Course Description, Objectives and Outcomes

Course Code and Title: IT8761- Security Lab

Course Description

Provided knowledge about Network Security and Security Issues which will help in their research activities. The main focus of this Course is to acquire the technical knowledge about security.

Course Objective

The students should be made to:

1. Be exposed to the different cipher techniques.
2. Learn to implement the algorithms like DES, RSA, MD5, SHA-1.
3. Understand the Digital Signature Standard.
4. Learn to use network security tools like GnuPG, Net Stalker.
5. Be familiar with the intrusion detection system.

Course Outcomes

At the end of the course students will be able to:

CO1: Develop code for classical Encryption Techniques to solve the problems.

CO2: Build cryptosystems by applying symmetric and public key encryption algorithms.

CO3: Construct code for authentication algorithms.

CO4: Develop a signature scheme using Digital signature standard.

CO5: Demonstrate the network security system using open source tools

ANNA UNIVERSITY CHENNAI
REGULATION-2017

IT8761 SECURITY LABORATORY

LIST OF EXPERIMENTS

1. Perform encryption, decryption using the following substitution techniques
(i) Ceaser cipher, (ii) playfair cipher iii) Hill Cipher iv) Vigenere cipher
2. Perform encryption and decryption using following transposition techniques
i) Rail fence ii) row & Column Transformation
3. Apply DES algorithm for practical applications.
4. Apply AES algorithm for practical applications.
5. Implement RSA Algorithm using HTML and JavaScript
6. Implement the Diffie-Hellman Key Exchange algorithm for a given problem.
7. Calculate the message digest of a text using the SHA-1 algorithm.
8. Implement the SIGNATURE SCHEME – Digital Signature Standard.
9. Demonstrate intrusion detection system (ids) using any tool eg. Snort or any other s/w.
10. Automated Attack and Penetration Tools Exploring N-Stalker, a Vulnerability Assessment Tool
11. Defeating Malware
i) Building Trojans ii) Rootkit Hunter

TOTAL: 60 PERIODS

LIST OF HARDWARE REQUIREMENTS & SOFTWARE REQUIREMENTS
SOFTWARE REQUIREMENTS

- C
- C++
- Java or equivalent compiler GnuPG
- KF Sensor or Equivalent
- Snort
- Net Stumbler or Equivalent

HARDWARE REQUIREMENTS

- Standalone desktops (or) Server supporting 30 terminals or more

TOTAL: 45 PERIODS

References

Reference Books

1. Cryptography and Network Security Principles and Practices, Fourth Edition By William Stallings
2. “Understanding Cryptography: A Textbook for Students and Practitioners” by Christof Paar and Jan Pelzl.
3. “Information Security: A Complete Guide to IT Security” by Rajat Khare

Ex.No1.a. Encryption and Decryption using Caesar Cipher

Aim

To encrypt the given message and decrypt the cipher text using Caesar cipher.

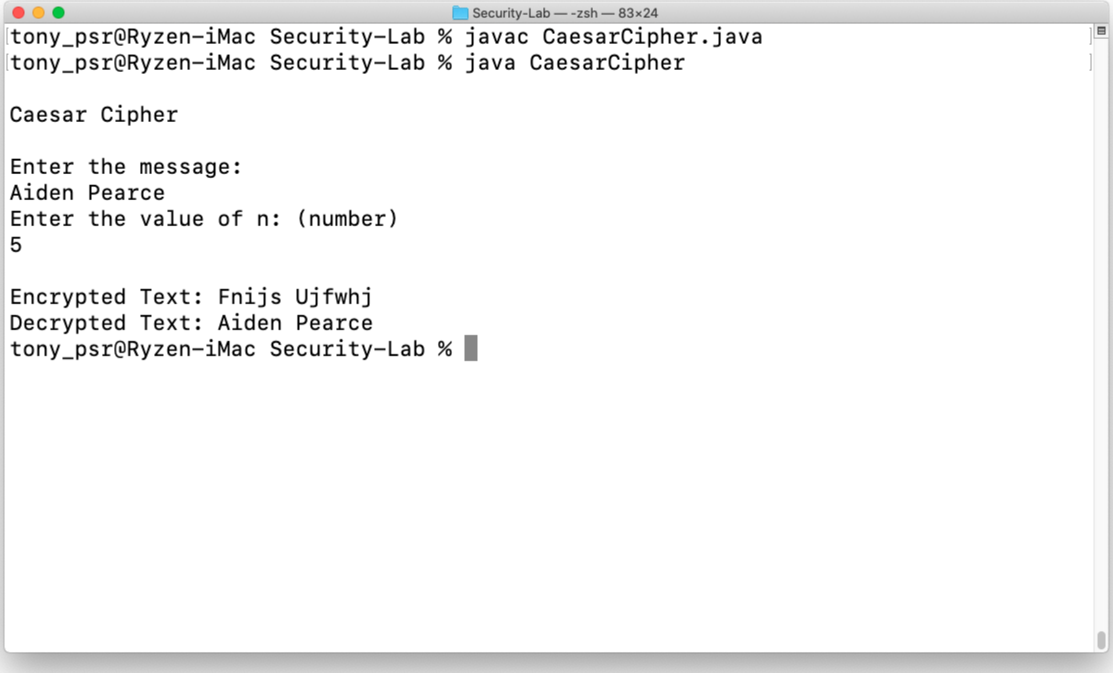
Algorithm

```
1. Read the message string and value of ‘n’ from the user.
2. For encryption,
    a. Increate each character value by n.
    b. If the value is greater than 26 (for Z), reset the value to 0 (A) and resume count.
3. For decryption,
    a. Decrease each character value by n.
    b. If the value is less than 0 (for A), reset the value to 26 (Z) and resume count.
4. Print the plain text and cipher text.
```

Source Code

```
import java.util.Scanner;

public class CaesarCipher {

   public static char[] encrypt(char[] plainText, int n){
char[] cipherText = new char[plainText.length];
for(int i=0; i<plainText.length; i++){
            int offset = Character.isUpperCase(plainText[i])?65:97;
            if(plainText[i] != ' ') {
cipherText[i] = (char) ((plainText[i] + n - offset) % 26 + offset);
            } else {
cipherText[i] = ' ';
            }
        }

        return cipherText;
   }

   public static char[] decrypt(char[] cipherText, int n){
char[] plainText = new char[cipherText.length];
for(int i=0; i<cipherText.length; i++){
            int offset = Character.isUpperCase(cipherText[i])?65:97;
            if(cipherText[i] != ' ') {
plainText[i] = (char) ((cipherText[i] + 26 - n - offset) % 26 + offset);
            } else {
plainText[i] = ' ';
            }
        }

        return plainText;
   }

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

char[] message;
        int n;
System.out.println("\nCaesar Cipher\n");

        //creating char array from string input
System.out.println("Enter the message: ");
        message = sc.nextLine().toCharArray();
System.out.println("Enter the value of n: (number)");
        n = sc.nextInt();

System.out.println();

        //encryption - input plain text
        String cipherText = String.valueOf(encrypt(message, n));
System.out.println("Encrypted Text: " + cipherText);

        //decryption - input encrypted cipher text
        String plainText = String.valueOf(decrypt(cipherText.toCharArray(), n));
System.out.println("Decrypted Text: " + plainText);

    }
}
```

Output



```
tony_psr@Ryzen-iMac Security-Lab % javac CaesarCipher.java
tony_psr@Ryzen-iMac Security-Lab % java CaesarCipher

Caesar Cipher

Enter the message:
Aiden Pearce
Enter the value of n: (number)
5

Encrypted Text: Fnijs Ujfwhj
Decrypted Text: Aiden Pearce
tony_psr@Ryzen-iMac Security-Lab % 
```

Conclusion

Thus, the program to encrypt the given plain text and decrypt the cipher text using Caesar cipher has been written, executed and output is also verified.

Ex.No1.b. Encryption and Decryption using Playfair Cipher

Aim

To encrypt the given message and decrypt the cipher text using Playfair cipher.

Algorithm

1. Read the message string and key string from the user.
2. Replace all ‘j’ in the message string and key string with ‘I’.
3. Generate the key matrix from the key string.
4. For encryption,
   a. Split the message into pairs of two characters.
   b. If both the characters in the pair are in the same row, but different columns, select the characters to right of character pair in the key matrix.
   c. If both the characters in the pair are in the same column, but different rows, select the characters to the bottom of the character pair in the key matrix.
   d. If both the characters in the pairarein different row and column, then assume the pair as atwo opposite corners of a rectangle in the key matrix. Select the characters horizontally opposite to each character in the character pair.
   e. The string formed with the selected characters form the cipher text.

5. For decryption,
    a. Split the message into pairs of two characters.
    b. If both the characters in the pair are in the same row, but different columns, select the characters to left of character pair in the key matrix.
    c. If both the characters in the pair are in the same column, but different rows, select the characters to the top of the character pair in the key matrix.
    d. If both the characters in the pair are in different row and column, then assume the pair as a two opposite corners of a rectangle in the key matrix. Select the characters horizontally opposite to each character in the character pair.
    e. The string formed with the selected characters form the cipher text.
6. Print the plain text and cipher text.

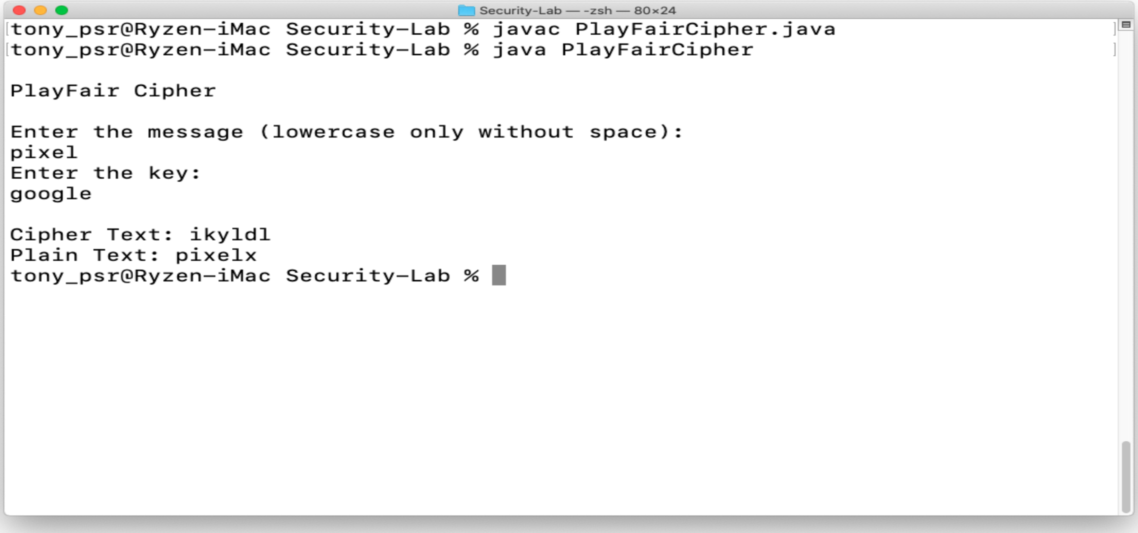
Source Code

```
import java.util.*;

public class PlayFairCipher {

    public static char[][] generateAndGetKeySquare(String key){
char[][] keyTable = new char[5][5];

        // for quicker access
        HashSet<Character>keySet = new HashSet<>();
for(char letter: key.toCharArray()) {
keySet.add(letter);
        }


        // for avoiding repetition
        HashSet<Character>letterSet = new HashSet<>();

        int row = 0;
        int col = 0;

for(int i=0; i<key.length(); i++){
            // to avoid duplication
if(letterSet.contains(key.charAt(i))) continue;

if(col == 5){
                row++;
                col = 0;
            }

            if(key.charAt(i) == 'j'){
                continue;
            }

keyTable[row][col] = key.charAt(i);
letterSet.add(key.charAt(i));
            col++;
        }

for(int i=0; i<26; i++) {
            // to avoid duplication
if(letterSet.contains('a'+i)) continue;

if(col == 5){
                row++;
                col = 0;
            }

if('a'+i == 'j'){
                continue;
            }

            if(!keySet.contains((((char)('a'+i)))){
keyTable[row][col] = (char) ('a'+i);
letterSet.add((char) ('a'+i));
                col++;
            }
        }

        return keyTable;
    }

    public static ArrayList<String>splitMessage(String message){
ArrayList<String>splittedMessage = new ArrayList<>();

        int duplicates=0;
for(int i=0; i<message.length()-1; i+=2){
            if((message.charAt(i) != message.charAt(i+1))){
splittedMessage.add(message.charAt(i) + "" + message.charAt(i+1));
            } else if(!(message.charAt(i) == 'x' &&message.charAt(i) == message.charAt(i))) {
splittedMessage.add(message.charAt(i) + "" + 'x');
i-=1;
                duplicates+=1;
            }
        }
        // if length of key is odd
        if((message.length() + duplicates )%2 != 0) {
splittedMessage.add("" + message.charAt(message.length() - 1) + "x");
        }

        return splittedMessage;
    }

    public static String encrypt(String plainText, String key){
        StringBuilder cipherText = new StringBuilder();
char[][] keyTable = generateAndGetKeySquare(key);

ArrayList<String>splittedPlainText = splitMessage(plainText);

for(String letterPair: splittedPlainText){
            char firstLetter = letterPair.charAt(0)=='j'?'i':letterPair.charAt(0);
            char secondLetter = letterPair.charAt(1)=='j'?'i':letterPair.charAt(1);
int[] firstLetterIndex = new int [2];
int[] secondLetterIndex = new int[2];


for(int i=0; i<5; i++){
for(int j=0; j<5; j++){
                    if(keyTable[i][j] == firstLetter){
firstLetterIndex[0] = i;
firstLetterIndex[1] = j;
                    } else if(keyTable[i][j] == secondLetter){
secondLetterIndex[0] = i;
secondLetterIndex[1] = j;
```

```
               }
            }
         }

         //Cases
         // same row, same column, diff row and column
if(firstLetter == secondLetter){
            cipherText.append(keyTable[firstLetterIndex[0]][(firstLetterIndex[1] + 1) % 5])
.append(keyTable[firstLetterIndex[0]][(firstLetterIndex[1] + 1) % 5]);
}else if(firstLetterIndex[0] == secondLetterIndex[0]){
            cipherText.append(keyTable[firstLetterIndex[0]][(firstLetterIndex[1] + 1) % 5])
.append(keyTable[secondLetterIndex[0]][(secondLetterIndex[1] + 1) % 5]);
         } else if (firstLetterIndex[1] == secondLetterIndex[1]){
            cipherText.append(keyTable[(firstLetterIndex[0]+1)%5][(firstLetterIndex[1])])
.append(keyTable[(secondLetterIndex[0]+1)%5][(secondLetterIndex[1])]);
         } else {
            cipherText.append(keyTable[firstLetterIndex[0]][(secondLetterIndex[1])])
.append(keyTable[secondLetterIndex[0]][(firstLetterIndex[1])]);
         }

      }

      return cipherText.toString();
   }

   public static String decrypt(String cipherText, String key){
      StringBuilder plainText = new StringBuilder();
char[][] keyTable = generateAndGetKeySquare(key);

ArrayList<String>splittedPlainText = splitMessage(cipherText);

for(String letterPair: splittedPlainText){
         char firstLetter = letterPair.charAt(0)=='j'?'i':letterPair.charAt(0);
         char secondLetter = letterPair.charAt(1)=='j'?'i':letterPair.charAt(1);
int[] firstLetterIndex = new int [2];
int[] secondLetterIndex = new int[2];

for(int i=0; i<5; i++){
for(int j=0; j<5; j++){
               if(keyTable[i][j] == firstLetter){
firstLetterIndex[0] = i;
firstLetterIndex[1] = j;
               } else if(keyTable[i][j] == secondLetter){
secondLetterIndex[0] = i;
secondLetterIndex[1] = j;
               }
            }
         }

         //Cases
         // same row, same column, diff row and column
         if(firstLetterIndex[0] == secondLetterIndex[0]){
            plainText.append(keyTable[firstLetterIndex[0]][(5+firstLetterIndex[1] - 1) % 5])
.append(keyTable[secondLetterIndex[0]][(5+secondLetterIndex[1] - 1) % 5]);
         } else if (firstLetterIndex[1] == secondLetterIndex[1]){
            plainText.append(keyTable[(5+firstLetterIndex[0]-1)%5][(firstLetterIndex[1])])
.append(keyTable[(5+secondLetterIndex[0]-1)%5][(secondLetterIndex[1])]);
         } else {
            plainText.append(keyTable[firstLetterIndex[0]][(secondLetterIndex[1])])
```

```
.append(keyTable[secondLetterIndex[0]][(firstLetterIndex[1])]);
            }

        }

        return plainText.toString();
    }

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

System.out.println("\nPlayFair Cipher\n");

System.out.println("Enter the message (lowercase only without space):");
        String message = sc.next().toLowerCase().replaceAll("j","i");
System.out.println("Enter the key: (a-z)");
        // replace all j with i
        String key = sc.next().toLowerCase().replaceAll("j","i");
System.out.println();

        String cipherText = encrypt(message, key);
System.out.println("Cipher Text: " + cipherText);
        String plainText = decrypt(cipherText, key);
System.out.println("Plain Text: " + plainText);
    }
}
```

Output



```
tony_psr@Ryzen-iMac Security-Lab % javac PlayFairCipher.java
tony_psr@Ryzen-iMac Security-Lab % java PlayFairCipher

PlayFair Cipher

Enter the message (lowercase only without space):
pixel
Enter the key:
google

Cipher Text: ikyldl
Plain Text: pixelx
tony_psr@Ryzen-iMac Security-Lab % 
```

Conclusion

Thus, the program to encrypt the given plain text and decrypt the cipher text using Playfair cipher has been written, executed and output is also verified.

Ex.No1.c. Encryption and Decryption using Hill Cipher

Aim

To encrypt the given message and decrypt the cipher text using Hill cipher.

Algorithm

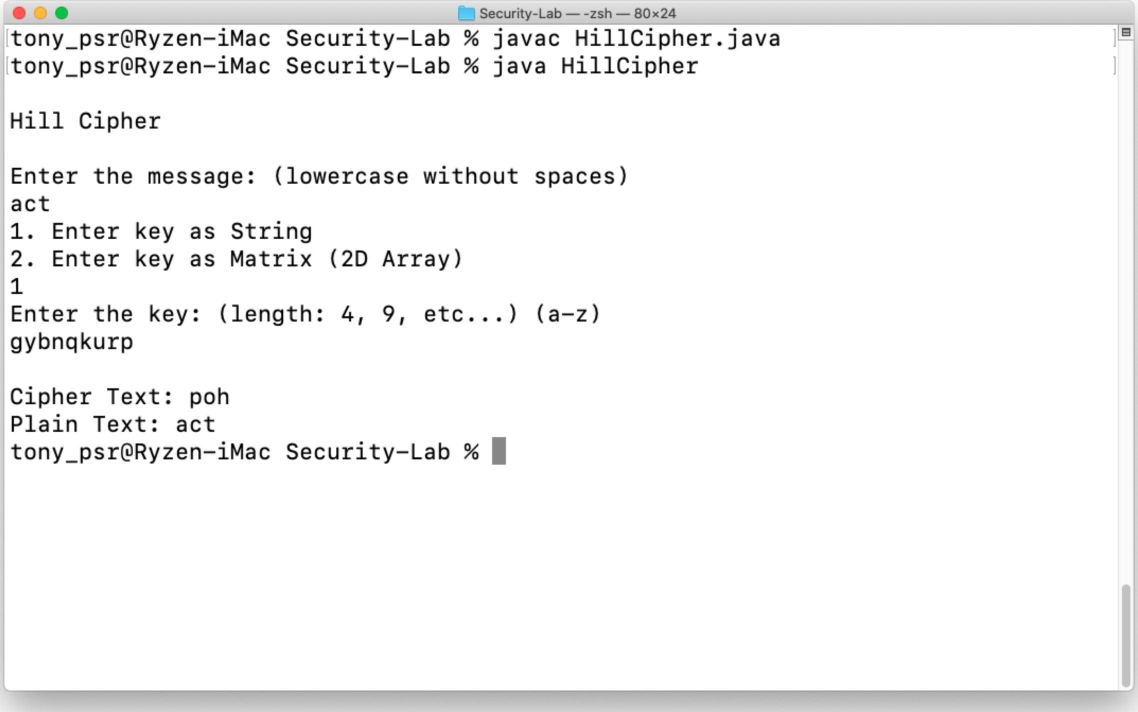
1. Read the message string and key string from the user.
2. Convert the message string to matrix and convert the key string to array vector.
3. Compute the cipher text using the formula,
    a. C = KP (mod26)
4. Compute the plain text using the formula,
    a. P = (K^-1)C (mod26)
5. Print the plain text and cipher text.

Source Code

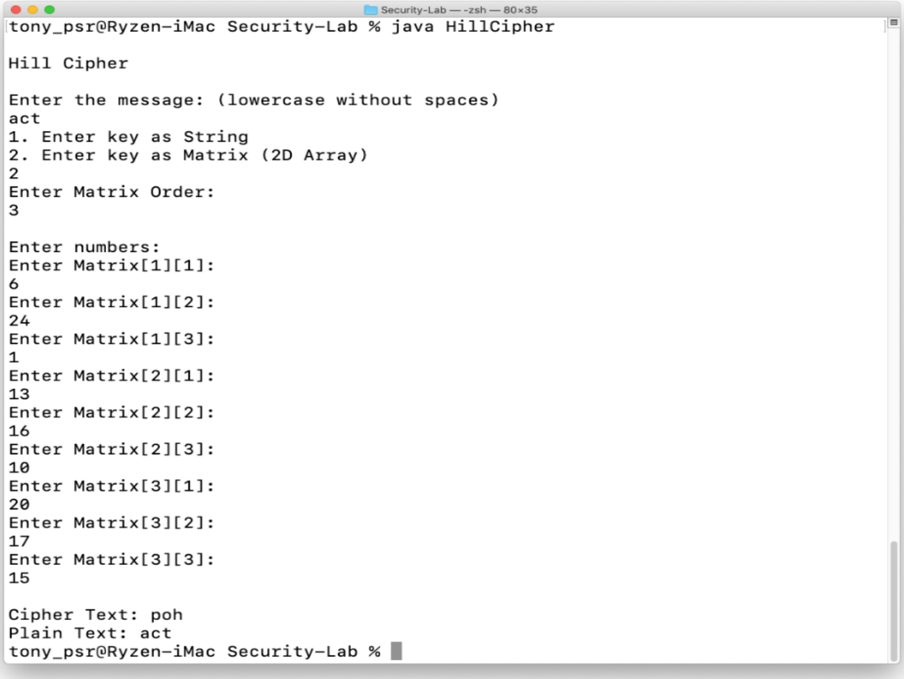
```
import java.util.ArrayList;
import java.util.Scanner;

public class HillCipher {

   public static ArrayList<String>splitMessage(String message, int splitBy) {
ArrayList<String>splittedMessage = new ArrayList<>();

       for (int i = 0; i<message.length() - splitBy + 1; i += splitBy) {
splittedMessage.add(message.substring(i, i + splitBy));
       }

       return splittedMessage;
   }


   public static int[][] multiply(int[][] a, int[][] b) {
int[][] c = new int[a.length][b[0].length];

       for (int i = 0; i<a.length; i++) {
          for (int j = 0; j <b[0].length; j++) {
             for (int k = 0; k <b.length; k++) {
                c[i][j] += (a[i][k] * b[k][j]);
             }
          }
       }

       return c;
   }

   public static int[][] stringToColumnVector(String text) {
int[][] colVector = new int[text.length()][1];
```

```
        for (int i = 0; i<text.length(); i++) {
colVector[i][0] = text.charAt(i) - 'a';
        }

        return colVector;
    }

    public static String encrypt(String plainText, int[][] key) {
        StringBuilder cipherText = new StringBuilder();

ArrayList<String>messageSplit = splitMessage(plainText, key.length);


        for (String textBlock :messageSplit) {
int[][] c = multiply(key, stringToColumnVector(textBlock));

            for (int i = 0; i<c.length; i++) {
cipherText.append((char) (c[i][0] % 26 + 'a'));
            }
        }

        return cipherText.toString();
    }

    public static booleanisValidKey(int[][] key){
        int determinant = determinant(key, key.length);

        if (determinant == 0) {
System.out.println("Key can't be inverted, determinant 0. Try another key");
            return false;
        }

        //prelims check
        if (Math.abs(gcd(determinant, 26)) != 1) {
System.out.println("Key doesn't have a modular inverse, try another key");
            return false;
        }

        return true;
    }

    public static String decrypt(String cipherText, int[][] key) {
        StringBuilder plainText = new StringBuilder();

ArrayList<String>messageSplit = splitMessage(cipherText, key.length);

        for (String textBlock :messageSplit) {
int[][] adjMatrix = adj(key);
            int determinant = determinant(key, key.length);

            if(!isValidKey(key)){
System.exit(0);
            }


int[][] c = multiply(adjMatrix, stringToColumnVector(textBlock));

            //finding suitable k such that
            // (det*k)mod26 === 1
            int k = 1;
            while ((determinant * k) % 26 != 1) {
                k++;
            }

            for (int i = 0; i<c.length; i++) {
                c[i][0] *= k;
plainText.append((char) (Math.abs(c[i][0]) % 26 + 'a'));
            }

        }

        return plainText.toString();
    }


    static int gcd(int n1, int n2) {
        if (n2 != 0)
            return gcd(n2, n1 % n2);
        else
            return n1;
    }

    static void getCofactor(int[][] A, 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++;
                    }
                }
            }
        }
    }

    static int determinant(int[][] A, int n) {
        int result = 0;

        if (n == 1)
            return A[0][0];

int[][] temp = new int[A.length][A.length];

        int sign = 1;

        for (int f = 0; f < n; f++) {
getCofactor(A, temp, 0, f, n);
            result += sign * A[0][f] * determinant(temp, n - 1);
            sign = -sign;
        }

        return result;
    }
    static int[][] adj(int A[][]) {
int[][] adj = new int[A.length][A.length];
        if (A.length == 1) {
adj[0][0] = 1;
            return adj;
        }

        int sign = 1;
int[][] temp = new int[A.length][A.length];

        for (int i = 0; i<A.length; i++) {
            for (int j = 0; j <A.length; j++) {
getCofactor(A, temp, i, j, A.length);
                sign = ((i + j) % 2 == 0) ?1 : -1;
                adj[j][i] = (sign) * (determinant(temp, A.length - 1));
            }
        }

        return adj;
    }

    public static int[][] stringToIntArray(String message) {
        int n = (int) Math.sqrt(message.length());
int[][] key = new int[n][n];

        int stringPtr = 0;
        for (int i = 0; i< n; i++) {
            for (int j = 0; j < n; j++) {
                key[i][j] = message.charAt(stringPtr) - 'a';
stringPtr += 1;
            }
        }

        return key;
    }


    public static void main(String[] args) {
        /* May fail for edge cases */
        /* Additional testing required */

        Scanner sc = new Scanner(System.in);

System.out.println("\nHill Cipher\n");

System.out.println("Enter the message: (lowercase without spaces)");
        String message = sc.next().toLowerCase();

int[][] key = null;

System.out.println("1. Enter key as String\n2. Enter key as Matrix (2D Array)");
        int choice = sc.nextInt();
        if (choice == 1) {
System.out.println("Enter the key: (length: 4, 9, etc...) (a-z)");
            String keyString = sc.next().toLowerCase();

            // if length of the key is not a square of an integer.
            if (Math.sqrt(keyString.length()) != Math.round(Math.sqrt(keyString.length()))) {
System.out.println("Invalid key length");
System.exit(0);
            }

            key = stringToIntArray(keyString);

        } else if (choice == 2) {
System.out.println("Enter Matrix Order: ");
            int order = sc.nextInt();
            key = new int[order][order];

System.out.println("\nEnter numbers:");

            for (int i = 0; i< order; i++) {
                for (int j = 0; j < order; j++) {
System.out.println("Enter Matrix[" + (i+1) + "][" + (j+1) + "]: ");
                    key[i][j] = sc.nextInt();
                }
            }
        } else {
System.out.println("Invalid choice");
        }
System.out.println();

        String cipherText = encrypt(message, key);
System.out.println("Cipher Text: " + cipherText);
        String plainText = decrypt(cipherText, key);
System.out.println("Plain Text: " + plainText);

    }
}
```

Output



```
tony_psr@Ryzen-iMac Security-Lab % javac HillCipher.java
tony_psr@Ryzen-iMac Security-Lab % java HillCipher

Hill Cipher

Enter the message: (lowercase without spaces)
act
1. Enter key as String
2. Enter key as Matrix (2D Array)
1
Enter the key: (length: 4, 9, etc...) (a-z)
gybnqkurp

Cipher Text: poh
Plain Text: act
tony_psr@Ryzen-iMac Security-Lab % 
```



```
tony_psr@Ryzen-iMac Security-Lab % java HillCipher

Hill Cipher

Enter the message: (lowercase without spaces)
act
1. Enter key as String
2. Enter key as Matrix (2D Array)
2
Enter Matrix Order:
3

Enter numbers:
Enter Matrix[1][1]:
6
Enter Matrix[1][2]:
24
Enter Matrix[1][3]:
1
Enter Matrix[2][1]:
13
Enter Matrix[2][2]:
16
Enter Matrix[2][3]:
10
Enter Matrix[3][1]:
20
Enter Matrix[3][2]:
17
Enter Matrix[3][3]:
15

Cipher Text: poh
Plain Text: act
tony_psr@Ryzen-iMac Security-Lab % 
```

Conclusion

Thus, the program to encrypt the given plain text and decrypt the cipher text using Hill cipher has been written, executed and output is also verified.

Ex.No.1.d.Encryption and Decryption using Vigenère Cipher

Aim

To encrypt the given message and decrypt the cipher text using Vigenère cipher.

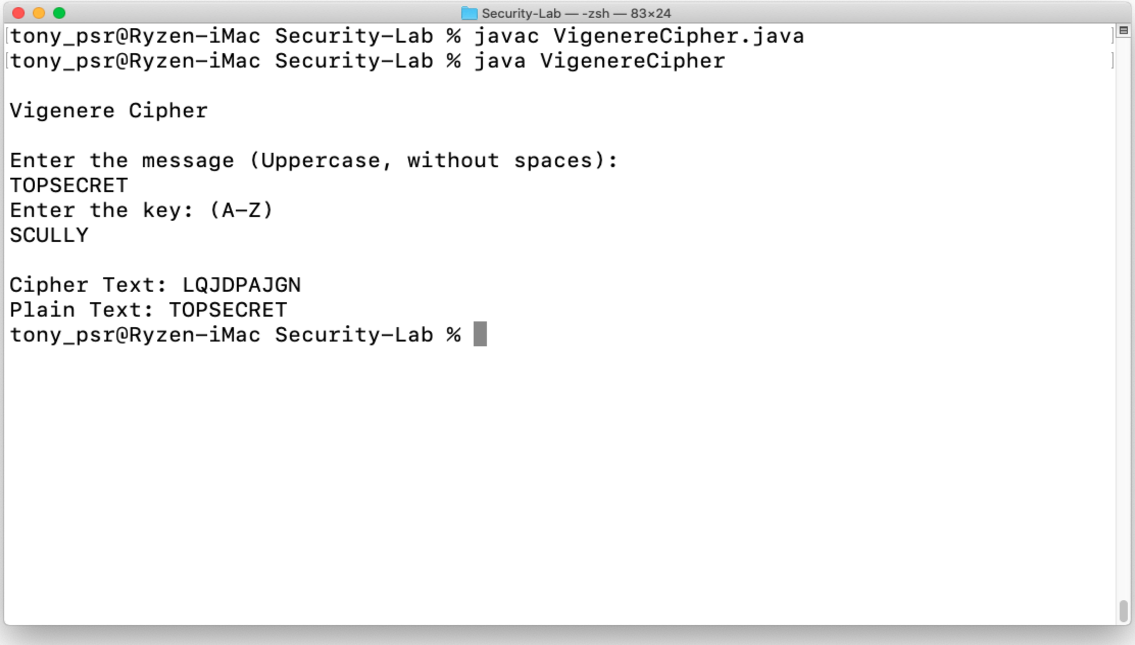
Algorithm

1. Read the message string and message string from the user.
2. For encryption,
   a. Select the characters in the Vigenère table corresponding to the plain text and key in row and column, respectively.
   b. Append the selected characters to the cipher text string.
3. For decryption,
   a. Select the characters in the Vigenère table corresponding to the key in row and cipher text in column.
   b. Append the selected characters to the plain text string.
4. Print the plain text and cipher text.

Source Code

```
import java.util.Scanner;

public class VigenereCipher {
   public static char[][] generateAndGetVigenereTable(){
char[][] table = new char[26][26];

      int offset = 0;

for(int i=0; i<26; i++){
for(int j=0; j<26; j++){
              table[i][j] = (char) ((j+offset)%26 +65);
          }
          offset++;
      }

      return table;
   }

   public static String encrypt(char[] plainText, char[] key){
      StringBuilder cipherText = new StringBuilder();
char[][] table = generateAndGetVigenereTable();

for(int i=0; i<plainText.length; i++){
          int row = plainText[i] - 65;
          int col = key[i%key.length] - 65;

cipherText.append(table[row][col]);
      }

      return cipherText.toString();
   }

   public static String decrypt(char[] cipherText, char[] key){
      StringBuilder plainText = new StringBuilder();
char[][] table = generateAndGetVigenereTable();

for(int i=0; i<cipherText.length; i++){
          int row = key[i%key.length] - 65;
          int col = 0;
for(int j=0; j<table[0].length; j++){
              if(table[row][j] == cipherText[i]){
                  col = j;
                  break;
              }
          }

plainText.append((char)(col + 65));
      }
      return plainText.toString();
  }

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

System.out.println("\nVigenere Cipher\n");

System.out.println("Enter the message (Uppercase, without spaces):");
      String message = sc.next().toUpperCase();
System.out.println("Enter the key: (A-Z)");
      String key = sc.next().toUpperCase();

System.out.println();

      String cipherText = encrypt(message.toCharArray(), key.toCharArray());
System.out.println("Cipher Text: " + cipherText);
      String plainText = decrypt(cipherText.toCharArray(), key.toCharArray());
System.out.println("Plain Text: " + plainText);

  }
}
```

Output



```
tony_psr@Ryzen-iMac Security-Lab % javac VigenereCipher.java
tony_psr@Ryzen-iMac Security-Lab % java VigenereCipher

Vigenere Cipher

Enter the message (Uppercase, without spaces):
TOPSECRET
Enter the key: (A-Z)
SCULLY

Cipher Text: LQJDPAJGN
Plain Text: TOPSECRET
tony_psr@Ryzen-iMac Security-Lab %
```

Conclusion

Thus, the program to encrypt the given plain text and decrypt the cipher text using Vigenère cipher has been written, executed and output is also verified.

Ex.No.2a.Encryption and Decryption using Rail Fence Cipher

Aim

To encrypt the given message and decrypt the cipher text using Rail Fence cipher.

Algorithm

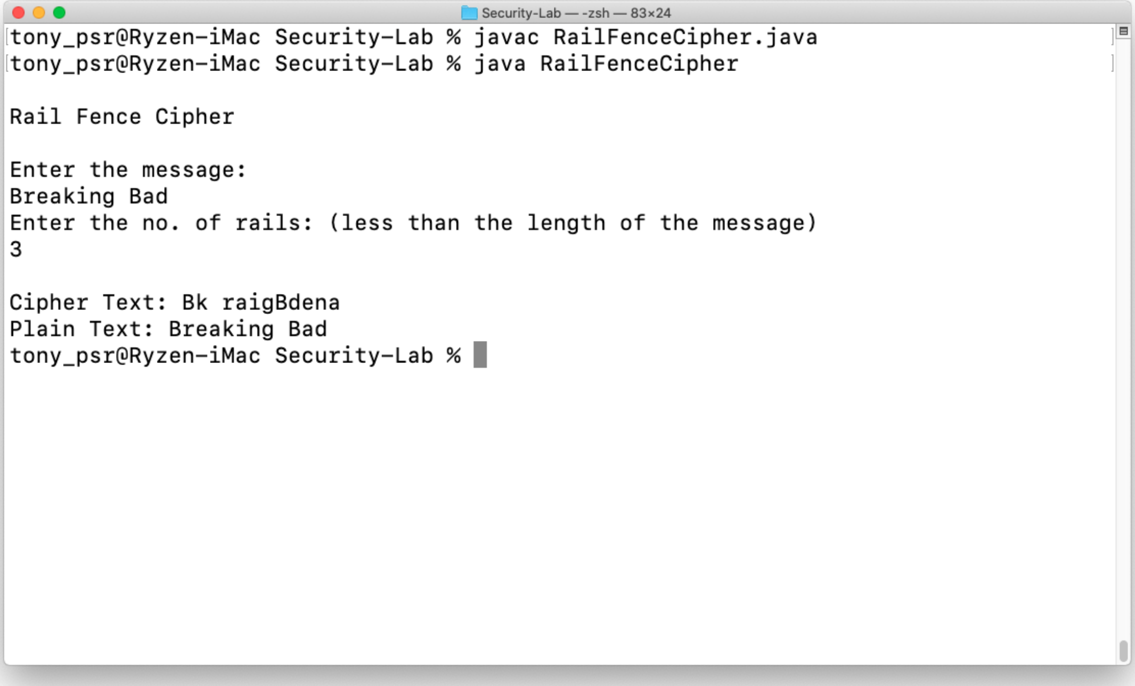
1. Read the message string and number of rails ‘n’ from the user.
2. For encryption,
    a. Create rails matrix with n rows and no. of columns equal to the length of the plain text.
    b. Place the plain text in the matrix in a zig zag manner from left to right and append it to cipher text.
3. For decryption,
    a. Create rails matrix with n rows and no. of columns equal to the length of the plain text.
    b. Recreate the matrix resulted in step 2.b. using the cipher text in rails matrix.
    c. From the matrix, extract the plain text.
4. Print the plain text and cipher text.

Source Code

```
import java.util.Arrays;
import java.util.Scanner;

public class RailFenceCipher {
   public static String encrypt(char[] plainText, int n){
      StringBuilder cipherText = new StringBuilder();

char[][] rails = new char[n][plainText.length];

      for(char[] rail: rails){
Arrays.fill(rail, '#');
      }

      int row = 0;
      int col = 0;
      int rowOffset = -1;


      //Creating rails table
      for (char letter :plainText) {
         if (row == n - 1 || row == 0) {
rowOffset = -rowOffset;
         }
         rails[row][col] = letter;
         row += rowOffset;
         col += 1;
      }

      //Uncomment the below 2 commented lines to view the rails
for(int i=0; i<rails.length; i++){
for(int j=0; j<rails[0].length; j++){
//            System.out.print(rails[i][j] + " ");
            if(rails[i][j]!='#')
cipherText.append(rails[i][j]);
         }
//          System.out.println();
      }
```

```
        return cipherText.toString();
    }

    public static String decrypt(char[] cipherText, int n){
        StringBuilder plainText = new StringBuilder();

char[][] rails = new char[n][cipherText.length];

        for(char[] rail: rails){
Arrays.fill(rail, '#');
        }

        int row = 0;
        int col = 0;
        int rowOffset = -1;

        //Creating rails table
        for (int i=0; i<cipherText.length; i++) {
            if (row == n - 1 || row == 0) {
rowOffset = -rowOffset;
            }
            rails[row][col] = '_';
            row += rowOffset;
            col += 1;
        }

        int x = 0;
for(int i=0; i<rails.length; i++){
for(int j=0; j<rails[0].length; j++) {
                if (rails[i][j] == '_') {
                    rails[i][j] = cipherText[x];
                    x++;
                }
            }
        }

        //Uncomment the below 2 commented lines to view the rails
//      System.out.println("Decryption Rails:");
for(int i=0; i<rails.length; i++){
for(int j=0; j<rails[0].length; j++){
//              System.out.print(rails[i][j] + " ");
            }
//          System.out.println();
        }

        row = 0;
        col = 0;
rowOffset = -1;

        //Creating rails table
        for (int i=0; i<cipherText.length; i++) {
            if (row == n - 1 || row == 0) {
rowOffset = -rowOffset;
            }
plainText.append(rails[row][col]);
            row += rowOffset;
            col += 1;
        }

        return plainText.toString();
    }

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

System.out.println("\nRail Fence Cipher\n");

System.out.println("Enter the message: ");
char[] message = sc.nextLine().toCharArray();
System.out.println("Enter the no. of rails: (less than the length of the message)");
        int n = sc.nextInt();

System.out.println();

        String cipherText = encrypt(message, n);
System.out.println("Cipher Text: " + cipherText);

        String plainText = decrypt(cipherText.toCharArray(), n);
System.out.println("Plain Text: " + plainText);


    }
}
```

Output



```
tony_psr@Ryzen-iMac Security-Lab % javac RailFenceCipher.java
tony_psr@Ryzen-iMac Security-Lab % java RailFenceCipher

Rail Fence Cipher

Enter the message:
Breaking Bad
Enter the no. of rails: (less than the length of the message)
3

Cipher Text: Bk raigBdena
Plain Text: Breaking Bad
tony_psr@Ryzen-iMac Security-Lab %
```

Conclusion

Thus, the program to encrypt the given plain text and decrypt the cipher text using Rail fence cipher has been written, executed and output is also verified.

Ex. No : 2(b) Row and Column Transformation Technique

AIM:
To implement a program for encryption and decryption by using row and column transformation technique.

ALGORITHM:
1.Consider the plain text hello world, and let us apply the simple columnartransposition technique as shown below

H	e	l	l
o	w	o	r
l	d		

2. The plain text characters are placed horizontally and the cipher text iscreated with vertical format as: holewdlolr.
3. Now, the receiver has to use the same table to decrypt the cipher text toplain text.

PROGRAM:
TransCipher.java

```
import java.util.*;
class TransCipher {
 public static void main(String args[]) {
Scanner sc = new Scanner(System.in);
System.out.println("Enter the plain text");
String pl = sc.nextLine();
sc.close();
 String s = "";
 int start = 0;
 for (int i = 0; i<pl.length(); i++) {
 if (pl.charAt(i) == ' ') {
 s = s + pl.substring(start, i);
 start = i + 1;
 }
 }
 s = s + pl.substring(start);
System.out.print(s);
System.out.println();
 // end of space deletion
 int k = s.length();
 int l = 0;
 int col = 4;
 int row = s.length() / col;
 char ch[][] = new char[row][col];
 for (int i = 0; i< row; i++) {
 for (int j = 0; j < col; j++) {
 if (l < k) {
ch[i][j] = s.charAt(l);
 l++;
 } else {
ch[i][j] = '#';
 }
 }
 }
 // arranged in matrix
 char trans[][] = new char[col][row];
 for (int i = 0; i< row; i++) {
 for (int j = 0; j < col; j++) {
 trans[j][i] = ch[i][j];
 }
 }
 for (int i = 0; i< col; i++) {
for (int j = 0; j < row; j++) {
System.out.print(trans[i][j]);
 }
 }
 // display
System.out.println();
 }
}
```

OUTPUT:

```
Enter the plain text
Security Lab
SecurityLab
Sreictuy
```

Conclusion:

Thus the java program for Row and Column Transposition Technique has been implemented and the output verified successfully.

Ex.No.3. Apply DES Algorithm for Practical Applications

Aim

To implement DES algorithm for practical applications.

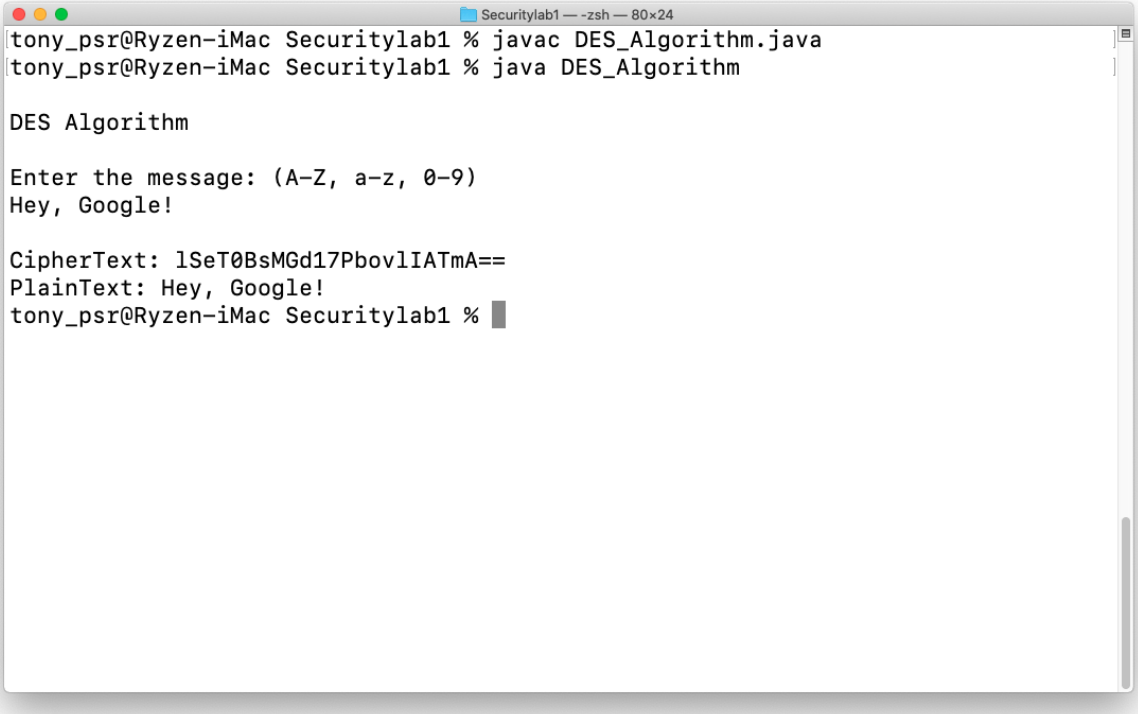
Algorithm

1. Import the necessary libraries.
2. Get the message string from the user.
3. Generate a secret key with “DES” instance of KeyGenerator.
4. Instantiate two ciphers for encryption and decryption in “DES” mode.
5. For encryption,
    a. Convert the message to byte array.
    b. Using doFinal() methon in cipher to get encoded bytes.
    c. Convert it to BASE64 and get the cipher text.
6. For decryption,
    a. Get he byte array from cipher text.
    b. Using doFinal(), get message bytes.
    c. Convert it to UTF-8 format and get the plain text.
7. Print the plain text and cipher text.

Source Code

```
package com.company.tonypsr.cryptographic_algorithms;

import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import java.nio.charset.StandardCharsets;
import java.util.Base64;
import java.util.Scanner;


public class DES_Algorithm {
    private static Cipher encryptionCipher;
    private static Cipher decryptionCipher;

    public static void initSecretKey(SecretKey key) throws Exception {
encryptionCipher = Cipher.getInstance("DES");
decryptionCipher = Cipher.getInstance("DES");
encryptionCipher.init(Cipher.ENCRYPT_MODE, key);
decryptionCipher.init(Cipher.DECRYPT_MODE, key);
    }

    public static String encrypt(String str) throws Exception {
byte[] messageBytes = str.getBytes(StandardCharsets.UTF_8);
byte[] encodedBytes = encryptionCipher.doFinal(messageBytes);
        return new String(Base64.getEncoder().encode(encodedBytes));
    }

    public static String decrypt(String str) throws Exception {
byte[] decodedBytes = Base64.getDecoder().decode(str);
byte[] messageBytes = decryptionCipher.doFinal(decodedBytes);
        return new String(messageBytes, StandardCharsets.UTF_8);
    }

    public static void main(String[] argv) {
        try {
SecretKey key = KeyGenerator.getInstance("DES").generateKey();
initSecretKey(key);

            Scanner sc = new Scanner(System.in);

System.out.println("\nDES Algorithm\n");
System.out.println("Enter the message: (A-Z, a-z, 0-9)");
            String message;
            message = sc.nextLine();

            String encryptedString = "";
            String decryptedString = "";

            try {
encryptedString = encrypt(message);
decryptedString = decrypt(encryptedString);
            } catch (Exception e) {
e.printStackTrace();
            }
System.out.println();

System.out.println("CipherText: " + encryptedString);
System.out.println("PlainText: " + decryptedString);
}catch (Exception e){
e.printStackTrace();
        }
    }
}
```

Output



```
tony_psr@Ryzen-iMac Securitylab1 % javac DES_Algorithm.java
tony_psr@Ryzen-iMac Securitylab1 % java DES_Algorithm

DES Algorithm

Enter the message: (A-Z, a-z, 0-9)
Hey, Google!

CipherText: lSeT0BsMGd17PbovlIATmA==
PlainText: Hey, Google!
tony_psr@Ryzen-iMac Securitylab1 % 
```

Conclusion

Thus, the program to implement DES for practical applications has been written, executed and output is also verified.

Ex.No.4. Apply AES Algorithm for Practical Applications

Aim

To implement AES algorithm for practical applications.

Algorithm

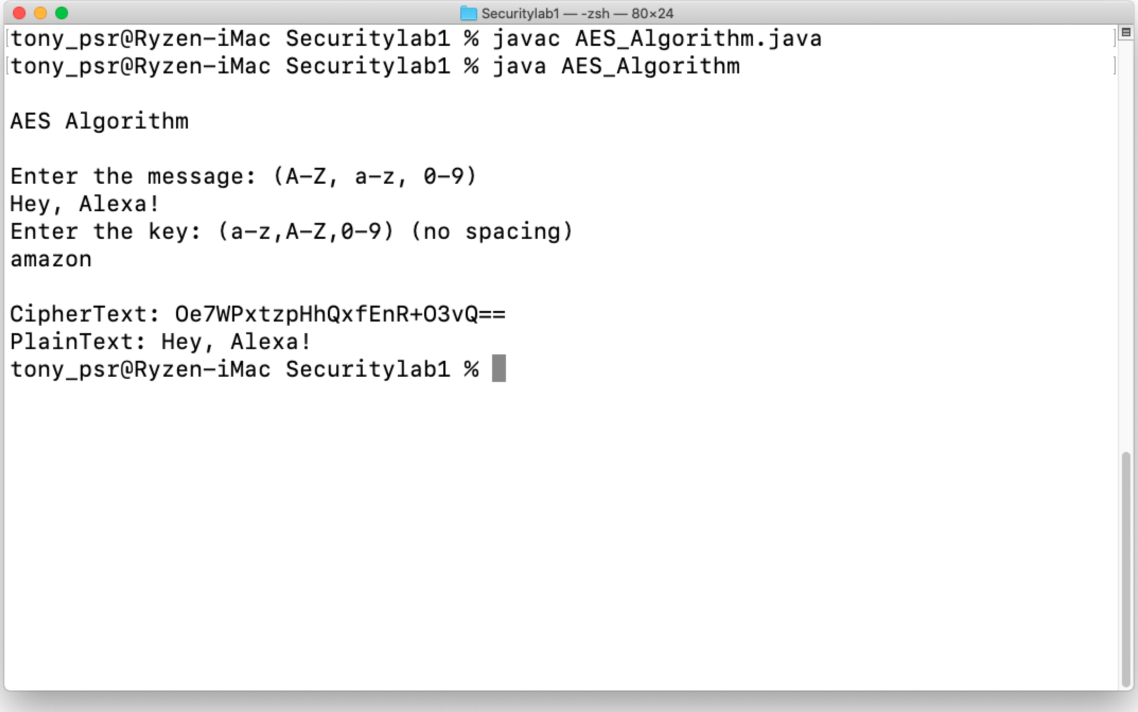
1. Import the necessary libraries.
2. Get the message string from the user.
3. Create a method for initializing key.
    a. Generate a MD5 hash code.
    b. Generate a secret key with the previously generated hashcode using SecretKeySpec object.
4. For encryption,
    a. Initialize the key.
    b. Initialize the cipher in AES/ECB mode.
    c. Using .doFinal() method in cipher, get the encrypted bytes.
    d. Using Base64 class, encode the message in UTF-8 format and return cipher text.
5. For decryption,
    a. Initialize the key.
    b. Initialize the cipher in AES/ECB mode.
    c. Using .doFinal() method in cipher, get the encrypted bytes.
    d. Using Base64 class, decode the encrypted and return plain text.
6. Print the plain text and cipher text.

Source Code

```
package com.company.tonypsr.cryptographic_algorithms;

import java.nio.charset.StandardCharsets;
import java.security.MessageDigest;
import java.util.Arrays;
import java.util.Base64;
import java.util.Scanner;
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;

public class AES_Algorithm {
    private static SecretKeySpecsecretKey;
    private static byte[] key;

    public static void initKey(String myKey) throws Exception {
MessageDigest md5Code = null;
        key = myKey.getBytes(StandardCharsets.UTF_8);
        md5Code = MessageDigest.getInstance("MD5");
        key = md5Code.digest(key);
        key = Arrays.copyOf(key, 16);
secretKey = new SecretKeySpec(key, "AES");
    }

    public static String encrypt(String strToEncrypt, String secret) throws Exception {
initKey(secret);
        Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
cipher.init(Cipher.ENCRYPT_MODE, secretKey);
```

```
        return
Base64.getEncoder().encodeToString(cipher.doFinal(strToEncrypt.getBytes(StandardCharsets.UTF_8)));
    }

    public static String decrypt(String strToDecrypt, String secret) throws Exception {
initKey(secret);
        Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5PADDING");
cipher.init(Cipher.DECRYPT_MODE, secretKey);
        return new String(cipher.doFinal(Base64.getDecoder().decode(strToDecrypt)));
    }

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

System.out.println("\nAES Algorithm\n");


System.out.println("Enter the message: (A-Z, a-z, 0-9)");
        String message;
        message = sc.nextLine();

System.out.println("Enter the key: (a-z,A-Z,0-9) (no spacing)");
        String secretKey;
secretKey = sc.next();

        String encryptedString = "";
        String decryptedString = "";

        try {
encryptedString = encrypt(message, secretKey);
decryptedString = decrypt(encryptedString, secretKey);
}catch (Exception e){
e.printStackTrace();
        }

System.out.println();

System.out.println("CipherText: " + encryptedString);
System.out.println("PlainText: " + decryptedString);
    }
}
```

Output



```
tony_psr@Ryzen-iMac Securitylab1 % javac AES_Algorithm.java
tony_psr@Ryzen-iMac Securitylab1 % java AES_Algorithm

AES Algorithm

Enter the message: (A-Z, a-z, 0-9)
Hey, Alexa!
Enter the key: (a-z,A-Z,0-9) (no spacing)
amazon

CipherText: Oe7WPxtzpHhQxfEnR+O3vQ==
PlainText: Hey, Alexa!
tony_psr@Ryzen-iMac Securitylab1 %
```

Conclusion

Thus, the program to implement AES for practical applications has been written, executed and output is also verified.

Ex.No.5.Implement RSA Algorithm using HTML and JavaScript

Aim

To implement RSA algorithm using HTML, CSS, and JavaScript.

Procedure

1. Create a HTML file defining the structure of the web page.
2. Create a CSS file styling the webpage.
3. Create a JavaScript file with the following algorithm.
    a. Generate or get two unique prime numbers ‘p’ and ‘q’ from the users.
    b. Compute the values of ‘n’ and ‘phi’ as follows
        i. n = p*q
        ii. phi = (p-1)*(q-1)
    c. Find the value of e such that it’s GCD(e,phi) == 1 (relatively prime).
    d. Compute the value of d such that (d * e) mod phi == 1.
    e. Perform Encryption {e,n} using the following formula
        i. C= M^e (mod n)
    f. Perform Decryption {d,n} using the following formula
        i. M = C^d (mod n)

SourceCode

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<link rel="stylesheet" href="style.css">
<script src="algorithm.js"></script>
<title>RSA Algorithm</title>
</head>
<body>
<div class="card">
<h1>RSA Algorithm</h1>
<div class="container">
<input type="number" placeholder="Message (0-99999)" id="message"/><br>
<button onclick="encrypt()">Encrypt</button>
<button onclick="decrypt()">Decrypt</button>
</div>

<div class="p-and-q-values">
<table>
<tr>
<td>
<p>p</p>
</td>
<td>
<input type="number" id="p"/><br>
</td>
</tr>

<tr>
<td>
<p>q</p>
</td>
<td>
<input type="number" id="q"/><br>
</td>
</tr>
</table>
</div>
```

```
<button id="generate-pq-button" onclick="generatePandQ()">Generate new P & Q</button>

<hr>
<h2>Computed Values</h2>
<div class="computed-values">
<table>
<tr>
<td>
<p>n = p*q</p>
</td>
<td>
<input type="number" id="n" disabled/><br>
</td>
</tr>

<tr>
<td>
<p>phi = (p-1)*(q-1)</p>
</td>
<td>
<input type="number" id="phi" disabled/><br>
</td>
</tr>

<tr>
<td>
<p>e</p>
</td>
<td>
<input type="number" id="e" disabled/><br>
</td>
</tr>

<tr>
<td>
<p>d</p>
</td>
<td>
<input type="number" id="d" disabled/><br>
</td>
</tr>

</table>
</div>

<hr>
<div class="result">
<table>
<tr>
<td>
<h3>Plain Text</h3>
</td>
<td>
<input type="number" id="plain-text" disabled/><br>
</td>
</tr>

<tr>
<td>
<h3>Cipher Text</h3>
</td>
```

```
<td>
<input type="number" id="cipher-text" disabled/><br>
</td>
</tr>
</table>
</div>
</body>
</html>
```

Style.css

```
h1 {
  font-family: "Geneva";
  text-align: center;
}

h2 {
  font-family: "Geneva";
  text-align: start;
}

h3 {
  font-family: "Geneva";
  text-align: end;
}

p {
  font-family: "Geneva";
  text-align: start;
}

#generate-pq-button {
  background-color: white;
  color: black;
  border: 2px solid #ff4081;
  width: 50%;
}

#generate-pq-button:hover  {
  background-color: #ff4081;
  color: white;
}

.p-and-q-values table {
  width: 100%;
}

.computed-values table {
  width: 100%;
}

.result table {
  width: 100%;
}

.card  {
  margin: auto;
  width: 50%;
  box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2);
  transition: 0.3s;
  border-radius: 5px;
  padding: 32px;
  text-align: center;
}

button {
  background-color: #3f51b5;
  border: none;
  color:  white;
  padding: 15px 32px;
  text-align: center;
  text-decoration: none;
  display: inline-block;
  font-size: 16px;
  margin: 8px 2px;
  cursor: pointer;
  transition-duration: 0.4s;
}

/* for disabled button */
.disabled {
  opacity: 0.6;
  cursor: not-allowed;
}

input[type="number"] {
  margin: 8px;
  padding: 8px;
  width: 80%;
}

input[type="number"]:focus {
  border: 1px solid #555;
}
```

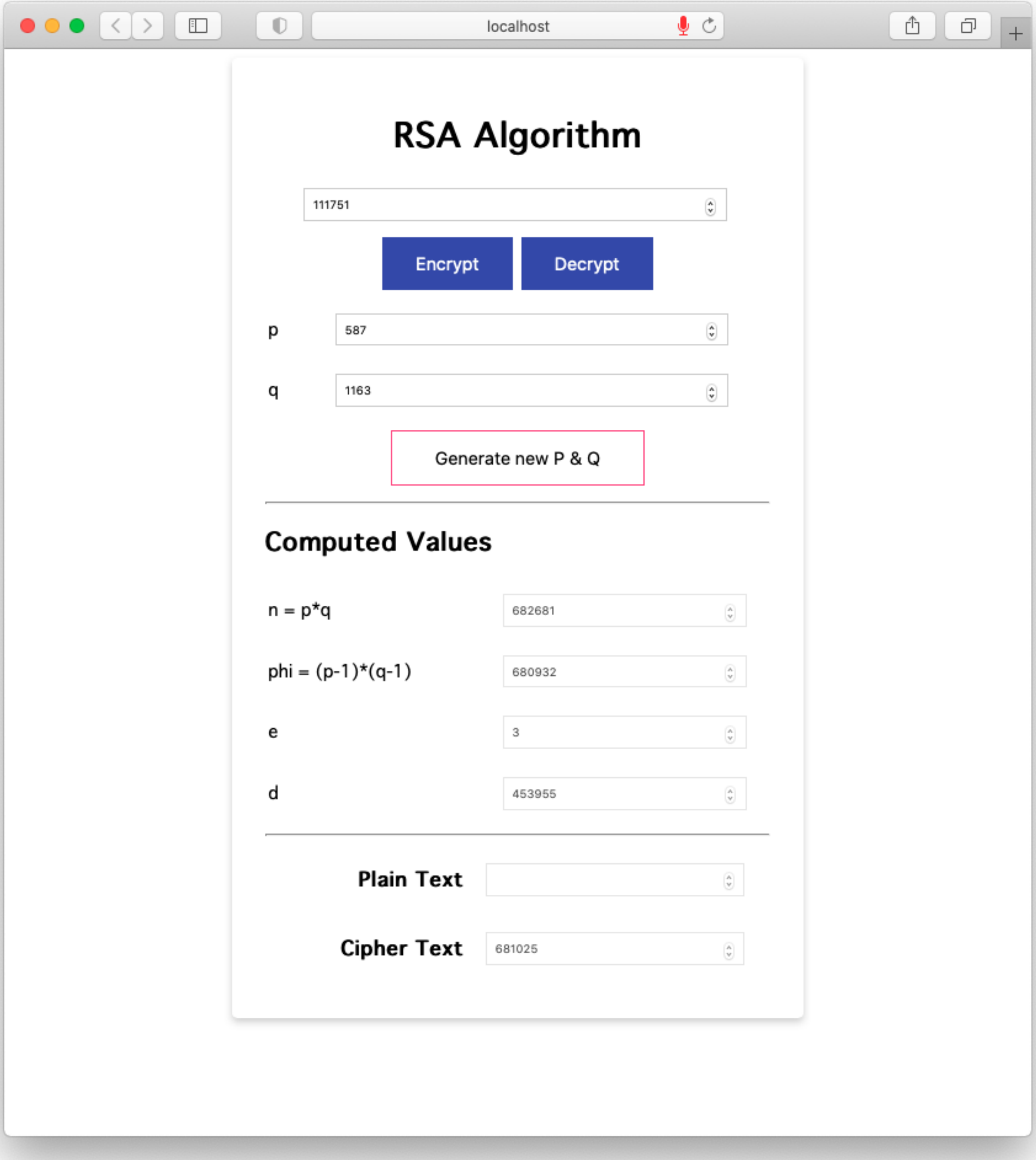
Algorithm.js

```
function isPrime(num) {
    for (let i = 2; i< num / 2; i++) {
        if (num % i === 0) {
            return false;
        }
    }
    return true;
}

function generateNthPrime(n) {
    let count = 0;

    let result = -1;
    for (let i = 2; count <= n; i++) {
        if (isPrime(i)) {
            result = i;
            count++;
        }
    }
    return result;
}

// findExponentialModulo() - fast exponentiation recursive algorithm
function findExponentialModulo(a, N, M) {
    a = a % M;
    let res = 1;
    let temp = a;

while(N > 0){
      var leastSignificantBit = N % 2;
      N = Math.floor(N / 2);

      if (leastSignificantBit == 1) {
        res = res * temp;
        res = res % M;
      }

      temp = temp * temp;
      temp = temp % M;
    }
    return res;
  }

function gcd(num1, num2) {
    let c;
    while (true) {
        c = num1 % num2;
        if (c === 0)
            return num2;
        num1 = num2;
        num2 = c;
    }
}

function encrypt(){
    if(document.getElementById("message").value == ""){
alert("Message field can't be blank");
    } else if (document.getElementById("p").value == "" ||
document.getElementById("q").value == "") {
alert("Either generate P and Q or fill it manually");
     } else {
computeValues();

        let plainText = document.getElementById("message").value;
        let e = document.getElementById("e").value;
        let n = document.getElementById("n").value;

        let cipherText = findExponentialModulo(plainText, e, n);

document.getElementById("plain-text").value = "";
document.getElementById("cipher-text").value = cipherText;
     }
}


function decrypt(){
    if(document.getElementById("message").value == ""){
```

```
alert("Message field can't be blank");
    } else if (document.getElementById("p").value == "" ||
document.getElementById("q").value == "") {
alert("Either generate P and Q or fill it manually");
     } else {
computeValues();

        let cipherText = document.getElementById("message").value;
        let d = document.getElementById("d").value;
        let n = document.getElementById("n").value;


        let plainText = findExponentialModulo(cipherText, d, n);

document.getElementById("plain-text").value = plainText;
document.getElementById("cipher-text").value = "";
     }
}


function computeValues(){

    const p = document.getElementById("p").value;
    const q = document.getElementById("q").value;

    // STEP 2
    let n = p * q;

    // STEP 3
    let phi = (p - 1) * (q - 1);

    // STEP 4
    let e = 2;
    while (e < phi) {
        if (gcd(e, phi) === 1)
            break;
        else
            e++;
    }

    // STEP 5
    let d = 0;
    while ((d * e) % phi != 1) {
        d++;
    }

document.getElementById("n").value = n;
document.getElementById("phi").value = phi;
document.getElementById("e").value = e;
document.getElementById("d").value = d;

}

function generatePandQ() {
    let p = generateNthPrime(Math.floor(Math.random() * 101) + 99);
    let q = generateNthPrime(Math.floor(Math.random() * 101) + 98);
    // make sure p and q are not the same
    while (p === q) {
        q = generateNthPrime(rand.nextInt(100) + 98);
    }

document.getElementById("p").value = p;
document.getElementById("q").value = q;

}
```

Output

Encryption



RSA Algorithm

111751

p 587

q 1163

Computed Values

n = p*q 682681

phi = (p-1)*(q-1) 680932

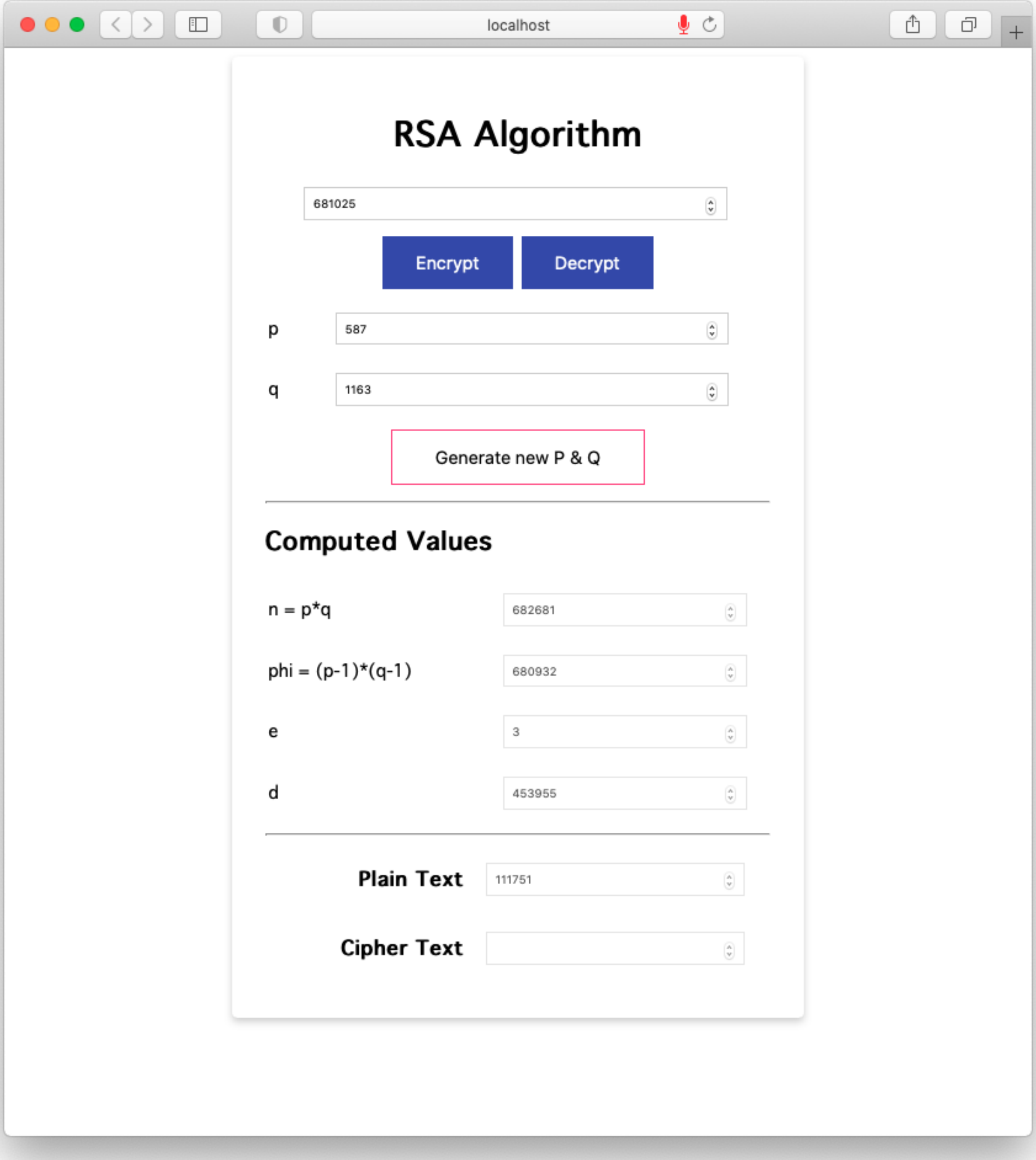
e 3

d 453955

Plain Text

Cipher Text 681025

Decryption



RSA Algorithm

681025

p 587

q 1163

Computed Values

n = p*q 682681

phi = (p-1)*(q-1) 680932

e 3

d 453955

Plain Text 111751

Cipher Text

Conclusion

Thus, RSA algorithm implemented using HTML and JavaScript and output is also verified successfully.

Ex.No.6. Implement Diffie-Hellman Key Exchange Algorithm for a given Problem Statement

Problem statement

Walter wants to send a secret message to Jessie. Hank is trying to intercept the message transfer. So, they decided to use Diffie-Hellman Key Exchange Algorithm to securely transfer secret key. Walter and Jessie publicly agreed to use prime q = 191. Walter chooses private key 73 and Jessie chooses private key 51. Find the public and secret keys in this scenario.

Aim

To solve the given problem using Diffie Hellman Key Exchange Algorithm.

Algorithm (general)

```
1. Read the prime q from the user.
2. Compute the value of alpha (primitive root of q)
3. Read the private key for A and B (Xa and Xb). (assume A is the sender and B is the receiver).
4. Generate public keys for A and B based on the private key using the following formulae
    a. Ya = (alpha^Xa) mod q.
    b. Yb = (alpha^Xb) mod q.
5. Generate secret key for A and B (Ka and Kb) using the following formulae.
    a. Ka = (Yb^Xa) mod q
    b. Kb = (Ya^Xb) mod q
6. Print the secret keys.
```

Algorithm (problem specific)

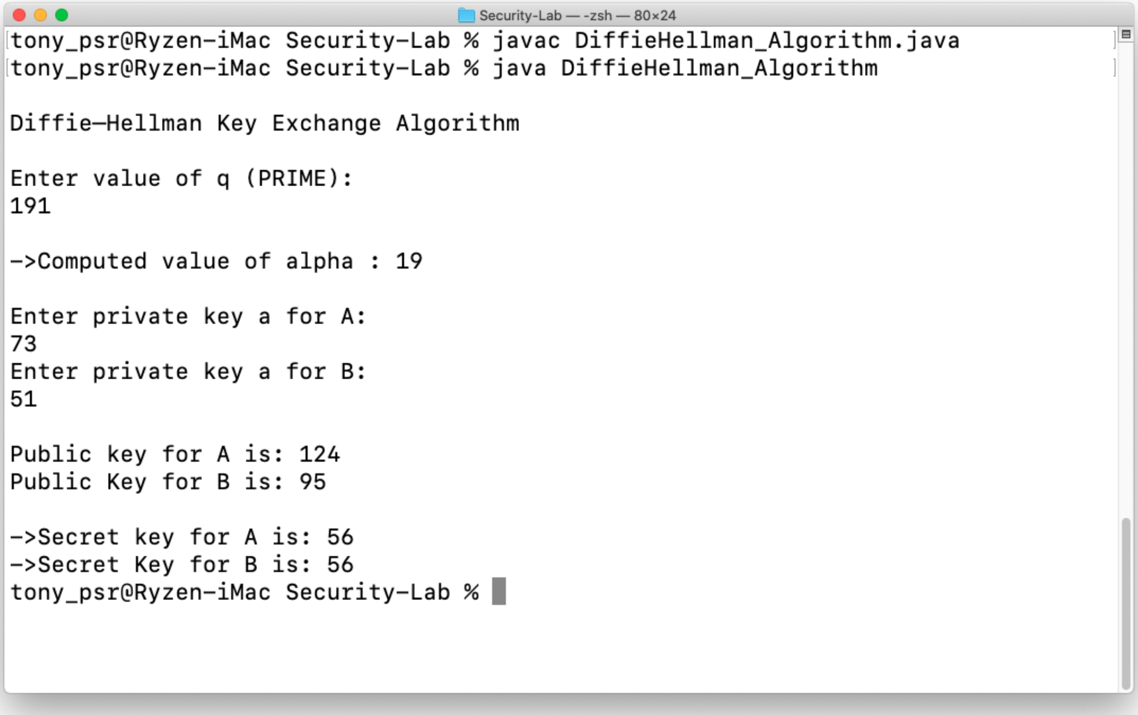
```
1. Set prime q=191.
2. Compute the value of alpha (primitive root of q)
3. Set private key for Walter, Xa = 73 and private key for Jessie, Xb = 51 from the user.
4. Generate public keys for Walter and Jessie based on the private key using the following formulae
    a. Ya = (alpha^Xa) mod q.
    b. Yb = (alpha^Xb) mod q.
5. Generate secret key for Walter and Jessie(Ka and Kb) using the following formulae.
    a. Ka = (Yb^Xa) mod q
    b. Kb = (Ya^Xb) mod q
6. Print the secret key.
```

Source Code

```java
package com.company.tonypsr.cryptographic_algorithms;

import java.util.HashSet;
import java.util.Scanner;

public class DiffieHellman_Algorithm {

    //findExponentialModulo() - fast exponentiation recursive algorithm
    public static long findExponentialModulo(long a, long N, long M) {
        if (N == 0) {
            return 1;
        } else {
            final long R = findExponentialModulo(a, N / 2, M);
            if (N % 2 == 0) {
                return (R * R) % M;
            } else {
                return (R * R * a) % M;
            }
        }
    }

    // custom solution
    public static int findPrimitiveRoot(long num){
        HashSet<Long> set = new HashSet<>();
        int primitiveRoot = -1;

for(int i=1; i<num; i++){
for(int j=1; j<num; j++){
                long val = findExponentialModulo(i, j, num);

                if(set.contains(val)){
                    break;
                } else {
set.add(val);
                }

                if(set.size() == num-1){
primitiveRoot = i;
                    break;
                }
            }
            set = new HashSet<>();
if(primitiveRoot != -1){
                break;
            }
        }

        return primitiveRoot;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        long q, alpha, Ya, Xa, Yb, Xb, Ka, Kb;

        /*
            q = prime
            alpha = primitive of prime q
Ya = public key of A
Xa = private key of A
            Yb = public key of B
Xb = private key of B
            Ka = secret key of A
            kb = secret key of B
         */

System.out.println("\nDiffie–Hellman Key Exchange Algorithm\n");

System.out.println("Enter value of q (PRIME): ");
        q = sc.nextInt();

        alpha = findPrimitiveRoot(q);
System.out.println("Computed value of alpha : " + alpha);

System.out.println("Enter private key a for A: ");
Xa = sc.nextInt();
Ya = findExponentialModulo(alpha, Xa, q);
```

```
System.out.println("Enter private key a for B: ");
Xb = sc.nextInt();
        Yb = findExponentialModulo(alpha, Xb, q);

System.out.println();
System.out.println("Public key for A is: " + Ya);
System.out.println("Public Key for B is: " + Yb);

        Ka = findExponentialModulo(Yb, Xa, q);
Kb = findExponentialModulo(Ya, Xb, q);

System.out.println();
System.out.println("->Secret key for A is: " + Ka);
System.out.println("->Secret Key for B is: " + Kb);
    }
}
```

Output



```
tony_psr@Ryzen-iMac Security-Lab % javac DiffieHellman_Algorithm.java
tony_psr@Ryzen-iMac Security-Lab % java DiffieHellman_Algorithm

Diffie-Hellman Key Exchange Algorithm

Enter value of q (PRIME):
191

->Computed value of alpha : 19

Enter private key a for A:
73
Enter private key a for B:
51

Public key for A is: 124
Public Key for B is: 95

->Secret key for A is: 56
->Secret Key for B is: 56
tony_psr@Ryzen-iMac Security-Lab %
```

Conclusion

Thus, the given problem is solved using Diffie-Hellman Key Exchange Algorithm.

# Ex.No.7.	Calculate the Message Digest of a Text using the SHA-1 Algorithm

Aim

To write a program to calculate the message digest of a text using the SHA-1 Algorithm.

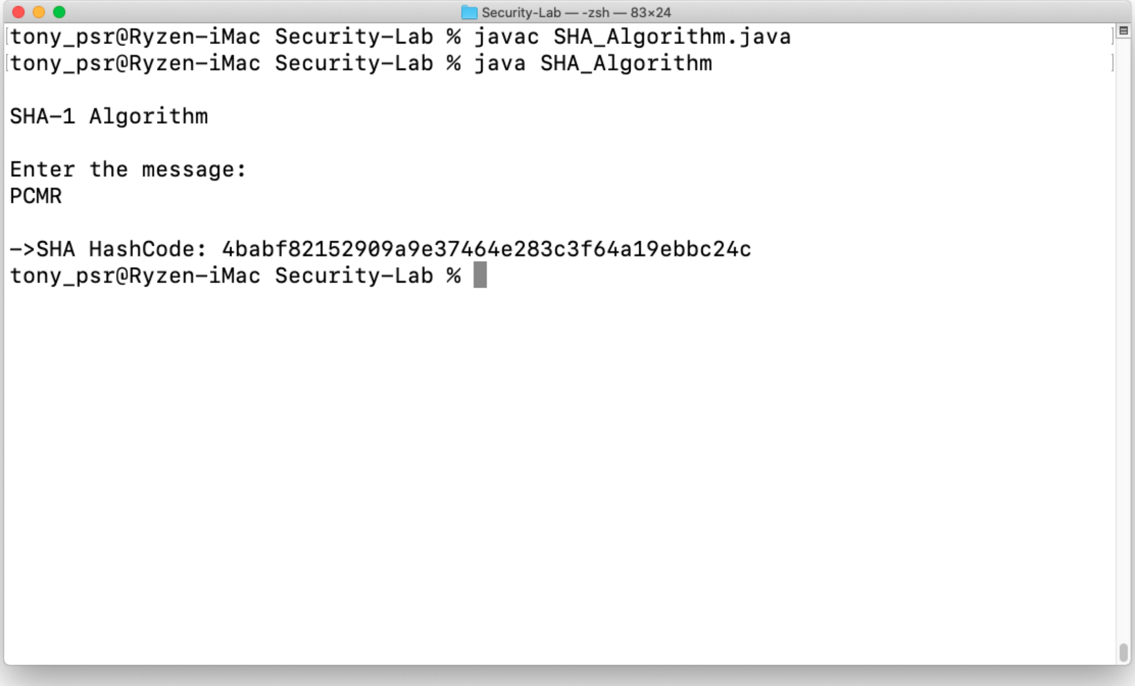
Algorithm

1. Get the message string from the user.
2. Create a MessageDigest instance and messageBytes byte array.
3. Convert the message string to bytes and store it in messageBytes array.
4. Update the messageDigest with the messageBytes array.
5. Using BigIntegerclass, convert the message bytes to string and store it in a StringBuilder object named hashCode.
6. If the length of the hashCode is less than 32, append the remaining places with ‘0’ using append() function.
7. Print the messageDigest as string.

Source Code

```
package com.company.tonypsr.cryptographic_algorithms;

import java.math.BigInteger;
import java.security.MessageDigest;
import java.util.Scanner;

public class SHA_Algorithm {
    public static String generateSHA1HashCode(String input) {
        StringBuilder hashCode = new StringBuilder();;
BigInteger temp;
MessageDigestmessageDigest;
byte[] messageBytes;

        try {
messageDigest = MessageDigest.getInstance("SHA-1");
messageBytes = messageDigest.digest(input.getBytes());

            temp = new BigInteger(1, messageBytes);

messageDigest.update(messageBytes);

hashCode = new StringBuilder(temp.toString(16));

            while (hashCode.length() < 32) {
hashCode.append("0");
            }
        } catch (Exception e) {
e.printStackTrace();
        }

        return hashCode.toString();
    }

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

System.out.println("\nSHA-1 Algorithm\n");
System.out.println("Enter the message: ");
        String message = sc.nextLine();

System.out.println("\n->SHA HashCode: " + generateSHA1HashCode(message));
    }
}
```

Output



```
tony_psr@Ryzen-iMac Security-Lab % javac SHA_Algorithm.java
tony_psr@Ryzen-iMac Security-Lab % java SHA_Algorithm

SHA-1 Algorithm

Enter the message:
PCMR

->SHA HashCode: 4babf82152909a9e37464e283c3f64a19ebbc24c
tony_psr@Ryzen-iMac Security-Lab % 
```

Conclusion

Thus, the program to calculate the message digest for a given text using SHA-1 algorithm has been written, executed and output is also verified.

Ex.No.8. Implement Digital Signature Scheme – Digital Signature

Standard

Aim

To write a program to implement Digital Signature Standard.

Algorithm

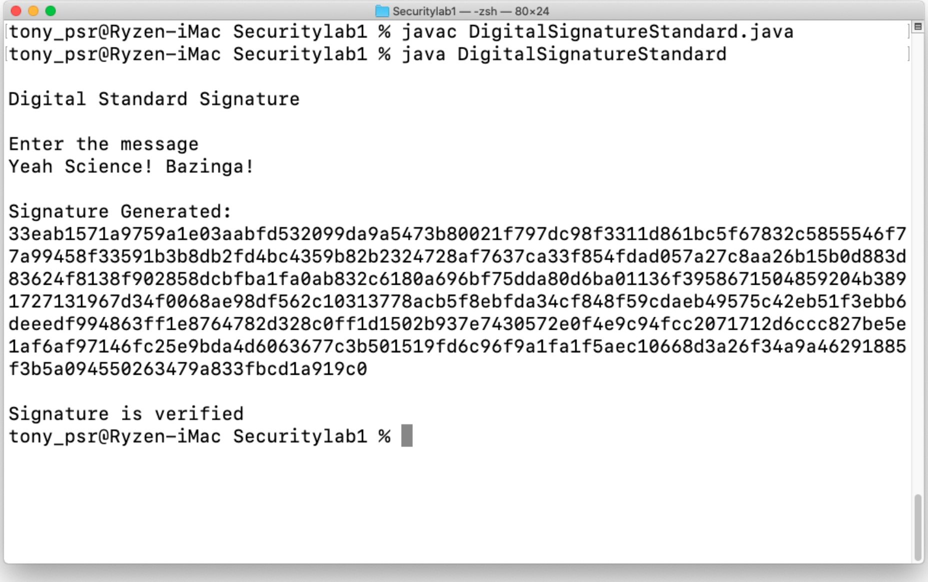
```
1. Import the necessary libraries.
2. Create a method for generating message signature
    a. Initialize a Signature object in “SHA256withRSA” mode.
    b. Sign the object using the key provided and update using the messageBytes.
    c. Return signature for the provided message and key.
3. Create a method to generate RSA key pair that random key pair in RSA instance with 2048 size.
4. Create a method to verify signature.
    a. Initialize a Signature object in “SHA256withRSA” mode.
    b. Using the signature object, public key and message bytes, verify if the sign is valid.
    c. If valid, return true, else return false.
5. Get the message from the user.
6. Generate RSA key pair.
7. Generate message signature.
8. Print verification status.
9. In case of an exception print the stack trace.
```

Source Code

```
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;

public class DigitalSignatureStandard {
    public static byte[] generateSignature(byte[] messageBytes, PrivateKey Key) throws Exception {
        Signature signature = Signature.getInstance("SHA256withRSA");
signature.initSign(Key);
signature.update(messageBytes);
        return signature.sign();
    }

    public static KeyPairgenerateRSAKeyPair() throws Exception {
SecureRandomsecureRandom = new SecureRandom();
KeyPairGeneratorkeyPairGenerator = KeyPairGenerator.getInstance("RSA");
keyPairGenerator.initialize(2048, secureRandom);
        return keyPairGenerator.generateKeyPair();
    }


    public static booleanisVerifiedSignature(byte[] messageBytes, byte[] signatureGenerated,
PublicKeypublicKey) {
        try {
            Signature signature = Signature.getInstance("SHA256withRSA");
signature.initVerify(publicKey);
signature.update(messageBytes);
            return signature.verify(signatureGenerated);
}catch (Exception e){
e.printStackTrace();
```

```
        }
         return false;
    }


    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
System.out.println("\nDigital Standard Signature\n");

System.out.println("Enter the message");
        String message = sc.next();

        try {
KeyPairkeyPair = generateRSAKeyPair();
byte[] signature = generateSignature(message.getBytes(), keyPair.getPrivate());

            // bytes to hex
System.out.println();
System.out.println("Signature Generated:");
            for (byte b : signature) {
                String hex = String.format("%02x", b);
System.out.print(hex);
            }

System.out.println("\n");

            if (isVerifiedSignature(message.getBytes(), signature, keyPair.getPublic())) {
System.out.println("Signature is verified");
            } else {
System.out.println("Signature is not verified");
            }

        } catch (Exception e){
e.printStackTrace();
        }

    }
}
```

Output



```
tony_psr@Ryzen-iMac Securitylab1 % javac DigitalSignatureStandard.java
tony_psr@Ryzen-iMac Securitylab1 % java DigitalSignatureStandard

Digital Standard Signature

Enter the message
Yeah Science! Bazinga!

Signature Generated:
33eab1571a9759a1e03aabfd532099da9a5473b80021f797dc98f3311d861bc5f67832c5855546f7
7a99458f33591b3b8db2fd4bc4359b82b2324728af7637ca33f854fdad057a27c8aa26b15b0d883d
83624f8138f902858dcbfba1fa0ab832c6180a696bf75dda80d6ba01136f3958671504859204b389
1727131967d34f0068ae98df562c10313778acb5f8ebfda34cf848f59cdaeb49575c42eb51f3ebb6
deeedf994863ff1e8764782d328c0ff1d1502b937e7430572e0f4e9c94fcc2071712d6ccc827be5e
1af6af97146fc25e9bda4d6063677c3b501519fd6c96f9a1fa1f5aec10668d3a26f34a9a46291885
f3b5a094550263479a833fbcd1a919c0

Signature is verified
tony_psr@Ryzen-iMac Securitylab1 %
```

Conclusion

Thus, the program to generate and verify DSS signatures has been written, executed and output is also verified.

Ex.No.9 Intrusion Detection System using Snort

Aim:

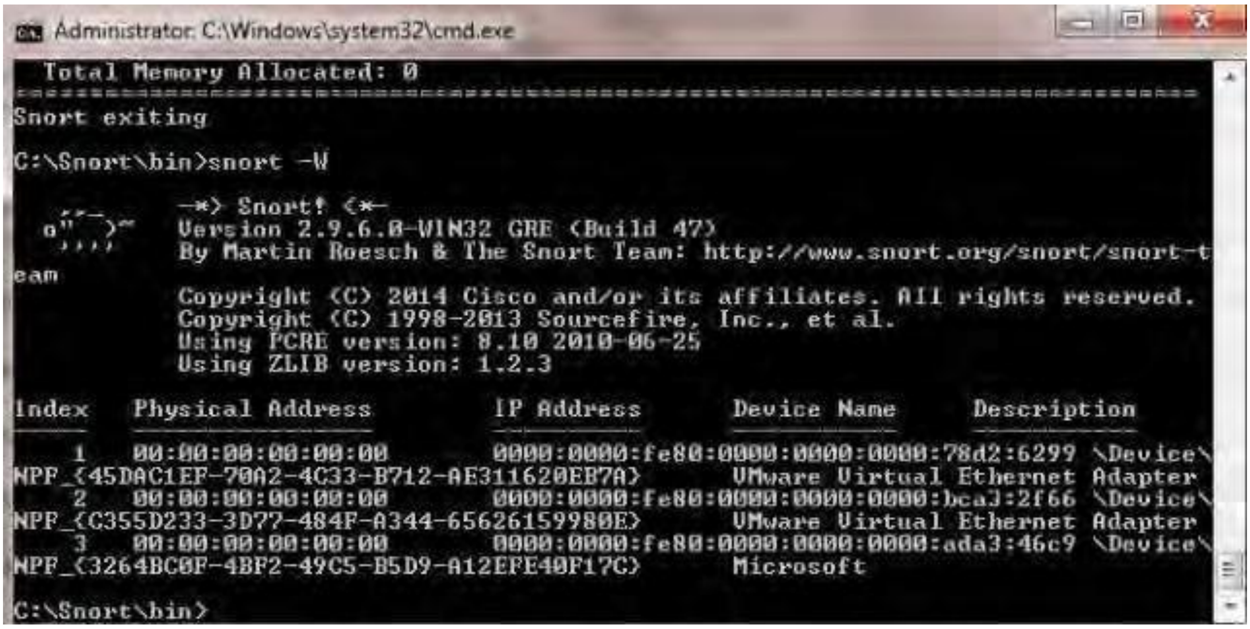
To demonstrate intrusion detection system (ids) using any snort.

Procedure:

STEPSONCONFIGURINGANDINTRUSIONDETECTION:

1. DownloadSnort fromtheSnort.orgwebsite.(http://www.snort.org/snort-downloads)
2. DownloadRules(https://www.snort.org/snort-rules).Youmustregistertogettherules.(Youshould downloadtheseoften)
3. Double click on the .exe to install snort.This will install snort in the "C:\Snort"folder.ItisimportanttohaveWinPcap(https://www.winpcap.org/install/)installed
4. ExtracttheRulesfile.YouwillneedWinRARforthe.gz file.
5. Copyallfilesfromthe"rules"folderoftheextractedfolder.Nowpastetherulesinto *"C:\Snort\rules"*folder.
6. Copy"snort.conf"filefromthe"etc"folderoftheextractedfolder.Youmustpasteitinto"C:\Snort\etc"folder.Overwriteany existing file.Remember ifyou modify your snort.conf file and download a new file, you must modify it forSnort towork.
7. Openacommandprompt(cmd.exe)andnavigateto folder"C:\Snort\bin"folder.(at thePrompt,typecd\snort\bin)
8. Tostart(execute)snortinsniffer modeusefollowingcommand:snort-dev-i3
-iindicatestheinterfacenumber.Youmustpickthecorrectinterfacenumber.Inmycase,itis3.
-devisusedtorunsnorttocapturepacketsonyournetwork.

Tochecktheinterfacelist,usefollowingcommand:snort
-W



Administrator: C:\Windows\system32\cmd.exe

```
  Total Memory Allocated: 0
================================================================================
Snort exiting

C:\Snort\bin>snort -W

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.6.0-WIN32 GRE (Build 47)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/snort/snort-t
eam
           Copyright (C) 2014 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using PCRE version: 8.10 2010-06-25
           Using ZLIB version: 1.2.3

Index   Physical Address        IP Address      Device Name     Description
-----   ----------------        ----------      -----------     -----------
    1   00:00:00:00:00:00       0000:0000:fe80:0000:0000:0000:78d2:6299 \Device\
NPF_{45DAC1EF-70A2-4C33-B712-AE311620EB7A}      VMware Virtual Ethernet Adapter
    2   00:00:00:00:00:00       0000:0000:fe80:0000:0000:0000:bca3:2f66 \Device\
NPF_{C355D233-3D77-484F-A344-65626159980E}      VMware Virtual Ethernet Adapter
    3   00:00:00:00:00:00       0000:0000:fe80:0000:0000:0000:ada3:46c9 \Device\
NPF_{3264BC0F-4BF2-49C5-B5D9-A12EFE40F17C}      Microsoft

C:\Snort\bin>
```

Findinganinterface

YoucantellwhichinterfacetousebylookingattheIndexnumberandfindingMicrosoft.As you can see in the above example, the other interfaces are forVMWare.Myinterfaceis3.

9. TorunsnortinIDS mode,youwillneed toconfigurethefile“snort.conf”accordingtoyournetwork environment.
10. Tospecifythenetworkaddressthatyouwanttoprotect insnort.conffile,lookforthefollowingline.
varHOME_NET192.168.1.0/24(Youwillnormally seeanyhere)

11. YoumayalsowanttosettheaddressesofDNS_SERVERS,ifyouhavesomeon yournetwork.

Example:

examplesnort
12. ChangetheRULE_PATHvariabletothepathofrulesfolder.varRUL E_PATHc:\snort\rules

pathto rules

13. Changethepathofalllibraryfileswiththenameandpathonyoursystem. andyoumustchangethepath of snort_dynamicpreprocessorvariable.C:\Snort\lib\snort_dynamiccpreprocessor You need to do this to all library files in the “C:\Snort\lib” folder. The old pathmightbe:“/usr/local/lib/…”. youwillneedto replacethatpathwithyoursystempath.UsingC:\Snort\lib
14. Changethepathofthe“dynamicengine”variablevalue inthe“snort.conf”file..
Example:

dynamicengineC:\Snort\lib\snort_dynamicengine\sf_engine.dll

15Addthepathsfor“includeclassification.config”and“includereference.config”files.
includec:\snort\etc\classification.configincludec
:\snort\etc\reference.config

16. Removethecomment(#)onthelinetoallowICMPrules, ifitiscommentedwitha#.
include$RULE_PATH/icmp.rules

17. YoucanalsoremovethecommentofICMP-inforulescomment,ifitiscommented.
include$RULE_PATH/icmp-info.rules

18. Toaddlogfilestostorealertsgeneratedbysnort,searchforthe“outputlog”test insnort.confandaddthefollowingline:
outputalert_fast:snort-alerts.ids

19. Comment(adda#)thewhitelist$WHITE_LIST_PATH/white_list.rulesandtheblacklist

Changethenested_ipinner,\tonested_ipinner#,\

20. Comment out (#) following lines:#preprocessornormalize_ip4
#preprocessornormalize_tcp:ipsecnstream#preprocessornormalize_icmp4#preprocessornormalize_ip6
#preprocessornormalize_icmp6

21. Savethe“snort.conf”file.

22. TostartsnortinIDSmode,runthefollowingcommand:

snort-c
c:\snort\etc\snort.conf-lc:\snort\log-i3(Note:3isusedformyinterfacecard)

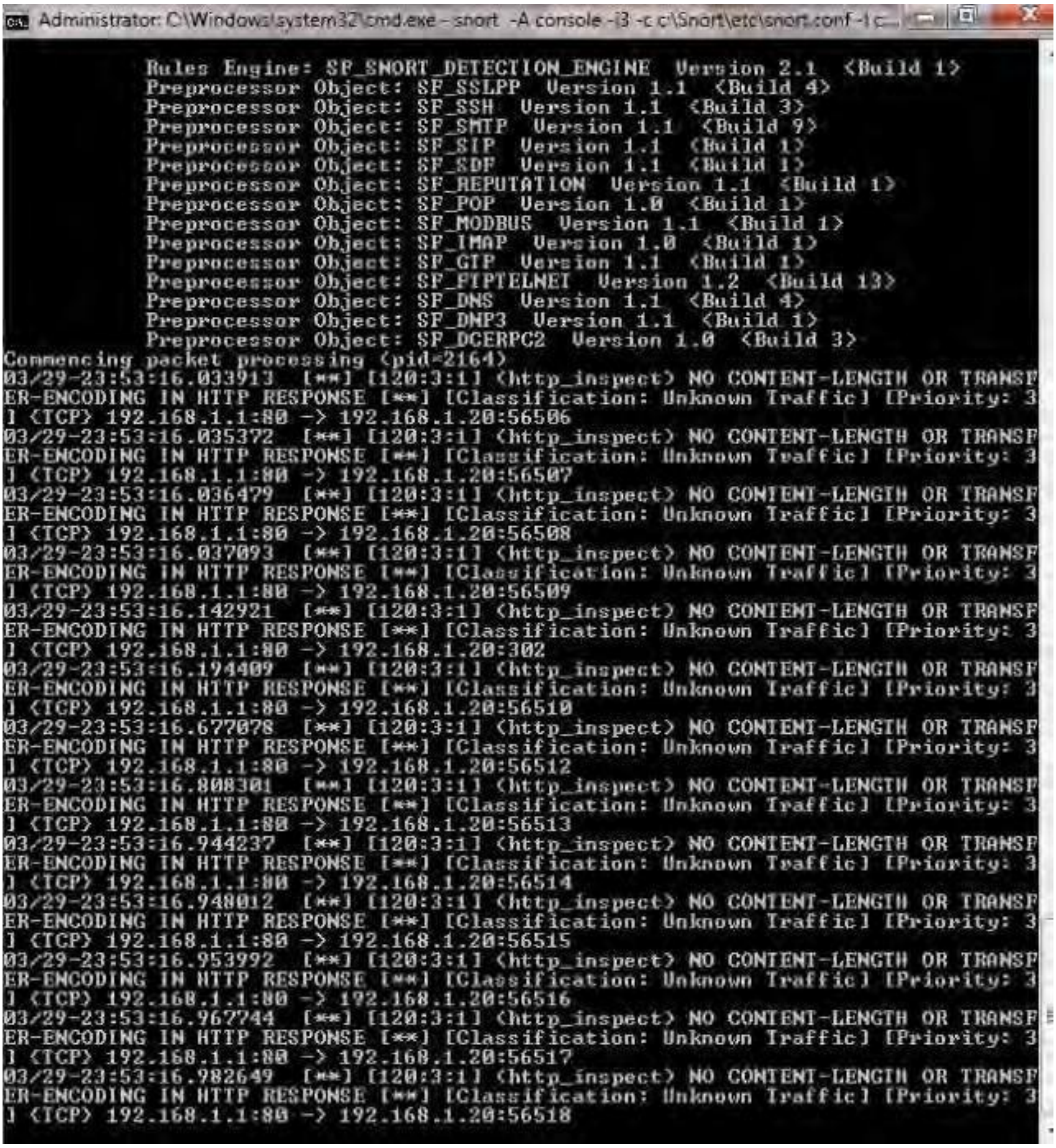
Ifalogiscreated,selecttheappropriateprogramtoopenit.YoucanuseWordPard orNotePad++toread thefile.

TogenerateLogfilesinASCIImode,youcanusefollowingcommandwhilerunningsnortinIDSmode:
snort-Aconsole-i3-cc:\Snort\etc\snort.conf-lc:\Snort\log-Kascii

23. ScanthecomputerthatisrunningsnortfromanothercomputerbyusingPINGorNMap(ZenMap).

After scanning or during the scan you can check the snort-alerts.ids file in the logfoldertoinsureitisloggingproperly.YouwillseeIPaddressfoldersappear.

Snortmonitoringtraffic –



Administrator: C:\Windows\system32\cmd.exe - snort -A console -i3 -c c:\Snort\etc\snort.conf -l c...

```
           Rules Engine: SF_SNORT_DETECTION_ENGINE  Version 2.1  <Build 1>
           Preprocessor Object: SF_SSLPP  Version 1.1  <Build 4>
           Preprocessor Object: SF_SSH  Version 1.1  <Build 3>
           Preprocessor Object: SF_SMTP  Version 1.1  <Build 9>
           Preprocessor Object: SF_SIP  Version 1.1  <Build 1>
           Preprocessor Object: SF_SDF  Version 1.1  <Build 1>
           Preprocessor Object: SF_REPUTATION  Version 1.1  <Build 1>
           Preprocessor Object: SF_POP  Version 1.0  <Build 1>
           Preprocessor Object: SF_MODBUS  Version 1.1  <Build 1>
           Preprocessor Object: SF_IMAP  Version 1.0  <Build 1>
           Preprocessor Object: SF_GTP  Version 1.1  <Build 1>
           Preprocessor Object: SF_FTPTELNET  Version 1.2  <Build 13>
           Preprocessor Object: SF_DNS  Version 1.1  <Build 4>
           Preprocessor Object: SF_DNP3  Version 1.1  <Build 1>
           Preprocessor Object: SF_DCERPC2  Version 1.0  <Build 3>
Commencing packet processing (pid=2164)
03/29-23:53:16.033913  [**] [120:3:1] (http_inspect) NO CONTENT-LENGTH OR TRANSF
ER-ENCODING IN HTTP RESPONSE [**] [Classification: Unknown Traffic] [Priority: 3
] {TCP} 192.168.1.1:80 -> 192.168.1.20:56506
03/29-23:53:16.035372  [**] [120:3:1] (http_inspect) NO CONTENT-LENGTH OR TRANSF
ER-ENCODING IN HTTP RESPONSE [**] [Classification: Unknown Traffic] [Priority: 3
] {TCP} 192.168.1.1:80 -> 192.168.1.20:56507
03/29-23:53:16.036479  [**] [120:3:1] (http_inspect) NO CONTENT-LENGTH OR TRANSF
ER-ENCODING IN HTTP RESPONSE [**] [Classification: Unknown Traffic] [Priority: 3
] {TCP} 192.168.1.1:80 -> 192.168.1.20:56508
03/29-23:53:16.037093  [**] [120:3:1] (http_inspect) NO CONTENT-LENGTH OR TRANSF
ER-ENCODING IN HTTP RESPONSE [**] [Classification: Unknown Traffic] [Priority: 3
] {TCP} 192.168.1.1:80 -> 192.168.1.20:56509
03/29-23:53:16.142921  [**] [120:3:1] (http_inspect) NO CONTENT-LENGTH OR TRANSF
ER-ENCODING IN HTTP RESPONSE [**] [Classification: Unknown Traffic] [Priority: 3
] {TCP} 192.168.1.1:80 -> 192.168.1.20:302
03/29-23:53:16.194409  [**] [120:3:1] (http_inspect) NO CONTENT-LENGTH OR TRANSF
ER-ENCODING IN HTTP RESPONSE [**] [Classification: Unknown Traffic] [Priority: 3
] {TCP} 192.168.1.1:80 -> 192.168.1.20:56510
03/29-23:53:16.677078  [**] [120:3:1] (http_inspect) NO CONTENT-LENGTH OR TRANSF
ER-ENCODING IN HTTP RESPONSE [**] [Classification: Unknown Traffic] [Priority: 3
] {TCP} 192.168.1.1:80 -> 192.168.1.20:56512
03/29-23:53:16.808301  [**] [120:3:1] (http_inspect) NO CONTENT-LENGTH OR TRANSF
ER-ENCODING IN HTTP RESPONSE [**] [Classification: Unknown Traffic] [Priority: 3
] {TCP} 192.168.1.1:80 -> 192.168.1.20:56513
03/29-23:53:16.944237  [**] [120:3:1] (http_inspect) NO CONTENT-LENGTH OR TRANSF
ER-ENCODING IN HTTP RESPONSE [**] [Classification: Unknown Traffic] [Priority: 3
] {TCP} 192.168.1.1:80 -> 192.168.1.20:56514
03/29-23:53:16.948012  [**] [120:3:1] (http_inspect) NO CONTENT-LENGTH OR TRANSF
ER-ENCODING IN HTTP RESPONSE [**] [Classification: Unknown Traffic] [Priority: 3
] {TCP} 192.168.1.1:80 -> 192.168.1.20:56515
03/29-23:53:16.953992  [**] [120:3:1] (http_inspect) NO CONTENT-LENGTH OR TRANSF
ER-ENCODING IN HTTP RESPONSE [**] [Classification: Unknown Traffic] [Priority: 3
] {TCP} 192.168.1.1:80 -> 192.168.1.20:56516
03/29-23:53:16.967744  [**] [120:3:1] (http_inspect) NO CONTENT-LENGTH OR TRANSF
ER-ENCODING IN HTTP RESPONSE [**] [Classification: Unknown Traffic] [Priority: 3
] {TCP} 192.168.1.1:80 -> 192.168.1.20:56517
03/29-23:53:16.982649  [**] [120:3:1] (http_inspect) NO CONTENT-LENGTH OR TRANSF
ER-ENCODING IN HTTP RESPONSE [**] [Classification: Unknown Traffic] [Priority: 3
] {TCP} 192.168.1.1:80 -> 192.168.1.20:56518
```

RESULT:

Thus the Intrusion Detection System (IDS) has been demonstrated by Using the Open Source Snort Intrusion Detection Tool.

Ex.No.10 ExploringN-Stalker,aVulnerabilityAssessmentTool

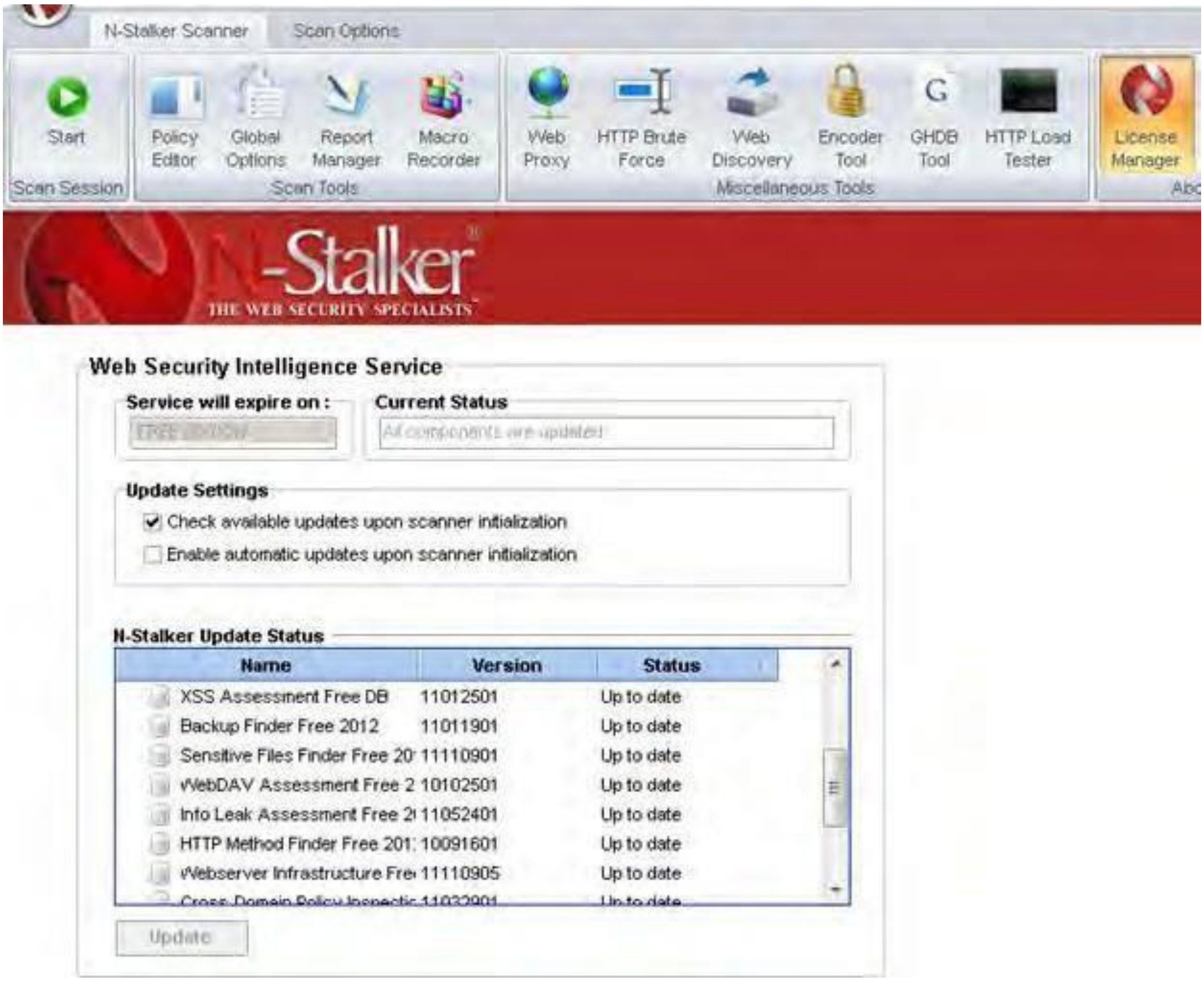
AIM:

TodownloadtheN-StalkerVulnerabilityAssessment Toolandexploringthe features.

EXPLORINGN-STALKER:

- N-StalkerWebApplicationSecurityScannerisaWebsecurityassessmenttool.
- Itincorporates withawell-knownN-StealthHTTPSecurityScannerand35,000Web attacksignaturedatabase.
- Thistoolalsocomesinbothfreeandpaid version.
- Beforescanningthetarget,goto“LicenseManager”tab, performtheupdate.
- Onceupdate, youwillnotethestatusasuptodate.
- YouneedtodownloadandinstallN-Stalkerfromwww.nstalker.com.

1. StartN-StalkerfromaWindowscomputer.TheprogramisinstalledunderStart ➪ Programs ➪ N-Stalker ➪ N-StalkerFreeEdition.
2. Enterahostaddressorarangeofaddressestoscan.
3. ClickStartScan.
4. Afterthescancompletes, theN-StalkerReportManagerwillprompt
5. youtoselectaformatfortheresultingreportaschooseGenerateHTML.
6. ReviewtheHTMLreportforvulnerabilities.



Web Security Intelligence Service

Service will expire on : FREE EDITION

Current Status: All components are updated

Update Settings

☑ Check available updates upon scanner initialization

□ Enable automatic updates upon scanner initialization

N-Stalker Update Status

Name	Version	Status
XSS Assessment Free DB	11012501	Up to date
Backup Finder Free 2012	11011901	Up to date
Sensitive Files Finder Free 20	11110901	Up to date
WebDAV Assessment Free 2	10102501	Up to date
Info Leak Assessment Free 2	11052401	Up to date
HTTP Method Finder Free 201	10091601	Up to date
Webserver Infrastructure Fre	11110905	Up to date
Cross Domain Policy Inspectic	11032901	Up to date

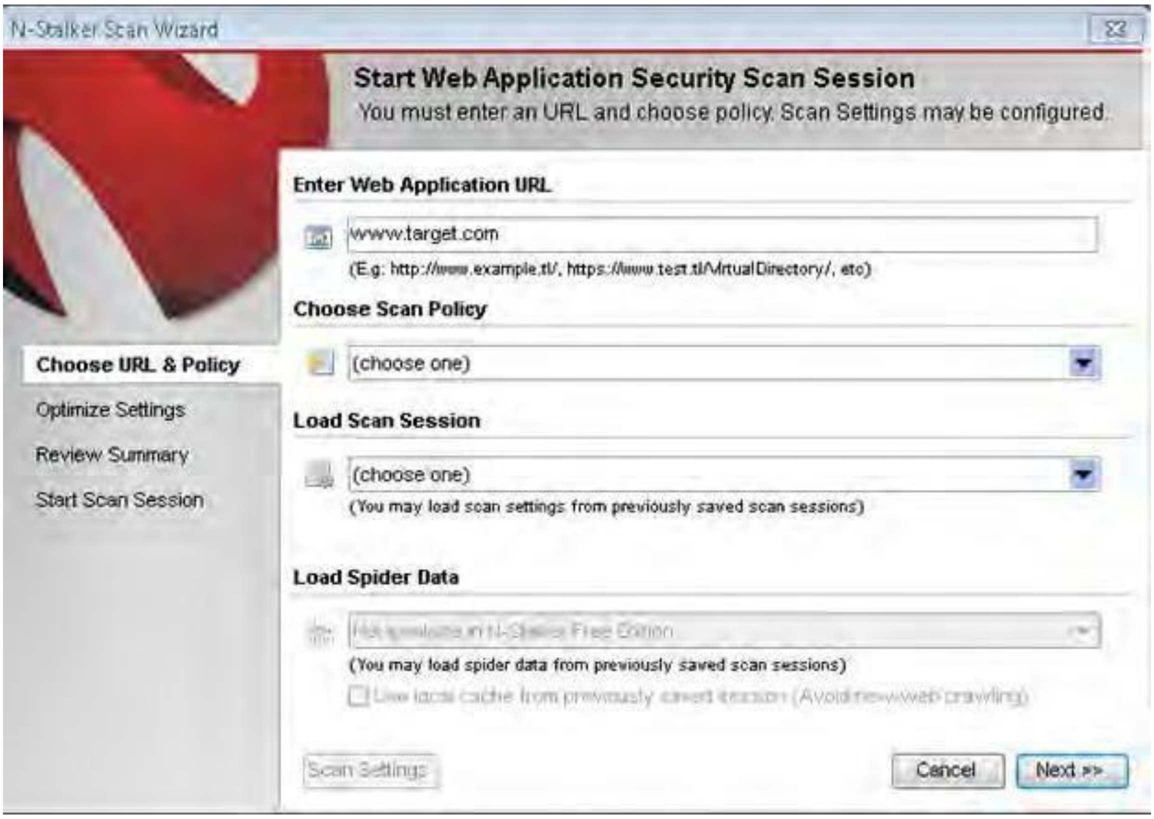
Nowgoto“ScanSession”,enterthetargetURL.

Inscanpolicy, youcanselectfromthefouroptions,

- Manualtestwhichwillcrawlthewebsiteandwillbewaitingformanualattacks.
- fullxssassessment
- owasppolicy
- Webserverinfrastructureanalysis.

Once,theoptionhasbeenselected,nextstepis“Optimizesettings”whichwillcrawlthewholewebsiteforfurtheranalysis.

Inreviewoption,youcangetalltheinformationlikehostinformation,technologiesused,policyname,etc.



Start Web Application Security Scan Session

You must enter an URL and choose policy. Scan Settings may be configured.

Enter Web Application URL

www.target.com

(E.g: http://www.example.tl/, https://www.test.tl/VirtualDirectory/, etc)

Choose Scan Policy

(choose one)

Load Scan Session

(choose one)

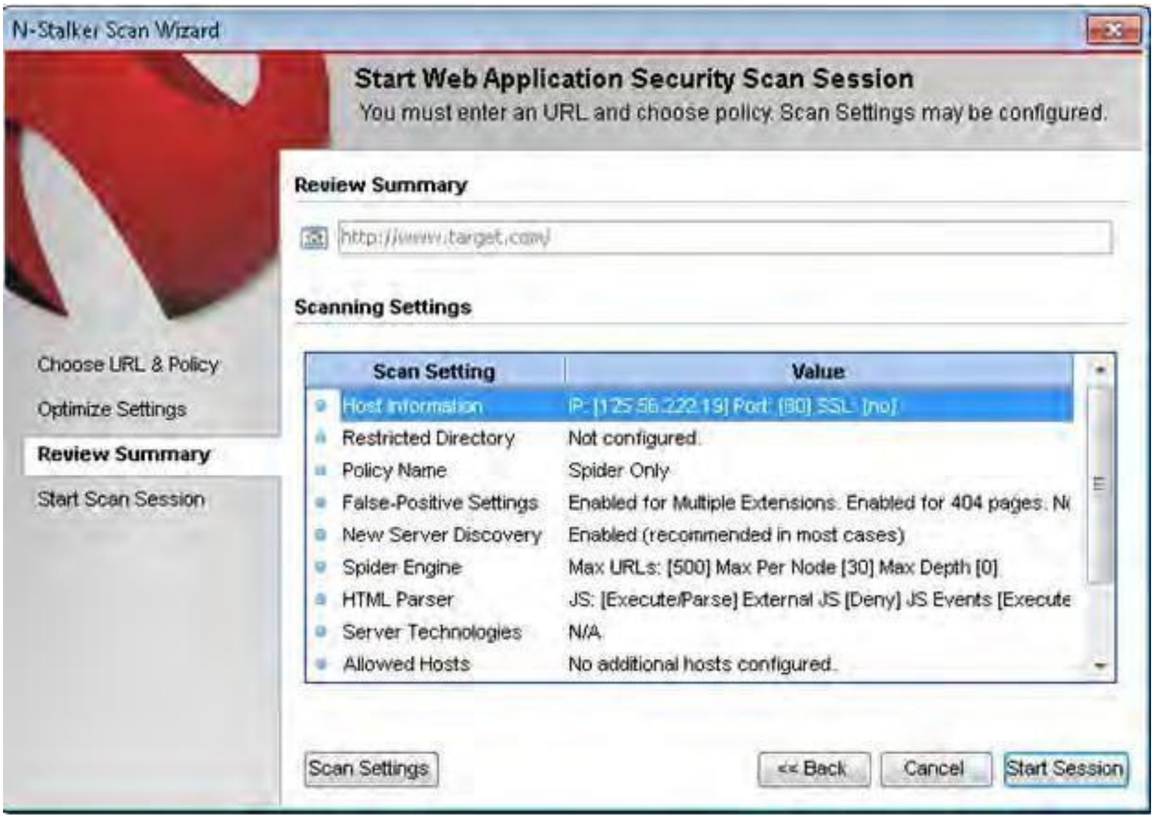
(You may load scan settings from previously saved scan sessions)

Load Spider Data

Not available in N-Stalker Free Edition

(You may load spider data from previously saved scan sessions)

□ Use local cache from previously saved session (Avoid new web crawling)



Start Web Application Security Scan Session

You must enter an URL and choose policy. Scan Settings may be configured.

Review Summary

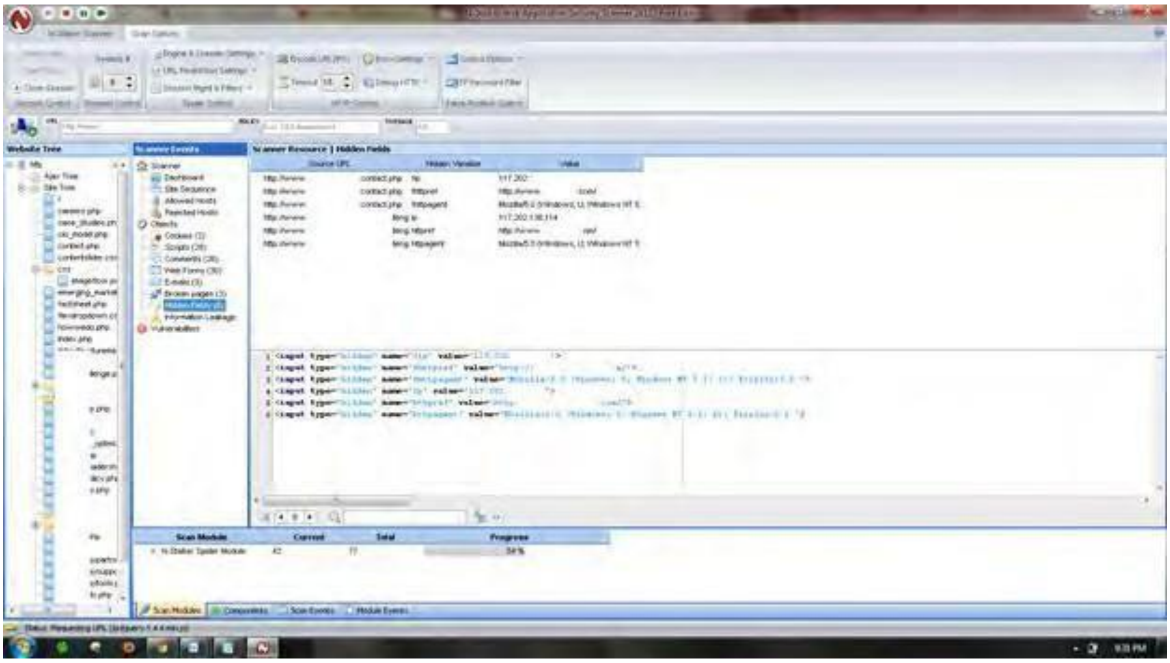
http://www.target.com/

Scanning Settings

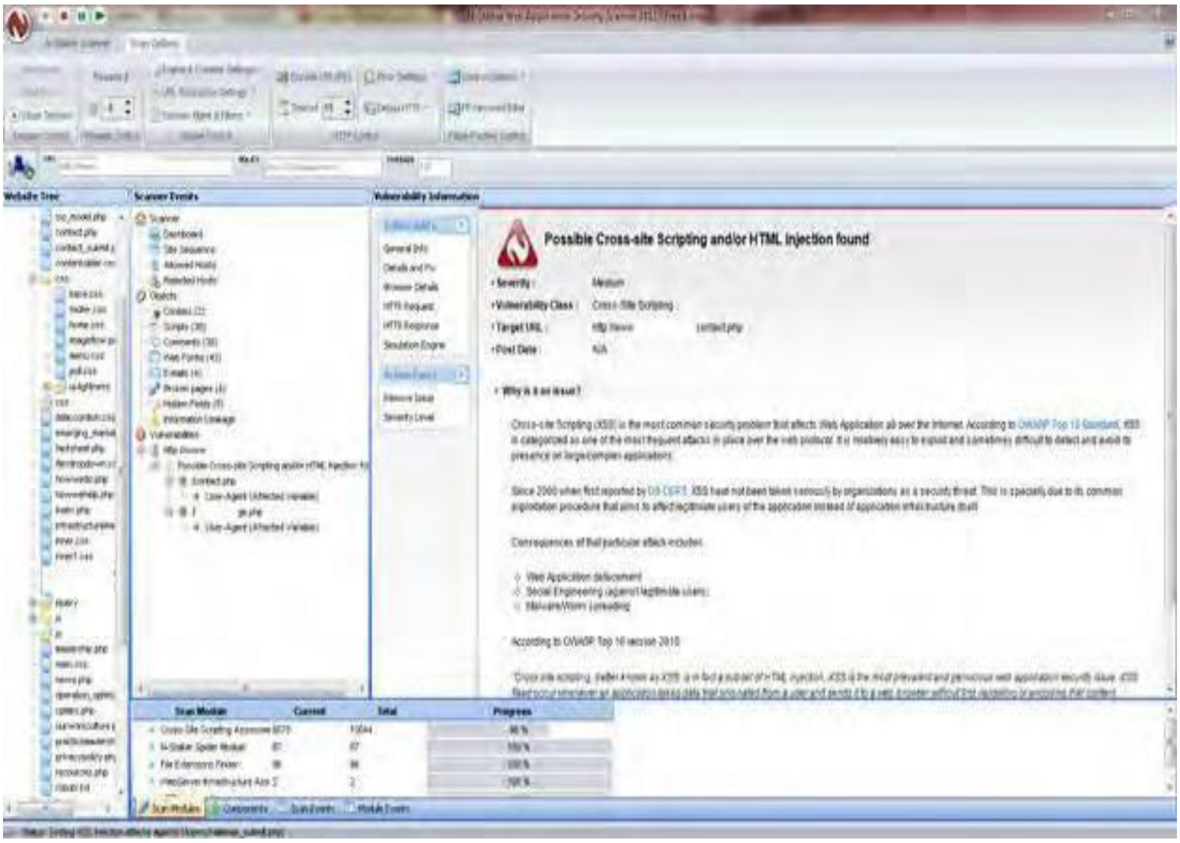
Scan Setting	Value
Host Information	IP: [125.56.222.19] Port: [80] SSL: [no]
Restricted Directory	Not configured.
Policy Name	Spider Only
False-Positive Settings	Enabled for Multiple Extensions. Enabled for 404 pages. N
New Server Discovery	Enabled (recommended in most cases)
Spider Engine	Max URLs: [500] Max Per Node [30] Max Depth [0]
HTML Parser	JS: [Execute/Parse] External JS [Deny] JS Events [Execute
Server Technologies	N/A.
Allowed Hosts	No additional hosts configured.

Oncedone,start the session and start the scan.

The scanner will crawl the whole website and will show the scripts,broken pages, hidden fields,information leakage, web forms related information which helps to analyze further.



Once the scan is completed,the NStalker scanner will show details like severity level, vulnerabilityclass, why is it an issue, the fix for the issue and the URLwhich is vulnerable to the particular vulnerability?



Possible Cross-site Scripting and/or HTML injection found

• Severity :	Medium
• Vulnerability Class :	Cross-Site Scripting
• Target URL :	http://www [illegible] contact.php
• Post Data :	N/A

• Why is it an issue?

Cross-site Scripting (XSS) is the most common security problem that affects Web Application all over the Internet. According to OWASP Top 10 Standard, XSS is categorized as one of the most frequent attacks in place over the web protocol. It is relatively easy to exploit and sometimes difficult to detect and avoid its presence on large/complex applications.

Since 2000 when first reported by US-CERT, XSS have not been taken seriously by organizations as a security threat. This is specially due to its common exploitation procedure that aims to affect legitimate users of the application instead of application infrastructure itself.

Consequences of that particular attack includes:

- Web Application defacement
- Social Engineering (against legitimate users)
- Malware/Worm Uploading

According to OWASP Top 10 version 2010:

"Cross site scripting, better known as XSS, is in fact a subset of HTML injection. XSS is the most prevalent and pernicious web application security issue. XSS flaws occur whenever an application takes data that originated from a user and sends it to a web browser without first validating or encoding that content.

Scan Module	Current	Total	Progress
Cross Site Scripting Assessment	8075	10044	80 %
N-Stalker Spider Module	87	87	100 %
File Extensions Finder	98	98	100 %
WebServer Infrastructure Ass...	2	2	100 %

RESULT:

Thus the N-Stalker Vulnerability Assessment tool has been downloaded, installed and the features has been explored by using a vulnerable website.

Ex.No.11.a SECURE DATA STORAGE, SECURE DATA TRANSMISSION AND FOR CREATING DIGITAL SIGNATURE(GNUPG)

AIM:

Demonstrate how to provide secure data storage, secure data transmission and for creating digital signatures (GnuPG).

INTRODUCTION:

- GnuPG is a complete and free implementation of the OpenPGP standard as defined by RFC4880 (also known as *PGP*).
- GnuPG allows you to encrypt and sign your data and communications; it features a versatile key management system, along with access modules for all kinds of public key directories.
- GnuPG, also known as *GPG,*is a command line tool with features for easy integration with other applications.
- A wealth of frontend applications and libraries are available. GnuPG also provides support for S/MIME and Secure Shell (ssh).
- Since its introduction in 1997, GnuPG is Free Software (meaning that it respects your freedom). It can be freely used, modified and distributed under the terms of the GNU General Public License .
- The current version of GnuPG is 2.2.17. See the download page for other maintained versions.

DESCRIPTION:

- The **rng-tools**is a set of utilities related to random number generation in kernel. The main program is rngd, a daemon developed to check and feed random data from hardware device to kernel entropy pool.
- The $**urandom**() system **function**provides a mechanism for generating pseudo-random numbers. The **function**returns a new unsigned 32-bit random number each time it is called.
- --full-generate-key: A **key**is used to encrypt and decrypt whatever data is being encrypted/decrypted. A device or program used to generate **keys**is called a **keygenerator**or keygen.

OUTPUT:

student@IT20:~$ sudo apt-get install gnupg Reading package lists... Done Building dependency tree

```
Reading state information... Done
gnupg is already the newest version (2.2.4-1ubuntu1.1).
The following packages were automatically installed and are no longer required: ca-
certificates-java fonts-dejavu-extra libatk-wrapper-java libatk-wrapper-java-jni
libgif7
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.

student@IT20:~$ sudo apt-get install rng-tools
Reading package lists... Done
Building dependency tree
Reading state information... Done
rng-tools is already the newest version (5-0ubuntu4).
The following packages were automatically installed and are no longer required: ca-
certificates-java fonts-dejavu-extra libatk-wrapper-java libatk-wrapper-java-jni
libgif7
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 0 not

upgraded. student@IT20:~$ sudorngd -r/dev/urandom
student@IT20:~$ gpg --gen-key
gpg (GnuPG) 2.2.4; Copyright (C) 2017 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Note: Use "gpg --full-generate-key" for a full featured key generation dialog.

GnuPG needs to construct a user ID to identify your key.

Real name: BhelAbinayaArthy
Email address: arthy.bhel@gmail.com
You selected this USER-ID:
"BhelAbinayaArthy<arthy.bhel@gmail.com>"

Change (N)ame, (E)mail, or (O)kay/(Q)uit? O
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
gpg: key B8F5B746814794F3 marked as ultimately trusted
gpg: revocation certificate stored as
'/home/student/.gnupg/openpgp-revocs.d/D370EB3A6135FD0D66D910E9B8F5B74681
4794F3.rev'
```

public and secret key created and signed.

```
pubrsa3072    2019-08-14    [SC]    [expires:    2021-08-13]
D370EB3A6135FD0D66D910E9B8F5B746814794F3
uidBhelAbinayaArthy<arthy.bhel@gmail.com>   sub rsa3072
2019-08-14 [E] [expires: 2021-08-13]
```

student@IT20:~$ gpg --full-generate-key

```
gpg (GnuPG) 2.2.4; Copyright (C) 2017 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Please select what kind of key you want:
       (1)      RSA and RSA (default)
       (2)      DSA and Elgamal
       (3)      DSA (sign only)
       (4)      RSA (sign only)
Your selection? 1
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (3072) 1024
Requested keysize is 1024 bits
Please specify how long the key should be valid.
0 = key does not expire
<n> = key expires in n days
<n>w = key expires in n weeks
<n>m = key expires in n months
<n>y = key expires in n years
Key is valid for? (0) 0
Key does not expire at all
Is this correct? (y/N) y

GnuPG needs to construct a user ID to identify your key.
```

Real name: BhelAbinayaArthy
Email address: arthy.bhel@gmail.com
Comment: This is a comment message
You selected this USER-ID:
"BhelAbinayaArthy (This is a comment message) <arthy.bhel@gmail.com>"

```
Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? O
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
We need to generate a lot of random bytes. It is a good idea to perform
```

```
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
gpg: key CAA5BC89F4B21362 marked as ultimately trusted
gpg: revocation certificate stored as
'/home/student/.gnupg/openpgp-revocs.d/891FC646D9447492CAE23D53CAA5BC89F4
B21362.rev'
public and secret key created and signed.

pubrsa1024              2019-08-14              [SC]
891FC646D9447492CAE23D53CAA5BC89F4B21362
uid             BhelAbinayaArthy (This is a comment message)
<arthy.bhel@gmail.com>
sub    rsa1024 2019-08-14 [E]
```

student@IT20:~$ vi arthy.txt

```
student@IT20:~$ gpg -e -r BhelAbinayaArthy arthy.txt
gpg: checking the trustdb
gpg: marginals needed: 3 completes needed: 1 trust model: pgp
gpg: depth: 0 valid:    4 signed:    0 trust: 0-, 0q, 0n, 0m, 0f, 4u
gpg: next trustdb check due at 2019-08-17
```

student@IT20:~$ ls

```
12.zargo                        jdk1.8.0_05
12.zargo~                        ly.zargo
43t5ju7i.zargo                   ly.zargo~
43t5ju7i.zargo~                   Music
aaa                             netbeans-8.0
Android                          NetBeansProjects
android-studio                    net.tcl
AndroidStudioProjects                ns-allinone-2.35
anibook.zargo                     oradiag_student
anibook.zargo~                     out.nam
apache-tomcat-8.0.3                 pass1
ArgoUML-0.34                       passport
arthy.txt                       passport1.zargo
arthy.txt.gpg                     passport1.zargo~
ATM.zargo                         passportt.zargo
ATM.zargo~                         passportt.zargo~
bank.zargo                        Pictures
bank.zargo~                        Public
Bank.zargo                        seetha.txt
Bank.zargo~                        seetha.txt.gpg
bookbank                          sema.c
BOOKbank                           'smily act stock.zargo'
bookbankbb.zargo                       'smily act stock.zargo~'
```

```
bookbankg                         'smilyclass.zargo'
bookbank.zargo                    'smilyclass.zargo~'
bookbank.zargo~                   'smily exam activity.zargo'
bookclass.uml                     'smily exam activity.zargo~'
bookclass.uml~                    'smily exam dep.zargo'
CLIENT.zargo                      'smily exam dep.zargo~'
CLIENT.zargo~                     'smily exam seq.zargo'
collo.smily.zargo                 'smily exam seq.zargo~'
collo.smily.zargo~                'smilyexam.zargo'
'database exp'                    'smilyexam.zargo~'
'db connectivity'                 'smily state exam.zargo'
dbms                              'smily state exam.zargo~'
'dbms answer key in 2nd internel.docx'          'smily stock case.zargo'
'dbmsquery.odt'                   'smily stock case.zargo~'
dep.smily.zargo                   'smily stock class.zargo'
dep.smily.zargo~                  'smily stock class.zargo~'
Desktop                           snap
document                          state.smily.zargo
Downloads                         state.smily.zargo~
'ER diagram for car insurance company.doc'  steffi2000.zargo
'exam collosmily.zargo'           steffi2000.zargo~
'exam collosmily.zargo~'          Templates
examples.desktop                  title
examreg.zargo                     'Untitled 1.odt'
examreg.zargo~                    Videos
examSRS                           Weka-3-7
expno1.zargo                      wekafiles
expno1.zargo~                     zz.zargo
freni                             zz.zargo~
glassfish-4.0
```

CONCLUSION:

Thus the secure data storage, secure data transmission and for creating digital signatures (GnuPG) was developed successfully.

Ex.No. 11b DATA TRANSMISSION USING DIGITAL SIGNATURES

AIM:

To implement the data transmission using digital signatures by sending keys using gnupg.

INTRODUCTION:

- GnuPG is a complete and free implementation of the OpenPGP standard as defined by RFC4880 (also known as *PGP*).
- GnuPG allows you to encrypt and sign your data and communications; it features a versatile key management system, along with access modules for all kinds of public key directories.
- GnuPG, also known as *GPG,*is a command line tool with features for easy integration with other applications.
- A wealth of frontend applications and libraries are available. GnuPG also provides support for S/MIME and Secure Shell (ssh).
- Since its introduction in 1997, GnuPG is Free Software (meaning that it respects your freedom). It can be freely used, modified and distributed under the terms of the GNU General Public License .
- The current version of GnuPG is 2.2.17. See the download page for other maintained versions.

DESCRIPTION:

- The **rng-tools**is a set of utilities related to random number generation in kernel. The main program is rngd, a daemon developed to check and feed random data from hardware device to kernel entropy pool.
- The $**urandom**() system **function**provides a mechanism for generating pseudo-random numbers. The **function**returns a new unsigned 32-bit random number each time it is called.
- --full-generate-key: A **key**is used to encrypt and decrypt whatever data is being encrypted/decrypted. A device or program used to generate **keys**is called a **keygenerator**or keygen.
- ASCII **armor**is a binary-to-textual encoding converter. ASCII **armor**is a feature of a type of encryption called pretty good privacy (PGP). ASCII **armor**involves encasing encrypted messaging in ASCII so that they can be sent in a standard messaging format such as email.
- **--send-keys [names]**:Sends one or more keystrokes to the active window as if they hadbeen entered at the keyboard. The **SendKeys**statement has two parameters. The first parameter keys is a string and is sent to the active window.
- **--list-keys**[names],**--list-public-keys**[names]: List all keys from the public keyrings,or just the ones given on the command line.
- **--recv-keys key IDs**: Import the keys with the given key IDs from a HKP keyserver.Option --keyserver must be used to give the name of this keyserver.

OUTPUT:

```
student@CSE-IC-LAB:~$ sudo apt-get install
gnupg[sudo] password for student: Reading package
lists... Done
Building dependency tree
Reading state information... Done
gnupg is already the newest version (2.2.4-1ubuntu1.2).
The following package was automatically installed and is no longer required:
gstreamer1.0-gtk3
Use 'sudo apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 20 not upgraded.

student@CSE-IC-LAB:~$ sudo apt-get install rng-tools
Reading package lists... Done Building dependency tree

Reading state information... Done
The following package was automatically installed and is no longer required:
gstreamer1.0-gtk3
Use 'sudo apt autoremove' to remove it.
The following NEW packages will be installed:
rng-tools
0 upgraded, 1 newly installed, 0 to remove and 20 not upgraded.
Need to get 22.5 kB of archives.
After this operation, 99.3 kB of additional disk space will be used.
Err:1 http://in.archive.ubuntu.com/ubuntu bionic/universe amd64 rng-tools amd64
5-0ubuntu4
  500 Operation not permitted [IP: 103.123.234.254 80]
E: Failed to fetch
http://in.archive.ubuntu.com/ubuntu/pool/universe/r/rng-tools/rng-tools_5-0ubuntu4_amd
64.deb 500 Operation not permitted [IP: 103.123.234.254 80]
E: Unable to fetch some archives, maybe run apt-get update or try with --fix-missing?

student@CSE-IC-LAB:~$ sudorngd -r
/dev/urandomsudo: rngd: command not found

student@CSE-IC-LAB:~$ gpg --full-generate-key
gpg (GnuPG) 2.2.4; Copyright (C) 2017 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Please select what kind of key you want:
    (1)   RSA and RSA (default)
    (2)   DSA and Elgamal
    (3)   DSA (sign only)
    (4)   RSA (sign only)
Your selection? 1
RSA keys may be between 1024 and 4096 bits long.
```

```
What keysize do you want? (3072) 3072
Requested keysize is 3072 bits

Please specify how long the key should be valid.
0 = key does not expire
<n> = key expires in n days
<n>w = key expires in n weeks
<n>m = key expires in n months
<n>y = key expires in n years

Key is valid for? (0) 1

Key expires at Thursday 05 September 2019 03:29:27 PM IST Is
this correct? (y/N) y

GnuPG needs to construct a user ID to identify your key.

Real name: AbinayaArthy
Email address: arthy@gmail.com
Comment: 4b
You selected this USER-ID:
"AbinayaArthy (4b) <arthy@gmail.com>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? O

We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
gpg: key 10AC5E617DC5B775 marked as ultimately trusted
gpg: directory '/home/student/.gnupg/openpgp-revocs.d' created
gpg: revocation certificate stored as
'/home/student/.gnupg/openpgp-revocs.d/75BD6877BADD422B446FD72510AC5E617
DC5B775.rev'
public and secret key created and signed.

pubrsa3072     2019-09-04     [SC]     [expires:     2019-09-05]
75BD6877BADD422B446FD72510AC5E617DC5B775
uidAbinayaArthy  (4b)  <arthy@gmail.com>  sub  rsa3072
2019-09-04 [E] [expires: 2019-09-05]

student@CSE-IC-LAB:~$ vi bhel.txt
```

```
student@CSE-IC-LAB:~$ gpg -e -r AbinayaArthy bhel.txt
gpg: checking the trustdb
gpg: marginals needed: 3 completes needed: 1 trust model: pgp
gpg: depth: 0 valid:       1 signed:      0 trust: 0-, 0q, 0n, 0m, 0f, 1u
gpg: next trustdb check due at 2019-09-05

student@CSE-IC-LAB:~$ ls

0001.pcap        client1.c          examples.desktopgrade.c        ns-allinone-2.35
rarpclient      serverpgm.class        stringfunc.c
444              client.c           exno1.java          grade.o           one3             rarpclient.class
simclient.class   swap
a.out            client.java        EX_NO_2b.c         http.class         one3.c
rarpclient.java    simclient.java       swap.c
arp.csv          clientpgm.class     file1.txt          httpclient.class one3.o
rarpserver.classsimserver.classswap.o
bhel.txt         dafturn-ofris.sh    fileclient.c       httpclient.java    onec.c
rarpserver.java   simserver.java       Templates
bhel.txt.gpg   Desktop             fileserver.c       http.java          Pictures         server1.c
snap             'Untitled Document 1'
chatclient1.c      Documents           file.txt          Music              prgm2b.c        server.c
srs.pcap         Videos
chatserver1.c      Downloads           grade         NetBeansProjects Public
server.java        strfunc.c

student@CSE-IC-LAB:~$ gpg -d -r AbinayaArthybhel.txt.gpg
gpg: encrypted with 3072-bit RSA key, ID 1E54B01355BBA830, created 2019-09-04
"AbinayaArthy (4b) <arthy@gmail.com>"

Security lab experiment 4b.
student@CSE-IC-LAB:~$      gpg      --armor      --output      publickey.txt      --export

AbinayaArthystudent@CSE-IC-LAB:~$ cat publickey.txt

-----BEGIN PGP PUBLIC KEY BLOCK-----
```

```
mQGNBF1vixoBDAC0p0Qt8eJ8tvnHMVdHy6GwR6zJXtIrJQfp8TXYiSMeWxNVESaB
BJ6u+92rGYqQKyUBwihrPQREBob2SC0wbiFMKX2dvxmbCDlD+WOFH2Z3tiRpGJF5
HDfZwKdYMl9oqnSHeDzKsP7oSmmDmsY02un5jxWnUvaAnFfN+/kPCxZRD1/s8HlH
IwaA4s3fv4+kWKSZ36rg+emlMAD01zFpY/oeeIMUq1LbymJU/tuD8Z0U1MVPed+c
jAnBRJMhdTQef4tUF+slfUmmSgcYRu8mX0FKCfkSwaRmyuJY5PZRblf7UPEAp9Z3
k7RL7icEcFPwoC3imClCnNT2aPLGsVfjib5606fqUYPAu8A3/bod7zUCLReWf4xW
lYMR+QiCGC48uq/OdfGam5OSrgBM/+GM9s6CfksN+Imh+7i/HGBSRRDDAVsWjnG6
bscoOfW76h/nPUoOL0Yam5Yx02C/f+zQEDd7kB0oncWdTRHdG0hA9tfVN6v/gbuY
xCpvwEx9iOGQAEcAEQEAAbQjQWJpbmF5YUFydGh5ICg0YikgPGFydGh5QGdtYWls
LmNvbT6JAdQEEwEKAD4WIQR1vWh3ut1CK0Rv1yUQrF5hfcW3dQUCXW+LGgIbAw
UJ
```

```
AAFRgAULCQgHAgYVCgkICwIEFgIDAQIeAQIXgAAKCRAQrF5hfcW3ddBLC/9yQiO
0
uqYGbtmWXRyzq0N2KYnyrlzjOko3Rmfigoxr+z6dJkbITQJpvDFWvi3Yai5YlQ8l
X7BihFBDHK0Joyj/kzJ7F8kDKJCb4wNhrM67vmkiLJwRxJbjnWn7TVqEYAvfIrDS
ezvDAknFJL7BH6jH4VI6MxXocGPA4/i5lM3iAZo9yqZes5XVYO9UhfU7/doRQjq6
SaDlofDOjdmdDagV1hLSqK5gwawDGzQDz6l5Q15b4VEUOlO6T38lkBCMHCn90Qwi
BFhRyB3me2BstoXLQDix7vHIG/15cLhmbsmQ6wtuHlne+XxyApQJ6618TEexyrAA

rcDRTjqw4HAcYFgzXxXz7NUPuytb15XdYqAqFKpKuOxZAdhc51pO5DMDhliY6sVd
cDwk32HV1JYoz1y6FDxEsG56Dqse84Nyg2VRl7We0rNSL31B4u4pr7PXJAiUWtBg
s44omYgoZcz9MOacYOo8ku/nYHCooaLF4HnEUevtcuDo383WpzFxlzv7uQ+5AY0E
XW+LGgEMAKuhGsdh0vd298FgJosRQ+LplfMKSBZ81NqehkaO/qBlaWmKm6SQ+aFS
+IZO672oBLKhGOPOZ+NVp2wlA2ue5yRMmC098mCcO6b8NapprUhWPLzZRRkxxe9
6
8pNoE/HbHBu814+12uX4GqEpc07MwxSBYSHHuMUBFFNk6onCme5TmymVRdbcGB
av
cOZDBCrZX3REukpZOFIOXzwNjHWWJlk36Cm2MAWsgJTzm6cJr2yg+OtNEym7cxil
WLbZ0hgVyehqPt9yRo2KRQ+sq6GaZV0GHwsAsORFFDZP+yHF6t+XW6L35i6KPDnN
8XlYvxJesROYIu8smB4v3qqWll95ExTULqWrX6yvGXEEnnIgFoaaejlSjDPm7SEh
sVGz/5Y8Lo7OmUkDdSHE1gqX7U1tgzkoruXkPI86SJLxyGMi5hIn8GfPnpprb+1j
28k1YLzWuAKToZ2XqbWS5uryeMl85Wd0shKC1I42KtLafKoDlhSjj7a/4jKxMn04
quCPhoKbAQARAQABiQG8BBgBCgAmFiEEdb1od7rdQitEb9clEKxeYX3Ft3UFAl1v
ixoCGwwFCQABUYAACgkQEKxeYX3Ft3UEXwwAoXgOs/wCgkJrS4rLiwh9Q6oBJPWr
ngh1Cwz0uteJ+89bgGB/Lq/0ixiUXXyxZgbB2btZuBlfItZQNta5hVdf6c7z2IoQ
BRl5RO36Es7Nk7gxl/KLa+/PFKoarZTXywa0+Lks+QQj0kIFgiunyqthPM4SN64O
Zrh1KxwP3hd28V0k7xM/KDOP2S6BLMI6aIIsqHeYof2rLg0uA9neQ2HJ5grMFaAO
P3IbBAf7vFgHp9HzCbtMY8Ly+4VI5UDrnqUMN2r7vEkAizmP2IuFp2ylR4UQWGVb
iz5LocB2XSOddMe2T181GUGReH5lkuFSS05wVhLVDvT4yMYOWpFDIvnH7AgsxqQJ
RfyO8mTJwY/WZ0PzXRd5j/pdtZBPTtn7+GYVFgwg9BRg90OT9MNEYtNg2UTPifql
4BUq/zpPIlM1pjOnt8m/+QOTUASaqEV1lFU3toe8ZhqqzpLKHkHKcZWSDJIIYFoG
7Lwu/SbV2xakYdQRWinhAAg7qJDTaxxLr6eP
=AT0a
-----END PGP PUBLIC KEY BLOCK-----
```

```
student@CSE-IC-LAB:~$ gpg --send-keys --keyserver keyserver.ubuntu.com
10AC5E617DC5B775

gpg: sending key 10AC5E617DC5B775 to hkp://keyserver.ubuntu.com

student@CSE-IC-LAB:~$ vi k.txt

student@CSE-IC-LAB:~$ cat k.txt

Cryptography and Network Security .

student@CSE-IC-LAB:~$ gpg --sign k.txt
```

```
student@CSE-IC-LAB:~$ ls
0001.pcap        client.c          EX_NO_2b.c     httpclient.class    one3.c
rarpclient.java     simserver.class     Templates
444              client.java       file1.txt      httpclient.java     one3.o
simserver.java     'Untitled Document 1'


rarpserver.class
a.out            clientpgm.class   fileclient.c   http.java    onec.c          rarpserver.java
snap             Videos
arp.csv          dafturn-ofris.sh  fileserver.c   k.txt        Pictures        server1.c
srs.pcap
bhel.txt         Desktop           file.txt  k.txt.gpg         prgm2b.c        server.c
strfunc.c
bhel.txt.gpg     Documents         grade          Music        Public          server.java
stringfunc.c
chatclient1.c    Downloads         grade.c        NetBeansProjects  publickey.txt
serverpgm.class swap
chatserver1.c    examples.desktop      grade.o        ns-allinone-2.35    rarpclient
simclient.class  swap.c
client1.c        exno1.java        http.class     one3         rarpclient.class simclient.java
swap.o

student@CSE-IC-LAB:~$ gpg --verify k.txt.gpg
gpg: Signature made Wednesday 04 September 2019 03:37:00 PM IST
gpg:                using RSA key 75BD6877BADD422B446FD72510AC5E617DC5B775
gpg: Good signature from "AbinayaArthy (4b) <arthy@gmail.com>" [ultimate]

student@CSE-IC-LAB:~$  gpg   --recv-keys   --keyserver   keyserver.ubuntu.com
10AC5E617DC5B775
gpg: key 10AC5E617DC5B775: "AbinayaArthy (4b) <arthy@gmail.com>" not changed
gpg: Total number processed: 1
gpg:              unchanged: 1

student@CSE-IC-LAB:~$              gpg             -d            -r
AbinayaArthyk.txt.gpgCryptography and Network Security .
gpg: Signature made Wednesday 04 September 2019 03:37:00 PM IST
gpg:                using RSA key 75BD6877BADD422B446FD72510AC5E617DC5B775
gpg: Good signature from "AbinayaArthy (4b) <arthy@gmail.com>" [ultimate]

student@CSE-IC-LAB:~$ gpg --list-keys
/home/student/.gnupg/pubring.kbx
---------------------
pub    rsa3072    2019-09-04    [SC]    [expires:    2019-09-05]
75BD6877BADD422B446FD72510AC5E617DC5B775 uid
```

```
[ultimate] AbinayaArthy (4b) <arthy@gmail.com> sub
rsa3072 2019-09-04 [E] [expires: 2019-09-05]
```

CONCLUSION:

Thus the data transmission was done using Digital Signatures by sending keys.