**INFORMATION SECURITY LAB**

**(ETCS – 451)**

**Faculty name:** Mr. Farzil Kidwai **Student name:** Ayush Pandey

**Roll No.:** 45014802718

**Semester:** 7th Semester

**Group:** 7-C-8



Maharaja Agrasen Institute of Technology

PSP Area, Sector – 22, Rohini, New Delhi – 110085



**MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY**

**VISION**

To nurture young minds in a learning environment of high academic value and imbibe spiritual and ethical values with technological and management competence.

**MISSION**

**The Institute shall endeavor to incorporate the following basic missions in the teaching methodology:**

**Engineering Hardware – Software Symbiosis**

Practical exercises in all Engineering and Management disciplines shall be carried out by Hardware equipment as well as the related software enabling deeper understanding of basic concepts and encouraging inquisitive nature.

**Life – Long Learning**

The Institute strives to match technological advancements and encourage students to keep updating their knowledge for enhancing their skills and inculcating their habit of continuous learning.

**Liberalization and Globalization**

The Institute endeavors to enhance technical and management skills of students so that they are intellectually capable and competent professionals with Industrial Aptitude to face the challenges of globalization.

**Diversification**

The Engineering, Technology and Management disciplines have diverse fields of studies with different attributes. The aim is to create a synergy of the above attributes by encouraging analytical thinking.

**Entrepreneurship**

The Institute strives to develop potential Engineers and Managers by enhancing their skills and research capabilities so that they become successful entrepreneurs and responsible citizens.



**MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY**

**COMPUTER SCIENCE AND ENGINEERING DEPARTMENT**

**VISION**

To produce “Critical Thinkers of Innovative Technology”.

**MISSION**

To foster an open, multidisciplinary and highly collaborative research environment for producing world-class engineers capable of providing innovative solutions to real-life problems and fulfil societal needs.

**MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**Outcome Based Learning**

**Course Outcomes (Revision)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Subject:** | **INFORMATION SECURITY** | **Max Marks Internal** | **60** |
| **Subject Code** | ETCS 451 |  |  |
| **Total Credit** | 1 | Evaluation Scheme | |
| Contact Hours |  | Theory/Lab | Lab |
|  |  |  |

**Course Objectives:**

The aim of this subject is to create security and privacy awareness among students. It covers concepts in information Systems-Threats and attacks, Basic principles of information security, Networks and E-security- Internet and WWW.Review of Internet Protocols, Physical security and bio metric as security and Network cryptography.

|  |  |
| --- | --- |
| **S. No.** | **Course Outcomes** |
| C401.1 | To understand the basics of Information security and common threats faced today. |
| C401.2 | To know about basics principle of information security and various security challenges and authentication issues in Mobile devices. |
| C401.3 | To understand advanced security issues of Internet including various protocols for security services and security acts to handle various forensics issues. |
| C401.4 | To apply physical and bio-metric security on various economic and social frameworks |
| C401.5 | To master fundamentals of private and public cryptography. |
| C401.6 | To have knowledge of various security concepts and issues for network protection in VPN. |

**INFORMATION SECURITY LAB**

**Paper Code: ETCS-451 L T/P C**

**Paper: INFORMATION SECURITY Lab 0 2 1**

**List of Experiments:**

1. **Make an experiment to implement WEP/WPA2 PSK, 802.1x EAP security protocol.**
2. **Implement firewall through App to login into bank-site; to implement E-commerce, debit card transaction through payment gateway**
3. **Implement bio-metric system to have physical security through different access control permissions.**
4. **Implement RSA algorithm.**
5. **Implement DES algorithm**
6. **Implement Diffie-Hellman algorithm**
7. **Make a study of anyone simulation tool based on parameters of information security**
8. **Implement VPN through Packet-Tracer or any other network simulator tool.**
9. **PGP tool usage using Command Line version of PGP.**

**PRACTICAL RECORD**

**PAPER CODE : ETCS-451**

**Name of the student : Ayush Pandey**

**University Roll No. : 45014802718**

**Branch : CSE**

**Group : 7C-8**

**PRACTICAL DETAILS**

1. Experiments according to IS lab syllabus prescribed by GGSIPU

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Exp. No.** | **Experiment Name** | **Marks** | | | | | **Total Marks** | **Signature with Date** |
| **R1** | **R2** | **R3** | **R4** | **R5** |  |  |
| 1. | Make an experiment to implement WEP/WPA2 PSK, 802.1x EAP security protocol. |  |  |  |  |  |  |  |
| 2. | Implement firewall through App to login into bank-site; to implement E-commerce, debit card transaction through payment gateway. |  |  |  |  |  |  |  |
| 3. | Implement bio-metric system to have physical security through different access control permissions. |  |  |  |  |  |  |  |
| 4. | To perform RSA Algorithm. |  |  |  |  |  |  |  |
| 5. | To perform DES Algorithm. |  |  |  |  |  |  |  |
| 6. | To perform Diffie-Hellman algorithm. |  |  |  |  |  |  |  |
| 7. | Make a study of anyone simulation tool based on parameters of information security. |  |  |  |  |  |  |  |
| 8. | Implement VPN through Packet-Tracer or any other network simulator tool. |  |  |  |  |  |  |  |
| 9. | PGP tool usage using Command Line version of PGP. |  |  |  |  |  |  |  |

**Date: 23/09/2021**

**Experiment-1**

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

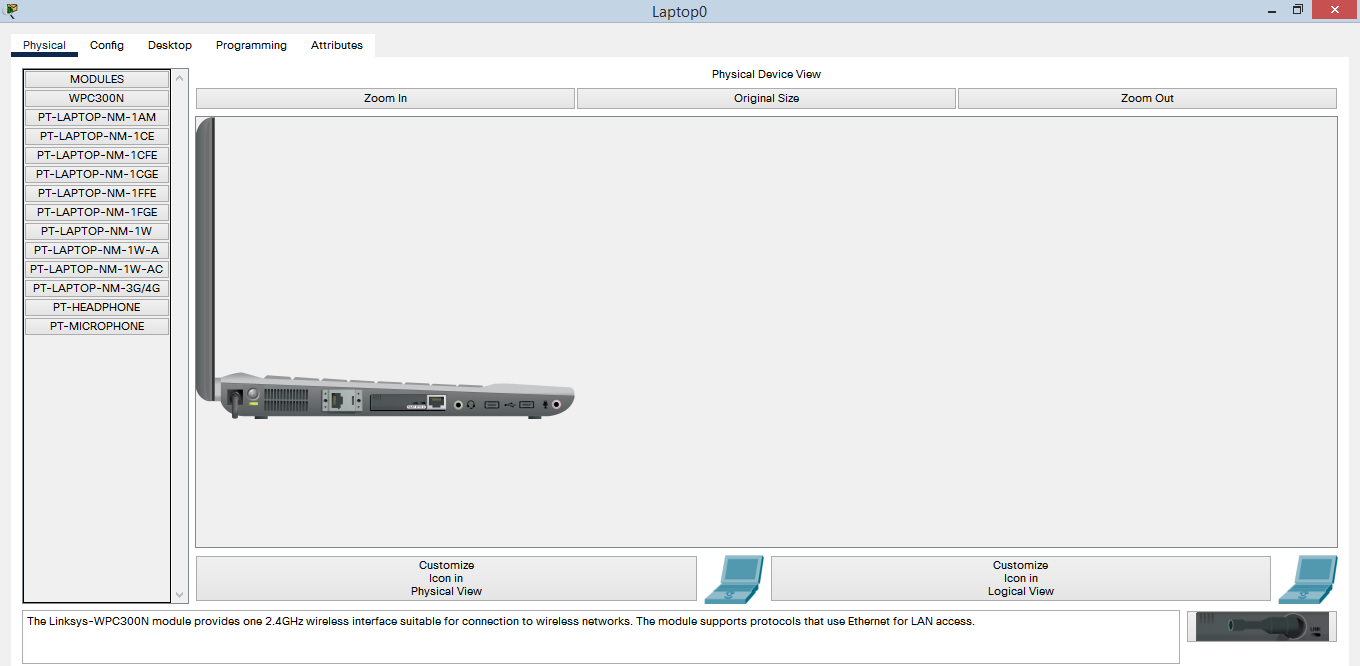
**Technology Requirement:** System Loaded with Cisco Packet Tracer

**Theory:**

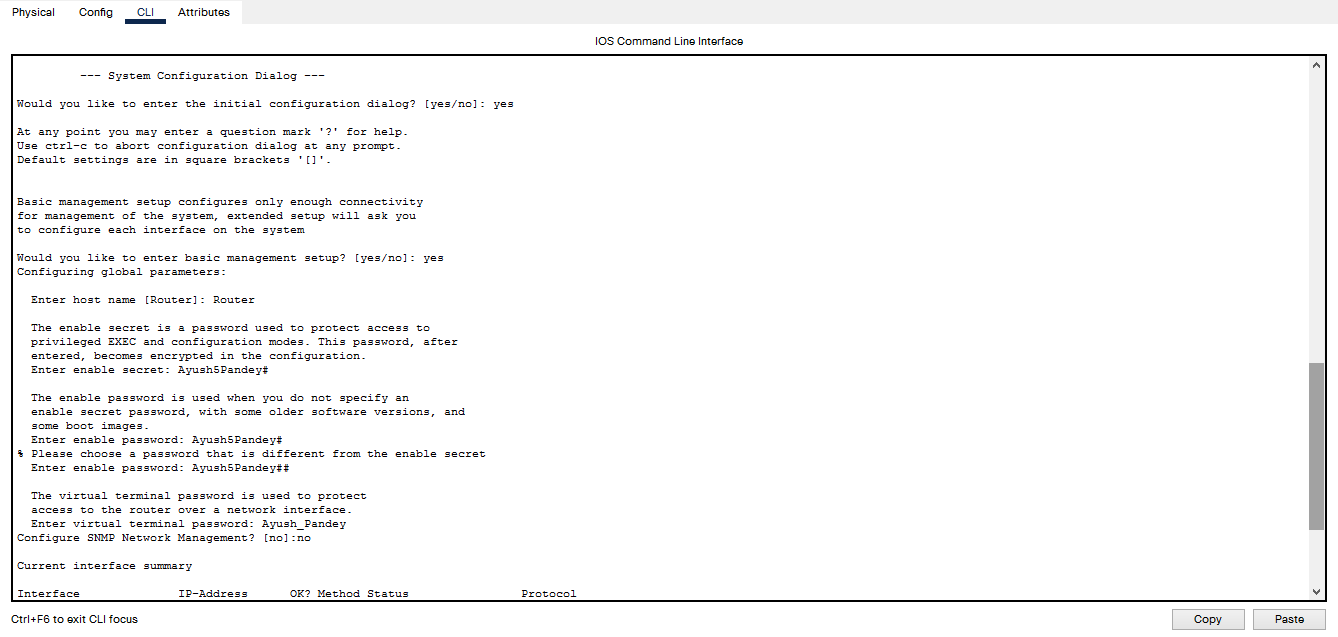
Cisco Packet Tracer is an innovative network simulation and visualization tool. This free software helps you to practice your network configuration and troubleshooting skills via your desktop computer or an Android or iOS based mobile device. Packet Tracer is available for both the Linux and Windows desktop environments.

With Packet Tracer you can choose to build a network from scratch, use a pre-built sample network, or complete classroom lab assignments. Packet Tracer allows you to easily explore how data traverses your network. Packet Tracer provides an easy way to design and build networks of varying sizes without expensive lab equipment. While this software is not a replacement for practicing on physical routers, switches, firewalls, and such.

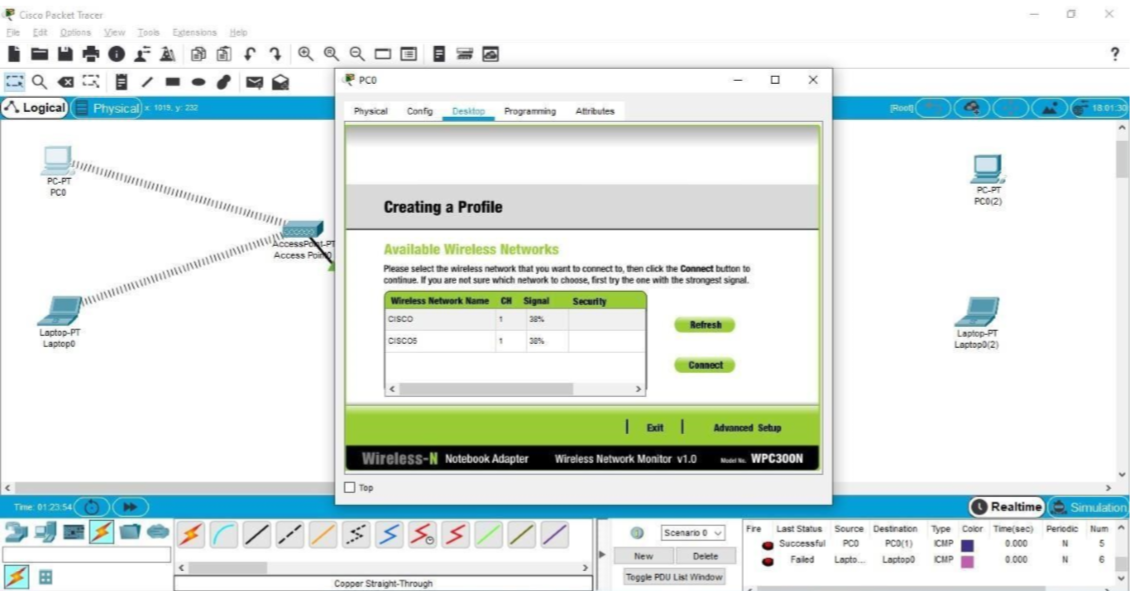
**Physical Device View**

****

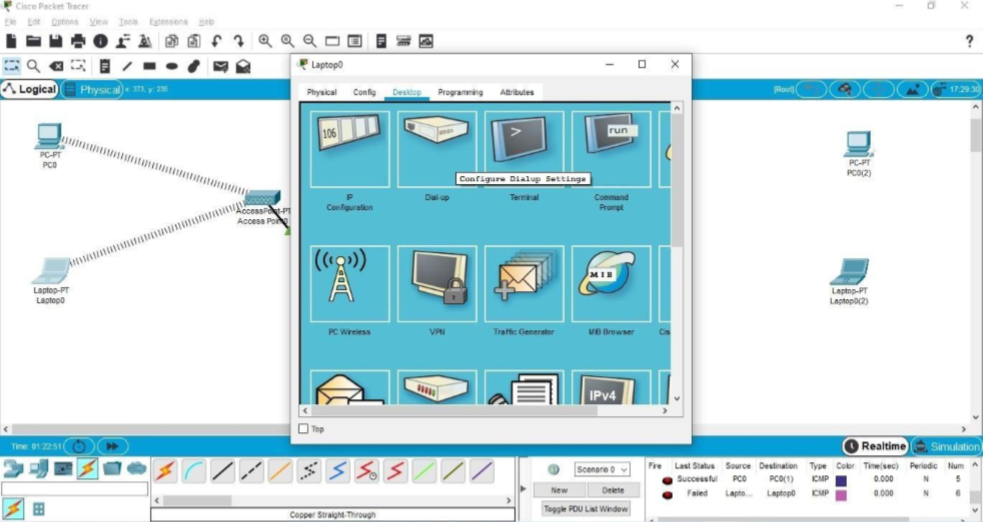
**IOS Command Line Interface**

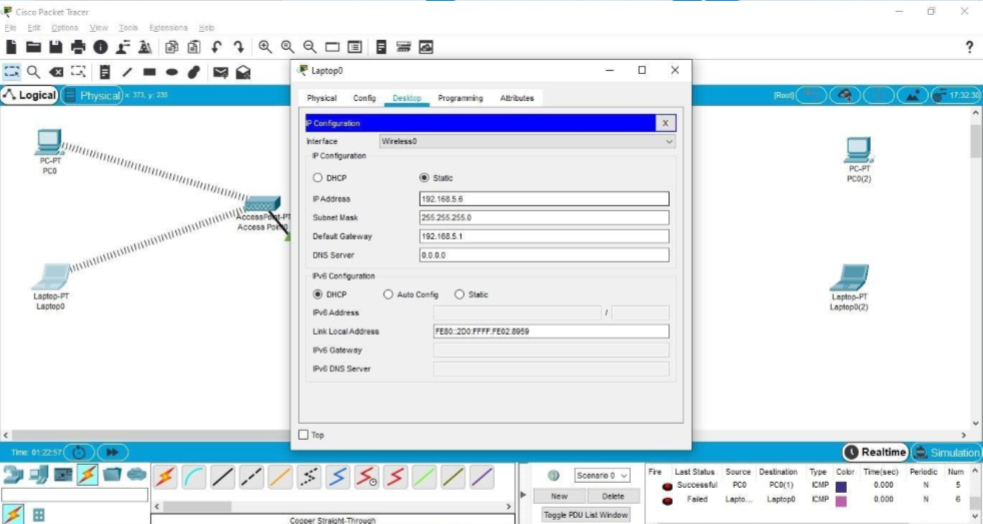
****

**Creating the Profile**

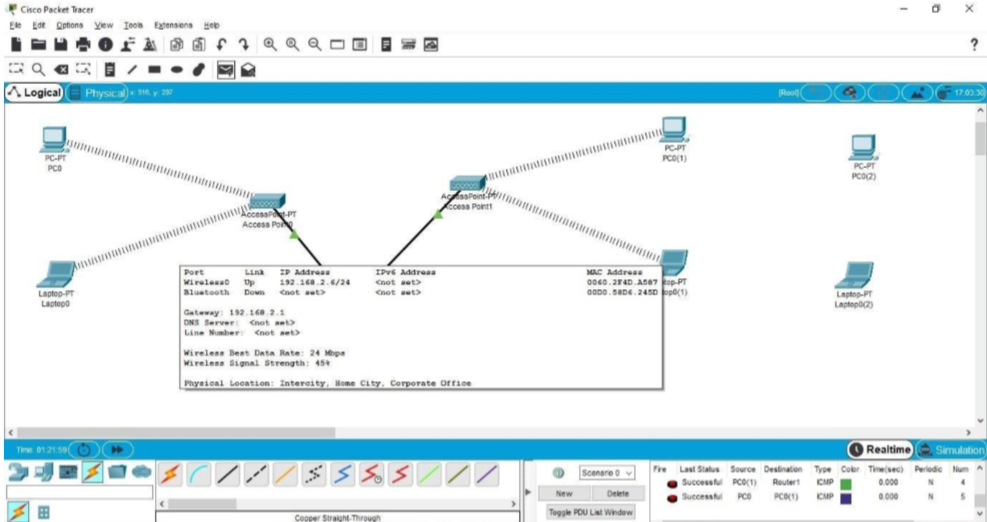
****

**Making Configurations**

****

****

**Setting Up the WEP Network**

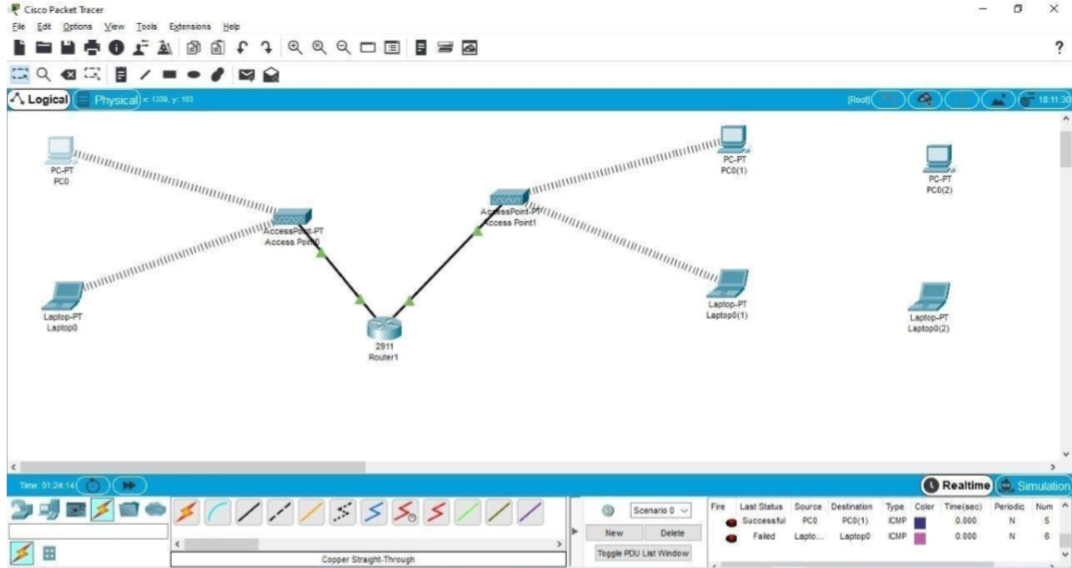
****

**WEP (Wired Equivalent Privacy)**

* It is a Security algorithm for wireless network.
* It is designed to provide Wireless Local Area Network (WLAN) with a level of security and privacy comparable to what is usually expected of a wired LAN.
* To provide data confidentiality comparable to that of a traditional wired network.
* A 64-bit WEP key is usually entered as a string of 10 hexadecimal (base 16) characters (0–9 and A–F). Each character represents 4 bits, 10 digits of 4 bits each gives 40 bits; adding the 24-bit IV produces the complete 64bit WEP key (4 bits × 10 + 24 bits IV = 64 bits of WEP key).

WEP is a security algorithm for IEEE 802.11 wireless networks, ratified in 1997, its intention was to provide data confidentiality comparable to that of a traditional wired network. WEP, recognizable by its key of 10 or 26 hexadecimal digits (40 or 104 bits), was at one time widely in use and was often the first security choice presented to users by router configuration tools.

**Setting Up the WPA2 Network**

****

WPA stands for Wi-Fi Protected Access is a security standard for users of computing devices equipped with wireless internet connections. WPA was developed by the Wi-Fi Alliance to provide more sophisticated data encryption and better user authentication than Wired Equivalent Privacy (WEP), the original Wi-Fi security standard. The new standard, which was ratified by the IEEE in 2004 as 802.11i, was designed to be backward-compatible with WEP to encourage quick, easy adoption. Network security professionals were able to support WPA on many WEP-based devices with a simple firmware update.

**Result:** The working of WEP and WPA2 was successfully implemented and understood.

**Viva Voce**

1. **Do WIFI Certificates Replace Wireless Security Protocols Like Wpa2?**

* No. WIFI certificates are only used to encrypt data during the signup process. They are not used to encrypt data that is passed while an end-user is browsing the Internet.

1. **If My Wireless Network Doesn’t Have a lot of Traffic, Is It Okay to Use WEP Because the Ivs Required to Crack the WEP Key Won’t Be Generated?**

* No. Automated tools are available that allow attackers to capture an ARP packet and reinject it to the access point very rapidly. This generates a significant amount of traffic and allows the attacker to capture enough unique initialization vectors to quickly crack the key.

1. **What is the difference between Active and Passive WLAN detection?**

* Active WLAN detection requires that the SSID be broadcast in the beacon frame. Passive WLAN detection listens to all traffic in range of the device and determines what WLANs are in range.

1. **How many types of extensible Authentication Protocols (eaps) are supported by Wpa/wpa2 and What Are They?**

* There are six fully supported EAP types for WPA/WPA2: EAP-TLS; EAP-TLS/MSCHAPv2; PEAPv0/EAP-MSCHAPv2; PEAPv1/EAP-GTC; EAP-SIM; and EAP-LEAP.

1. **What is the primary difference between 802.11g and 802.11a?**

* 802.11g operates in the 2.4 GHz frequency range, as do 802.11b and 802.11i, whereas 802.11a operates in the 5 GHz frequency range.

1. **What is the difference between the hostap drivers and The Wlan-ng Drivers for linux?**

* Both of these drivers work with a variety of cards; however, only the HostAP drivers allow you to place your card in monitor mode.

1. **Who determines the wireless standards?**

* The IEEE develops and determines the wireless standards (802.11a, b, g, and so on). The WiFi Alliance, the group that owns the WiFi trademark, then certifies the interoperability of these devices.

**Date: 30/09/2021**

**Experiment-2**

**Aim:** Implement a firewall on Cisco Packet Tracer

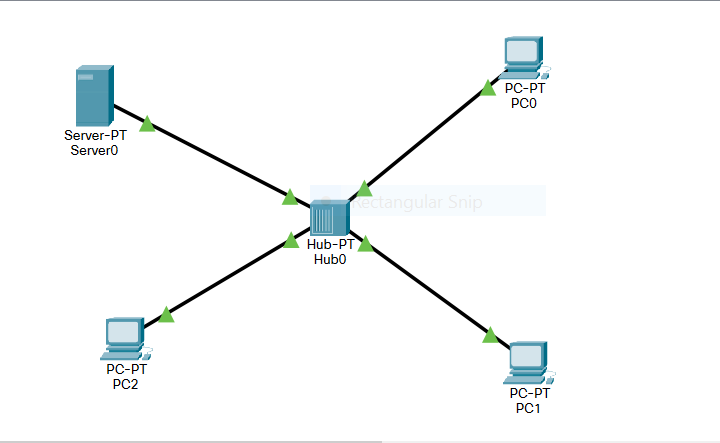
**Technology Requirement:** System Loaded with Cisco Packet Tracer

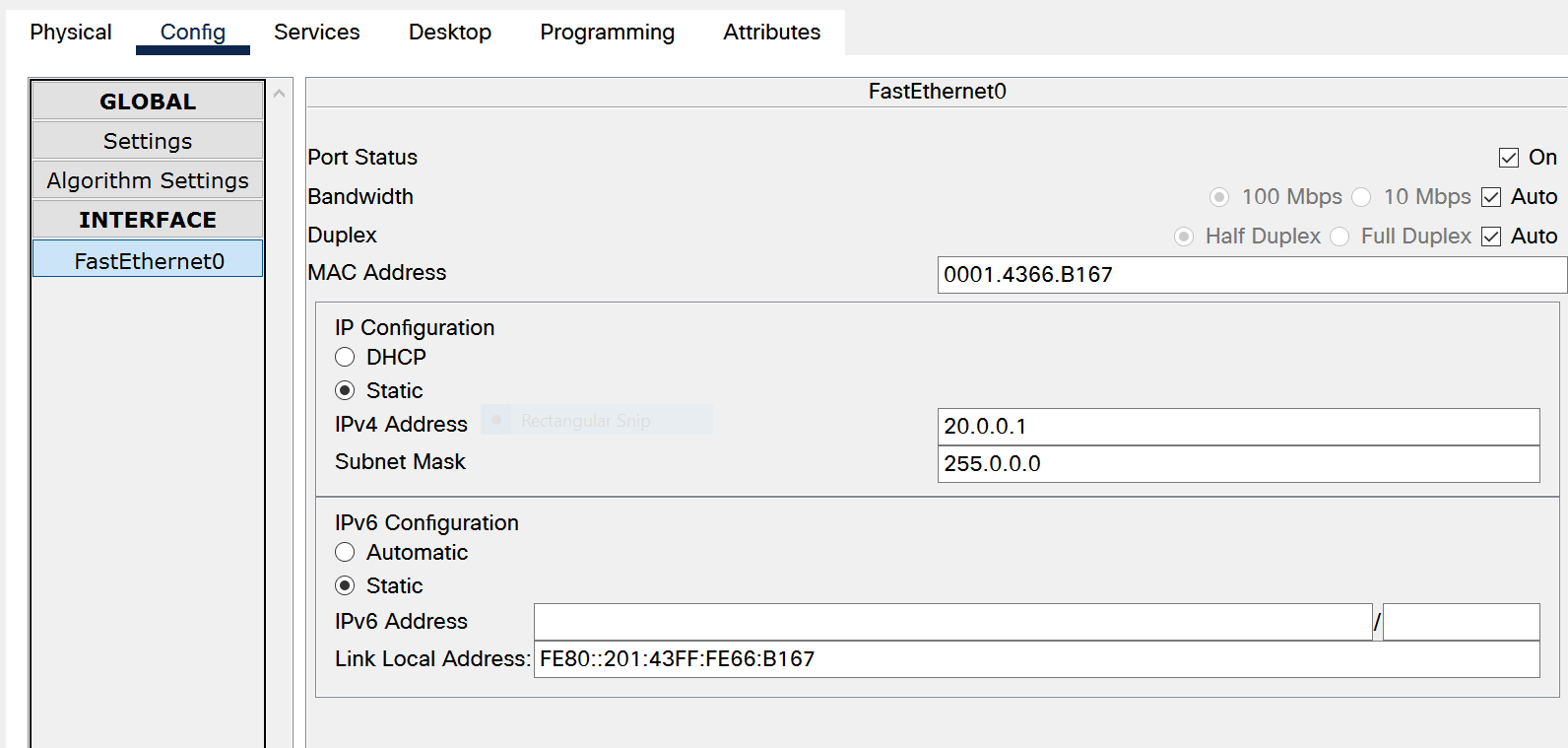
**Theory:**

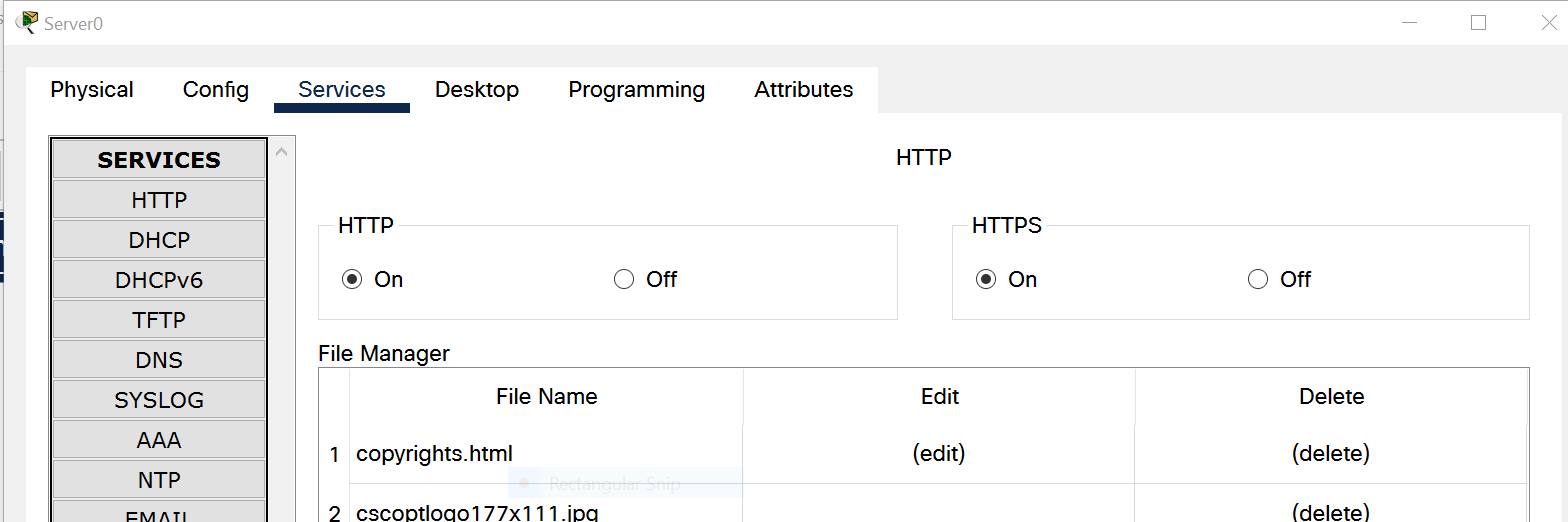
A firewall is a network security device that monitors incoming and outgoing network traffic and permits or blocks data packets based on a set of security rules. Its purpose is to establish a barrier between your internal network and incoming traffic from external sources (such as the internet) in order to block malicious traffic like viruses and hackers.

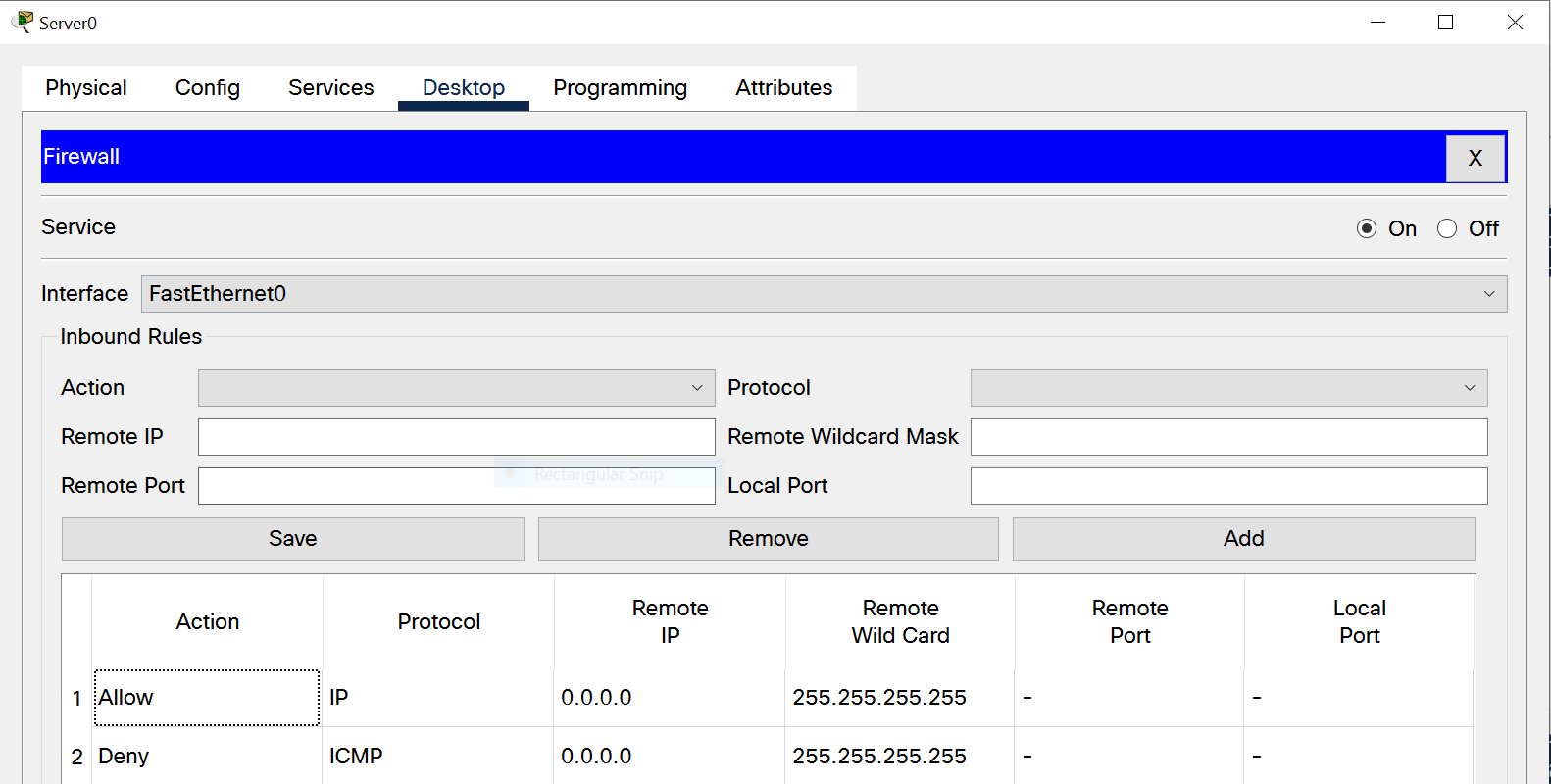
Firewalls carefully analyze incoming traffic based on pre-established rules and filter traffic coming from unsecured or suspicious sources to prevent attacks. Firewalls guard traffic at a computer’s entry point, called ports, which is where information is exchanged with external devices. For example, “Source address 172.18.1.1 is allowed to reach destination 172.18.2.1 over port 22."

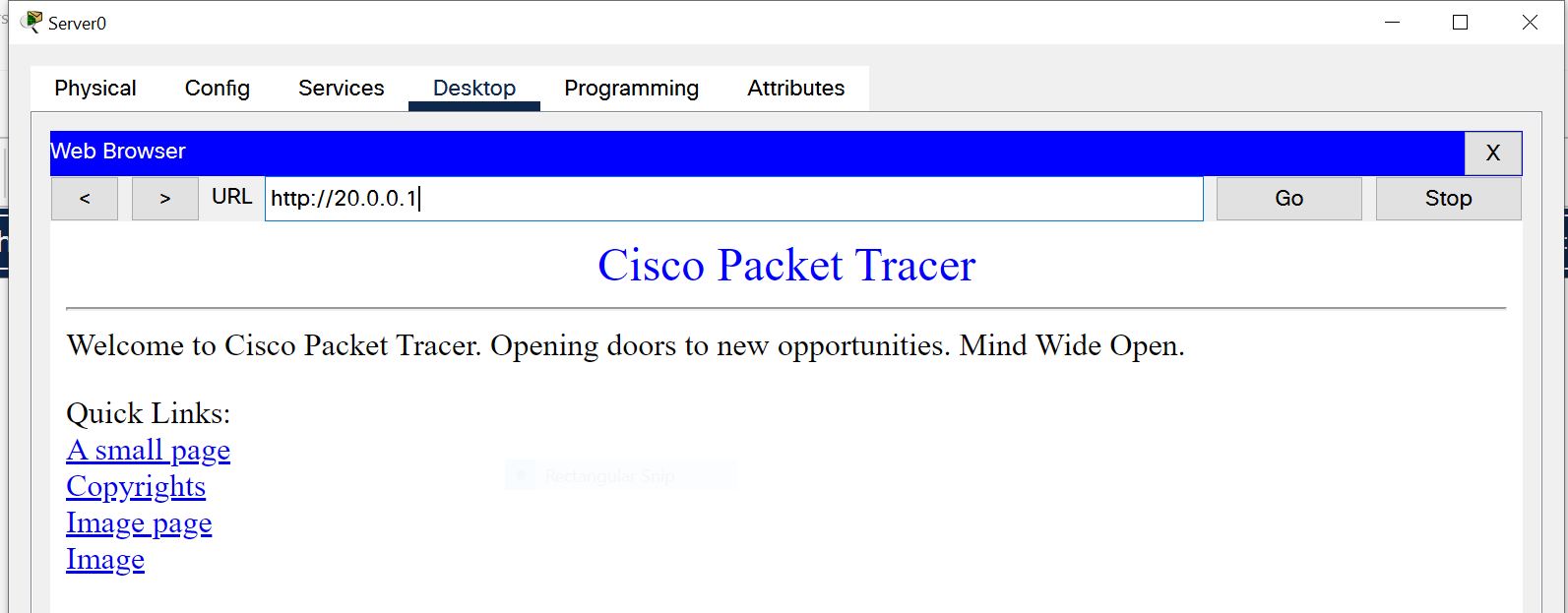
**Screenshots:**

****

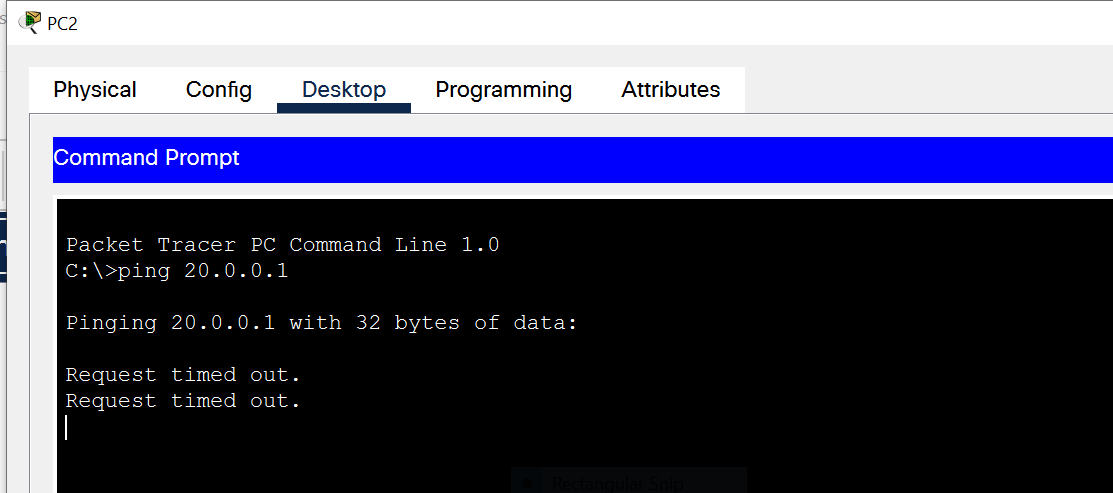








**Positive Response in Browser**

****

Request Timed out on PC2.

**Result:** Working of a Firewall is Understood and Implemented.

**Viva Voce**

1. **What is a network firewall?**

* A firewall is a system or group of systems that enforces an access control policy between two networks. The actual means by which this is accomplished varies widely, but in principle, the firewall can be thought of as a pair of mechanisms: one which exists to block traffic, and the other permitting traffic.

1. **What is synchronization and why is it important?**

* With respect to multithreading, synchronization is the capability to control the access of multiple threads to shared resources. Without synchronization, it is possible for one thread to modify a shared object while another thread is in the process of using or updating that object's value. This often leads to significant errors.

1. **What are the critical resources in a firewall?**

* 1. Service Critical Resource

2. Email

3. Disk I/O

4. Netnews Disk I/O

5. Web Host

1. **What are some common attacks, and how can I protect my system against them?**

* Each site is a little different from every other in terms of what attacks are likely to be used against it. Some recurring themes do arise, though.

**Date: 07/10/2021**

**Experiment-3**

**Aim:** Classical Cryptography: Using Classical Ciphers with Pycipher.

**Technology Requirement:** System Loaded with Cisco Packet Tracer

**Theory:** Pycipher Python Library

**ADFGX Cipher:** This cipher uses a keysquare as part of its key. The ADFGX Cipher has a key consisting of a 5x5 key square and a word e.g. ‘GERMAN’. The key square consists of the letters A-Z with J omitted (25 characters total).

Parameters:

* Key: The keysquare, as a 25 character string
* Keyword: The keyword, any word or phrase will do

**ADFGVX Cipher:** This cipher uses a keysquare as part of its key. The ADFGVX Cipher has a key consisting of a 6x6 key square and a word. The key square consists of the letters A-Z and the numbers 0-9 (36 characters total).

Parameters:

* key: The keysquare, as a 36 character string
* keyword: The keyword, any word or phrase will do

**Affine Cipher:** The Affine Cipher has two components to the key, numbers a and b. This cipher encrypts a letter according to the following equation:

c = (a\*p + b)%26

where c is the ciphertext letter, p the plaintext letter. b is an integer 0-25, a is an integer that has an inverse (mod 26). Allowable values for a are: 1,3,5,7,9,11,15,17,19,21,23,25

Parameters:

* a: The multiplicative part of the key. Allowable values are: 1,3,5,7,9,11,15,17,19,21,23,25
* b: The additive part of the key. Allowable values are integers 0-25

**Caesar Cipher:** The Caesar Cipher has a key consisting of an integer 1-25. This cipher encrypts a letter according to the following equation:

c = (p + key)%26

where c is the ciphertext letter, p the plaintext letter.

Parameters:

* Key: The additive key. Allowable values are integers 0-25.

**Playfair Cipher:** This cipher uses a keysquare as part of its key. The Playfair Cipher enciphers pairs of characters, the key consists of a keysquare 25 characters in length.

Parameters:

* Key: The keysquare, as a 25 character string.

**Simple Substitution Cipher:** The Simple Substitution Cipher has a key consisting of the letters A-Z jumbled up. e.g. ‘AJPCZWRLFBDKOTYUQGENHXMIVS’ This cipher encrypts a letter according to the following equation:

plaintext = ABCDEFGHIJKLMNOPQRSTUVWXYZ

ciphertext = AJPCZWRLFBDKOTYUQGENHXMIVS

To convert a plaintext letter into ciphertext, read along the plaintext row until the desired letter is found, then substitute it with the letter below it.

Parameters:

* Key: The key, a permutation of the 26 characters of the alphabet

**Vigenere Cipher:** The Vigenere Cipher has a key consisting of a word. His cipher encrypts a letter according to the Vigenere tableau.

Parameters:

* Key: The keyword, any word or phrase will do. Must consist of alphabetical characters only, no punctuation of numbers

**OUTPUT:**

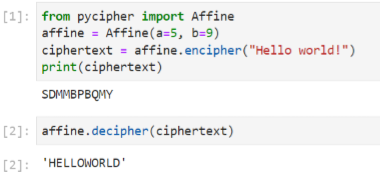
* **ADFGX Cipher:**

****

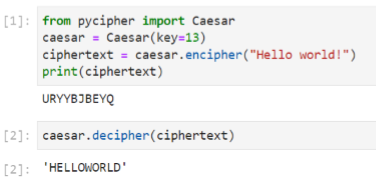
* **ADFGVX Cipher:**

****

* **Affine Cipher:**

****

* **Caesar Cipher:**

****

* **Playfair Cipher:**

****

* **Simple Substitution Cipher:**

****

* **Vigenere Cipher:**

****

**Result:** Classical cryptography techniques are implemented and understood.

**Date: 14/10/2021**

**Experiment-4**

**Aim:** To perform RSA Algorithm.

**Theory:**

The RSA algorithm is an asymmetric cryptography algorithm; this means that it uses a public key and a private key (i.e. two different, mathematically linked keys). As their names suggest, a public key is shared publicly, while a private key is secret and must not be shared with anyone. It is also one of the oldest. The acronym RSA comes from the surnames of Ron Rivest, Adi Shamir and Leonard Adleman, who publicly described the algorithm in 1977.

**Why RSA Encryption is secure?**

The idea of making one of your own encryption algorithms public on the internet seems very strange at first. However, this is actually one of the most important steps in RSA encryption.

If Person C intercepts your message to Person B, they already know the encryption key (exponent e, modulus n). However, what he/she doesn't have is the decryption exponent d. Since you encrypted your message with Person B's encryption key, only Person B has the decryption key (exponent d, modulus n) to decrypt it. Person C is only missing one piece of information, exponent d, which turns out to be the hardest piece of information to find.

Person C also knows that

de ≡ 1 (mod ϕ(n)), or de ≡ 1 (mod (p - 1)(q - 1)).

Since he/she knows that n = pq, the simplest way to find n would be to somehow factor n into the exact primes used by Person B in the algorithm. From there, he/she could simply calculate the congruence to find d.

With larger (which are more secure) primes, this turns out to be nearly impossible to do.

If p = 7717 and q = 7919, n would be 61110923. If we let e = 5, then all Person

C knows is

e = 5, n = 61110923.

Clearly, it would take very long to factor n, but imagine what would happen if

p = 982451653, q = 961748941.

Then n would be 944871836856449473.

Now factoring n is basically impossible to do by hand. However, even this value of n is smaller than most values of n used in RSA Encryption. It took 290 computers over the internet and a supercomputer 4 months to find that

n=10941738641570527421809707322040357612003732945449205990913842131476349984288934784717997257891267332497625752899781833797076537244027146743 531593354333897

Had prime factors

p=102639592829741105772054196573991675900716567808038066803341933521790711307779,

q=106603488380168454820927220360012878679207958575989291522270608237193062808643.

In this case n had only 155 digits. Many values of n have over 200 digits, making the RSA algorithm nearly unbreakable.

**CODE:**

#include<iostream>

#include<math.h>

using namespace std;

int gcd(int a, int b) {

int t;

while(1) {

t= a%b;

if(t==0)

return b;

a = b;

b= t; } }

int main() {

double p = 13;

double q = 11;

double n=p\*q;

double gcdmain;

double z= (p-1)\*(q-1);

double e=7;

while(e<z) {

gcdmain= gcd(e,z);

if(gcdmain==1)

break;

else

e++; }

double d1=1/e;

double d=fmod(d1,z);

double message = 20;

double c = pow(message,e);

double m = pow(c,d);

c=fmod(c,n);

m=fmod(m,n);

cout<<"Message = "<<message;

cout<<"\n"<<"p = "<<p;

cout<<"\n"<<"q = "<<q;

cout<<"\n"<<"n = pq = "<<n;

cout<<"\n"<<"z = "<<z;

cout<<"\n"<<"e = "<<e;

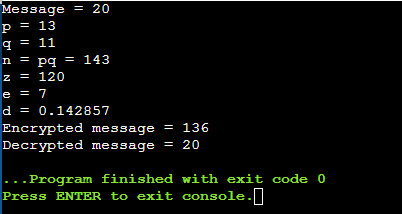
cout<<"\n"<<"d = "<<d;

cout<<"\n"<<"Encrypted message = "<<c;

cout<<"\n"<<"Decrypted message = "<<m;

return 0; }

**OUTPUT:**



**Viva Voce**

1. **How Fast Is RSA?**

* RSA is considerably slow due to the calculation with large numbers. In particular the decryption where d is used in the exponent is slow. There are ways to speed it up by remembering p and q, but it is still slow in comparison to symmetric encryption algorithms. Asymptotic time complexity for an implementation of RSA using elementary algorithms, commonly used in practice, is O(n3) for private key use (signature generation, and decryption) and O(n2) for public-key use (signature verification, and encryption), where the public modulus N has n bits and public exponent e has a fixed size.

1. **What Would It Take to Break RSA?**

* It would take a classical computer around 300 trillion years to break a RSA-2048 bit encryption key.

1. **Are Strong Primes Necessary In RSA?**

* In the literature pertaining to RSA, it has often been suggested that in choosing a key pair, one should use so-called "strong" prime’s p and q to generate the modulus n. Strong primes have certain properties that make the product n hard to factor by specific factoring methods. However, advances in factoring over the last ten years appear to have obviated the advantage of strong primes; the elliptic curve factoring algorithm is one such advance. The new factoring methods have as good a chance of success on strong primes as on "weak" primes. Therefore, choosing traditional "strong" primes alone does not significantly increase security. Choosing large enough primes is what matters.

1. **How large a modulus (key) should be used in RSA?**

* The best size for an RSA modulus depends on one's security needs. The larger the modulus, the greater the security, but also the slower the RSA operations. One should choose a modulus length upon consideration, first, of one's security needs, such as the value of the protected data and how long it needs to be protected, and, second, of how powerful one's potential enemies are. Key sizes are now 768 bits for personal use, 1024 bits for corporate use, and 2048 bits for extremely valuable keys like the key pair of a certifying authority.

1. **How Large Should the Primes Be?**

* As for whether collisions are possible- modern key sizes (depending on your desired security) range from 1024 to 4096, which means the prime numbers range from 512 to 2048 bits. That means that your prime numbers are on the order of 2^512: over 150 digits long.

**Date: 21/10/2021**

**Experiment-5**

**Aim:** To perform DES Algorithm.

**Theory:**

**Feistel Cipher:**

Feistel Cipher model is a structure or a design used to develop many block ciphers such as DES. Feistel cipher may have invertible, non-invertible and self-invertible components in its design. Same encryption as well as decryption algorithm is used. A separate key is used for each round. However same round keys are used for encryption as well as decryption.

Feistel cipher algorithm

• Create a list of all the Plain Text characters.

• Convert the Plain Text to Ascii and then 8-bit binary format.

• Divide the binary Plain Text string into two halves: left half (L1) and right half (R1)

• Generate a random binary key (K1 and K2) of length equal to the half the length of the Plain Text for the two rounds.

• First Round of Encryption a. Generate function f1 using R1 and K1 as follows: f1= xor (R1, K1) b. Now the new left half (L2) and right half (R2) after round 1 are as follows: R2= xor (f1, L1) L2=R1

• Second Round of Encryption a. Generate function f2 using R2 and K2 as follows: f2= xor (R2, K2) b. Now the new left half (L2) and right half (R2) after round 1 are as follows: R3= xor(f2, L2) L3=R2

• Concatenation of R3 to L3 is the Cipher Text

• Same algorithm is used for decryption to retrieve the Plain Text from the Cipher Text.

**Encryption Process**

• The input block to each round is divided into two halves that can be denoted as L and R for the left half and the right half.

• In each round, the right half of the block, R, goes through unchanged. But the left half, L, goes through an operation that depends on R and the encryption key. First, we apply an encrypting function ‘f’ that takes two input − the key K and R. The function produces the output f(R, K). Then, we XOR the output of the mathematical function with L.

• In real implementation of the Feistel Cipher, such as DES, instead of using the whole encryption key during each round, a round-dependent key (a sub-key) is derived from the encryption key. This means that each round uses a different key, although all these sub-keys are related to the original key.

• The permutation step at the end of each round swaps the modified L and unmodified R. Therefore, the L for the next round would be R of the current round. And R for the next round be the output L of the current round.

• Above substitution and permutation steps form a ‘round’. The number of rounds are specified by the algorithm design.

• Once the last round is completed then the two sub blocks, ‘R’ and ‘L’ are concatenated in this order to form the cipher-text block.

**Decryption Process:** The process of decryption in Feistel cipher is almost similar. Instead of starting with a block of plaintext, the cipher-text block is fed into the start of the Feistel structure and then the process thereafter is exactly the same as described in the given illustration. The process is said to be almost similar and not exactly same. In the case of decryption, the only difference is that the sub-keys used in encryption are used in the reverse order. The final swapping of ‘L’ and ‘R’ in last step of the Feistel Cipher is essential. If these are not swapped then the resulting cipher-text could not be decrypted using the same algorithm.

**Data encryption standard (DES)**

Data encryption standard (DES) has been found vulnerable against very powerful attacks and therefore, the popularity of DES has been found slightly on decline. DES is a block cipher, and encrypts data in blocks of size of 64 bit each, means 64 bits of plain text goes as the input to DES, which produces 64 bits of cipher text. The same algorithm and key are used for encryption and decryption, with minor differences. The key length is 56 bits.

We have mention that DES uses a 56 bit key. Actually, the initial key consists of 64 bits. However, before the DES process even starts, every 8th bit of the key is discarded to produce a 56 bit key. That is bit position 8, 16, 24, 32, 40, 48, 56 and 64 are discarded.

Thus, the discarding of every 8th bit of the key produces a 56-bit key from the original 64-bit key.

DES is based on the two fundamental attributes of cryptography: substitution (also called as confusion) and transposition (also called as diffusion). DES consists of 16 steps, each of which is called as a round. Each round performs the steps of substitution and transposition. Let us now discuss the broad-level steps in DES.

1. In the first step, the 64-bit plain text block is handed over to an initial Permutation (IP) function.

2. The initial permutation performed on plain text.

3. Next the initial permutation (IP) produces two halves of the permuted block; says Left Plain Text (LPT) and Right Plain Text (RPT).

4. Now each LPT and RPT to go through 16 rounds of encryption process.

5. In the end, LPT and RPT are re-joined and a Final Permutation (FP) is performed on the combined block

6. The result of this process produces 64-bit cipher text.

**Initial Permutation (IP)** – As we have noted, the Initial permutation (IP) happens only once and it happens before the first round. It suggests how the transposition in IP should proceed, as show in figure. For example, it says that the IP replaces the first bit of the original plain text block with the 58th bit of the original plain text, the second bit with the 50th bit of the original plain text block and so on.

This is nothing but jugglery of bit positions of the original plain text block. The same rule applies for all the other bit positions.

**Step-1: Key transformation –** We have noted initial 64-bit key is transformed into a 56-bit key by discarding every 8th bit of the initial key. Thus, for each a 56-bit key is available. From this 56-bit key, a different 48bit Sub Key is generated during each round using a process called as key transformation. For this the 56-bit key is divided into two halves, each of 28 bits. These halves are circularly shifted left by one or two positions, depending on the round.

For example, if the round number 1, 2, 9 or 16 the shift is done by only position for other rounds, the circular shift is done by two positions. The number of key bits shifted per round is show in figure.

After an appropriate shift, 48 of the 56 bit are selected. For selecting 48 of the 56 bits the table show in figure given below. For instance, after the shift, bit number 14 moves on the first position, bit number 17 moves on the second position and so on. If we observe the table carefully, we will realize that it contains only 48 bit positions. Bit number 18 is discarded (we will not find it in the table), like 7 others, to reduce a 56-bit key to a 48-bit key. Since the key transformation process involves permutation as well as selection of a 48-bit sub set of the original 56-bit key it is called Compression Permutation.

Because of this compression permutation technique, a different subset of key bits is used in each round. That’s make DES not easy to crack.

**Step-2: Expansion Permutation –** Recall that after initial permutation, we had two 32-bit plain text areas called as Left Plain Text(LPT) and Right Plain Text(RPT). During the expansion permutation, the RPT is expanded from 32 bits to 48 bits. Bits are permuted as well hence called as expansion permutation. This happens as the 32 bit RPT is divided into 8 blocks, with each block consisting of 4 bits. Then, each 4 bit block of the previous step is then expanded to a corresponding 6 bit block, i.e., per 4 bit block, 2 more bits are added.

This process results into expansion as well as permutation of the input bit while creating output. Key transformation process compresses the 56-bit key to 48 bits. Then the expansion permutation process expands the 32-bit RPT to 48-bits. Now the 48-bit key is XOR with 48bit RPT and resulting output is given to the next step, which is the S-Box substitution.

**CODE:**

#include< bits/stdc++.h>

using namespace std;

string hex2bin(string p){

string ap="";

int l=p.length();

for(int i=0;i<l;i++){

string st="";

if(p[i]>='0'&&p[i]<='9'){

int te=int(p[i])-48;

while(te>0){

st+=char(te%2+48);

te/=2;

}

while(st.length()!=4)

st+='0';

for(int j=3;j>=0;j--)

ap+=st[j];

}

else{

int te=p[i]-'A'+10;

while(te>0){

st+=char(te%2+48);

te/=2;

}

for(int j=3;j>=0;j--)

ap+=st[j];

}}

return ap;}

int main(){

string p,l,r,ap="",ke,kp,rtem;

pre:;

cout<<"Enter the plain text in hexadecimal form (16bit) \n";

cin>>p;

if(p.length()!=16){

cout<<"enter all the bits\n";

goto pre;

}

for(int i=0;i<16;i++){

if((p[i]>='0'&&p[i]<='9')||(p[i]>='A'&&p[i]<='F'))

;

else{

cout<<"Not a valid hexadecimal string\n";

goto pre;

}}

int key1[64];

pr:;

cout<<"Enter the key in hexadecimal form (16bit) \n";

cin>>ke;

if(ke.length()!=16){

cout<<"enter all the key bits\n";

goto pr;

}

for(int i=0;i<16;i++){

if((ke[i]>='0'&&ke[i]<='9')||(ke[i]>='A'&&ke[i]<='F'))

;

else{

cout<<"Not a valid hexadecimal key string\n";

goto pr;

}}

p=hex2bin(p);

kp=hex2bin(ke);

for(int i=0;i<64;i++)

key1[i]=kp[i]-'0';

int keyp[56]={57 , 49 , 41 ,33 , 25 , 17 , 9,

1 , 58 ,50 , 42 , 34 , 26 , 18,

10 , 2 , 59 , 51 , 43 , 35 , 27,

19 , 11 , 3 , 60 , 52 , 44 , 36,

63 , 55 , 47 , 39 , 31 , 23 , 15,

7 , 62 , 54 , 46 , 38 , 30 , 22,

14 , 6 , 61 , 53 , 45 , 37 , 29,

21 , 13 , 5 , 28 , 20 , 12 , 4};

int key2[48]={14, 17, 11, 24, 1, 5,

3 , 28 , 15 , 6 , 21 , 10 ,

23 ,19 , 12 , 4 , 26 , 8,

16 ,7 , 27 , 20 , 13 , 2,

41 , 52 , 31 , 37 , 47 , 55,

30 , 40 , 51 , 45 , 33 , 48,

44 ,49 , 39 , 56 , 34 , 53,

46 ,42 , 50 , 36 , 29 , 32};

int key[16][48],keyl[28],keyr[28],nshift,temp1,temp2,pkey[56];

l=p.substr(0,32);

r=p.substr(32,32);

int i,t=1,j,row,col,temp,round=16;

j=0;

for(i=0;i<56;i++)

pkey[i]=key1[keyp[i]-1];

for(i=0;i<28;i++)

keyl[i]=pkey[i];

for(i=0;i<28;i++)

keyr[i]=pkey[i+28];

for(i=0;i<16;i++){

if(i==0||i==1||i==8||i==15)

nshift=1;

else

nshift=2;

while(nshift--){

temp1=keyl[0];

temp2=keyr[0];

for(j=0;j<27;j++){

keyl[j]=keyl[j+1];

keyr[j]=keyr[j+1];

}

keyl[27]=temp1;

keyr[27]=temp2;

}

for(j=0;j<24;j++)

key[i][j]=keyl[key2[j]-1];

for(j=24;j<48;j++)

key[i][j]=keyr[key2[j]-1-28];

}

int per[32]={16 , 7 , 20 ,21,

29 , 12 , 28 , 17,

1 , 15, 23, 26,

5 , 18 , 31 ,10,

2 , 8 , 24 , 14,

32 ,27, 3 , 9,

19 ,13, 30, 6,

22 , 11 , 4 , 25};

int s[8][4][16]={{

14,4,13,1,2,15,11,8,3,10,6,12,5,9,0,7,

0,15,7,4,14,2,13,1,10,6,12,11,9,5,3,8,

4,1,14,8,13,6,2,11,15,12,9,7,3,10,5,0,

15,12,8,2,4,9,1,7,5,11,3,14,10,0,6,13

},

{

15,1,8,14,6,11,3,4,9,7,2,13,12,0,5,10,

3,13,4,7,15,2,8,14,12,0,1,10,6,9,11,5,

0,14,7,11,10,4,13,1,5,8,12,6,9,3,2,15,

13,8,10,1,3,15,4,2,11,6,7,12,0,5,14,9

},

{

10,0,9,14,6,3,15,5,1,13,12,7,11,4,2,8,

13,7,0,9,3,4,6,10,2,8,5,14,12,11,15,1,

13,6,4,9,8,15,3,0,11,1,2,12,5,10,14,7,

1,10,13,0,6,9,8,7,4,15,14,3,11,5,2,12

},

{

7,13,14,3,0,6,9,10,1,2,8,5,11,12,4,15,

13,8,11,5,6,15,0,3,4,7,2,12,1,10,14,9,

10,6,9,0,12,11,7,13,15,1,3,14,5,2,8,4,

3,15,0,6,10,1,13,8,9,4,5,11,12,7,2,14

},

{

2,12,4,1,7,10,11,6,8,5,3,15,13,0,14,9,

14,11,2,12,4,7,13,1,5,0,15,10,3,9,8,6,

4,2,1,11,10,13,7,8,15,9,12,5,6,3,0,14,

11,8,12,7,1,14,2,13,6,15,0,9,10,4,5,3

},

{

12,1,10,15,9,2,6,8,0,13,3,4,14,7,5,11,

10,15,4,2,7,12,9,5,6,1,13,14,0,11,3,8,

9,14,15,5,2,8,12,3,7,0,4,10,1,13,11,6,

4,3,2,12,9,5,15,10,11,14,1,7,6,0,8,13

},

{

4,11,2,14,15,0,8,13,3,12,9,7,5,10,6,1,

13,0,11,7,4,9,1,10,14,3,5,12,2,15,8,6,

1,4,11,13,12,3,7,14,10,15,6,8,0,5,9,2,

6,11,13,8,1,4,10,7,9,5,0,15,14,2,3,12

},

{

13,2,8,4,6,15,11,1,10,9,3,14,5,0,12,7,

1,15,13,8,10,3,7,4,12,5,6,11,0,14,9,2,

7,11,4,1,9,12,14,2,0,6,10,13,15,3,5,8,

2,1,14,7,4,10,8,13,15,12,9,0,3,5,6,11

}};

while(round--){

rtem=r;

t=1;

string ep="",xorout="",sout="",soutt;

ep+=r[31];

for(i=0;i<32;i++){

if((t+1)%6==0){

ep+=r[4\*((t+1)/6)];

t++;

}

if(t%6==0&&i!=0){

ep+=r[4\*(t/6)-1];

t++;

}

ep=ep+r[i];

t++;

}

ep+=r[0];

for(i=0;i<48;i++)

xorout+=char(((int(ep[i])-48)^key[16-round-1][i])+48);

for(i=0;i<48;i+=6){

row=(int(xorout[i+5])-48)+(int(xorout[i])-48)\*2;

col= (int(xorout[i+1])-48)\*8+(int(xorout[i+2])-48)\*4+(int(xorout[i+3])48)\*2+(int(xorout[i+4])-48);

temp=s[i/6][row][col];

soutt="";

while(temp>0){

soutt+=char(temp%2+48);

temp/=2;

}

while(soutt.length()!=4)

soutt+='0';

for(j=soutt.length()-1;j>=0;j--)

sout+=soutt[j];

}

char pc[32];

for(i=0;i<32;i++)

pc[i]=sout[per[i]-1];

r="";

for(i=0;i<32;i++)

r+=char(((int(pc[i])-48)^(int(l[i])-48))+48);

l=rtem;

cout<<"Output after Round"<<16-round<<endl;

string cip="";

for(i=0;i<32;i+=4){

int te;

te=(int(l[i])-48)\*8+(int(l[i+1])-48)\*4+(int(l[i+2])-48)\*2+(int(l[i+3])-48);

if(te<10)

cip+=char(te+48);

else

cip+=char(te+55);

}

for(i=0;i<32;i+=4){

int te;

te=(int(r[i])-48)\*8+(int(r[i+1])-48)\*4+(int(r[i+2])-48)\*2+(int(r[i+3])-48);

if(te<10)

cip+=char(te+48);

else

cip+=char(te+55);

}

cout<<cip<<endl;

}

round=16;

string ltem;

while(round--)

{

ltem=l;

t=1;

string ep="",xorout="",sout="",soutt;

ep+=l[31];

for(i=0;i<32;i++){

if((t+1)%6==0){

ep+=l[4\*((t+1)/6)];

t++;

}

if(t%6==0&&i!=0){

ep+=l[4\*(t/6)-1];

t++;

}

ep=ep+l[i];

t++;

}

ep+=l[0];

for(i=0;i<48;i++)

xorout+=char(((int(ep[i])-48)^key[round][i])+48);

for(i=0;i<48;i+=6){

row=(int(xorout[i+5])-48)+(int(xorout[i])-48)\*2;

col= (int(xorout[i+1])-48)\*8+(int(xorout[i+2])-48)\*4+(int(xorout[i+3])48)\*2+(int(xorout[i+4])-48);

temp=s[i/6][row][col];

soutt="";

while(temp>0){

soutt+=char(temp%2+48);

temp/=2;

}

while(soutt.length()!=4)

soutt+='0';

for(j=soutt.length()-1;j>=0;j--)

sout+=soutt[j];

}

char pc[32];

for(i=0;i<32;i++)

pc[i]=sout[per[i]-1];

l="";

for(i=0;i<32;i++)

l+=char(((int(pc[i])-48)^(int(r[i])-48))+48);

r=ltem;

cout<<"Decrypted Output after Round"<<16-round<<endl;

string cip="";

for(i=0;i<32;i+=4){

int te;

te=(int(l[i])-48)\*8+(int(l[i+1])-48)\*4+(int(l[i+2])-48)\*2+(int(l[i+3])-48);

if(te<10)

cip+=char(te+48);

else

cip+=char(te+55);

}

for(i=0;i<32;i+=4){

int te;

te=(int(r[i])-48)\*8+(int(r[i+1])-48)\*4+(int(r[i+2])-48)\*2+(int(r[i+3])-48);

if(te<10)

cip+=char(te+48);

else

cip+=char(te+55);

}

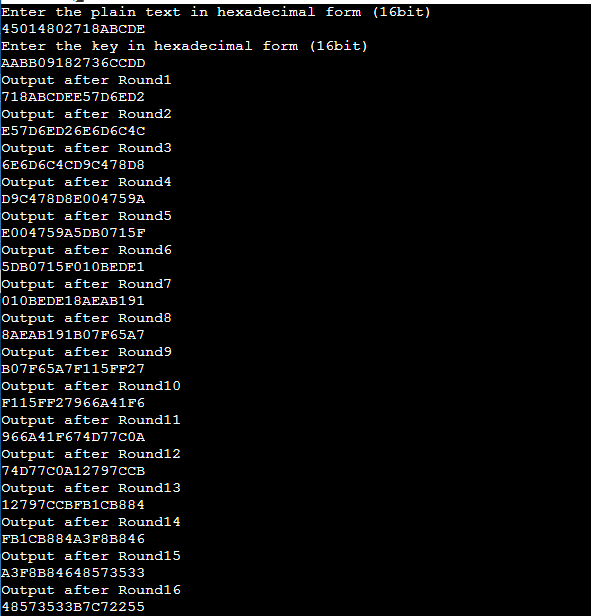
cout<<cip<<endl;

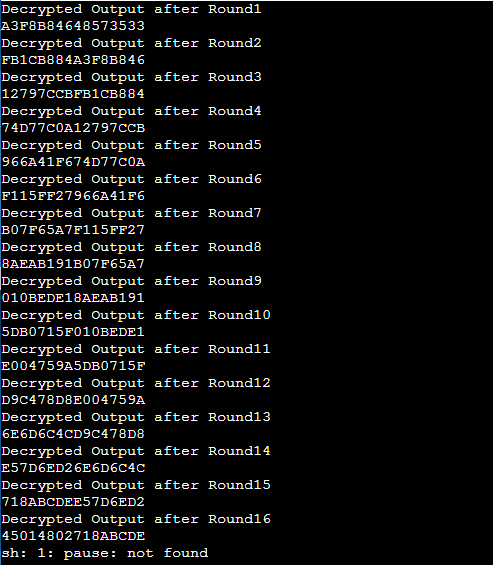
}

system("pause");

}

**OUTPUT:**





**Viva Voce**

1. **Give a one- or two-sentence definition of Symmetric key cryptography:**

* Encryption:

cyphertext = plaintext + encryption key

* Decryption:

ciphertext + decryption key = plaintext

such that decryption only succeeds if the encryption key and decryption key are the same key.

1. **Give a one- or two-sentence definition Asymmetric key cryptography.**

* As for symmetric key cryptography, except that the encryption key and decryption key do not have to be equal, they have to be inverses of each other. When an asymmetric key pair is generated, an encryption key and its inverse decryption key are generated.

1. **How can a message between 2 people be authenticated using symmetric encryption?**

* 1. Create a hash / digest of the message.

2. Encrypt the digest it with the shared key

3. Send the message and the encrypted digest to the recipient

4. Recipient creates a new digest of just the message

5. Recipient decrypts the received digest and compares it to his digest

**Date: 28/10/2021**

**Experiment-6**

**Aim:** To perform Diffie-Hellman algorithm.

**Theory:**

The Diffie-Hellman algorithm is being used to establish a shared secret that can be used for secret communications while exchanging data over a public network using the elliptic curve to generate points and get the secret key using the parameters.

• For the sake of simplicity and practical implementation of the algorithm, we will consider only 4 variables one prime P and G (a primitive root of P) and two private values a and b.

• P and G are both publicly available numbers. Users (say Alice and Bob) pick private values a and b and they generate a key and exchange it publicly, the opposite person received the key and from that generates a secret key after which they have the same secret key to encrypt.

**CODE:**

#include<stdio.h>

#include<math.h>

long long int power(long long int a, long long int b, long long int P) {

if (b == 1)

return a;

else

return (((long long int)pow(a, b)) % P); }

int main() {

long long int P= 23, G, x, a, y, b, ka, kb;

printf("The value of P : %lld\n", P);

G = 9;

printf("The value of G : %lld\n\n", G);

a = 4;

printf("The private key a for Ayush : %lld\n", a);

x = power(G, a, P);

b = 6;

printf("The private key b for Pandey : %lld\n\n", b);

y = power(G, b, P);

ka = power(y, a, P);

kb = power(x, b, P);

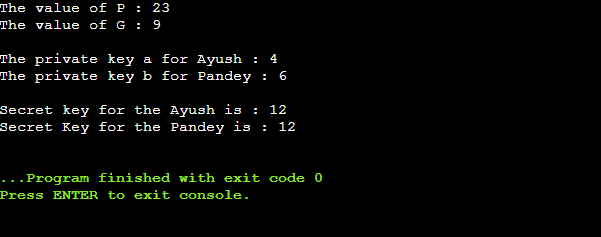
printf("Secret key for the Ayush is : %lld\n", ka);

printf("Secret Key for the Pandey is : %lld\n", kb);

return 0;

}

**OUTPUT:**



**Viva Voce**

1. **How can a message between 2 people be authenticated using asymmetric encryption?**

* 1. Create a hash / digest of the message.

2. Encrypt the digest with the sender’s private key, which can only be decrypted with the sender’s matching public key

3. Send the message and the encrypted digest to the recipient

4. Recipient creates a new digest of just the message

5. Recipient decrypts the received digest with the sender’s public and compares it to his digest

1. **The exercise is to calculate what the key would be for Alice and Bob using these known values:**

**Prime number selected is p = 337**

**Random number selected is g = 3 \*\*Generator**

**Alice random private number selected is xa = 7**

**Bob’s random private number selected is xb = 11. What is the key that Alice and Bob will use?**

* Alice: gamod p

37 mod 337

2187 mod 337= 165 – Public Value

gab= (gb)amod p

37\*11= (311)7mod 337

5.47 x 1036 = 1771477mod 337

5.47 x 1036 = 5.47 x 1036 mod 337= 5 – Shared Secret Key

Bob: gb mod p = 3

311 mod 337

177147 mod 337

= 222 – Public Value

311\*7= (37)11mod 337

5.474e+36 = 5.474e+36 mod 337 = 5 -Shared Key

**Date: 04/11/2021**

**Experiment-7**

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

NeSSi consists of three distinct components, the Graphical User Interface, the simulation backend and the result database. Each of these modules may be run on separate machines depending on the computational requirements; furthermore, this modular design facilitates network security researchers using NeSSi to easily exchange

**Graphical User Interface:** The graphical frontend of NeSSi allows the user to create and edit network topologies, attach runtime information, and schedule them for execution at the simulation backend. On the other hand, finished (or even currently executing, long-running) simulations can be retrieved from the database server and the corresponding simulation results are visualized in the GUI.

**Simulation Backend:** The actual simulation is performed on machine with hardware dedicated solely to this purpose, the simulation backend. At the DAI-Labor for example, the NeSSi simulation backend runs on a Sun XFire 4600 blade server (8 blades, 8 cores per blade). Once a session is submitted for execution from the GUI, the simulation backend parses the desired session 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.

**Parallel Execution Model:** Simulations in large-scale networks are very costly in terms of processing time and memory consumption. Therefore, NeSSi has been designed as a distributed simulation, allowing the subdivision of tasks to different computers and processes in a parallel-execution model.

**Discrete Event Simulation:** NeSSi² is a discrete-event-based simulation tool, which allows to plan and to schedule time-based events such as network failures, attacks, etc.

**Simulation Result Database Server**

In NeSSi, we refer to a scenario where we generate traffic via pre-defined profiles on a single network over a certain amount of time as a session. The accurate reproduction of a session enables users to use different detection methods or various deployments of detection units for the same traffic data set. This allows the comparison of performance and detection efficiency of different security framework setups. The data types to be logged are specified by the user in the session parameters. The network model is saved in an XML file. This network file is stored and annotated with a version number based on its hash code in order to link a network uniquely to a session. Additionally, attack related events can be stored in the database for evaluation purposes.

**Standard Components and Plugin API**

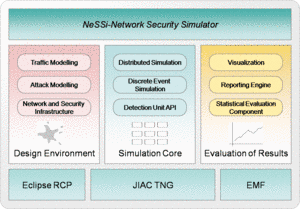
NeSSi² has been designed as a modularized application. Building on the Eclipse framework, it uses the inherent plugin mechanism to allow users to easily extend the functionality of NeSSi² and share it with other developers. Often, security researchers have very specific demands regarding the protocols and features the simulation tool needs to offer. Naturally, NeSSi² provides a rich set of basic protocols and detection unit implementations; nevertheless, the special needs of various application areas (wireless networks, sensor networks, MPLS etc.) necessitates a plugin API to allow the user to adapt NeSSi² to his needs and add the functionality that is not provided by NeSSi out-of-the-box.

Hence, the NeSSi² extension API allows the creation of

• New device types with user-defined properties

• New protocols defining the behavior of the network at runtime

• Application definitions, allowing dynamic behavior to be defined, attached to a device or link, and scheduled for execution in the simulation



**Date: 11/11/2021**

**Experiment-8**

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

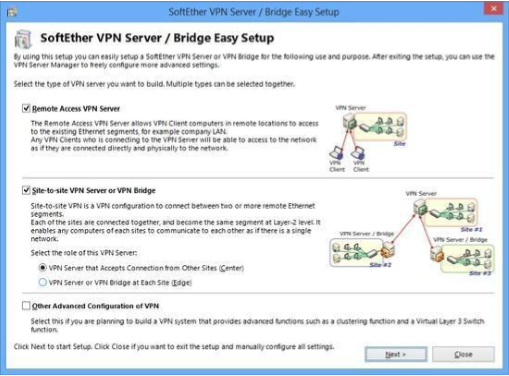
**STEPS:**

SoftEther VPN can construct distributed virtual Ethernet segment. If you can make some geologically distributed computers enable to communicate each other as if they are connected to the single Ethernet network, using SoftEther VPN is the easiest way.

First, set up a VPN Server. Next, set up VPN Clients on each member PCs. Finally start VPN connections on each VPN client. Then each client can use any kinds of IP-based or Ethernet based protocols via the VPN even if they are distributed around the world.

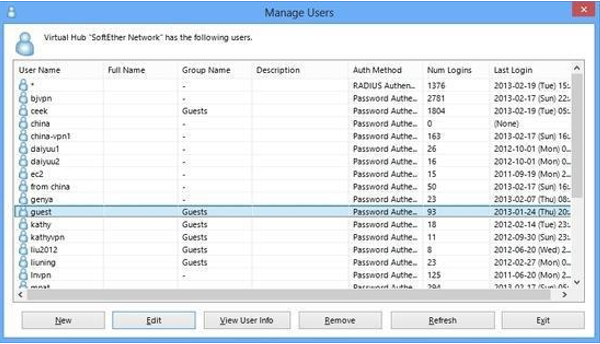
**1. Set up SoftEther VPN Server**

Designate a computer in the group as the VPN Server. Set up SoftEther VPN Server on that computer. It is very easy by using Installer and Initial Setup Wizard based GUI.



**2. Create Users**

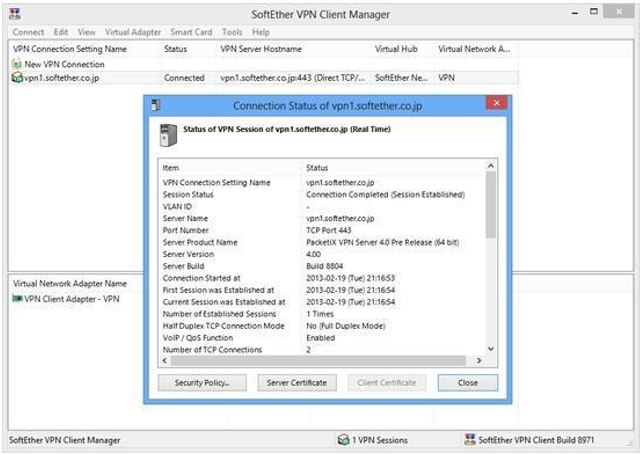
On the VPN Server you can add several user objects on the Virtual Hub. Each user object has a password. After that, distribute pairs of username and password to each member of the VPN.



**3. Set up VPN Client on Each Member’s PC**

On each member’s PC install SoftEther VPN Client. Enter the server address, username and password for each PC.

If a member of the VPN is Mac OS X, iPhone or Android, set up L2TP/IPsec VPN client on each PC instead of SoftEther VPN. Another solution is to use OpenVPN Client on Mac OS X, iPhone or Android to connect to SoftEther VPN Server.

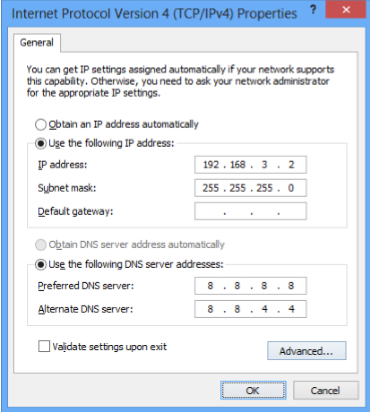


**4. Set up IP Addresses**

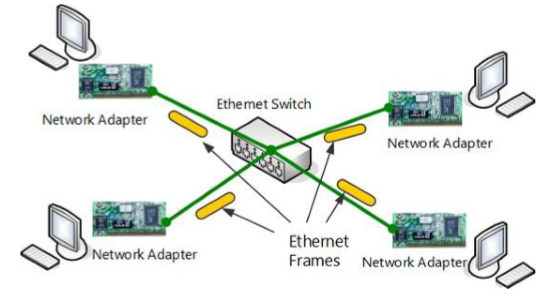
The characteristics of Soft Ether’s virtual private network are exactly same to a physical Ethernet segment. So you should decide the IP addresses of every member PCs.

Like the physical Ethernet, the simplest way is to set up private IP addresses to each PC, for example 192.168.0.0/24. Make sure not to overlap to physical-using private IPs.

Another solution is to use DHCP server for automated IP address allocation. You can activate Virtual DHCP Server Function on the SoftEther VPN Server and it will distribute 192.168.30.0/24 by default.



**5. Communicate Like Physical Ethernet**

Once every computer is connected to the Virtual Hub on SoftEther VPN Server, all computers and smart-phones can now communicate mutually as if they are all connected to the single Ethernet network. You can enjoy File Sharing protocols, Remote Printing applications, Remote Desktop applications, SQL Database applications and any other LANbased applications despite the distances and differences of physical location.

**Viva Voce**

1. **If you receive an error related to Protocol Error is occurring, what all troubleshooting measures can be taken up?**

* The first thing to do when you get a Protocol Driver Error/Error 1046 is to verify Citrix Receiver/Workspace is upgraded to the latest version if possible or the latest version that has been validated in the destination environment. If upgrading receiver doesn’t help, or if the latest version is already running on the affected workstation, next recommendation would be to perform a full uninstall of receiver utilizing the Citrix Receiver Cleanup Utility.

1. **There is a large number of broadcast packets constantly being sent over the network. What should I check?**

* 1. Storm control and equivalent protocols allow you to rate-limit broadcast packets. If your switch has such a mechanism, turn it on.

2. Ensure IP-directed broadcasts are disabled on your Layer 3 devices. There’s little to no reason why you’d want broadcast packets coming in from the internet going to a private address space. If a storm is originating from the WAN, disabling IP-directed broadcasts will shut it down.

3. Split up your broadcast domain. Creating a new VLAN and migrating hosts into it will load balance the broadcast traffic to a more acceptable level. Broadcast traffic is necessary and useful, but too much of it eventually leads to a poor network experience.

4. Check how often ARP tables are emptied.

1. **Which protocol is suited for communication over insecure channel?**

* A secure communication over an insecure channel without any prior exchanged key can be established with the help of an authentication step to exchange a public key and then using public-key cryptography such as RSA.

1. **How many firewalls should be there in the network?**

* If your network is entirely client-protecting, or is client-protecting with just a few incoming services, such as email, then one firewall (or a pair of firewalls configured as a high-availability pair) is probably all you need.

**Date: 11/11/2021**

**Experiment-9**

**Aim:** PGP tool usage using Command Line version of PGP.

PGP Command Line is a command line product for performing cryptography and key management tasks. It operates as a stand-alone product that performs those tasks locally. It can also operate as a client product that interacts with Symantec Encryption Management Server to perform those tasks.

With PGP Command Line, you can write command line scripts that use Symantec encryption technology to perform these tasks:

* Encrypt, sign, and decrypt individual files or collections of files
* Create and manage keys on a local keyring
* Access keys on Symantec Encryption Management Server and other keyservers
* Manage keys on Symantec Encryption Management Server
* Create consumer (user) accounts on Symantec Encryption Management Server
* Manage X.509 certificates, including requesting and validating a certificate
* Encrypt, sign, and decrypt email

You can insert PGP Command Line commands into scripts for automating tasks. PGP Command Line commands are easily added to shell scripts or scripts written with scripting languages, such as Perl or Python.

For example, consider a company that regularly backs up a large sensitive database to an off-site location. A script runs automatically to perform the backup. This company can add PGP Command Line commands to that script to compress and encrypt the database before transmitting it to the off-site location. It can also add commands to decrypt and uncompress the database when it arrives at its destination.

**To Install on Windows**

To install PGP Command Line onto a Windows system:

1. Close all Windows applications.
2. Download the installer application, PGPCommandLine[version]Win.zip, to a known location on your system.
3. Unzip the file PGPCommandLine[version]Win.zip. You will get the following file: PGPCommandLine[version]Win.msi.
4. Double click on PGPCommandLine[version]Win.msi.
5. Follow the on-screen instructions.
6. If prompted, restart your machine. A restart is needed only if other Symantec encryption products are also installed on the same machine.

The Windows PGP Command Line application, pgp.exe, is installed into:

C:\Program Files\PGP Corporation\PGP Command Line\

After you run PGP Command Line for the first time, its home directory will be created automatically in the user’s home directory:

C:\Documents and Settings\\My Documents\PGP\

Application data is stored in the directory:

C:\Documents and Settings\\Application Data\PGP Corporation\PGP

Locations may be different for the different Windows versions.