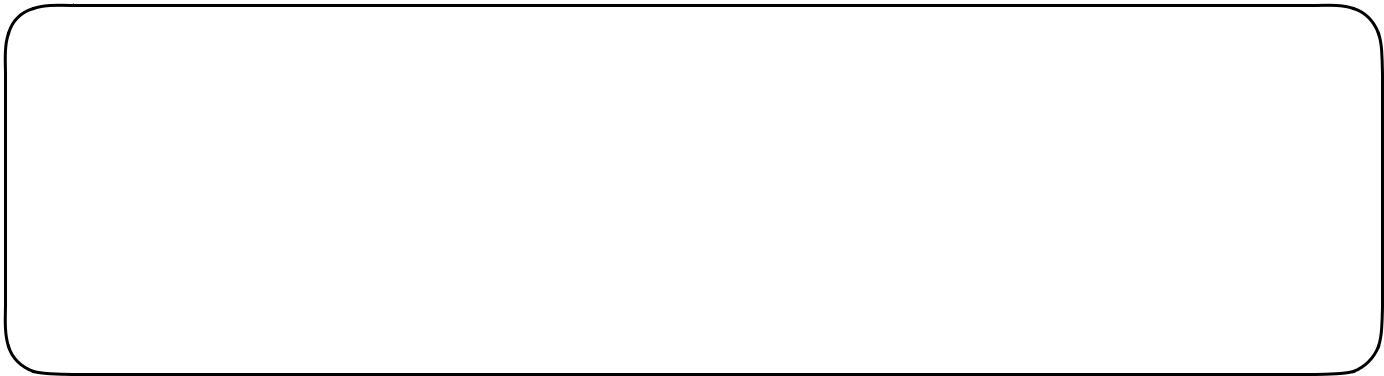
***Institute’s Vision***



To be an organisation with potential for excellence in engineering and

management for the advancement of society and human kind.

***Institute’s Mission***



To excel in academics, practical engineering, management and to commence research endeavors.

To prepare students for future opportunities.

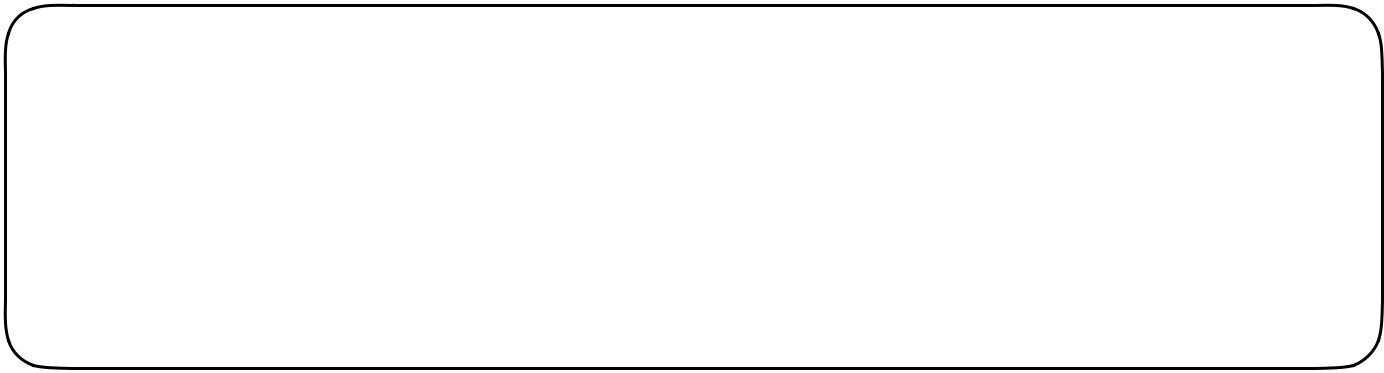
To nurture students with social and ethical responsibilities.

***Department’s Vision***



To create IT graduates with ethical and employable skills.

***Department’s Mission***

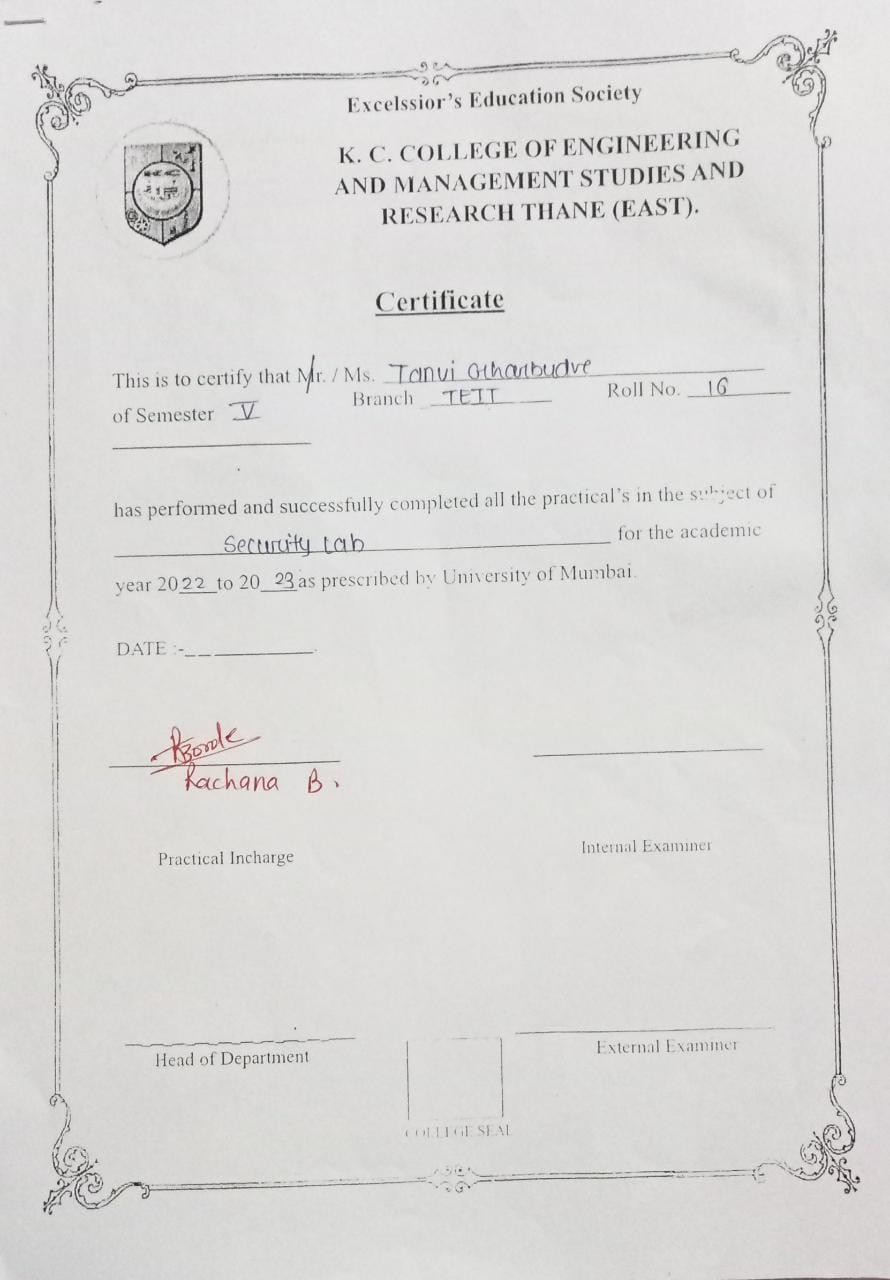


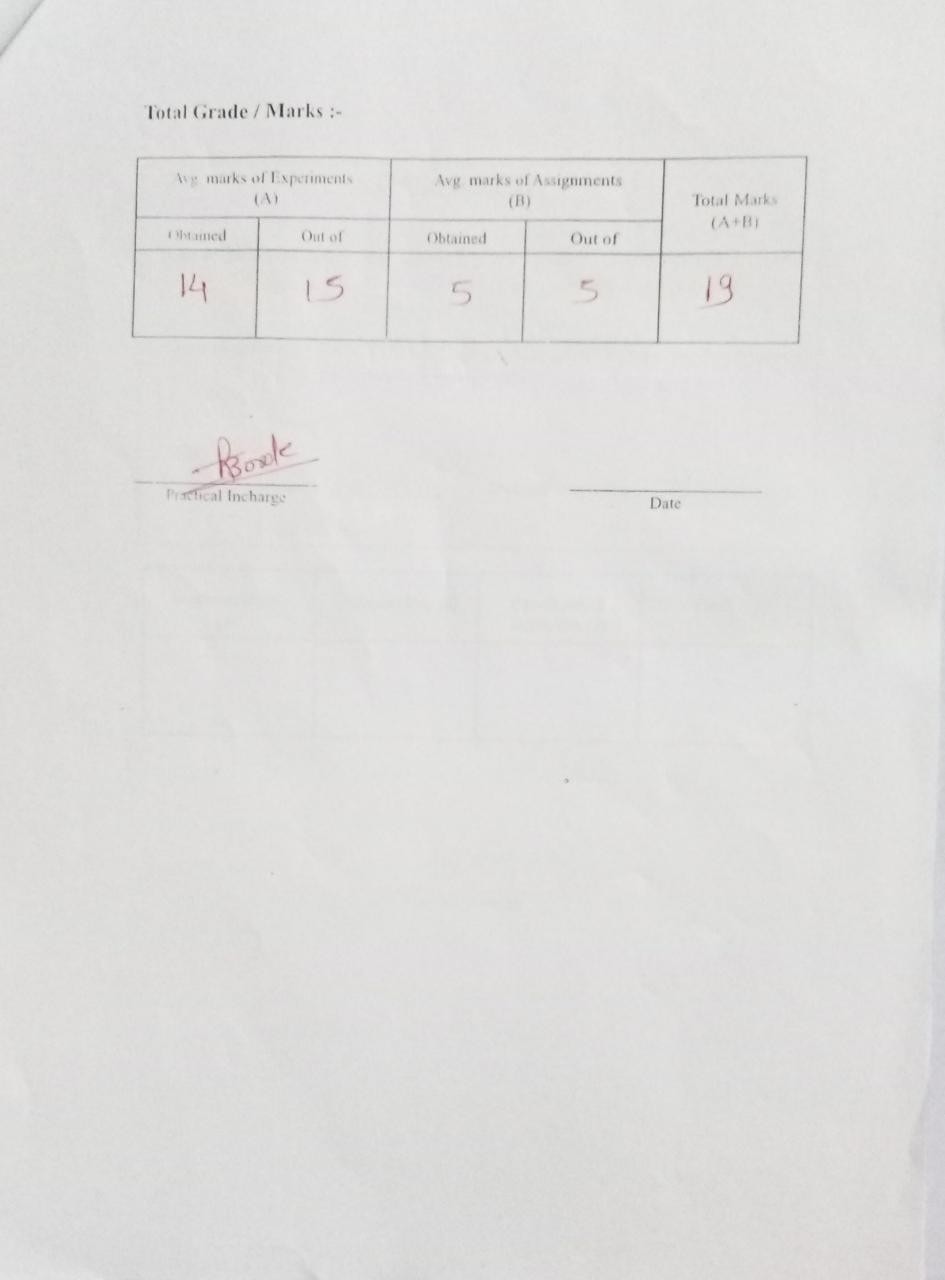
To imbibe problem solving and analytical skills through teaching

learning process.

To impart technical and managerial skills to meet the industry requirement.

To encourage ethical and value based education.





#### SYLLABUS

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr.**  **No.** | **Description** | **Hours** | **LO**  **Mapping** |
| 1 | Breaking the Mono-alphabetic Substitution Cipher using Frequency analysis method. | 2 | LO1 |
| 2 | Design and Implement a product cipher using Substitution ciphers. | 4 | LO1 |
| 3 | . Cryptanalysis or decoding Playfair, vigenere cipher. | 2 | LO1 |
| 4 | Encrypt long messages using various modes of operation using AES or DES. | 2 | LO2 |
| 5 | Cryptographic Hash Functions and Applications (HMAC): tounderstand the need, design and applications of collision resistant hash functions. | 2 | LO2 |
| 6 | Implementation and analysis of RSA cryptosystem and Digital signature scheme using RSA. | 2 | LO2 |
| 7 | Study the use of network reconnaissance tools like WHOIS, dig,traceroute, nslookup to gather information about networks and  domain registrars. | 2 | LO4 |
| 8 | Study of packet sniffer tools wireshark: - a. Observer performance in promiscuous as well as non-promiscuous mode. b. Show the packets can be traced based on different filters. | 2 | LO3 |
| 9 | Download, install nmap and use it with different options to scan open ports, perform OS fingerprinting, ping scan, tcp port scan,udp port scan, etc. | 2 | LO3 |
| 10 | Study of malicious software using different tools:   1. Keylogger attack using a keylogger tool. 2. Simulate DOS attack using Hping or other tools 3. Use the NESSUS/ISO Linux tool to scan the network for vulnerabilities. | 2 | LO5 |
| 11 | Study of Network security by   1. Set up IPSec under Linux. 2. Set up Snort and study the logs. 3. Explore the GPG tool to implement email security | 2 | LO6 |
| 12 | Study and use of wireless security tool- Kismet | 2 | LO6 |

**Program Outcomes**

Engineering Graduates will be able to:

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineeringfundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis**: Identify, formulate, review research literature, and analyze complexengineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems**: Use research-based knowledge and researchmethods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assesssocietal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability**: Understand the impact of the professional engineeringsolutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and normsof the engineering practice.
9. **Individual and team work**: Function effectively as an individual, and as a member or leader indiverse teams, and in multidisciplinary settings.
10. **Communication**: Communicate effectively on complex engineering activities with theengineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning**: Recognize the need for, and have the preparation and ability to engage inindependent and life-long learning in the broadest context of technological change.

**Department of Information Technology Subject: Security Lab**

**Semester:**

**VClass:**

**TE**

**Lab Outcomes**

|  |  |
| --- | --- |
| Lab Code ITL702 | Lab Outcomes |
| At the end of experiment student will able to |
| ITL502.1 | Illustrate symmetric cryptography by implementing classical ciphers. |
| ITL502.2 | Analyze Key management, distribution and user authentication |
| ITL502.3 | Use tools like sniffers, port scanners and other related tools for  analyzing packets in a network |
| ITL502.4 | Understand Explore the different network reconnaissance tools to gather information about networks. |
| ITL502.5 | Use open-source tools to scan the network for vulnerabilities and  simulate attacks. |
| ITL502.6 | Demonstrate the network security system using open source tools. |

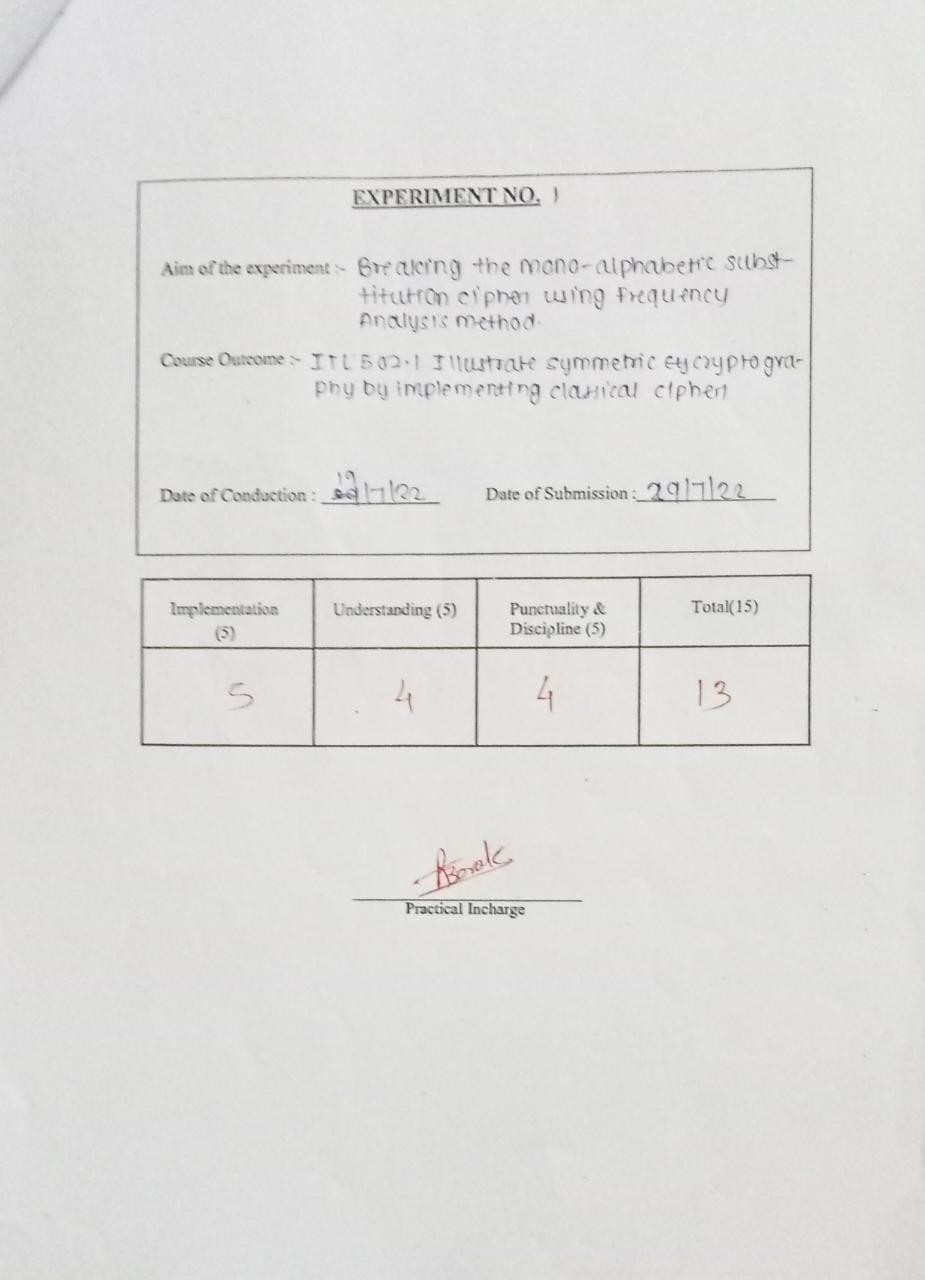
**Rubrics of Practical**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Rubrics Description | Maximum Marks  Weight | 15-12 | 12-9 | 9-6 | 6-0 |
| Implementation | 5 | Successful | Output | Few errors | Incorrect |
| (R1) | completion | correct but | in the | Output |
|  | with accurate output (5-4) | not precise (4-3) | output (3-2) | (2-0) |
| Understanding | 5 | Understanding | Understand | Improper | No |
| (R2) | Experiment | Experiment | Conclusion | Conclusion |
|  | and drawn | but | (3-2) | (2-0) |
|  | correct | conclusion |  |  |
|  | conclusion | less |  |  |
|  | (5-4) | accurate (4-3) |  |  |
| Punctuality and | 5 | Submission | Submission | Submission | Submission |
| Discipline | within a week | after week | after two | after three |
| (R3) | (5-4) | (4-3) | weeks (3-2) | weeks and more (2-0) |

TABLE OF CONTENTS

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Sr. No | Name of Experiment | Date of Conduction | Date of Submission | Page No. | Grade  /  Marks | Sign |
|  |  |  |  |  |  |  |
| 1 | Breaking the Mono-alphabetic Substitution Cipher using  Frequency analysis method. | 19/7/22 | 29/7/22 |  |  |  |
| 2 | Design and Implement a product cipher using Substitution ciphers | 26/8/22 | 20/7/22 |  |  |  |
| 3 | Cryptanalysis or decoding Playfair, vigenere cipher | 2/8/22 | 9/8/22 |  |  |  |
| 4 | Encrypt long messages using various modes of operation using  AES or DES. | 9/8/22 | 23/8/22 |  |  |  |
| 5 | Cryptographic Hash Functions and Applications (HMAC): to  understand the need, design and applications of collision resistant hash functions | 23/8/22 | 20/9/22 |  |  |  |
| 6 | Implementation and analysis of RSA cryptosystem and  Digital signature scheme using RSA. | 20/9/22 | 4/10/22 |  |  |  |
| 7 | Study the use of network reconnaissance tools like  WHOIS, dig,traceroute, nslookup to gather information about networks anddomain registrars | 27/9/22 | 4/10/22 |  |  |  |
| 8 | Study of packet sniffer tools wireshark: - a. Observer performance in promiscuous as well as  non-promiscuous mode. b. Show the packets can be traced based on different filters. | 4/10/22 | 10/10/22 |  |  |  |
| 9 | Download, install nmap and use it with different options to  scan open ports, perform OS fingerprinting, ping scan, tcp port scan,udp port scan,  etc. | 10/10/22 | 11/10/22 |  |  |  |
| 10 | Study of malicious software using different tools:   1. Keylogger attack using a keylogger tool. 2. Simulate DOS attack using Hping or | 11/10/22 | 21/10/22 |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | other tools  c) Use the NESSUS/ISO Kali Linux tool to scan the network for vulnerabilities. |  |  |  |  |  |
| 11 | Study of Network security by   1. Set up IPSec under Linux. 2. Set up Snort and study the logs. 3. Explore the GPG tool to implement email security | 11/10/22 | 21/10/22 |  |  |  |
| 12 | Study and use of wireless security tool- Kiset | 27/10/22 | 4/10/22 |  |  |  |



# Experiment 1

### Aim:

Breaking the Mono-alphabetic Substitution Cipher using Frequency analysis method.

### Theory:

Consider we have the plain text "cryptography". By using the substitution table below, we can encrypt our plain text as follows: abcdefgh i j k l mnopqr s t u vwxyz

JI BRKTCNOFQYG AUZHSVWMXL DEP

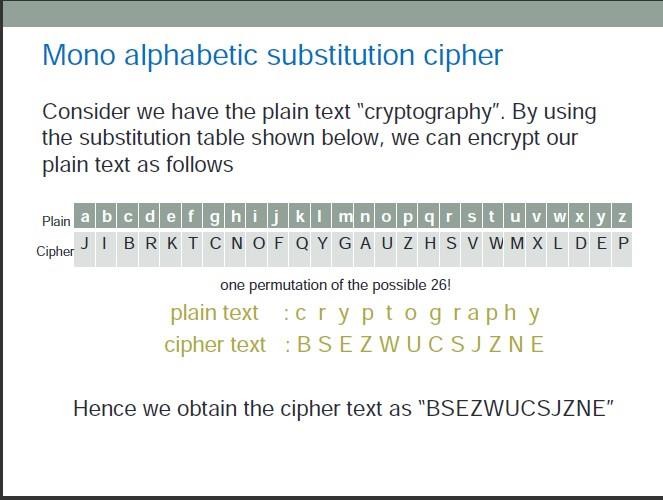
plain text: c r y p t o g r a p h y

cipher text: B S E Z W U C S J Z N E

Hence we obtain the cipher text as "BSEZWUCSJZNE".

### Cryptanalysis

Note that the frequency of occurrence of characters in the plaintext is "preserved" in the ciphertext. For instance, the most frequent character in the ciphertext is likely to be the encryption of the plaintext character "e" which is the most frequently occurring charecter in English. For a very brief theory of the mono- alphabetic substitution cipher and its cryptanalysis, click [here](https://cse29-iiith.vlabs.ac.in/exp/substitution-cipher/docs/monoalphacipher.pdf)



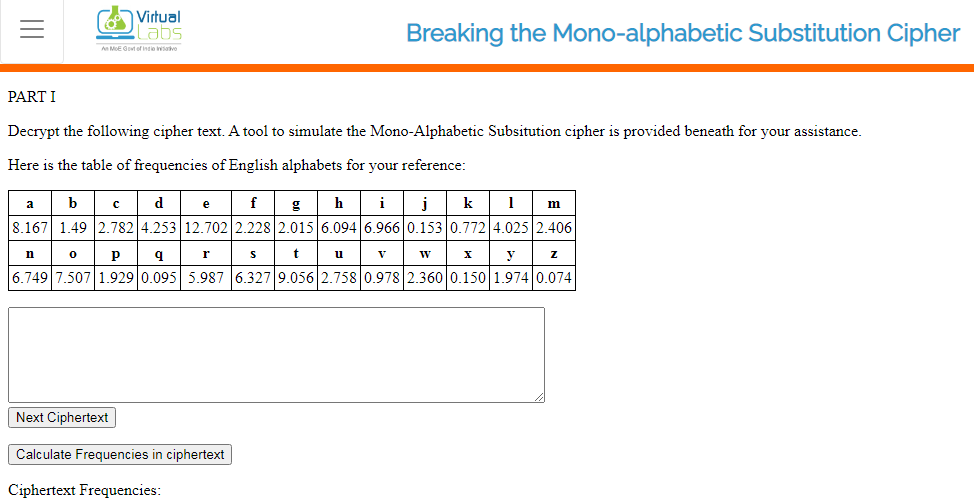
**Vlabs:** Vlabs:

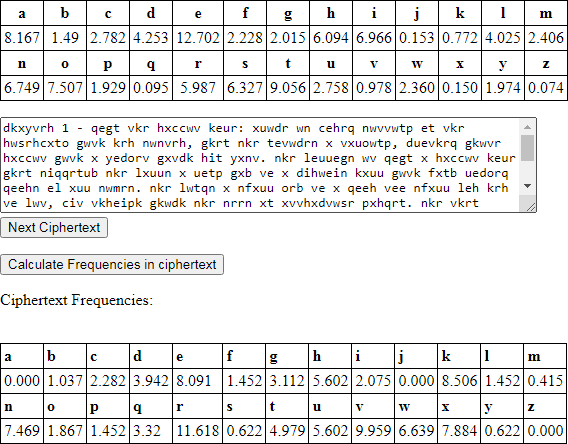
[http://cse29-](http://cse29-iiith.vlabs.ac.in/exp2/Theory.html?domain=Computer%20Science&lab=Cryptography%20Lab) [iiith.vlabs.ac.in/exp2/Theory.html?domain=Computer%20Science&lab=Cryptography%20Lab](http://cse29-iiith.vlabs.ac.in/exp2/Theory.html?domain=Computer%20Science&lab=Cryptography%20Lab)

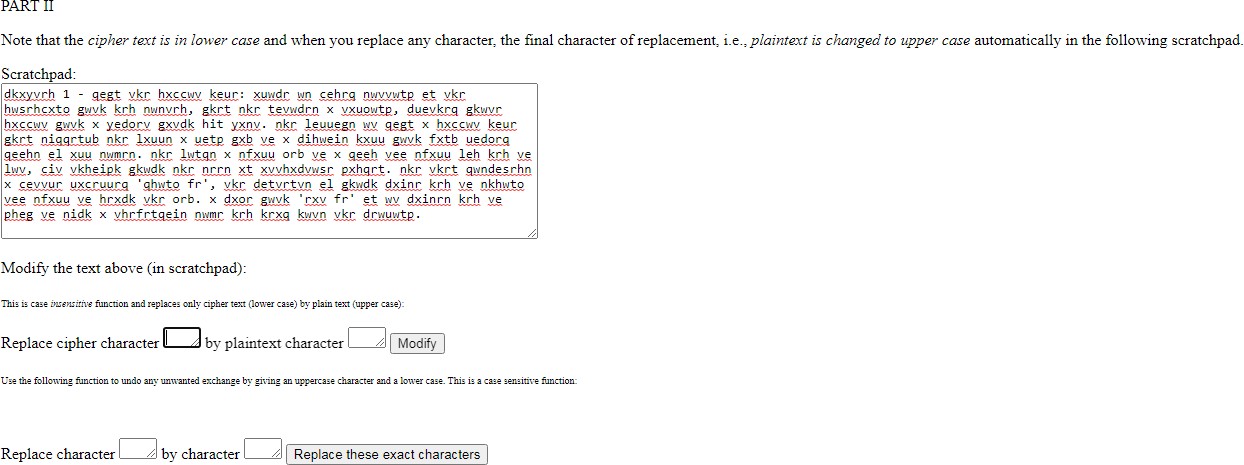
### EXP

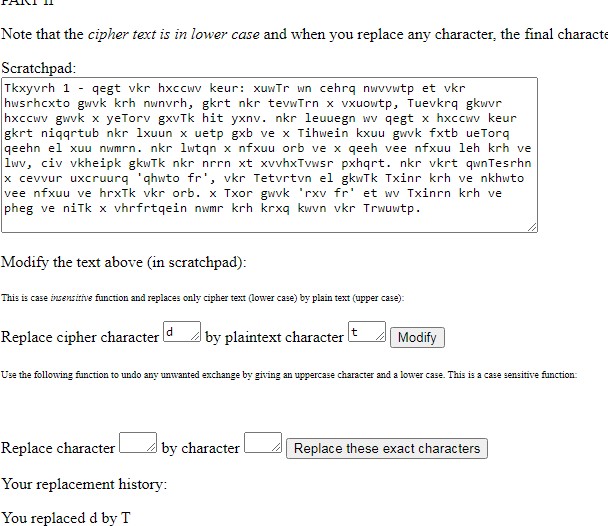


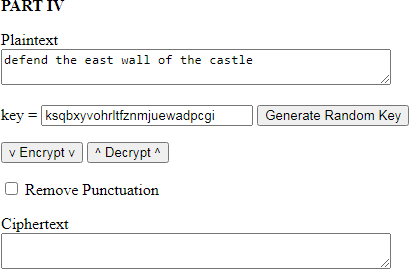
**Output:**



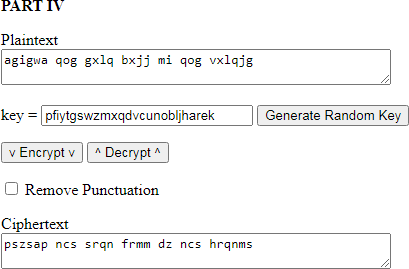






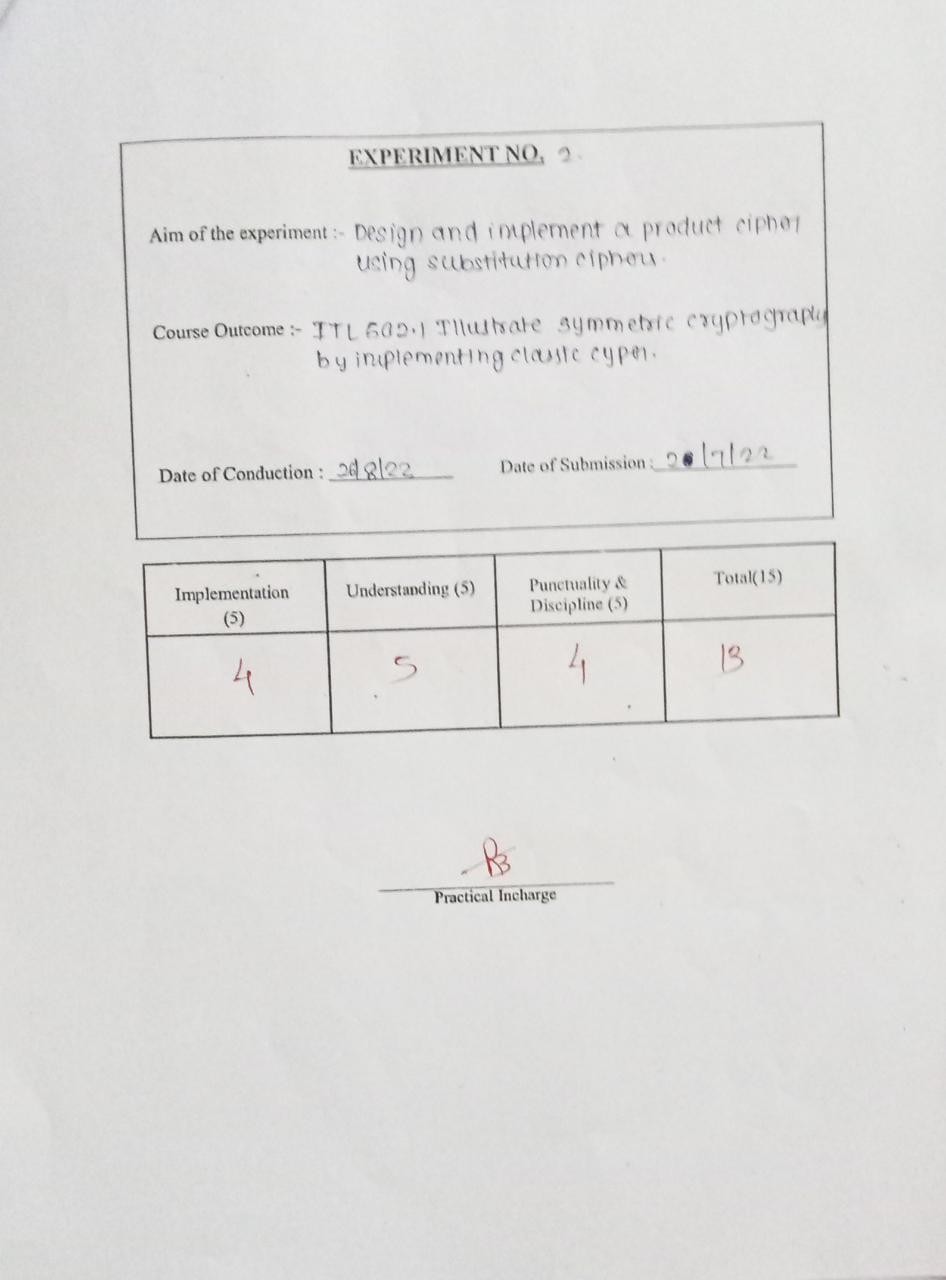






### Conclusion:

Successfully Breaked the Mono-alphabetic Substitution Cipher using Frequency analysis method.



**Aim:**

**Experiment no 2**

Design and Implement a product cipher using Substitution ciphers.

**Theory:**

Product cipher, data encryption scheme in which the ciphertext produced by encrypting a plaintext document is subjected to further encryption. By combining two or more simple transposition ciphers or substitution ciphers, a more secure encryption may result.

Product ciphers were a useful device for cryptographers, and in fact double transposition or product ciphers on key word-based rectangular matrices were widely used. There was also some use of a class of product ciphers known as fractionation systems, wherein a substitution was first made from symbols in the plaintext to multiple symbols (usually pairs, in which case the cipher is called a biliteral cipher) in the ciphertext, which was then encrypted by a final transposition, known as superencryption. One of the most famous field ciphers of all time was a fractionation system, the ADFGVX cipher employed by the German army during World War I. This system used a 6 × 6 matrix to substitution-encrypt the 26 letters and 10 digits into pairs of the symbols A, D, F, G, V, and X. The resulting biliteral cipher was then written into a rectangular array and route encrypted by reading the columns in the order indicated by a key word.

#### Algorithm for Substitution Cipher:

Input:

* A String of both lower- and upper-case letters, called PlainText.
* An Integer denoting the required key. Procedure:
* Create a list of all the characters.
* Create a dictionary to store the substitution for all characters.
* For each character, transform the given character as per the rule, depending on whether we’re encrypting or decrypting the text.
* Print the new string generated.

### Program:

# Python program to demonstrate # Substitution Cipher

import string

# A list containing all characters all\_letters= string.ascii\_letters

"""

create a dictionary to store the substitution for the given alphabet in the plain text based on the key

dict1 = {} key = 4

for i in range(len(all\_letters)):

dict1[all\_letters[i]] = all\_letters[(i+key)%len(all\_letters)]

plain\_txt= "I am studying Data Encryption" cipher\_txt=[]

# loop to generate ciphertext for char in plain\_txt:

if char in all\_letters:

temp = dict1[char] cipher\_txt.append(temp)

else:

temp =char cipher\_txt.append(temp)

cipher\_txt= "".join(cipher\_txt) print("Cipher Text is: ",cipher\_txt)

"""

create a dictionary to store the substitution for the given alphabet in the cipher

text based on the key """

dict2 = {}

for i in range(len(all\_letters)):

dict2[all\_letters[i]] = all\_letters[(i-key)%(len(all\_letters))]

# loop to recover plain text decrypt\_txt = []

for char in cipher\_txt:

if char in all\_letters:

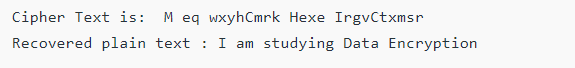
temp = dict2[char] decrypt\_txt.append(temp)

else:

temp = char decrypt\_txt.append(temp)

decrypt\_txt = "".join(decrypt\_txt) print("Recovered plain text :", decrypt\_txt)

### Output:



**Conclusion**: successfully Design and Implement a product cipher using Substitution ciphers.

**Lab Outcome :-** ITL502.1 Illustrate symmetric cryptography by implementing classical ciphers.

**Aim:**

# Experiment 3

Cryptanalysis or decoding Playfair, vigenere cipher

**Theory:**

Monoalphabetic cipher is a substitution cipher in which for a given key, the cipher alphabet for each plain alphabet is fixed throughout the encryption process. For example, if ‘A’ is encrypted as ‘D’, for any number of occurrence in that plaintext, ‘A’ will always get encrypted to ‘D’.

All of the substitution ciphers we have discussed earlier in this chapter are monoalphabetic; these ciphers are highly susceptible to cryptanalysis.

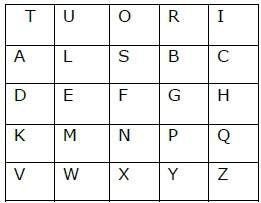
Polyalphabetic Cipher is a substitution cipher in which the cipher alphabet for the plain alphabet may be different at different places during the encryption process. The next two examples, **playfair and Vigenere Cipher are polyalphabetic ciphers**.

#### Playfair Cipher

In this scheme, pairs of letters are encrypted, instead of single letters as in the case of simple substitution cipher.

In playfair cipher, initially a key table is created. The key table is a 5×5 grid of alphabets that acts as the key for encrypting the plaintext. Each of the 25 alphabets must be unique and one letter of the alphabet (usually J) is omitted from the table as we need only 25 alphabets instead of 26. If the plaintext contains J, then it is replaced by I.

The sender and the receiver deicide on a particular key, say ‘tutorials’. In a key table, the first characters (going left to right) in the table is the phrase, excluding the duplicate letters. The rest of the table will be filled with the remaining letters of the alphabet, in natural order. The key table works out to be −



Process of Playfair Cipher

* First, a plaintext message is split into pairs of two letters (digraphs). If there is an odd number of letters, a Z is added to the last letter. Let us say we want to encrypt the message “hide money”. It will be written as −

HI DE MO NE YZ

* The rules of encryption are −

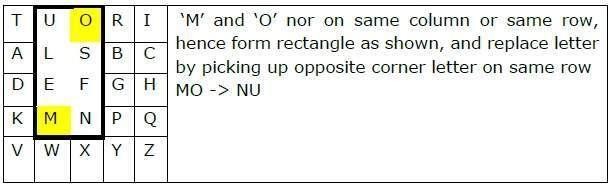
o If both the letters are in the same column, take the letter below each one (going back to the top if at the bottom)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| T | U | O | R | I | ‘H’ and ‘I’ are in same column, hence take letter below them to replace. HI  → QC |
| A | L | S | B | C |
| D | E | F | G | H |
| K | M | N | P | Q |
| V | W | X | Y | Z |

* If both letters are in the same row, take the letter to the right of each one (going back to the left if at the farthest right)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| T | U | O | R | I | ‘D’ and ‘E’ are in same row, hence take letter to the right of them to replace. DE → EF |
| A | L | S | B | C |
| D | E | F | G | H |
| K | M | N | P | Q |
| V | W | X | Y | Z |

* If neither of the preceding two rules are true, form a rectangle with the two letters and take the letters on the horizontal opposite corner of the rectangle.



Using these rules, the result of the encryption of ‘hide money’ with the key of ‘tutorials’ would be −

QC EF NU MF ZV

Decrypting the Playfair cipher is as simple as doing the same process in reverse. Receiver has the same key and can create the same key table, and then decrypt any messages made using that key.

Security Value

It is also a substitution cipher and is difficult to break compared to the simple substitution cipher. As in case of substitution cipher, cryptanalysis is possible on the Playfair cipher as well, however it would be against 625 possible pairs of letters (25x25 alphabets) instead of 26 different possible alphabets.

The Playfair cipher was used mainly to protect important, yet non-critical secrets, as it is quick to use and requires no special equipment.

#### Vigenere Cipher

This scheme of cipher uses a text string (say, a word) as a key, which is then used for doing a number of shifts on the plaintext.

For example, let’s assume the key is ‘point’. Each alphabet of the key is converted to its respective numeric value: In this case,

p → 16, o → 15, i → 9, n → 14, and t → 20.

Thus, the key is: 16 15 9 14 20. Process of Vigenere Cipher

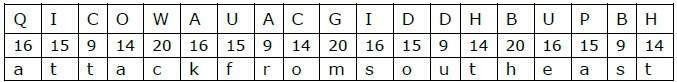
* The sender and the receiver decide on a key. Say ‘point’ is the key. Numeric representation of this key is ‘16 15 9 14 20’.
* The sender wants to encrypt the message, say ‘attack from south east’. He will arrange plaintext and numeric key as follows −



* He now shifts each plaintext alphabet by the number written below it to create ciphertext as shown below −



* Here, each plaintext character has been shifted by a different amount – and that amount is determined by the key. The key must be less than or equal to the size of the message.
* For decryption, the receiver uses the same key and shifts received ciphertext in reverse order to obtain the plaintext.



Security Value

Vigenere Cipher was designed by tweaking the standard Caesar cipher to reduce the effectiveness of cryptanalysis on the ciphertext and make a cryptosystem more robust. It is significantly **more secure than a regular Caesar Cipher**.

In the history, it was regularly used for protecting sensitive political and military information. It was referred to as the **unbreakable cipher** due to the difficulty it posed to the cryptanalysis.

Variants of Vigenere Cipher

There are two special cases of Vigenere cipher −

* The keyword length is same as plaintect message. This case is called **Vernam Cipher**. It is more secure than typical Vigenere cipher.
* Vigenere cipher becomes a cryptosystem with perfect secrecy, which is called **One-time pad**.

### Programs:

**Vigenère Cipher**

CODE-

class Main

{

static String generateKey(String str, String key)

{

int x = str.length(); for (int i = 0; ; i++)

{

if (x == i)

i = 0;

if (key.length() == str.length()) break;

key+=(key.charAt(i));

}

return key;

}

static String cipherText(String str, String key)

{

String cipher\_text="";

for (int i = 0; i < str.length(); i++)

{

int x = (str.charAt(i) + key.charAt(i)) %26; x += 'A';

cipher\_text+=(char)(x);

}

return cipher\_text;

}

static String originalText(String cipher\_text, String key)

{

String orig\_text="";

for (int i = 0 ; i < cipher\_text.length() &&

i < key.length(); i++)

{

// converting in range 0-25 int x = (cipher\_text.charAt(i) -

key.charAt(i) + 26) %26;

// convert into alphabets(ASCII) x += 'A';

orig\_text+=(char)(x);

}

return orig\_text;

}

static String LowerToUpper(String s)

{

StringBuffer str =new StringBuffer(s); for(int i = 0; i < s.length(); i++)

{

if(Character.isLowerCase(s.charAt(i)))

{

str.setCharAt(i, Character.toUpperCase(s.charAt(i)));

}

}

s = str.toString(); return s;

}

public static void main(String[] args)

{

String Str = "save the king from tiger"; String Keyword = "attack";

String str = LowerToUpper(Str);

String keyword = LowerToUpper(Keyword); String key = generateKey(str, keyword); String cipher\_text = cipherText(str, key); System.out.println("Ciphertext : "

+ cipher\_text + "\n");

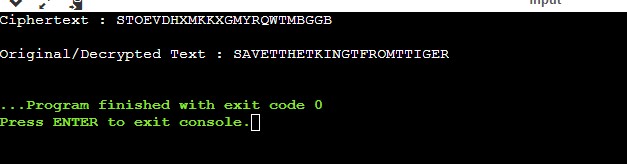
System.out.println("Original/Decrypted Text : "

+ originalText(cipher\_text, key));

}

}

Output-



**Python**

def generateKey(string, key): key = list(key)

if len(string) == len(key): return(key)

else:

for i in range(len(string) - len(key)):

key.append(key[i % len(key)]) return("" . join(key))

def cipherText(string, key): cipher\_text = []

for i in range(len(string)): x = (ord(string[i]) +

ord(key[i])) % 26

x += ord('A')

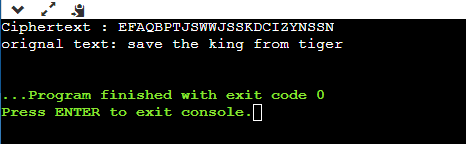
cipher\_text.append(chr(x)) return("" . join(cipher\_text))

def originalText(cipher\_text, key): orig\_text = []

if name == " main ":

string = "save the king from tiger" keyword = "attack"

key = generateKey(string, keyword) cipher\_text = cipherText(string,key) print("Ciphertext :", cipher\_text) print("orignal text:",string)





### Experiment 4

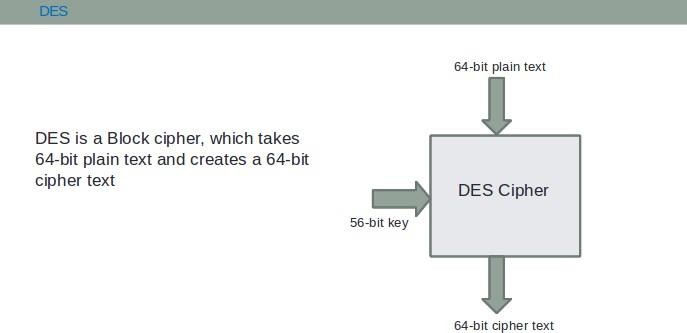
Aim:Encrypt long messages using various modes of operation using AES or DES.

### Theory:

**DES**

Data encryption standard (DES) has been found vulnerable against very powerful attacks and therefore, the popularity of DES has been found slightly on the 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. The basic idea is shown in the figure.



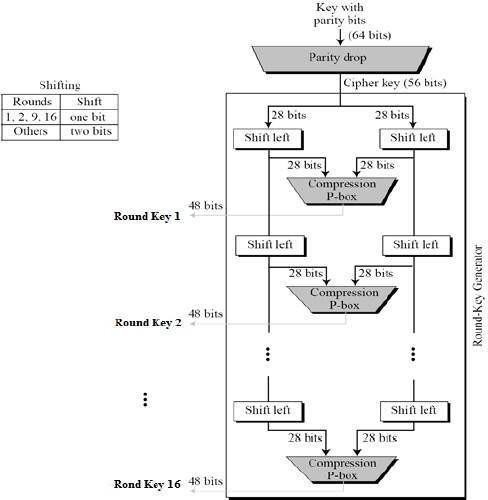
DES is based on the two fundamental attributes of cryptography: substitution (also called as confusion) and transposition (also called diffusion). DES consists of 16 steps, each of which is called as a round. Each round performs the steps of substitution and transposition. Let us now discuss the broad-level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

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

1. The initial permutation is performed on plain text.
2. Next, the initial permutation (IP) produces two halves of the permuted block; says Left Plain Text (LPT) and Right Plain Text (RPT).
3. Now each LPT and RPT the go through 16 rounds of encryption process.
4. In the end, LPT and RPT are rejoined and a Final Permutation (FP) is performed on the combined block
5. The result of this process produces 64 bit cipher text.

Key Generation

The round-key generator creates sixteen 48-bit keys out of a 56-bit cipher key. The process of key generation is depicted in the following illustration −



The logic for Parity drop, shifting, and Compression P-box is given in the DES description.

DES Analysis

The DES satisfies both the desired properties of block cipher. These two properties make cipher very strong.

* **Avalanche effect** − A small change in plaintext results in the very great change in the ciphertext.
* **Completeness** − Each bit of ciphertext depends on many bits of plaintext.

During the last few years, cryptanalysis have found some weaknesses in DES when key selected are weak keys. These keys shall be avoided.

DES has proved to be a very well designed block cipher. There have been no significant cryptanalytic attacks on DES other than exhaustive key search.

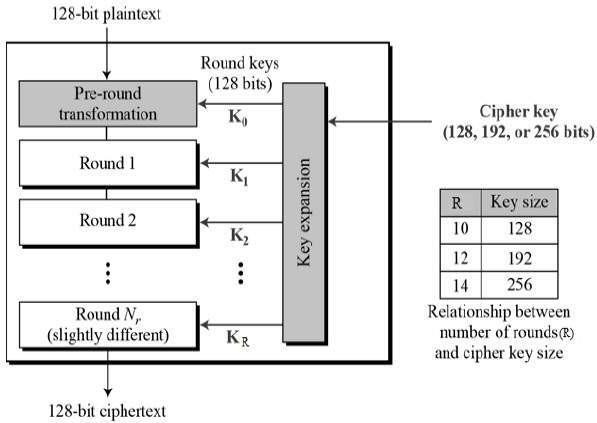
**AES**

AES is an iterative rather than Feistel cipher. It is based on ‘substitution–permutation network’. It comprises of a series of linked operations, some of which involve replacing inputs by specific outputs (substitutions) and others involve shuffling bits around (permutations).

Interestingly, AES performs all its computations on bytes rather than bits. Hence, AES treats the 128 bits of a plaintext block as 16 bytes. These 16 bytes are arranged in four columns and four rows for processing as a matrix −

Unlike DES, the number of rounds in AES is variable and depends on the length of the key. AES uses 10 rounds for 128-bit keys, 12 rounds for 192-bit keys and 14 rounds for 256-bit keys. Each of these rounds uses a different 128-bit round key, which is calculated from the original AES key.

The schematic of AES structure is given in the following illustration −



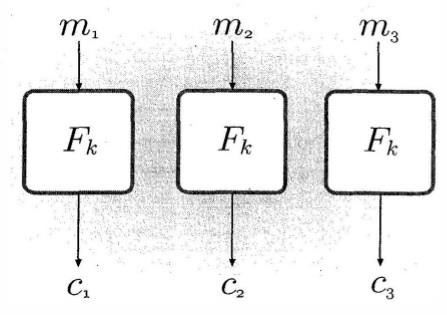
Encryption Process

According to the AES, the AES algorithm uses a 128-bit symmetric, or single-key, block cipher that encrypts and decrypts information. The AES encryption process creates ciphertext, which is an unreadable, effectively indecipherable conversion of plaintext data, the version of information that humans can read and understand. The output of the encryption process, the AES ciphertext, cannot be read until a secret AES key is used to decrypt it.

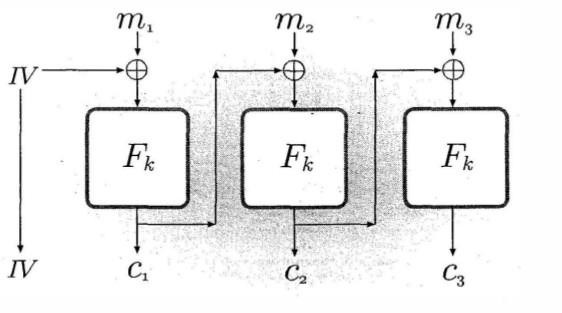
The encryption and decryption processes can use 128-, 192-, and 256-bit key lengths to convert plaintext into ciphertext and ciphertext into plaintext. These processes are known as encryption and decryption, respectively.

In AES-encrypted communications, a sender and recipient are given the same AES secret key, which is used to convert information into ciphertext as well as into readable plaintext. If this information were to be intercepted by a hacker, he or she wouldn’t be able to read it without the AES secret key, which, hopefully, would only be known by the users sending and receiving encrypted information.

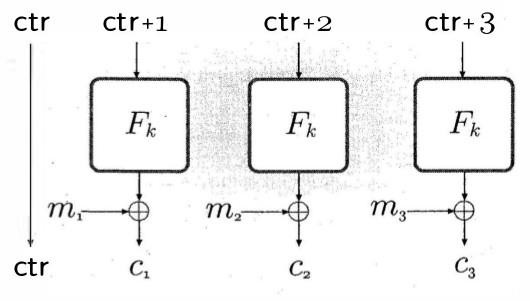
For a very brief theory of "Modes of Encryption" and their analysis, click here](docs/Modes-of- operation.pdf)



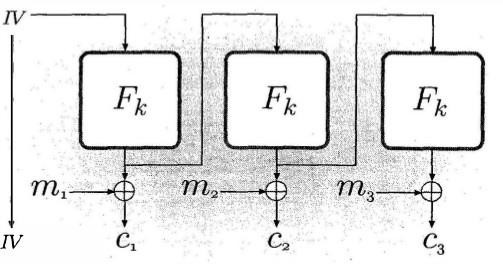
Electronic Code Book(ECB) mode



Cipher Block Chaining(CBC) mode



Contur mode



Output FeedBack mode

Decryption Process

The process of decryption of an AES ciphertext is similar to the encryption process in the reverse order. Each round consists of the four processes conducted in the reverse order −

* Add round key
* Mix columns
* Shift rows
* Byte substitution

Since sub-processes in each round are in reverse manner, unlike for a Feistel Cipher, the encryption and decryption algorithms need to be separately implemented, although they are very closely related.

AES Analysis

In present day cryptography, AES is widely adopted and supported in both hardware and software. Till date, no practical cryptanalytic attacks against AES have been discovered. Additionally, AES has built-in flexibility of key length, which allows a degree of ‘future- proofing’ against progress in the ability to perform exhaustive key searches.

However, just as for DES, the AES security is assured only if it is correctly implemented, and good key management is employed.

### Program:

# Python3 code for the above approach # Hexadecimal to binary conversion

def hex2bin(s):

mp = {'0': "0000",

'1': "0001",

'2': "0010",

'3': "0011",

'4': "0100",

'5': "0101",

'6': "0110",

'7': "0111",

'8': "1000",

'9': "1001",

'A': "1010",

'B': "1011",

'C': "1100",

'D': "1101",

'E': "1110",

'F': "1111"}

bin = ""

for i in range(len(s)):

bin = bin + mp[s[i]] return bin

# Binary to hexadecimal conversion

def bin2hex(s):

mp = {"0000": '0',

"0001": '1',

"0010": '2',

"0011": '3',

"0100": '4',

"0101": '5',

"0110": '6',

"0111": '7',

"1000": '8',

"1001": '9',

"1010": 'A',

"1011": 'B',

"1100": 'C',

"1101": 'D',

"1110": 'E',

"1111": 'F'}

hex = ""

for i in range(0, len(s), 4): ch = ""

ch = ch + s[i]

ch = ch + s[i + 1] ch = ch + s[i + 2] ch = ch + s[i + 3] hex = hex + mp[ch]

return hex

# Binary to decimal conversion

def bin2dec(binary):

binary1 = binary decimal, i, n = 0, 0, 0

while(binary != 0):

dec = binary % 10

decimal = decimal + dec \* pow(2, i) binary = binary//10

i += 1

return decimal

# Decimal to binary conversion

def dec2bin(num):

res = bin(num).replace("0b", "") if(len(res) % 4 != 0):

div = len(res) / 4 div = int(div)

counter = (4 \* (div + 1)) - len(res) for i in range(0, counter):

res = '0' + res

return res

# Permute function to rearrange the bits

def permute(k, arr, n):

permutation = "" for i in range(0, n):

permutation = permutation + k[arr[i] - 1] return permutation

# shifting the bits towards left by nth shifts

def shift\_left(k, nth\_shifts):

s = ""

for i in range(nth\_shifts):

for j in range(1, len(k)):

s = s + k[j] s = s + k[0]

k = s s = ""

return k

# calculating xow of two strings of binary number a and b

def xor(a, b):

ans = ""

for i in range(len(a)):

if a[i] == b[i]:

ans = ans + "0"

else: return ans

ans = ans + "1"

# Table of Position of 64 bits at initial level: Initial Permutation Table initial\_perm = [58, 50, 42, 34, 26, 18, 10, 2,

60, 52, 44, 36, 28, 20, 12, 4,

62, 54, 46, 38, 30, 22, 14, 6,

64, 56, 48, 40, 32, 24, 16, 8,

57, 49, 41, 33, 25, 17, 9, 1,

59, 51, 43, 35, 27, 19, 11, 3,

61, 53, 45, 37, 29, 21, 13, 5,

63, 55, 47, 39, 31, 23, 15, 7]

# Expansion D-box Table exp\_d = [32, 1, 2, 3, 4, 5, 4, 5,

6, 7, 8, 9, 8, 9, 10, 11,

12, 13, 12, 13, 14, 15, 16, 17,

16, 17, 18, 19, 20, 21, 20, 21,

22, 23, 24, 25, 24, 25, 26, 27,

28, 29, 28, 29, 30, 31, 32, 1]

# Straight Permutation Table per = [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]

# S-box Table

sbox = [[[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]]]

# Final Permutation Table

final\_perm = [40, 8, 48, 16, 56, 24, 64, 32,

39, 7, 47, 15, 55, 23, 63, 31,

38, 6, 46, 14, 54, 22, 62, 30,

37, 5, 45, 13, 53, 21, 61, 29,

36, 4, 44, 12, 52, 20, 60, 28,

35, 3, 43, 11, 51, 19, 59, 27,

34, 2, 42, 10, 50, 18, 58, 26,

33, 1, 41, 9, 49, 17, 57, 25]

def encrypt(pt, rkb, rk):

pt = hex2bin(pt)

# Initial Permutation

pt = permute(pt, initial\_perm, 64) print("After initial permutation", bin2hex(pt))

# Splitting left = pt[0:32]

right = pt[32:64]

for i in range(0, 16):

# Expansion D-box: Expanding the 32 bits data into 48 bits right\_expanded = permute(right, exp\_d, 48)

# XOR RoundKey[i] and right\_expanded xor\_x = xor(right\_expanded, rkb[i])

# S-boxex: substituting the value from s-box table by calculating row and column sbox\_str = ""

for j in range(0, 8):

row = bin2dec(int(xor\_x[j \* 6] + xor\_x[j \* 6 + 5])) col = bin2dec(

int(xor\_x[j \* 6 + 1] + xor\_x[j \* 6 + 2] + xor\_x[j \* 6 + 3] + xor\_x[j \* 6 + 4])) val = sbox[j][row][col]

sbox\_str = sbox\_str + dec2bin(val)

# Straight D-box: After substituting rearranging the bits sbox\_str = permute(sbox\_str, per, 32)

# XOR left and sbox\_str result = xor(left, sbox\_str) left = result

# Swapper if(i != 15):

left, right = right, left print("Round ", i + 1, " ", bin2hex(left),

" ", bin2hex(right), " ", rk[i])

# Combination combine = left + right

# Final permutation: final rearranging of bits to get cipher text cipher\_text = permute(combine, final\_perm, 64)

return cipher\_text

pt = "123456ABCD132536" key = "AABB09182736CCDD"

# Key generation # --hex to binary

key = hex2bin(key)

# --parity bit drop table

keyp = [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]

# getting 56 bit key from 64 bit using the parity bits key = permute(key, keyp, 56)

# Number of bit shifts shift\_table = [1, 1, 2, 2,

2, 2, 2, 2,

1, 2, 2, 2,

2, 2, 2, 1]

# Key- Compression Table : Compression of key from 56 bits to 48 bits key\_comp = [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]

# Splitting

left = key[0:28] # rkb for RoundKeys in binary

right = key[28:56] # rk for RoundKeys in hexadecimal

rkb = []

rk = []

for i in range(0, 16):

# Shifting the bits by nth shifts by checking from shift table left = shift\_left(left, shift\_table[i])

right = shift\_left(right, shift\_table[i])

# Combination of left and right string combine\_str = left + right

# Compression of key from 56 to 48 bits round\_key = permute(combine\_str, key\_comp, 48)

rkb.append(round\_key) rk.append(bin2hex(round\_key))

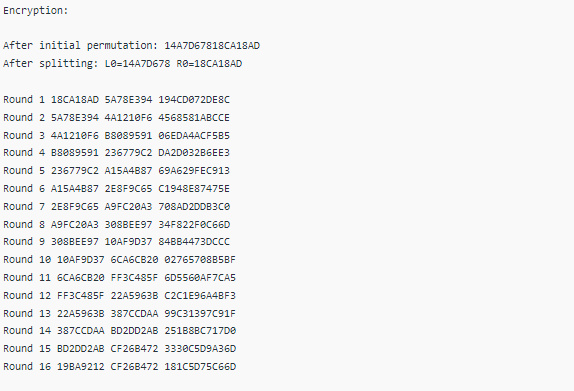
print("Encryption")

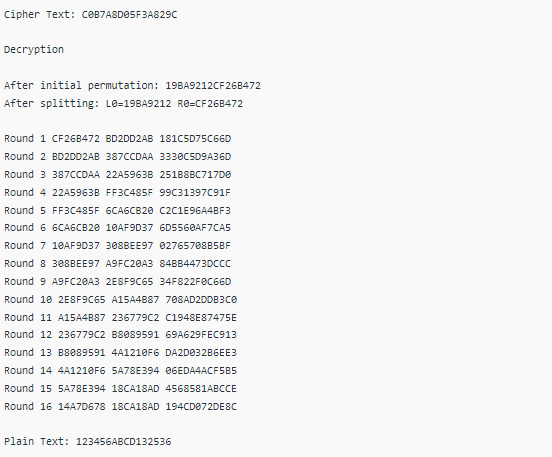
cipher\_text = bin2hex(encrypt(pt, rkb, rk)) print("Cipher Text : ", cipher\_text)

print("Decryption") rkb\_rev = rkb[::-1] rk\_rev = rk[::-1]

text = bin2hex(encrypt(cipher\_text, rkb\_rev, rk\_rev)) print("Plain Text : ", text)

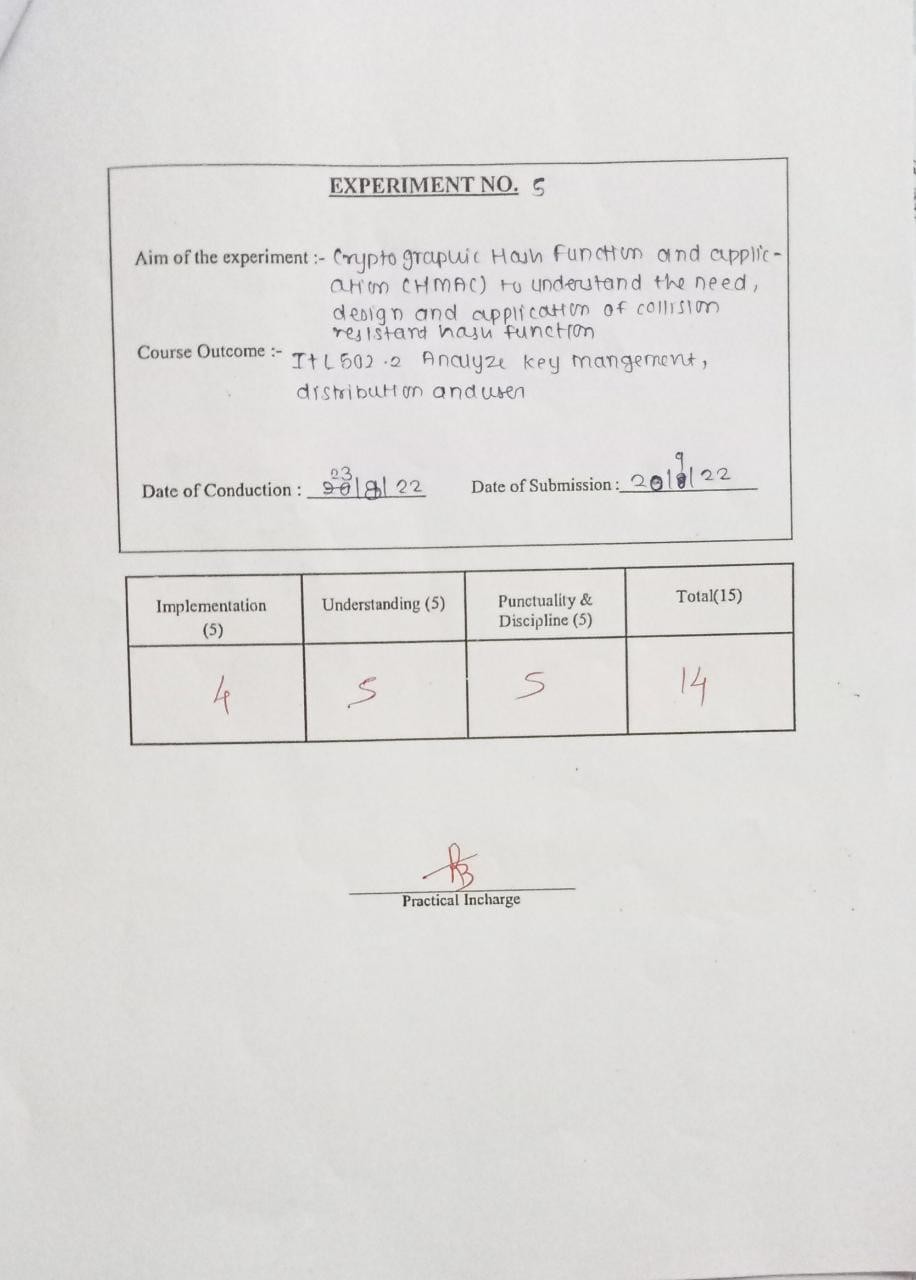
### Output:





**Conclusion:**

Successfully encrypt long messages using various modes of operation using AES or DES using python



## EXPERIMENT NO: 05

**Aim:** Cryptographic Hash Functions and Applications(HMAC): to

understand the need, design and applications ofcollision resistant hash functions

### Theory:

MD5 (Message Digest algorithm 5) is a widely used cryptographic hash function with a 128bit hash value. An MD5 hash is typically expressed as a 32 digit hexadecimal number. MD5 processes a variable length message into a fixed length output of 128 bits. The input message is broken up into chunks of 512 bit blocks (sixteen 32bit little endian integers) ; The message is padded so that its length is divisible by 512. The padding works as follows: first a single bit, 1, is appended to the end of the message. This is followed by as many zeros as are required to bring the length of the message up to 64 bits less than a multiple of 512. The remaining bits are filled up with a 64bit integer representing the length of the original message, in bits.

The main MD5 algorithm operates on a 128bit state, divided into four 32bit words, denoted *A*, *B*, *C* and *D*. These are initialized to certain fixed constants. The main algorithm then operates oneach 512bit message block in turn, each block modifying the state. The processing of a message block consists of four similar stages, termed *rounds*; each round is composed of 16 similar operations based on a nonlinear function *F*, modular addition, and left rotation.

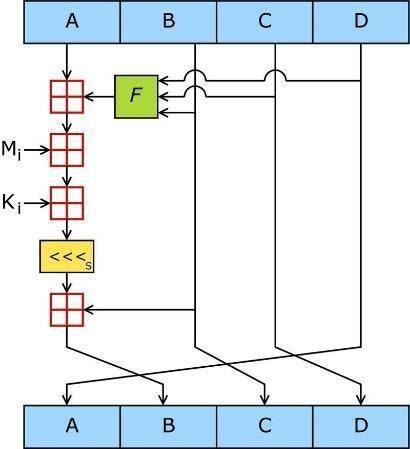


Figure 1: One MD5 operation. MD5 consists of 64 of these operations, grouped in four rounds of 16 operations. *F* is a nonlinear function; one function is used in each round. *Mi* denotes a 32bit block of the message input, and *Ki* denotes a 32bit constant, different for each operation.

Figure 1 illustrates one operation within a round. There are four possible functions *F*; a different one is used in each round:

|  |  |  |  |
| --- | --- | --- | --- |
| *F* ( *X*, *Y* , *Z* )=( *X Y* ) | ( | *X* | *Z* ) |
| *G* ( *X*, *Y* , *Z* )=( *X Z* ) | (*Y* | *Z* ) |  |
| *I* ( *X*, *Y* , *Z* )= *Y* ( *X* | *Z* ) |  |  |

, , ,denote the XOR, AND, OR and NOToperations respectively.

**Algorithm:**

**1.Append Padding Bits**

The message is "padded" (extended) so that its length (in bits) is congruent to 448, modulo 512. That is, the message is extended so that it is just 64 bits shy of being a multiple of 512 bits long. Padding is always performed, even if the length of the message is already congruent to 448, modulo

512. Padding is performed as follows: a single "1" bit is appended to the message, and then "0" bits are appended so that the length in bits of the padded message becomes congruent to 448, modulo 512. In all, at least one bit and at most 512 bits are appended.

#### Append Length

A 64bit representation of b (the length of the message before the padding bits were added) is appended to the result of the previous step. In the unlikely event that b is greater than 2^64, then only the loworder 64 bits of b are used. (These bits are appended as two 32bit words and appended loworder word first in accordance with the previous conventions.) At this point the resulting message (after padding with bits and with b) has a length that is an exact multiple of512 bits. Equivalently, this message has a length that is an exact multiple of 16 (32 bit) words. Let M[0 ... N1] denote the words of the resulting message, where N is a multiple of 16.

#### Initialize MD Buffer

A fourword buffer (A,B,C,D) is used to compute the message digest. Here each of A, B, C, D is a 32bit register. These registers are initialized to the following values in hexadecimal, loworder bytes first):

#### Process Message in 16Word Blocks

We first define four auxiliary functions that each take as input three 32bit words and produce as output one 32bit word.

#### Output

The message digest produced as output is A, B, C, D. That is, we begin with the low order byte of A, and end with the highorder byte of D.

### Code:-

// import required classes and package if any import java.security.MessageDigest;

import java.util.Scanner;

// create class MessageDigestExample to understand the use of MessageDigest class public class MessageDigestExample {

// main() method start

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

// create an instance of Scanner class Scanner sc = new Scanner(System.in);

System.out.println("Enter the message of any arbitrary length:"); String msg = sc.nextLine();

// close Scanner class sc.close();

//create an instance of the MessageDigest by using the getInstance() method with the MD5 algorithm MessageDigest obj = MessageDigest.getInstance("MD5");

//use update() method for passing data to the created MessageDigest Object obj.update(msg.getBytes());

//use the digest() method for computing the message digest

byte[] byteArray = obj.digest(); System.out.println(byteArray);

//convert the byte array in to Hex String format StringBuffer hexData = new StringBuffer(); for (int i = 0; i < byteArray.length; i++) {

hexData.append(Integer.toHexString(0xFF & byteArray[i]));

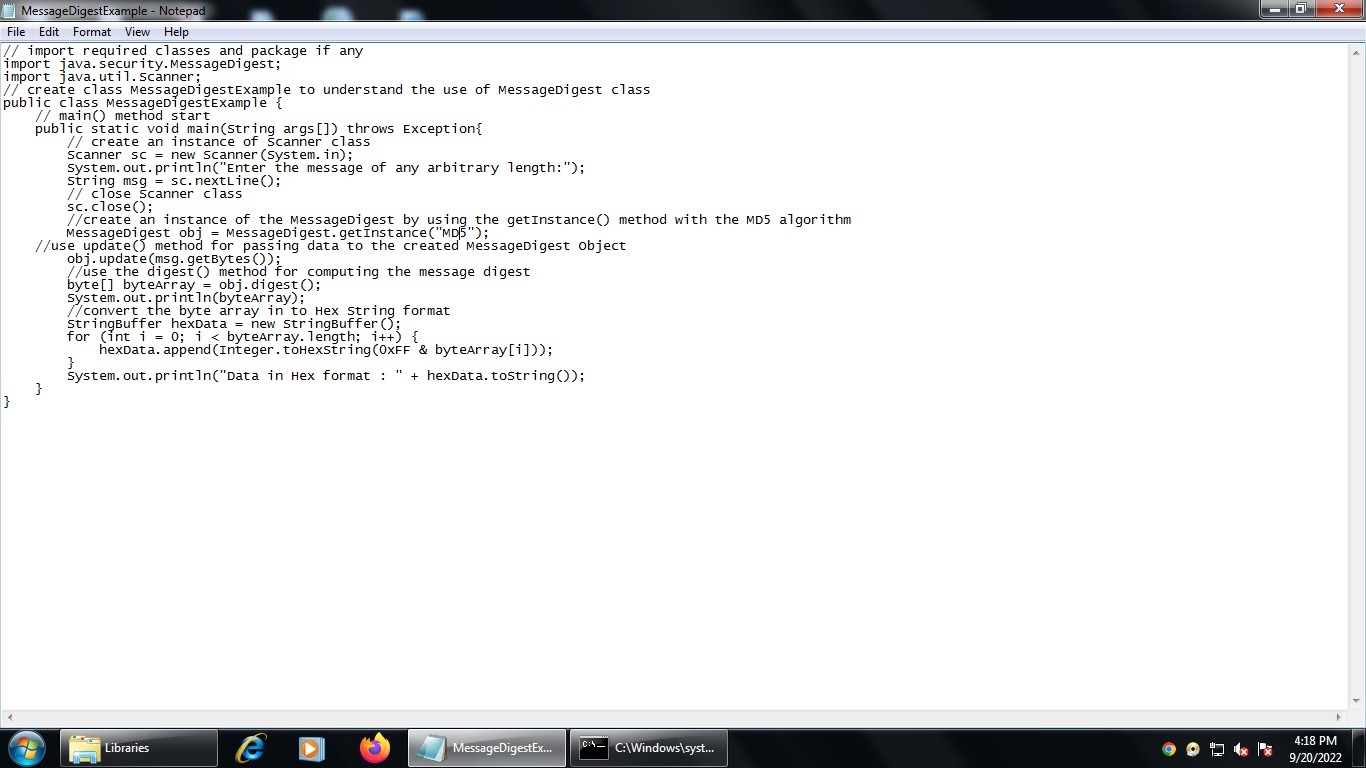
}

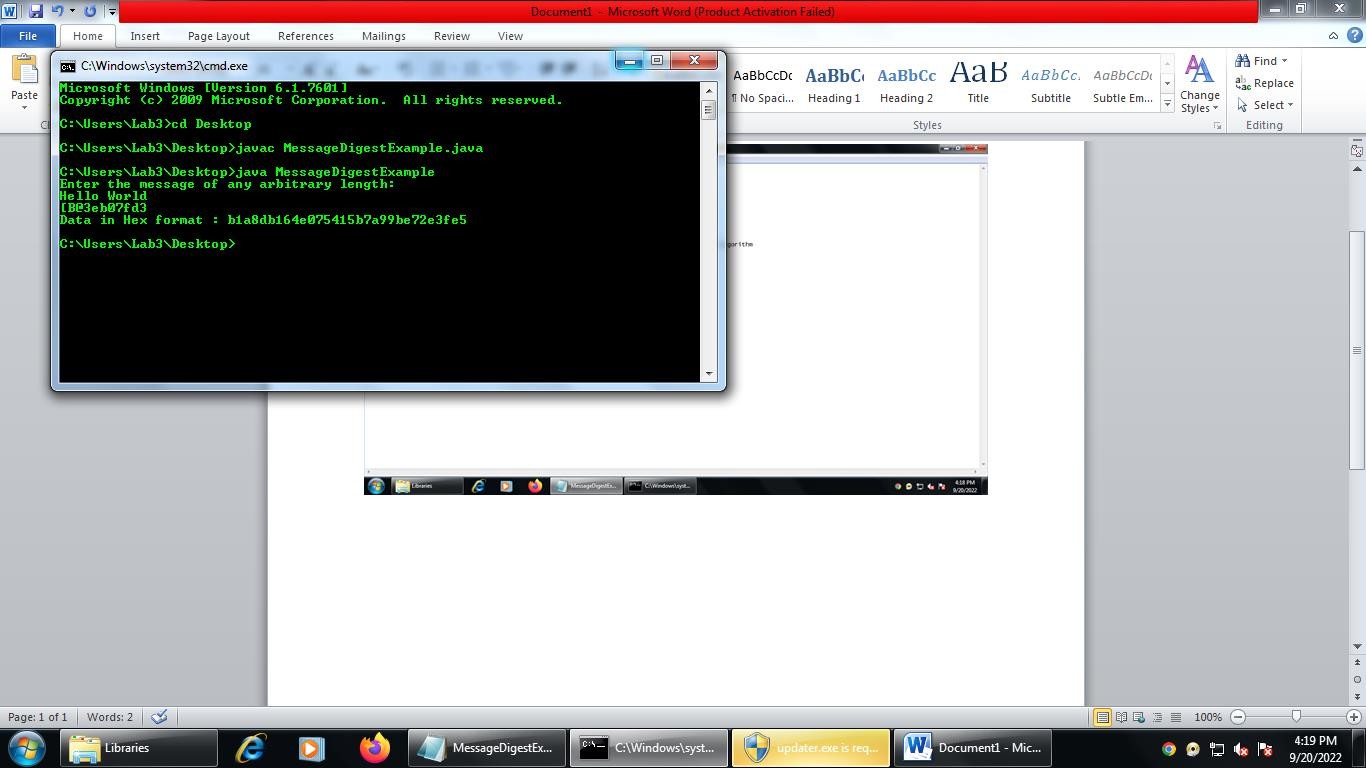
System.out.println("Data in Hex format : " + hexData.toString());

}

}

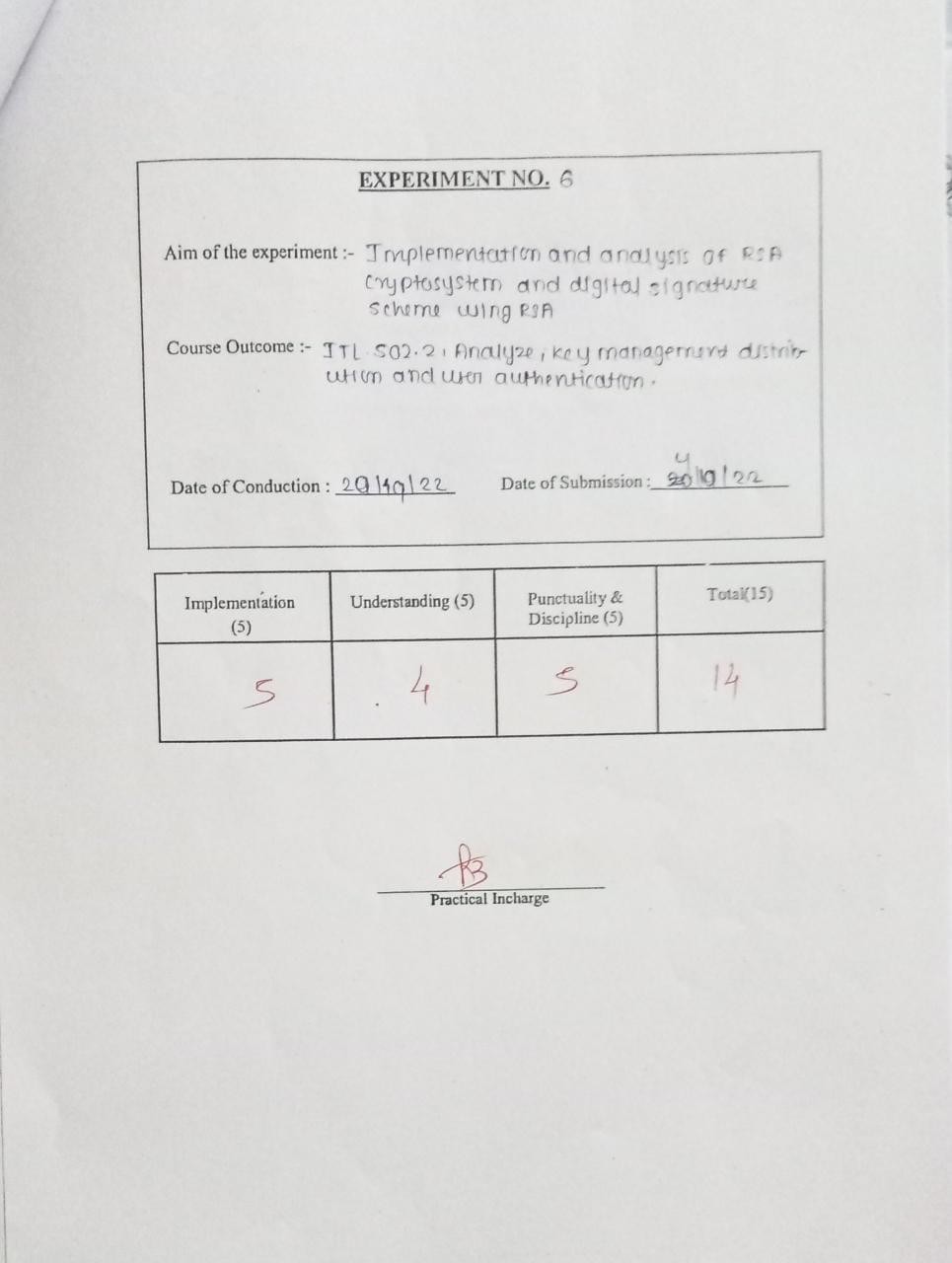
### Output





**Conclusion:**

Cryptographic hash functions implemented successfully



**Aim:**

# Experiment 6

Implementation and analysis of RSA cryptosystem and Digital signature scheme using RSA.

**Theory:**

RSA algorithm is asymmetric cryptography algorithm. Asymmetric actually means that it works on two different keys i.e. Public Key and Private Key. As the name describes that the Public Key is given to everyone and Private key is kept private.

Algorithm

The RSA algorithm holds the following features −

* RSA algorithm is a popular exponentiation in a finite field over integers including prime numbers.
* The integers used by this method are sufficiently large making it difficult to solve.
* There are two sets of keys in this algorithm: private key and public key. You will have to go through the following steps to work on RSA algorithm −

Step 1: Generate the RSA modulus

The initial procedure begins with selection of two prime numbers namely p and q, and then calculating their product N, as shown −

N=p\*q Here, let N be the specified large number.

Step 2: Derived Number (e)

Consider number e as a derived number which should be greater than 1 and less than (p-1) and (q-1). The primary condition will be that there should be no common factor of (p-1) and (q-1) except 1

Step 3: Public key

The specified pair of numbers **n** and **e** forms the RSA public key and it is made public. Step 4: Private Key

Private Key **d** is calculated from the numbers p, q and e. The mathematical relationship between the numbers is as follows −

ed = 1 mod (p-1) (q-1)

The above formula is the basic formula for Extended Euclidean Algorithm, which takes p and q as the input parameters.

Encryption Formula

Consider a sender who sends the plain text message to someone whose public key is **(n,e).** To encrypt the plain text message in the given scenario, use the following syntax −

C = Pe mod n

Decryption Formula

The decryption process is very straightforward and includes analytics for calculation in a systematic approach. Considering receiver **C** has the private key **d**, the result modulus will be calculated as −

Plaintext = Cd mod n

**Program:**

# Function to find gcd # of two numbers

def euclid(m, n):

if n == 0:

return m

else:

r = m % n

return euclid(n, r)

# Program to find

# Multiplicative inverse def exteuclid(a, b):

r1 = a r2 = b

s1 = int(1) s2 = int(0) t1 = int(0) t2 = int(1)

while r2 > 0:

q = r1//r2

r = r1-q \* r2 r1 = r2

r2 = r

s = s1-q \* s2 s1 = s2

s2 = s

t = t1-q \* t2 t1 = t2

t2 = t

if t1 < 0:

t1 = t1 % a

return (r1, t1)

# Enter two large prime # numbers p and q

p = 823

q = 953

n = p \* q

Pn = (p-1)\*(q-1)

# Generate encryption key # in range 1<e<Pn

key = []

for i in range(2, Pn):

gcd = euclid(Pn, i) if gcd == 1:

key.append(i)

# Select an encryption key # from the above list

e = int(313)

# Obtain inverse of

# encryption key in Z\_Pn r, d = exteuclid(Pn, e)

if r == 1:

d = int(d)

print("decryption key is: ", d)

else:

print("Multiplicative inverse for\ the given encryption key does not \

exist. Choose a different encryption key ")

# Enter the message to be sent M = 19070

# Signature is created by Alice S = (M\*\*d) % n

# Alice sends M and S both to Bob

# Bob generates message M1 using the # signature S, Alice's public key e

# and product n.

M1 = (S\*\*e) % n

# If M = M1 only then Bob accepts # the message sent by Alice.

if M == M1:

print("As M = M1, Accept the\ message sent by Alice")

else:

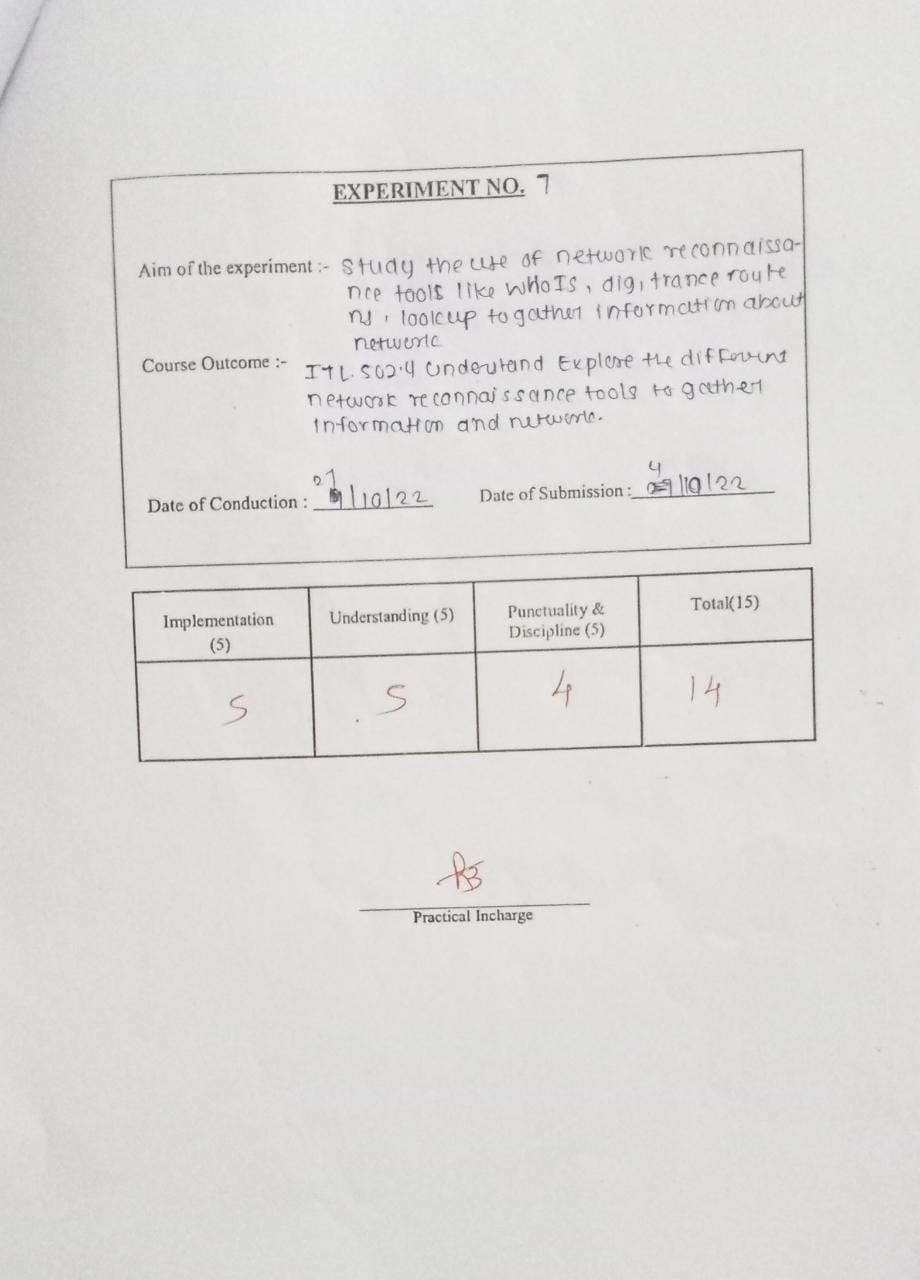
print("As M not equal to M1,\ Do not accept the message\ sent by Alice ")

Output:



**Conclusion:**

RSA algorithm implemented successfully.



# Experiment 7

### Aim:

Study the use of network reconnaissance tools like WHOIS, dig,traceroute, nslookup to gather information about networks anddomain registrars.

### Theory:

1. **Whois** - whois searches for an object in a WHOIS database. WHOIS is a query and response protocol that is widely used for querying databases that store the registered users of an Internet resource, such as a domain name or an IP address block, but is also used for a wider range of other information. Most modern versions of whois try to guess the right server to ask for the specified object. If no guess can be made, whois will connect to whois.networksolutions.com for NIC handles or whois.arin.net for IPv4 addresses and network names.

Examples:

* + Obtaining the domain WHOIS record for computersolutions.com
  + WHOIS record by IP querying
  + Querying WHOIS in google search engine

1. **Dig** - Dig is a networking tool that can query DNS servers for information. It can be very helpful for diagnosing problems with domain pointing and is a

14

good way to verify that your configuration is working. The most basic way to use dig is to specify the domain we wish to query: dig example.com

1. Traceroute - Traceroute prints the route that packets take to a network host. Traceroute utility uses the TTL field in the IP header to achieve its operation. For users who are new to TTL field, this field describes how much hops a particular packet will take while traveling on network. So, this effectively outlines the lifetime of the packet on network. This field is usually set to 32 or 64. Each

time the packet is held on an intermediate router, it decreases the TTL value by 1.

When a router finds the TTL value of 1 in a received packet then that packet is not forwarded but instead discarded. After discarding the packet, router sends an ICMP error message of ―Time exceededǁ back to the source from

where packet generated. The ICMP packet that is sent back contains the IP address of the router. So now it can be easily understood that traceroute operates by sending packets with TTL value starting from 1 and then incrementing by one each time. Each time a router receives the packet, it checks the TTL field, if TTL field is 1 then it discards the packet and sends the ICMP error packet containing its IP address and this is what traceroute requires. So traceroute incrementally fetches the IP of all the routers between the source and the destination.

Example: traceroute example.com

traceroute to example.com (64.13.192.208), 64 hops max, 40 byte

packets 1 72.10.62.1 (72.10.62.1) 1.000 ms 0.739 ms 0.702 ms

2 10.101.248.1 (10.101.248.1) 0.683 ms 0.385 ms 0.315 ms

3 10.104.65.161 (10.104.65.161) 0.791 ms 0.703 ms 0.686 ms

4 10.104.65.161 (10.104.65.161) 0.791 ms 0.703 ms 0.686 ms

5 10.0.10.33 (10.0.10.33) 2.652 ms 2.260 ms 5.353 ms

6 acmkokeaig.gs01.gridserver.com (64.13.192.208) 3.384 ms 8.001 ms 2.439ms

1. **Nslookup** - The nslookup command is used to query internet name servers interactively for information. nslookup, which stands for "name server lookup", is a

useful tool for finding out information about a named domain. By default, nslookup will

translate a domain name to an IP address (or vice versa). For instance, to find out what

15

the IP address of microsoft.com is, you could run the command: nslookup microsoft.com

Server: 8.8.8.8

Address: 8.8.8.8#53 Non-authoritative answer: Name: microsoft.com

Address: 134.170.185.46

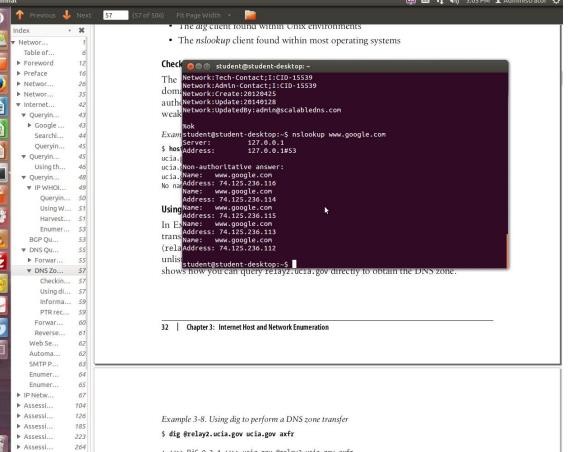
Name: microsoft.com

Address: 134.170.188.221

Here, 8.8.8.8 is the address of our system's Domain Name Server. This is the server our

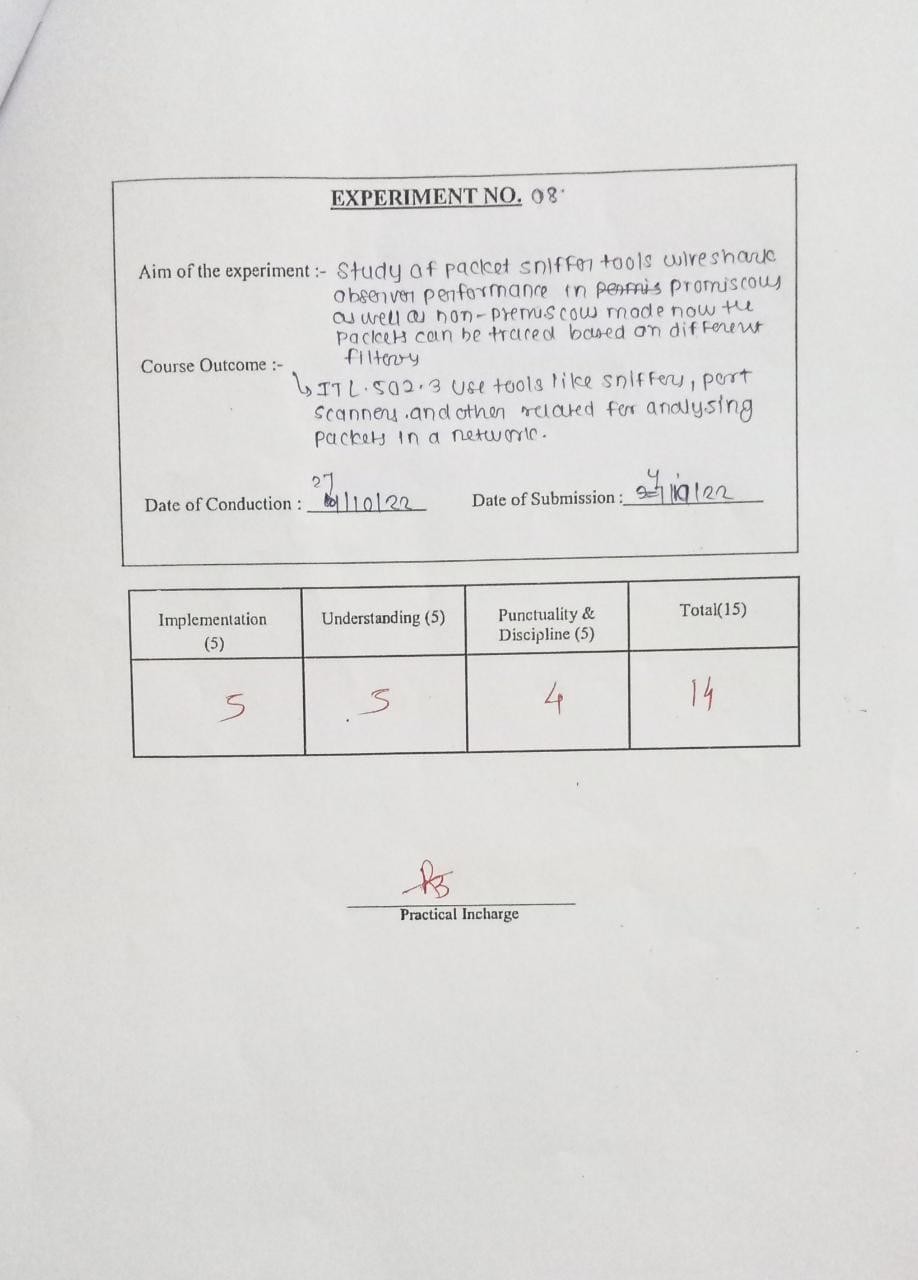
systemis configured to use to translate domain names into IP addresses. "#53" indicates that we are communicating with it on port 53, which is the standard port number domain name servers use to accept queries. Below this, we have our lookup information for microsoft.com. Our name server returned two entries, 134.170.185.46 and 134.170.188.221. This indicates that microsoft.com uses a round robin setup to distribute server load. When you accessmicrsoft.com, you may be directed to either of these servers and your packets will be routed to the correct destination. You can see that we have received a "Nonauthoritative answer" to our query. An answer is "authoritative" only if our DNS has the complete zone file information for the domain in question. More often, our DNS will have a cache of information representing the last authoritative answer it received when it made a similar query, this information is passed on to you, but the server qualifies it as "nonauthoritative": the information was recently received from an authoritative source, but the DNS server is not itself that authority.

### Commands



**Conclusion**

Successfully studied the use of network reconnaissance tools like WHOIS, dig,traceroute,nslookup to gather information about networks anddomain registrars.



**Aim:**

# Experiment 8

Study of packet sniffer tools wireshark: - a. Observer performance in promiscuous as well as non-promiscuous mode. b. Show the packets can be traced based on different filters

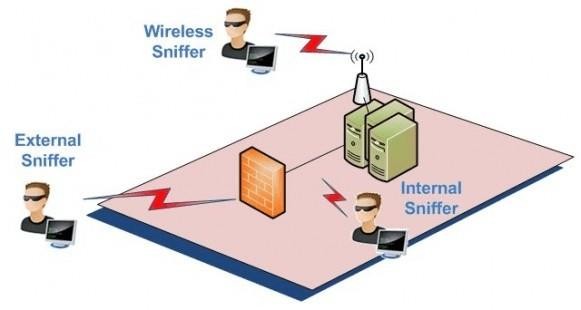
**Theory:**

Computers communicate by broadcasting messages on a network using IP addresses. Once a message has been sent on a network, the recipient computer with the matching IP address responds with its MAC address.

**Network sniffing is the process of intercepting data packets sent over a network.**This can be done by the specialized software program or hardware equipment. Sniffing can be used to;

* Capture sensitive data such as login credentials
* Eavesdrop on chat messages
* Capture files have been transmitted over a network The following are protocols that are vulnerable to sniffing
* Telnet
* Rlogin
* HTTP
* SMTP
* NNTP
* POP
* FTP
* IMAP

The above protocols are vulnerable if login details are sent in plain text



Wireshark is a free open source tool that analyzes network traffic in real-time for Windows, Mac, Unix, and Linux systems. It captures data packets passing through a network interface (such as Ethernet, LAN, or SDRs) and translates that data into valuable information for IT professionals and cybersecurity teams.

Wireshark is a type of packet sniffer (also known as a network protocol analyzer, protocol analyzer, and network analyzer). Packet sniffers intercept network traffic to understand the activity being processed and harvest useful insights.

Wireshark (formerly known as ethereal) offers a series of different display filters to transform each captured packet into a readable format. This allows users to identify the cause of network security issues and even discover potential cybercriminal activity.

When a packet sniffer is used in 'promiscuous mode' users can analyze network traffic regardless of its destination - like a fly on a wall watching office activity. While this empowers IT professionals to perform a quick and thorough diagnosis of network security, in the wrong hands, Wireshark could be used for cyberattack reconnaissance campaigns.

Because you can download Wireshark for free, cybercriminals have liberal access to it, so it's best security practice to assume the software is currently being used with hostile intentions. Luckily, there are some security measures you can implement to protect against network sniffing.

What is Wireshark Used For?

Packet analysis software like Wireshark is used by entities that must remain informed about the state of security of their network, as such, the software is commonly used by governments, schools, and technology businesses.

Common Wireshark use cases include:

Identify the cause of a slow internet connection Investigating lost data packets

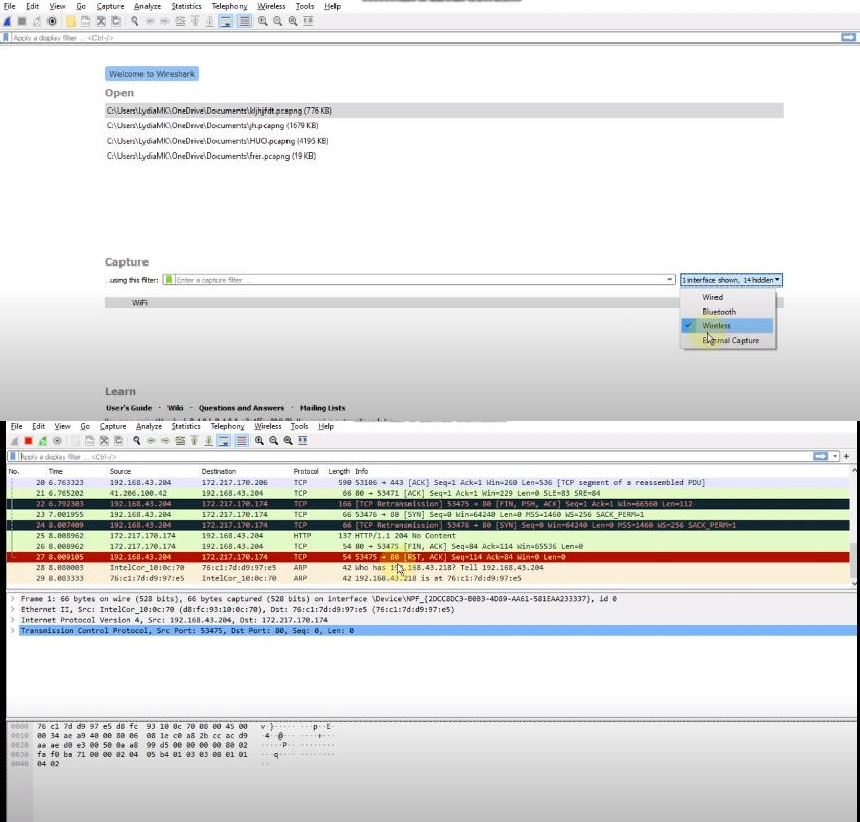
Troubleshooting latency issues Detecting malicious network activity Identify unauthorized data exfiltration Analyzing bandwidth usage

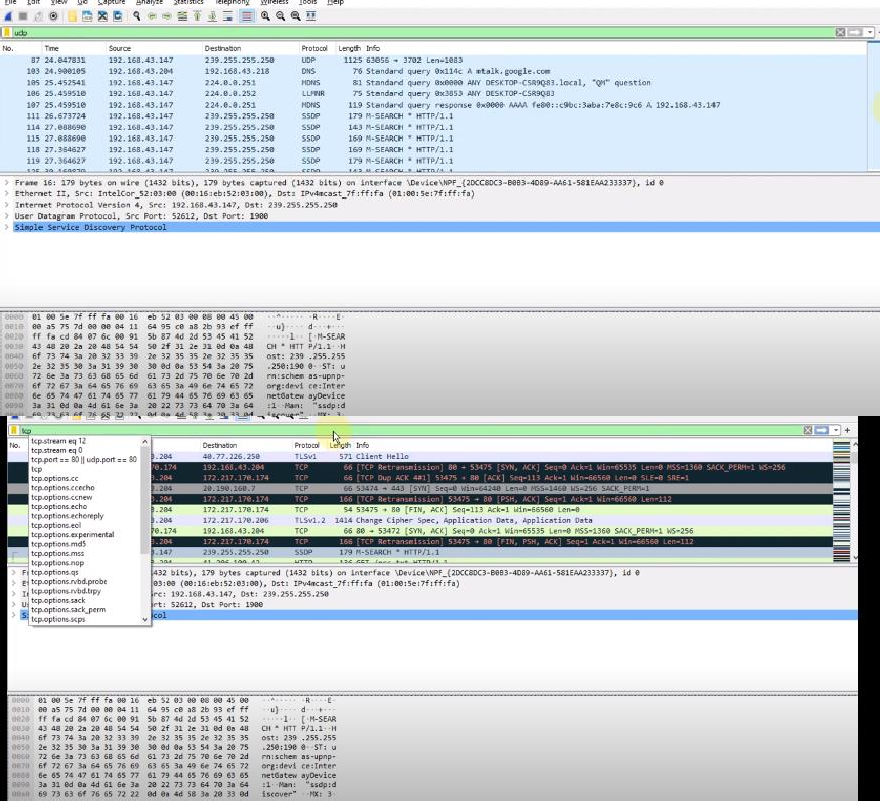
Tracing voice over Internet (VoIP) calls over the network Intercepting Man-in-the-Middle (MITM) attacks

Wireshark makes all of the above use cases possible by rendering and translating traffic into readable formats - saving users the frustrations of having to translate binary information manually. All of this is done in real-time so that detected issues can be rapidly addressed before they develop into a service outage, or worse, a data breach.

**Procedure and Screen shots**: <https://www.wireshark.org/download.html>

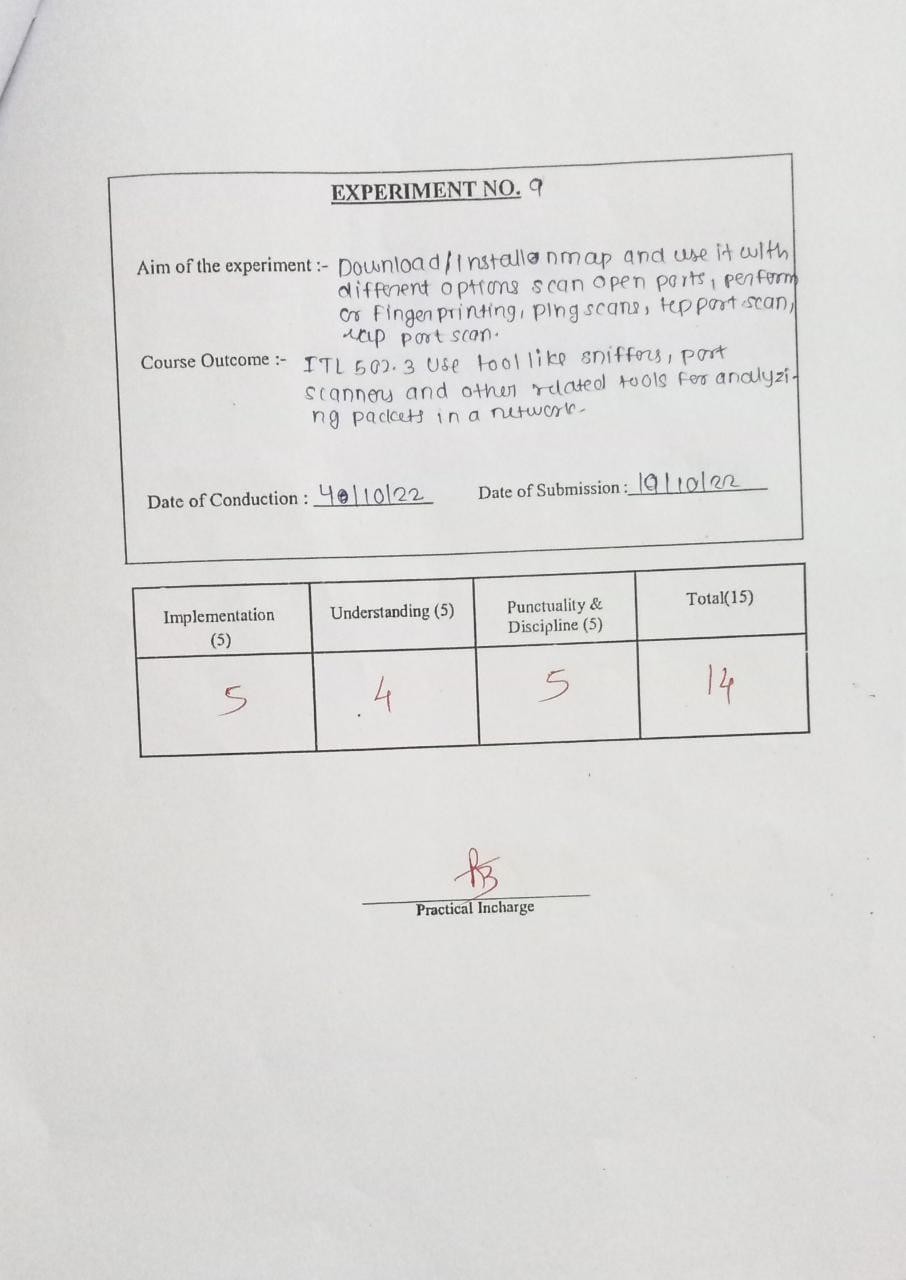
Output:





### Conclusion:

Successfully Studied the packet sniffer tools wireshark: - a. Observer performance in promiscuousas well as non-promiscuous mode. b. Show the packets can be traced based on different filters



**Aim:**

# Experiment 9

Download, install nmap and use it with different options to scan open ports, perform OS fingerprinting, ping scan, tcp port scan, edp port scan, etc.

**Theory:**

Nmap, short for Network Mapper, is a free, open-source tool for vulnerability scanning and network discovery. Network administrators use Nmap to identify what devices are running on their systems, discovering hosts that are available and the services they offer, finding open ports and detecting security risks.

Nmap can be used to monitor single hosts as well as vast networks that encompass hundreds of thousands of devices and multitudes of subnets.

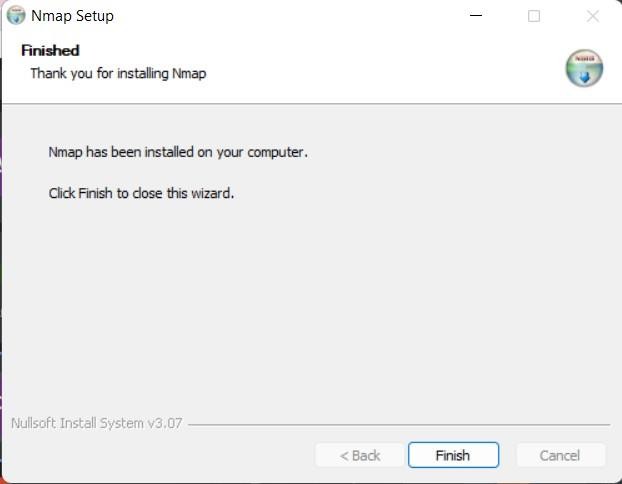
Though Nmap has evolved over the years and is extremely flexible, at heart it's a port-scan tool, gathering information by sending raw packets to system ports. It listens for responses and determines whether ports are open, closed or filtered in some way by, for example, a firewall. Other terms used for port scanning include port discovery or enumeration.

Port scanning

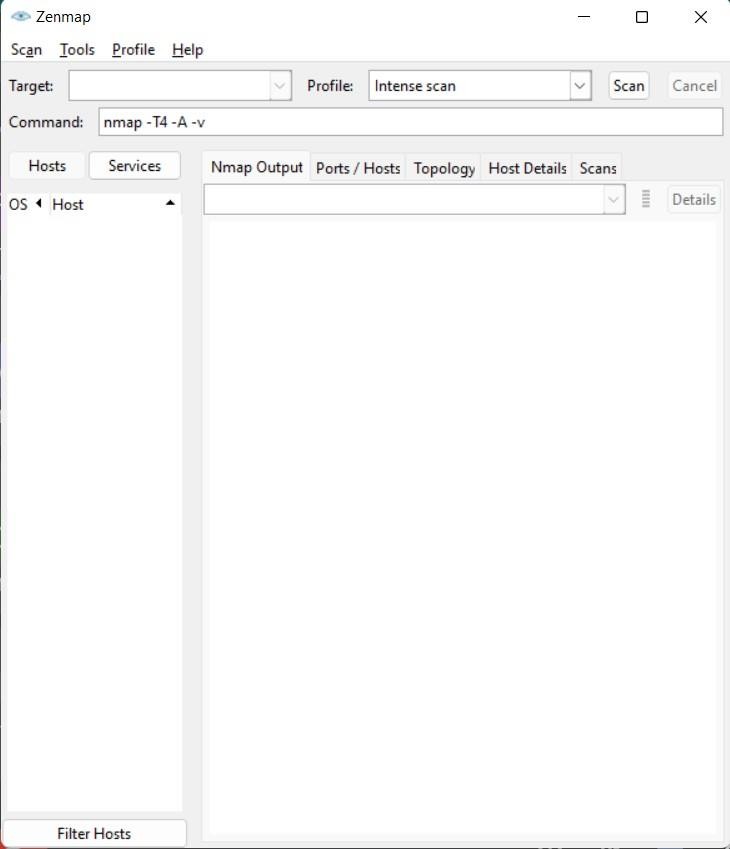
The packets that Nmap sends out return with IP addresses and a wealth of other data, allowing you to identify all sorts of network attributes, giving you a profile or map of the network and allowing you to create a hardware and software inventory.

Different protocols use different types of packet structures. Nmap employs transport layer protocols including TCP (Transmission Control Protocol), UDP (User Datagram Protocol), and SCTP (Stream Control Transmission Protocol), as well as supporting protocols like ICMP (Internet Control Message Protocol), used to send error messages.

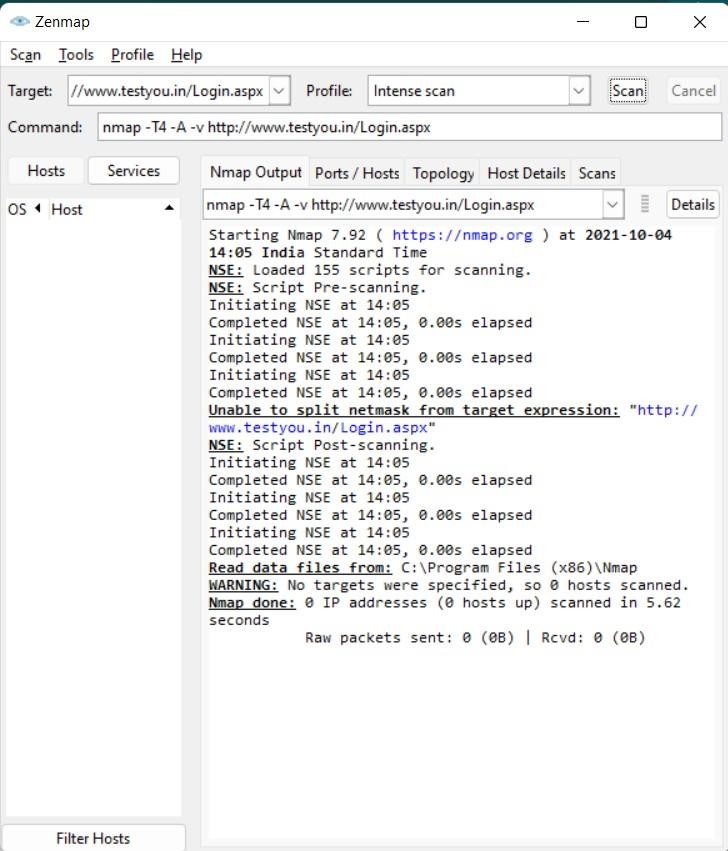
The various protocols serve different purposes and system ports. For example, the low resource overhead of UDP is suited for real-time streaming video, where you sacrifice some lost packets in return for speed, while non-real time streaming videos in YouTube are buffered and use the slower, albeit more reliable TCP. **Steps to Follow:**



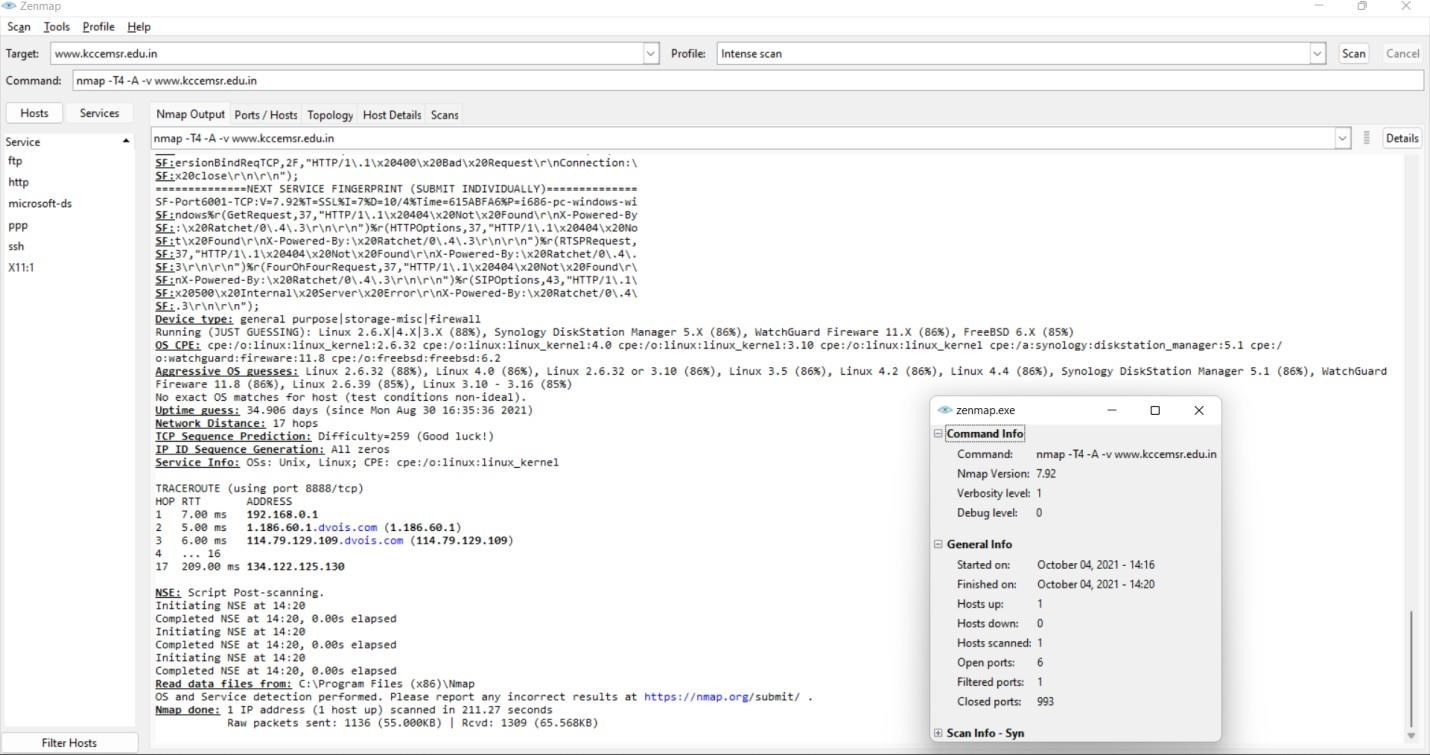
Loading Screen:

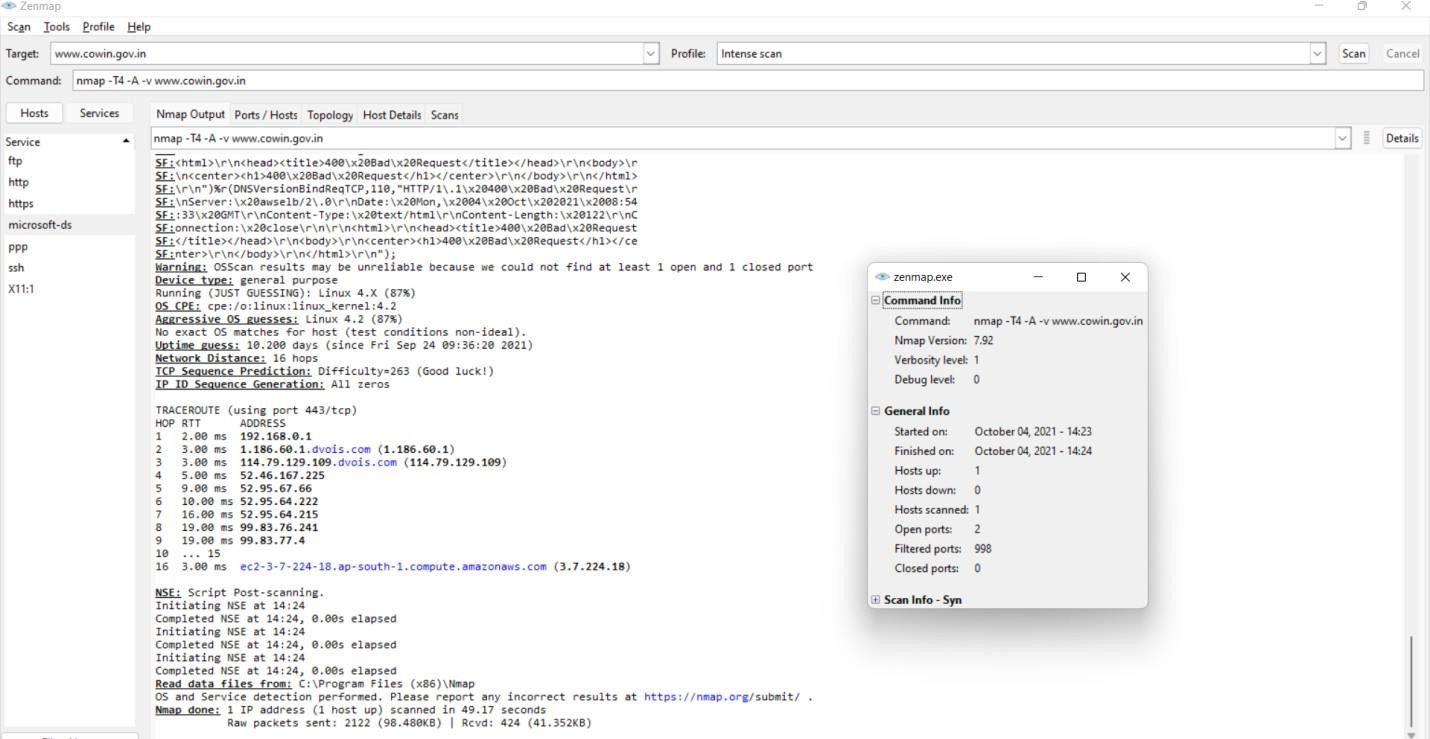


Put the target link in the target and then select intense scan and then press the scan button.



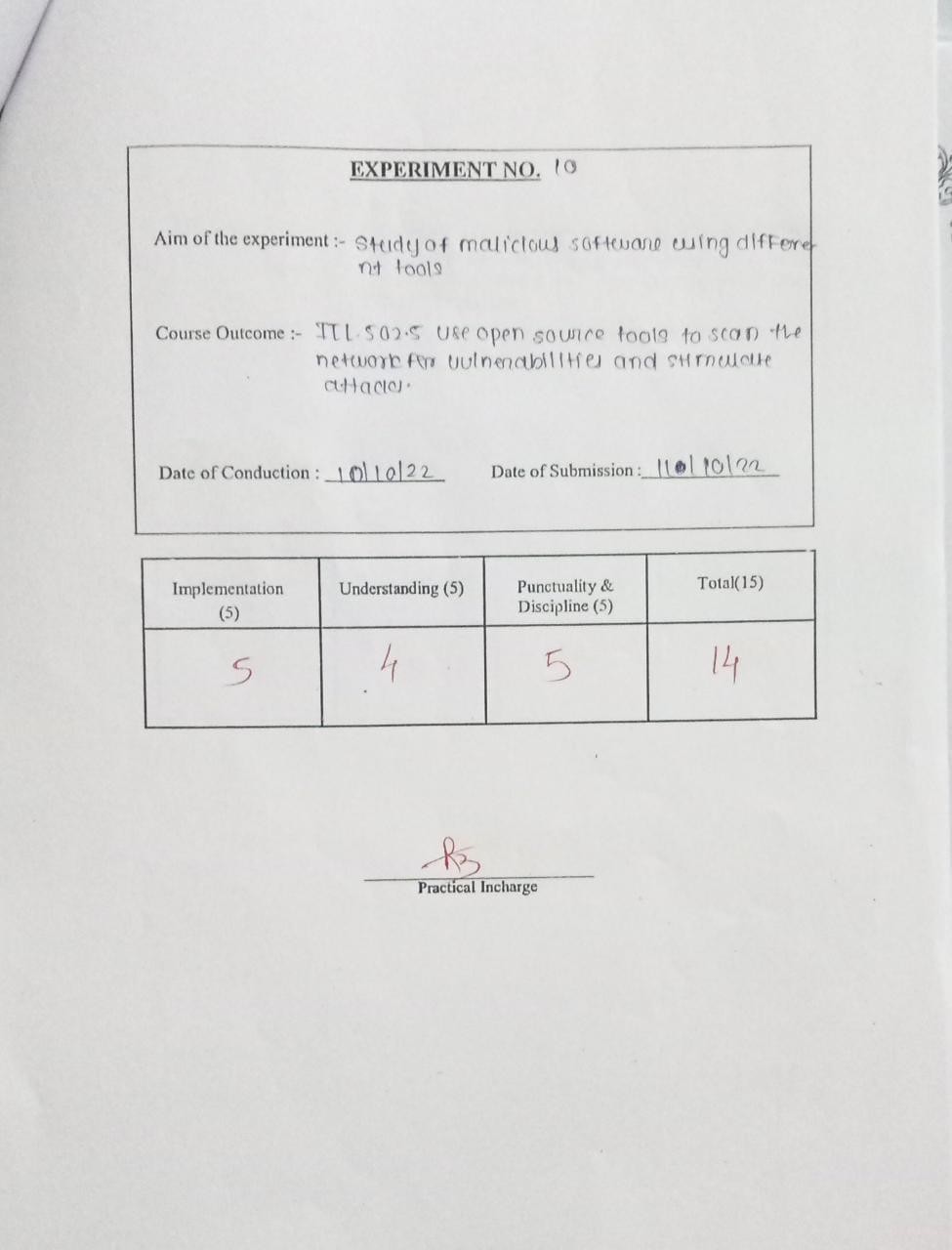
For college web site, [www.kccemsr.edu.in](http://www.kccemsr.edu.in/)



For site [www.covin.gov.in](http://www.covin.gov.in/)

### Conclusion:

Download, install nmap and use it with different options to scan open ports, perform OS fingerprinting, ping scan, tcp port scan,udp port scan has been done with proper installation steps and screen shots for the same.



**Aim:**

# Experiment: 10

Study of malicious software using different tools:

1. Keylogger attack using a keylogger tool.
2. Simulate DOS attack using Hping or other tools
3. Use the NESSUS/ISO Kali Linux tool to scan the network for vulnerabilities.

**Theory:**

Nessus is one of the most popular and capable vulnerability scanners, particularly for UNIX systems. It was initially free and open source, but they [closed the source code](http://www.linux.com/articles/48745) in 2005 and [removed the free"Registered Feed" version](http://sectools.org/stf/nessus_feed_letter.pdf) in 2008. It now costs $1,200 per year, which still beats many of its competitors. A free “Home Feed” is also available, though it is limited and only licensed for home network use. Nessus is constantly updated with more than 46,000 plug-ins. Key features include remote and local (authenticated) security checks, client/server architecture with a web-based interface, and an embedded scripting language for writing your own plug-ins or understanding the existing ones.

Nmap, short for Network Mapper, is a free, open-source tool for vulnerability scanning and network discovery. Network administrators use Nmap to identify what devices are running on their systems, discovering hosts that are available and the services they offer, finding open ports and detecting security risks.

Nmap can be used to monitor single hosts as well as vast networks that encompass hundreds of thousands of devices and multitudes of subnets.

Though Nmap has evolved over the years and is extremely flexible, at heart it's a port- scan tool, gathering information by sending raw packets to system ports. It listens for responses and determines whether ports are open, closed or filtered in some way by, for example, a firewall. Other terms used for port scanning include port discovery or enumeration.

Port scanning

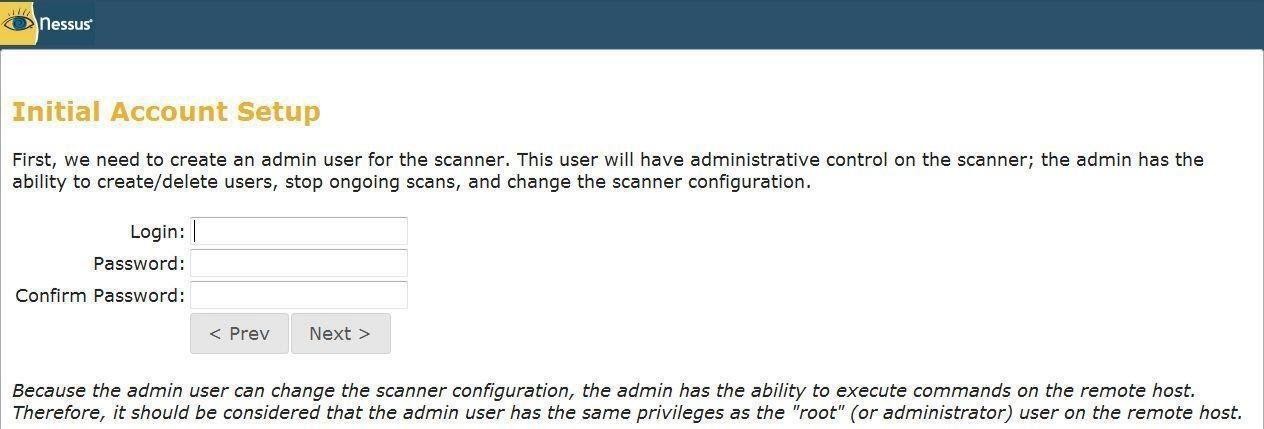
The packets that Nmap sends out return with IP addresses and a wealth of other data, allowing you to identify all sorts of network attributes, giving you a profile or map of the network and allowing you to create a hardware and software inventory.

Different protocols use different types of packet structures. Nmap employs transport layer protocols including TCP (Transmission Control Protocol), UDP (User Datagram Protocol), and SCTP (Stream Control Transmission Protocol), as well as supporting protocols like ICMP (Internet Control Message Protocol), used to send error messages.

The various protocols serve different purposes and system ports. For example, the low resource overhead of UDP is suited for real-time streaming video, where you sacrifice some lost packets in return for speed, while non-real time streaming videos in YouTube are buffered and use the slower, albeit more reliable TCP.

### Procedure and Installation:

1. Download Nessus setup file
   * Go to [www.tenable.com](http://www.tenable.com/) -> products -> Nessus-> download
   * Download Nessus for ubuntu14.4
2. Install Nessus
   * Open a Terminal and go to the download directory (cd)
   * Run sudo dpkg -i Nessus\*.deb. Enter root password.
   * Start it sudo /etc/init.d/nessusd start
3. After installation, go to https://localhost:8834
   * Click on Get started for registration
   * Initial account setup: provide login details



Plug-in feed registration

1. Go to <http://www.nessus.org/register/>for registration and activation code. Register by entering user details and valid mail id. Activation code will be sent to given mail id.

Activate using supplied activation code

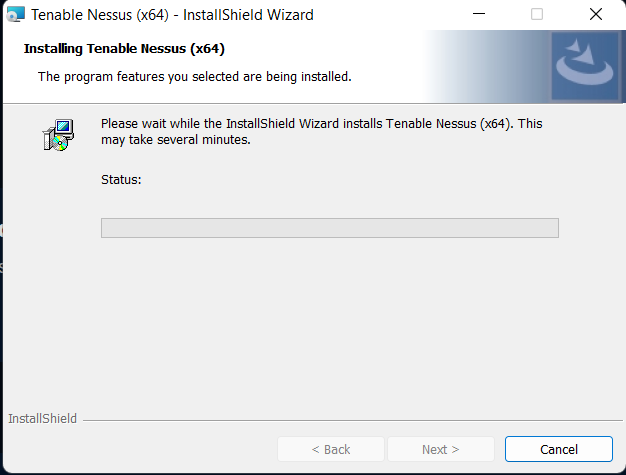
1. Click on download plug-in
2. It will show following fetching plug-ins window

- Sign in for Nessus vulnerability scanner using login name and password

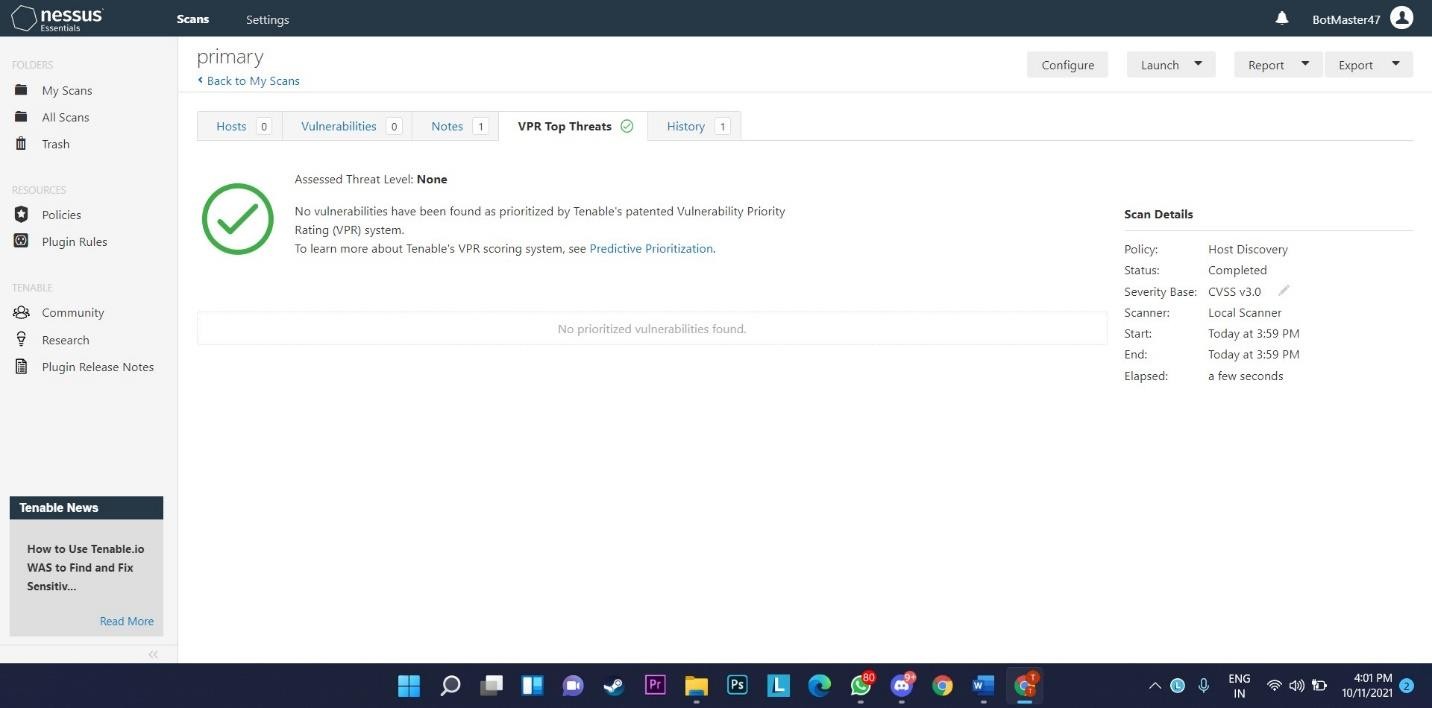
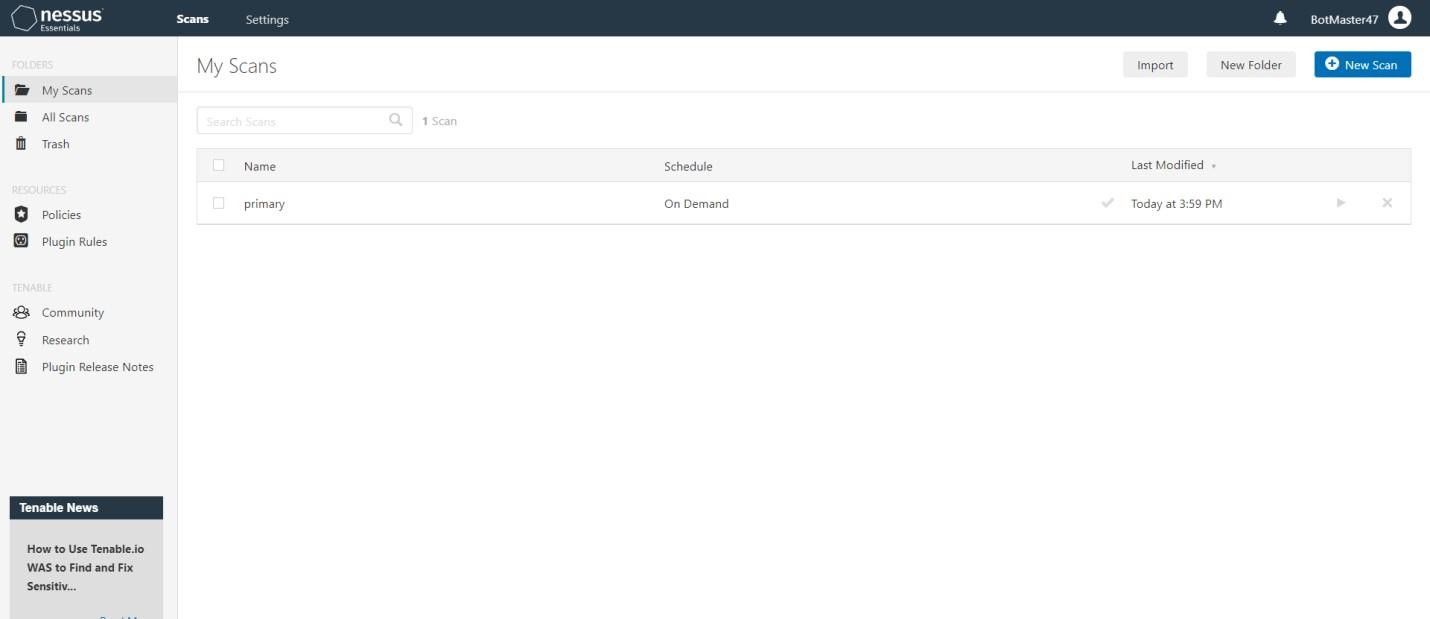
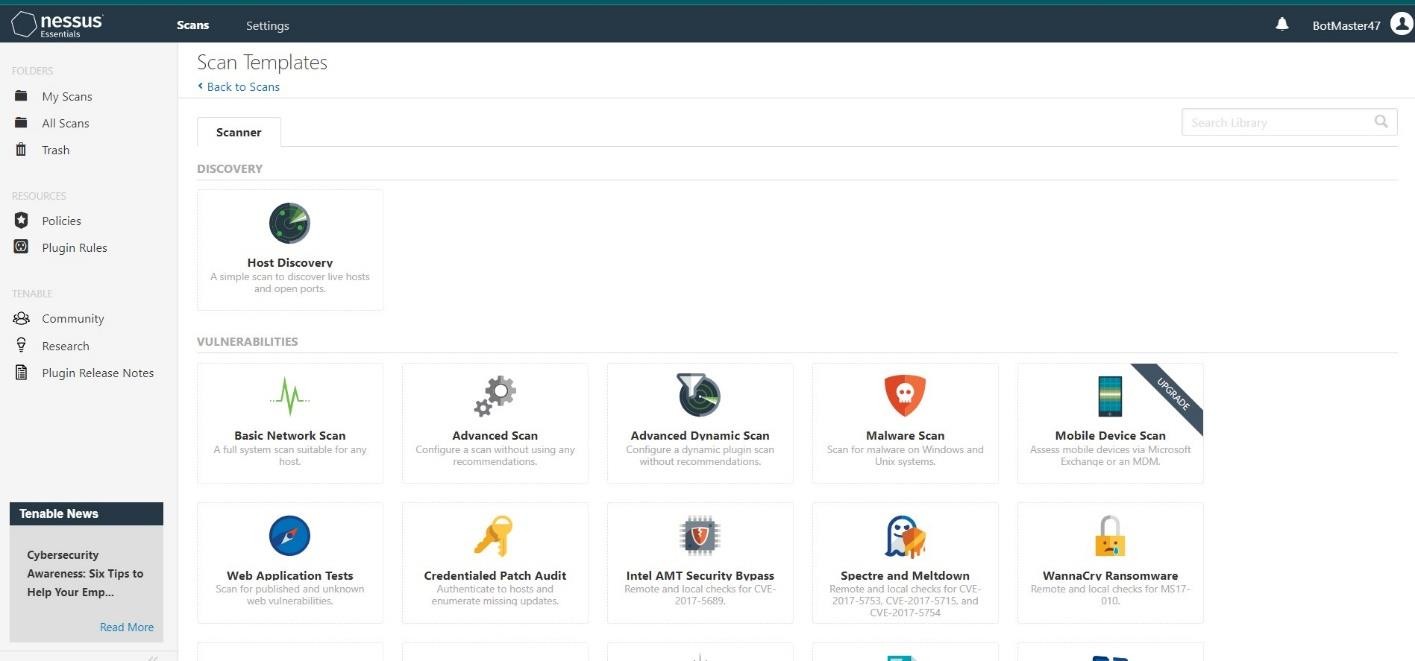
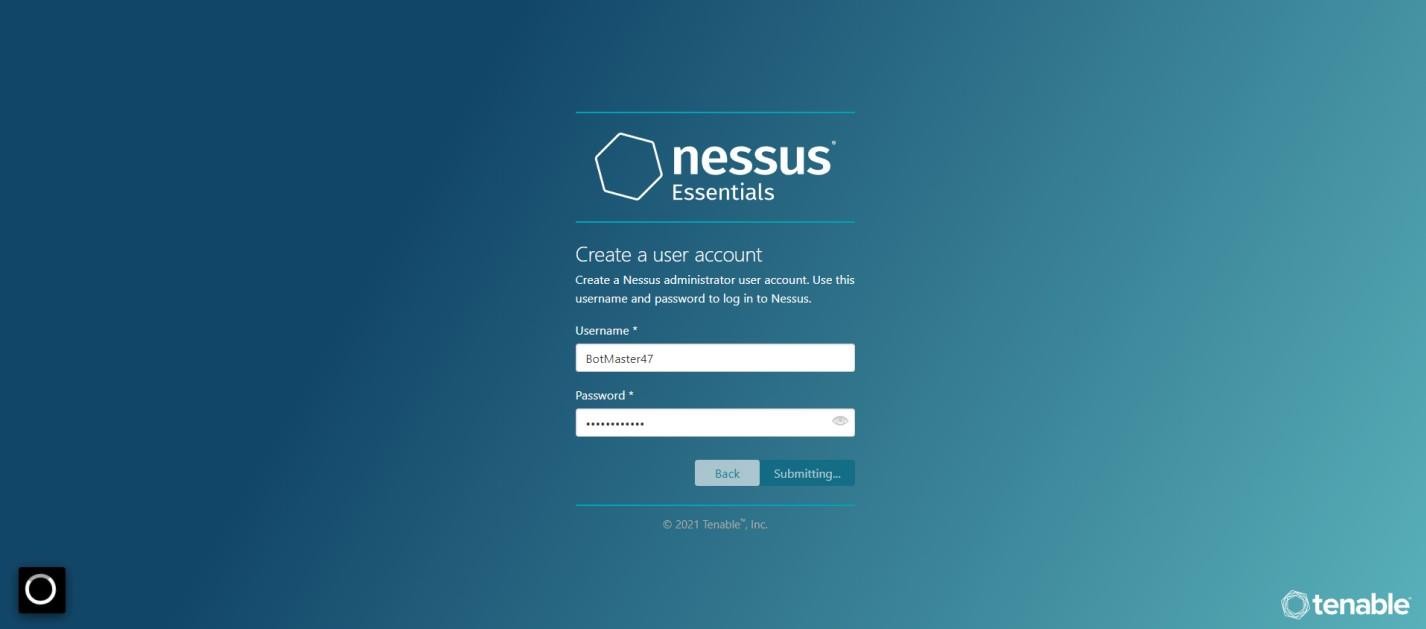
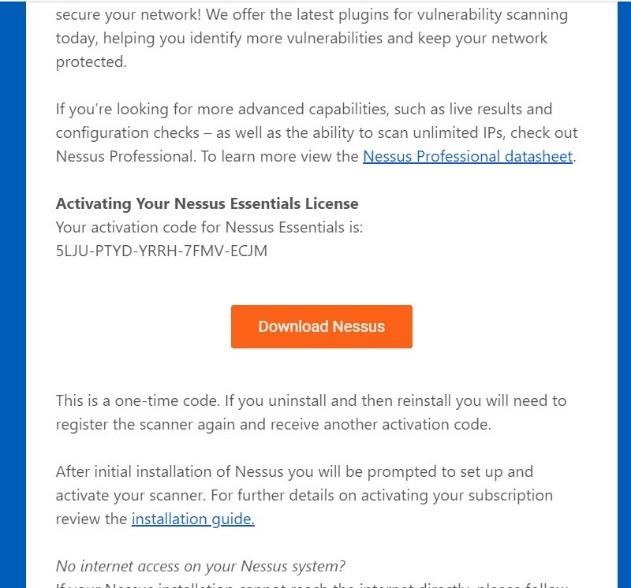
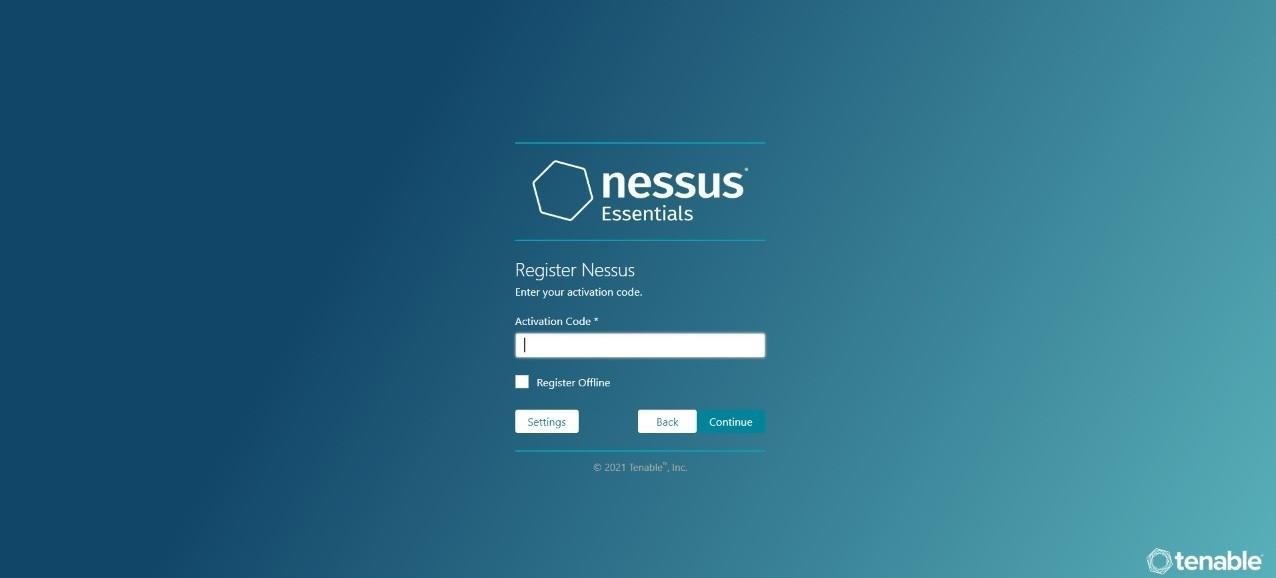
Create scan by clicking scan-> add scan -> provide scan details(scan name, type of scan, target addretc)

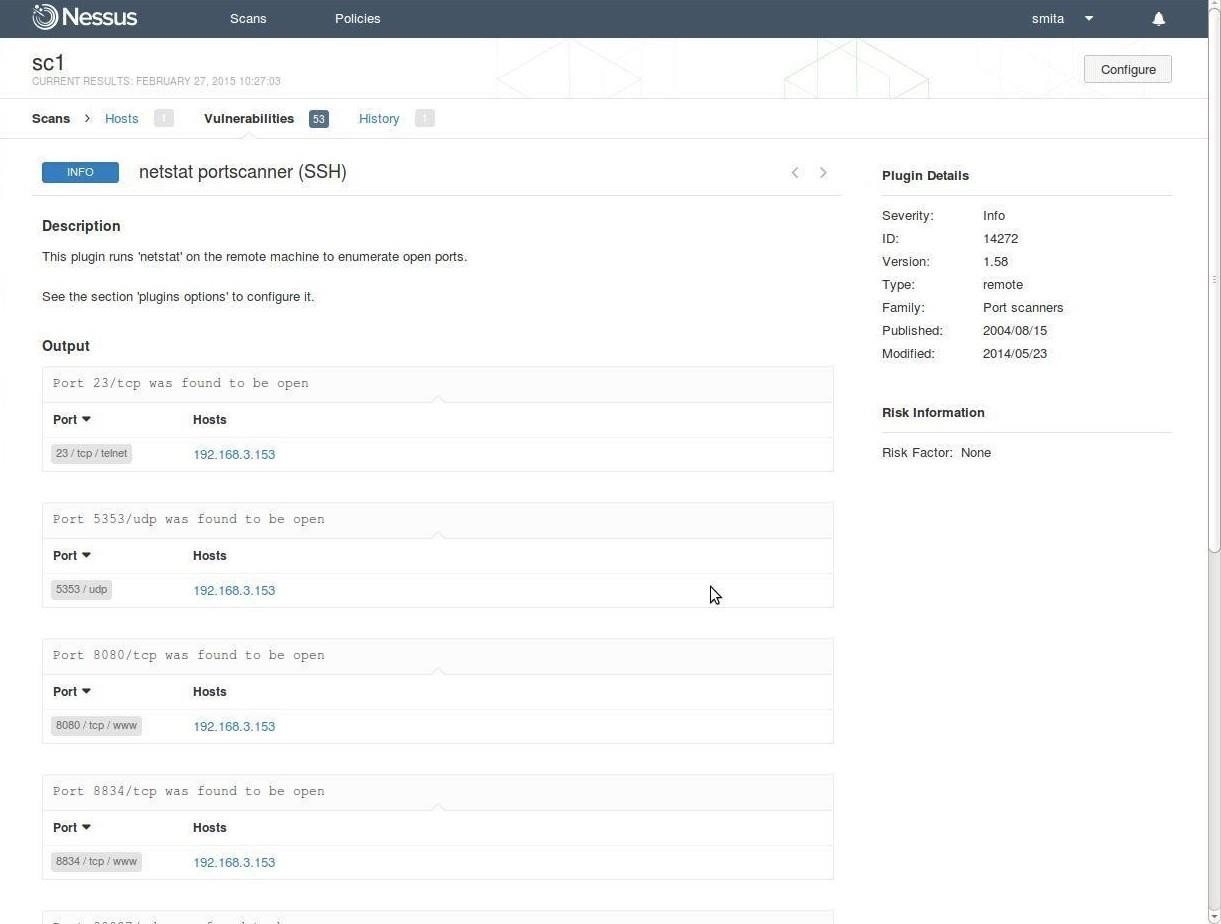
Check vulnerability report in Results Basic Network scanning

installation done



Nessus webserver

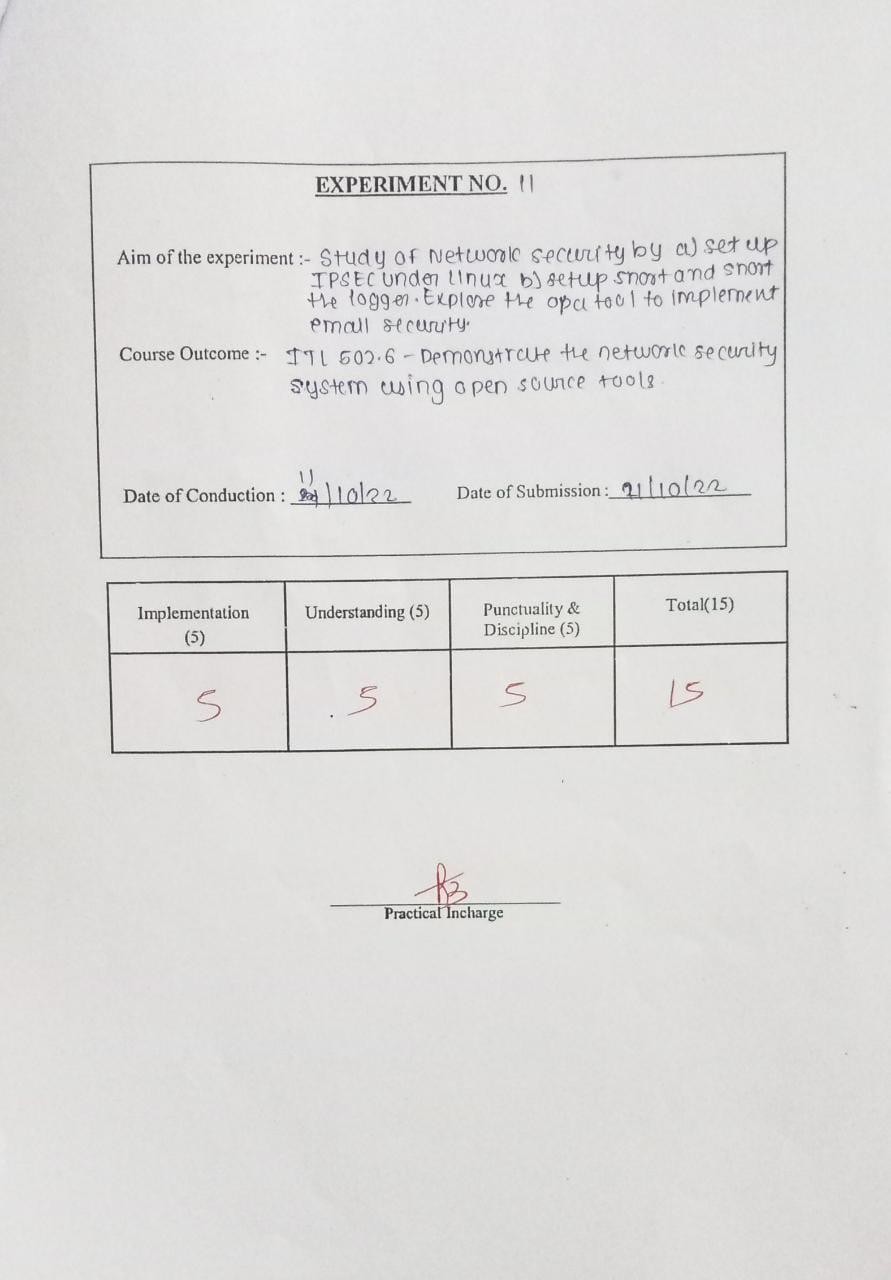




### Conclusion:

Studied the malicious software using different tools:

* 1. Keylogger attack using a keylogger tool.
  2. Simulate DOS attack using Hping or other tools
  3. Use the NESSUS/ISO Kali Linux tool to scan the network forvulnerabilities



### Aim:

Study of Network security by

# Experiment 11

### Theory:

**IPSec**

* + 1. Set up IPSec under Linux.
    2. Set up Snort and study the logs.
    3. Explore the GPG tool to implement email security

Internet Protocol Security (IPsec) is a protocol suite for securing Internet Protocol (IP) communications by authenticating and encrypting each IP packet of a communication session. IPsec includes protocols for establishing mutual authentication between agents at the beginning of the session and negotiation of cryptographic keys to be used during the session. IPsec can be used in protecting data flows between a pair of hosts (host-to-host), between a pair of security gateways (network-to-network), or between a security gateway and a host (network-to-host)

Internet Protocol security (IPsec) uses cryptographic security services to protect communications over Internet Protocol (IP) networks. IPsec supports network- level peer authentication, data origin authentication, data integrity, data confidentiality (encryption), and replay protection.

IPsec is an end-to-end security scheme operating in the Internet Layer of the Internet Protocol Suite, while some other Internet security systems in widespread use, such as Transport Layer Security (TLS) and Secure Shell (SSH), operate in the upper layers at Application layer. Hence, only IPsec protects any application traffic over an IP network. Applications can be automatically secured by IPsec at the IP layer.

Here ipsec is implemented using strongSwan tool. strongSwan is a IPsec implementation. It uses openSSL pluugin (Eliptic Curve Cryptography).

### Snort and Study Log

Snort is an open source network intrusion prevention and detection system (IDS/IPS) developed by [Sourcefire.](http://sourcefire.com/) Combining the benefits of signature, protocol, and anomaly-based inspection, Snort is the most widely deployed IDS/IPS technology worldwide. With millions of downloads and nearly 400,000 registered users, Snort has become the de facto standard for IPS.

Snort can be configured to run in three modes:

1. **Sniffer mode** : It simply reads the packets off of the network and displays them for you in acontinuous stream on the console (screen)

1. **Packet Logger mode** : logs the packets to disk

**Network Intrusion Detection System (NIDS) mode**: it performs detection and analysis onnetwork traffic. This is the most complex and configurable mode.

#### Input/Output

1. **Sniffer Mode**
   * Printing out the TCP/IP packet headers to the screen (i.e. sniffer mode) C:\snort\bin> snort -v

This command will run Snort and just show the IP and TCP/UDP/ICMP headers

* + If you want to see the application data in transit C:\snort\bin> snort -vd

This instructs Snort to display the packet data as well as the headers.

* + For more descriptive display, showing the data link layer headers C:\snort\bin>snort -vde

As an aside, notice that the command line switches can be listed separately or in a combined form. Same result can be obtained using

C:\snort\bin>snort -d -v –e

* + Other commands

C:\snort\bin> snort –W : show the interface number

* + - Packet dump mode C:\snort\bin> snort

–v –i1

### GPG tool to implement email security

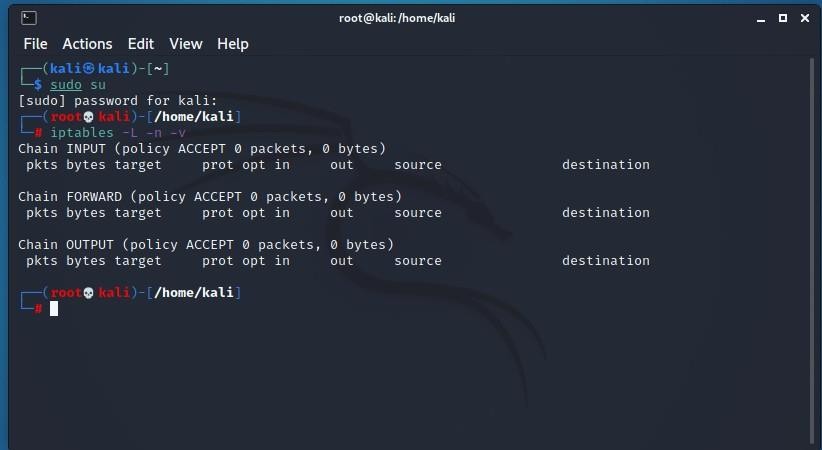
Pretty Good Privacy (PGP) is a data encryption and decryption computer program that provides cryptographic privacy and authentication for data communication. PGP is often used for signing, encrypting, and decrypting texts, e-mails, files, directories, and whole disk partitions and to increase the security of e-mail communications.

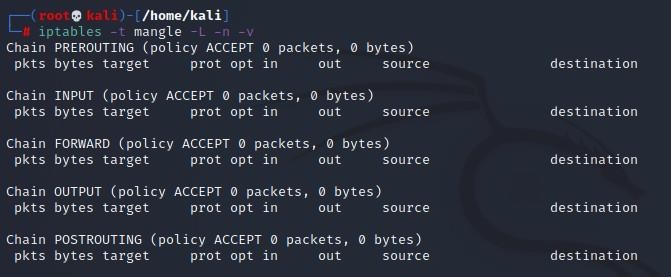
PGP encryption uses a serial combination of hashing, data compression, symmetric-key cryptography, and finally public-key cryptography; each step uses one of several supported algorithms. Each public key is bound to a user name and/or an e-mail address. The first version of this system was generally known as a web of trust to contrast with the X.509 system, which uses a hierarchical approach based on certificate authority and which was added to PGP implementations later. Current versions of PGP encryption include both options through an automated key management server.

GNU Privacy Guard (GnuPG or GPG) is a free software replacement for Symantec's PGP cryptographic software suite. GnuPG is a hybrid-encryption software program because it uses a combination of conventional symmetric-key cryptography for speed, and public-key cryptography for ease of secure key exchange, typically by using the recipient's public key to encrypt a session key which is only used once. This mode of operation is part of the OpenPGP standard and has been part of PGP from its first version.

**Procedure:**

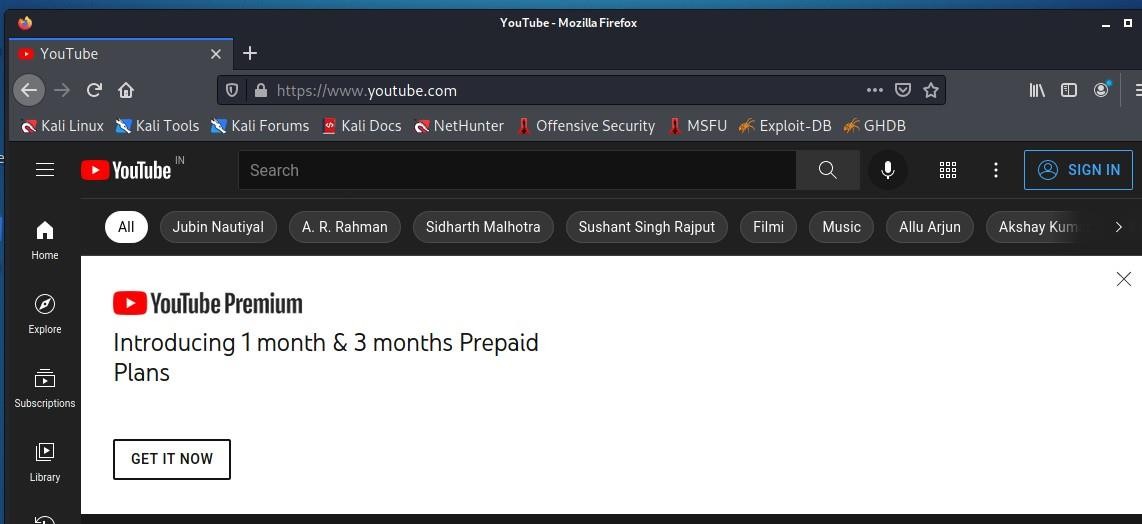
### Iptables: (To display current state of the filter table)

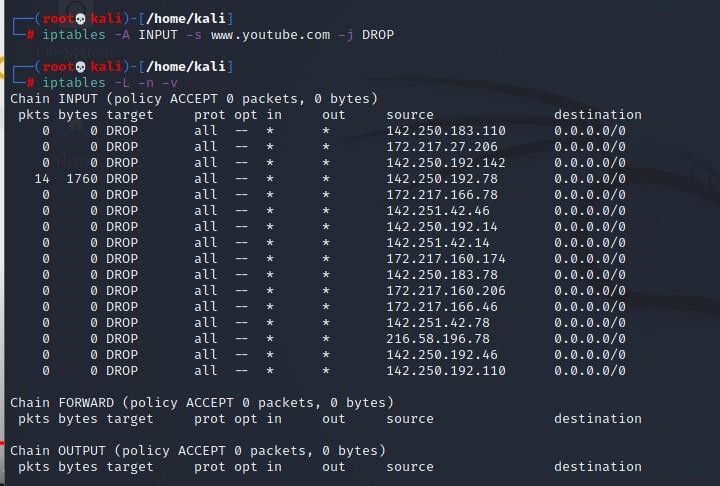


to display mangle table

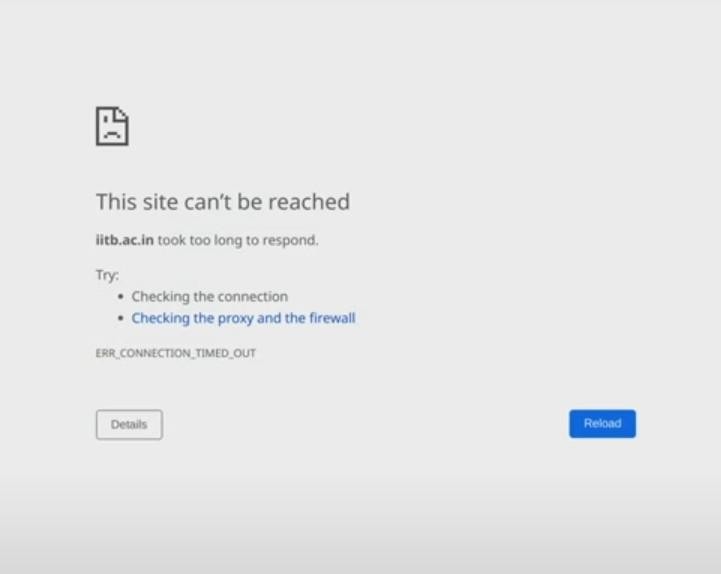
To block a website

select a website

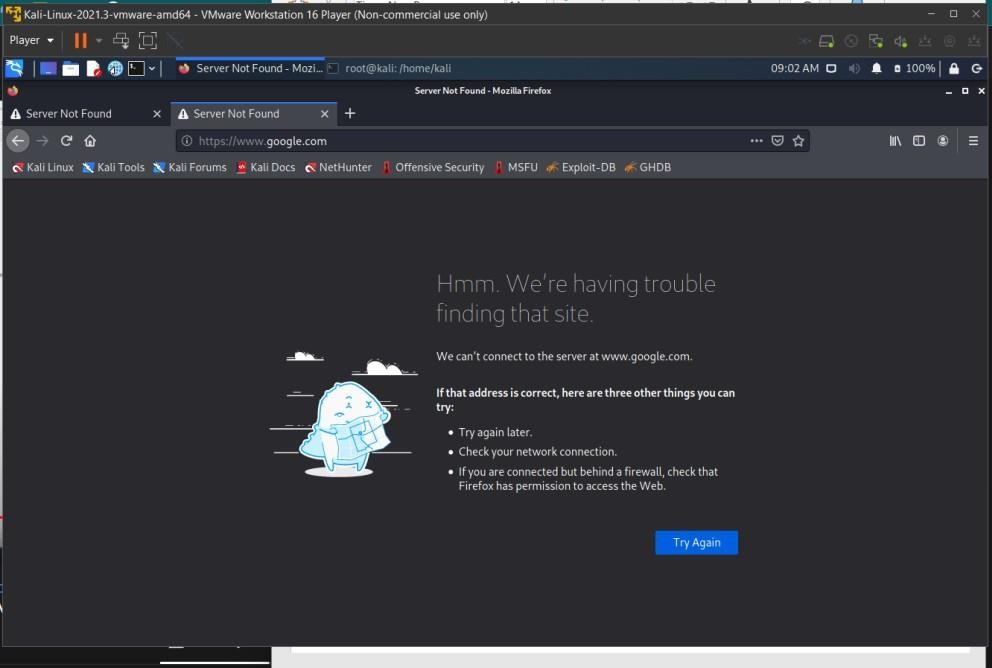
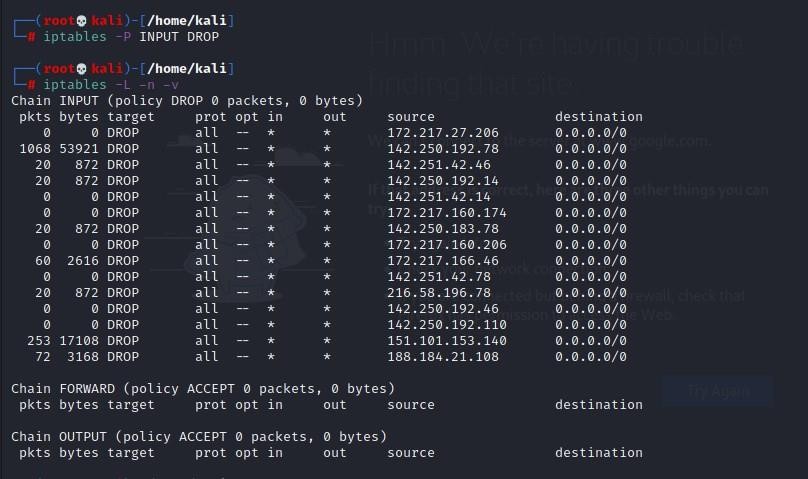


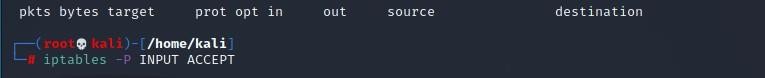
run the following commands

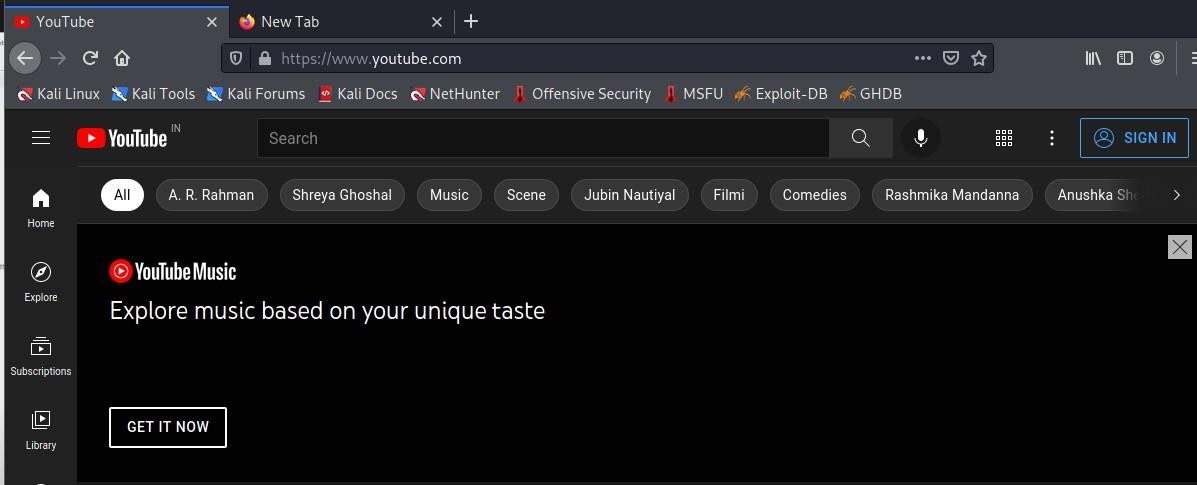
try reopening the site the following message is displayed



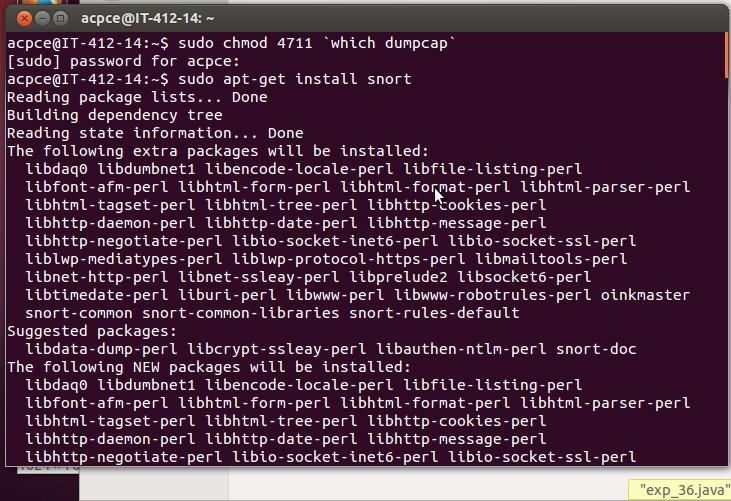
to remove the rule that you applied run the following commands Command to block the entire network



To change the policy back to accept



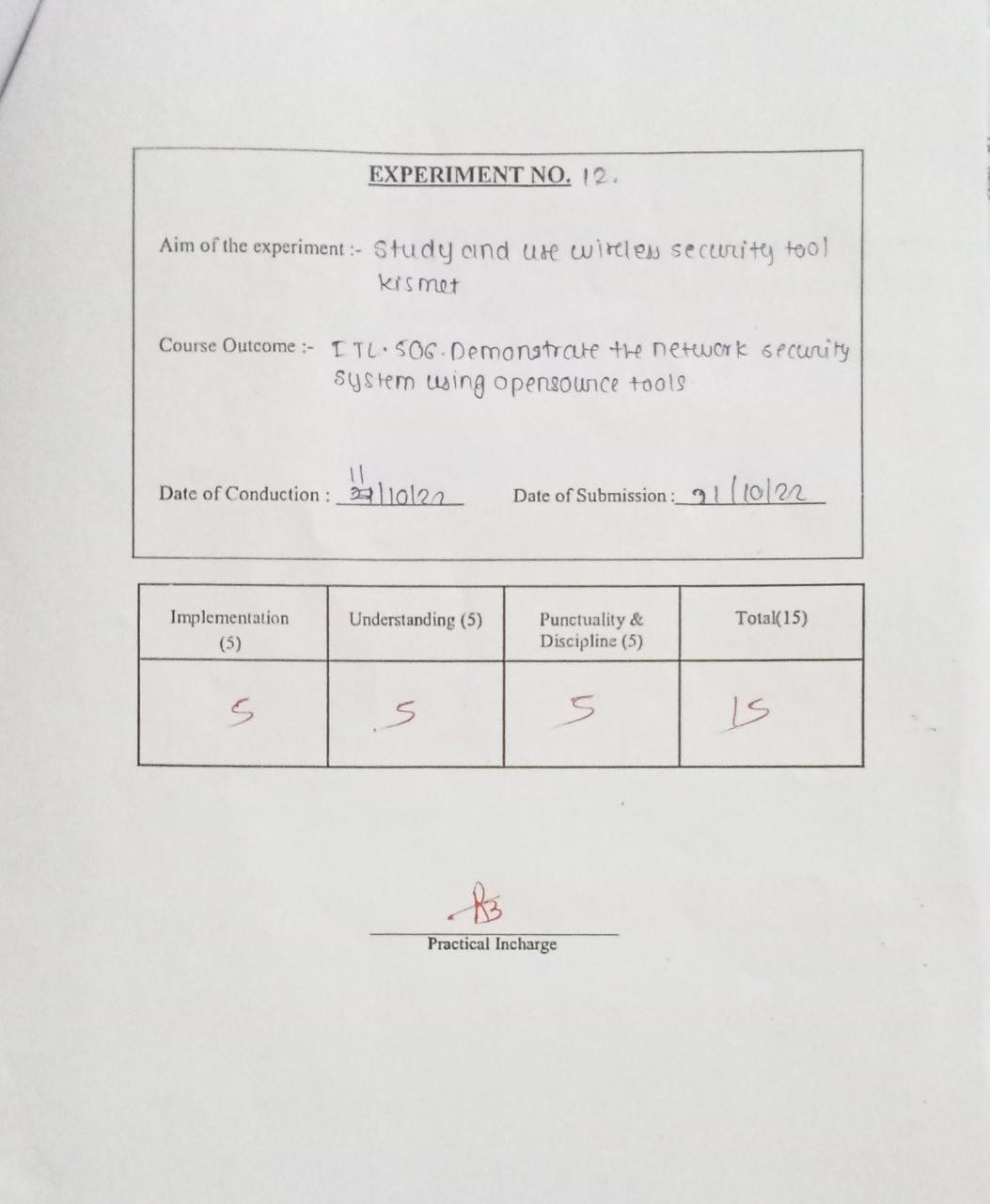
### Snort and study the logs



#### Network Intrusion Detection System (NIDS) mode

**Predefined snort rules**

cg16@cg16:/etc/snort/rules$ ls



**Aim:**

# Experiment 12

Study and use of wireless security tool- Kismet.

**Theory:**

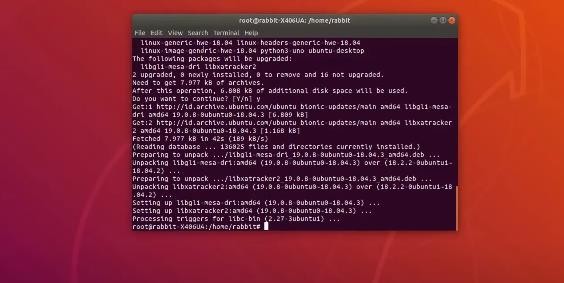
Kismet is an open source wireless network analyzer running under the Linux, Unix and Mac OS. Kismet is an 802.11 wireless network detector, sniffer, and intrusion detection system. Kismet works with any wireless card which supports raw monitoring mode, and can sniff 802.11b, 802.11a, 802.11g, and 802.11n traffic. Kismet identifies networks by passively collecting packets and detecting networks.

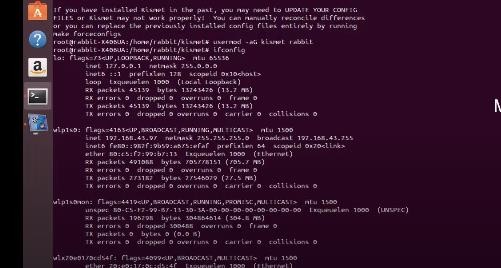
#### Installation of Kismet:

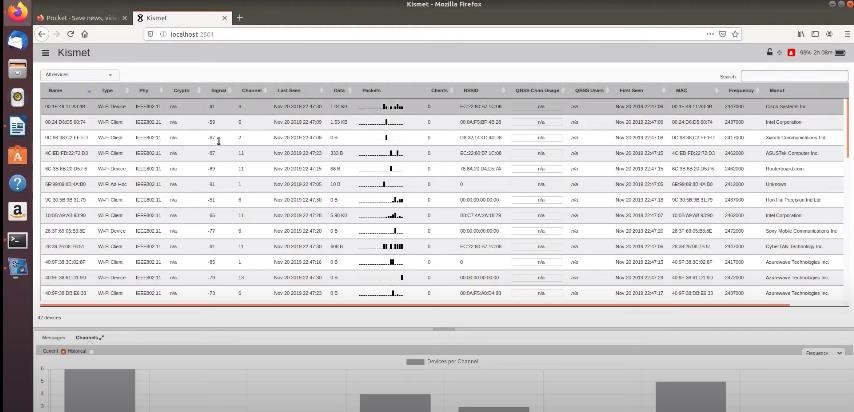
1. Download and install Kismet #sudo apt-get install kismet
2. Set the Kismet configuration(/etc/kismet.kismet.conf) file

1. Set wireless source and wireless network interface in config file

**Output:**







**Conclusion:** studied the use of wireless security tool- Kismet.