**Project 3**

**1. Encryption source code**

#include <iostream>

#include <fstream>

#include <string>

#include <vector>

#include <sstream>

using namespace std;

vector<int> findKeyStream(string plain\_text, string key)

{

    int key\_size = key.length();

    int S[256];

    int K[256];

    // initialize S array

    for (int i = 0; i < 256; i++)

    {

        S[i] = i;

    }

    // initialize K array

    int j;

    for (int i = 0; i < 256; i++)

    {

        j = i % key\_size;

        K[i] = key[j];

    }

    j = 0;

    for (int i = 0; i < 256; i++)

    {

        j = (j + S[i] + K[i]) % 256;

        swap(S[i], S[j]);

    }

    j = 0;

    // Find keyStream

    int text\_size = plain\_text.length();

    vector<int>

        keyStream;

    int t;

    int index;

    for (int i = 0; i < text\_size; i++)

    {

        index = i;

        index = (index + 1) % 256;

        j = (j + S[index]) % 256;

        swap(S[index], S[j]);

        t = (S[index] + S[j]) % 256;

        keyStream.push\_back(S[t]);

    }

    return keyStream;

}

string decToHex(int dec)

{

    int i = 0;

    char hex[10];

    while (dec != 0)

    {

        int temp;

        temp = dec % 16;

        if (temp >= 10)

        {

            hex[i] = temp + 55;

            i++;

        }

        else

        {

            hex[i] = temp + 48;

            i++;

        }

        dec = dec / 16;

    }

    string str = "";

    for (int j = i - 1; j >= 0; j--)

        str = str + hex[j];

    return str;

}

vector<string> encryption(string plain\_text, vector<int> keyStream)

{

    // Xor and parse number to hex

    int text\_size = plain\_text.length();

    vector<string> hex\_vector;

    for (int i = 0; i < text\_size; i++)

    {

        hex\_vector.push\_back(decToHex(keyStream[i] ^ int(plain\_text[i])));

    }

    return hex\_vector;

}

int main()

{

    ifstream inputFile("text.txt");

    ofstream outputFile("hex.txt");

    string plain\_text;

    getline(inputFile, plain\_text);

    string key;

    cout << "Please enter a key: ";

    cin >> key;

    while (key.length() < 5 || key.length() > 32)

    {

        cout << "Please enter a key between 5 bytes and 32: ";

        cin >> key;

    }

    vector<int> keyStream = findKeyStream(plain\_text, key);

    string hex = "";

    vector<string> hex\_vector = encryption(plain\_text, keyStream);

    for (int i = 0; i < hex\_vector.size(); i++)

    {

        hex = hex + hex\_vector[i];

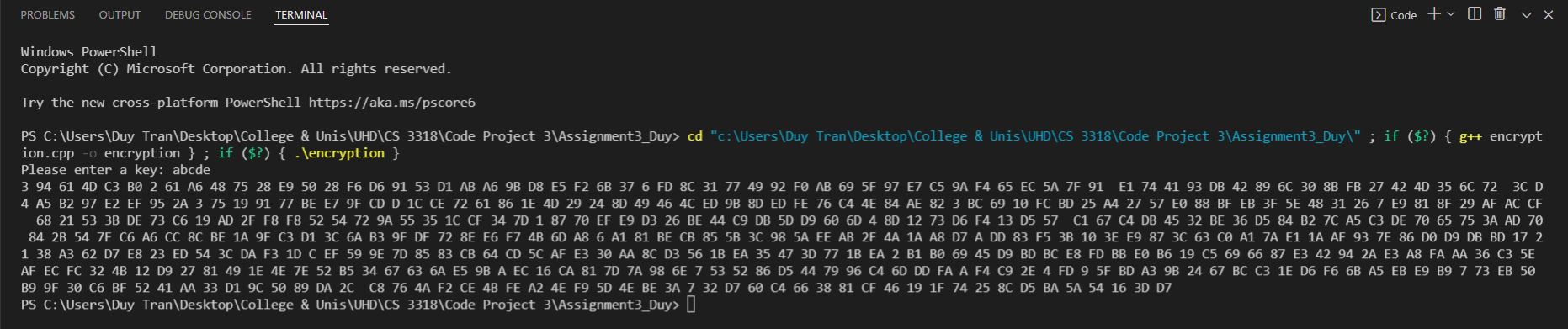
        hex = hex + " ";

    }

    cout << hex << endl;

    outputFile << hex;

}

**2. Encryption screenshot**

**3. Encryption cipher text**



4. **Decryption source code**

#include <iostream>

#include <fstream>

#include <string>

#include <vector>

#include <sstream>

#include <string.h>

using namespace std;

int hexToDec(string hex)

{

    int size = hex.length();

    int base = 1;

    int dec = 0;

    for (int i = size - 1; i >= 0; i--)

    {

        if (hex[i] >= 'A' && hex[i] <= 'F')

        {

            dec += (int(hex[i]) - 55) \* base;

            base = base \* 16;

        }

        else if (hex[i] >= '0' && hex[i] <= '9')

        {

            dec += (int(hex[i]) - 48) \* base;

            base = base \* 16;

        }

    }

    return dec;

}

vector<int> findKeyStream(string plain\_text, string key)

{

    int key\_size = key.length();

    int S[256];

    int K[256];

    // initialize S array

    for (int i = 0; i < 256; i++)

    {

        S[i] = i;

    }

    // initialize K array

    int j;

    for (int i = 0; i < 256; i++)

    {

        j = i % key\_size;

        K[i] = key[j];

    }

    j = 0;

    for (int i = 0; i < 256; i++)

    {

        j = (j + S[i] + K[i]) % 256;

        swap(S[i], S[j]);

    }

    j = 0;

    // Find keyStream

    int text\_size = plain\_text.length();

    vector<int>

        keyStream;

    int t;

    int index;

    for (int i = 0; i < text\_size; i++)

    {

        index = i;

        index = (index + 1) % 256;

        j = (j + S[index]) % 256;

        swap(S[index], S[j]);

        t = (S[index] + S[j]) % 256;

        keyStream.push\_back(S[t]);

    }

    return keyStream;

}

string decryption(vector<string> hex\_vector, vector<int> keyStream)

{

    string plain\_text = "";

    for (int i = 0; i < hex\_vector.size(); i++)

    {

        plain\_text = plain\_text + char(keyStream[i] ^ hexToDec(hex\_vector[i]));

    }

    return plain\_text;

}

int main()

{

    ifstream inputFile("hex.txt");

    string hex;

    getline(inputFile, hex);

    string key;

    cout << "Please enter a key: ";

    cin >> key;

    while (key.length() < 5 || key.length() > 32)

    {

        cout << "Please enter a key between 5 bytes and 32: ";

        cin >> key;

    }

    vector<int> keyStream = findKeyStream(hex, key);

    vector<string> hex\_vector;

    string a = "";

    for (int i = 0; i < hex.length(); i++)

    {

        if (int(hex[i]) != 32)

        {

            a = a + hex[i];

        }

        else if (int(hex[i] == 32))

        {

            hex\_vector.push\_back(a);

            a = "";

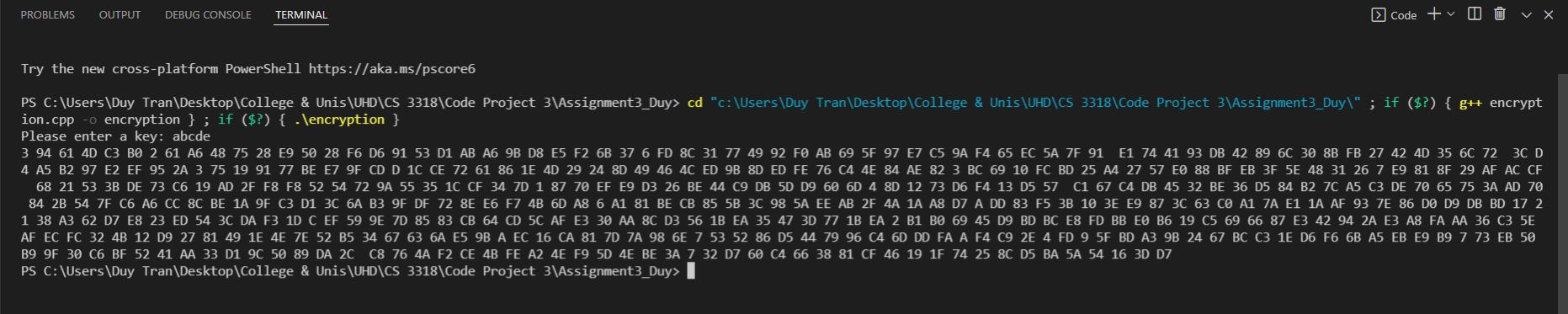
        }

    }

    string decrypted\_text = decryption(hex\_vector, keyStream);

    cout << decrypted\_text << endl;

}

**5. Decryption screenshot**

**6. Decryption plain text**

In cryptography, RC4 (Rivest Cipher 4 also known as ARC4 or ARCFOUR meaning Alleged RC4) is a stream cipher. While remarkable for its simplicity and speed in software, multiple vulnerabilities have been discovered in RC4, rendering it insecure. It is especially vulnerable when the beginning of the output keystream is not discarded, or when nonrandom or related keys are used. Particularly problematic uses of RC4 have led to very insecure protocols such as WEP.