Maharaja Surajmal Institute Of Technology

Information Security

Practical Lab File

CSE (7th semester) Evening Shift

Submitted To:

Poonam Dhankar

Submitted By:

Anmol Mishra
S No. – 4
00396302717
BTech CSE (E)

TABLE OF CONTENTS

EXPERIMENT - 1(a)

AIM: Make an experiment to implement to WEP, WPA 2 PSK, and 802.1 XEAP security probabilities.

THEORY:

WEP:

- Wired equivalent privacy is a security algorithm for IEEE 802.11 wireless networks.
- Its intention is to provide data confidentiality comparable to that of a traditional wired h/w.
- WEP recognizable by the key of 10cr hexa-decimal digits was at once widely in use and was often the first security choice presented to users by router configuration tools.

ENCRYPTION DETAILS:

WEP uses the stream cipher RC4 for confidentiality and the CRC-32 checksum for integrity.

Authentication:

Two methods of authentication:

1. Open System authentication:

    - No credentials are required to the access point during authentication, any client can authenticate with the access point.
    - WEP keys can be used for encrypting data frames. At this point, the client must have the data frames.

2. Shared key authentication: WEP key is used for authentication in a four step challenge response handshake:

- The client sends an authentication request to the access point.
- The access point replies with a clear-test challenge.
- The client encrypts the challenge –test challenge using the configured WEP key and sends it back in another authentication request.
- The access point decrypts the response .If this matches the challenge text, the access point sends back a positive reply.

WPA 2 PSK:

- It stands for protected access 2 pre-shared keys.
- It is a method of securing out h/w using WPA2 with the use of the optional pre-shared key authentication, which was designed for house users without an enterprise authentication server. To encrypt n/w with WPA2 PSK we provide our router not with an encryption key, but rather with a plain English passphrase

between 8 & 63 characters long. Using a technology called TKIP (for temporal key Integrity Protocol) , that phrase along with the n/w SSI is used to generate unique encryption keys for each wireless client. And those encryption keys are constantly changed. Although WEP also supports pass phrase it does so only as a way to more create static keys, which are usually composed of the hex characters 0-9 and A-F.

802.1 x:

802.1 x uses three terms that we need to know:

- The user or client that wants to be authenticated is called supplicant.
- The actual server doing the authentication typically a RADIUS server is called the authentication typically a RADIUS server is called the authentication server.
- Wireless access point –authenticator. The protocol in 802.1x is called EAP encapsulation over LANs (EAPOL).

802.1 x helps wireless security

- IEEE 802.1x standard to help authenticate and secure both wireless and wired LANS.
- 802.1x authentications help mitigate many of the risks involved in using WEP.
- With 802.1x each station would have a unique WEP key for every session.
- 802.1x does not guarantee improved security.

EXPERIMENT - 1(b)

AIM: Implement RC4 Algorithm.

PROGRAM:

```
#include<bits/stdc++.h>
void swap (int *a,int *b) {
        int c=*a;
        *a=*b;
        *b=c;
}
char *create Keystream(char *str) {
         int length =strlen(str),i,j,k,l,s[256];
         char *key=malloc (length); for(i=0;i<256;i++)
         s[i]=i; j=0;
         for(i=0;i<256;i++) {
                 j=(j+s[i]+str[i%length])%256;
                 swap (&s[i],&s[j]);
}

//PRGA (Pseudo random generation algo) {
         i=0; j=0;
         for(i=0; i<length ;i++) {
                 i=(i+1)%256;
                 j=(j+s[i])%256;
                 k=s(s[i]+s[j])%256;
                 key[l]=(char)k;
         }
         key[length]=0;
         return key;
}
//function to perform RC4 encryption & decryption
char *RC4(char *str,char *key){
        char *msg=malloc [1024];
        int l,length = strlen[(str)];
        //Perform XOR operation
        for(i=0; i<length ;i++)
                 msg[i]=str[i] ^ key[i];
                 msg[length]=0;
        return msg;
}
```

```
 int main(){
        unsigned char *str = malloc(1024),
        *ex= malloc(1024),kdec=malloc(1024),*key=malloc(1024);
        int i;
        printf("\nEnter string:"); scanf("%[^\n]^",str);
        fflush(stdin);
        printf("\nInput:\n");
        for(i=0;i<strlen(str);i++)
                printf("\n%02x",str[i]);
        key=CreateKeystream(str);
        for(i=0;i<strlen(key);i++)
                enc=RC4(str,key);
                printf("\nEncrypted string\n");
                for(i=0;i<strlen(enc);i++)
                  printf("\n%02x",enc[i]); dec=RC4(enc,,key);
                  printf("\nDecrypted string\n");
                  for(i=0;i<strlen(dec);i++)
                          printf("\n%02x",dec[i]);
                          free(str);
        free(enc);
        free(dec);
        free(key);
        return 0;
}
```

EXPERIMENT - 2

AIM: Write a program for a simple RSA algorithm to encrypt and decrypt the data.

ALGORITHM:

1. Generate two large random primes, P and Q, of approximately equal size.
2. Compute N=PXQ
3. Compute Z=(p-1)Q-1
4. Choose an Integer E, 1<E<2 such that G<D (E,Z) =1
5. Compute the secret exponent D, 1<D<2 such that ex D=1
6. The Public key is (nE) and the private key is (n,D).

SOURCE CODE:

```
#include<bits/stdc++.h>
using namespace std;
long int p, q, n, t, flag, e[100], d[100], temp[100], j, m[100], en[100], i;
char msg[100];
int prime(long int);
void ce();
long int cd(long int);
void encrypt();
void decrypt();
int prime(long int pr) {
        int i;
        j = sqrt(pr);
        for (i = 2; i <= j; i++) {
            if (pr % i == 0)
            return 0;
        }
        return 1;
}
int main() {
        cout << "\nENTER FIRST PRIME NUMBER\n";
        cin >> p;
        flag = prime(p);
        if (flag == 0) {
                cout << "\nWRONG INPUT\n";
                exit(1);
        }
        cout << "\nENTER ANOTHER PRIME NUMBER\n";
        cin >> q;
```

```
        flag = prime(q);
        if (flag == 0 || p == q) {
            cout << "\nWRONG INPUT\n";
            exit(1);
        }
        cout << "\nENTER MESSAGE\n";
        fflush(stdin);
        cin >> msg;
        for (i = 0; msg[i] != '\0'; i++)
            m[i] = msg[i];
        n = p * q;
        t = (p - 1) * (q - 1);
        ce();
        cout << "\nPOSSIBLE VALUES OF e AND d ARE\n";
        for (i = 0; i < j - 1; i++)
            cout << e[i] << "\t" << d[i] << "\n";
            encrypt();
            decrypt();
            return 0;
}

void encrypt() {
        long int pt, ct, key = e[0], k, len; i = 0;
        len = strlen(msg);
        while (i != len) {
                pt = m[i]; pt = pt - 96; k = 1;
                for (j = 0; j < key; j++) {
                        k = k * pt;
                        k = k % n;
                }
                temp[i] = k;
                ct = k + 96;
                en[i] = ct; i++;
        }
        en[i] = -1;
        cout << "\nTHE ENCRYPTED MESSAGE IS\n";
        for (i = 0; en[i] != -1; i++)
        printf("%c", en[i]);
}

 void decrypt() {
         long int pt, ct, key = d[0], k; i = 0;
         while (en[i] != -1) {
                 ct = temp[i];
```

```
            k = 1;
            for (j = 0; j < key; j++) {
                k = k * ct;
                k = k % n;
            }
            pt = k + 96;
            m[i] = pt;
            i++;
        }
        m[i] = -1;
        cout << "\nTHE DECRYPTED MESSAGE IS\n";
        for (i = 0; m[i] != -1; i++)
        printf("%c", m[i]);
}
```

OUTPUT:

```
ENTER FIRST PRIME NUMBER
5

ENTER ANOTHER PRIME NUMBER
17

ENTER MESSAGE
welldone

POSSIBLE VALUES OF e AND d ARE
3       43
7       55
11      35
13      5
19      27
23      39

THE ENCRYPTED MESSAGE IS
1►||
►►x►
THE DECRYPTED MESSAGE IS
welldone
```

EXPERIMENT - 3

AIM: Write a program for a simple AES algorithm to encrypt and decrypt the data.

SOURCE CODE:

```
package isLabs;
import java.util.Scanner;
// AES Algorithm to encrypt and decrypt data
public class Experiment03 {
	public static void main(String[] args) {
	Scanner s = new Scanner(System.in);
	System.out.printf("Enter a message to encrypt:");
	String message = s.nextLine();
	System.out.printf("Enter key:");
	int key = s.nextInt();
	encryptMessage(message, key);
	System.out.printf("Enter a message to decrypt:");
	message = s.next();
	System.out.printf("Enter key:");
	key = s.nextInt();
	decryptMessage(message, key);
	s.close();
}

/*
*  @param message	message to decrypt
*  @param key		key use for decryption
*/
private static void decryptMessage(String message, int key) {
	int n = message.length();
	char ch;
	StringBuilder msg = new StringBuilder();
	for (int index = 0; index < n; index++) {
		ch = message.charAt(index);
		if ((ch >= 'a') && (ch <= 'z')) {
			ch += key;
			if (ch > 'z')
				ch = (char) (ch - 'z' + 'a' - 1);
			msg.append(ch);
		} else if ((ch >= 'A') && (ch <= 'Z')) {
			ch += key;
			if (ch > 'Z')
```

```
                    ch = (char) (ch - 'Z' + 'A' - 1);
                    msg.append(ch);
                }
            }
            System.out.println("Encrypted message: " + msg);
    }

    /*
    *  @param message          message to encrypt
    *  @param key              key use for decryption
    */
    private static void encryptMessage(String message, int key) {
            int n = message.length();
            char ch;
            StringBuilder msg = new StringBuilder();
            for (int index = 0; index < n; index++) {
                ch = message.charAt(index);
                if ((ch >= 'a') && (ch <= 'z')) {
                    ch -= key;
                    if (ch < 'a')
                    ch = (char) (ch + 'z' - 'a' + 1);
                    msg.append(ch);
                } else if ((ch >= 'A') && (ch <= 'Z')) {
                    ch -= key;
                    if (ch < 'A')
                    ch = (char) (ch + 'Z' - 'A' + 1);
                    msg.append(ch);
                }
            }
            System.out.println("Decrypted message: " + msg);
    }
```

OUTPUT:

```
<terminated> Experiment03 [Java Application] /usr/lib/jvm/java-11-openjdk-amd64/bin/java (22-Sep-2020, 10:14:14 pm)
Enter a message to encrypt:axzd
Enter key:4
Decrypted message: wtvz
Enter a message to decrypt:wtvz
Enter key:4
\Encrypted message: axzd
```

EXPERIMENT - 4

AIM: Write a program to perform Caesar Cipher technique.

SOURCE CODE:

```
package isLabs;
import java.util.Scanner;
// to perform Ceaser Cipher Technique
public class Experiment04 {
	public static void main(String[] args) {
	Scanner s = new Scanner(System.in);
	System.out.println("Enter a message to encrypt: ");
	String message = s.nextLine();
	System.out.println("Enter Shift key: ");
	int key = s.nextInt();
	encryptMessage(message, key);
	s.close();
	}
	private static void decryptMessage(StringBuilder message, int key) {
		int n = message.length();
		StringBuilder decryptedMessage = new StringBuilder();
		for (int i = 0; i < n; i++) {
			char ch = message.charAt(i);
			if (Character.isUpperCase(ch)) {
				char code = (char) (((int) ch - key - 65) % 26 + 65);
				decryptedMessage.append(code);
			} else {
				char code = (char) (((int) ch - key - 97) % 26 + 97);
				decryptedMessage.append(code);
			}
		}
		System.out.println("decrypted Message is: " + decryptedMessage);
	}
	private static void encryptMessage(String message, int key) {
```

```
        int n = message.length();
        StringBuilder encryptedMessage = new StringBuilder();
        for (int i = 0; i < n; i++) {
            char ch = message.charAt(i);
            if (Character.isUpperCase(ch)) {
                char code = (char) (((int) ch + key - 65) % 26 + 65);
                encryptedMessage.append(code);
            } else {
                char code = (char) (((int) ch + key - 97) % 26 + 97);
                encryptedMessage.append(code);
            }
        }
        System.out.println("encrypted Message is: " + encryptedMessage);
        decryptMessage(encryptedMessage, key);
    }
}
```

OUTPUT:

EXPERIMENT - 5

AIM: Write a program to perform MONO-ALPHABETIC CIPHER technique.

SOURCE CODE:

```
package isLabs;
import java.util.Scanner;
//to perform MONO-ALPHABETIC CIPHER technique
public class Experiment05 {
	public static char p[] = { 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r',
	's', 't', 'u', 'v', 'w', 'x', 'y', 'z' };
	public static char ch[] = { 'Q', 'W', 'E', 'R', 'T', 'Y', 'U', 'I', 'O', 'P', 'A', 'S', 'D', 'F', 'G',
	'H', 'J', 'K', 'L', 'Z', 'X', 'C', 'V', 'B', 'N', 'M' };
	public static String doEncryption(String s) {
		char c[] = new char[(s.length())];
		for (int i = 0; i < s.length(); i++)
			for (int j = 0; j < 26; j++)
				if (p[j] == s.charAt(i)) c[i] = ch[j]; break;
		return (new String(c));
	}
	public static String doDecryption(String s) {
		char p1[] = new char[(s.length())];
		for (int i = 0; i < s.length(); i++)
			for (int j = 0; j < 26; j++)
				if (ch[j] == s.charAt(i))
					p1[i] = p[j]; break;
		return (new String(p1));
	}
	public static void main(String args[]) {
		Scanner sc = new Scanner(System.in);
		System.out.println("Enter the message: ");
		String en = doEncryption(sc.next().toLowerCase());
		System.out.println("Encrypted message: " + en);
		System.out.println("Decrypted message: " + doDecryption(en));
```

```
            sc.close();
        }
    }
```

OUTPUT:

EXPERIMENT - 6

AIM: Write a program to perform encryption/decryption using Hill Cipher algorithm.

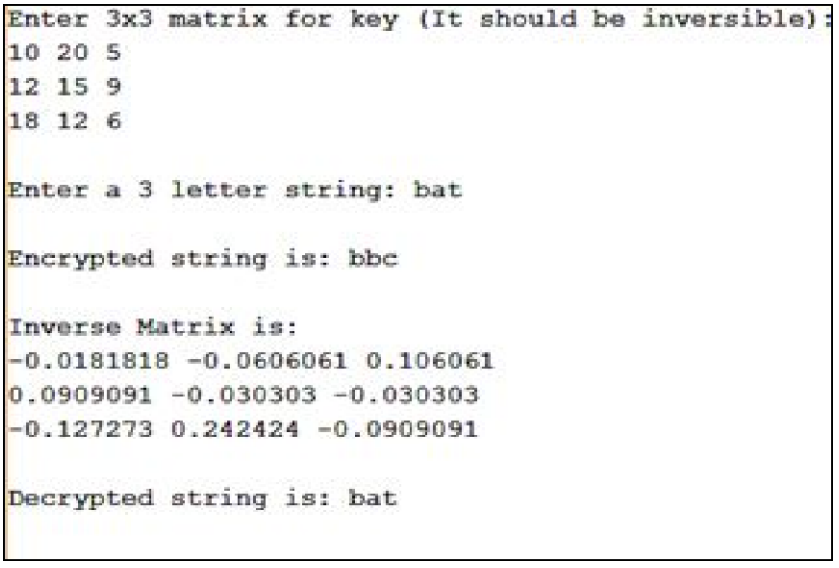
SOURCE CODE:

```
#include<bits/stdc++.h>
using namespace std;
float encrypt[3][1], decrypt[3][1], a[3][3], b[3][3], mes[3][1], c[3][3];
void encryption(); //encrypts the message
void decryption(); //decrypts the message
void getKeyMessage(); //gets key and message from user
void inverse(); //finds inverse of key matrix

int main() {
        getKeyMessage();
        encryption();
        decryption();
}

void encryption() {
        int i, j, k;
        for (i = 0; i < 3; i++)
                for (j = 0; j < 1; j++)
                        for (k = 0; k < 3; k++)
                                encrypt[i][j] = encrypt[i][j] + a[i][k] * mes[k][j];
        cout << "\nEncrypted string is: ";
        for (i = 0; i < 3; i++)
                cout << (char)(fmod(encrypt[i][0], 26) + 97);
}

void decryption() {
        int i, j, k;
        inverse();
```

```
        for (i = 0; i < 3; i++)
                for (j = 0; j < 1; j++)
                        for (k = 0; k < 3; k++)

                                decrypt[i][j] = decrypt[i][j] + b[i][k] * encrypt[k][j];
        cout << "\nDecrypted string is: ";
        for (i = 0; i < 3; i++)
                cout << (char)(fmod(decrypt[i][0], 26) + 97);
        cout << "\n";
}

void getKeyMessage() {
        int i, j;
        char msg[3];
        cout << "Enter 3x3 matrix for key (It should be inversible):\n";
         for (i = 0; i < 3; i++)
                for (j = 0; j < 3; j++) {
                        cin >> a[i][j];
                        c[i][j] = a[i][j];
                }
        cout << "\nEnter a 3 letter string: "; cin >> msg;
        for (i = 0; i < 3; i++)
                mes[i][0] = msg[i] - 97;
}

void inverse() {
        int i, j, k; float p, q;
        for (i = 0; i < 3; i++)
                for (j = 0; j < 3; j++) {
                        if (i == j) {b[i][j] = 1;}
                        else {b[i][j] = 0;}
                }
        for (k = 0; k < 3; k++) {
                for (i = 0; i < 3; i++) {
```

```
                    p = c[i][k];
                    q = c[k][k];
                    for (j = 0; j < 3; j++) {
                        if (i != k) {
                            c[i][j] = c[i][j] * q - p * c[k][j];
                            b[i][j] = b[i][j] * q - p * b[k][j];
                        }
                    }
                }
            }
        for (i = 0; i < 3; i++)
            for (j = 0; j < 3; j++)
                b[i][j] = b[i][j] / c[i][i];
            cout << "\n\nInverse Matrix is:\n";
        for (i = 0; i < 3; i++) {
            for (j = 0; j < 3; j++)
                cout << b[i][j] << " ";
            cout << "\n";
        }
}
```

OUTPUT:



EXPERIMENT - 7

AIM: Study and implement PlayFair Cipher.

THEORY:

Playfair cipher is a multi- alphabet letter encryption cipher, which deals with letters in plaintext as single units and renders these units into Ciphertext letters. The Playfair algorithm is based on the use of a 5X5 matrix of letters built using a keyword.

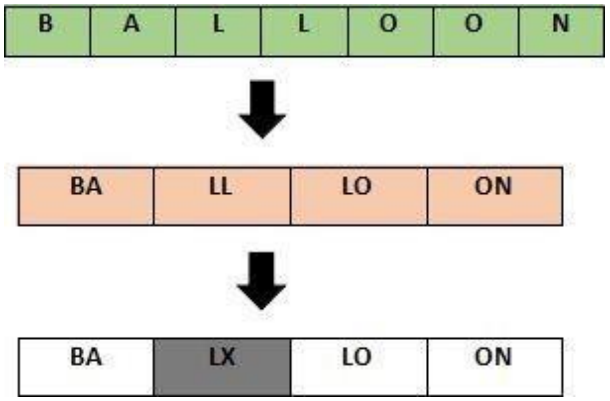
0	1	2	3	4
5	6	7	8	9
10	11	12	13	14
15	16	17	18	19
20	21	22	23	24

A	B	C	D	E
F	G	H	I/J	K
L	M	N	O	P
Q	R	S	T	U
V	W	X	Y	Z

Now the question is how to fill that 5x5 matrix? - To fill that, we need a keyword or a message, after that you fill the letters of the keyword into the 5x5 matrix from left to right and from top to bottom and then fill the remainders of the matrix with the remaining letters in alphabetical order.
The letter I and J are treated as one letter and they are settled in the same box of the matrix like this - I/J. Plaintext is encrypted two letters at a time, so initially you have to pair the plaintext.

RULES TO ENCRYPT THE MESSAGE

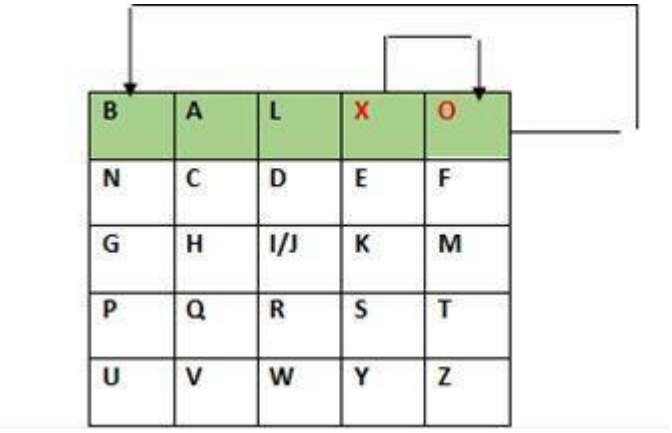
Step1:
Reoccurring plaintext letters that are in the same pair have to come apart with a filler such as "X", let's say Balloon is there for the message, so how are we going to proceed?



Here you can see we paired the plaintext, and we also can see that repeated letters are paired with a filler, like a pair in “Balloon” - -> “LL” is replaced with filler -- > “X” à “LX” and “LO”.

Step 2:

Two plaintext letters that drop in the same [ROW] in the matrix, have to be replaced with the letter to the right. Let’s say for example = “Balloon” is your plaintext.

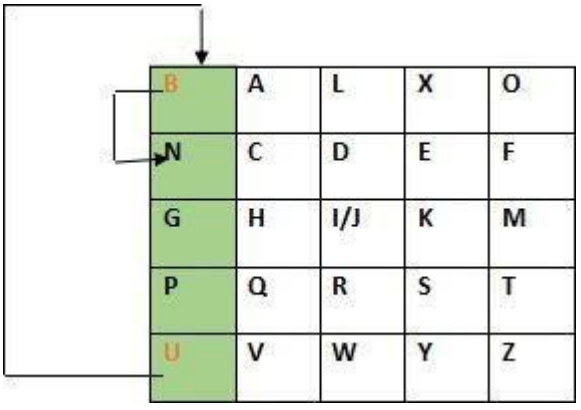


Now if you want to encrypt the pair of [X, O] - -> than it will be like this - > as [X, O] falls in the same row, then you have to replace the letter to the right, that is [O, B].

[X, O] - - > [O, B]

Step 3:

The above step is for the Row Matrix, this step will show you the process of Column matrix. Let’s say two plaintext letters that drop in the same [Column] in the matrix have to be replaced with the letter beneath, with the last element of the column circularly following the top element.

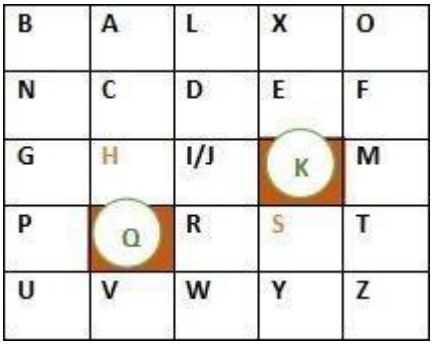


The step shows how to encrypt a pair - > [B, U], the step says you have to replace the letter beneath the letter you want to encrypt, with the last element of the column circularly following the top element.

[B, U] - -> [N, B]

Step 4

Otherwise you just have to find the corners of the letter to encrypt, let’s you want to encrypt [H, S] letter, then just find out the corner, you get your encrypted letter.



SOURCE CODE:

```
#include <bits/stdc++.h>
#define MX 5
void playfair(char ch1, char ch2, char key[MX][MX]) {
        int i, j, w, x, y, z;
        FILE * out;
        if ((out = fopen("cipher.txt", "a+")) == NULL)      printf("File Currupted.");
        for (i = 0; i < MX; i++) {
            for (j = 0; j < MX; j++) {
                if (ch1 == key[i][j]) {
                    w = i; x = j;
                } else if (ch2 == key[i][j]) {
                    y = i; z = j;
                }
            }
        }
        if (w == y) {
            x = (x + 1) % 5; z = (z + 1) % 5;
            printf("%c%c", key[w][x], key[y][z]);
            fprintf(out, "%c%c", key[ w][x], key[y][z]);
        } else if (x == z) {
            w = (w + 1) % 5; y = (y + 1) % 5;
            printf("%c%c", key[w][x], key[y][z]);
            fprintf(out, "%c%c", key[ w][x], key[y][z]);
        } else {
            printf("%c%c", key[w][z], key[y][x]);
            fprintf(out, "%c%c", key[ w][z], key[y][x]);
        }
        fclose(out);
}
void main() {
        int i, j, k = 0, l, m = 0, n;
        char key[MX][MX], keyminus[25], keystr[10], str[25]={0};
```

```
char alpa[26] = { 'A','B','C','D','E','F','G','H','I','J','K','L','M',
'N','O','P','Q','R','S','T','U','V','W','X','Y ','Z'};
printf("\nEnter key:"); gets(keystr);
printf("\nEnter the plain text:"); gets(str);
n = strlen(keystr);
for (i = 0; i < n; i++) {
        if (keystr[i] == 'j') keystr[i] = 'i';
        else if (keystr[i] == 'J') keystr[i] = 'I'; keystr[i] = toupper(keystr[i]);
}
for (i = 0; i < strlen(str); i++) {
        if (str[i] == 'j') str[i] = 'i';
        else if (str[i] == 'J') str[i] = 'I'; str[i] = toupper(str[i]);
}
j = 0;
for (i = 0; i < 26; i++) {
        for (k = 0; k < n; k++) {
                if (keystr[k] == alpa[i]) break;
                else if (alpa[i] == 'J') break;
        }
        if (k == n) {
        keyminus[j] = alpa[i]; j++;
        }
}
k = 0;
for (i = 0; i < MX; i++) {
        for (j = 0; j < MX; j++) {
                if (k < n) {
                        key[i][j] = keystr[k]; k++;
                  }  else {
                        key[i][j] = keyminus[m]; m++;
                  }
                  printf("%c ", key[i][j]);
          }
          printf("\n");
}
printf("\n\nEntered text :%s\nCipher Text :", str);
for (i = 0; i < strlen(str); i++) {
          if (str[i] == 'J') str[i] = 'I';
          if (str[i + 1] == '\0') playfair(str[i], 'X', key);
          else {
                if (str[i + 1] == 'J') str[i + 1] = 'I';
                if (str[i] == str[i + 1]) playfair(str[i], 'X', key);
                else {
                        playfair(str[i], str[i + 1], key);
```

```
                    i++;
                }
            }
        }
        getch();
}
```

OUTPUT:

```
Enter key:security

Enter the plain text:information
S E C U R
I T Y A B
D F G H K
L M N O P
Q V W X


Entered text :INFORMATION
Cipher Text :YLHMEPBYALOW
```

EXPERIMENT - 8

AIM: Write a program to perform encryption/ decryption using Diffie Hellman Key Exchange Technique.

SOURCE CODE:

```
public class Main {
        private static long power(long a, long b, long P) {
                if (b == 1)
                        return a;
                else
                        return ((long)((Math.pow(a, b)) % P));
        }
        public static void main(String args[]) {
                long P, G, x, a, y, b, ka, kb;
                P = 23; System.out.printf("The value of P : %d\n", P);
                G = 9;    System.out.printf("The value of G : %d\n\n", G);
                a = 4;    System.out.printf("The private key a for Alice : %d\n", a);
                x = power(G, a, P);
                b = 3;    System.out.printf("The private key b for Bob : %d\n\n", b);
                y = power(G, b, P);
                ka = power(y, a, P);
                kb = power(x, b, P);
                System.out.printf("Secret key for the Alice is : %d\n", ka);
                System.out.printf("Secret Key for the Bob is : %d\n", kb);
        }
}
```

OUTPUT:

```
The value of P : 23
The value of G : 9

The private key a for Alice : 4
The private key b for Bob : 3

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

EXPERIMENT - 9

AIM: Make a study of a simulation tool related to Information Security.

THEORY:

WIRESHARK - It is a free and open source packet analyser. It is used for network trouble - shooting analysis, software accommodation protocol development and education.

FUNCTIONALITY - It is very similar to tepdumps, it has a graphical front end, plus some integrated sorting and filtering options. Wireshark lets the user put n/w interface controller that supports a promiscuous model with that mode.

FEATURES - It is a data capturing program that understands the structure at different networks protocols.

SECURITY - Capturing raw network traffic from an interface requires elevated privileges on some platforms.

WHAT IS WIRESHARK?

Wireshark is known as the world's leading network traffic analyzer. It's the best tool for system administrators and IT professionals for troubleshooting network errors in real time. Wireshark quickly detects network issues such as latency, suspicious activity, and dropped packets. It can drill down into the traffic and find out the root cause of an issue. Usually, network administrators use Wireshark to resolve latency issues caused by equipment used to route traffic around the world and to monitor data exfiltration attempts against the business operations. Wireshark is a powerful yet free tool requiring extensive knowledge of the networking basics. To make the best use of the tool, administrators need to have a solid understanding of protocols such as TCP/IP and DHCP.

HOW DOES IT WORK?

Wireshark is a popular open-source tool to capture network packets and convert them into human-readable binary format. It provides every single detail of the organization's network infrastructure. It consists of devices designed to help measure the ins and outs of the network. The information collected through Wireshark can be utilized for various purposes, such as real-time or offline network analysis, identification of the traffic coming into your network, its frequency, and its latency between specific hops. This helps network administrators generate statistics based on real -time data.

Besides network monitoring, organizations use Wireshark for debugging programs, examining security issues, and learning network protocol internals. Wireshark is designed to efficiently perform packet-related functions and analyze and display the network metrics over the management console.

- Display packet information in the form of graphs, charts, and more
- Capture, export, search, save, and import live data packets
- Build reports based on real-time statistical data
- Filter packets based on the priority

Wireshark also offers a great user-interface as it's easy to use once the teams get familiar with the basics of capturing packets.

WHAT ARE THE CORE FEATURES OF WIRESHARK?

Wireshark consists of a rich feature set including the following:

- Live capture and offline analysis
- Rich VoIP analysis
- Read/write many different capture file formats
- Capture compressed files (gzip) and decompress them on the fly
- Deep inspection of hundreds of protocols
- Standard three-pane packet browser
- Captured network packets can be browsed via a GUI or TShark utility.
- Multi-platform easily ran on Linux, Windows, OS X, and FreeBSD
- Powerful display filters
- Output can be exported to XML, CSV, PostScript, or as a plain text
- Packet list can use coloring rules for quick and intuitive analysis.

HOW TO CAPTURE AND ANALYZE DATA PACKETS USING WIRESHARK?

One of the major functions of Wireshark is to capture data packets for conducting detailed network analysis. However, it can be tricky for beginners who aren't familiar with the steps involved. There are majorly three important steps:

- Get access to administrative privileges to start capturing the real -time data directly from the device.
- Choose the right network interface to capture packet data.
- Choose the right location within the network to capture packet data.

After following the above steps, the Wireshark is ready to capture packets. Usually, there are two capturing modes: promiscuous and monitor. Promiscuous mode sets the network interface to capture only the packets for which it's assigned. Monitor mode is used by Unix/Linux systems and sets the wireless interface to capture as much of the network as it can. The data gathered while capturing packets is displayed in a human - readable format, so it's easier to grasp.

Usually, promiscuous mode is used by system administrators to get a bird's-eye view of the network packets transfer. On disabling this mode, only a small snapshot of the network is provided, which isn't enough to conduct quality analysis. The promiscuous mode can easily be activated by clicking on the capture options provided in the dialog box. However,

promiscuous mode isn't available on every software or operating system. Therefore, users need to cross confirm about software compatibility either by visiting the Wireshark's website or using the Device manager to check the settings (in case you're a Windows user).

HOW TO ANALYZE THE CAPTURED NETWORK PACKETS?

Once the network data is captured, the list of network packets is displayed on the centralized management console or the dashboard. The screen consists of three panes: packet list, packet bytes, and packet details. However, to get important information on individual packets, users need to click on any of the panes mentioned above.

Packet List: The packet list pane consists of browsable captured network packets based on the timestamp to show exactly when the packet was captured, the packet's protocol name, the source and destination of the packet, and support information.

Packet Details: Packet details pane provides information about the chosen packet. Users can expand each section and apply filters to get information about specific items.

Packet Bytes: Packet bytes pane consists of the internal data of the packet the user selects. The data is displayed in a hexadecimal format by default.

HOW DOES WIRESHARK HELP IN MONITORING NETWORK PERFORMANCE?

Once the network packets are captured, the next step is to monitor and analyze them. Wireshark is a great tool to analyze and monitor the organization's active network. Before doing this, it's important to close down all the active applications on the network to reduce traffic, which helps in giving a clear picture of the network. Turning off all the applications won't result in packet loss; in fact, the sending and receiving of packets is still an ongoing process.

However, conceptualizing the huge amount of data altogether isn't easier. Therefore, Wireshark breaks these components into different forms to see what's going on within the network. To help users quickly understand and analyze the data coming in, Wireshark uses color coding, filters, network statistics to segregate the network data.

CONCLUSION

Wireshark is a simple, yet versatile and powerful network monitoring tool. It's easy to use and easy to learn. Besides monitoring, Wireshark offers additional network analysis features such as IO graphs, Color coding and filters.

EXPERIMENT - 10

AIM: Study the steps of VPN via packet tracer.

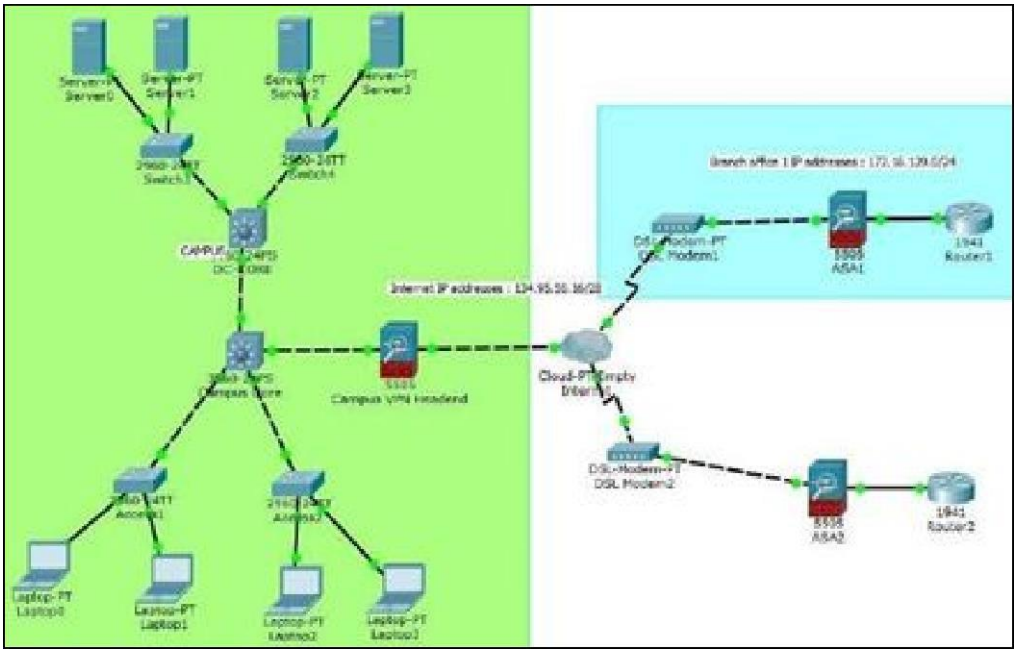
THEORY:

Objective:-

i. Enables security features
ii. Configure IPsec parameter OUR1.
iii. Configure IPsec parameter OUR2.
iv. Verify the IPsec

VPN. Part 1 – Enable security features
Step 1 – Activate key security module.
Ø R1(config) # license boot module.
Ø R1(config)# end
Ø R1(config)# copy
Ø R1(config)# reload

Part2 – Configure IPsec parameters.
Ø Test – connectivity.
Ø Identify intersecting traffic ob RV.
Ø Configure the crypt map on outgoing interface.



ADDRESSING TABLE:

NODE	INTERFACE	IP ADDRESS	SUBNET MASK	DEFAULT GATEWAY
R1	60/0 50/0/0	192.168.1.1 10.1.1.2	255.255.255.0 255.255.255.252	N/A N/A
R2	60/0 50/0/0 50/0/1	192.168.2.1 10.1.1.1 10.2.2.1	255.255.255.0 255.255.255.252 255.255.255.252	N/A N/A N/A
R3	G0/0 50/0/1	192.168.3.1 10.2.2.2	255.255.255.0 255.255.255.0	N/A
PC-A	N/C	192.168.1.3	255.255.255.0	192.168.1.1
PC-B	N/C	192.168.2.3	255.255.255.0	192.168.2.1
PC-C	N/C	192.168.3.3	255.255.255.0	192.168.3.1