Assignment – 4

1. Given a text file with 10 sentences as input, write a client server C program where 40 marks

   - client encrypts the text file using bitwise operations (as a key) at the client side and sends the encrypted file to the server. **10 marks**
   - A copy of the key will be located in both, client and server.
   - Server program will take the key as the input and will decrypt the original file. **10 marks**
   - Client will display the ASCII format of original texts as well as the encrypted texts. **5 marks**
   - Server will also display ASCII format the encrypted texts and the original texts after decryption. **5 marks**
   - Client can only send the encrypted texts to the server. **10 marks**

Solution:

Server.c program

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/time.h>
#include <arpa/inet.h>
#include <netinet/in.h>
#include <sys/socket.h>

#define MAX_LINE 50
#define LINSTENPORT 7788
#define SERVERPORT 8877
#define BUFFSIZE 50
#define KEY 10

void writefile(int sockfd, FILE *fp);
void printFile(FILE *fp);
void decryptData(FILE *fp1, FILE* fp2);
int bitwisesub(int x, int y);
int bitwiseadd(int x, int y);

ssize_t total=0;

int main(int argc, char *argv[])
{
    struct timeval start_time;
    struct timeval end_time;

    int sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd == -1)
    {
        perror("Can't allocate sockfd");
        exit(1);
    }

    struct sockaddr_in clientaddr, serveraddr;
    memset(&serveraddr, 0, sizeof(serveraddr));
    serveraddr.sin_family = AF_INET;
    serveraddr.sin_addr.s_addr = inet_addr("10.0.0.1");
    serveraddr.sin_port = htons(SERVERPORT);

    if (bind(sockfd, (const struct sockaddr *) &serveraddr, sizeof(serveraddr)) == -1)
    {
        perror("Bind Error");
        exit(1);
    }

    if (listen(sockfd, LINSTENPORT) == -1)
    {
        perror("Listen Error");
        exit(1);
    }

    printf("\nServer Listening on port %d \n", SERVERPORT);
```

```
while(1){
   socklen_t addrlen = sizeof(clientaddr);
   int connfd = accept(sockfd, (struct sockaddr *) &clientaddr, &addrlen);
   if (connfd == -1)
   {
      perror("Connect Error");
      exit(1);
   }

   FILE *fp = fopen("cipher.txt", "wb");
   if (fp == NULL)
   {
      perror("Can't open file");
      exit(1);
   }

   char addr[INET_ADDRSTRLEN];
   printf("\n------------------------------------------------------------\n");
   printf("Uploading file: %s by %s\n", "PlainText", inet_ntop(AF_INET, &clientaddr.sin_addr, addr, INET_ADDRSTRLEN));
   writefile(connfd, fp);
   fclose(fp);

   printf("Upload Success, Total Bytes = %ld\n", total);

   // file pointers
   FILE *fp1 = fopen("plaintext.txt","r+");
   FILE *fp2 = fopen("cipher.txt","r+");

   // decrypt text
   decryptData(fp2, fp1);
   sleep(3);

   // Move the file pointer to the start.
   fseek(fp1, 0, SEEK_SET);
```

```
        fseek(fp2, 0, SEEK_SET);

        printf("\n\nCipher Text: \n");
        printFile(fp2);
        printf("\n\n");

        printf("\n\nPlain Text: \n");
        printFile(fp1);
        printf("\n\n");

        close(connfd);
    }

    return 0;
}

// write file fn def
void writefile(int sockfd, FILE *fp)
{
    ssize_t n;
    char buff[MAX_LINE] = {0};
    while ((n = recv(sockfd, buff, MAX_LINE, 0)) > 0)
    {
                    total+=n;
        if (n == -1)
        {
            perror("Receive File Error");
            exit(1);
        }

        if (fwrite(buff, sizeof(char), n, fp) != n)
        {
            perror("Write File Error");
            exit(1);
        }
        memset(buff, 0, MAX_LINE);
    }
}


int bitwiseadd(int x, int y)
{
    while (y != 0)
    {
        int carry = x & y;
        x = x ^ y;
        y = carry << 1;
    }
    return x;
}

int bitwisesub(int x, int y)
{
    while (y != 0)
    {
        int carry = (~x) & y;
        x = x ^ y;
        y = carry << 1;
    }
    return x;
}

void decryptData(FILE *fp1, FILE* fp2){
    char ch;
    while ((ch = fgetc(fp1)) != EOF)
    {
        fputc(bitwisesub(ch, KEY), fp2);
    }
}
```

```
void printFile(FILE *fp){
  char ch;
   while ((ch = fgetc(fp)) != EOF)
   {
      printf("%c", ch);
   }
}
```

Client.c program

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <libgen.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <time.h>
#include <sys/time.h>
#include <netinet/in.h>
#include <sys/socket.h>

#define MAX_LINE 50
#define LINSTENPORT 7788
#define SERVERPORT 8877
#define BUFFSIZE 50
#define KEY 10

void sendfile(FILE *fp, int sockfd);
int bitwiseadd(int x, int y);
int bitwisesub(int x, int y);
void encryptData(FILE *fp1, FILE* fp2);
void printFile(FILE *fp);

ssize_t total=0;
```

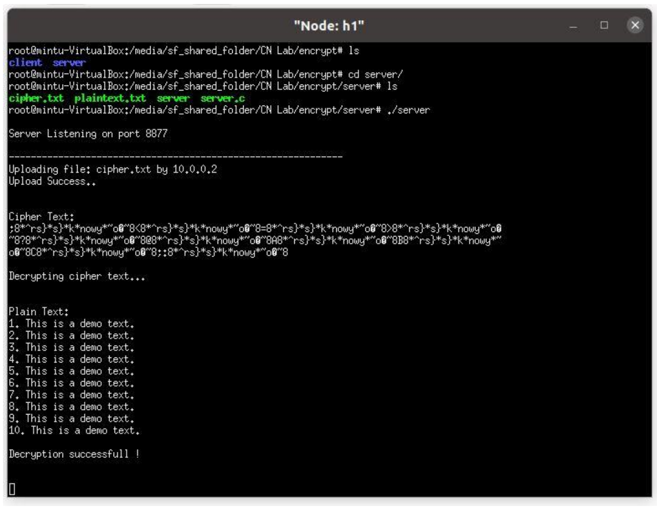
```
int main(int argc, char* argv[])
{
    char buff[BUFFSIZE] = {0};
    struct timeval start_time;
    struct timeval end_time;

    // if (argc != 3)
    // {
    //     perror("usage:./client upload <filepath>");
    //     exit(1);
    // }

    int sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0)
    {
        perror("Can't allocate sockfd");
        exit(1);
    }

    struct sockaddr_in serveraddr;
    memset(&serveraddr, 0, sizeof(serveraddr));
    serveraddr.sin_family = AF_INET;
    serveraddr.sin_port = htons(SERVERPORT);
    serveraddr.sin_addr.s_addr = inet_addr("10.0.0.1");


    if (connect(sockfd, (const struct sockaddr *) &serveraddr, sizeof(serveraddr)) < 0)
    {
        perror("Connect Error");
        exit(1);
    }

    // file pointers
    FILE *fp1 = fopen("plaintext.txt","r");
    FILE *fp2 = fopen("cipher.txt","r+");
```

```
printf("\nPlain Text: \n");
printFile(fp1);

// Move the file pointer to the start.
fseek(fp1, 0, SEEK_SET);

// encrypting data
encryptData(fp1, fp2);

// Move the file pointer to the start.
fseek(fp2, 0, SEEK_SET);

printf("\n\nCipher Text: \n");
printFile(fp2);
printf("\n\n");



FILE *fp = fopen("cipher.txt", "rb");
if (fp == NULL)
{
   perror("Can't open file");
   exit(1);
}

gettimeofday(&start_time, NULL);
sendfile(fp, sockfd);
gettimeofday(&end_time, NULL);

printf("Encrypted File Upload Success..");

fclose(fp);
fclose(fp1);
fclose(fp2);
close(sockfd);
```

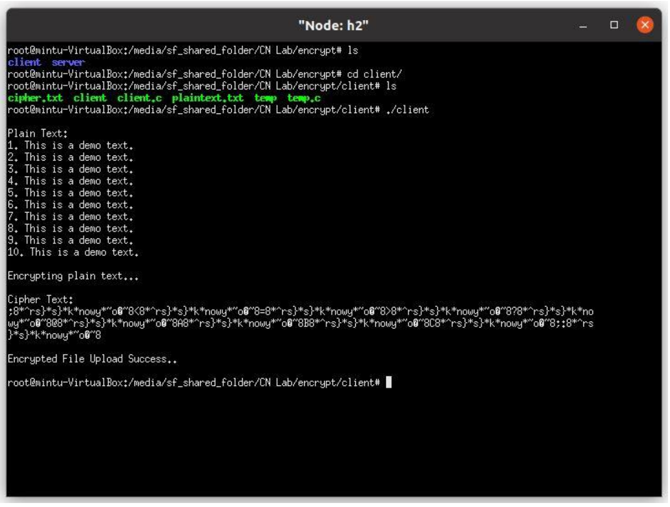
```
    return 0;
}

void sendfile(FILE *fp, int sockfd)
{
    int n;
    char sendline[MAX_LINE] = {0};
    while ((n = fread(sendline, sizeof(char), MAX_LINE, fp)) > 0)
    {
                  total+=n;
        if (n != MAX_LINE && ferror(fp))
        {
            perror("Read File Error");
            exit(1);
        }

        if (send(sockfd, sendline, n, 0) == -1)
        {
            perror("Can't send file");
            exit(1);
        }
        memset(sendline, 0, MAX_LINE);
    }
}

int bitwiseadd(int x, int y)
{
    while (y != 0)
    {
        int carry = x & y;
        x = x ^ y;
        y = carry << 1;
    }
    return x;
}
```

```
int bitwisesub(int x, int y)
{
    while (y != 0)
    {
        int carry = (~x) & y;
        x = x ^ y;
        y = carry << 1;
    }
    return x;
}

void encryptData(FILE *fp1, FILE* fp2){
    char ch;
    while ((ch = fgetc(fp1)) != EOF)
    {
        fputc(bitwiseadd(ch, KEY), fp2);
    }
}

void printFile(FILE *fp){
   char ch;
    while ((ch = fgetc(fp)) != EOF)
    {
        printf("%c", ch);
    }
}
```

<u>Output:</u>


"Node: h1"

```
root@mintu-VirtualBox:/media/sf_shared_folder/CN Lab/encrypt# ls
client  server
root@mintu-VirtualBox:/media/sf_shared_folder/CN Lab/encrypt# cd server/
root@mintu-VirtualBox:/media/sf_shared_folder/CN Lab/encrypt/server# ls
cipher.txt  plaintext.txt  server  server.c
root@mintu-VirtualBox:/media/sf_shared_folder/CN Lab/encrypt/server# ./server

Server Listening on port 8877

------------------------------------------------------------
Uploading file: cipher.txt by 10.0.0.2
Upload Success..


Cipher Text:
;8*^rs}*s}*k*nowy*~o@~8<8*^rs}*s}*k*nowy*~o@~8=8*^rs}*s}*k*nowy*~o@~8>8*^rs}*s}*k*nowy*~o@
~8?8*^rs}*s}*k*nowy*~o@~8@8*^rs}*s}*k*nowy*~o@~8A8*^rs}*s}*k*nowy*~o@~8B8*^rs}*s}*k*nowy*~
o@~8C8*^rs}*s}*k*nowy*~o@~8;:8*^rs}*s}*k*nowy*~o@~8

Decrypting cipher text...


Plain Text:
1. This is a demo text.
2. This is a demo text.
3. This is a demo text.
4. This is a demo text.
5. This is a demo text.
6. This is a demo text.
7. This is a demo text.
8. This is a demo text.
9. This is a demo text.
10. This is a demo text.

Decryption successfull !
```


"Node: h2"

```
root@mintu-VirtualBox:/media/sf_shared_folder/CN Lab/encrypt# ls
client  server
root@mintu-VirtualBox:/media/sf_shared_folder/CN Lab/encrypt# cd client/
root@mintu-VirtualBox:/media/sf_shared_folder/CN Lab/encrypt/client# ls
cipher.txt  client  client.c  plaintext.txt  temp  temp.c
root@mintu-VirtualBox:/media/sf_shared_folder/CN Lab/encrypt/client# ./client

Plain Text:
1. This is a demo text.
2. This is a demo text.
3. This is a demo text.
4. This is a demo text.
5. This is a demo text.
6. This is a demo text.
7. This is a demo text.
8. This is a demo text.
9. This is a demo text.
10. This is a demo text.

Encrypting plain text...

Cipher Text:
;8*^rs}*s}*k*nowy*~o@~8<8*^rs}*s}*k*nowy*~o@~8=8*^rs}*s}*k*nowy*~o@~8>8*^rs}*s}*k*nowy*~o@~8?8*^rs}*s}*k*no
wy*~o@~8@8*^rs}*s}*k*nowy*~o@~8A8*^rs}*s}*k*nowy*~o@~8B8*^rs}*s}*k*nowy*~o@~8C8*^rs}*s}*k*nowy*~o@~8;:8*^rs
}*s}*k*nowy*~o@~8

Encrypted File Upload Success..

root@mintu-VirtualBox:/media/sf_shared_folder/CN Lab/encrypt/client#
```