**4CS401: Cryptography and Network Security**

**B.Tech. (CSE) – I [ 2022-23 ]**

**Assignment No - 7**

**Advanced Encryption Standard**

**Title: Advanced Encryption Standard**

**Aim: To Demonstrate Advanced Encryption Standard**

**Theory:**

**AES algorithm (Rijndael algorithm) is a symmetric block cipher algorithm. The length of the data packet must be 128 bits, and the length of the key used should be 128, 192 or 256 bits. For three AES algorithms with different key lengths, they are called "AES-128", "AES-192", "AES-256".**

**Code:**

#include <iostream>

#include <fstream>

#include <cstring>

#include <sstream>

#include "key\_expand.h"

#include "encoding.h"

#include "decoding.h"

#include <typeinfo>

#include <unistd.h>````````````````````````

using namespace std;

int main()

{

    // we will read from file input.txt

    int extendedlength = 0;

    int choice;

    string myText;

label:

    cout << "Welcome to 128 bits AES encryption" << endl;

    cout << endl;

    cout << "Enter you choice " << endl;

    cout << "1- Encoding" << endl;

    cout << "2- Decoding" << endl;

    cin >> choice;

    switch (choice)

    {

    case 1:

    {

        // encryption of text data

        ifstream File;

        string filepath = "encryption.aes";

        // clearing encryption.aes before editing

        File.open(filepath.c\_str(), std::ifstream::out | std::ifstream::trunc);

        if (!File.is\_open() || File.fail())

        {

            File.close();

            printf("\nError : failed to erase file content !");

        }

        File.close();

        // reading plain text from input.txt

        fstream newfile;

        newfile.open("input.txt", ios::in); // open a file to perform read operation using file object

        if (newfile.is\_open())

        { // checking whether the file is open

            cout << "Reading plain text from input.txt .........\n";

            usleep(1000);

            string tp;

            cout << "Reading KEY from key.txt ......\n";

            usleep(1000);

            cout << "Now encrypting ....\n";

            usleep(1000);

            cout << "writing encrypted data in encryption.aes ..\n";

            usleep(1000);

            cout << endl;

            while (getline(newfile, tp))

            {

                // read data from file object and put it into string.

                int messlength = tp.length();

                int extendedlength;

                if ((messlength % 16) != 0)

                {

                    extendedlength = messlength + (16 - (messlength % 16));

                }

                else

                {

                    extendedlength = messlength;

                }

                unsigned char \*encryptedtext = new unsigned char[extendedlength];

                for (int i = 0; i < extendedlength; i++)

                {

                    if (i < messlength)

                        encryptedtext[i] = tp[i];

                    else

                        encryptedtext[i] = 0;

                }

                // getting key from key.txt

                string k;

                ifstream infile;

                infile.open("key.txt");

                if (infile.is\_open())

                {

                    getline(infile, k); // The first line of file should be the key

                    infile.close();

                }

                else

                    cout << "Unable to open file";

                istringstream tempkey(k);

                unsigned char key[16];

                unsigned int x;

                for (int i = 0; i < 16; i++)

                {

                    tempkey >> hex >> x;

                    key[i] = x;

                }

                // extending key

                unsigned char extendedkeys[176];

                Key\_extenxion(key, extendedkeys);

                // encrypting our plain text

                for (int i = 0; i < extendedlength; i += 16)

                {

                    unsigned char \*temp = new unsigned char[16];

                    for (int j = 0; j < 16; j++)

                    {

                        temp[j] = encryptedtext[i + j];

                    }

                    encryption(temp, extendedkeys);

                    for (int j = 0; j < 16; j++)

                    {

                        encryptedtext[i + j] = temp[j];

                    }

                }

                // storing our encrypted data in encryption.aes

                ofstream fout; // Create Object of Ofstream

                ifstream fin;

                fin.open("encryption.aes");

                fout.open("encryption.aes", ios::app); // Append mode

                if (fin.is\_open())

                    fout << encryptedtext << "\n"; // Writing data to file

                fin.close();

                fout.close();

            }

            cout << "128-bit AES encryption is done sucessfully\n";

            cout << "Data has been appended to file encryption.aes";

            newfile.close(); // close the file object.

        }

        break;

    }

    case 2:

    {

        cout << "Reading encrypted data from encryption.txt .........\n";

        usleep(1000);

        string tp;

        cout << "Reading KEY from key.txt ......\n";

        usleep(1000);

        cout << "Now Decrypting ....\n";

        usleep(1000);

        cout << "writing decrypted data in outputtext.txt ..\n";

        usleep(1000);

        cout << endl;

        cout << "Following is our decrypted text:- \n";

        // clearing outputtext file

        ifstream File;

        string filepath = "outputtext.txt";

        File.open(filepath.c\_str(), std::ifstream::out | std::ifstream::trunc);

        if (!File.is\_open() || File.fail())

        {

            File.close();

            printf("\nError : failed to erase file content !");

        }

        File.close();

        ifstream MyReadFile;

        MyReadFile.open("encryption.aes", ios::in | ios::binary);

        if (MyReadFile.is\_open())

        {

            while (getline(MyReadFile, myText))

            {

                cout.flush();

                char \*x;

                x = &myText[0];

                int messlength = strlen(x);

                char \*msg = new char[myText.size() + 1];

                strcpy(msg, myText.c\_str());

                int n = strlen((const char \*)msg);

                unsigned char \*decryptedtext = new unsigned char[n];

                // decrypting our encrypted data

                for (int i = 0; i < n; i++)

                {

                    decryptedtext[i] = (unsigned char)msg[i];

                }

                // reading key from key.txt file

                string k;

                ifstream infile;

                infile.open("key.txt");

                if (infile.is\_open())

                {

                    getline(infile, k); // The first line of file should be the key

                    infile.close();

                }

                else

                    cout << "Unable to open file";

                istringstream tempkey(k);

                unsigned char key[16];

                unsigned int x1;

                for (int i = 0; i < 16; i++)

                {

                    tempkey >> hex >> x1;

                    key[i] = x1;

                }

                // extending key

                unsigned char extendedkeys[176];

                Key\_extenxion(key, extendedkeys);

                // decrypting our data

                for (int i = 0; i < messlength; i += 16)

                {

                    unsigned char \*temp = new unsigned char[16];

                    for (int j = 0; j < 16; j++)

                        temp[j] = decryptedtext[i + j];

                    decryption(temp, extendedkeys);

                    for (int j = 0; j < 16; j++)

                        decryptedtext[i + j] = temp[j];

                }

                // printing our plain text

                for (int i = 0; i < messlength; i++)

                {

                    cout << decryptedtext[i];

                    if (decryptedtext[i] == 0 && decryptedtext[i - 1] == 0)

                        break;

                }

                // storing plain text in outputtext.txt file

                cout << endl;

                ofstream fout; // Create Object of Ofstream

                ifstream fin;

                fin.open("outputtext.txt");

                fout.open("outputtext.txt", ios::app); // Append mode

                if (fin.is\_open())

                    fout << decryptedtext << "\n"; // Writing data to file

                fin.close();

                fout.close(); // Closing the file

                usleep(500);

            }

        }

        else

        {

            cout << "Can not open input file\n ";

        }

        cout << "\n Data has been appended to file outputtext.txt";

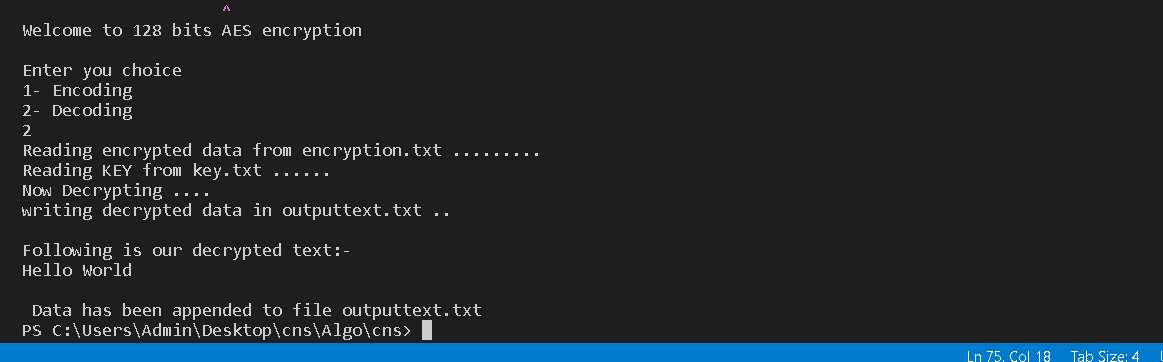
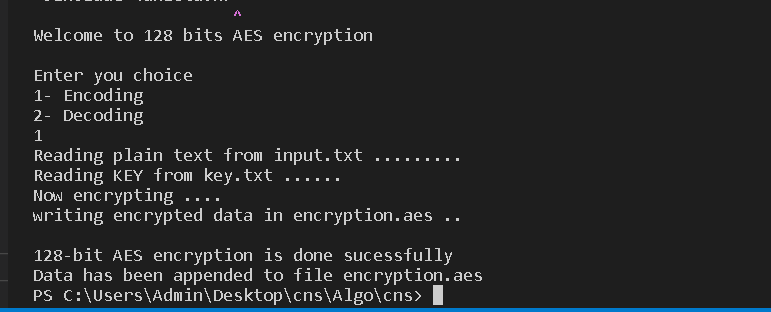
        MyReadFile.close();

        break;

    }

    }

}



**Conclusion:**

**AES instruction set is now integrated into the CPU (offers throughput of several GB/s) to improve the speed and security of applications that use AES for encryption and decryption. Even though it’s been 20 years since its introduction we have failed to break the AES algorithm as it is infeasible even with the current technology.**