ATME COLLEGE OF ENGINEERING

13thKM Stone, Bannur Road, Mysore - 570028

Department of Computer Science & Engineering – Cyber Security

(ACADEMIC YEAR 2025-26)

LABORATORY MANUAL

SUBJECT: NETWORK SECURITY LAB

SUBJECT CODE: BCYL606

SEMESTER: VI

Outcome Based Education (OBE) and Choice Based Credit System (CBCS)

(Effective from the academic year 2025-26)

Composed by	Verified by	Approved by
Mr. Mukesh	Mrs. Suhasini	Dr. NASREEN FATHIMA
PROGRAMMER	FACULTY CO-ORDINATOR	HOD, CSD

INSTITUTIONAL MISSION AND VISION

Objectives

- To provide quality education and groom top-notch professionals, entrepreneurs and leaders for different fields of engineering, technology and management.
- To open a Training-R & D-Design-Consultancy cell in each department, gradually introduce doctoral and postdoctoral programs, encourage basic & applied research in areas of social relevance, and develop the instituteas a center of excellence.
- To develop academic, professional and financial alliances with the industry as well as the academia at national and transnational levels.
- To cultivate strong community relationships and involve the students and the staff in local community service.
- To constantly enhance the value of the educational inputs with the participation of students, faculty, parents and industry.

Vision

- Development of academically excellent, culturally vibrant, socially responsible and globally competent human resources.

Mission

- To keep pace with advancements in knowledge and make the students competitive and capable at the global level.
- To create an environment for the students to acquire the right physical, intellectual, emotional and moral foundations and shine as torch bearers of tomorrow's society.
- To strive to attain ever-higher benchmarks of educational excellence.

Network Security Lab

Subject Code	:	BCYL606	CIE Marks	:	50
Teaching Hours/Week (L:T:P: S)	:	0:0:2:0	SEE Marks	:	50
Credits	:	01	Total Marks	:	100
Examination type (SEE)		Practical	Exam Hours	:	02

Course objectives:

- To get Practical exposure of Substitution Techniques
- To get Practical exposure on Symmetric Cryptographic Algorithms
- To get Practical exposure on Asymmetric Cryptographic Algorithms

Sl.NO	Experiments (Implement using C/C++/Java Programming Languages)
1	Implement Caesar Cipher substitution method
2	Implement Mono alphabetic cipher with the given key and input
3	Implement Poly alphabetic Cipher
4	Encrypt the given text using Play fair Cipher with the given key. Also perform the decry operation
5	Implement Encryption and Decryption techniques in Hill Cipher
6	Demonstrate Single and Double Transposition techniques
7	Implement Simple DES/DES Algorithm
8	Generate Pseudo random numbers using Linear Congruential method
9	Generate Pseudo random numbers using Blum Blum Shub Generator
10	Implement RSA Algorithm
11	Demonstrate Diffie Hellman Key exchange Algorithm
12	Implement Fermat's and Euler's Theorem (Course Instructor need to explain the theoren before execution as the topic not covered in Theory part)

Course outcomes:

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

- Implement Substitution Ciphers
- Design the various Transposition Techniques
- Demonstrate the working of various Symmetric Cipher algorithms
- Demonstrate the working of various aymmetric Cipher algorithms

Assessment Details (both CIE and SEE)

The weightage of Continuous Internal Evaluation (CIE) is 50% and for Semester End Exam (SEE) is 50%. The minimum passing mark for the CIE is 40% of the maximum marks (20 marks out of 50) and for the SEE minimum passing mark is 35% of the maximum marks (18 out of 50 marks). A student shall be deemed to have satisfied the academic requirements and earned the credits allotted to each subject/ course if the student secures a minimum of 40% (40 marks out of 100) in the sum total of the CIE (Continuous Internal Evaluation) and SEE (Semester End Examination) taken together

Continuous Internal Evaluation (CIE):

CIE marks for the practical course are **50 Marks**.

The split-up of CIE marks for record/ journal and test are in the ratio **60:40**.

- Each experiment is to be evaluated for conduction with an observation sheet and record write-up. Rubrics for the evaluation of the journal/write-up for hardware/software experiments are designed by the faculty who is handling the laboratory session and are made known to students at the beginning of the practical session.
- Record should contain all the specified experiments in the syllabus and each experiment write-up will be evaluated for 10 marks.
- Total marks scored by the students are scaled down to **30 marks** (60% of maximum marks).
- Weightage to be given for neatness and submission of record/write-up on time.
- Department shall conduct a test of 100 marks after the completion of all the experiments listed in the syllabus.
- In a test, test write-up, conduction of experiment, acceptable result, and procedural knowledge will carry a weightage of 60% and the rest 40% for viva-voce.
- The suitable rubrics can be designed to evaluate each student's performance and learning ability.
- The marks scored shall be scaled down to **20 marks** (40% of the maximum marks).

The Sum of scaled-down marks scored in the report write-up/journal and marks of a test is the total CIE marks scored by the student.

Semester End Evaluation (SEE):

- SEE marks for the practical course are 50 Marks.
- The examination schedule and names of examiners are informed to the university before the conduction of the examination. These practical examinations are to be conducted between the schedule mentioned in the academic calendar of the University.
- All laboratory experiments are to be included for practical examination.
- (Rubrics) Breakup of marks and the instructions printed on the cover page of the answer script to be strictly adhered to by the examiners. **OR** based on the course requirement evaluation rubrics shall be decided jointly by examiners.
- Students can pick one question (experiment) from the questions lot prepared by the examiners jointly.
- Evaluation of test write-up/ conduction procedure and result/viva will be conducted jointly by examiners.

General rubrics suggested for SEE are mentioned here, writeup-20%, Conduction procedure and result in -60%, Viva-voce 20% of maximum marks. SEE for practical shall be evaluated for 100 marks and scored marks shall be scaled down to 50 marks (however, based on course type, rubrics shall be decided by the examiners)
Change of experiment is allowed only once and 15% of Marks allotted to the procedure part are to be made zero.
The minimum duration of SEE is 02 hours

Suggested Learning Resources:

- https://www.bugcrowd.com/glossary/symmetric-encryption-algorithms/
- https://www.ibm.com/think/topics/symmetric-encryption
- https://www.wikihow.com/Create-Substitution-Ciphers

CONTENTS

Introduction to Network Security

Network security refers to the policies, practices, and technologies designed to protect data, devices, and networks from unauthorized access, misuse, modification, or attack. The main goal of network security is to ensure that data remains secure during transmission and storage.

The foundation of network security lies in cryptography, which provides techniques to protect information by transforming it into a secure format.

Goals of Network Security

The primary goals of network security are often summarized as the **CIA Triad**:

Confidentiality - Confidentiality ensures that information is accessible only to authorized users. Encryption techniques are used to prevent unauthorized access.

Integrity - Integrity ensures that data is not altered or modified by unauthorized parties. Hash functions and digital signatures help maintain data integrity.

Availability - Availability ensures that information and resources are accessible to authorized users whenever required.

Cryptography

Cryptography is the science of securing information by converting it into an unreadable format. It ensures secure communication in the presence of adversaries.

Basic Terminology

- **Plaintext**: Original readable message.
- **Ciphertext**: Encrypted unreadable message.
- **Encryption**: Process of converting plaintext into ciphertext.
- **Decryption**: Process of converting ciphertext back into plaintext.
- **Key**: A value used in encryption and decryption.

Types of Cryptography

Cryptographic algorithms are mainly classified into:

1. Classical Cryptography
2. Symmetric Key Cryptography
3. Asymmetric Key Cryptography

Classical Cryptography

Classical cryptography techniques are traditional methods used before modern computers. They are mainly divided into:

1.Substitution Techniques

In substitution techniques, letters of the plaintext are replaced with other letters or symbols.

Examples:

Caesar Cipher

Each letter in the plaintext is shifted by a fixed number of positions in the alphabet.

Example:

If shift = 3

A → D

B → E

Monoalphabetic Cipher

Each letter in the plaintext is mapped to a fixed substitute letter.

Polyalphabetic Cipher

Uses multiple substitution alphabets to improve security. Example: Vigenère Cipher.

Playfair Cipher

Encrypts pairs of letters using a 5×5 matrix.

Hill Cipher

Uses matrix multiplication to encrypt blocks of letters.

2.Transposition Techniques

In transposition techniques, the positions of letters are rearranged without changing the actual letters.

Examples:

- Single Columnar Transposition
- Double Transposition

These methods provide better security than simple substitution methods.

Symmetric Key Cryptography

Symmetric key cryptography uses the same key for both encryption and decryption. It is also known as **secret key cryptography**.

Features:

- Fast and efficient

- Suitable for large data encryption
- Requires secure key sharing

Example: Data Encryption Standard (DES)

Asymmetric Key Cryptography

Asymmetric cryptography uses two keys:

- Public Key (for encryption)
- Private Key (for decryption)

It solves the key distribution problem found in symmetric encryption.

Example - RSA Algorithm

Random Number Generation in Cryptography

Random number generation is an essential component of cryptography. Many cryptographic algorithms depend on random numbers for generating secret keys, initialization vectors, nonces, digital signatures, and session keys.

There are two main types of random numbers: True Random Numbers (TRNG) and Pseudo Random Numbers (PRNG). True random numbers are generated from physical processes such as thermal noise or atmospheric noise and are completely unpredictable. Pseudo random numbers are generated using mathematical algorithms and an initial value called a seed.

Program - 1

Implement RSA Algorithm

```
package rsa;
import java.math.BigInteger;
import java.util.Random;
import java.io.DataInputStream;
public class RSA {
    BigInteger p, q, N, phi, e, d;
    int bitlength = 1024;
    RSA() {
        Random r = new Random();
        p = BigInteger.probablePrime(bitlength, r);
        q = BigInteger.probablePrime(bitlength, r);
        System.out.println("Prime number p is" + p);
        System.out.println("prime number q is" + q);
        N = p.multiply(q);
        phi = p.subtract(BigInteger.ONE).multiply(q.subtract(BigInteger.ONE));
        e = BigInteger.probablePrime(bitlength / 2, r);
        while (phi.gcd(e).intValue() > 1) e = e.add(BigInteger.ONE);
        System.out.println("Public key is" + e);
        d = e.modInverse(phi);
        System.out.println("Private key is" + d);
    }
    public static void main(String[] args) throws Exception {
        RSA rsa = new RSA();
        DataInputStream in = new DataInputStream(System.in);
        System.out.println("Enter the plain text:");
        String msg = in.readLine();
        System.out.println("Encrypting string:" + msg);
        System.out.println("string in bytes:" + bytes(msg.getBytes()));
        byte[] enc = rsa.encrypt(msg.getBytes());
        byte[] dec = rsa.decrypt(enc);
        System.out.println("Dcrypting Bytes:" + bytes(dec));
        System.out.println("Dcrypted string:" + new String(dec));
    }
    static String bytes(byte[] b) {
        String s = "";
        for (byte x : b) s += x;
        return s;
    }
    byte[] encrypt(byte[] m) { return new BigInteger(m).modPow(e, N).toByteArray(); }
    byte[] decrypt(byte[] m) { return new BigInteger(m).modPow(d, N).toByteArray(); }
}
```

Program – 2

Implement Caesar Cipher substitution method

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.Scanner;

public class CeaserCipher {
    static Scanner sc = new Scanner(System.in);
    static BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

    public static void main(String[] args) throws IOException {
        System.out.print("Enter any String: ");
        String str = br.readLine();

        System.out.print("Enter the Key: ");
        int key = sc.nextInt();

        String encrypted = encrypt(str, key);
        System.out.println("\nEncrypted String is: " + encrypted);

        String decrypted = decrypt(encrypted, key);
        System.out.println("Decrypted String is: " + decrypted);
    }

    public static String encrypt(String str, int key) {
        String encrypted = "";
        for (int i = 0; i < str.length(); i++) {
            int c = str.charAt(i);

            if (Character.isUpperCase(c)) {
                c = c + (key % 26);
                if (c > 'Z') c = c - 26;
            } else if (Character.isLowerCase(c)) {
                c = c + (key % 26);
                if (c > 'z') c = c - 26;
            }
            encrypted += (char) c;
        }
        return encrypted;
    }

    public static String decrypt(String str, int key) {
        String decrypted = "";
        for (int i = 0; i < str.length(); i++) {
            int c = str.charAt(i);
            if (Character.isUpperCase(c)) {
                c = c - (key % 26);
                if (c < 'A') c = c + 26;
            } else if (Character.isLowerCase(c)) {
                c = c - (key % 26);
                if (c < 'a') c = c + 26;
            }
```

```
            decrypted += (char) c;
        }
        return decrypted;
    }
}
```

Program 3

Implement Mono alphabetic cipher with the given key and input

```
import java.util.*;
public class MAC {
    static boolean isValidKey(String key) {
        if (key.length() != 26)
            return false;
        boolean[] used = new boolean[26];
        for (int i = 0; i < 26; i++) {
            char ch = key.charAt(i);
            if (ch < 'A' || ch > 'Z')
                return false;
            if (used[ch - 'A'])
                return false;
            used[ch - 'A'] = true;
        }
        return true;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter 26-letter key: ");
        String key = sc.nextLine().toUpperCase();
        if (!isValidKey(key)) {
            System.out.println("Invalid Key! Must contain 26 unique letters A-Z.");
            return;
        }
        System.out.print("Enter plaintext: ");
        String text = sc.nextLine().toUpperCase();
        StringBuilder encrypted = new StringBuilder();
        StringBuilder decrypted = new StringBuilder();
        // Encryption
        for (int i = 0; i < text.length(); i++) {
            char ch = text.charAt(i);
            if (ch >= 'A' && ch <= 'Z')
                encrypted.append(key.charAt(ch - 'A'));
            else
                encrypted.append(ch);
        }
        // Decryption
        for (int i = 0; i < encrypted.length(); i++) {
            char ch = encrypted.charAt(i);
            if (ch >= 'A' && ch <= 'Z')
                decrypted.append((char) ('A' + key.indexOf(ch)));
            else
                decrypted.append(ch);
        }
        System.out.println("Encrypted Text: " + encrypted);
        System.out.println("Decrypted Text: " + decrypted);
    }
}
```

Program 4

Implement Poly alphabetic Cipher

```
package polycipher;
import java.util.*;
public class PolyCipher {
    static String process(String text, String key, boolean encrypt) {
        StringBuilder result = new StringBuilder();
        int j = 0;
        for (int i = 0; i < text.length(); i++) {
            char c = text.charAt(i);
            if (Character.isLetter(c)) {
                int shift = key.charAt(j % key.length()) - 'A';
                char base = Character.isUpperCase(c) ? 'A' : 'a';
                result.append((char) ((encrypt
                        ? (c - base + shift)
                        : (c - base - shift + 26)) % 26 + base));
                j++;
            } else {
                result.append(c);
            }
        }
        return result.toString();
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter Text: ");
        String text = sc.nextLine();
        System.out.print("Enter Key: ");
        String key = sc.nextLine();
        if (text == null || text.trim().isEmpty()|| !text.matches("[a-zA-Z ]+")) {
            System.out.println("Invalid Text!");
            return;
        }

        if (key == null || key.trim().isEmpty() || !key.matches("[a-zA-Z]+")) {
            System.out.println("Invalid Key! Use letters only.");
            return;
        }
        key = key.toUpperCase();
        String enc = process(text, key, true);
        System.out.println("Encrypted: " + enc);
        System.out.println("Decrypted: " + process(enc, key, false));
        sc.close();
    }
}
```