

Secure File Transfer using Diffie- Hellman & 3DES

ANKIT SRIVASTAVA (212IS003)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA,
SURATHKAL, INDIA

Abstract. The safe transmission of data between client and server utilising the Diffie-Hellman key exchange and the 3DES algorithm. To begin, the sender and recipient exchange the key for the encrypted file using the Diffie-Hellman key exchange. As soon as the recipient receives the file, they decrypt it so that it may be read by the receiver. In this project, the application integrated the Diffie-Hellman Key Exchange with the 3DES algorithm to create a secure data transmission panel. This programme was the first to employ Diffie-Hellman Key Exchange to exchange shared secret keys.

Keywords: Encryption, Decryption, 3DES, Diffie-Hellman Key Exchange

1. Introduction

With the rising number of assaults and threats in today's world, data protection has become one of the most important requirements. Information is a precious asset that each firm creates, acquires, stores, and transfers. Safeguarding this from internal or external corruption and unauthorised access shields a firm from monetary loss, reputation damage, deterioration of consumer confidence, and image degradation. The relevance of information security stems from the fact that it safeguards our sensitive information, permits the safe functioning of applications deployed on the company's information technology system, and allows the business to function. It is advantageous in the sphere of business. As technology advances, so, too, do crime rates. As a result, information security must be used to secure your data.

Algorithms used in modern cryptography include symmetric key, asymmetric key, hash function, and key exchange algorithms. To some extent, different cryptographic algorithms meet different security criteria.

Nevertheless, cryptographic approaches must be used to offer a strong layer of security that assures the safe and secure transmission of messages to their intended recipients. Symmetric encryption techniques, for example, must employ a key exchange protocol to generate a more secure session key between two parties. Although hybrid cryptographic systems meet stringent security requirements, their performance might suffer as execution times lengthen.

1.1 Problem Statement

In today's world, data is one of the most precious assets for both users and enterprises. Data breaches can be extremely damaging and can bring us to our knees both financially and psychologically. If data is sent in an unencrypted format, it is quite likely that it can be sniffed. This can provide an attacker with not just important information, as well as the information server-side, allowing him to capture a transaction. Furthermore, non-secure communication can be tampered

with by an intruder. In the upcoming section, we have come up with some secure ways to tackle such problems.

1.2 Motivation

Data breaches are a fairly regular issue that arises on a daily basis. Data leaking causes tremendous destruction, depending on the sort of data involved. Such repercussions include database destruction or corruption, the leakage of secret information, industrial espionage, and regulatory obligations to disclose and perhaps repay people impacted. The main motivation is to build a secure pipeline for data transmission so that attackers don't get easy access into his data flow penal. In this project, we have tried to integrate Diffie-Helman Key Exchange with triple Decryption to make a secure channel of data flow.

1.3 Scope

Data security entails implementing particular controls, standard rules, and processes to safeguard data from a wide range of threats such as unauthorised access, accidental loss, and destruction. Poor data privacy may expose a corporation to a variety of risks, including costly penalties and lawsuits, economic loss, and reputation harm. The loss or illegal exposure of sensitive data may be extremely expensive to a company. As technology improves, organisations become increasingly vulnerable to attack, necessitating the need for data security to protect information from attackers and provide a highly isolated channel for data flow. As a result, the data security sector is one of the most broad and trending topics to work on.

1.4 Objectives

The ultimate purpose of network security is to secure data from dangers such as accidental or intentional data loss, destruction, or abuse. These attacks jeopardise the communication over network which causes integrity and access loss. In this paper, we are dealing with two major algorithm to secure the communication panel of data flow. Using Deffie-Helman Key Exchange, we allow two people conversing via a public channel to establish a shared secret without transmitting it over the Internet. Along with 3DES, which runs DES algorithm thrice using

three separate keys, making it more difficult for the attackers to gain the unauthorized access.

1.5 Organization of the Report

The remainder of this work is structured as follows: Section 2 provides a detailed literature review of prior work in this topic. Section 3 identifies and provides the most significant aspect of this work, which includes several algorithms to deal with such attacks with a detailed description of the Deffie-Helman Key Exchange and triple decryption algorithm. Section 4 outlines all of the experimental results, which are then followed by the Conclusion and Reference.

2. Related Work

Data breaches may be incredibly costly, both economically and socially. If we continue to send data in an unencrypted manner, the odds of it being hacked are pretty significant. To solve this issue, several researchers have worked hard to devise solutions that make it more difficult for attackers to break security barriers. Some focused on cloud data security, while others attempted to design methods to deal with cryptographic issues. In this part, certain well-known works are analysed in depth.

At IEEE Transactions on Information Theory, Nov 1976, W. Diffie, Department of Electrical Engineering, University of Stanford, Stanford, CA, USA and M. Hellman from Department of Electrical Engineering, University of Stanford, Stanford, CA, USA proposed first view of Deffie-Helman Key Exchange. It was developed in 1976 by Whitfield Diffie and Martin Hellman as the first viable mechanism for creating a shared secret across an unsecured communications channel. They examined two types of recent advances in cryptography that have resulted in the need for new forms of cryptographic systems that reduce the requirement for safe key distribution methods while providing the equivalent of a written signature. This study provides solutions to the difficulties that are currently unsolved.

At 2013 International Conference on Communication Systems and Network Technologies, Mr. Prashant Rewagad, HOD, Dept of Computer Science \& Engineering and Ms.Yogita Pawar, M.E Student, Dept of Computer Science \& Engineering,G.H.Raisoni Institute of Engg and Management Affiliated to North Maharashtra University Jalgaon, India, proposed different methodology to safeguard the secrecy of data saved in the cloud. They proposed an integration of digital signature and Diffie Hellman key exchange with the (AES) Advanced Encryption Standard encryption method. They thought that even if the key in transit is hacked, the facility given by Diffie Hellman key exchange renders it meaningless, because the key in transit is useless without the user's private key, which is restricted to the genuine user. This suggested three-way technique makes it difficult for hackers to breach the security system, therefore safeguarding cloud resources.

In 2013, at International Journal of Computer Applications, Gurpreet Singh,a M.Tech Research Scholar and Supriya, a Assistant Professor, Department of Computer Science and Engineering Sri Guru Granth Sahib World University, Fatehgarh Sahib, Punjab, India reviewed a study of all encryption algorithms (RSA, DES, 3DES and AES) for Information Security. This paper was a general study and comparison among different encryption algorithm in tabular form. They presented an overview of existing efforts on encryption schemes. All of the strategies are beneficial for real-time encryption, but each is unique in its own way, which may be acceptable for different applications and has its own set of advantages and disadvantages. According to their investigation and review of the literature, they discovered that the AES algorithm is the most efficient in terms of speed, time, throughput, and avalanche impact.

During International Journal of Scientific and Research Publications, Volume 5, Issue 6, June 2015, Mrs.Mamatha and Mr.Pradeep Kanchan, Department of CSE, NMAMIT, NITTE, proposed a hybrid cryptographic algorithm to enhance the security of cloud data. This study combines the concepts of AES and 3DES to create a hybrid model that is used for uploading data to the cloud server by encrypting it and retrieving data from the cloud server by decrypting it.

At 2018 6th International Conference on Cyber and IT Service Management (CITSM), Yusfrizal Yusfrizal, Abdul Meizar, Helmi Kurniawan, Fherly Agustin proposed an article on cryptographic application for data security by using the Diffie-Hellman key exchange and the AES algorithm. They developed security solutions for protecting cloud data from malevolent users, which the Diffie Hellman key exchange algorithm overlooked. They also handle access control issues by employing a good authentication technique based on two factors.

During INTERNATIONAL JOURNAL OF INFORMATION AND COMPUTING SCIENCE, Celeste Murnal,a PG Student and K.Pramilarani, a Senior Assistant professor, Department of Computer Science and Engineering, New Horizon College of Engineering, Bangalore, India proposed a paper titled “Secure Text Transfer”. Their paper presents a secure file storage cloud system based on encryption and Diffie-Hellman. The approach encrypts the file stored in the cloud and uses Diffie-Hellman to authenticate the user in order to decrypt the needed file. According to the technique, the basic implementation of the provided methodology that may be improved and adjusted based on the needs of the user. It added a second layer of protection to files saved in the cloud by employing encryption and the Diffie Hellman Algorithm.

3. Proposed Approach

For a secure file transfer with client and server, we have implemented Diffie-Hellman and 3DES algorithms together. In order to proceed with the process, first, we need to create two different directories client and server to transfer files from the server directory to the client directory.

The server directory contains the below-mentioned files/folders.

- A directory named **files** to store all the files that need to be transferred to client.
- A code implementation file is coded in CPP.
- An output file.
- A header file that contains all the header needed in algorithm implementation.
- A Makefile to create a **server** file for data transmission.

The client directory contains the below-mentioned files/folders.

- A directory named **downloads** to store all the files that are transferred from the server on request.
- A code implementation file is coded in CPP.
- An output file.
- A header file that contains all the header needed in algorithm implementation.
- A Makefile to create a **client** file for data transmission.

We have CPP file in both the directories which contains the implementation of the Diffie-Hellman Key Exchange Algorithm. The DH algorithm is basically an asymmetric key algorithm. We can also say, the Diffie-Hellman (DH) Algorithm, a key-exchange protocol that allows two parties interacting over a public network to develop a unified secret without exposing it to the Internet. DH allows the two to perform symmetric cryptography to encrypt and decrypt their communication or data using a public key.

To implement Diffie-Hellman, the two end-users, client and server, mutually agree on positive whole integers p and q while

communicating via a channel they know is private, such that p is a prime number and q is a generator of p . When raised to positive whole-number powers smaller than p , the generator q never generates the same output for any two such whole numbers. Although the value of p might be significant, the value of q is generally minimal.

They chose positive whole-number personal keys a and b , both less than the prime-number modulus p , as soon as both sides agreed on p and q in secret. Neither user shares their personal key with anybody; they should memorise these numbers rather than writing them down or storing them anywhere. Then, using the algorithms, the client and server compute public keys a^* and b^* based on their personal keys where,

$$a^* = q^a \bmod p$$

$$b^* = q^b \bmod p$$

The two users can exchange their public keys a^* and b^* across an unsecured communication medium, such as the Internet or a corporate-wide area network (WAN). On the basis of their own personal keys, either user can produce a number x from these public keys. The server uses the formula to calculate x .

$$x = (b^*)^a \bmod p$$

$$x = (a^*)^b \bmod p$$

According to any of the above two formulae, the value of x is the same. The personal keys a and b , which are crucial in calculating x , have not been transferred through a public media, however. Because x is such a huge and seemingly random number, even with the aid of a powerful computer and millions of attempts, a potential hacker has essentially no chance of successfully predicting it. In principle, the two users can communicate secretly via a public channel using their preferred encryption mechanism and the decryption key x .

After completion of connection establishment, triple DES is applied on the file to make it more sure for the transmission. DES is a Feistel network-based symmetric-key technique. It's a symmetric key cipher,

which means it utilizes the same key for encryption and decryption. The Feistel network renders each of these processes almost identical, resulting in a more efficient technique to implement.

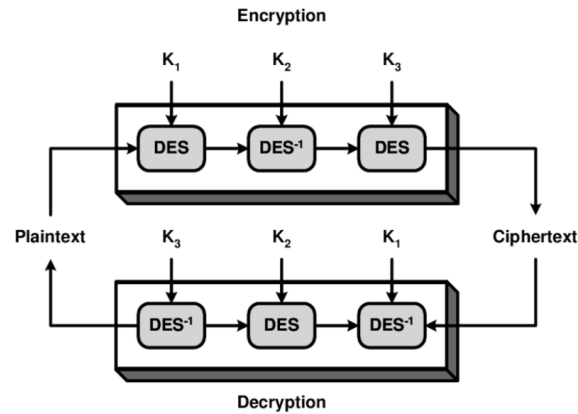


Fig 1: 3-DES Structure

Although DES has a 64-bit block and key size, the key only provides 56-bits of security in reality. Because of DES's short key length, 3DES was created as a more secure alternative. The DES algorithm is executed three times with three different keys in 3DES; nevertheless, it is only deemed secure if three different keys are utilized.

In this project, we have implemented Diffie-Hellman to generate a channel between the client and server with a public key shared with each one and the private key is kept secret. Once the connection is established, the server wraps the file three times with an encryption mechanism which is terms as triple DES algorithm and sends it to the client who is requesting the file. The next section shows the implementation details and analysis of the algorithm.

4. Experimental Results and Analysis

We need to create two different directories **client** and **server** to transfer files from the server directory to the client directory.

Name	Date modified	Type	Size
.git	25-02-2022 13:08	File folder	
CLIENT	23-04-2022 00:10	File folder	
SERVER	23-04-2022 00:10	File folder	
readme.md	25-02-2022 13:08	MD File	1 KB



Client Directory:

downloads	23-04-2022 13:24	File folder	
2018201103_assign_1_client.cpp	25-02-2022 13:08	CPP File	10 KB
2018201103_assign_1_client.o	25-02-2022 13:17	O File	32 KB
client	23-04-2022 00:10	File	38 KB
header.h	25-02-2022 13:08	H File	2 KB
Makefile	25-02-2022 13:08	File	1 KB

Server Directory:

files	22-04-2022 23:20	File folder	
2018201103_assign_1_server.cpp	25-02-2022 13:08	CPP File	13 KB
2018201103_assign_1_server.o	25-02-2022 13:16	O File	56 KB
header.h	25-02-2022 13:08	H File	2 KB
Makefile	25-02-2022 13:08	File	1 KB
server	23-04-2022 00:10	File	55 KB

Content of files in server:

Name	Date modified	Type	Size
 new	23-04-2022 13:20	File	1 KB
 temp	25-02-2022 13:12	File	1 KB

Content of downloads in client:

> Project > Secure-File-Transfer-Using-Diffie-Hellman-and-3DES > CLIENT > downloads			
Date modified	Type	Size	
This folder is empty.			

To start the process of file transmission, we need to open the Linux terminal and type the below command.

1. `./server`

```

root@LAPTOP-G5I4TINV: /mnt/c/Users/Ankit Srivastava/Desktop/2nd Semester/Network Security
root@LAPTOP-G5I4TINV:/mnt/c/Users/Ankit Srivastava/Desktop/2nd Semester/Network Security/Project/Secure-File-Transfer-Using-Diffie-Hellman-and-3DES/SERVER# ./server
Waiting for client to connect

```

2. `./client 127.0.0.1`

```

root@LAPTOP-G5I4TINV: /mnt/c/Users/Ankit Srivastava/Desktop/2nd Semester/Network Security
root@LAPTOP-G5I4TINV:/mnt/c/Users/Ankit Srivastava/Desktop/2nd Semester/Network Security/Project/Secure-File-Transfer-Using-Diffie-Hellman-and-3DES/CLIENT# ./client 127.0.0.1

```

3. Connection Setup:

In the below terminal snapshot, we can see that there is a connection establishment followed by a triple DES algorithm.

```
Transfer-Using-Diffie-Hellman-and-3DES/CLIENT# ./client 127.0.0.1
Prime number      :2140836371
p is 2140836371
Primitive root    :3

Private key of Client :1579272657
Public key of Client :309850136

PPP - Sending 2140836371 3 309850136
Public key of Server :1902669489

Shared key        :360929422

Prime number      :2140836371
p is 2140836371
Primitive root    :3

Private key of Client :1579272657
Public key of Client :309850136

PPP - Sending 2140836371 3 309850136
Public key of Server :216099212

Shared key        :945147406

Prime number      :2140836371
p is 2140836371
Primitive root    :3

Private key of Client :1579272657
Public key of Client :309850136

PPP - Sending 2140836371 3 309850136
Public key of Server :1183702496

Shared key        :219710879

The three keys are:
360929422 945147406 219710879

1. Request a file
0. Exit

Enter your input :
```

```
root@LAPTOP-G5I4TINV: /mnt/c/Users/Ankit Srivastava/Desktop/2nd Semester/IT
Transfer-Using-Diffie-Hellman-and-3DES/SERVER# ./server
Waiting for client to connect
Connection established

new connection with socket id : 4

Prime number          :2140836371
Primitive root         :3

Private key of Server  :1610428765
Public key of Server   :1902669489

Public key of Client   :309850136

Shared key is          :360929422
```

In the below snapshot, the three keys used in triple encryption are shown.


Client requests for a file “new” from the server.

```
root@LAPTOP-G5I4TINV: /mnt/c/Users/Ankit Srivastava/Desktop/IT
The three keys are:
360929422 945147406 219710879

1. Request a file
0. Exit

Enter your input :
    1
Please enter the file name: new
Sending the request message REQ to the server
Request for file sent
DONE
```

As the file has been transferred to the client, the **client** download folder has a **new** file that was not present before.

Name	Date modified	Type	Size
 new	23-04-2022 13:20	File	1 KB

So, the transmission of files from the **server** to **client** using **Diffie-Hellman** and **3DES** algorithm is completed.

5. Code & Algorithm

Client.cpp:

```
#include "header.h"

#define DEFAULT_SERVER "127.0.0.1"

char ctos_buff[MAX];
char stoc_buff[MAX];

char *ip;
int sock,port;

ull des_key[3];

/*
Fast exponential algo to find power
*/
ull power(ull base,ull n,ull prime)
{
    unsigned long long result=1;
    unsigned long long y=base;

    while(n>0)
    {
        if(n & 1)
            result = (result * y) % prime;

        y = (y * y) % prime;

        n = n>>1;
    }
    ull finalresult = result % prime;
    return finalresult;
}

/*
Function to test Miller Test
*/
int my_miller(ull value)
```

```

{
    if(value%2==0|| value<2)
        return 0;
    ull q = value - 1;
    ull k = 0;

    while(q%2==0)
    {
        q>>=1;
        k++;
    }
    for(int i=0;i<15;i++)
    {
        long long int a= rand() % (value - 4) + 2;
        long long int result = power(a,q,value);
        int flag=1;

        for(;q != value - 1 ; q*=2)
        {
            if(result==1 || result == value - 1)
            {
                flag=0;
                break;
            }
            result = result * result % value;
        }
        if(flag==1)
            return 0;
    }
    return 1;
}

/*
This function generate large prime number which pass miller test
*/
ull get_prime()
{
    srand(time(NULL));
    while(1)
    {
        // cout<<"trying "<<prime<<endl;
        ull prime= rand() % LLONG_MAX;
        if(my_miller(prime))

```



```

        return prime;
    }
}

/*
This function returns primitive root of n
*/
ull my_Proot(ull n)
{
    ull MaxSize = 1000000;

    ull prime[MaxSize];
    memset(prime,1,sizeof(prime));
    for(int p=2; p*p<MaxSize;p++)
    {
        if(prime[p])
        {
            for(int i=p*2;i<MaxSize;i+=p)
                prime[i]=0;
        }
    }
    ull phi=n-1;
    ll flag=1;
    ll i = 2;
    while(i++)
    {
        int j;
        for(j=2;j<=sqrt(phi);j++)
        {
            if(prime[j] && phi%prime[j]==0 && power(i,phi/prime[j],n)==1)
            {
                flag=0;
                break;
            }
        }
        if(flag)
            return i;
    }
    return -1;
}

/*
This function Make connection with server & set sock for socket communication

```

```

*/
void connecttoServer()
{
    int s_port=PORT;
    struct sockaddr_in serv_addr;
    sock = socket(AF_INET, SOCK_STREAM,0);
    memset(&serv_addr,'0',sizeof(serv_addr));

    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(s_port);

    inet_pton(AF_INET , "127.0.0.1" , &serv_addr.sin_addr); //converts the character string
src into a network address structure in the af address family, then copies the network address
structure to dst

    if(connect(sock,(struct sockaddr *)&serv_addr, sizeof(serv_addr)))
    {
        printf("Connection not established\n");
        exit(0);
    }
}

/*
Send given data to server
*/
void send_to_server(char *s)
{
    strncpy(ctos_buff,s,strlen(s));

    send(sock,ctos_buff,strlen(ctos_buff),0);
}

/*
Receive data from server
*/
void receive_from_server()
{
    int i=read(sock,stoc_buff, MAX);
    stoc_buff[i]='\0';
}

/*

```

This function Generate public key & establish secreate key (caesar key) between client and server

```

*/
void generatePublicNumberforKey()
{
    for(int it=0;it<3;it++){
        ull prime_num=get_prime();
        printf("Prime number      :%llu\n",prime_num);
        cout<<"p is "<<prime_num<<endl;
        ull primitive_root=my_Proot(prime_num);
        printf("Primitive root    :%llu\n\n",primitive_root);

        ull c_private_key = rand() % (prime_num - 1) + 1;
        printf("Private key of Client  :%llu\n",c_private_key);

        ull c_public_key = power(primitive_root,c_private_key,prime_num);
        printf("Public key of Client   :%llu\n\n",c_public_key);

        if(it==0) connecttoServer();

        sprintf(ctos_buff,"%llu$%llu$%llu",prime_num, primitive_root, c_public_key);

        PubKey pk;
        pk.q = prime_num;
        pk.r = primitive_root;
        pk.Y = c_public_key;
        cout<<" PPP - Sending "<<pk.q<<" "<<pk.r<<" "<<pk.Y<<endl;
        send(sock, &pk, sizeof(PubKey), 0);
        // send_to_server(ctos_buff);

        receive_from_server();

        ull s_public_key = atol(stoc_buff);
        printf("Public key of Server   :%llu\n\n",s_public_key);

        ull shared_key = power(s_public_key, c_private_key , prime_num);
        printf("Shared key    :%llu\n\n",shared_key);

        des_key[it]=shared_key;
    }
}

void send_disconnect(){
    int src_addr, dest_addr;

```

```

Msg send_msg;

dest_addr = inet_addr("DEFAULT_SERVER");
src_addr = inet_addr("127.0.0.1");

send_msg.hdr.opcode = DISCONNECT;
send_msg.hdr.s_addr = src_addr;
send_msg.hdr.d_addr = dest_addr;

Disconnect dis;
send_msg.body.disconnect = dis;

send(sock, &send_msg, sizeof(Msg), 0);
// close(sock);
}

void request_file(){

    cout<<"Please enter the file name: ";
    string file_name;
    cin>>file_name;

    string file_path = "./downloads/"+file_name;

    ofstream myfile;
    myfile.open (file_path,ios::out | ios::binary);

    int src_addr, dest_addr;
    Msg send_msg;

    dest_addr = inet_addr("DEFAULT_SERVER");
    src_addr = inet_addr("127.0.0.1");

    /* send the request message REQ to the server */
    printf("Sending the request message REQ to the server\n");
    send_msg.hdr.opcode = REQSERV;
    send_msg.hdr.s_addr = src_addr;
    send_msg.hdr.d_addr = dest_addr;

    ReqServ reqserv;
    strcpy(reqserv.file_name, file_name.c_str());
    send_msg.body.reqserv = reqserv;
    cout<<"Request for file sent"<<endl;
    send(sock, &send_msg, sizeof(Msg), 0);
}

```

```

int chunk=0;
Msg recv_msg;
int nbytes;
while((nbytes = recv(sock, &recv_msg, sizeof(Msg), 0)) > 0){
    if (nbytes == -1) {
        fprintf(stderr, "*** Server error: unable to receive\n");
    }
    else{
        if(recv_msg.hdr.opcode==DISCONNECT){
            cout<<endl<<"File not found at server"<<endl;
            send_disconnect();
            break;
        }
        else if(recv_msg.hdr.opcode==ENCMMSG){
            unsigned char out[BUFSIZE], back[BUFSIZE];
            // char *key1 = "2000141204";
            // char *key2 = "101636102";
            // char *key3 = "1916220121";
            // cout<<"The three keys are:"<<endl;
            // cout<<des_key[0]<<" "<<des_key[1]<<" "<<des_key[2]<<endl;

            char *key1 = (char *)to_string(des_key[0]).c_str();
            char *key2 = (char *)to_string(des_key[1]).c_str();
            char *key3 = (char *)to_string(des_key[2]).c_str();

            // cout<<"KEYS-----"<<endl;
            // cout<<key1<<" "<<key2<<" "<<key3<<endl;
            int i, num, len, result;
            int n = 0;

            DES_cblock k1,k2,k3;
            DES_cblock ivsetup = {0xE1, 0xE2, 0xE3, 0xD4, 0xD5, 0xC6, 0xC7, 0xA8};
            DES_key_schedule ks1,ks2,ks3;
            DES_cblock ivec;

            memset(out, 0, sizeof(out));
            memset(back, 0, sizeof(back));

            DES_string_to_key(key1, &k1);
            DES_string_to_key(key2, &k2);
            DES_string_to_key(key3, &k3);

            if ((result = DES_set_key_checked((DES_cblock *)k1, &ks1)) != 0) {

```

```

        if (result == -1) {
            printf("ERROR: key parity is incorrect\n");
        } else {
            printf("ERROR: weak or semi-weak key\n");
        }
        exit(1);
    }

    if ((result = DES_set_key_checked((DES_cblock *)k2, &ks2)) != 0) {
        if (result == -1) {
            printf("ERROR: key parity is incorrect\n");
        } else {
            printf("ERROR: weak or semi-weak key\n");
        }
        exit(1);
    }

    if ((result = DES_set_key_checked((DES_cblock *)k3, &ks3)) != 0) {
        if (result == -1) {
            printf("ERROR: key parity is incorrect\n");
        } else {
            printf("ERROR: weak or semi-weak key\n");
        }
        exit(1);
    }

    EncMsg em = recv_msg.body.encmsg;
    long long chunksize = em.size;

    // strcpy((char *)out,(const char *)em.cipher_block);
    for(int i=0;i<chunksize;i++){
        out[i]=em.cipher_block[i];
    }
    // cout<<em.cipher_block<<endl;

    memcpy(ivec, ivsetup, sizeof(ivsetup));
    num = 0;

    for (i = 0; i < chunksize; i++) {
        DES_ede3_ofb64_encrypt(&(out[i]), &(back[i]), 1, &ks1, &ks2, &ks3, &ivec,
&num);
    }
    // myfile.write((const char *)out,chunksize);
    // cout<<"writing chunk no "<<(++chuk)<<" of size "<<chunksize<<endl;

```

```

        myfile.write((const char *)back, chunksize);
        memset(back, 0, sizeof(back));
        memset(out, 0, sizeof(out));

    }
    else if(recv_msg.hdr.opcode==REQCOMP){
        cout<<"DONE"<<endl;
        break;
    }
}
}

myfile.close();
}

int main(int argc, char* argv[])
{
    //cout<<"char encoding size : "<<sizeof(encoding_scheme)/sizeof(char);

    if(argc!=2)
    {
        printf("Please provide IP address \n");
        return 0;
    }
    srand(time(NULL));

    generatePublicNumberforKey();

    cout<<"The three keys are:"<<endl;
    cout<<des_key[0]<<" "<<des_key[1]<<" "<<des_key[2]<<endl;

    int flag=1;

    // Menu driven Program for user
    do
    {
        int ch;
        cout<<"\n\n";
        cout<<"1. Request a file "<<endl;
        cout<<"0. Exit"<<endl;
        cout<<"\nEnter your input : "<<endl;
        cin>>ch;
        switch(ch)
        {

```

```

        case 1 :
            // loginrequest(argv[1]);
            request_file();
            break;
        case 0:
            flag=0;
            break;
        default:
            cout<<"Please give correct input :("<<endl;
            break;
    }
}while(flag);
cout<<"\n\nThank You, Have a happy CryptoDay !!!"<<endl;

return 0;
}

```

Server.cpp:

```

#include "header.h"

// int caesar_key;
FILE *plaintext;
int server_fd;
struct sockaddr_in address;
int addrlen = sizeof(address);
char encoding_scheme[]={'
', 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z', ',', '!', '?', '0',
'1', '2', '3', '4', '5', '6', '7', '8', '9', 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z'
, '!'};

/*
    returns size of the file
*/
long long getFileSize(string file)
{

```



```

struct stat sb;
stat(file.c_str(), &sb);
return sb.st_size;
}

/*
This Function for fast exponential
*/
ull power(ull base,ull n,ull prime)
{
    unsigned long long result=1;
    unsigned long long y=base;

    while(n>0)
    {
        if(n & 1)
            result = (result * y) % prime;

        y = (y * y) % prime;

        n = n>>1;
    }
    ull finalresult = result % prime;
    return finalresult;
}

/*
This Function create new socket for server
*/
void createServerSocket()
{
    int opt=1;
    // int s_port=PORT;
    server_fd = socket(AF_INET , SOCK_STREAM , 0); //AF_INET for ipv4,sock_stream for
    TCP,0 for internet protocol

    if(setsockopt(server_fd , SOL_SOCKET , SO_REUSEADDR | SO_REUSEPORT, &opt,
    sizeof(opt)))
        //helps in reuse of address and port. optional
    {
        printf("setsockopt\n");
        exit(0);
    }
}

```

```

    }

    address.sin_family = AF_INET;
    address.sin_addr.s_addr = INADDR_ANY; //INADDR_ANY for localhost
    address.sin_port = htons( PORT ); //The htons() function converts the unsigned short
integer hostshort from host byte order to network byte order.

    if (bind(server_fd, (struct sockaddr *)&address,
              sizeof(address))<0)
    {
        printf("bind failed");
        exit(0);
    }
    listen(server_fd, 5);
}

/*
This Function used for initial communication while establishing secret key
Send the data to client
*/
void sendInitialtoClient(char *stoc_buff,int new_socket)
{
    send(new_socket,stoc_buff,strlen(stoc_buff),0);
}

/*
This Function used for initial communication while establishing secret key
Receive data from client
*/
void receiveInitialFromClient(char *ctos_buff,int new_socket)
{
    int i=0;
    i=read( new_socket , ctos_buff, MAX);
    ctos_buff[i]='\0';
}

/*
Write Log into serverlog.txt file
Generate Public key and establish secret key (caesar key) with given socket_client
*/
void extractDataAndGenerateKeys(int new_socket, vector<ull> &des_key)

```

```

{
    for(int it=0;it<3;it++){
        char ctos_buff[MAX];
        char stoc_buff[MAX];
        // receiveInitialFromClient(ctos_buff,new_socket);
        PubKey recv_key;
        int nbytes = recv(new_socket, &recv_key, sizeof(PubKey), 0);

        ull prime_num = recv_key.q;
        ull primitive_root = recv_key.r;
        ull c_public_key = recv_key.Y;

        int i=0,j=0;
        char temp[MAX];

        printf("Prime number      :%llu\n",prime_num);
        printf("Primitive root    :%llu\n\n",primitive_root);

        ull s_private_key = rand() % (prime_num - 1) + 1;
        printf("Private key of Server :%llu\n",s_private_key);

        ull s_public_key = power(primitive_root,s_private_key,prime_num);
        printf("Public key of Server  :%llu\n\n",s_public_key);

        printf("Public key of Client  :%llu\n\n",c_public_key);

        sprintf(stoc_buff,"%llu",s_public_key);
        strncpy(stoc_buff,stoc_buff,strlen(stoc_buff));
        sendInitialtoClient(stoc_buff,new_socket);

        ull shared_key = power(c_public_key, s_private_key , prime_num);
        printf("Shared key is      :%llu\n\n",shared_key);

        cout<<"des key of "<<it<<" = "<<shared_key<<endl;
        des_key[it]=shared_key;
    }
}

/*
This function find SHA1 hash of given string
*/
// string findHash(string str)
// {

```

```

// int len = str.length()+1;
// char * schunk = new char [len];
// strcpy (schunk, str.c_str());

// unsigned char hash[SHA_DIGEST_LENGTH];
// char buf[SHA_DIGEST_LENGTH * 2];
// SHA1((unsigned char *)schunk, len , hash);

// for (int i = 0; i < SHA_DIGEST_LENGTH; i++)
//     sprintf((char *)&(buf[i * 2]), "%02x", hash[i]);

// string ans;
// for (int i = 0; i < SHA_DIGEST_LENGTH * 2; i++)
// {
//     ans += buf[i];
// }
// return ans;

// }

/*
send Msg to client
*/
int sendMsgtoClient(Msg reply, int new_socket)
{
    int status = send(new_socket, &reply, sizeof(Msg), 0);
    return status;
}

bool file_present(string s){
    DIR *dir;
    struct dirent *ent;

    if ((dir = opendir (".files")) != NULL) {
        /* print all the files and directories within directory */
        while ((ent = readdir (dir)) != NULL) {
            string current_file = ent->d_name;

            struct stat st;
            string file_path = ".files/"+current_file;
            if( stat(file_path.c_str(),&st) == 0 ){
                if( st.st_mode & S_IFREG ){
                    //it's a file
                    if(current_file==s){

```

```

        closedir (dir);
        return true;
    }
}

}

closedir (dir);
} else {
    /* could not open directory */
    perror ("");
    return false;
}
return false;
}

void send_disconnect(int client_socket){
    int src_addr, dest_addr;
    Msg send_msg;

    dest_addr = inet_addr("DEFAULT_SERVER");
    src_addr = inet_addr("127.0.0.1");

    send_msg.hdr.opcode = DISCONNECT;
    send_msg.hdr.s_addr = src_addr;
    send_msg.hdr.d_addr = dest_addr;

    Disconnect dis;
    send_msg.body.disconnect = dis;

    send(client_socket, &send_msg, sizeof(Msg), 0);
    // close(client_socket);
}

void send_ENC_MSG(unsigned char out[BUFSIZE], int client_socket, int chunksize){
    // cout<<"in send ENC MSG with cs = "<<chunksize<<endl;
    int src_addr, dest_addr;
    Msg send_msg;

    dest_addr = inet_addr("DEFAULT_SERVER");
    src_addr = inet_addr("127.0.0.1");

    send_msg.hdr.opcode = ENCMSG;
    send_msg.hdr.s_addr = src_addr;

```

```

send_msg.hdr.d_addr = dest_addr;

EncMsg em;
em.size = chunksize;
for(int i=0;i<BUFSIZE;i++)
{
    em.cipher_block[i]=out[i];
}
// strcpy((char *)em.cipher_block, (const char *)out);
send_msg.body.encmsg = em;

send(client_socket, &send_msg, sizeof(Msg), 0);
}

void send_request_comp(int client_socket){
    int src_addr, dest_addr;
    Msg send_msg;

    dest_addr = inet_addr("DEFAULT_SERVER");
    src_addr = inet_addr("127.0.0.1");

    send_msg.hdr.opcode = REQCOMP;
    send_msg.hdr.s_addr = src_addr;
    send_msg.hdr.d_addr = dest_addr;

    ReqCom rq;
    send_msg.body.reqcom = rq;

    send(client_socket, &send_msg, sizeof(Msg), 0);
}

void send_file(string file_name, int client_socket, vector<ull> &des_key){

    file_name = "./files/"+file_name;

    unsigned char in[BUFSIZE], out[BUFSIZE];

    string s1,s2,s3;
    s1 = to_string(des_key[0]);
    s2 = to_string(des_key[1]);
    s3 = to_string(des_key[2]);

    cout<<"sending file "<<file_name<<" with keys "<<endl;
    // cout<<"STRING KEYS-----"<<endl;

```

```
cout<<s1<<" "<<s2<<" "<<s3<<endl;

char* ckey1 = &*(s1).begin();
char* ckey2 = &*(s2).begin();
char* ckey3 = &*(s3).begin();

ifstream fin;
int i, num, result;

DES_cblock k1,k2,k3;
DES_cblock ivsetup = {0xE1, 0xE2, 0xE3, 0xD4, 0xD5, 0xC6, 0xC7, 0xA8};
DES_key_schedule ks1,ks2,ks3;
DES_cblock ivec;

memset(in, 0, sizeof(in));
memset(out, 0, sizeof(out));
// memset(back, 0, sizeof(back));

DES_string_to_key(ckey1, &k1);
DES_string_to_key(ckey2, &k2);
DES_string_to_key(ckey3, &k3);

if ((result = DES_set_key_checked((DES_cblock *)k1, &ks1)) != 0) {
    if (result == -1) {
        printf("ERROR: key parity is incorrect\n");
    } else {
        printf("ERROR: weak or semi-weak key\n");
    }
    exit(1);
}

if ((result = DES_set_key_checked((DES_cblock *)k2, &ks2)) != 0) {
    if (result == -1) {
        printf("ERROR: key parity is incorrect\n");
    } else {
        printf("ERROR: weak or semi-weak key\n");
    }
    exit(1);
}

if ((result = DES_set_key_checked((DES_cblock *)k3, &ks3)) != 0) {
    if (result == -1) {
        printf("ERROR: key parity is incorrect\n");
    } else {
```

```

    printf("ERROR: weak or semi-weak key\n");
}
exit(1);
}

fin.open(file_name, ios::binary | ios::in);
if (!fin) {
    printf(" ERROR: opening input file\n");
    exit(1);
}
int chunk=0;
int chunksize=BUFSIZE;
long long sizeLeftToSend = getFileSize(file_name);
if (sizeLeftToSend < chunksize)
    chunksize = sizeLeftToSend;
while (fin.read((char *)in, chunksize))
{
    sizeLeftToSend -= chunksize;

    memcpy(ivec, ivsetup, sizeof(ivsetup));
    num = 0;
    for (i = 0; i < chunksize; i++) {
        DES_ede3_ofb64_encrypt(&(in[i]), &(out[i]), 1, &ks1, &ks2, &ks3, &ivec, &num);
    }

    if (chunksize == 0)
        break;

    // cout<<"sending chunk "<<(++chuk)<<" of size "<<chunksize<<endl;
    send_ENC_MSG(out, client_socket, chunksize);

    // send_ENC_MSG(in, client_socket, chunksize);

    usleep(1000);

    memset(in, 0, sizeof(in));
    memset(out, 0, sizeof(out));
    if (sizeLeftToSend < chunksize)
        chunksize = sizeLeftToSend;
}

fin.close();
send_request_comp(client_socket);

```



```

}

void send_not_found(int client_socket){
    send_disconnect(client_socket);
}

void *serverservice(void *socket_desc)
{
    vector<ull> des_key(3);
    int new_socket = *(int *)socket_desc;
    cout<<"new connection with socket id : "<<new_socket<<endl<<endl;
    extractDataAndGenerateKeys(new_socket,des_key);

    // cout<<"The three keys are:"<<endl;
    // cout<<des_key[0]<<" "<<des_key[1]<<" "<<des_key[2]<<endl;
    int flag=1;
    do
    {
        Msg recv_msg;
        cout<<"Waiting..."<<endl;
        int nbytes = recv(new_socket, &recv_msg, sizeof(Msg), 0);
        if (nbytes == -1) {
            fprintf(stderr, "*** Server error: unable to receive\n");
            return NULL;
        }
        int opcode = recv_msg.hdr.opcode;

        //according to Message opcode call appropriate function call to handle it
        if(opcode==PUBKEY)
        {
        }
        else if(opcode == REQSERV)
        {

            // cout<<"The three keys are:"<<endl;
            // cout<<des_key[0]<<" "<<des_key[1]<<" "<<des_key[2]<<endl;
            string s = recv_msg.body.reqserv.file_name;
            if(file_present(s)){
                send_file(s,new_socket,des_key);
            }
            else{
                send_not_found(new_socket);
            }
        }
    }
    while(flag);
}

```

```

    }
}
else if(opcode == ENCMSG)
{
}
else if(opcode == REQCOMP)
{
}
else if(opcode == DISCONNECT)
{
    send_disconnect(new_socket);
}
else
{
    cout<<"Invalid code receive in server side"<<endl;
    flag=0;
}
recv_msg.hdr.opcode=0;

}while(flag);

return socket_desc;
}

int main()
{
    srand(time(NULL));
    // ReadPasswordFile();
    printf("Waiting for client to connect\n");

    createServerSocket();
    pthread_t thread_id;
    int new_socket;
    while (1)
    {
        if ((new_socket = accept(server_fd, (struct sockaddr *)&address, (socklen_t *)&addrlen)) < 0)
        {
            perror("Error in accept connection");
            exit(0);
        }

        printf("Connection established\n\n");
    }
}

```

```
if (pthread_create(&thread_id, NULL, serverservice, (void *)&new_socket) < 0)
{
    perror("\ncould not create thread\n");
}

}

return 0;
}
```

6. Conclusion

This project examines the popular Encryption Algorithms 3DES and Diffie-Hellman Key Exchange in detail. The usage of the internet and networks is growing dramatically. As a result, there is an increased need for securing data exchanged via various networks and services. Different encryption methods are employed to ensure the network and data's security. Secure data transfer is created in this proposal utilizing the two most impactful methods, both of which are based on encryption techniques. To summarise, all of the strategies may be used to encrypt data in real-time. Each approach is distinct in its own way, suited for a variety of applications, and with its own set of benefits and drawbacks.

References

1. W. Diffie, M. Hellman "New directions in cryptography" in Proceedings of the IEEE Transactions on Information Theory (Volume: 22, Issue: 6, Nov 1976)
2. Prashant Rewagad, Yogita Pawar "Use of Digital Signature with Diffie Hellman Key Exchange and AES Encryption Algorithm to Enhance Data Security in Cloud Computing". Communication Systems and Network Technologies (CSNT), 2013. DOI:10.1109/CSNT.2013.97
3. Gurpreet Singh, Supriya. "A Study of Encryption Algorithms (RSA, DES, 3DES and AES) for Information Security," in Proceedings of the International Journal of Computer Applications (0975 – 8887) Volume 67– No.19, April 2013.
4. Mrs.Mamatha, Mr.Pradeep Kanchan, "Use of Digital Signature with Diffie Hellman Key Exchange and Hybrid Cryptographic algorithm to Enhance Data Security in Cloud Computing" in Proceedings of the International Journal of Scientific and Research Publications, Volume 5, Issue 6, June 2015, ISSN 2250-3153.
5. Yusfrizal Yusfrizal, Abdul Meizar, Helmi Kurniawan, Fherly Agustin "Key Management Using Combination of Diffie–Hellman Key Exchange with AES Encryption" in 2018 6th International Conference on Cyber and IT Service Management (CITSM), DOI: 10.1109/CITSM.2018.8674278.
6. Celeste Murnal, K.Pramilarani "Secure Text Transfer" in 2019 INTERNATIONAL JOURNAL OF INFORMATION AND COMPUTING SCIENCE. ISSN NO: 0972-1347.