DELHI TECHNOLOGICAL UNIVERSITY

(FORMERLY DELHI COLLEGE OF ENGINEERING)

SHAHBAD DAULATPUR ,BAWANA ROAD,DELHI-110042

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INFORMATION AND NETWORK SECURITY

LAB FILE
CS 305
IPM LAB

SUBMITTED TO:
DR. ASHISH KUMAR
ASSISTANT PROFESSOR
DEPT. OF CSE

SUBMITTED BY:
KRRISH KUMAR
BTECH: 3rd YEAR
23/EE/320

INDEX

S.No.	Experiment	Date	Sign.
1.	Write a program to encrypt and decrypt a message using a caesar cipher		
2.	Write a program to encrypt and decrypt a message using Playfair Cipher.		
3.	Write a Program to implement Hill Cipher algorithm that encrypts pairs of letters and also implement Decryption Algorithm.		
4.	Implement the RSA public-key encryption and decryption process, including key generation.		
5	Write a Program to demonstrate how two parties can establish a shared secret key over an insecure communication channel using **Diffie-Hellman key exchange algorithm.**		
6.	Write a program to implement the multiplicative inverse of a number under mod.		
7.	Write a program that demonstrates the critical properties of the SHA-512 hash function.		
8.	Write a program to Implementation of Digital Signature Standard (DSS) using RSA based digital signature scheme.		
9.	Installation and usage of Wireshark or tcpdump to capture and analyze network traffic (UDP/TCP/HTTP sessions).		
10.	Experimenting with OpenSSL to set up a secure web server-browser communication channel (SSL/TLS).		

EXPERIMENT NO. – 1

Aim Write a program to encrypt and decrypt a message using a caesar cipher

Theory

Topic _____ Date _____

Theory of Caesar Cipher

Caesar cipher is one of the simplest method of encryption. It is a substitution cipher, meaning each letter in the plaintext is replaced by another letter that is a fixed number of positions down the alphabet.

Principle:

Assign numerical value to letters:

$A = 0, \quad B = 1, \quad C = 2, \quad \ldots\ldots, \quad Z = 25$

Encryption $\Rightarrow \quad C = (P + K) \bmod 26$

Decryption $\Rightarrow \quad P = (C - K) \bmod 26$

Applications:

Not used in for real-world security, it helps in understanding the basics of symmetric key cryptography and modular arithmetic.

Numerical Problem

Topic _____ Date _____

Numerical

Plaintext : "HELLO"
Key = 3

H = 7, E = 4, L, = 11, O = 14

Encryption formula:

$C = (P + K) \bmod 26$

Letter	P	K	C=(P+K) mod 26	Cipher Letter
H	7	3	10	K
E	4	3	7	H
L	11	3	14	O
L	11	3	14	O
O	14	3	17	R

Decryption :

$P = (C - K) \bmod 26$

K=3 (given in question)

$P = (C - 3) \bmod 26$

K (10) : P = 10 - 3 = 7 → H
H (7) : P = 7 - 3 = 4 → E
O (14) : P = 14 - 3 = 11 → L
R (17) : P = 17 - 3 = 14 → O

"HELLO"

Code

```
#include <bits/stdc++.h>
using namespace std;

// Encryption Function
void Encrypt(string inputText, int shift) {

    // Iterate through each character
    for(char &s: inputText) {

        // Check for alphabet
        if(isalpha(s)) {

            // Check for uppercase or lowercase
            if(s<'a') {
                s+=shift;           // Shift ascii value right
                if(s>'Z') s-=26;    // If shift exceeds z, cycle start from a
            } else {
                s+=shift;           // Same procedure as above
                if(s>'z') s-=26;
            }
        }
    }

    cout << "The encoded message is " << inputText << "\n";
}

//Decryption Function
void Decrypt(string inputText, int shift) {

    // Iterate through each character
    for(char &s: inputText) {

        // Check for alphabet
        if(isalpha(s)) {

            // Check for uppercase or lowercase
            if(s<'a') {
                s-=shift;           // Shift ascii value left
                if(s<'A') s+=26;    // If shift preceeds a, cycle back from z
            } else {
                s-=shift;           // Same procedure as above
                if(s<'a') s+=26;
            }
```

```
            }
        }
    }

    cout << "The decoded message is " << inputText << "\n";
}

int main() {

    string codeType, inputText;
    int shift;

    cin >> codeType;

    // Invalid case
    if(codeType != "encode" && codeType != "decode") {
        cout << "Invalid input\n";
        return 0;
    }

    cout << "Type the message: ";
    cin >> inputText;

    cout << "Enter the number of shifts: ";
    cin >> shift;

    if(codeType == "encode") Encrypt(inputText,shift);
    else Decrypt(inputText,shift);

    return 0;
}
```

Output

encrypt

Sample Input

```
encode
apple
3
```

Your Output

```
Type the message: Enter the number of shifts: The encoded message is dssoh
```

Decrypt:

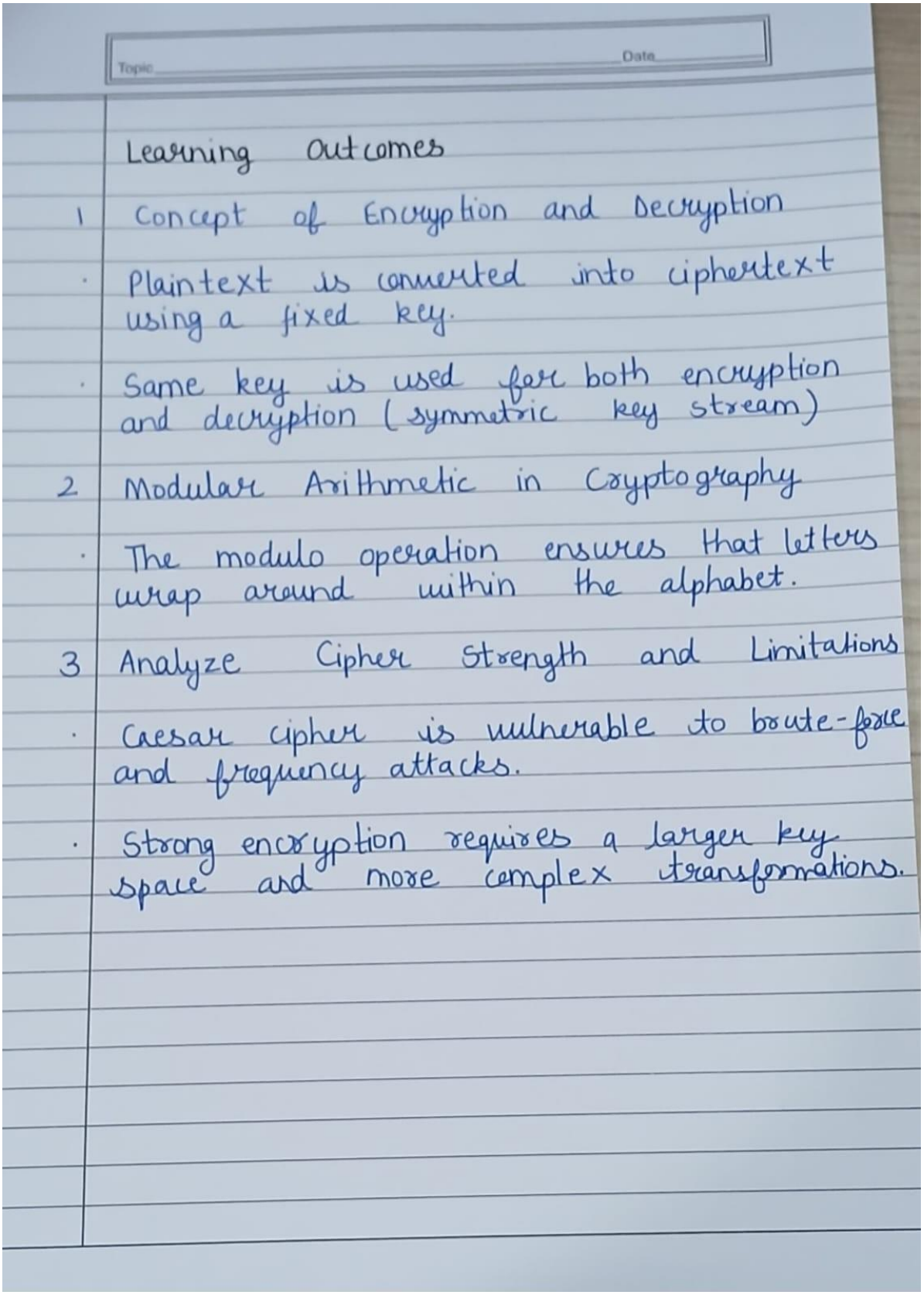
Sample Input

```
decode
dssoh
3
```

Your Output

```
Type the message: Enter the number of shifts: The decoded message is apple
```

Learning Outcome



Topic Date

Learning Outcomes

1 Concept of Encryption and Decryption

- Plaintext is converted into ciphertext using a fixed key.
- Same key is used for both encryption and decryption (symmetric key stream)

2 Modular Arithmetic in Cryptography

- The modulo operation ensures that letters wrap around within the alphabet.

3 Analyze Cipher Strength and Limitations

- Caesar cipher is vulnerable to brute-force and frequency attacks.
- Strong encryption requires a larger key space and more complex transformations.

EXPERIMENT NO. – 2

Aim Write a program to encrypt and decrypt a message using Playfair Cipher.

Theory

Topic _____ Date _____

Playfair Cipher – Theory

It is a digraph substitution cipher, meaning it encrypts pairs of letters (known as digraphs) instead of single letters.

The cipher uses a 5×5 matrix of letters constructed using a keyword.

Features:

- Encrypts digraphs, increasing complexity over monoalphabetic ciphers.
- Resistant to frequency analysis since letter pairs, not single letters, are used.
- Simple but provide good manual encryption security.

Limitations:

- Still vulnerable to modern frequency analysis on digraphs.
- Cannot handle punctuation or numbers directly.
- The 'I/J' combination reduces alphabet clarity.

Numerical Problem

Topic _____ Date _____

Numerical:

Keyword : MONARCHY
Plaintext : INSTRUMENTS

Key matrix

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

IN	ST	RU	ME	NT	SX

IN → GA
ST → ~~TS~~ TL
RU → MZ
ME → CL
NT → RQ
SX → XA

Ciphertext : GA TL MZ CL RQ XA

Code

```
#include <bits/stdc++.h>
using namespace std;

// Function to convert the string to lowercase
void toLowerCase(string &plain) {
    int n = plain.size();
    for (int i = 0; i < n; i++) {
        if (plain[i] > 64 && plain[i] < 91)
            plain[i] += 32;
    }
}

// Function to remove all spaces in a string
void removeSpaces(string &plain) {
    int n = plain.size();
    string temp;
    for (int i = 0; i < n; i++) {
        if (plain[i] != ' ') {
            temp += plain[i];
        }
    }
    plain = temp;
}

// Function to generate the 5x5 key square
void generateKeyTable(string &key,
        vector<vector<char>> &keyT) {
    int n = key.size();

    // 5x5 key table
    keyT.resize(5, vector<char>(5, 0));

    // a 26 character hashmap
    // to store count of the alphabet
    vector<int> hash(26, 0);
```

```
    int i, j, k, flag = 0;
    for (i = 0; i < n; i++) {
        if (key[i] != 'j')
            hash[key[i] - 97] = 2;
    }

    hash['j' - 97] = 1;

    i = 0;
    j = 0;

    for (k = 0; k < n; k++) {
        if (hash[key[k] - 97] == 2) {
            hash[key[k] - 97] -= 1;
            keyT[i][j] = key[k];
            j++;
            if (j == 5) {
                i++;
                j = 0;
            }
        }
    }

    for (k = 0; k < 26; k++) {
        if (hash[k] == 0) {
            keyT[i][j] = (char)(k + 97);
            j++;
            if (j == 5) {
                i++;
                j = 0;
            }
        }
    }
}
```

```
void search(vector<vector<char>> &keyT,
        char a, char b, vector<int> &arr) {
    int i, j;

    if (a == 'j')
        a = 'i';
    else if (b == 'j')
        b = 'i';

    for (i = 0; i < 5; i++) {

        for (j = 0; j < 5; j++) {

            if (keyT[i][j] == a) {
                arr[0] = i;
                arr[1] = j;
            }
            else if (keyT[i][j] == b) {
                arr[2] = i;
                arr[3] = j;
            }
        }
    }
}

// Function to make the plain text length to be even
int prepare(string &str) {
    if (str.size() % 2 != 0) {
        str += 'z';
    }
    int n = str.size();
    return n;
}
```

```
void encrypt(string &str, vector<vector<char>> &keyT) {
    int n = str.size();
    vector<int> arr(4);

    for (int i = 0; i < n; i += 2) {

        search(keyT, str[i], str[i + 1], arr);

        if (arr[0] == arr[2]) {
            str[i] = keyT[arr[0]][(arr[1] + 1) % 5];
            str[i + 1] = keyT[arr[0]][(arr[3] + 1) % 5];
        }
        else if (arr[1] == arr[3]) {
            str[i] = keyT[(arr[0] + 1) % 5][arr[1]];
            str[i + 1] = keyT[(arr[2] + 1) % 5][arr[1]];
        }
        else {
            str[i] = keyT[arr[0]][arr[3]];
            str[i + 1] = keyT[arr[2]][arr[1]];
        }
    }
}

// Function to encrypt using Playfair Cipher
void encryptByPlayfairCipher(string &str, string &key) {
    vector<vector<char>> keyT;
    removeSpaces(key);
    toLowerCase(key);
    toLowerCase(str);
    removeSpaces(str);
    prepare(str);
    generateKeyTable(key, keyT);
    encrypt(str, keyT);
}

int main() {
    string key = "cricket";
    string str = "football";
    cout << "Key text: " << key << endl;
    cout << "Plain text: " << str << endl;
    encryptByPlayfairCipher(str, key);
    cout << "Cipher text: " << str << endl;
    return 0;
```

Output

Your Output

```
Key text: cricket
Plain text: football
Cipher text: tuvgdbhb
```

Learning Outcome

Topic _____ Date _____

Learning Outcomes

1 Concept of polygraphic substitution cipher

- Playfair encrypts pairs of characters rather than individual characters to increase cryptographic strength.

2 Construct and use a key matrix:

- Creating a 5×5 matrix from any keyboard

3 Perform manual encryption and decryption

- Ability to manually encrypt or decrypt a message using the Playfair method by applying the three encryption rules (same row, same column, rectangle).

EXPERIMENT NO. – 3

Aim Write a Program to implement Hill Cipher algorithm that encrypts pairs of letters and also implement Decryption Algorithm

Theory

Topic :- ..... Date : __/__/____

Theory of Hill Cipher

Hill cipher is a type of polyalphabetic substitution cipher. It encrypts a group (block) of letters together using matrix multiplication.

Basic Principle:

Each letter of the alphabet is first assigned a numerical value:

$A=0$, $B=1$, $C=2$, ... $Z=25$

It uses linear algebra to encrypt plaintext.

$$C = K \times P \pmod{26}$$

P = plaintext vector
C = ciphertext vector
K = key matrix

Decryption Process:

$$P = K^{-1} \times C \pmod{26}$$

K^{-1} is the modular inverse of K modulo 26

Numeical Problem

Topic _____ Date _____

Numerical :

Key matrix : $K = \begin{bmatrix} 3 & 3 \\ 2 & 5 \end{bmatrix}$

Plaintext : HI → H = 7 , I = 8

$$P = \begin{bmatrix} 7 \\ 8 \end{bmatrix}$$

Encryption :

$C = K \times P \pmod{26}$

$$= \begin{bmatrix} 3 & 3 \\ 2 & 5 \end{bmatrix} \begin{bmatrix} 7 \\ 8 \end{bmatrix} = \begin{bmatrix} 3\times7 + 3\times8 \\ 2\times7 + 5\times8 \end{bmatrix}$$

$$= \begin{bmatrix} 21+24 \\ 14+40 \end{bmatrix}$$

$$= \begin{bmatrix} 45 \\ 54 \end{bmatrix}$$

take mod 26

$$C = \begin{bmatrix} 19 \\ 2 \end{bmatrix}$$

19 → T

2 → C

Topic _____ Date _____

Decryption:

$|K| = 3\times5 - 3\times2 = 15-6 = 9$

$9x \equiv 1 \pmod{26}$

$x = 3$

$9^{-1} \equiv 3 \pmod{26}$

$$adj(K) = \begin{bmatrix} d & -b \\ -c & a \end{bmatrix} = \begin{bmatrix} 5 & -3 \\ -2 & 3 \end{bmatrix}$$

$$= \begin{bmatrix} 5 & 23 \\ 24 & 3 \end{bmatrix}$$

$K^{-1} = (9^{-1}) \times adj(K) \pmod{26}$

$$= 3 \times \begin{bmatrix} 5 & 23 \\ 24 & 3 \end{bmatrix} = \begin{bmatrix} 15 & 69 \\ 72 & 9 \end{bmatrix}$$

mod 26

$$= \begin{bmatrix} 15 & 17 \\ 20 & 9 \end{bmatrix}$$

"TC" → T = 19, C = 2

$$C = \begin{bmatrix} 19 \\ 2 \end{bmatrix}$$

$P = K^{-1} \times C \pmod{26}$

$$= \begin{bmatrix} 15 & 17 \\ 20 & 9 \end{bmatrix} \begin{bmatrix} 19 \\ 2 \end{bmatrix} = \begin{bmatrix} 15\times19 + 17\times2 \\ 20\times19 + 9\times2 \end{bmatrix} = \begin{bmatrix} 319 \\ 398 \end{bmatrix}$$

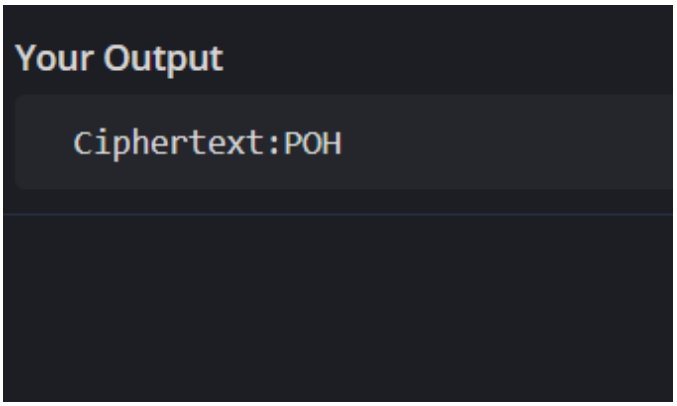
$P = \begin{bmatrix} 7 \\ 8 \end{bmatrix}$ $\quad 7 \rightarrow H, 8 \rightarrow I \quad$ "HI"

Code

```
#include <iostream>
using namespace std;

// Following function generates the
//  key matrix for the key string
void getKeyMatrix(string key, int keyMatrix[][3])
{
    int k = 0;
    for (int i = 0; i < 3; i++)
    {
        for (int j = 0; j < 3; j++)
        {
            keyMatrix[i][j] = (key[k]) % 65;
            k++;
        }
    }
}

// Following function encrypts the message
void encrypt(int cipherMatrix[][1],
             int keyMatrix[][3],
             int messageVector[][1])
{
    int x, i, j;
    for (i = 0; i < 3; i++)
    {
        for (j = 0; j < 1; j++)
        {
            cipherMatrix[i][j] = 0;

            for (x = 0; x < 3; x++)
            {
                cipherMatrix[i][j] +=
                    keyMatrix[i][x] * messageVector[x][j];
            }

            cipherMatrix[i][j] = cipherMatrix[i][j] % 26;
        }
    }
}
```

```
void HillCipher(string message, string key)
{
    // Get key matrix from the key string
    int keyMatrix[3][3];
    getKeyMatrix(key, keyMatrix);

    int messageVector[3][1];

    // Generate vector for the message
    for (int i = 0; i < 3; i++)
        messageVector[i][0] = (message[i]) % 65;

    int cipherMatrix[3][1];

    // Following function generates
    // the encrypted vector
    encrypt(cipherMatrix, keyMatrix, messageVector);

    string CipherText;

    // Generate the encrypted text from
    // the encrypted vector
    for (int i = 0; i < 3; i++)
        CipherText += cipherMatrix[i][0] + 65;

    // Finally print the ciphertext
    cout << " Ciphertext:" << CipherText;
}

// Driver function for above code
int main()
{
    // Get the message to be encrypted
    string message = "ACT";

    // Get the key
    string key = "GYBNQKURP";

    HillCipher(message, key);

    return 0;
```

Output



Learning Outcome

Topic _____ Date _____

Learning Outcomes:

1. Conceptual understanding
   - Matrix Algebra can be used in cryptography
   - The difference between monoalphabetic and polyalphabetic ciphers.
   - Recognize that Hill Cipher performs block encryption, improving security by mixing multiple letters.

EXPERIMENT NO. – 4

Aim Implement the RSA public-key encryption and decryption process, including key generation

Theory

Topic :- ..... Date : __/__/____

RSA Algorithm - Theory

RSA is one of the public-key cryptographic algorithms. It is used for secure data transmission, digital signatures, and encryption in modern communication systems like HTTPS, email encryption and secure logins.

Basic concept:

RSA is based on the mathematical ability of factoring large prime numbers. It uses two keys:

Public key (e, n) : used for encryption
Private key (d, n) : used for decryption

The algorithm ensures that even if the public key is known, the private key cannot be easily derived due to the complexity of factoring large numbers.

Advantages:
- High security for data transmission
- Supports both encryption and digital signatures.
- Widely adopted and mathematically proven reliable.

Limitations:
- Require large key sizes for strong security
- Computationally slower than symmetric-key algorithms.

Good Write

Teacher's Sign. .....

Numerical Problem

KRRISH KUMAR 23/EE/320

Topic :- Date : / /

1 $p = 13$, $q = 11$, RSA

1 $n = p \times q$
$= 13 \times 11$
$= 143$

2 $\phi(n) = \phi(p) \times \phi(q)$
$= (p-1)(q-1)$ { p, q being prime numbers }
$= (13-1)(11-1)$
$= 12 \times 10$
$= 120$

3 $e = ?$ $1 < e < \phi(n)$

$e = 7$ { 7, 120 are co-primes; gcd (7, 120) = 1 }

4 $d = ?$

$ed = 1 \mod \phi(n)$
$7 \times d = 1 \mod 120$

d is multiplicative inverse of 7

$d = 7^{-1} \pmod{120}$
$= 7^{\phi(143)-1} \pmod{120}$
$= 7^{120-1} \pmod{120}$

Good Write Teacher's Sign. .....

Topic :- ..... Date : __/__/____

$= 7^{119} \mod (120)$

$= (7^3)^{39} \times 7^2 \mod (120)$

$= (343)^{39} \times 49 \mod (120)$

$= (-17)^{39} \times 49 \mod (120)$

$= ((-17)^2)^{19} \times (-17)(49) \mod 120$

$= (289)^{19} \times (-17)(49) \mod 120$

$= (49)^{19} \times (-17)(49) \mod 120$

$= (49^2)^9 \times (49)^2(-17) \mod 120$

$= (2401)^{11} (-17) \mod 120$

$= 1^{11} (-17) \mod 120$

$= -17 \mod 120$

$d = 103$

5 Public key : $\{e, n\} : \{7, 143\}$

Private key : $\{d, n\} : \{103, 143\}$

Topic :- ..... Date : / /

6 Encryption

$c = p^e \pmod n$

Let $p = 21$

$c = 21^7 \pmod{143}$

$= (21^2)^3 \times 21 \pmod{143}$

$= (441)^3 \times 21 \pmod{143}$

$= 12^3 \times 21 \pmod{143}$

$= 1728 \times 21 \pmod{143}$

$= 12 \times 21 \pmod{143}$

$= 252 \pmod{143}$

$= 109$

7 Decryption

$M = c^d \bmod 143$

$= 109^{103} \bmod 143$

$= (-34)^{103} \bmod 143$

$= ((-34)^2)^{51} \times (-34) \bmod 143$

$= (1156)^{51} \times (-34) \bmod 143$

Good Write

Teacher's Sign. .....

Topic :- ..... Date : __/__/____

$= 12^{51} \times (-34) \mod 143$

$= (12^2)^{25} \times 12(-34) \mod 143$

$= 144^{25} \times 12 \times (-34) \mod 143$

$= 1^{25} \times 12 \times (-34) \mod 143$

$= -408 \mod 143$

$= (429 - 408) \mod 143$

$M = 21$

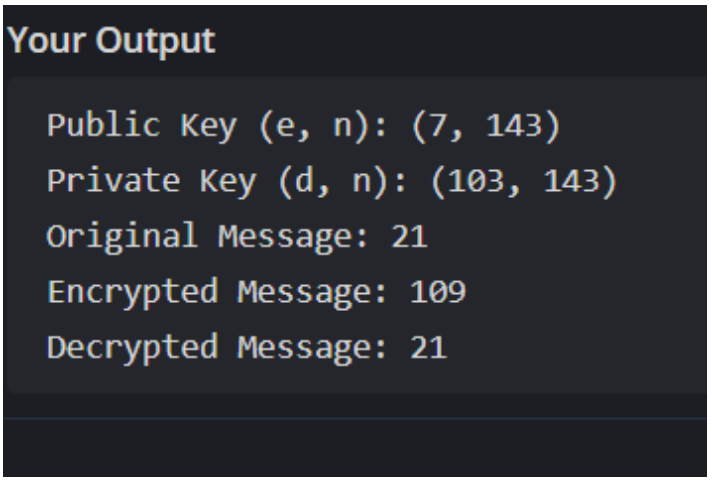
Message = 21, which is same as p which we assumed

Code

```cpp
#include <bits/stdc++.h>

using namespace std;

// Function to compute base^expo mod m
int power(int base, int expo, int m) {
    int res = 1;
    base = base % m;
    while (expo > 0) {
        if (expo & 1)
            res = (res * 1LL * base) % m;
        base = (base * 1LL * base) % m;
        expo = expo / 2;
    }
    return res;
}

// Function to find modular inverse of e modulo phi(n)
// Here we are calculating phi(n) using Hit and Trial Method
// but we can optimize it using Extended Euclidean Algorithm
int modInverse(int e, int phi) {
    for (int d = 2; d < phi; d++) {
        if ((e * d) % phi == 1)
            return d;
    }
    return -1;
}

// RSA Key Generation
void generateKeys(int &e, int &d, int &n) {
    int p = 13;
    int q = 11;

    n = p * q;
    int phi = (p - 1) * (q - 1);

    // Choose e, where 1 < e < phi(n) and gcd(e, phi(n)) == 1
    for (e = 2; e < phi; e++) {
        if (__gcd(e, phi) == 1)
            break;
    }
```

```
// Encrypt message using public key (e, n)
int encrypt(int m, int e, int n) {
    return power(m, e, n);
}

// Decrypt message using private key (d, n)
int decrypt(int c, int d, int n) {
    return power(c, d, n);
}

int main() {
    int e, d, n;

    // Key Generation
    generateKeys(e, d, n);

    cout << "Public Key (e, n): (" << e << ", " << n << ")\n";
    cout << "Private Key (d, n): (" << d << ", " << n << ")\n";

    // Message
    int M = 21;
    cout << "Original Message: " << M << endl;

    // Encrypt the message
    int C = encrypt(M, e, n);
    cout << "Encrypted Message: " << C << endl;

    // Decrypt the message
    int decrypted = decrypt(C, d, n);
    cout << "Decrypted Message: " << decrypted << endl;

    return 0;
}
```

Output


Your Output
Public Key (e, n): (7, 143)
Private Key (d, n): (103, 143)
Original Message: 21
Encrypted Message: 109
Decrypted Message: 21

Learning Outcome

Topic :- ..... Date :__/__/____

Learning Outcomes:

1 Concept of Public-key Cryptography

- Explain how asymmetric cryptography differs from symmetric cryptography.

2 Mathematical foundation of RSA

- Modular arithmetic and Euler's totient function
- Comprehend how encryption and decryption are inverse operations using modular exponentiation

3 Key Generation and Encryption / Decryption

- To choose suitable prime numbers and compute n, $\phi(n)$, e and d.
- Apply RSA equations to encrypt and decrypt messages manually for small examples.
- Recognize how real implementation use very large primes for strong encryption.

4 Evaluate Security Strength

- Analyze how RSA resists brute-force and factorization attacks.

Good Write

Teacher's Sign. .....

EXPERIMENT NO. – 5

Aim Write a Program to demonstrate how two parties can establish a shared secret key over an insecure communication channel using **Diffie-Hellman key exchange algorithm.**

Theory

Topic :- ..... Date :__/__/____

Theory of Diffie-Hellman Key Exchange Algorithm

It is the method for securely exchanging cryptographic keys over a public communication channel. It allows two parties, who have never met before, to establish a shared secret key that can be used for encryption and decryption of messages.

The main concept behind the Diffie Hellman is the use of modular arithmetic and the difficulty of solving the discrete logarithm problem. Even though the initial parameters are shared publicly, the secret key derived remains secure because it is computationally infeasible to determine the private keys.

Applications

- Used in VPNs, SSH for secure communication.
- Forms the basis of public-key cryptography systems.

Limitations

- Vulnerable to man-in-the-middle (MITM) attacks if authentication is not used.
- Does not provide authentication by itself.

Good Write

Teacher's Sign. .....

Numerical Problem

Topic :- ..... Date : __/__/____

2 Diffie - Hellman Key Exchange

Prime Number = 11

$X_a = 3$

$X_b = 4$

$K = ?$

1 prime number, $q = 11$

2 α is the primitive root of 11

Primitive root

$\alpha = 7, \quad q = 11$

$7^1 \bmod 11 = 7$

$7^2 \bmod 11 = 5$

$7^3 \bmod 11 = 2$

$7^4 \bmod 11 = 3$

$7^5 \bmod 11 = 10$

$7^6 \bmod 11 = 4$

$7^7 \bmod 11 = 6$

$7^8 \bmod 11 = 9$

$7^9 \bmod 11 = 8$

$7^{10} \bmod 11 = 1$

$= \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$

$\therefore$ 7 is primitive root of 11

3 $X_a = 3$ $\quad \alpha = 7, \ q = 11$

$X_a < 11$

X : Private

Y : Public

$$Y_A = \alpha^{X_A} \bmod q$$

$$= 7^3 \bmod 11 = 2$$

Good Write

Teacher's Sign. .....

Topic :- Date : __/__/____

4 $X_b = 4$, $\alpha = 7$, $q = 11$

$X_b < q$

$Y_b = \alpha^{X_b} \bmod q$

$= 7^4 \bmod 11$

$= 3$

5 Secret keys

$K_1 \Rightarrow$ Person A

$K_2 \Rightarrow$ Person B

$K_1 = (Y_B)^{X_A} \bmod q$

$K_2 = (Y_A)^{X_B} \bmod q$

After calculating, if $K_1 = K_2$ then success

$K_1 = 3^3 \bmod 11$

$= 5$

$K_2 = (2)^4 \bmod 11$

$= 5$

$K_1 = K_2$

Code

```
#include <bits/stdc++.h>
using namespace std;

// Power function to return value of a ^ b mod P
long long int power(long long int a, long long int b,
                    long long int P)
{
    if (b == 1)
        return a;

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

// Driver program
int main()
{
    long long int P, G, x, a, y, b, ka, kb;

    // Both the persons will be agreed upon the
    // public keys G and P
    P = 23; // A prime number P is taken
    cout << "The value of P : " << P << endl;

    G = 9; // A primitive root for P, G is taken
    cout << "The value of G : " << G << endl;

    // Alice will choose the private key a
    a = 4; // a is the chosen private key
    cout << "The private key a for Alice : " << a << endl;

    x = power(G, a, P); // gets the generated key

    // Bob will choose the private key b
    b = 3; // b is the chosen private key
    cout << "The private key b for Bob : " << b << endl;

    y = power(G, b, P); // gets the generated key
```

```
    y = power(G, b, P); // gets the generated key

    // Generating the secret key after the exchange
    // of keys
    ka = power(y, a, P); // Secret key for Alice
    kb = power(x, b, P); // Secret key for Bob
    cout << "Secret key for the Alice is : " << ka << endl;

    cout << "Secret key for the Bob is : " << kb << endl;

    return 0;
}
```

Output

```
Your Output

The value of P : 23
The value of G : 9
The private key a for Alice : 4
The private key b for Bob : 3
Secret key for the Alice is : 9
Secret key for the Bob is : 9
```

Learning Outcome

Topic :- ..... Date : __/__/____

Learning Outcomes:

1 Understand the concept of key exchange:

- Two parties can securely share a secret key over an insecure channel.

2 The mathematical foundation:

- The role of prime numbers, modular arithmetic and the discrete logarithm problem in securing the algorithm.
- Understanding the exponentiation module p prowides security.

3 Illustrate the working of the algorithm:

- Step through each phase: parameter selection, key generation, exchange and shared key computation.
- Perform simple regarding the shared key.

4 Analyze the security aspects:

- Identity potential vulnerabilities like MITM attacks and ways to prevent them.

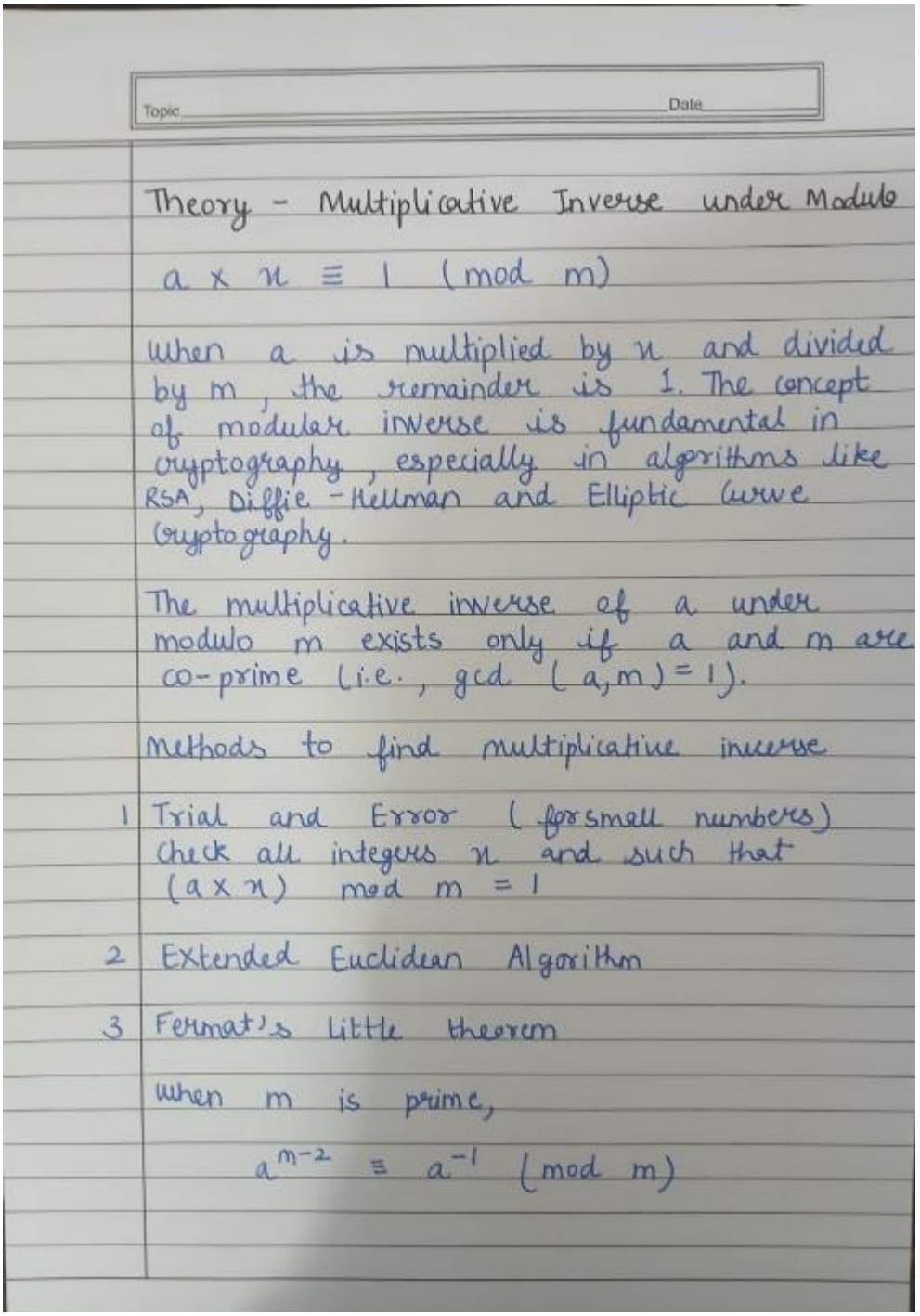
Good Write

Teacher's Sign. .....

EXPERIMENT NO. – 6

Aim Write a program to implement the multiplicative inverse of a number under mod.

Theory:



Topic _____ Date _____

Theory - Multiplicative Inverse under Modulo

$a \times x \equiv 1 \pmod{m}$

When a is multiplied by x and divided by m, the remainder is 1. The concept of modular inverse is fundamental in cryptography, especially in algorithms like RSA, Diffie-Hellman and Elliptic Curve Cryptography.

The multiplicative inverse of a under modulo m exists only if a and m are co-prime (i.e., gcd (a, m) = 1).

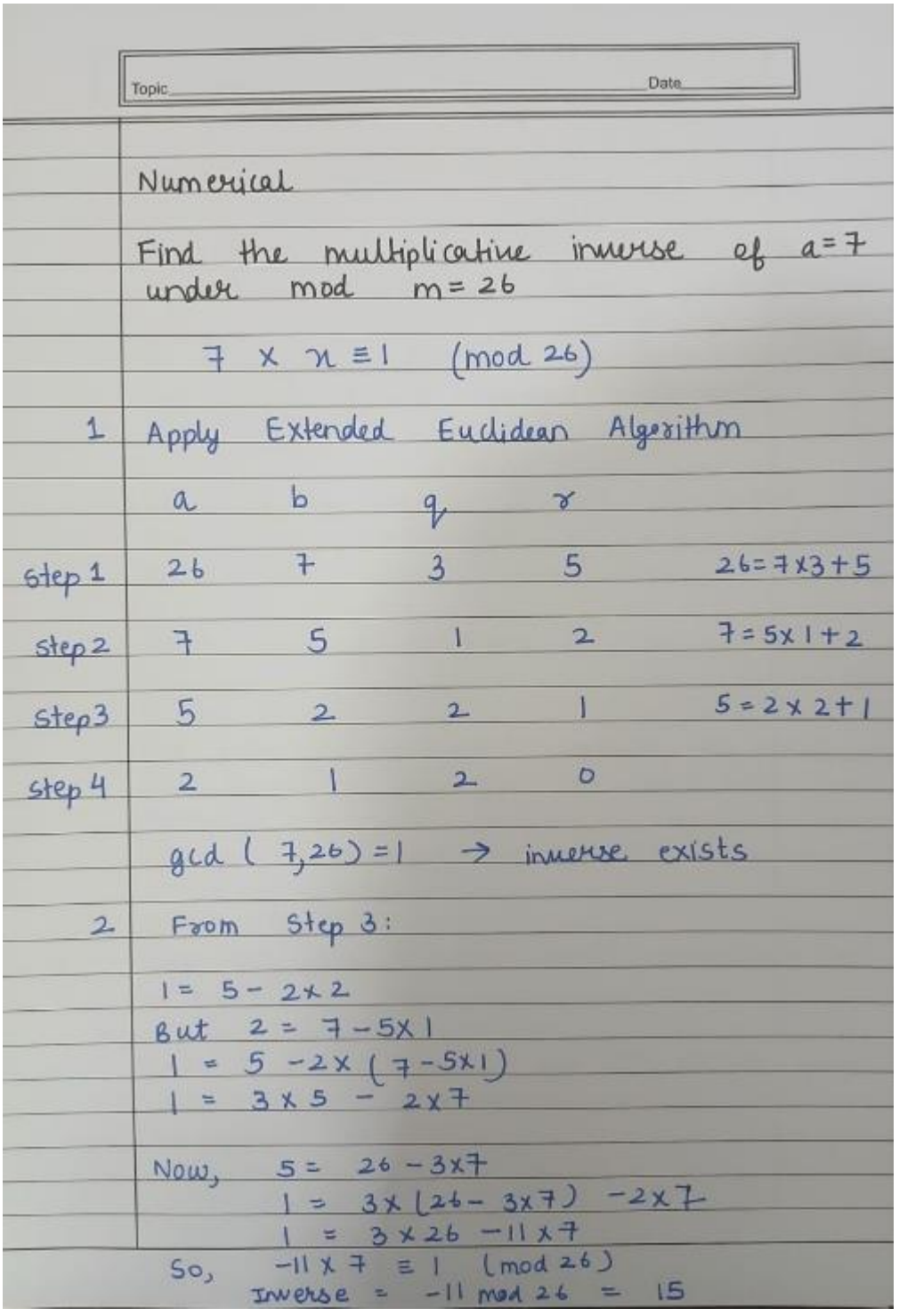
Methods to find multiplicative inverse

1. Trial and Error (for small numbers)
   Check all integers x and such that
   $(a \times x) \mod m = 1$
2. Extended Euclidean Algorithm
3. Fermat's Little theorem

   When m is prime,

   $a^{m-2} \equiv a^{-1} \pmod{m}$

Numerical Problem



Topic _____ Date _____

Numerical

Find the multiplicative inverse of a=7 under mod m=26

$7 \times x \equiv 1 \pmod{26}$

1 Apply Extended Euclidean Algorithm

	a	b	q	r	
step 1	26	7	3	5	26=7x3+5
step 2	7	5	1	2	7=5x1+2
step 3	5	2	2	1	5=2x2+1
step 4	2	1	2	0	

gcd (7,26) = 1 → inverse exists

2 From Step 3:

1 = 5 − 2x2

But 2 = 7 − 5x1

1 = 5 − 2x(7 − 5x1)

1 = 3x5 − 2x7

Now, 5 = 26 − 3x7

1 = 3x(26 − 3x7) − 2x7

1 = 3x26 − 11x7

So, $-11 \times 7 \equiv 1 \pmod{26}$

Inverse = −11 mod 26 = 15

Code

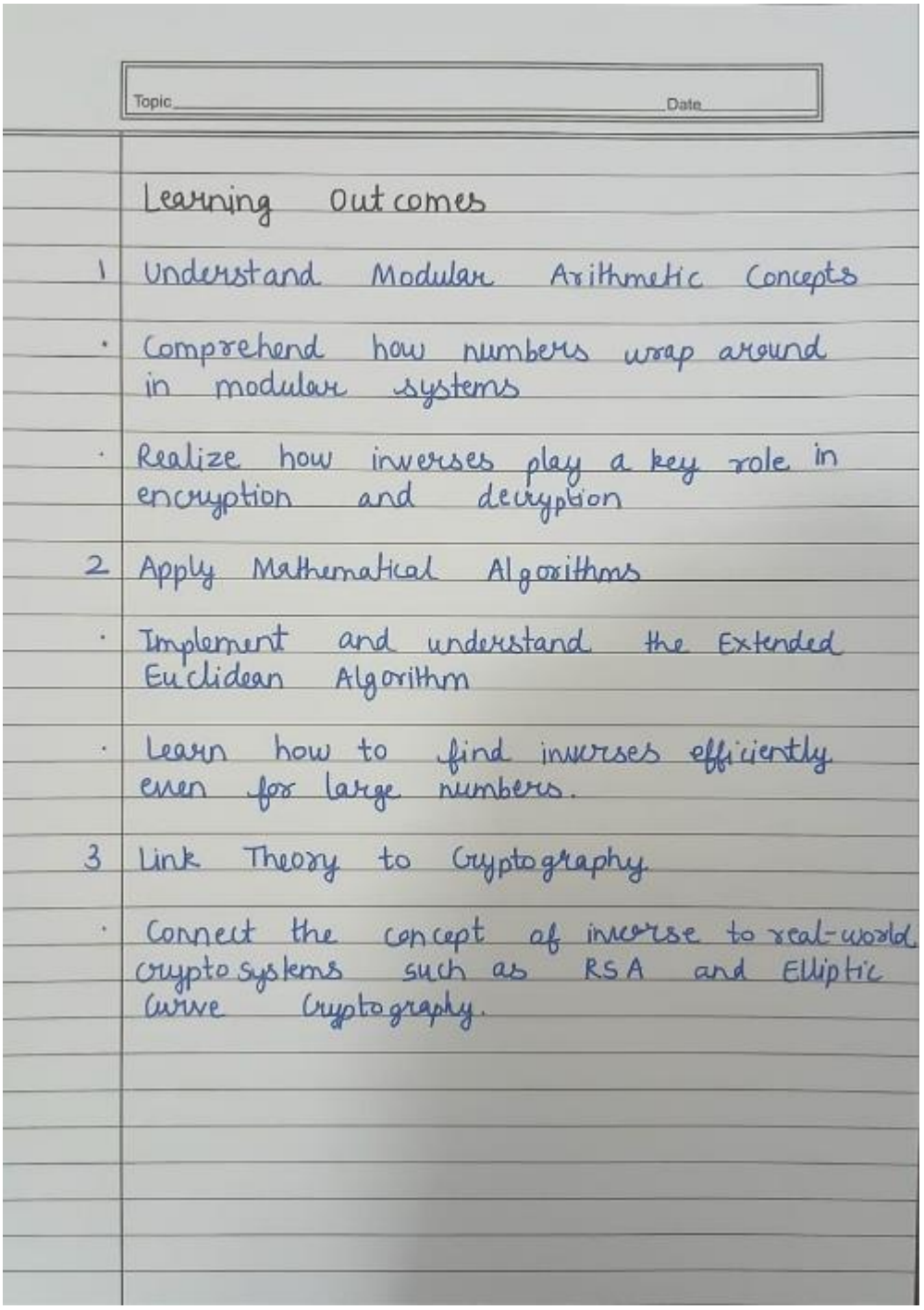
```cpp
#include <iostream>
using namespace std;

int modInverse(int a, int m) {
    a = a % m;
    for (int x = 1; x < m; x++) {
        if ((a * x) % m == 1)
            return x;
    }
    return -1;
}

int main() {
    int a = 7, m = 26;
    int inv = modInverse(a, m);

    if (inv == -1)
        cout << "No multiplicative inverse exists for " << a << " under mod "
            << m << endl;
    else
        cout << "Multiplicative inverse of " << a << " under mod " << m << "
            is: " << inv << endl;

    return 0;
}
```

Output

Your Output

```
Multiplicative inverse of 7 under mod 26 is: 15
```

Learning Outcome



Topic _____ Date _____

Learning Outcomes

1 Understand Modular Arithmetic Concepts

- Comprehend how numbers wrap around in modular systems
- Realize how inverses play a key role in encryption and decryption

2 Apply Mathematical Algorithms

- Implement and understand the Extended Euclidean Algorithm
- Learn how to find inverses efficiently even for large numbers.

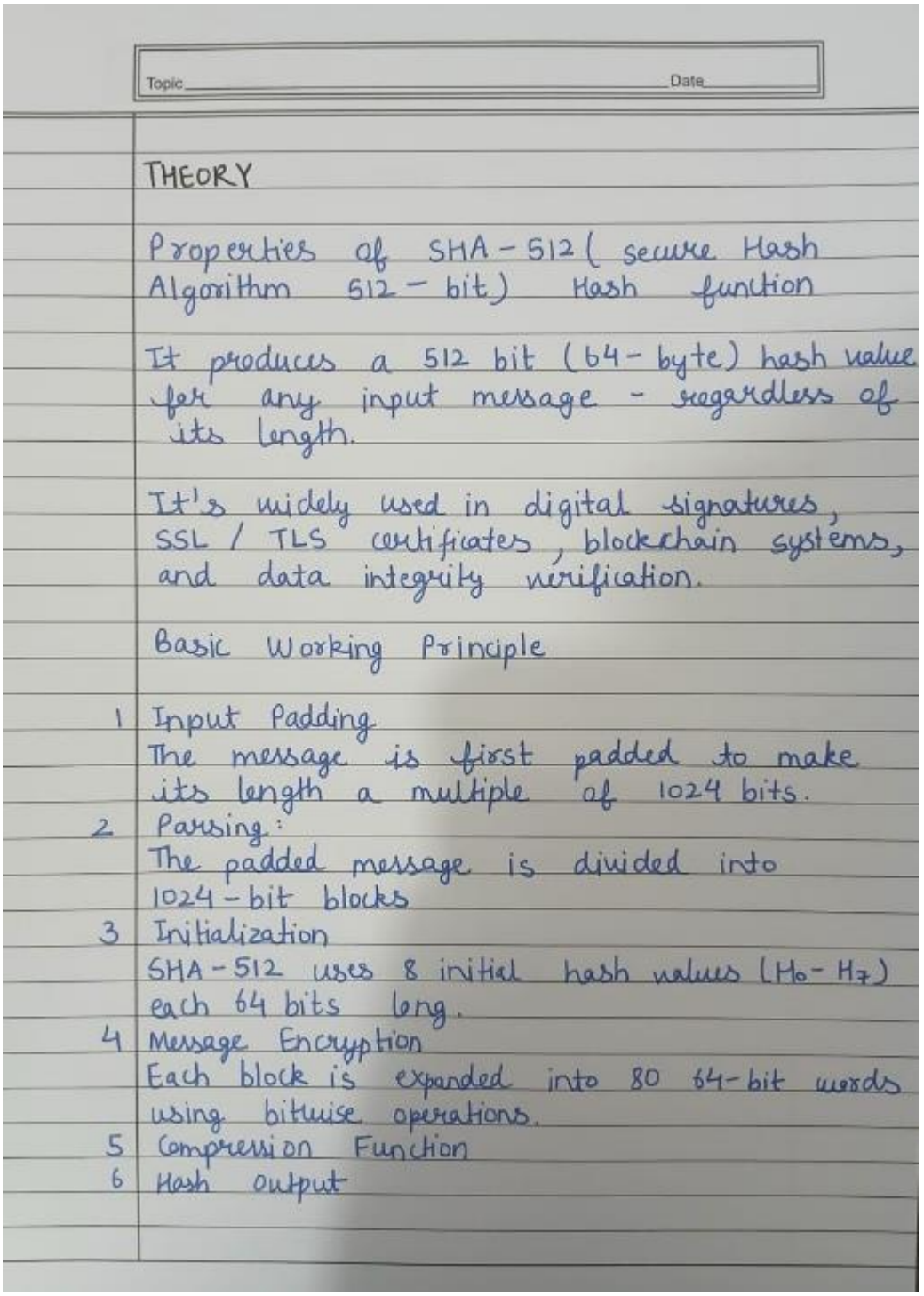
3 Link Theory to Cryptography

- Connect the concept of inverse to real-world crypto systems such as RSA and Elliptic Curve Cryptography.

EXPERIMENT NO. – 7

Aim Write a program that demonstrates the critical properties of the SHA-512 hash function.

Theory



Topic _____ Date _____

THEORY

Properties of SHA-512 (secure Hash Algorithm 512 - bit) Hash function

It produces a 512 bit (64-byte) hash value for any input message - regardless of its length.

It's widely used in digital signatures, SSL / TLS certificates, blockchain systems, and data integrity verification.

Basic Working Principle

1 Input Padding
The message is first padded to make its length a multiple of 1024 bits.

2 Parsing:
The padded message is divided into 1024-bit blocks

3 Initialization
SHA-512 uses 8 initial hash values (H_0 - H_7) each 64 bits long.

4 Message Encryption
Each block is expanded into 80 64-bit words using bitwise operations.

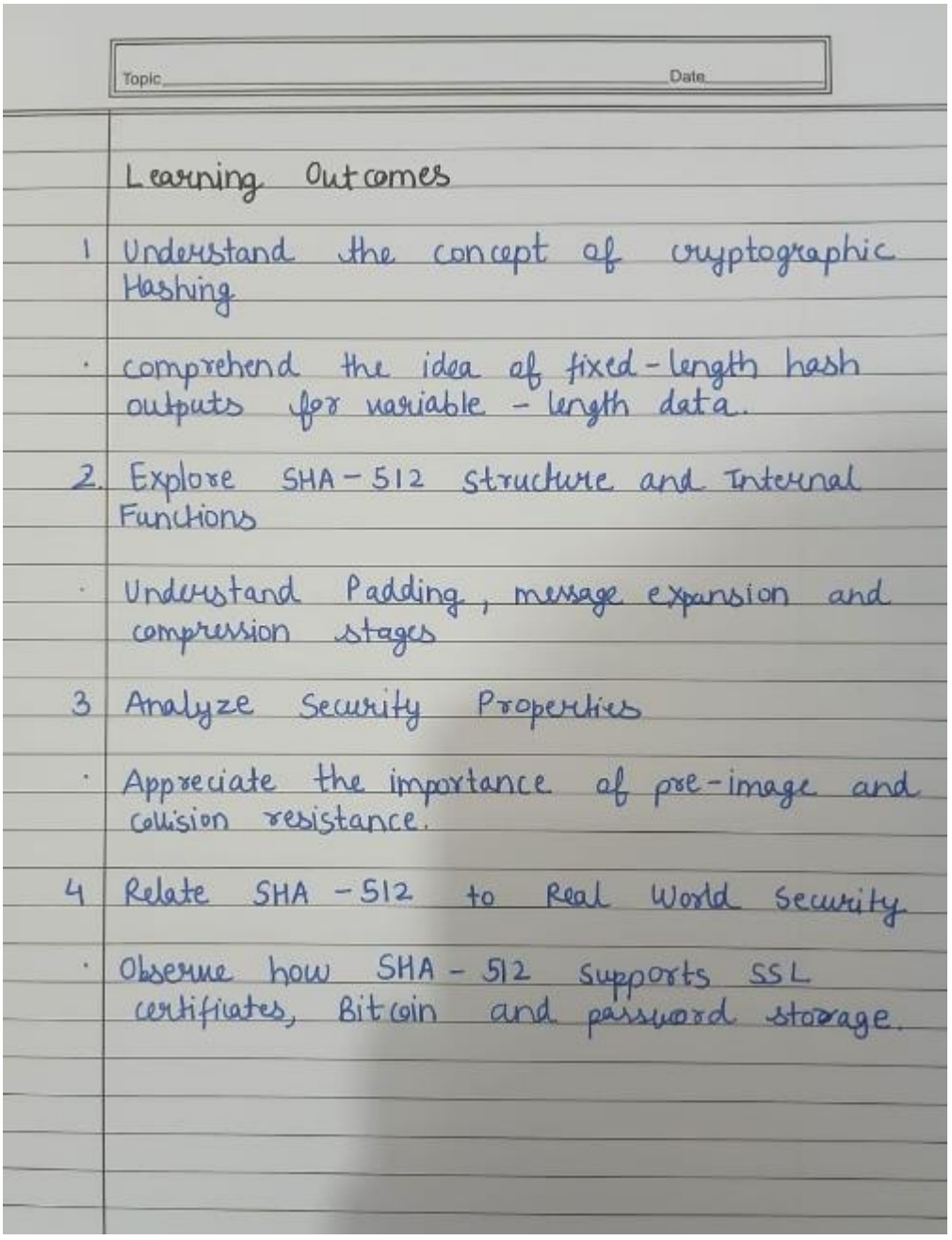
5 Compression Function

6 Hash output

Code

```cpp
#include <iostream>
#include <iomanip>
#include <cstring>
#include <openssl/sha.h>
using namespace std;

void sha512(const string &msg) {
    unsigned char hash[SHA512_DIGEST_LENGTH];
    SHA512((const unsigned char*)msg.c_str(), msg.length(), hash);

    cout << "Message: " << msg << endl;
    cout << "SHA-512: ";
    for (int i = 0; i < SHA512_DIGEST_LENGTH; i++)
        cout << hex << setw(2) << setfill('0') << (int)hash[i];
    cout << endl << endl;
}

int main() {
    string msg1 = "hello";
    string msg2 = "Hello";

    sha512(msg1);
    sha512(msg2);

    return 0;
}
```

Learning Outcome



Topic _____ Date _____

Learning Outcomes

1 Understand the concept of cryptographic Hashing

- comprehend the idea of fixed-length hash outputs for variable-length data.

2. Explore SHA-512 structure and Internal Functions

- Understand Padding, message expansion and compression stages

3 Analyze Security Properties

- Appreciate the importance of pre-image and collision resistance.

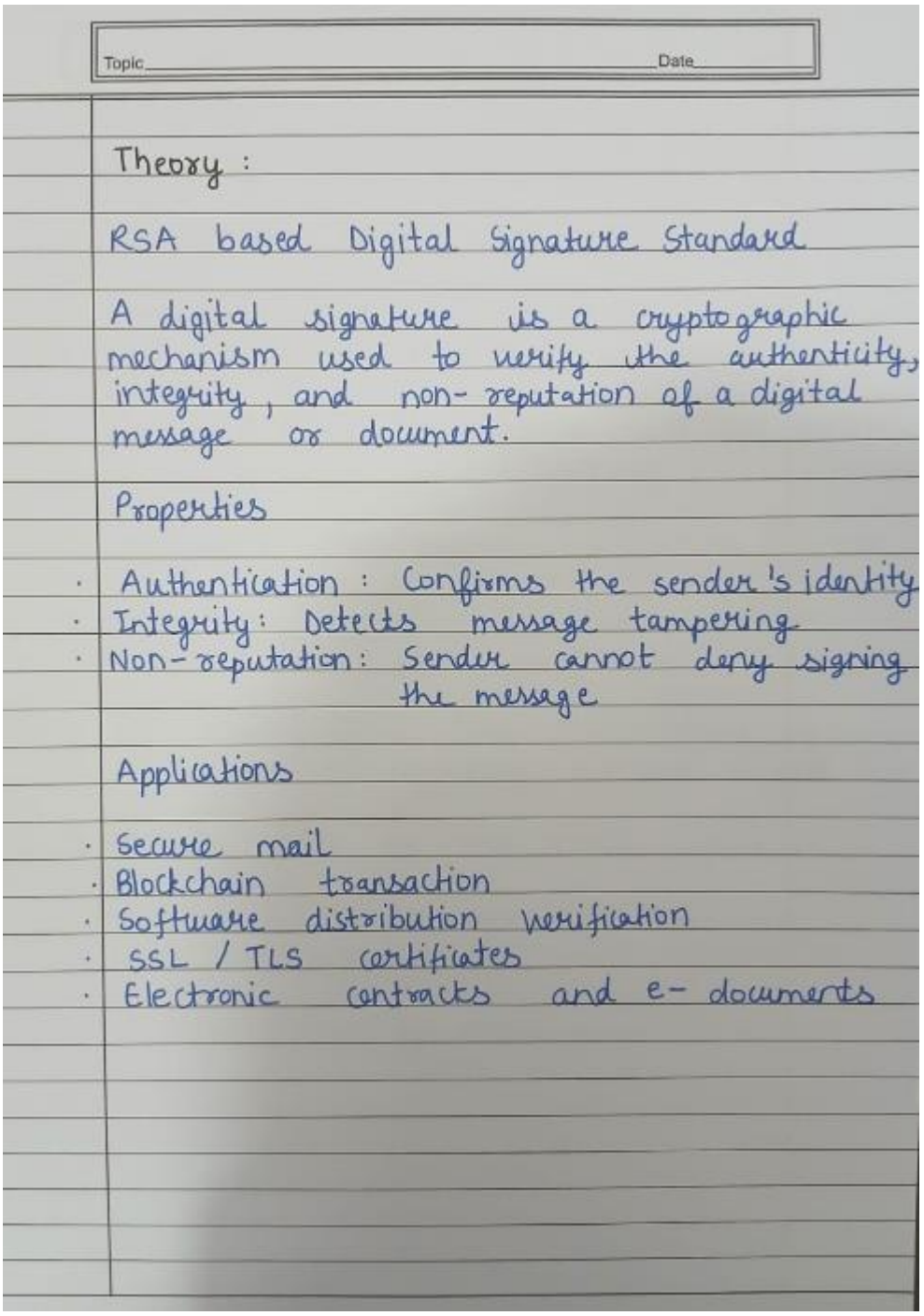
4 Relate SHA-512 to Real World Security

- Observe how SHA-512 supports SSL certificates, Bitcoin and password storage.

EXPERIMENT NO. – 8

Aim Write a program to Implementation of Digital Signature Standard (DSS) using RSA based digital signature scheme.

Theory



Topic_____ Date_____

Theory :

RSA based Digital Signature Standard

A digital signature is a cryptographic mechanism used to verify the authenticity, integrity, and non-reputation of a digital message or document.

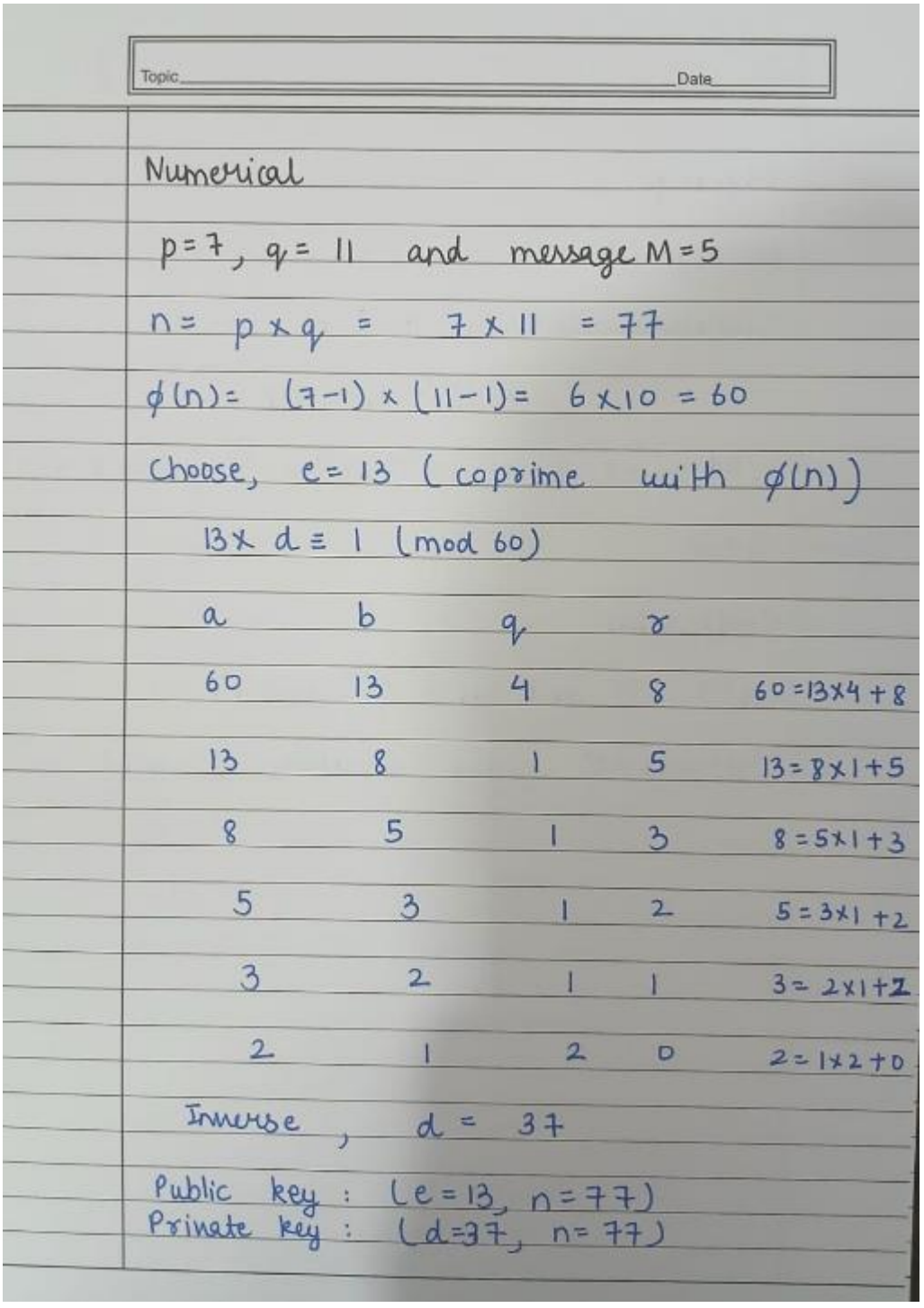
Properties

- Authentication : Confirms the sender's identity
- Integrity : Detects message tampering
- Non-reputation : Sender cannot deny signing the message

Applications

- Secure mail
- Blockchain transaction
- Software distribution verification
- SSL / TLS certificates
- Electronic contracts and e-documents

Numerical Problem



Topic _____ Date _____

Numerical

$p = 7$, $q = 11$ and message $M = 5$

$n = p \times q = 7 \times 11 = 77$

$\phi(n) = (7-1) \times (11-1) = 6 \times 10 = 60$

Choose, $e = 13$ (coprime with $\phi(n)$)

$13 \times d \equiv 1 \pmod{60}$

a	b	q	r	
60	13	4	8	$60 = 13 \times 4 + 8$
13	8	1	5	$13 = 8 \times 1 + 5$
8	5	1	3	$8 = 5 \times 1 + 3$
5	3	1	2	$5 = 3 \times 1 + 2$
3	2	1	1	$3 = 2 \times 1 + 1$
2	1	2	0	$2 = 1 \times 2 + 0$

Inverse, $d = 37$

Public key : $(e = 13, n = 77)$
Private key : $(d = 37, n = 77)$

Topic _____ Date _____

Signing

Message $M = 5$

Signature $S = M^d \bmod n$

$= 5^{37} \bmod 77$

$5^{37} \bmod 77 = 5^{32} \times 5^4 \times 5^1 \bmod 77 = 5$

$S = 5$

Verification

$H' = S^e \bmod n = 5^{13} \bmod 77 = 5$

$H' = M = 5$, the signature is valid

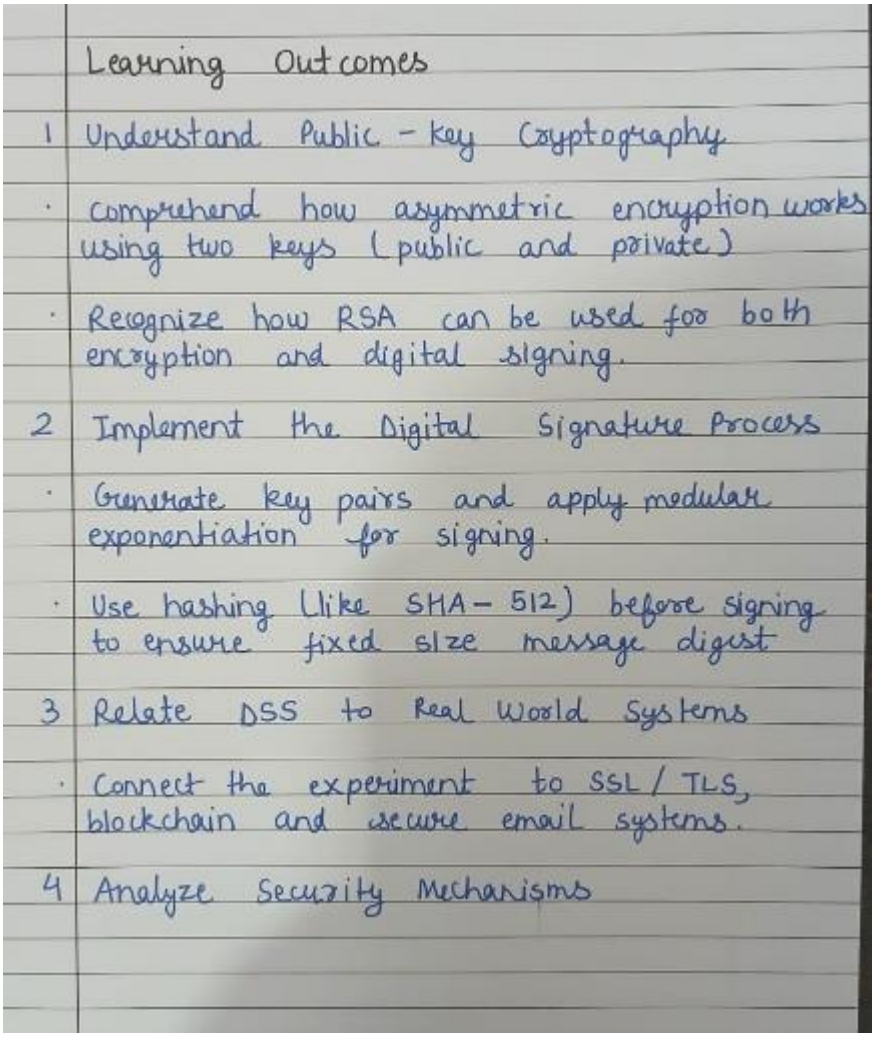
Code

```cpp
// Experiment 8: Digital Signature using RSA (C++)
#include <iostream>
#include <cmath>
using namespace std;

long long powerMod(long long base, long long exp, long long mod) {
    long long result = 1;
    base %= mod;
    while (exp > 0) {
        if (exp % 2 == 1)
            result = (result * base) % mod;
        base = (base * base) % mod;
        exp /= 2;
    }
    return result;
}

int main() {
    long long p = 7, q = 11;
    long long n = p * q;
    long long phi = (p - 1) * (q - 1);
    long long e = 13, d = 37;
    long long message = 5;

    long long signature = powerMod(message, d, n);
    long long verify = powerMod(signature, e, n);

    cout << "Public key (e, n): (" << e << ", " << n << ")\n";
    cout << "Private key (d, n): (" << d << ", " << n << ")\n";
    cout << "Message: " << message << endl;
    cout << "Signature: " << signature << endl;
    cout << "Verification result: " << verify << endl;

    if (verify == message)
        cout << "✅ Signature verified successfully!\n";
    else
        cout << "❌ Verification failed.\n";

    return 0;
}
```

Output

```
Your Output

Public key (e, n): (13, 77)
Private key (d, n): (37, 77)
Message: 5
Signature: 47
Verification result: 5
✅ Signature verified successfully!
```

Learning Outcome



Learning Outcomes

1 Understand Public - Key Cryptography

- Comprehend how asymmetric encryption works using two keys (public and private)
- Recognize how RSA can be used for both encryption and digital signing.

2 Implement the Digital Signature Process

- Generate key pairs and apply modular exponentiation for signing.
- Use hashing (like SHA - 512) before signing to ensure fixed size message digest

3 Relate DSS to Real World Systems

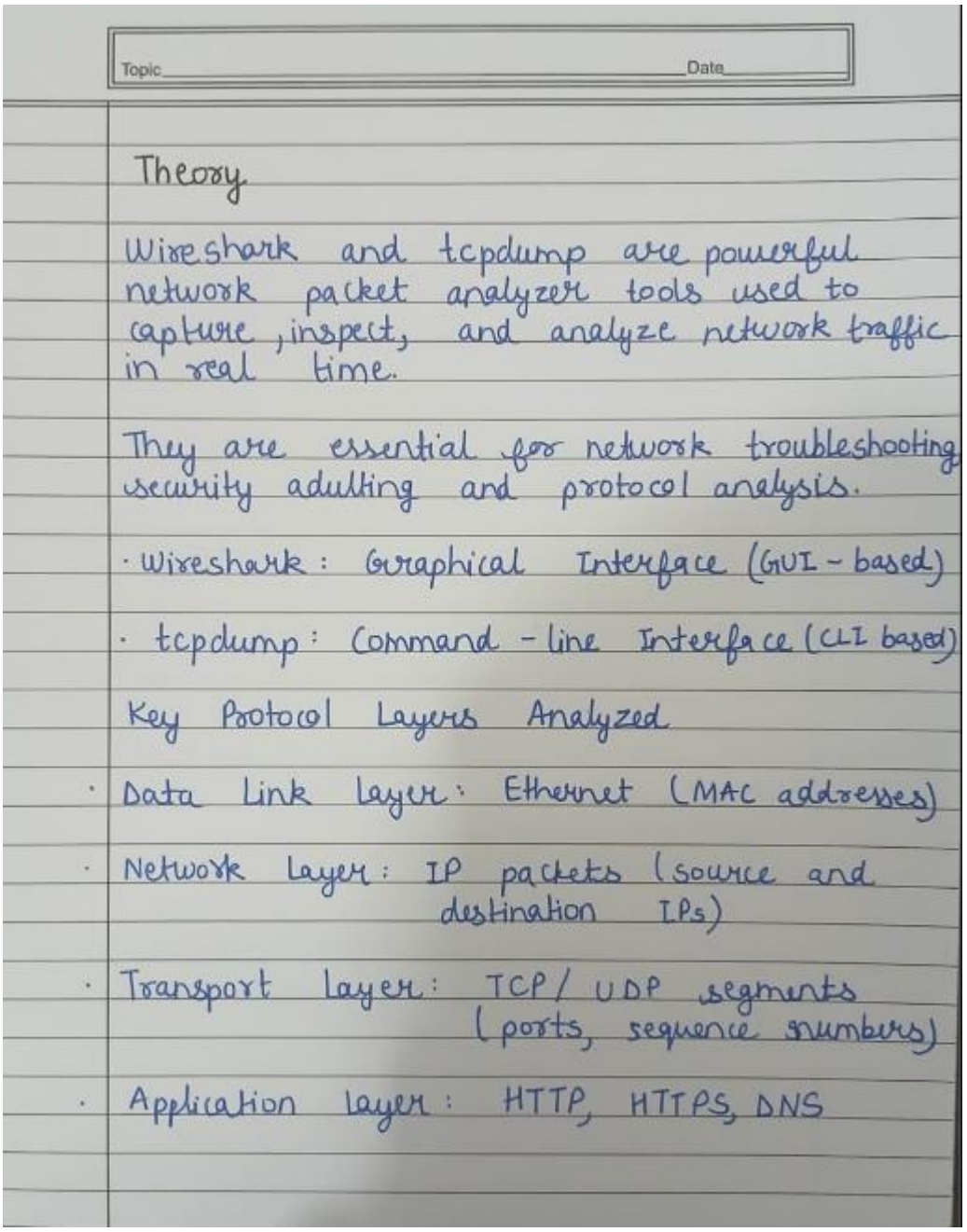
- Connect the experiment to SSL / TLS, blockchain and secure email systems.

4 Analyze Security Mechanisms

EXPERIMENT NO. – 9

Aim Installation and usage of Wireshark or tcpdump to capture and analyze network traffic (UDP/TCP/HTTP sessions).

Theory



Topic _____ Date _____

Theory

Wireshark and tcpdump are powerful network packet analyzer tools used to capture, inspect, and analyze network traffic in real time.

They are essential for network troubleshooting security auditing and protocol analysis.

- Wireshark: Graphical Interface (GUI - based)
- tcpdump: Command - line Interface (CLI based)

Key Protocol Layers Analyzed

- Data Link layer: Ethernet (MAC addresses)
- Network Layer: IP packets (source and destination IPs)
- Transport Layer: TCP / UDP segments (ports, sequence numbers)
- Application layer: HTTP, HTTPS, DNS

Code

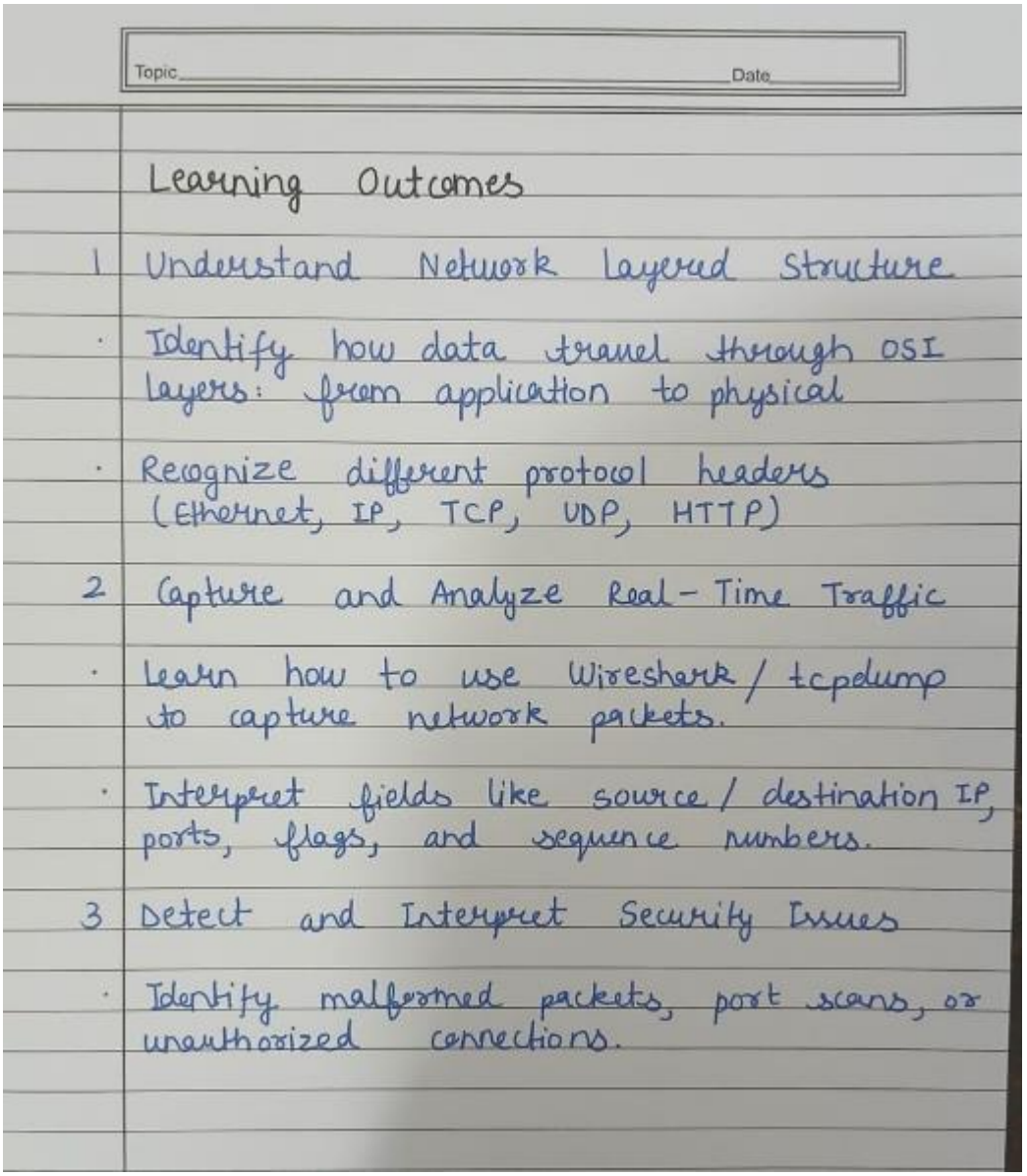
```cpp
// Experiment 9: Simulated network packet capture (C++)
#include <iostream>
#include <vector>
using namespace std;

struct Packet {
    string src_ip, dst_ip, protocol;
};

int main() {
    vector<Packet> packets = {
        {"192.168.1.2", "192.168.1.5", "TCP"},
        {"192.168.1.3", "192.168.1.5", "UDP"},
        {"192.168.1.2", "192.168.1.6", "HTTP"}
    };

    cout << "All captured packets:\n";
    for (auto &p : packets)
        cout << p.src_ip << " -> " << p.dst_ip << " (" << p.protocol << ")\n"
             ;

    cout << "\nFiltered packets (TCP only):\n";
    for (auto &p : packets)
        if (p.protocol == "TCP")
            cout << p.src_ip << " -> " << p.dst_ip << " (" << p.protocol << "
                )\n";

    return 0;
}
```

Output

Your Output

```
All captured packets:
192.168.1.2 -> 192.168.1.5 (TCP)
192.168.1.3 -> 192.168.1.5 (UDP)
192.168.1.2 -> 192.168.1.6 (HTTP)

Filtered packets (TCP only):
192.168.1.2 -> 192.168.1.5 (TCP)
```

Learning Outcome



Topic _____ Date _____

Learning Outcomes

1 Understand Network Layered Structure

- Identify how data travel through OSI layers: from application to physical
- Recognize different protocol headers (Ethernet, IP, TCP, UDP, HTTP)

2 Capture and Analyze Real-Time Traffic

- Learn how to use Wireshark / tcpdump to capture network packets.
- Interpret fields like source / destination IP, ports, flags, and sequence numbers.

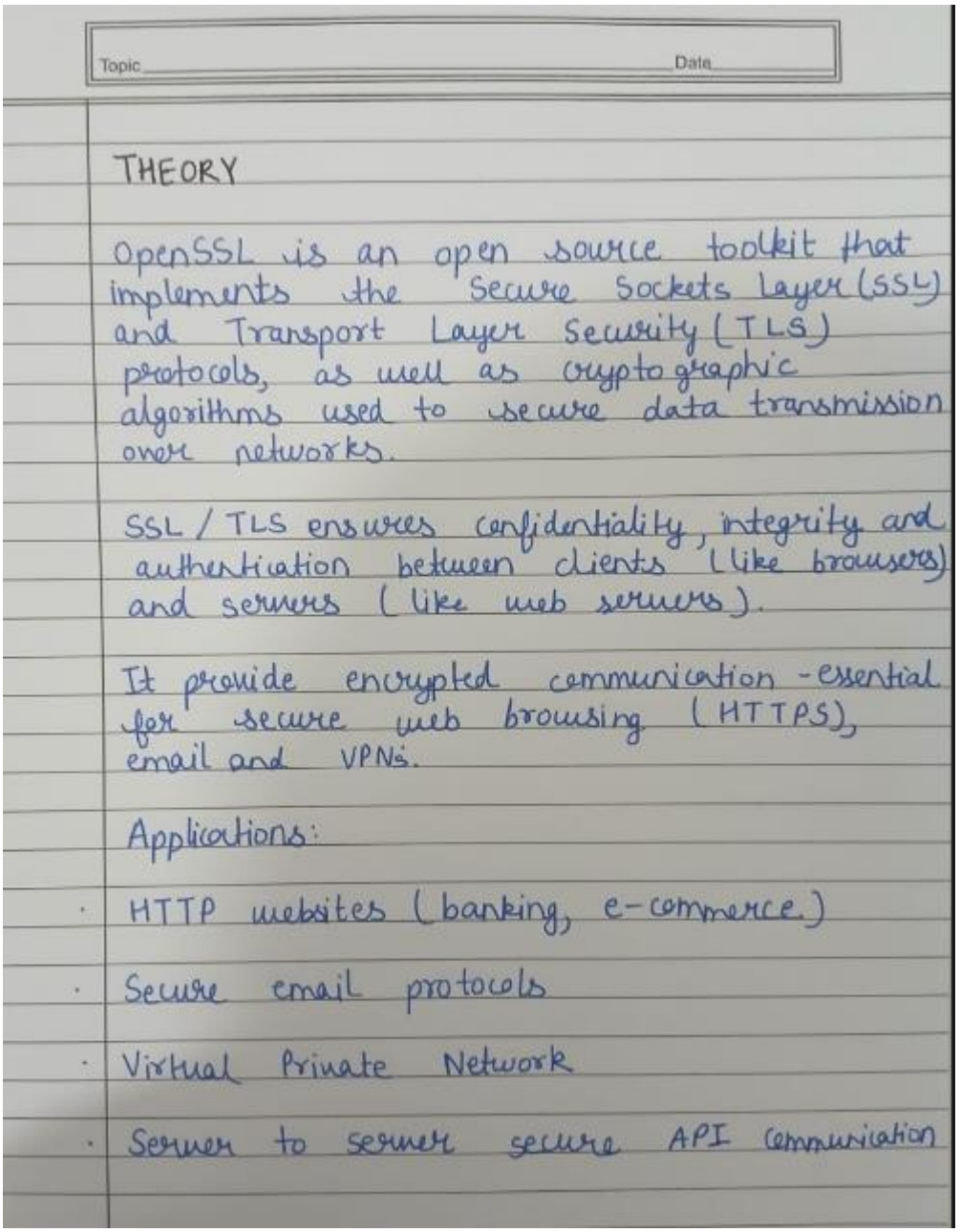
3 Detect and Interpret Security Issues

- Identify malformed packets, port scans, or unauthorized connections.

EXPERIMENT NO. – 10

Aim Experimenting with OpenSSL to set up a secure web server-browser communication channel (SSL/TLS).

Theory



Topic _____ Date _____

THEORY

OpenSSL is an open source toolkit that implements the Secure Sockets Layer (SSL) and Transport Layer Security (TLS) protocols, as well as cryptographic algorithms used to secure data transmission over networks.

SSL / TLS ensures confidentiality, integrity and authentication between clients (like browsers) and servers (like web servers).

It provide encrypted communication - essential for secure web browsing (HTTPS), email and VPNs.

Applications:

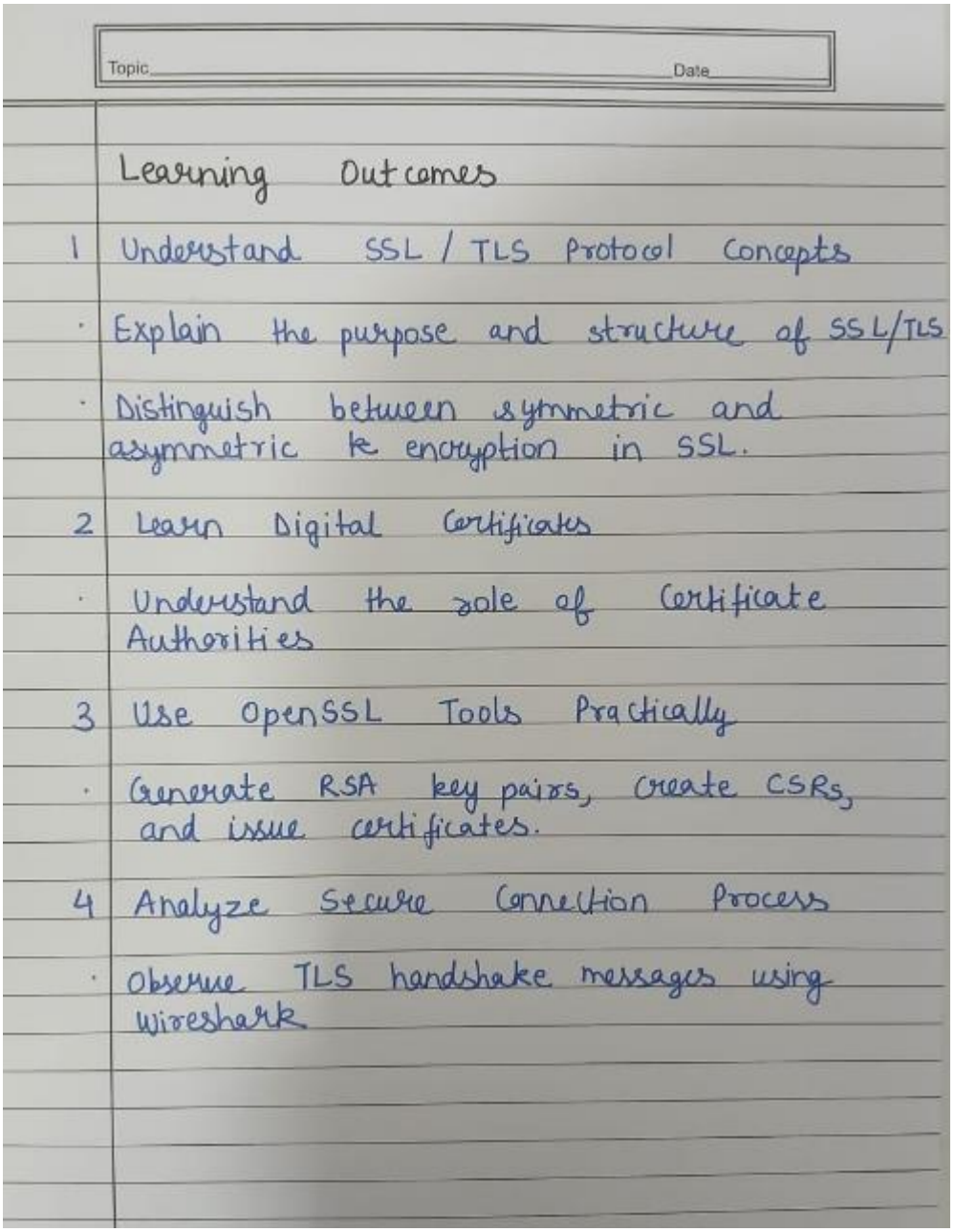
- HTTP websites (banking, e-commerce)
- Secure email protocols
- Virtual Private Network
- Server to server secure API communication

Code

```cpp
#include <iostream>
#include <openssl/ssl.h>
#include <openssl/err.h>
#include <netdb.h>
#include <unistd.h>
using namespace std;

int main() {
    SSL_library_init();
    SSL_load_error_strings();

    const SSL_METHOD *method = TLS_client_method();
    SSL_CTX *ctx = SSL_CTX_new(method);

    string hostname = "www.google.com";
    int port = 443;

    struct hostent *host = gethostbyname(hostname.c_str());
    int sock = socket(AF_INET, SOCK_STREAM, 0);

    struct sockaddr_in addr;
    addr.sin_family = AF_INET;
    addr.sin_port = htons(port);
    addr.sin_addr.s_addr = *(long*)(host->h_addr);

    connect(sock, (struct sockaddr*)&addr, sizeof(addr));

    SSL *ssl = SSL_new(ctx);
    SSL_set_fd(ssl, sock);
    if (SSL_connect(ssl) == 1) {
        cout << "✅ SSL/TLS connection established successfully!\n";
        cout << "Cipher: " << SSL_get_cipher(ssl) << endl;
    } else {
        cout << "❌ SSL connection failed.\n";
    }

    SSL_shutdown(ssl);
    SSL_free(ssl);
    close(sock);
    SSL_CTX_free(ctx);

    return 0;
}
```

Output

Learning Outcome



Topic _____ Date _____

Learning Outcomes

1 Understand SSL / TLS Protocol Concepts

- Explain the purpose and structure of SSL/TLS
- Distinguish between symmetric and asymmetric & encryption in SSL.

2 Learn Digital Certificates

- Understand the role of Certificate Authorities

3 Use OpenSSL Tools Practically

- Generate RSA key pairs, create CSRs, and issue certificates.

4 Analyze Secure Connection Process

- Observe TLS handshake messages using Wireshark.