

Experiment No – 1

Aim: Make an experiment to implement WEP/WPA2 PSK, 802.1x EAP security protocol.

WEP: WEP relies on a broken RC4 implementation and has severe flaws in various aspects of its protocol, which make breaking WEP near trivial. Anyone with a laptop and a \$20 Wi-Fi antenna can send special de-auth packets, which cause legitimate clients to re-authenticate to the AP. The attacker can then capture the initialization vectors from these re-authentication packets and use them to crack the WEP key in minutes. Due to the severity of the break, WEP is now considered deprecated

WPA: WPA improves upon this, by combining RC4 with TKIP, which helps defend against the IV-based attacks found in WEP. It also improves upon the old handshake mechanism, to make it more resistant to de-auth attacks. Whilst this makes a large improvement, vulnerabilities were found in the way that the protocol worked, allowing an attacker to break TKIP keys in about 15-20 minutes.

WPA2: WPA2 closes holes in WPA, and introduces an enhanced version of TKIP, as well as CCMP. The standard also brings support for AES, which provides even further security benefits. At current, there are no known generic attacks against WPA2.

By 2001, hacker attacks on WEP had made strengthened wireless security imperative. The IEEE began work on 802.11i, an improved standard. In 2003, rather than wait until final approval of the standard, the Wi-Fi Alliance created Wi-Fi Protected Access (WPA), which is based on a subset of the then-current 802.11i draft.

Authentication

WPA can be used in of two modes: either Personal or Enterprise.

- **Personal mode:** This utilizes manually configured keys in the same manner as WEP. All clients use the same initial master key.
- **Enterprise mode:** The AP uses Extensible Authentication Protocol (EAP) to negotiate a pair-wise master key with each client individually. The AP then verifies the identity of the client with an 802.1x server. The result is that each client that is permitted to use the network is validated against information configured in the 802.1x server and uses a key different from the keys used by other clients.

EAP, defined by RFC 3748, is an extensible protocol. It does not define a specific authentication protocol but simply specifies a set of functions and formats. A large number of EAP methods have been defined. The Wi-Fi Alliance has chosen a subset of the available methods.

In Enterprise mode, after successfully authenticating -- using one of the EAP methods -- the client and AP receive messages from the 802.1x server that both use to create the PMK. They then exchange messages to create the PTK. The PTK is then used to encrypt and decrypt messages.

In both cases, Personal and Enterprise, a group temporal key (GTK) is created during the exchange between the client and AP. The GTK is used to decrypt broadcast and multi-cast messages.

WPA2 also adds methods to speed the handoff as a client moves from AP to AP. The process of authenticating with an 802.1x server and generating keys takes enough time to cause a noticeable interruption of a voice over wireless call. WPA2 specifies ways in which a client can pre-authorize with neighboring APs. APs and clients can also retain keys so that a client returning to an AP can quickly resume communication.

These security features protect the data traffic on your wireless LAN:

- WEP (Wired Equivalent Privacy)—WEP is an 802.11 standard encryption algorithm originally designed to provide your wireless LAN with the same level of privacy available on a wired LAN. However, the basic WEP construction is flawed, and an attacker can compromise the privacy with reasonable effort.
- TKIP (Temporal Key Integrity Protocol)—TKIP is a suite of algorithms surrounding WEP that is designed to achieve the best possible security on legacy hardware built to run WEP. TKIP adds four enhancements to WEP:

A per-packet key mixing function to defeat weak-key attacks

- A new IV sequencing discipline to detect replay attacks
 - A cryptographic message integrity Check (MIC), called *Michael*, to detect forgeries such as bit flipping and altering packet source and destination
 - An extension of IV space, to virtually eliminate the need for re-keying
- CKIP (Cisco Key Integrity Protocol)—Cisco's WEP key permutation technique based on an early algorithm presented by the IEEE 802.11i security task group.
 - CMIC (Cisco Message Integrity Check)—Like TKIP's *Michael*, Cisco's message integrity check mechanism is designed to detect forgery attacks.

Creating WEP Keys

Beginning in privileged EXEC mode, follow these steps to create a WEP key and set the key properties: This example shows how to create a 128-bit WEP key in slot 2 for VLAN 1 and sets the key as the transmit key:

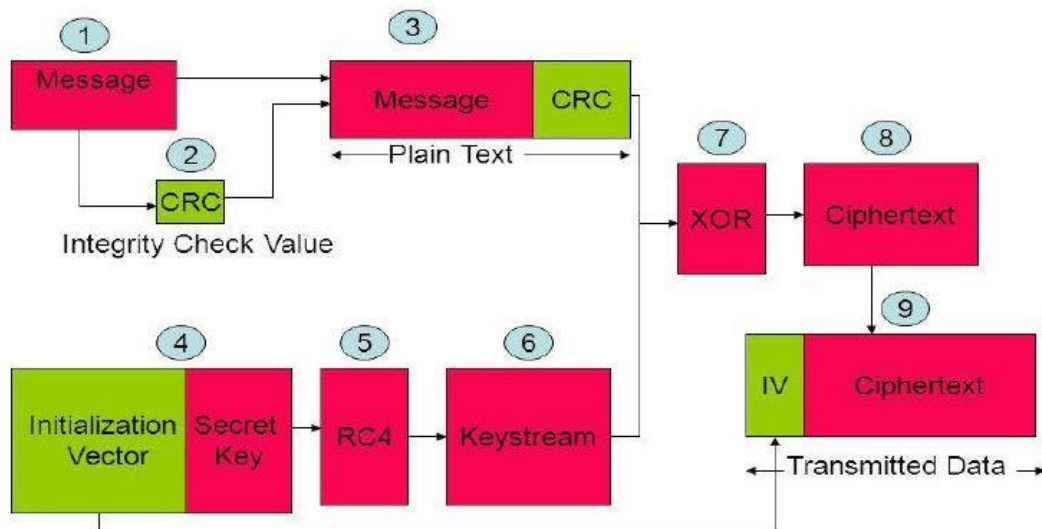
```
bridge# configure terminal
```

```
bridge(config)# configure interface dot11radio 0
```

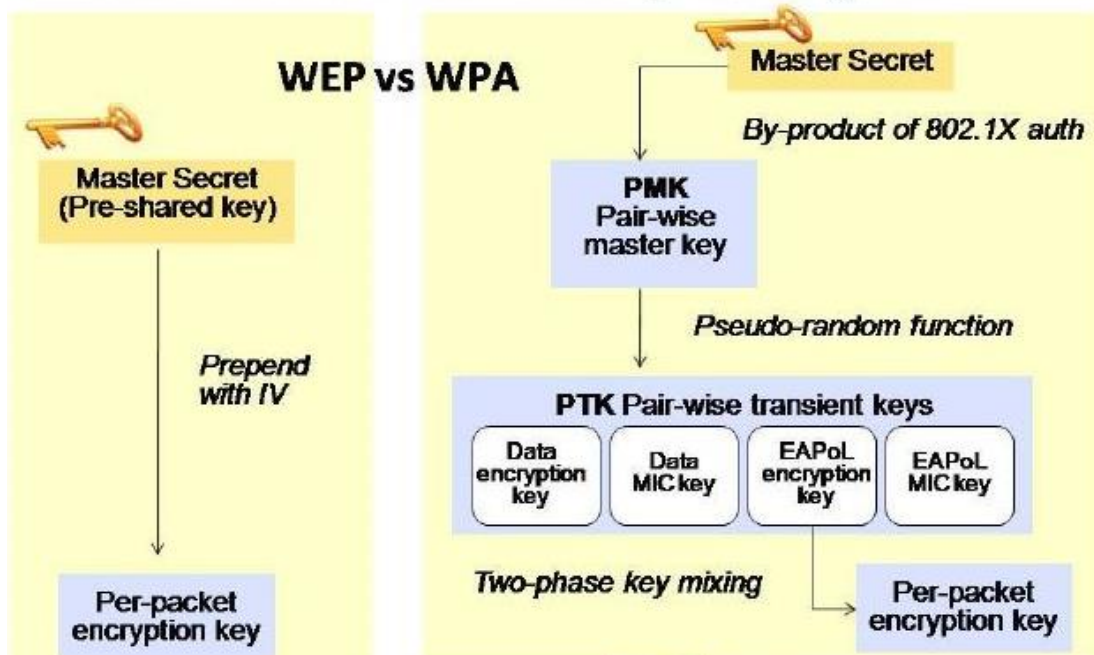
```
bridge(config-if)# encryption vlan 1 key 2 size 128 12345678901234567890123456 transmit-key
```

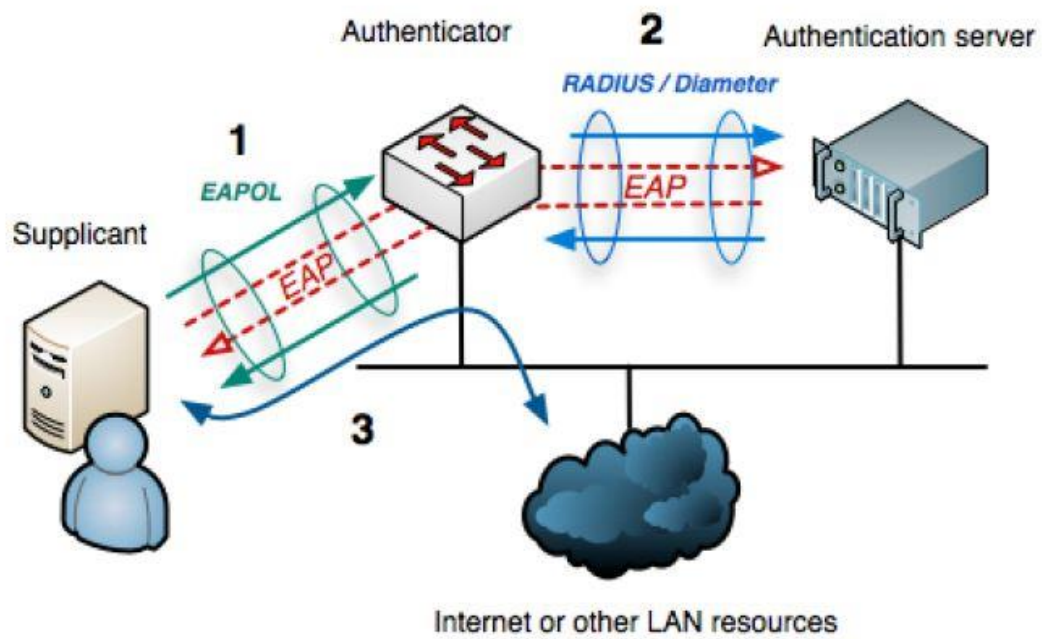
```
bridge(config-if)# end
```

Flow diagram:



802.1X and Wireless > Re-Engineering WEP





Experiment No – 2

Aim: Implement firewall through App to login into bank-site; to implement E-commerce, debit card transaction through payment gateway.

Technology Used: Cyber-roam.

1. Creating Firewall rules: Select Firewall → Create Rule
2. For every field

Source: Specify source zone and host IP address/network address to which the rule applies.

To define host group based firewall rule you need to define host group.

Under Select Address, click Create Host Group to define host group from firewall rule itself or from **Firewall → Host Group → Create**

Under Select Address, click Add Host to define host group from firewall rule itself rule itself or from **Firewall → Host → Add Host**

Check Identity

Click Enable to check the user identity.

Enable check identity to apply following policies per user:

- Internet Access policy for Content Filtering (User's Internet access policy will be applied automatically but will not be effective till the Web and Content Filtering module is subscribed).
- Schedule Access
- IDP (User's IDP policy will be applied automatically but will not be effective till the IDP module is subscribed)
- Anti-Virus scanning (User's anti-virus scanning policy will be applied automatically but it will not be effective till the Gateway Anti-Virus module is subscribed)
- Anti-Spam scanning (User's anti-spam scanning policy will be applied automatically but it will not be effective till the Gateway Anti-Spam module is subscribed)
- Bandwidth policy - User's bandwidth policy will be applied automatically
- The policy selected in Route through Gateway is the static routing policy that is applicable only if more than one gateway is defined and used for load balancing and limit access to available services.

Destination

Under Select Address, click Create Host Group to define host group from firewall rule itself or from **Firewall → Host Group → Create**

Under Select Address, click Add Host to define host group from firewall

Rule itself rule itself or from **Firewall → Host → Add Host**

Service/Service Group

Under Select Here, click Create Service Group to define service group from firewall rule itself rule itself or from **Firewall → Service → Create Service**

Cyberoam provides several standard services and allows creating the custom services also. Under Select Here, click Create Service to define service from firewall rule itself rule itself or from **Firewall → Service → Create Service**

Action

Select rule action

Accept – Allow access

Drop – Silently discards

Reject – Denies access and 'ICMP port unreachable' message will be sent to the source.

When sending response it might be possible that response is sent using a different interface than the one on which request was received. This may happen depending on the Routing configuration done on Cyberoam.

For example,

If the request is received on the LAN port using a spoofed IP address (public IP address or the IP address not in the LAN zone network) and specific route is not defined, Cyberoam will send a response to these hosts using default route. Hence, response will be sent through the WAN port.

Create Firewall Rule

[Wizard](#)
[Cyberoam](#)
[Help](#)
[Logout](#)

Matching Criteria

Source *

Select Zone

Select Address

☐ Check Identity

Destination*

Select Zone

Select Address

Hosts

[Add Host...]

Any Host

#Port D - 10.10.3.1/255.255

#Port C - 10.10.2.1/255.255

#Port B - 203.88.135.202/255

#Port A - 192.168.1.27/255

192.168.1.241

203.88.135.190

203.88.128.11

192.168.1.76

192.168.1.162

203.88.140.116

krundl

192.168.1.33

192.168.1.34

192.168.1.32

Service/Service Group*

Select Here

Apply Schedule

All the Time

Firewall Action When Criteria Match

Action*

Select Here

☐ Apply Source NAT

Advanced Settings (Destination NAT, IDP Policy, Internet Access Policy, Bandwidth Policy, Route Through Gateway, Anti-Virus & Anti-Spam Settings, Log Traffic)

Destination NAT Settings

Destination NAT Policy

Select Here

Policies

IDP Policy

Select Here

Internet Access Policy

Select Here

Bandwidth Policy

Select Here

Route Through Gateway

Load Balance

Anti-Virus & Anti-Spam Settings

Scan Protocol(s)

☐ SMTP
 ☐ POP3
 ☐ IMAP
 ☐ HTTP

Log Traffic

Log Traffic

☐ Enable

Description

Description

Create

Cancel

Create Firewall Rule

[Wizard](#)
[Cyberoam](#)

Matching Criteria

Source *

Select Zone

Select Address

☐ Check Identity

Destination*

Select Zone

Select Address

Host Groups

[Create Host Group...]

Hosts

[Add Host...]

Any Host

#Port D - 203.88.135.210/255

#Port C - 203.88.140.116/255

#Port B - 192.169.1.25/255

Service/Service Group*

Select Here

Apply Schedule

All the Time

Firewall Action When Criteria Match

Action*

Select Here

Experiment No – 3

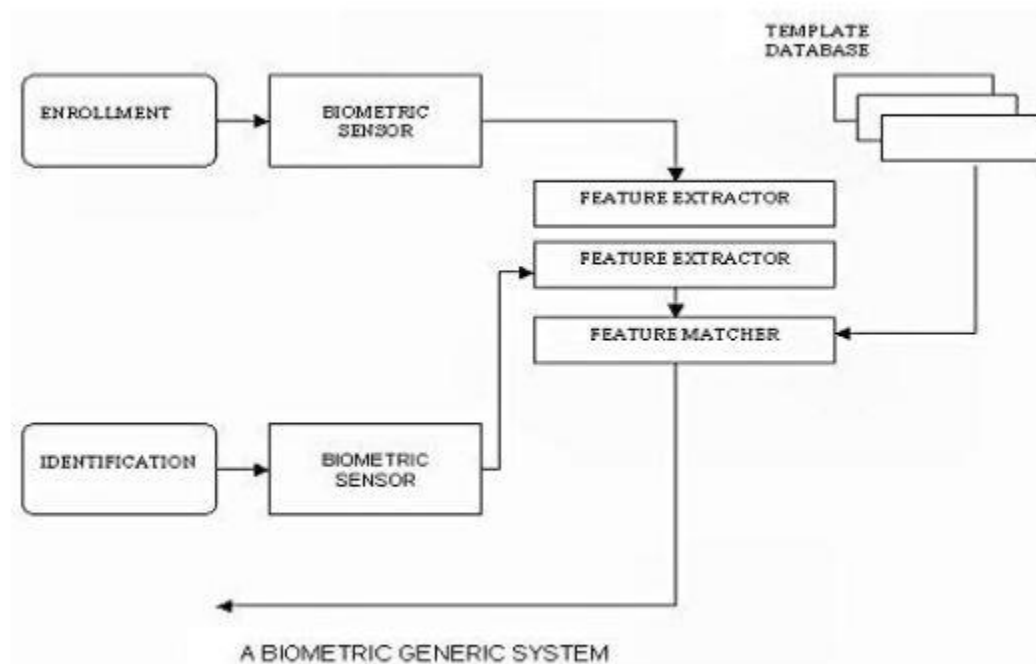
Aim: Implement biometric system to have physical security through different access control permissions.

Language Used: MATLAB

Algorithm:

1. Enrollment: Add the biometric sample to the database.
2. Biometric sensor will sense the data.
3. Feature Extractor extracts the desired features that are necessary.
4. Feature matcher matches the enrolled data with the stored database.
5. It results whether the individual is authentic or not.

Flow Chart:



Result: The desired biometric system performs in a correct manner, thus performs normally.

Experiment No – 4

Aim: Implement RSA algorithm.

Code:

```
import java.math.BigInteger;
import java.util.Random;
import java.io.*;

public class RSASexp5 {
    public static void main(String[] args) throws IOException {
        BigInteger p,q,N,z,e,d;
        int bitlength = 1024;
        Random r;
        r = new Random();
        p = BigInteger.probablePrime(bitlength, r);
        q = BigInteger.probablePrime(bitlength, r);
        N = p.multiply(q);
        z = p.subtract(BigInteger.ONE).multiply(q.subtract(BigInteger.ONE));
        e = BigInteger.probablePrime(bitlength/2, r);
        while (z.gcd(e).compareTo(BigInteger.ONE) > 0 && e.compareTo(z) < 0 )
        {
            e.add(BigInteger.ONE);
        }
        d = e.modInverse(z);
        DataInputStream in=new DataInputStream(System.in);
        String msg ;
        System.out.println("Enter the plain text:");
        msg = in.readLine();
        // encrypt
        int sum=0, rem=0;
        byte[] encrypted = new BigInteger(msg.getBytes()).modPow(e, N).toByteArray();
```

```

for(int i=0; i<encrypted.length; i++)
{
    sum +=(int)(Math.abs(encrypted[i]));
}
StringBuffer sb = new StringBuffer();

for(int i=0; i<String.valueOf(sum).length(); i++)
{
    rem = sum%10;
    sum/=10;
    sb.append((char)(rem+97));
}
System.out.println("Encrypted text is: "+sb);
// decrypt
byte[] decrypted = new BigInteger(encrypted).modPow(d, N).toByteArray();
System.out.println("Decrypted String: " + new String(decrypted));
}
}

```

Output:



```

C:\Windows\System32\cmd.exe
D:\J2se>java RSAISexp5
Enter the plain text:
hmritm
Encrypted text is: egd
Decrypted String: hmritm
D:\J2se>

```

Experiment No – 5

Aim: Implement DES Algorithm

Code:

```
import java.io.*;
import java.lang.*;
class SDES
{ public int K1, K2;
  public static final int P10[] = { 3, 5, 2, 7, 4, 10, 1, 9, 8, 6};
  public static final int P10max = 10;
  public static final int P8[] = { 6, 3, 7, 4, 8, 5, 10, 9};
  public static final int P8max = 10;
  public static final int P4[] = { 2, 4, 3, 1};
  public static final int P4max = 4;
  public static final int IP[] = { 2, 6, 3, 1, 4, 8, 5, 7};
  public static final int IPmax = 8;
  public static final int IPI[] = { 4, 1, 3, 5, 7, 2, 8, 6};
  public static final int IPImax = 8;
  public static final int EP[] = { 4, 1, 2, 3, 2, 3, 4, 1};
  public static final int EPmax = 4;
  public static final int S0[][] = {{ 1, 0, 3, 2},{ 3, 2, 1, 0},{ 0, 2, 1, 3},{ 3, 1, 3, 2}};
  public static final int S1[][] = {{ 0, 1, 2, 3},{ 2, 0, 1, 3},{ 3, 0, 1, 2},{ 2, 1, 0, 3}};
  public static int permute( int x, int p[], int pmax)
  {
    int y = 0;
    for( int i = 0; i < p.length; ++i)
    {
      y <<= 1;
      y |= (x >> (pmax - p[i])) & 1;
    }
    return y;
  }

  public static int F( int R, int K)
  {
    int t = permute( R, EP, EPmax) ^ K;
    int t0 = (t >> 4) & 0xF;
    int t1 = t & 0xF;
    t0 = S0[ ((t0 & 0x8) >> 2) | (t0 & 1) ][ (t0 >> 1) & 0x3 ];
    t1 = S1[ ((t1 & 0x8) >> 2) | (t1 & 1) ][ (t1 >> 1) & 0x3 ];
    t = permute( (t0 << 2) | t1, P4, P4max);
    return t;
  }
}
```

```

}
public static int fK( int m, int K)
{
int L = (m >> 4) & 0xF;
int R = m & 0xF;
return ((L ^ F(R,K)) << 4) | R;
}
public static int SW( int x)
{
return ((x & 0xF) << 4) | ((x >> 4) & 0xF);
}
public byte encrypt( int m)
{
System.out.println("\nEncryption Process Starts.....\n\n");
m = permute( m, IP, IPmax);
System.out.print("\nAfter Permutation : ");
printData( m, 8);
m = fK( m, K1);
System.out.print("\nbefore Swap : ");
printData( m, 8);
m = SW( m);
System.out.print("\nAfter Swap : ");
printData( m, 8);
m = fK( m, K2);
System.out.print("\nbefore IP inverse : ");
printData( m, 8);
m = permute( m, IPI, IPImax);
return (byte) m;
}
public byte decrypt( int m)
{
System.out.println("\nDecryption Process Starts.....\n\n");
printData( m, 8);
m = permute( m, IP, IPmax);
System.out.print("\nAfter Permutation : ");
printData( m, 8);
m = fK( m, K2);
System.out.print("\nbefore Swap : ");
printData( m, 8);
m = SW( m);
System.out.print("\nAfter Swap : ");
printData( m, 8);
m = fK( m, K1);
System.out.print("\nBefore Extraction Permutation : ");
printData( m, 4);
m = permute( m, IPI, IPImax);
System.out.print("\nAfter Extraction Permutation : ");

```

```

printData( m, 8);
return (byte) m;
}
public static void printData( int x, int n)
{
int mask = 1 << (n-1);
while( mask > 0)
{
System.out.print( ((x & mask) == 0) ? '0' : '1');
mask >>= 1;
}
}
public SDES( int K)
{
K = permute( K, P10, P10max);
int t1 = (K >> 5) & 0x1F;
int t2 = K & 0x1F;
t1 = ((t1 & 0xF) << 1) | ((t1 & 0x10) >> 4);
t2 = ((t2 & 0xF) << 1) | ((t2 & 0x10) >> 4);
K1 = permute( (t1 << 5) | t2, P8, P8max);
t1 = ((t1 & 0x7) << 2) | ((t1 & 0x18) >> 3);
t2 = ((t2 & 0x7) << 2) | ((t2 & 0x18) >> 3);
K2 = permute( (t1 << 5) | t2, P8, P8max);
}
}
// Main operations
public class DES
{
public static void main( String args[]) throws Exception
{
DataInputStream inp=new DataInputStream(System.in);
System.out.println("Enter the 10 Bit Key :");
int K = Integer.parseInt(inp.readLine(),2);
SDES A = new SDES( K);
System.out.println("Enter the 8 Bit message To be Encrypt : ");
int m = Integer.parseInt(inp.readLine(),2);
System.out.print("\nKey K1: ");
SDES.printData( A.K1, 8);
System.out.print("\nKey K2: ");
SDES.printData( A.K2, 8);
m = A.encrypt( m);
System.out.print("\nEncrypted Message: ");
SDES.printData( m, 8);
m = A.decrypt( m);
System.out.print("\nDecrypted Message: ");
SDES.printData( m, 8);
}
}

```

Output:

```
C:\Windows\System32\cmd.exe

D:\J2se>java DES
Enter the 10 Bit Key :
1011011010
Enter the 8 Bit message To be Encrypt :
10110110

Key K1: 11110101
Key K2: 01100011
Encryption Process Starts.....

After Permutation : 01111001
before Swap : 00001001
After Swap : 10010000
before IP inverse : 10000000
Encrypted Message: 01000000
Decryption Process Starts.....

01000000
After Permutation : 10000000
before Swap : 10010000
After Swap : 00001001
Before Extraction Permutation : 1001
After Extraction Permutation : 10110110
Decrypted Message: 10110110
D:\J2se>
```

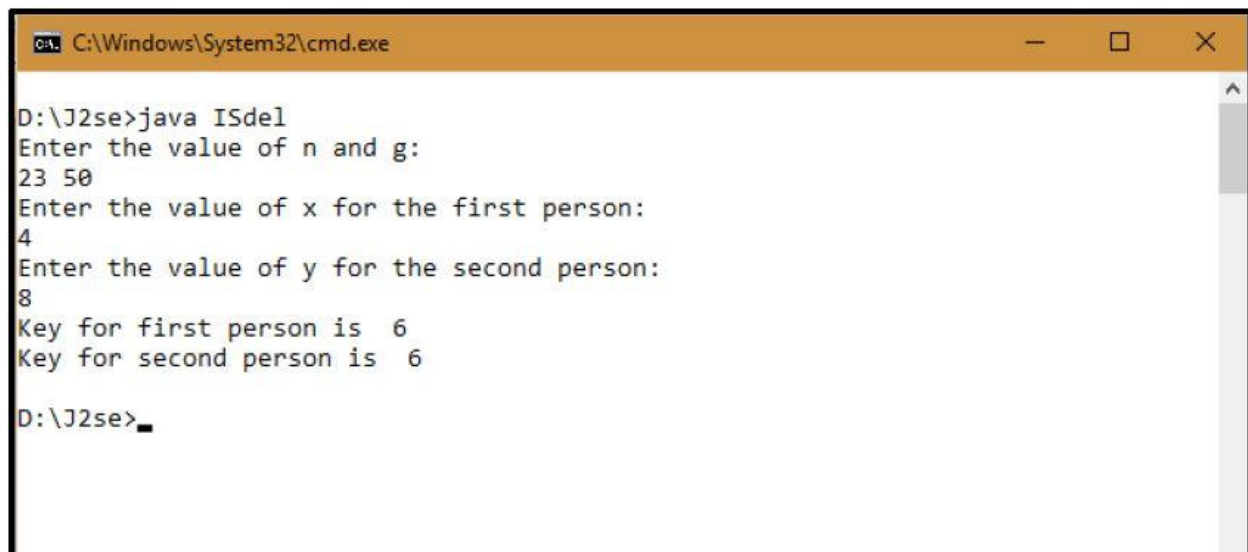
Experiment No – 6

Aim: Implement Diffie-Hellman algorithm

Code:

```
import java.util.Scanner;
import java.util.*;
public class ISdel {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n,g,x,a,y,b;
        System.out.println("Enter the value of n and g: ");
        n = sc.nextInt();
        g = sc.nextInt();
        System.out.println("Enter the value of x for the first person: ");
        x = sc.nextInt();
        a = (int) power(g,x,n);
        System.out.println("Enter the value of y for the second person: ");
        y = sc.nextInt();
        b = (int) power(g,y,n);
        System.out.println("Key for first person is "+" "+(int) power(b,x,n));
        System.out.println("Key for second person is "+" "+(int) power(a,y,n));
    }
    public static long power(int a,int b,int mod)
    {
        long t;
        if(b==1)
            return a;
        t=power(a,b/2,mod);
        if(b%2==0)
            return (t*t)%mod;
        else
            return (((t*t)%mod)*a)%mod;
    }
    public static long calculateKey(int a,int x,int n)
    {
        return power(a,x,n);
    }
}
```

Output:



```
C:\Windows\System32\cmd.exe
D:\J2se>java ISdel
Enter the value of n and g:
23 50
Enter the value of x for the first person:
4
Enter the value of y for the second person:
8
Key for first person is 6
Key for second person is 6
D:\J2se>
```


Experiment No – 7

Aim: Make a study of anyone simulation tool based on parameters of information security.

Case Study of Simulator:

Simulation is the imitation of the operation of a real-world process or system over time. The act of simulating something first requires that a model be developed; this model represents the key characteristics or behaviors/functions of the selected physical or abstract system or process. The model represents the system itself, whereas the simulation represents the operation of the system over time. Simulation is used in many contexts, such as simulation of technology for performance optimization, safety engineering, testing, training, education, and video games.

Nessi

With NeSSi, we aim to provide a network simulation tool specifically tailored to meet the needs of security experts and network administrators. This target group will be able to test and evaluate the performance of commonly available security solutions as well as new research approaches. As a distinguishing feature to the previous tools described above, NeSSi provides extensive support of complex application-level scenarios on top of a faithful simulation of the TCP/IP protocol stack. Simulated networks are modeled to reflect real-world network topologies by supporting subnet-layer modeling and different node types with varying capabilities (Core and Access subnets, wireless networks etc.). In particular, NeSSi follows a strict object-oriented design pattern and fosters the integration of third-party applications via a standardized socket interface. Furthermore, it provides a comprehensive Detection API for the integration and evaluation of predefined as well as external detection units. In particular, special common attack scenarios are supported and can be simulated, for example worm-spread scenarios, botnet-based DDoS attacks and many more. In addition, customized profiles expressing the node behavior can be applied within the simulation.

The application layer simulation capabilities in NeSSi are provided by distributed software agents, introducing another layer of complexity. In order to maintain scalability, a parallel execution model is used in conjunction with a discrete event model. In this context, the agent platforms are running on multiple parallel machines and connect independently to a central database module. A graphical user front-end allows for real-time inspection and configuration of scenarios

Nessi2

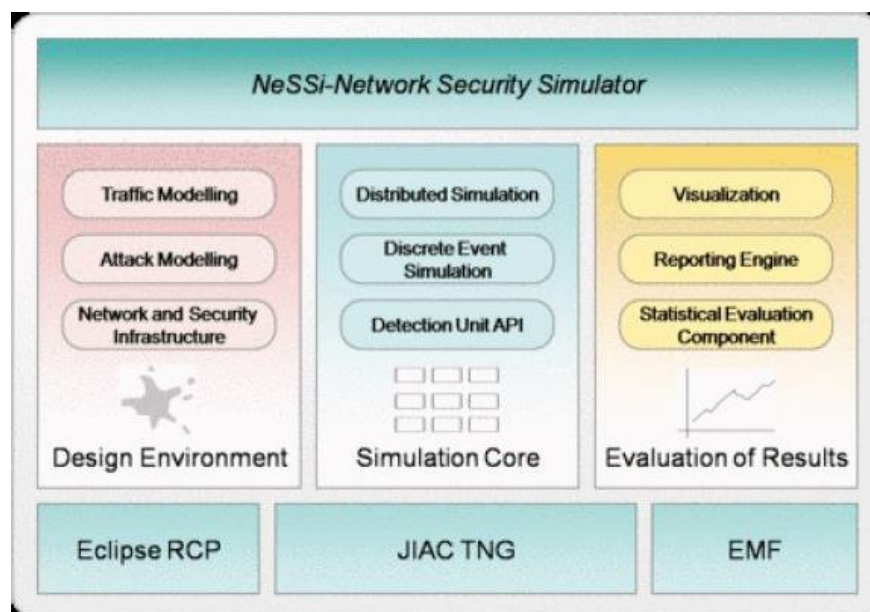
NeSSi2 aims to fill the gap by offering simulation and evaluation features specifically tailored to meet the needs of security experts and network administrators. This target group will be able to test and evaluate the performance of commonly available security solutions as well as new research approaches. As a distinguishing feature to the previous tools described above, *NeSSi2* provides extensive support of complex application-level scenarios on top of a faithful simulation of the TCP/IP protocol stack. Simulated networks are modeled to reflect real-world network topologies by supporting subnet-layer modeling and different node types with varying capabilities (Core and Access subnets, wireless networks etc.). In particular, *NeSSi2* adheres to a modular design pattern and fosters the integration of third-party applications via a standardized plugin interface.

Furthermore, it provides a comprehensive Detection API for the integration and evaluation of simulated as well as external detection units. In particular, special common attack scenarios are supported and can be simulated, worm-spread scenarios and botnet-based DDoS attacks are only two of the example scenarios supported by *NeSSi2*. In addition, customized profiles expressing the node behavior can be applied within the simulation.

The application layer simulation capabilities in *NeSSi2* are provided by distributed software agents, introducing another layer of abstraction. In order to maintain scalability, a parallel execution model is used in conjunction with a discrete event model. In this context, the agent platforms are running on multiple parallel machines and connect independently to a database server from which simulation results can be retrieved in an asynchronous and concurrent fashion. The graphical user interface allows for real-time inspection and configuration of scenarios.

NeSSi2 is designed to extend conventional network simulation tool features by supporting detailed examination and testing opportunities of security-related network algorithms, detection units and frameworks. The main focus of *NeSSi2* is to provide a realistic packet-level simulation environment as a testbed for the development of new detection units as well as existing ones.

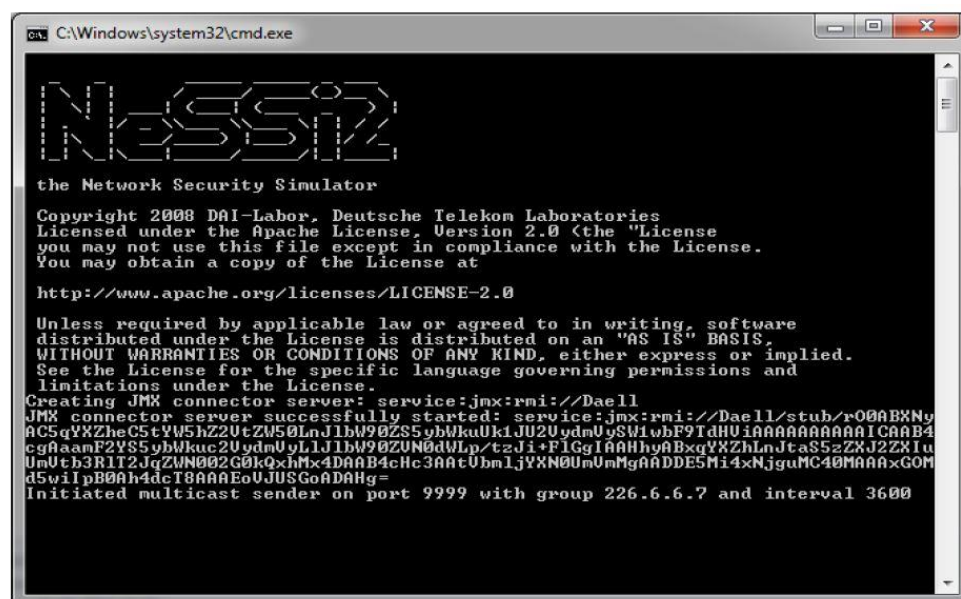
1. **Traffic Generation:** Network traffic in the form of IP packets, complete with header and body, can be generated by different means. Implementing the TCP/IP protocol stack, *NeSSi2* features an application layer module which is based on standard Java socket implementations. *NeSSi2* incorporates several application level protocols (HTTP, SMTP, etc.) and supports static and dynamic routing protocols which can be selected by the user.
2. **Protocol Stack:** The TCP/IP reference model is the de-facto standard for Internet communication. Due to its importance, *NeSSi2* also offers an implementation for it.
3. **Security Feature:** The distinguishing feature of *NeSSi2* is the focus on network security framework and algorithm evaluation.



The simulation setup in *NeSSi2* is not only comprised of network creation and attachment of traffic profiles, but additionally security related settings can be configured.

When a security framework composed of several detection units is to be tested, profiles can also be used in *NeSSi2* to simulate attacker behavior and attack patterns. Accordingly, *NeSSi2* provides out-of-the box support for various attack scenarios such as bot networks initiating DDoS attacks. Here, infected end device nodes, “zombies”, are controlled by the bot net commander via the Internet Relay Chat application. The commander is capable of initiating different kinds of DDoS attacks like the SYN Flooding or UDP Storm. To this end, the attacker connects to an IRC communication server and sends attack commands to a chat channel all the bots are listening to. As a result, the bots execute the desired attack.

The actual simulation is performed on a machine with hardware dedicated solely to this purpose, the *simulation backend*. At the Berlin Institute of Technology for example, the *NeSSi2* simulation backend runs on a Sun XFire 4600 blade server (8 blades, 8 cores per blade). Once a simulation is submitted for execution, the simulation backend parses the desired simulation parameters (which event types to log, how many runs to execute etc.), creates a corresponding simulation environment, sets up the database connection and schedules the simulation to run as soon as the necessary processing resources are available.



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

NeSSi2

the Network Security Simulator

Copyright 2008 DAI-Labor, Deutsche Telekom Laboratories
Licensed under the Apache License, Version 2.0 (the "License"
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
Creating JMX connector server: service:jmx:rmi:///Dael1
JMX connector server successfully started: service:jmx:rmi:///Dael1/stub/r00ABXNy
AC5qYXZheC5tYW5hZ2U0LnJlbW90ZS5ybWkuUk1JU2UydmUySW1wbF9TdHUiaAAAAAIAAB4
cgAaanF2YS5ybWkuc2UydmUyLlJlbW90ZUN0dWlp/tzJi+F1GgIAAHhyABxqYXZlLnJtaS5zZXJ2ZXIu
Um0tb3R1T2JqZWNo02G0kQxhMx4DAAB4cHc3AAU0bm1jYXN0Um0mGAADDE5Mi4xNjguMC40MAAAxGOM
d5wilpB0Ah4dcT8AAAEoUJUSGoADAHg=
Initiated multicast sender on port 9999 with group 226.6.6.7 and interval 3600
```

Experiment No – 8

Aim: Implement VPN through Packet-Tracer or any other network simulation tool.

Technology Used: Cisco Packet Tracer

Procedure/Algorithm

1. Open Cisco Packet Tracer.
2. Draw the design of logical network topology using combination of routers, switches, pc devices etc.
3. **IP address configuration for PC device:** For every PC in the logical network, configure the IP address. It is done by clicking the PC icon.
4. Open the settings for IP configuration and set any IP address along with the default gateway. (Repeat this step for every PC device).
5. **Router Configuration:** For every router, configure the router settings in CLI (IOS Command Line Interface).
6. Write the following commands
Router>en
Router#conf t
Router (config) # int fa0/0
Router (config-if) # ip add 192.168.10.1 255.255.255.0
Router (config-if) #no shut
Router (config) #
%LINK-5-CHANGED: Interface FastEthernet0/0, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface
FastEthernet0/0, changed state to up
Router (config) # int fa0/1
Router (config-if) # ip add 10.0.0.1 255.0.0.0
Router (config-if) #no shut
7. After the configuration of connections, connection has been set. Now, a protocol is followed to exchange the information between each other. We have used RIP (Router Information Protocol)
8. Execute the following commands in CLI (IOS Command Line Interface) on every router .
Router (config-if) #ex
Router (config) #router rip
Router (config-router) # network 192.168.10.0
Router (config-router) # network 10.0.0.0
Router (config-router) # ex
Router (config) #
9. **Creation of VPN:** Execute following commands
Router>en
Router #conf t

```
Router (config) # crypto isakmp policy 10
Router (config-isakmp) #authentication pre-share
Router (config-isakmp) # hash sha
Router (config-isakmp) # encryption aes 256
Router (config-isakmp) # group 2
Router (config-isakmp) # lifetime 86400
Router (config-isakmp) # exit
Router (config) # crypto isakmp key toor address 10.0.0.1
Router (config) # crypto ipsec transform-set TSET esp-aesesp-sha-hmac
Router (config) # access-list 101 permit ip 192.168.20.0 0.0.0.255 192.168.10.0 0.0.0.255
Router (config) # crypto map CMAP 10 ipsec-isakmp
Router (config-crypto-map) # set peer 10.0.0.1
Router (config-crypto-map) # match address 101
Router (config-crypto-map) # set transform-set TSET
Router (config-crypto-map) # ex
Router (config) # int fa0/1
Router (config-if) # crypto map CMAP
Router (config-if) # do wr
Router (config-if) #
```

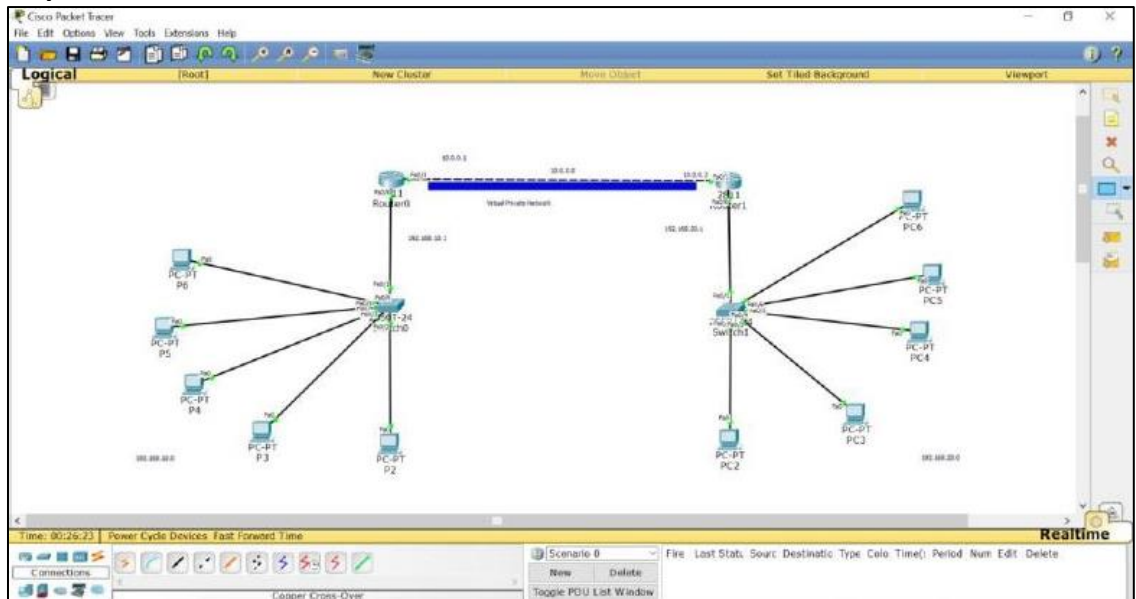
10. **Verification of VPN**

Execute following commands:

```
Router (config-if) # ex
Router (config) # ex
Router#
```

```
Router#show crypto isakmpsa
```

Output:



Router0

Physical Config CLI

IOS Command Line Interface

```
[OK]
Router(config-if)#ex
Router(config)#ex
Router#
%SYS-5-CONFIG_I: Configured from console by console

Router#show crypto isakmp sa
IPv4 Crypto ISAKMP SA
dst          src          state          conn-id slot status
10.0.0.2     10.0.0.1     QM_IDLE       1064      0 ACTIVE

IPv6 Crypto ISAKMP SA

Router#
```

Copy Paste