This project will demonstrate the functionality of the server portion of a Client and Server based encryption service. Below are the necessary files required to program the server. For this project, use cloud9 IDE to better visualize what’s going on. Because this is a server without the client portal, ssh will be simulated by dropping files into the correct directories.

Place the fallowing contents after the # block into a file called Decrypt.cpp

##########################################################################################################################################################################

#include <iostream>

#include <cmath>

#include <algorithm>

#include <string>

#include <map>

#include <random>

#include <time.h>

#include <fstream>

#include <vector>

using namespace std;

void Clear\_Screen()

{

// If Windows

//system("CLS");

// If Linux

//system("clear");

}

char Conversion\_Table(int i)

{

std::map<int, char> Conversion;

Conversion[0] = 'A'; Conversion[1] = 'a'; Conversion[2] = 'B'; Conversion[3] = 'b'; Conversion[4] = 'C'; Conversion[5] = 'c';

Conversion[6] = 'D'; Conversion[7] = 'd'; Conversion[8] = 'E'; Conversion[9] = 'e'; Conversion[10] = 'F'; Conversion[11] = 'f';

Conversion[12] = 'G'; Conversion[13] = 'g'; Conversion[14] = 'H'; Conversion[15] = 'h'; Conversion[16] = 'I'; Conversion[17] = 'i';

Conversion[18] = 'J'; Conversion[19] = 'j'; Conversion[20] = 'K'; Conversion[21] = 'k'; Conversion[22] = 'L'; Conversion[23] = 'l';

Conversion[24] = 'M'; Conversion[25] = 'm'; Conversion[26] = 'N'; Conversion[27] = 'n'; Conversion[28] = 'O'; Conversion[29] = 'o';

Conversion[30] = 'P'; Conversion[31] = 'p'; Conversion[32] = 'Q'; Conversion[33] = 'q'; Conversion[34] = 'R'; Conversion[35] = 'r';

Conversion[36] = 'S'; Conversion[37] = 's'; Conversion[38] = 'T'; Conversion[39] = 't'; Conversion[40] = 'U'; Conversion[41] = 'u';

Conversion[42] = 'V'; Conversion[43] = 'v'; Conversion[44] = 'W'; Conversion[45] = 'w'; Conversion[46] = 'X'; Conversion[47] = 'x';

Conversion[48] = 'Y'; Conversion[49] = 'y'; Conversion[50] = 'Z'; Conversion[51] = 'z';

return Conversion.at(i);

}

vector<int> Set\_True\_Seed(vector<std::string> &x)

{

vector<int> True\_Seed;

for (int i = 0; i < x.size(); ++i)

{

True\_Seed.push\_back(stoi(x[i]));

}

int Spacing = (True\_Seed[0] + True\_Seed[1] \* True\_Seed[2] / True\_Seed[3] \* True\_Seed[4]) % 100;

int Shifting = (True\_Seed[5] + True\_Seed[6] \* True\_Seed[7] \* True\_Seed[8] + True\_Seed[9]) % 1000;

vector<int> Carry;

for (int i = 0; i < 2; ++i)

{

if (i == 0) { Carry.push\_back(Spacing); }

if (i == 1) { Carry.push\_back(Shifting); }

}

return Carry;

}

vector<string> String\_2\_Vector(string &x)

{

string Temp = x;

vector<string> y;

for (int i = 0; i < Temp.size(); ++i)

{

string x2;

x2 = Temp[i];

y.push\_back(x2);

}

return y;

}

void Print\_Vector(vector<string> &x)

{

Clear\_Screen();

for (int i = 0; i < x.size(); ++i)

{

std::cout << x[i] << std::endl;

}

std::string i;

cin >> i;

Clear\_Screen();

}

int main()

{

fstream Extract\_Seed;

fstream Read\_Encripted\_File;

fstream Write\_Translated\_File;

std::ofstream XML{ "Translated.txt" };

Extract\_Seed.open("ECR.vvn");

Write\_Translated\_File.open("Translated.txt");

string seed;

getline(Extract\_Seed, seed);

Extract\_Seed.close();

vector<string> Vector\_Seed = String\_2\_Vector(seed);

vector<int> True\_Seed = Set\_True\_Seed(Vector\_Seed);

char Temp;

string Number\_Check;

int Real\_Value;

string junk;

Read\_Encripted\_File.open("ECR.vvn");

getline(Read\_Encripted\_File, junk);

while (Read\_Encripted\_File.good())

{

Read\_Encripted\_File.get(Temp);

if (Temp == '\0')

{

Read\_Encripted\_File.get(Temp);

while (Temp != '\0')

{

Number\_Check = Number\_Check + Temp;

Read\_Encripted\_File.get(Temp);

}

Real\_Value = stoi(Number\_Check);

Write\_Translated\_File << Conversion\_Table(Real\_Value - True\_Seed[1]);

Number\_Check.clear();

}

else

if (Temp == ' ')

{

Write\_Translated\_File << " ";

}

else

if (Temp == '\n')

{

Write\_Translated\_File << "\n";

}

else

{

Write\_Translated\_File << Temp;

}

}

Write\_Translated\_File.close();

}

##########################################################################################################################################################################

**Place the fallowing contents after the #block into a file called Encrypt.cpp**

##########################################################################################################################################################################

#include <iostream>

#include <cmath>

#include <algorithm>

#include <string>

#include <map>

#include <random>

#include <time.h>

#include <fstream>

#include <vector>

using namespace std;

void Clear\_Screen()

{

// If Windows

system("CLS");

// If Linux

//system("clear");

}

string Create\_Random\_Number\_String()

{

srand(time(NULL));

string Random\_Name;

for (int i = 0; i < 10; ++i)

{

Random\_Name = Random\_Name + to\_string(rand() % 10);

}

return Random\_Name;

}

vector<string> Get\_Config\_Information()

{

string Really\_Temp;

string Line;

vector<string> Hold\_Cfg;

std::ifstream Conf\_File;

Conf\_File.open("XxPre\_SetxX.cfg");

while (getline(Conf\_File, Line)) {

Really\_Temp = Line;

Hold\_Cfg.push\_back(Really\_Temp);

Really\_Temp.clear();

}

return Hold\_Cfg;

}

void Print\_Vector(vector<string> &x)

{

Clear\_Screen();

for (int i = 0; i < x.size(); ++i)

{

std::cout << x[i] << std::endl;

}

std::string i;

cin >> i;

Clear\_Screen();

}

void Print\_Vector(vector<int> &x)

{

Clear\_Screen();

for (int i = 0; i < x.size(); ++i)

{

std::cout << x[i] << std::endl;

}

int i;

cin >> i;

Clear\_Screen();

}

vector<int> Set\_True\_Seed(vector<std::string> &x)

{

vector<int> True\_Seed;

for (int i = 0; i < x.size(); ++i)

{

True\_Seed.push\_back(stoi(x[i]));

}

int Spacing = (True\_Seed[0] + True\_Seed[1] \* True\_Seed[2] / True\_Seed[3] \* True\_Seed[4]) % 100;

int Shifting = (True\_Seed[5] + True\_Seed[6] \* True\_Seed[7] \* True\_Seed[8] + True\_Seed[9]) % 1000;

vector<int> Carry;

for (int i = 0; i < 2; ++i)

{

if (i == 0) { Carry.push\_back(Spacing); }

if (i == 1) { Carry.push\_back(Shifting); }

}

return Carry;

}

vector<string> String\_2\_Vector(string &x)

{

string Temp = x;

vector<string> y;

for (int i = 0; i < Temp.size(); ++i)

{

string x2;

x2 = Temp[i];

y.push\_back(x2);

}

return y;

}

int Conversion\_Table(char i)

{

std::map<char, int> Conversion;

Conversion['A'] = 0; Conversion['a'] = 1; Conversion['B'] = 2; Conversion['b'] = 3; Conversion['C'] = 4; Conversion['c'] = 5;

Conversion['D'] = 6; Conversion['d'] = 7; Conversion['E'] = 8; Conversion['e'] = 9; Conversion['F'] = 10; Conversion['f'] = 11;

Conversion['G'] = 12; Conversion['g'] = 13; Conversion['H'] = 14; Conversion['h'] = 15; Conversion['I'] = 16; Conversion['i'] = 17;

Conversion['J'] = 18; Conversion['j'] = 19; Conversion['K'] = 20; Conversion['k'] = 21; Conversion['L'] = 22; Conversion['l'] = 23;

Conversion['M'] = 24; Conversion['m'] = 25; Conversion['N'] = 26; Conversion['n'] = 27; Conversion['O'] = 28; Conversion['o'] = 29;

Conversion['P'] = 30; Conversion['p'] = 31; Conversion['Q'] = 32; Conversion['q'] = 33; Conversion['R'] = 34; Conversion['r'] = 35;

Conversion['S'] = 36; Conversion['s'] = 37; Conversion['T'] = 38; Conversion['t'] = 39; Conversion['U'] = 40; Conversion['u'] = 41;

Conversion['V'] = 42; Conversion['v'] = 43; Conversion['W'] = 44; Conversion['w'] = 45; Conversion['X'] = 46; Conversion['x'] = 47;

Conversion['Y'] = 48; Conversion['y'] = 49; Conversion['Z'] = 50; Conversion['z'] = 51;

return Conversion.at(i);

}

std::string Vetor\_Seed\_2\_String(vector<string> &x)

{

std::string carry;

for (int i = 0; i < x.size(); ++i)

{

carry = carry + x[i];

}

return carry;

}

int main()

{

std::fstream New\_File;

std::vector<string> V\_CFV;

std::vector<string> Vector\_Seed;

std::vector<int> True\_Seed;

string Security\_Level;

int shifting = 0;

int spacing = 0;

bool Is\_Random = true;

V\_CFV = Get\_Config\_Information();

Security\_Level = (V\_CFV[1]);

//Print\_Vector(V\_CFV);

Vector\_Seed = String\_2\_Vector(V\_CFV[0]);

//Print\_Vector(Vector\_Seed);

Security\_Level = V\_CFV[1];

True\_Seed = Set\_True\_Seed(Vector\_Seed);

//Print\_Vector(True\_Seed);

ifstream Old\_File;

Old\_File.open("Working\_With.txt");

string File\_Name = "ECR.vvn";

ofstream file{ File\_Name };

New\_File.open(File\_Name);

if (!Old\_File.is\_open())

{

cout << "Fuck!" << std::endl;

}

New\_File << Vetor\_Seed\_2\_String(Vector\_Seed);

New\_File << endl;

char char\_data\_hold;

while (Old\_File.get(char\_data\_hold))

{

if (char\_data\_hold >= 'a' && char\_data\_hold <= 'z' || char\_data\_hold >= 'A' && char\_data\_hold <= 'Z')

{

char b = '\0';

New\_File << b;

New\_File << Conversion\_Table(char\_data\_hold) + True\_Seed[1];

New\_File << b;

cout << char\_data\_hold;

}

else

if (char\_data\_hold == ' ')

{

New\_File << " ";

cout << " ";

}

else

if (char\_data\_hold == '\n')

{

New\_File << endl;

cout << endl;

}

else

{

New\_File << char\_data\_hold;

cout << char\_data\_hold;

}

}

New\_File.close();

}

##########################################################################################################################################################################

**Place the fallowing content after the #block into a file called Settings.cpp**

##########################################################################################################################################################################

#include <iostream>

#include <cmath>

#include <algorithm>

#include <string>

#include <map>

#include <random>

#include <time.h>

#include <fstream>

using namespace std;

void Clear\_Screen()

{

// If Windows

//system("CLS");

// If Linux

system("clear");

}

std::string Trunk\_The\_Key(string x)

{

string y = "";

for (int i = 0; i < 10; ++i)

{

y = y + x[i];

}

return y;

}

std::string Padding(string x)

{

srand(time(NULL));

for (int i = 0; i < 10; ++i)

{

x = x + to\_string(rand() % 10);

}

return x;

}

void Send\_To\_File(string x, string y)

{

std::ofstream file{ "XxPre\_SetxX.cfg" };

fstream Pre\_Set;

Pre\_Set.open("XxPre\_SetxX.cfg");

for (int i = 0; i < 2; ++i)

{

if (i == 0) { Pre\_Set << x << std::endl; }

if (i == 1) { Pre\_Set << y << std::endl; }

}

Pre\_Set.close();

}

std::string Set\_To\_Basic(string Level\_Of\_Security)

{

string Hold\_Option;

while (Hold\_Option != "2")

{

Clear\_Screen();

std::cout << "The Basic Level Of Security Will Set A Random Seed With A Fixed Calcualtor. (Will Not Stop Brute Force Attacks!)" << std::endl;

std::cout << "[1] Enable Basic Security" << std::endl;

std::cout << "[2] Back" << std::endl;

cin >> Hold\_Option;

if (Hold\_Option == "1")

{

Level\_Of\_Security = "Basic";

}

}

return Level\_Of\_Security;

}

std::string Set\_To\_Intermediate(string Level\_Of\_Security)

{

string Hold\_Option;

while (Hold\_Option != "2")

{

Clear\_Screen();

std::cout << "The Intermediate Level Of Security Will Set A Random Seed With A Random Calcualtor. (Will Not Stop Brute Force Attacks!)" << std::endl;

std::cout << "[1] Enable Intermediate Security" << std::endl;

std::cout << "[2] Back" << std::endl;

cin >> Hold\_Option;

if (Hold\_Option == "1")

{

Level\_Of\_Security = "Intermediate";

}

}

return Level\_Of\_Security;

}

std::string Set\_To\_Advanced(string Level\_Of\_Security)

{

string Hold\_Option;

while (Hold\_Option != "2")

{

Clear\_Screen();

std::cout << "The Advanced Level Of Security Will Set A Random Seed With A Random Calcualtor While Taking Some Of The File And Moving It To The Key." << std::endl;

std::cout << "[1] Enable Advanced Security" << std::endl;

std::cout << "[2] Back" << std::endl;

cin >> Hold\_Option;

if (Hold\_Option == "1")

{

Level\_Of\_Security = "Advanced";

}

}

return Level\_Of\_Security;

}

std::string Set\_Level\_Of\_Secutriy(string Level\_Of\_Security)

{

string Hold\_Option;

while (Hold\_Option != "4")

{

Clear\_Screen();

std::cout << "Current Level Of Security Set To =======> [" << Level\_Of\_Security << "]\n";

std::cout << "There are three levels of secutriy. Please Select Once!\n[By Default, Encription Security Will Be Set To Intermediate Upon Leaving This Menu]\n";

std::cout << "[1] Basic" << std::endl;

std::cout << "[2] Intermediate" << std::endl;

std::cout << "[3] Advanced" << std::endl;

std::cout << "[4] Back" << std::endl;

cin >> Hold\_Option;

if (Hold\_Option == "1")

{

Level\_Of\_Security = Set\_To\_Basic(Level\_Of\_Security);

}

if (Hold\_Option == "2")

{

Level\_Of\_Security = Set\_To\_Intermediate(Level\_Of\_Security);

}

if (Hold\_Option == "3")

{

Level\_Of\_Security = Set\_To\_Advanced(Level\_Of\_Security);

}

if (Hold\_Option == "4")

{

return Level\_Of\_Security;

}

}

}

void Print\_Converted\_Encription\_Key(string x)

{

Clear\_Screen();

std::cout << x << endl;

std::cout << "Enter a character to continue;" << endl;

string i = "";

cin >> i;

}

void Print\_Standard\_Encription\_Key(string x)

{

Clear\_Screen();

std::cout << x << endl;

std::cout << "Enter a character to continue;" << endl;

string i = "";

cin >> i;

}

std::string Set\_Seed\_Func()

{

Clear\_Screen();

string Seed = "";

string True\_Seed = "";

std::cout << "Please Enter Seed Below\n";

cin >> Seed;

for (int i = 0; i < Seed.size() - 1; ++i)

{

True\_Seed = (Seed + True\_Seed[i]);

}

True\_Seed = Padding(True\_Seed);

return Trunk\_The\_Key(True\_Seed);

}

void Main\_Menu()

{

std::cout << "Welcome To The Encryption Software" << std::endl;

std::cout << "[1] Set Seed" << std::endl;

std::cout << "[2] Set Level Of Security" << std::endl;

std::cout << "[3] Quit Program" << std::endl;

}

std::string Set\_Seed(string x)

{

string Absolute\_Seed = "";

string Option = "";

string Encription\_Block = x;

while (Option != "2")

{

Clear\_Screen();

std::cout << "The Seed Will Randomize The Encription. If Left [Blank], One Will Be Generated For You." << std::endl;

std::cout << "Current Encription Seed [" << Encription\_Block[0] << "]" << "[" << Encription\_Block[1] << "]" << "[" << Encription\_Block[2] << "]" << "[" << Encription\_Block[3] << "]" << "[" << Encription\_Block[4] << "]" << "[" << Encription\_Block[5] << "]" << "[" << Encription\_Block[6] << "]" << "[" << Encription\_Block[7] << "]" << "[" << Encription\_Block[8] << "]" << "[" << Encription\_Block[9] << "]";

for (int i = 0; i < 2; ++i)

{

std::cout << "\n" << std::endl;

}

std::cout << "[1] Change Seed" << std::endl;

std::cout << "[2] Back" << std::endl;

cin >> Option;

if (Option == "1")

{

Clear\_Screen;

Encription\_Block = Set\_Seed\_Func();

}

if (Option == "2")

{

break;

}

}

return Trunk\_The\_Key(Encription\_Block);

}

std::string Conversion(string x)

{

srand(time(NULL));

string To\_Hold = "";

bool Is\_Nothing = true;

int Number\_Hold = 0;

std::map<char, int> Conversion;

string Converted\_Encription\_Key = "";

Conversion['A'] = 0; Conversion['a'] = 0; Conversion['B'] = 1; Conversion['b'] = 1; Conversion['C'] = 2; Conversion['c'] = 2;

Conversion['d'] = 3; Conversion['D'] = 3; Conversion['E'] = 4; Conversion['e'] = 4; Conversion['F'] = 5; Conversion['f'] = 5;

Conversion['G'] = 6; Conversion['g'] = 6; Conversion['H'] = 7; Conversion['h'] = 7; Conversion['I'] = 8; Conversion['i'] = 8;

Conversion['J'] = 9; Conversion['j'] = 9; Conversion['K'] = 0; Conversion['k'] = 0; Conversion['L'] = 1; Conversion['l'] = 1;

Conversion['M'] = 2; Conversion['m'] = 2; Conversion['N'] = 3; Conversion['n'] = 3; Conversion['O'] = 4; Conversion['o'] = 4;

Conversion['P'] = 5; Conversion['p'] = 5; Conversion['Q'] = 6; Conversion['q'] = 6; Conversion['R'] = 7; Conversion['r'] = 7;

Conversion['S'] = 8; Conversion['s'] = 8; Conversion['T'] = 9; Conversion['t'] = 9; Conversion['U'] = 0; Conversion['u'] = 0;

Conversion['V'] = 1; Conversion['v'] = 1; Conversion['W'] = 2; Conversion['w'] = 2; Conversion['X'] = 3; Conversion['x'] = 3;

Conversion['Y'] = 4; Conversion['y'] = 4; Conversion['Z'] = 5; Conversion['z'] = 5;

for (int i = 0; i < x.size(); ++i)

{

int Random\_Number = rand() % 10;

Is\_Nothing = true;

string Convert\_To\_Int = to\_string(x[i]);

int CN = (stoi(Convert\_To\_Int) - 48);

cout << CN;

if (CN > -1 || CN < 10)

{

To\_Hold = x[i];

}

else

if (x[i] >= 'a' && x[i] <= 'z' || x[i] >= 'A' && x[i] <= 'Z')

{

Number\_Hold = Conversion.at(x[i]);

To\_Hold = to\_string(Number\_Hold);

}

else

{

Is\_Nothing = false;

}

if (Is\_Nothing == 1)

{

To\_Hold = to\_string(Random\_Number);

}

Converted\_Encription\_Key = Converted\_Encription\_Key + To\_Hold;

To\_Hold.clear();

}

return Trunk\_The\_Key(Converted\_Encription\_Key);

}

int main()

{

string Level\_Of\_Security = "Intermediate";

string Absolute\_Seed = "0000000000";

string Option = "";

bool is\_Off = 0;

while (is\_Off != 1)

{

Clear\_Screen();

Main\_Menu();

//cin >> Option;

Option = "3";

if (Option == "1")

{

Absolute\_Seed = Set\_Seed(Absolute\_Seed);

}

if (Option == "2")

{

Level\_Of\_Security = Set\_Level\_Of\_Secutriy(Level\_Of\_Security);

}

if (Option == "3")

{

if (Absolute\_Seed == "0000000000")

{

srand(time(NULL));

Absolute\_Seed.clear();

for (int i = 0; i < 10; ++i)

{

Absolute\_Seed = Absolute\_Seed + to\_string(rand() % 10);

}

}

Absolute\_Seed = Conversion(Absolute\_Seed);

Send\_To\_File(Absolute\_Seed, Level\_Of\_Security);

return 0;

}

if (Option == "-4")

{

Absolute\_Seed = Conversion(Absolute\_Seed);

}

if (Option == "-5")

{

Print\_Standard\_Encription\_Key(Absolute\_Seed);

}

}

}

##########################################################################################################################################################################

**Place the fallowing contents after the #block into a file called E.sh**

##########################################################################################################################################################################

#!/bin/bash

IFS=','

HomDir="$(pwd)"

mkdir encrypt

mkdir Lock

cd Lock

Dim="$(pwd)"

cd $HomDir

cd Engine

mv $HomDir/Lock/\* $HomDir/Engine/Working\_With.txt

mv $HomDir/Lock/\* $HomDir/Engine/Working\_With.txt

g++ -std=c++11 Settings.cpp -o Sh

./Sh

g++ -std=c++11 Encrypt.cpp -o Ec

./Ec

mv $HomDir/Engine/ECR.vvn $HomDir/encrypt

rm Sh

rm Ec

rm Working\_With.txt

rm XxPre\_SetxX.cfg

cd $HomDir

./b.sh

##########################################################################################################################################################################

**Place the fallowing content after the #block into a file named ServerStart.sh**

##########################################################################################################################################################################

#!/bin/bash

mkdir Holding

mkdir Keys

mkdir Main

mkdir Response

chmod 700 B.sh

Dim="$(pwd)"

File\_Dir=$Dim/Holding

File\_Type\_Encrypt="Request.exp"

File\_Type\_Decrypt="Request.dxp"

File\_Test=""

File\_Key=""

while :

do

printf "\033c"

printf "Encription Server Running!\n"

pwd

#if [ "$(ls -A Q )" ]; #While Something Is In The Q Folder

if [ "$(ls -A Holding )" ]; #While Something Is In The Holding Folder!

then

if [ -f "$File\_Dir/$File\_Type\_Decrypt" ];

then

printf "Lets start to Decrypt!\n"

cd "$Dim/Holding"

File\_Test=$(cat $File\_Type\_Decrypt | sed -n '1p')

File\_Pars=${File\_Test::-4}

cd "$Dim/Keys"

if [ -f "$Dim/Keys/$File\_Pars" ];

then

printf "Can Decrypt!\n"

Saving\_The\_Key=$(head -1 $File\_Pars)

cd "$Dim/Engine"

cd "$Dim"

mv "$Dim/Holding/$File\_Test" "$Dim/Engine/"

cd "$Dim/Engine/"

mv $File\_Test ECR.vvn

sed -i "1s/^/$Saving\_The\_Key\n/" ECR.vvn

g++ -std=c++11 Decrypt.cpp -o Dc

./Dc

sleep 3

cd "$Dim/Keys"

File\_Pars=${File\_Test::-4}

rm $File\_Pars

cd "$Dim/Holding"

rm $File\_Type\_Decrypt

cd "$Dim/Engine"

rm Dc

rm ECR.vvn

mv Translated.txt "$Dim/Response"

rm Translated.txt

#ls

else

printf "Can't Decrypt!\n"

fi

pwd

fi

if [ -f "$File\_Dir/$File\_Type\_Encrypt" ];

then

printf "Lets start to Encrypt!\n"

cd "$Dim/Holding"

File\_Test=$(cat $File\_Type\_Encrypt | sed -n '1p')

File\_Pars=${File\_Test::-4}

mv $File\_Test "$Dim/Lock"

cd "$Dim/Keys"

touch $File\_Test

cd "$Dim"

./E.sh

cd "$Dim/encrypt"

File\_Key=$(cat ECR.vvn | sed -n '1p')

echo File\_Key

sed -i '1d' ECR.vvn

mv ECR.vvn $File\_Test.vvn

mv $File\_Test.vvn "$Dim/Response"

cd "$Dim/Holding"

rm $File\_Type\_Encrypt

rm $File\_Pars

cd "$Dim"

cd "$Dim/Keys"

echo $File\_Key > $File\_Test

fi

#exit 1

fi

File\_Test=""

File\_Key=""

cd "$Dim"

done

rm $File\_Test

sleep 3

rmdir Holding

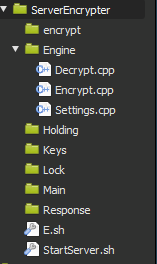
rmdir Keys

rmdir Main

rmdir Response

##########################################################################################################################################################################

**Make Sure the files are placed into directories as shown exactly below. (Some directories are created automatically when the server is started. To be safe, create them yourself.**



**The “Engine” directory of the server will hold the encryption engine necessary for the server to handle requests.**

**Decrypt.cpp is called when a decrypt request is sent to the server.**

**Both Settings.cpp and Encrypt.cpp are called when a encryption request is sent to the server. The Settings.cpp will set a random seed and push it to a XxPre\_SetxX.cfg file alongside the contents of the file needing to be encrypted so that the Encrypt.cpp program can Encrypt it. \*Most of the .cfg files will only last for a few seconds while the encryption request is taking place!\***

**The “Holding” directory will be were the Requests are stored for the server to handle. There are two types of requests.**

**Request.dxp (Request for Decryption)**

**Request.exp (Request for Encryption)**

**Each type of request requires a target file. Inside the “Request.exp” file, the target is simply the name.**

**Inside the “Request.dxp” file, the target is the name of the file fallowed by .vvn \*EXAMPLE.vvn\***

**For now, the .vvn file must also be moved to the “Holding” directory if the said file must be decrypted.**

**The “Keys” Directory will contain the keys required to decrypt files. If a key is missing, the file is skipped and the request is removed. If a file has been decrypted successfully , the key linked to the file will be removed so that it does not waste space.**

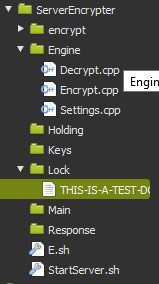
**The “Lock” Directory will contain files targeted for encryption. Without a target, these files will not doing anything.**

**The “Main” Directory is unused and will be added onto at a later time.**

**The “Response” Directory will contain all files marked to be sent back to a client. Since we did not set a up a client portal, this folder will act as the output for both encryption and decryption requests.**

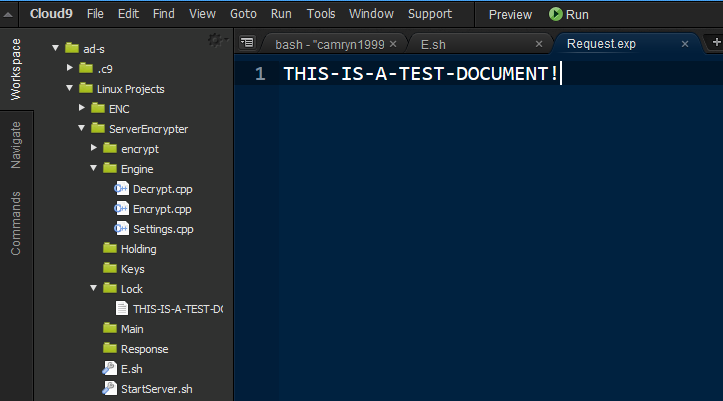
**The “E.sh” script preps targeted files for the encryption engine so that the engine understands how to handle it.**

**The “StartServer” script is the script that starts the server. Once on, the terminal will be sent into a indefinite loop where it checks the “Holding” directory for requests. To quit, press CTRL+C.**

**To test this project, you must first choose a file to encrypt. This file should only contain printable characters. Place the said file into the “Lock” directory. It should look like this.** 

**Once done, you must mark the file for encryption by targeting it with a request. While the server is off, create a file called “Request.exp” and place it outside the “Holding” directory.**

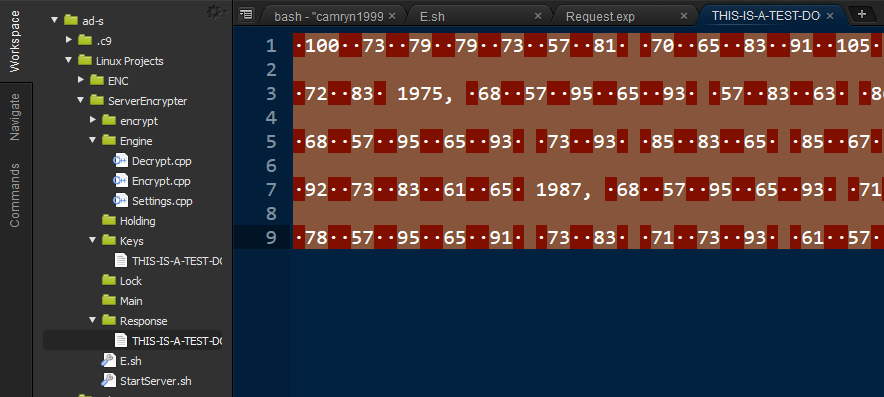
**Open the file and add the name of the file you placed inside the lock folder as the first line of the “Request.exp” Make sure to save the file.**



**Start the server by using ./StartServer.sh**

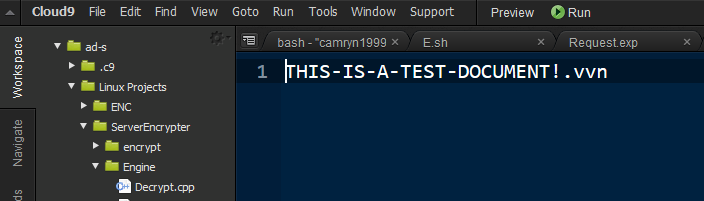
**Move “Request.exp” into the “Holding” Directory.**

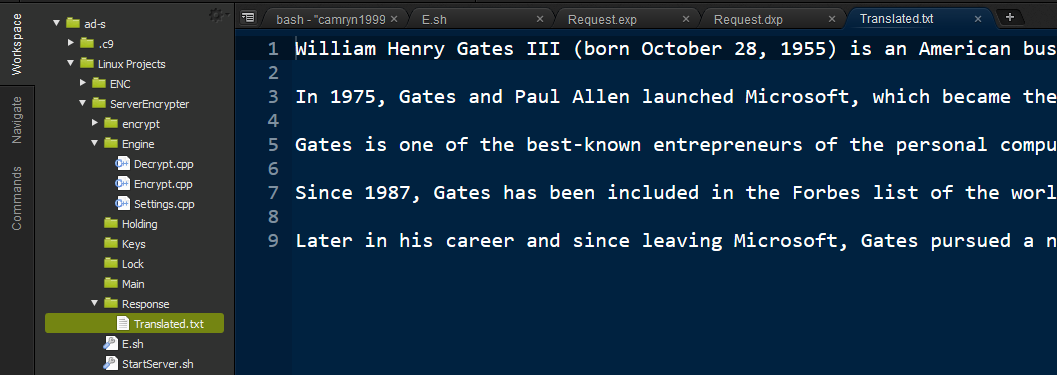
**If done right, after about 1-10 seconds (depending how large your file was), you will see a new file appear in the “Response” directory. This file should be encrypted and have the ending of .vvn. Notice how a KEY was generated in the “Keys” directory.**



**For dencryption, you must move the .vvn file into “Holding”**

**Secdonly, you must create a Request.dxp outside the “Holding” directory. Once done, open the “Request.dxp” file and the name of the targetd file fallowed by .vvn. Here is an example.**



**Once done, move the “Request.dxp” into the “Holding” directory. After some time, you should have a translated.txt pop up in the “Response” directory. Notice that the key has also disappeared.** 

**If you have done all the steps correctly, you will have both encrypted and decrypted a file utilizing this server. The server will handle cleanup after every request to avoid filling the memory.**