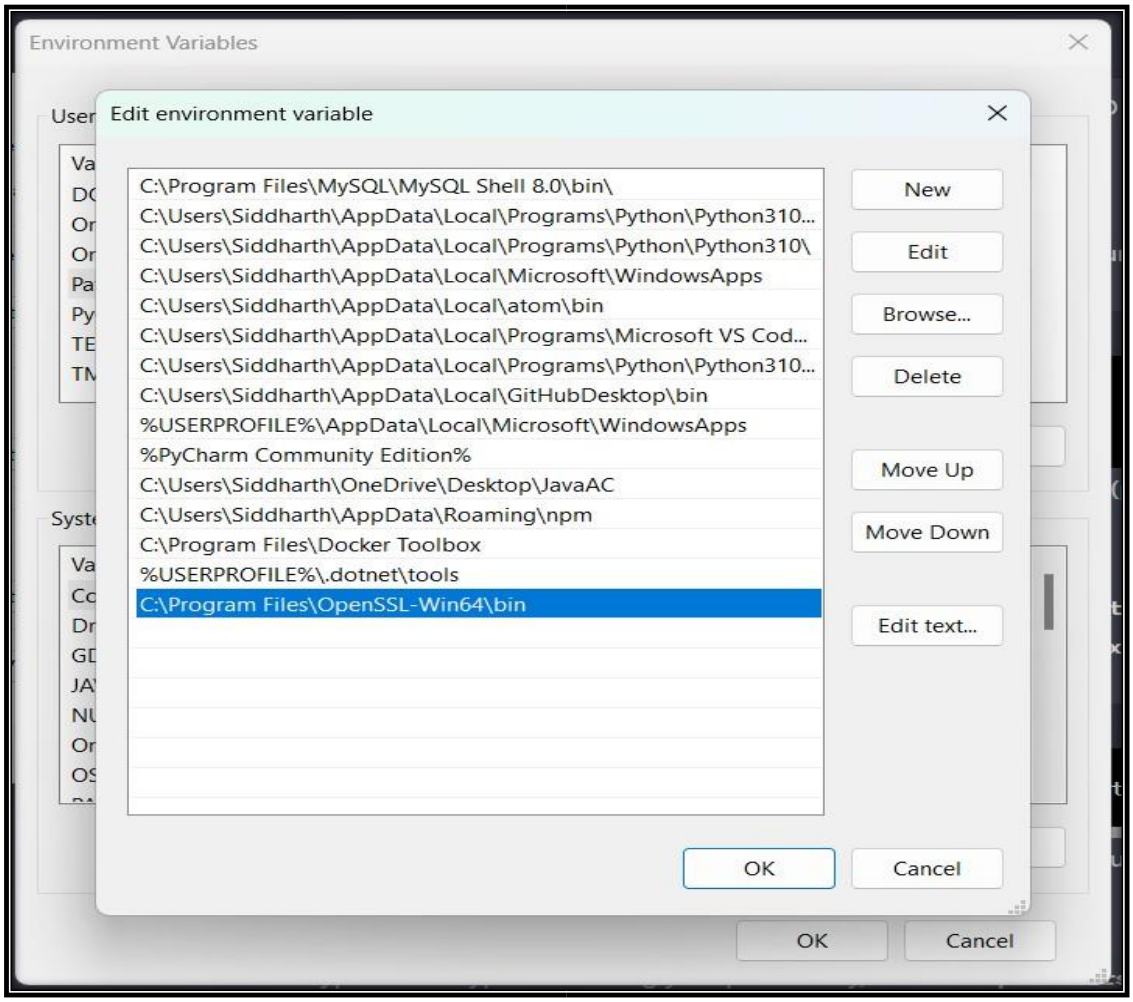






Installed step.

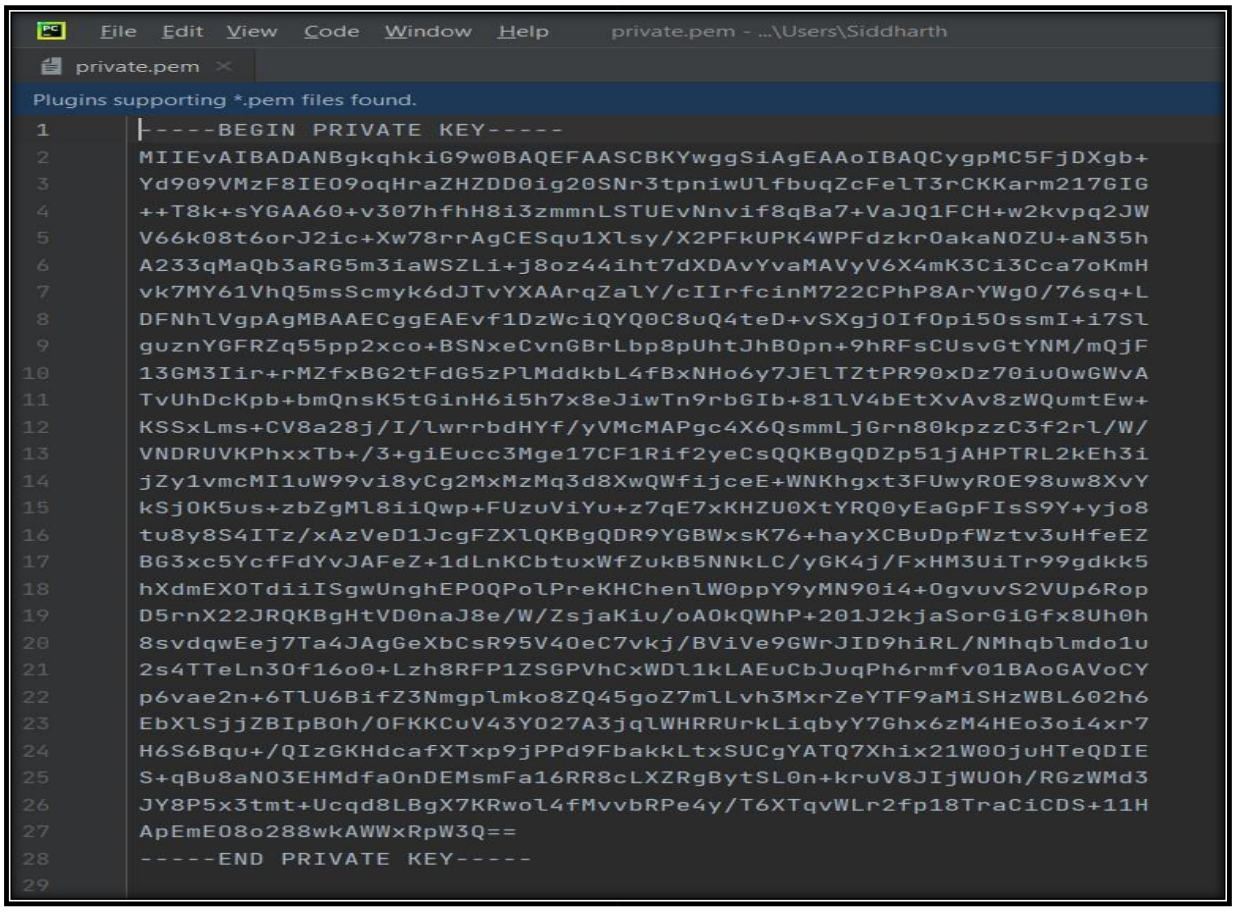
2. Edit in system environment variable



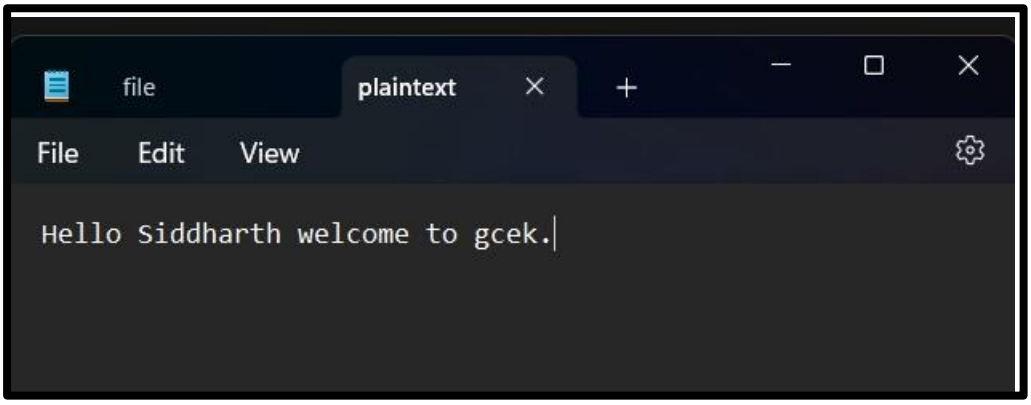
3. Check openssl version command & Create the private key.

Private key created as private.pem

Private key created:



4. Plaintext file - plaintext.txt



5. Extract the public key (public.pem) from it.

```
C:\Users\Siddharth> openssl rsa -pubout -in private.pem -out public.pem
writing RSA key

C:\Users\Siddharth>
```

6. Encryption: To encrypt a message using the recipient's public key, use the pkeyutl command. Replace public.pem with the recipient's public key file and plaintext.txt with the file

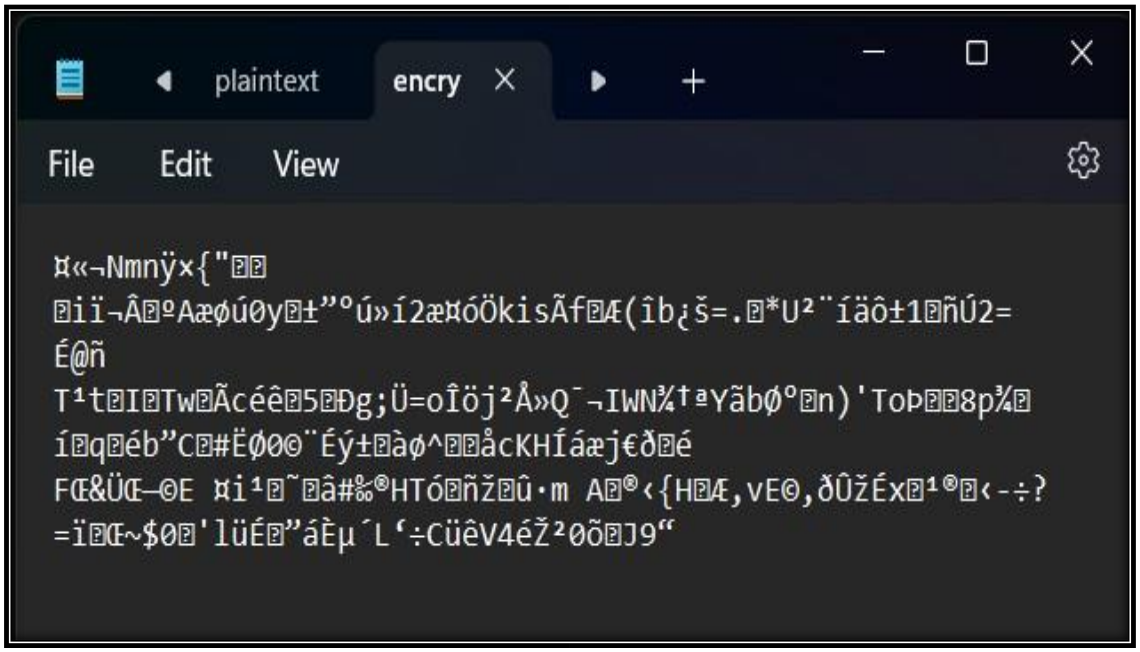
```
C:\Users\Siddharth> openssl rsautl -encrypt -pubin -inkey public.pem -in plaintext.txt -out encrypted.txt
The command rsautl was deprecated in version 3.0. Use 'pkeyutl' instead.

C:\Users\Siddharth>openssl pkeyutl -encrypt -in plaintext.txt -out encrypted.txt -inkey public.pem -pubin -padding
pkeyutl: Unknown option: -padding
pkeyutl: Use -help for summary.

C:\Users\Siddharth>openssl pkeyutl -encrypt -in plaintext.txt -out encrypted.txt -inkey public.pem -pubin

C:\Users\Siddharth>
```

7. Encrypted file –



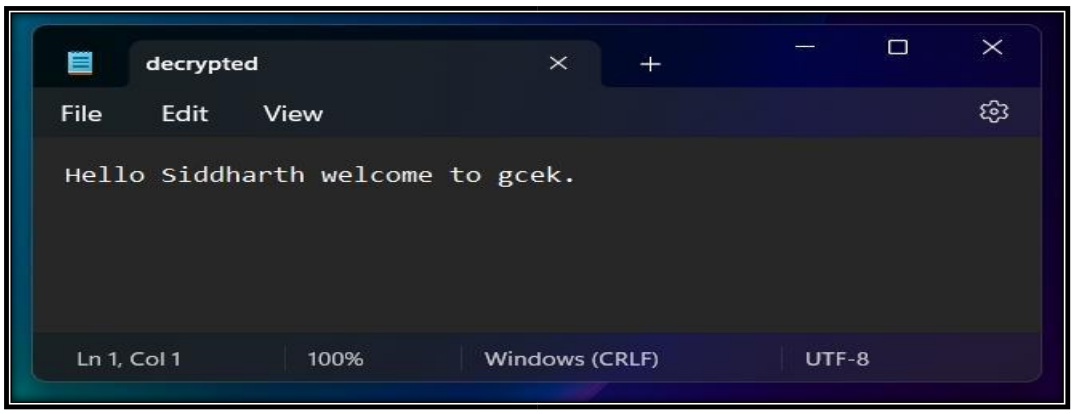
8. Decryption-

To decrypt the encrypted message using your private key, use the pkeyutl command again. Replace private.pem with your private key file and encrypted.txt with the file.

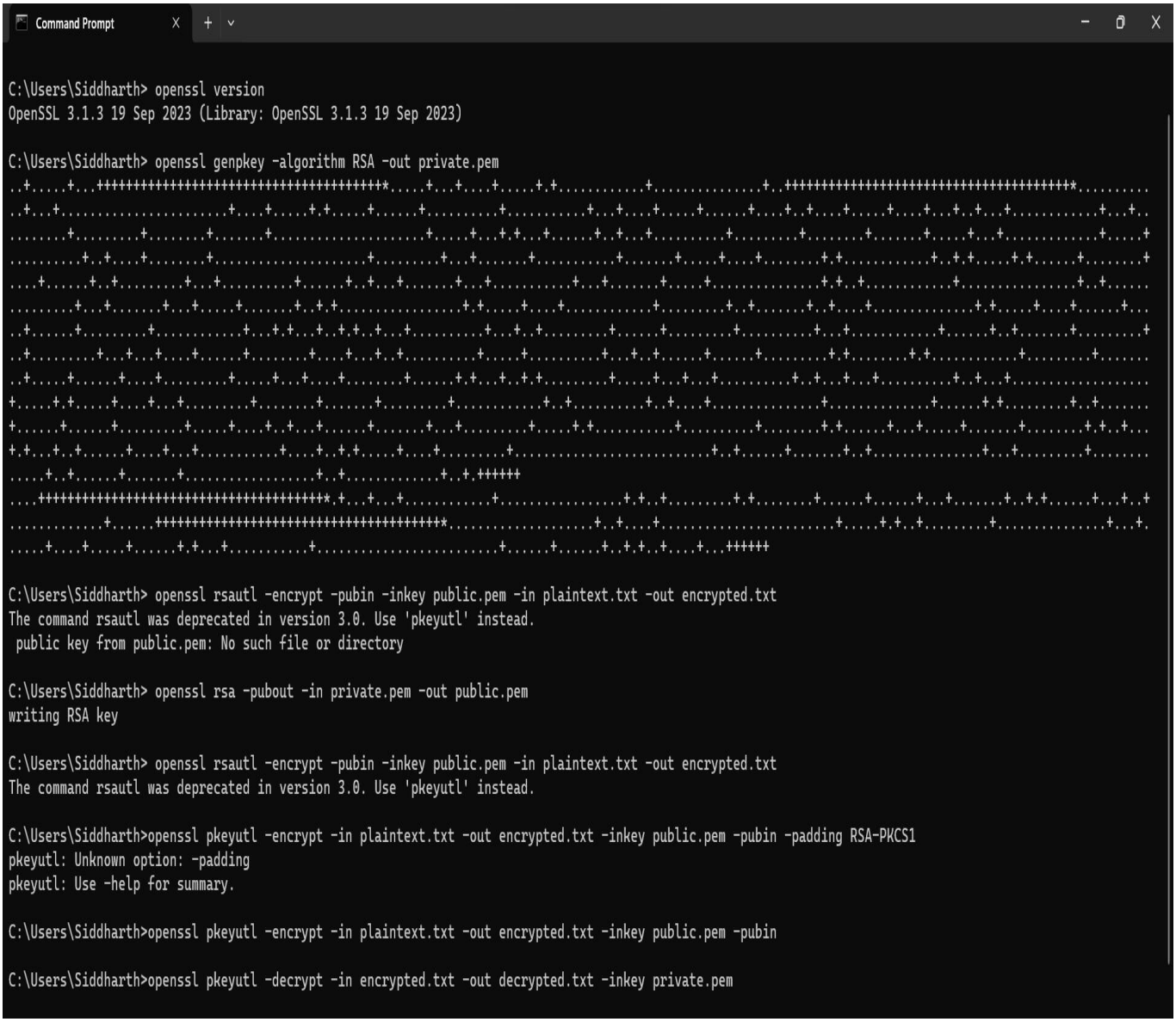
```
C:\Users\Siddharth>openssl pkeyutl -decrypt -in encrypted.txt -out decrypted.txt -inkey private.pem

C:\Users\Siddharth>
```

9. Decrypted file –



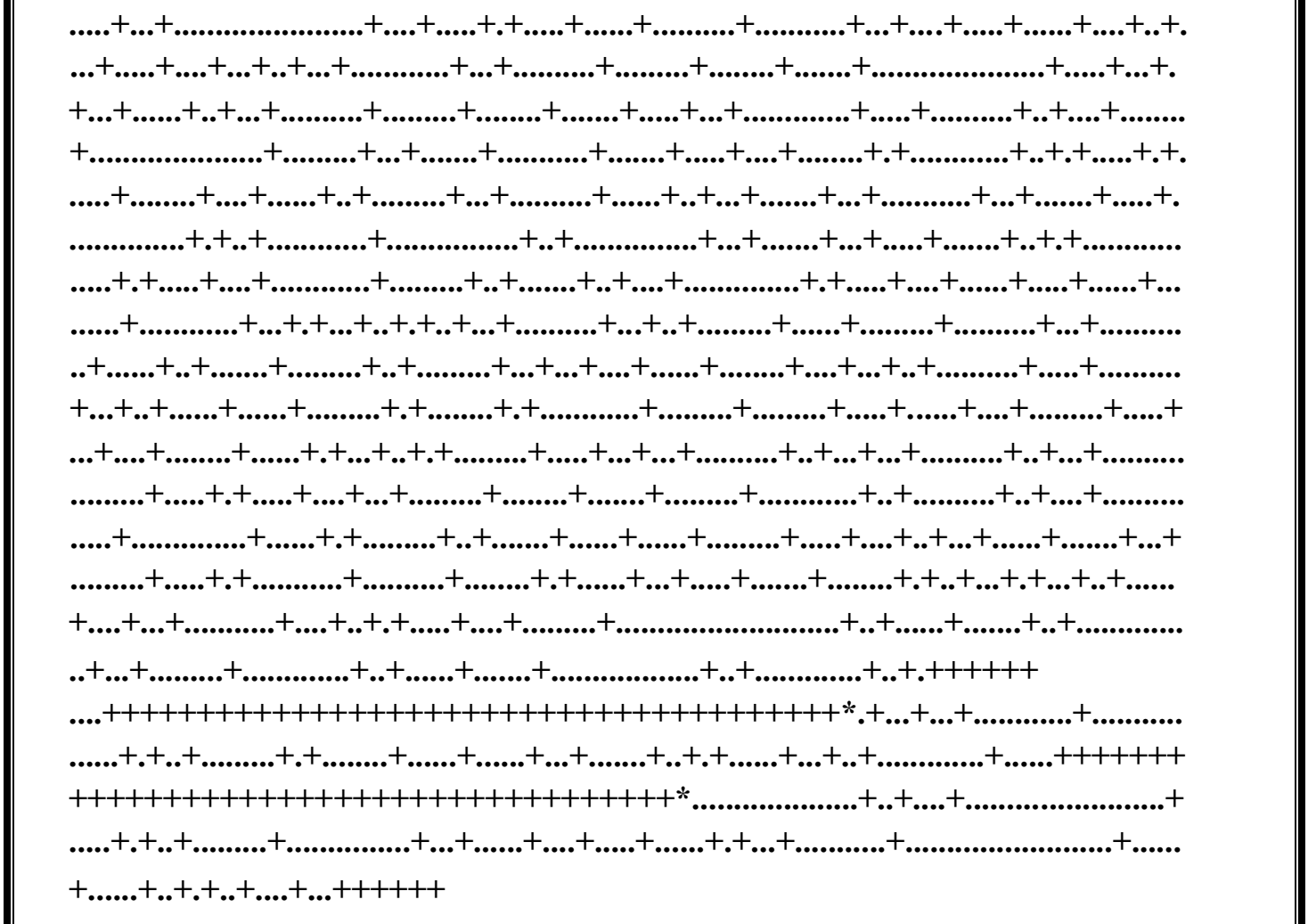
Complete commands:



Microsoft Windows [Version 10.0.22621.2428] (c)
Microsoft Corporation. All rights reserved.

C:\Users\Siddharth> openssl version
OpenSSL 3.1.3 19 Sep 2023 (Library: OpenSSL 3.1.3 19 Sep 2023)

C:\Users\Siddharth> openssl genpkey -algorithm RSA -out private.pem
..+.....+...++*.....+...+....+.....+.
+............+.............+..++*.....



C:\Users\Siddharth> openssl rsautl -encrypt -pubin -inkey public.pem -in plaintext.txt -out encrypted.txt
The command rsautl was deprecated in version 3.0. Use 'pkeyutl' instead.
public key from public.pem: No such file or directory

C:\Users\Siddharth> openssl rsa -pubout -in private.pem -out public.pem
writing RSA key

C:\Users\Siddharth> openssl rsautl -encrypt -pubin -inkey public.pem -in plaintext.txt -out encrypted.txt
The command rsautl was deprecated in version 3.0. Use 'pkeyutl' instead.

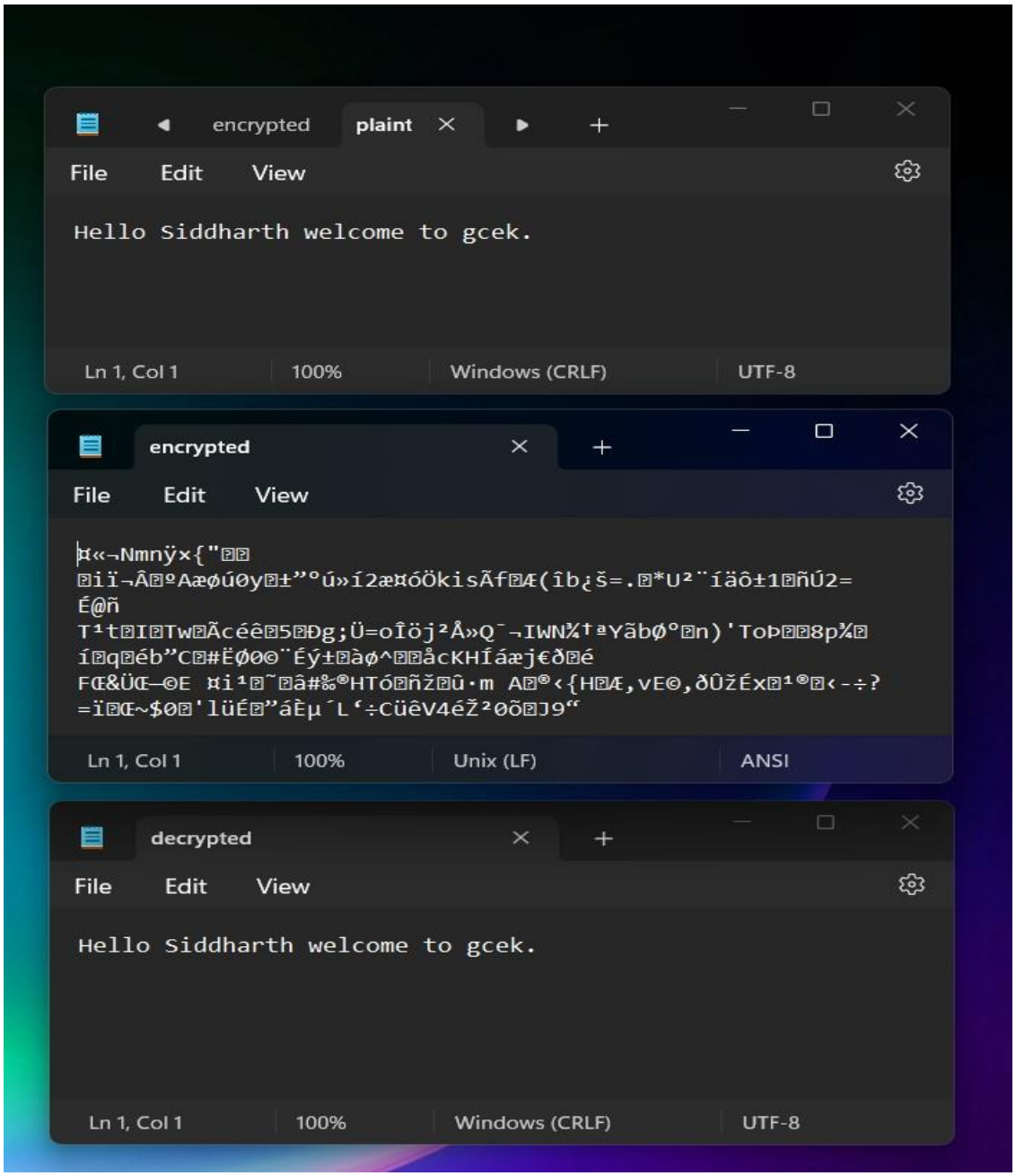
C:\Users\Siddharth>openssl pkeyutl -encrypt -in plaintext.txt -out encrypted.txt -inkey public.pem -pubin -padding RSA-PKCS1 pkeyutl: Unknown option: -padding pkeyutl: Use -help for summary.

C:\Users\Siddharth>openssl pkeyutl -encrypt -in plaintext.txt -out encrypted.txt -inkey public.pem -pubin

C:\Users\Siddharth>openssl pkeyutl -decrypt -in encrypted.txt -out decrypted.txt -inkey private.pem

C:\Users\Siddharth>

Output:



Sender File

Conclusion:

In this case study, I have studied and implemented roll of Private & Public Key.