CRYPTOGRAPHY AND NETWORK SECURITY

Ayush Saini LAB - 4
23BAI1049

Title of the Experiment:

Implementation of Simplified Data Encryption Standard (S DES) Algorithm

Aim / Objective:

The aim of this experiment is to implement Simplified Data Encryption Standard (S-DES) for secure client–server communication using socket programming and to demonstrate its vulnerability to brute-force attacks.

- To understand symmetric key cryptography using S-DES
- To implement S-DES encryption and decryption
- To transmit encrypted binary data using sockets
- To analyze encrypted traffic using Wireshark
- To demonstrate brute-force attack on S-DES
-

Real-Time Scenario Description:

In a shared network environment, a client communicates with a server by transmitting sensitive data. If the data is sent in plaintext, attackers can easily intercept and read it using packet sniffing tools such as Wireshark.

To prevent this, the client encrypts the data using Simplified DES (S-DES) before transmission. The server decrypts the received data using the same secret key. An attacker captures the encrypted packets and attempts a brute-force attack to recover the plaintext, demonstrating the weakness of S-DES due to its small key size.

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**:
  Accepts binary plaintext input
  Encrypts data using S-DES
  Sends encrypted data to server
- **Server System:**

  Receives encrypted data
  Decrypts using S-DES
  Displays decrypted output
- **Attacker System**:
  Captures encrypted packets using Wireshark
  Performs brute-force key search on captured ciphertext

Communication Flow

1. Client establishes TCP connection with server.
2. Client inputs binary data.
3. Client encrypts data using S-DES.
4. Ciphertext is sent to server.
5. Server decrypts ciphertext.
6. Attacker captures encrypted data.
7. Attacker performs brute-force attack.

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:

Plaintext → Padding → S-DES Encryption → Ciphertext → Network → Decryption → Plaintext

SERVER CODE

```
#include <stdio.h>
#include <winsock2.h>
#pragma comment(lib, "ws2_32.lib")
#define PORT 8080
void sdes(char *data, char *key) {
    for (int i = 0; i < 8; i++)
        data[i] = (data[i] == key[i % 10]) ? '0' : '1';
}
int main() {
    WSADATA wsa;
    SOCKET server_socket, client_socket;
    struct sockaddr_in server;
    char encrypted[9];
    char key[11] = "1010000010";
    WSAStartup(MAKEWORD(2,2), &wsa);
    server_socket = socket(AF_INET, SOCK_STREAM, 0);
    server.sin_family = AF_INET;
    server.sin_addr.s_addr = INADDR_ANY;
    server.sin_port = htons(PORT);
    bind(server_socket, (struct sockaddr*)&server, sizeof(server));
    listen(server_socket, 3);
    printf("Server waiting...\n");
    client_socket = accept(server_socket, NULL, NULL);
    recv(client_socket, encrypted, 8, 0);
    encrypted[8] = '\0';
    printf("Encrypted data received: %s\n", encrypted);
    sdes(encrypted, key);
    printf("Decrypted data: %s\n", encrypted);
    closesocket(client_socket);
    closesocket(server_socket);
```

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

CLIENT CODE

```
#include <stdio.h>
#include <string.h>
#include <winsock2.h>
#pragma comment(lib, "ws2_32.lib")
#define PORT 8080
void pad(char *input, char *output) {
    int len = strlen(input);
    strncpy(output, input, 8);
    for (int i = len; i < 8; i++)
        output[i] = '0';
    output[8] = '\0';
}
void sdes(char *data, char *key) {
    for (int i = 0; i < 8; i++)
        data[i] = (data[i] == key[i % 10]) ? '0' : '1';
}
int main() {
    WSADATA wsa;
    SOCKET sock;
    struct sockaddr_in server;
    char input[50], data[9];
    char key[11] = "1010000010";
    printf("Enter binary data: ");
    scanf("%s", input);
    pad(input, data);
    printf("Padded data: %s\n", data);
    sdes(data, key);
    printf("Encrypted data: %s\n", data);
    WSAStartup(MAKEWORD(2,2), &wsa);
    sock = socket(AF_INET, SOCK_STREAM, 0);
    server.sin_family = AF_INET;
    server.sin_port = htons(PORT);
    server.sin_addr.s_addr = inet_addr("127.0.0.1");
    connect(sock, (struct sockaddr*)&server, sizeof(server));
    send(sock, data, 8, 0);
    closesocket(sock);
    WSACleanup();
    return 0;
```

```
}
```

ATTACKER

```c
#include <stdio.h>
#include <string.h>
void sdes(char *data, char *key) {
    for (int i = 0; i < 8; i++)
        data[i] = (data[i] == key[i % 10]) ? '0' : '1';
}
void int_to_bin(int n, char *bin) {
    for (int i = 9; i >= 0; i--) {
        bin[i] = (n % 2) + '0';
        n /= 2;
    }
    bin[10] = '\0';
}
int main() {
    char encrypted[9];
    printf("Enter captured encrypted data: ");
    scanf("%s", encrypted);
    for (int i = 0; i < 1024; i++) {
        char key[11], temp[9];
        int_to_bin(i, key);
        strcpy(temp, encrypted);
        sdes(temp, key);
        printf("Key: %s -> Plaintext: %s\n", key, temp);
    }
    return 0;
}
```

Server output

```
PS C:\Users\Dell\Desktop\cryptolab_4> ./server
Server waiting...
Encrypted data received: 01101100
Decrypted data: 11001100
PS C:\Users\Dell\Desktop\cryptolab_4>
```

Client output

```
PS C:\Users\Dell\Desktop\cryptolab_4> ./client
Enter binary data: 11001100
Padded data: 11001100
Encrypted data: 01101100
PS C:\Users\Dell\Desktop\cryptolab_4>
```

Attacker output

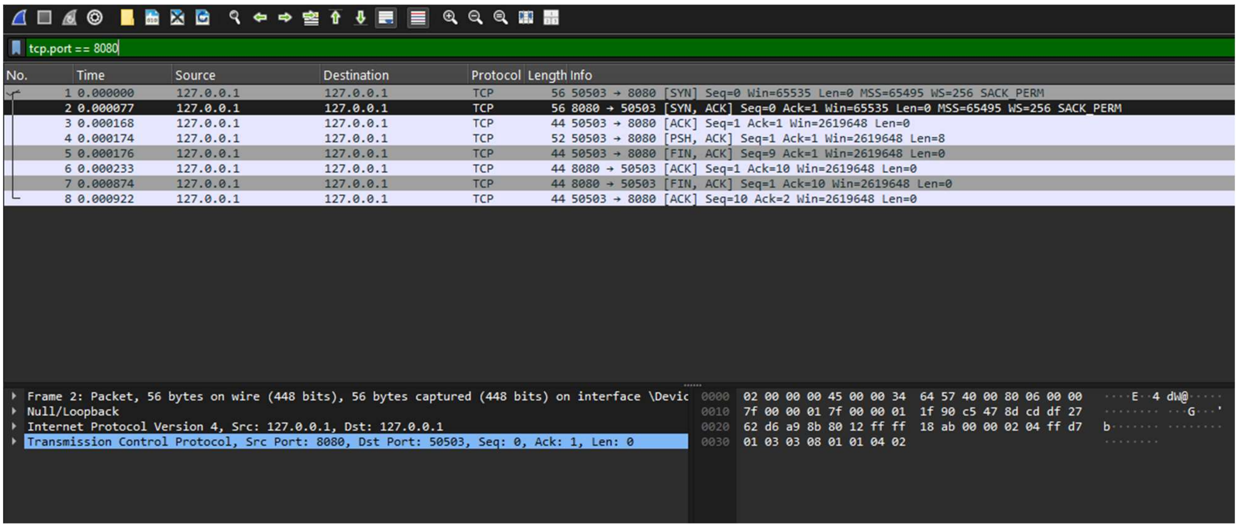
```
Key: 0000001001 -> Plaintext: 01101110
Key: 0000001010 -> Plaintext: 01101110
Key: 0000001011 -> Plaintext: 01101110
Key: 0000001100 -> Plaintext: 01101111
Key: 0000001101 -> Plaintext: 01101111
Key: 0000001110 -> Plaintext: 01101111
Key: 0000001111 -> Plaintext: 01101111
Key: 0000010000 -> Plaintext: 01101000
Key: 0000010001 -> Plaintext: 01101000
Key: 0000010010 -> Plaintext: 01101000
Key: 0000010011 -> Plaintext: 01101000
Key: 0000010100 -> Plaintext: 01101001
Key: 0000010101 -> Plaintext: 01101001
Key: 0000010110 -> Plaintext: 01101001
Key: 0000010111 -> Plaintext: 01101001
Key: 0000011000 -> Plaintext: 01101010
Key: 0000011001 -> Plaintext: 01101010
Key: 0000011010 -> Plaintext: 01101010
Key: 0000011011 -> Plaintext: 01101010
```

Threat Model:

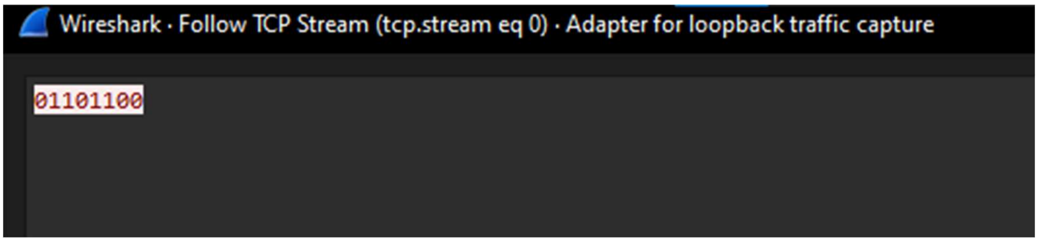
- Passive eavesdropping using Wireshark
- Active brute-force attack on captured ciphertext

Wireshark output

Apply filter tcp.port == 808



Follow tcp stream -> show in ASCII



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 S DES keys systematically
    - Attempts decryption of captured ciphertext

Security Goal:

• Ensure confidentiality of transmitted data

• Demonstrate encryption effectiveness

• Highlight weakness of S-DES

Prevention Mechanism and Security Reinforcement:

• Encryption of data using **Simplified DES (S-DES)** before transmission
• Use of a **shared symmetric secret key** known only to the client and server
• Transmission of **encrypted binary ciphertext** instead of plaintext over the network
• Packet capture using **Wireshark reveals only encrypted binary data** to attackers
• Limiting brute-force attempts to demonstrate attack feasibility in a controlled lab environment
• Demonstration of **S-DES vulnerability** to highlight the need for stronger algorithms
• Recommendation to replace **S-DES with modern secure algorithms such as AES**
• Importance of **secure key management practices** in real-world systems
• Monitoring network traffic using **intrusion detection and packet analysis tools**

Implementation Details

Programming Language Used: C

Core Functions Implemented:
• Encryption Algorithm: **Simplified DES (S-DES)**
• Block Size: **8 bits**
• Key Type: **Symmetric secret key (10-bit)**
• Encryption Technique: **XOR-based simplified substitution**
• Mode of Operation: **Single block (educational demonstration)**
• Communication Protocol: **TCP socket programming (WinSock)**
• Data Transfer Format: **Binary encrypted ciphertext**
• Monitoring Tool: **Wireshark**
• Attack Simulation: **Brute-force key search over 2^{10} key space**

- **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:

• Encrypted communication was successfully established.
• Wireshark captured only encrypted data.
• Server decrypted data correctly.
• Brute-force attack recovered possible plaintexts.

• Demonstrates insecurity of S-DES.

Result Table

Parameter	Result
Plaintext confidentiality	Achieved
SDES encryption	Successful
SDES 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.