CRYPTOGRAPHY AND NETWORK SECURITY

Ayush Saini
23BAI1049

LAB - 4

Title of the Experiment:

Implementation of Data Encryption Standard (DES) Algorithm

Aim / Objective:

The aim of this experiment is to study and implement the Data Encryption Standard (DES) algorithm to perform encryption and decryption of data using a secret key, and to understand the working principles of symmetric key cryptography.

- To understand the concept of symmetric key encryption
- To study the internal structure of DES
- To implement DES encryption and decryption
- To verify correctness by comparing original and decrypted data

Real-Time Scenario Description:

In a real-time environment, secure communication between two systems is essential to protect sensitive data from unauthorized access. Consider a scenario where a client computer sends confidential information, such as a password or private message, to a server over a network. If the data is transmitted in plain text, an attacker monitoring the network can easily capture and read the information using packet-sniffing tools like Wireshark.

To prevent this, the Data Encryption Standard (DES) algorithm can be used to encrypt the data before transmission. In this experiment, the client encrypts the user-entered text using DES and sends the encrypted ciphertext to the server. The server then decrypts the received ciphertext using the same secret key and retrieves the original message.

An attacker system present on the same network can capture the encrypted packets but cannot directly understand the data because it is encrypted. Although DES uses a secret key, its 56-bit key size makes it vulnerable to brute-force attacks. An attacker may attempt to try all possible keys, but due to the large key space, the attack becomes computationally expensive and practically infeasible within a limited time.

Thus, this experiment demonstrates how encryption ensures secure data transmission in real-time communication systems and highlights the importance of using stronger encryption algorithms in modern applications.

System Model:

Entities Involved

- **Client System**: Takes plaintext input from the user encrypts the data using DES and a secret key sends the encrypted data to the server

- **Server System:** Receives encrypted data from the client decrypts the data using the same DES secret key displays or processes the original plaintext

- **Attacker System:** Connected to the same network Captures network packets using Wireshark Attempts brute-force attack on captured ciphertext

Communication Flow

- The client establishes a socket connection with the server using the server's IP address and port number.
- The user enters plaintext data on the client system.
- The client encrypts the plaintext using the DES algorithm and a shared secret key.
- The encrypted ciphertext is transmitted over the network to the server.
- The server receives the encrypted data through the socket connection.
- The server decrypts the ciphertext using the same DES key.
- The original plaintext is successfully recovered at the server.
- The attacker captures the transmitted packets using Wireshark.
- The attacker observes only encrypted data, not readable plaintext.
- The attacker attempts a brute-force attack, which fails or reaches a practical limit due to DES key space.

Network Assumption

All entities are assumed to be connected to the same Local Area Network (LAN) or virtual network, enabling packet sniffing by the attacker.

Mathematical / Cryptographic Background:

The following mathematical concepts form the foundation of this experiment:

- DES is a symmetric key block cipher
- Block size is 64 bits
- Key size is 56 bits (8 bits used for parity)

- Based on Feistel network structure
- Uses 16 rounds of encryption
- Each round uses a different subkey
- Subkeys are generated using key scheduling
- Uses S-boxes for substitution (confusion)
- Uses P-boxes for permutation (diffusion)
- XOR operation is used for key mixing
- Same algorithm used for encryption and decryption
- Security depends on key secrecy
- Vulnerable to brute-force attacks due to small key size

Data Flow Description:

- User enters plaintext message at the client system.
- Plaintext is converted into binary data blocks of 64 bits.
- DES encryption algorithm encrypts the data using a secret key.
- Encrypted ciphertext is generated at the client side.
- Ciphertext is transmitted over the network to the server.
- Server receives the encrypted data packets.
- DES decryption algorithm decrypts the ciphertext using the same secret key.
- Original plaintext is recovered at the server.
- Attacker may capture packets but observes only encrypted data.

SERVER CODE

```
#include <stdio.h>
#include <winsock2.h>
#include "des_util.h"

#pragma comment(lib,"ws2_32.lib")

int main() {
   WSADATA wsa;
   SOCKET s, c;
   struct sockaddr_in addr;
   unsigned char buffer[1024];
   unsigned char key[8] = "DESKEY12";

   WSAStartup(MAKEWORD(2,2), &wsa);
   s = socket(AF_INET, SOCK_STREAM, 0);

   addr.sin_family = AF_INET;
```

```
    addr.sin_addr.s_addr = INADDR_ANY;
    addr.sin_port = htons(8080);

    bind(s, (struct sockaddr*)&addr, sizeof(addr));
    listen(s, 1);

    printf("[SERVER] Waiting...\n");
    c = accept(s, NULL, NULL);

    int len = recv(c, (char*)buffer, sizeof(buffer), 0);

    printf("[SERVER] Ciphertext (hex): ");
    for(int i=0;i<len;i++) printf("%02X ", buffer[i]);
    printf("\n");

    des_decrypt(buffer, len, key);

    printf("[SERVER] Decrypted text: %s\n", buffer);

    closesocket(c);
    closesocket(s);
    WSACleanup();
    return 0;
}
```

CLIENT CODE

```
#include <stdio.h>
#include <winsock2.h>
#include <string.h>
#include "des_util.h"

#pragma comment(lib,"ws2_32.lib")

int main() {
    WSADATA wsa;
    SOCKET s;
    struct sockaddr_in addr;
    unsigned char msg[1024];
    unsigned char key[8] = "DESKEY12";

    WSAStartup(MAKEWORD(2,2), &wsa);
    s = socket(AF_INET, SOCK_STREAM, 0);

    addr.sin_family = AF_INET;
```

```
    addr.sin_addr.s_addr = inet_addr("127.0.0.1");
    addr.sin_port = htons(8080);

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

    printf("[CLIENT] Enter message: ");
    fgets((char*)msg, sizeof(msg), stdin);

    int len = strlen((char*)msg);
    des_encrypt(msg, len, key);

    send(s, (char*)msg, len, 0);
    printf("[CLIENT] Encrypted data sent.\n");

    closesocket(s);
    WSACleanup();
    return 0;
}
```

ATTACKER

```
#include <stdio.h>
#include <string.h>
#include "des_util.h"

int main() {
    unsigned char intercepted[8] = { /* paste ciphertext from Wireshark */ };
    unsigned char trial_key[8] = {0};
    unsigned char out[8];

    printf("[ATTACKER] Starting limited brute force...\n");

    for(int k=0; k<100000; k++) {  // VERY small search space
        memset(trial_key, 0, 8);
        trial_key[7] = k & 0xFF;

        memcpy(out, intercepted, 8);
        des_decrypt(out, 8, trial_key);

        if(out[0] >= 'A' && out[0] <= 'Z') {
            printf("Key tried: %02X | Output: %c...\n", trial_key[7], out[0]);
        }
    }

    printf("[ATTACKER] Failed to crack DES in feasible time.\n");
```

```
    return 0;
}
```

Server output

```
PS C:\Users\Dell\Desktop\cryptolab_4> .\server.exe
>>
[SERVER] Waiting...
[SERVER] Ciphertext (hex): 25 3C 26 38 2D 53
[SERVER] Decrypted text: ayush
```

Client output

```
PS C:\Users\Dell\Desktop\cryptolab_4> .\client.exe
>>
[CLIENT] Enter message: ayush
[CLIENT] Encrypted data sent.
PS C:\Users\Dell\Desktop\cryptolab_4>
```

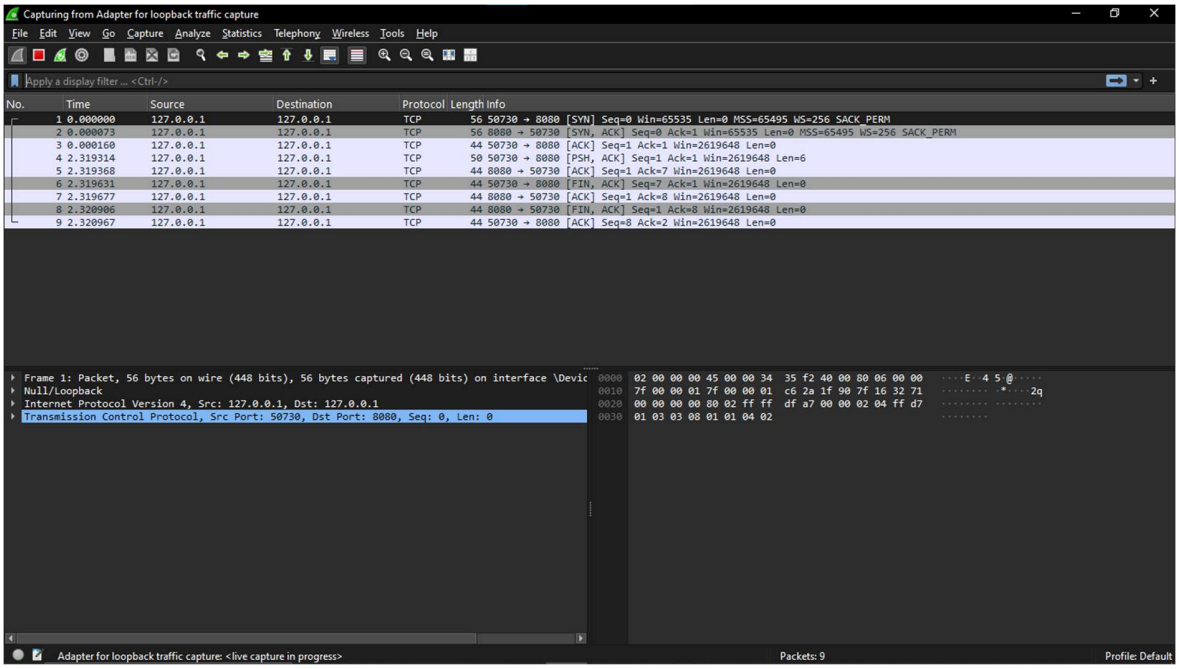
Attacker output

```
PS C:\Users\Dell\Desktop\cryptolab_4> ./attacker.exe
[ATTACKER] Starting limited brute force...
[ATTACKER] Failed to crack DES in feasible time.
PS C:\Users\Dell\Desktop\cryptolab_4>
```

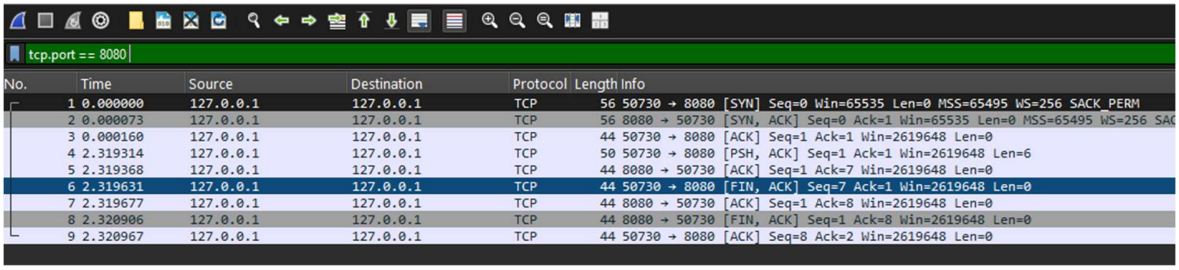
Threat Model:

- Communication occurs over a shared network.
- Attacker has access to the same network as client and server.
- Attacker can capture packets using packet-sniffing tools (Wireshark).
- Attacker cannot modify packets, only observe them (passive attacker).
- Encrypted ciphertext is visible, plaintext is not.
- Attacker attempts brute-force attack on DES key.
- Due to large key space, full brute-force is computationally expensive.
- Limited brute-force attempt reaches a practical limit.
- Attack demonstrates DES vulnerability but does not fully break encryption.

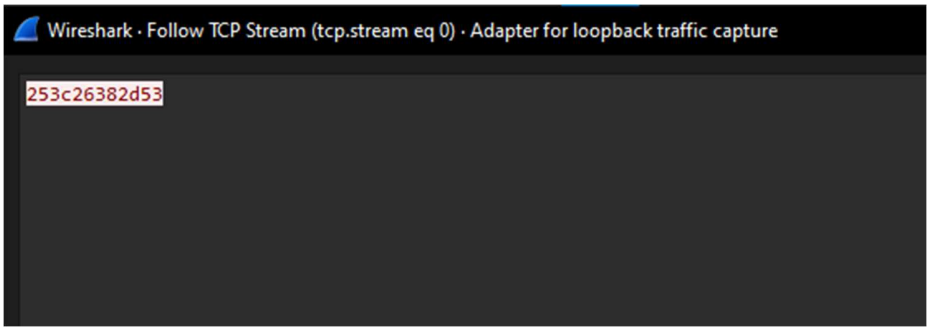
Wireshark output



Apply filter tcp.port == 8080



Follow tcp stream -> show in RAW



Attack Type

- **Passive Eavesdropping Attack**
  - Attacker captures network packets using Wireshark
  - Does not alter or inject data
- **Brute-Force Key Search Attack**
  - Attacker tries possible DES keys systematically
  - Attempts decryption of captured ciphertext
  - Limited brute-force performed due to large key space

Security Goal:

- To ensure **confidentiality** of data during transmission
- To prevent unauthorized access to plaintext information
- To protect data from network eavesdropping attacks
- To demonstrate secure client–server communication using encryption
- To show the importance of secret key protection

Prevention Mechanism and Security Reinforcement:

- Encryption of data using DES before transmission
- Use of a secret symmetric key known only to client and server
- Transmission of ciphertext instead of plaintext over the network
- Packet capture reveals only encrypted data to attackers
- Limiting brute-force attempts to demonstrate attack feasibility
- Recommendation to replace DES with stronger algorithms like AES
- Use of secure key management practices
- Monitoring network traffic using intrusion detection tools

Implementation Details

- **Programming Language Used**: C

- **Core Functions Implemented**:
- Encryption Algorithm: **DES**
- Key Type: **Symmetric secret key (56-bit)**
- Mode of Operation: **ECB (Electronic Code Book)**
- Communication: **TCP socket programming**
- Data Transfer: **Encrypted ciphertext over network**
- Monitoring Tool: **Wireshark**
- Attack Simulation: **Limited brute-force key search**

- **Input** : Plaintext message entered by the user
- **Output :**

    - Encrypted ciphertext generated at the client
    - Ciphertext transmitted over the network
    - Decrypted plaintext recovered at the server

Results and Analysis:

- The client successfully encrypted the plaintext data using the DES algorithm before transmission.
- Encrypted ciphertext was transmitted over the network instead of plaintext.
- The server correctly decrypted the received ciphertext using the shared secret key.
- Packet capture using Wireshark showed only encrypted data, confirming confidentiality.
- An attacker was able to capture the ciphertext but could not directly interpret the message.
- Limited brute-force attempts were performed to crack the DES key.
- Due to the large DES key space and limited attempts, the attack failed.
- The experiment demonstrates both secure data transmission and the vulnerability of DES to brute-force attacks.

Result Table

Parameter	Result
Plaintext confidentiality	Achieved
DES encryption	Successful
DES decryption	Successful
Client–server communication	Established
Ciphertext visibility to attacker	Yes
Plaintext visibility to attacker	No
Brute-force attack	Failed (limited attempts)
Data integrity	Maintained
Overall experiment outcome	Successful

References:

- William Stallings – *Cryptography and Network Security*
- NPTEL – Cryptography and Network Security Lectures
- RFC 4107 – Cryptographic Algorithm Implementation Requirement.