**CS6008 Cryptography and Network Security.**

**Assignment No.8**

**Name:** G Vignesh **Reg No:**2018103078

**Date:**7.6.2022

**MODULE VII: Block Cipher**

**External Learning-** Implementing block ciphers using openssl in C/C++.

**Aim:**

To implement any of the block ciphers such as blowfish, AES, DES currently in existence using openssl library in c or c++ program. That is to encrypt the given data using the key to a cipher text and to decrypt the cipher to plain text using the key via functionalities provided by **OPENSSL (Open secure Socket Layer)** Library/package via our c/c++ program.

**Tools used:**

* + Ubuntu terminal
  + Vscode (Code editor)
  + Openssl
  + G++( C ++ program compiler)

**Description:**

OpenSSL is a cryptography toolkit developed to provide cryptography algorithm implementations as a simpler package via which program can use the Cryptography based functionalities by just importing and run with proper implementation instead implementing those each and every time which is a tideous, time consuming, error creating process.

OpenSSL provides numbers cryptography algorithms such as blowfish, AES, DES etc with various operation modes such as ECB, CBC, FCB etc with varying key sizes. It also provides Hashing, MAC algos such as SHA,MD5 etc.

We are going to implement Block cipher Advanced Encryption Standard with 128 bit key length in ECB(Electronic Code Book). Electronic Code Book (ECB) is a simple mode of operation with a [block cipher](https://www.techtarget.com/searchsecurity/definition/block-cipher) that's mostly used with symmetric [key](https://www.techtarget.com/searchsecurity/definition/key) [encryption](https://www.techtarget.com/searchsecurity/definition/encryption). It is a straightforward way of processing a series of sequentially listed message blocks.

The input [plaintext](https://www.techtarget.com/searchsecurity/definition/plaintext) is broken into numerous blocks. The blocks are individually and independently encrypted ([ciphertext](https://www.techtarget.com/whatis/definition/ciphertext)) using the encryption key. As a result, each encrypted block can also be decrypted individually.

**Input:** Plain text data along with Key passed to AES encryption.

**Output:** Cipher text data obtained after encryption by AES in ECB mode.

**Execution:**

**Step-1: Installing OPENSSL**

In ubuntu linux based system OPENSSL can be installed by using the command

wget http://www.openssl.org/source/openssl-1.0.1g.tar.gz

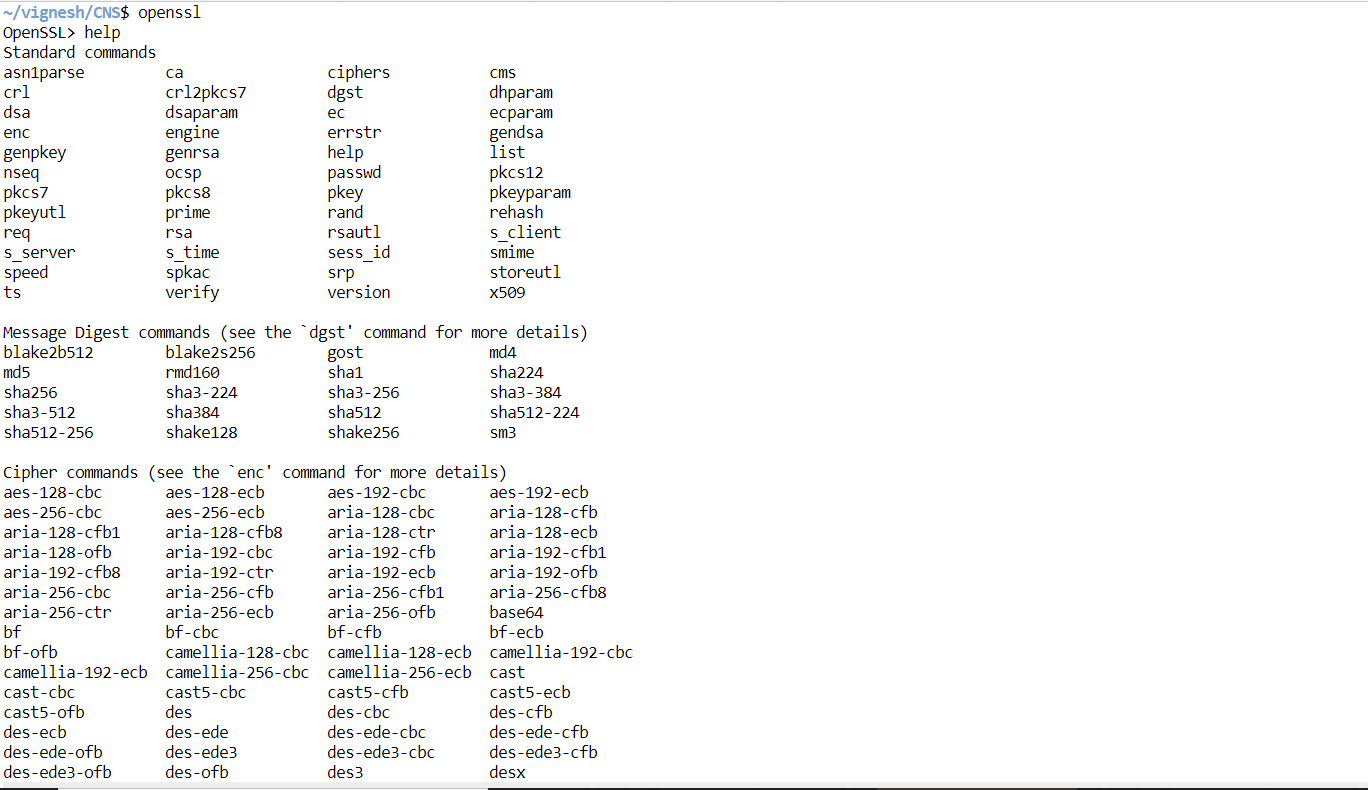
and then extracting and running the make file to install it in our system.

1.tar -xvzf openssl-1.0.1g.tar.gz

2.make

3.sudo make install

Once installed succesfuuly can be verified by giving openssl help, this will show what are all the options available in openssl to explore and use.



**Step-2: Encryption Of plain Text**

AES-encryptor.cpp

#include<stdio.h>

#include <openssl/evp.h>

#include <openssl/conf.h>

#include<string.h>

**int** encrypt(**unsigned** **char\*** text,**int** text\_len,**unsigned** **char\*** key,**unsigned** **char\***  cipher){

**int** cipher\_len=0;

**int** len=0;

    EVP\_CIPHER\_CTX\* ctx=EVP\_CIPHER\_CTX\_new();

    if(!ctx){

        perror("Error");

        exit(-1);

    }

    if(!EVP\_EncryptInit\_ex(ctx,EVP\_aes\_128\_ecb(),NULL,key,NULL)){ # aes ecb mode

        perror("Error in aes initailaization");

        exit(-1);

    }

    if(!EVP\_EncryptUpdate(ctx,cipher,&len,text,text\_len)){

        perror("Error in updation");

        exit(-1);

    }

    cipher\_len+=len;

    if(!EVP\_EncryptFinal\_ex(ctx,cipher +len ,&len)){

        perror("Error in execution");

        exit(-1);

    }

    cipher\_len +=len;

    EVP\_CIPHER\_CTX\_free(ctx);

    return cipher\_len;

}

This is the cpp program to encrypt using aes ecb mode in openssl.

**Plain text used:** Hello this is a confidential information regarding bank.

**Key used:**  **0123456789abbcdef**

**int** main(**int** argc,**char** **\*\***argv){

**unsigned** **char**\* key=(**unsigned** **char**\*)"0123456789abbcdef";

**unsigned** **char**\* text=(**unsigned** **char**\*)"Hello this is a confidential information regarding bank";

**int** text\_len=strlen((**const** **char**\*)text);

**unsigned** **char** cipher[64];

    printf("cipher=\n");

**int** cipher\_len=encrypt(text,text\_len,key,cipher);

    for(**int** i=0;i<cipher\_len;i++){

        printf("%02x ",cipher[i]);

    }

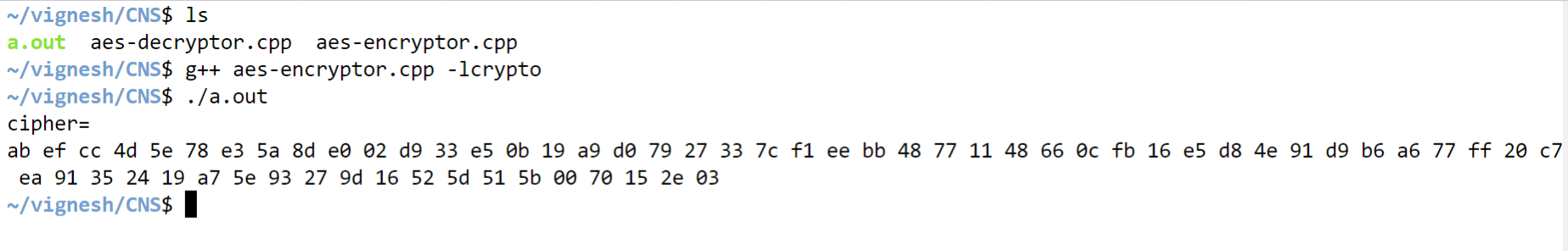
    printf("\n");

    return 0;

}

This is the driver program containing plaintext and key. When executed prints out the equivalent cipher text after performing aes encryption. All the ciphers are represented in hexaedecimal evnen if convert into ascii format we cannot understand anything useful from it so its used as such as hexadecimal for programming convenience to store in arrays.

**Step-3: Execution of Encryptor.**

****

**Plain text:** Hello this is a confidential information regarding bank.

**Cipher text obtained:**

**ab ef cc 4d 5e 78 e3 5a 8d e0 02 d9 33 e5 0b 19 a9 d0 79 27 33 7c f1 ee bb 48 77 11 48 66 0c fb 16 e5 d8 4e 91 d9 b6 a6 77 ff 20 c7 ea 91 35 24 19 a7 5e 93 27 9d 16 52 5d 51 5b 00 70 15 2e 03**

**Key used: 0123456789abbcdef**

When same key is used and cipher text is passed to decryptor we can get back the original data.

**Aes-decryptor.cpp:**

#include<stdio.h>

#include <openssl/evp.h>

#include <openssl/conf.h>

#include<string.h>

**int** decrypt(unigned **char\*** cipher,**int** cipher\_len,**unsigned** **char\*** key,**unsigned** **char\*** text){

**int** text\_len=0;

**int** len=0;

    EVP\_CIPHER\_CTX\* ctx=EVP\_CIPHER\_CTX\_new();

    if(!ctx){

        perror("Error in cipher ctx");

        exit(-1);

    }

     if(!EVP\_DecryptInit\_ex(ctx,EVP\_aes\_128\_ecb(),NULL,key,NULL)){

        perror("Error in aes decryption");

        exit(-1);

    }

    if(!EVP\_DecryptUpdate(ctx,text,&len,cipher,cipher\_len)){

        perror("errorin updation");

        exit(-1);

    }

    text\_len += len;

    if(!EVP\_DecryptFinal\_ex(ctx,text +len ,&len)){

        perror("error in execution");

        exit(-1);

    }

    text\_len+=len;

    EVP\_CIPHER\_CTX\_free(ctx);

    return text\_len;

}

**Step-4: Decryption of cipher text.**

**int** main(**int** argc,**char** **\*\***argv){

**unsigned** **char**\* key=(**unsigned** **char**\*)"0123456789abbcdef";

**unsigned** **char**\* text=(**unsigned** **char**\*)"Hello this is a confidential information regarding bank";

**int** text\_len=strlen((**const** **char**\*)text);

**unsigned** **char** cipher[64];

    printf("cipher=\n");

**int** cipher\_len=encrypt(text,text\_len,key,cipher);

    for(**int** i=0;i<cipher\_len;i++){

        printf("%02x ",cipher[i]);

    }

    printf("\n");

    printf("decrypted=\n");

**unsigned** **char** decrypted[64];

**int** dec\_len=decrypt(cipher,cipher\_len,key,decrypted);

    for(**int** i=0;i<dec\_len;i++){

        printf("%c",(**const** **char**)decrypted[i]);

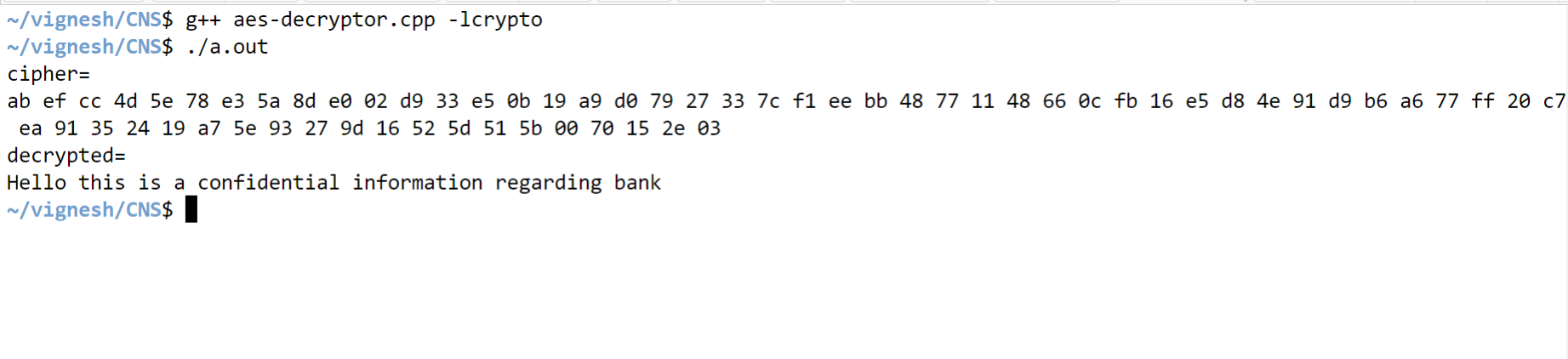
    }

    printf("\n");

    return 0;

}

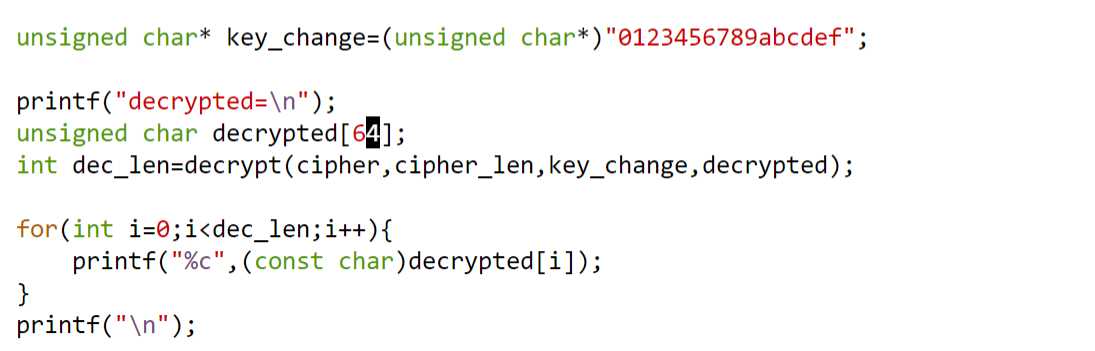
when cipher obtained from encryptor is passed to the decryptor function we can attain the plain text back if the keys are same.



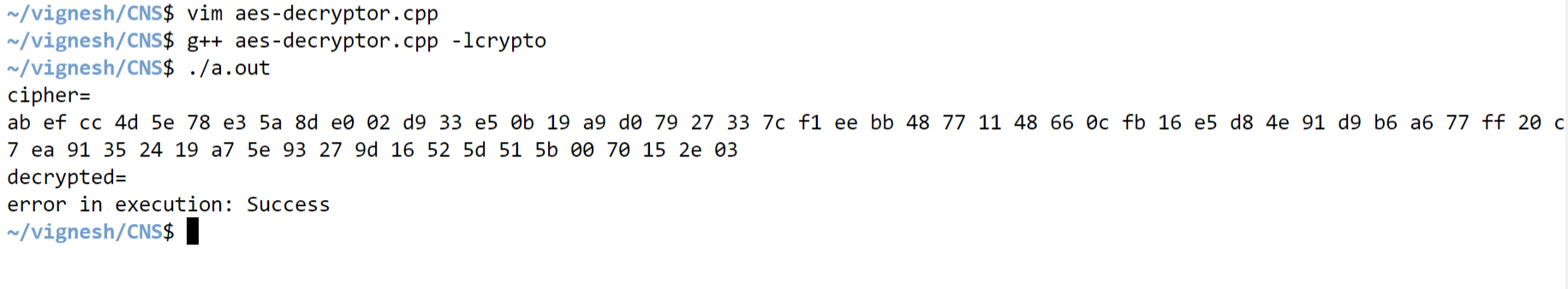
We have got back our original plain text **“ Hello this is a confidential information regarding bank”**

Once we change the key from **: 0123456789abbcdef** to **0123456789abcdef**

we get an error in execution of aes this proves that our aes encryptor is working fine and providing confidentiality.



**Execution with wrong password 0123456789abcdef**



**Conclusion:**

There are several other block cipher including symmetric, Asymmetric cryptography encryption standards available as functionalities, provided by OpenSSL which can be executed in similar fashion just by changing the function names, this openssl was introduced to maintain secure connection between application all over the world and also allows everyone to follow standard algorithms. Introduced in 1998 to provide set of encryption tools to overcome attacks by the hackers over computer network and safeguard the data. Various encryption standards provided are:

