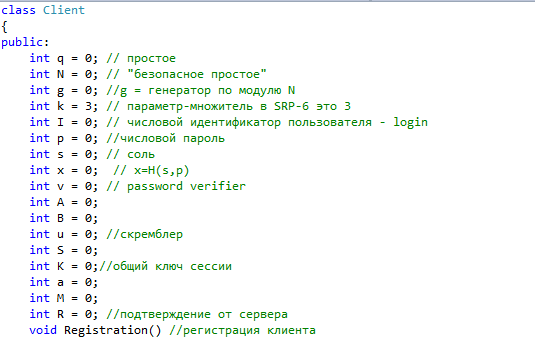
**Садов Д.В. ИВБО 7-14.**

**Лабораторная работа №6** В данной лабораторной работе нужно было реализовать алгоритм SRP.

При создании программы я руководствовался статьей - <https://habrahabr.ru/post/121021/>

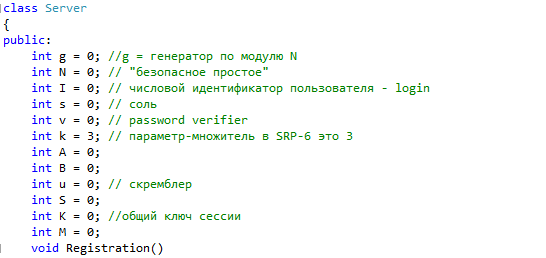
1. Создаются 2 класса – Client и Server со своими полями и методами

Класс Client:



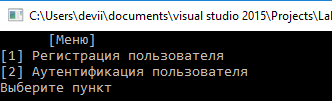


Класс Server:



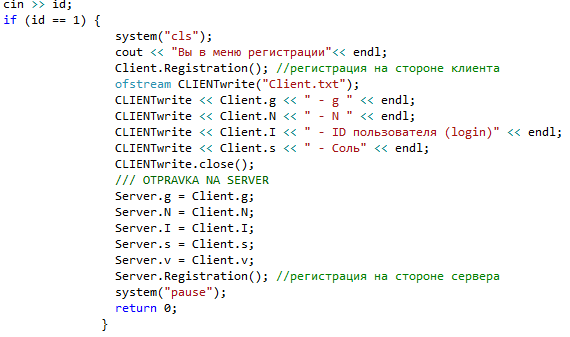


1. При запуске программы пользователю будет показано меню :



Рассмотрим первый пункт – регистрацию пользователя

При выборе будет выполнятся следующий блок команд:

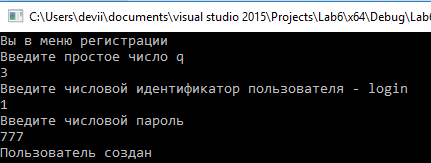


Регистрация на стороне клиента реализована в методе Registration

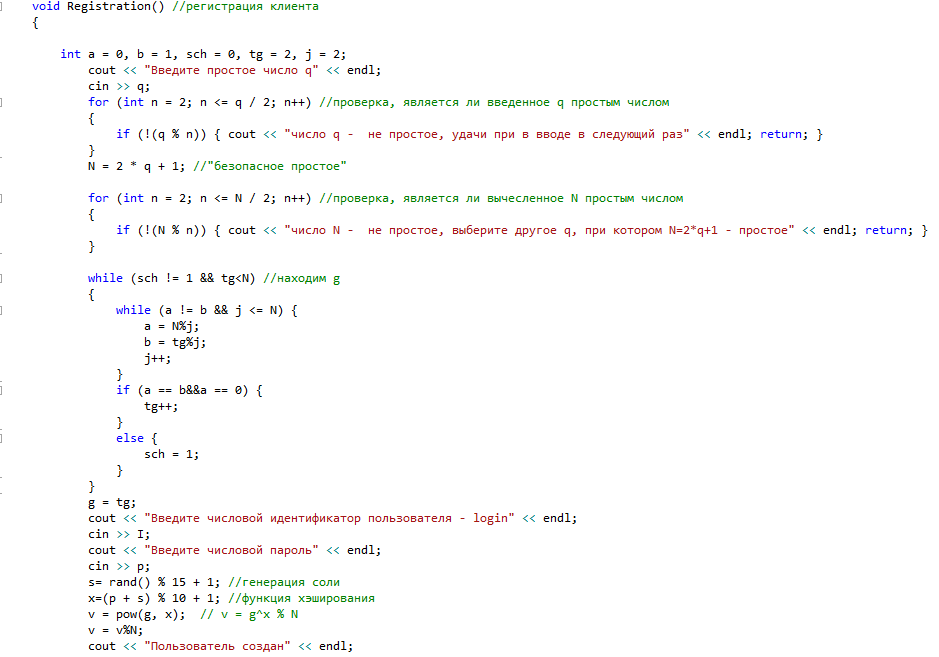
Пользователь вводит простое число, после чего осуществляется проверка, действительно ли оно простое. Далее вычисляется «безопасное простое» число N=2\*q+1 и также осуществляется его проверка. Следом вычисляем генератор по модулю N : g. После этого пользователю необходимо будет заполнить следующие поля: I- числовой идентификатор пользователя (login), числовой пароль. Далее рандомно генерируется соль s

и осуществляется хэширование следующей функцией x=(s+p)%10+1, где p – введенный пароль. Далее считается password verifier по формуле v=g^x % N.

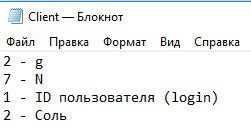
После этого пользователю выдается на экран сообщение о созданном пользователе



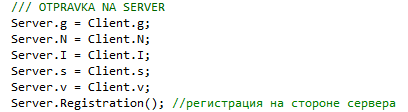
Сам метод выглядит так:



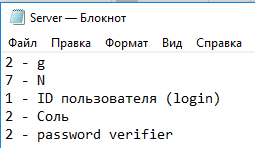
Полученные данные а именно g, N, I – login , s записываются в файле Client.txt



После этого данные пересылаются на сервер и осуществляется регистрация на стороне сервера

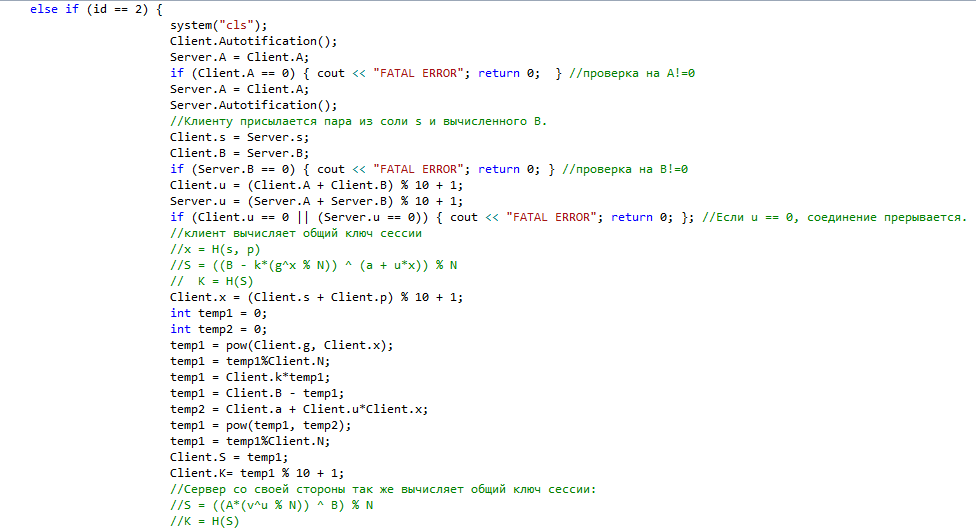


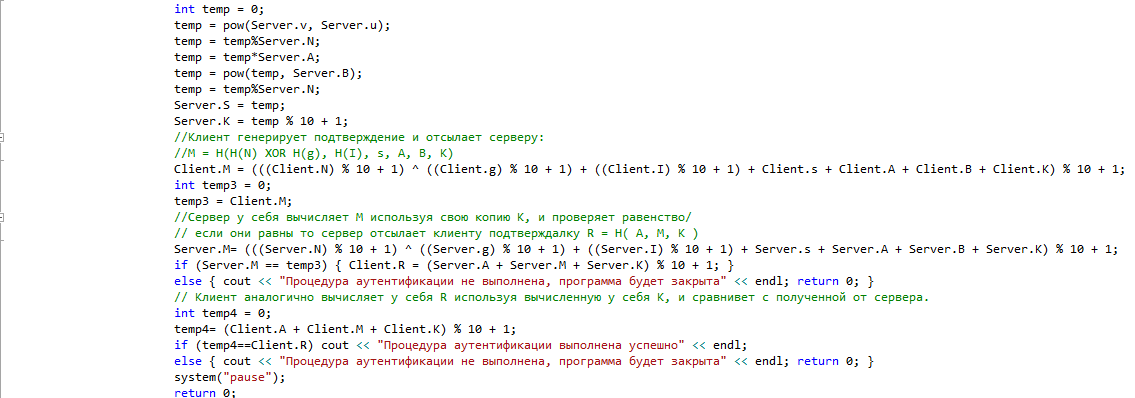
Метод Registration записывает эти данные в файл Server.txt



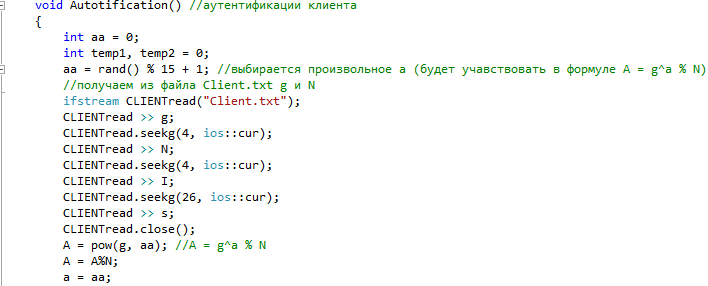
1. Теперь про аутентификацию.

При выборе в меню пункта аутентификации будет выполнен следующий блок команд:



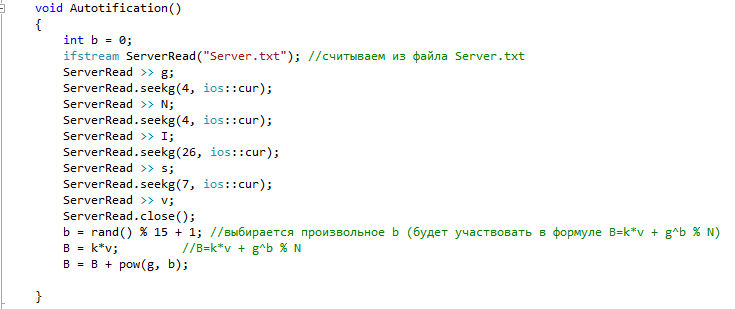


Сначала выполнится метод клиента Autotification



Вычисляется A по формуле A=g^a %N и затем передается серверу, убедившись что A!=0;

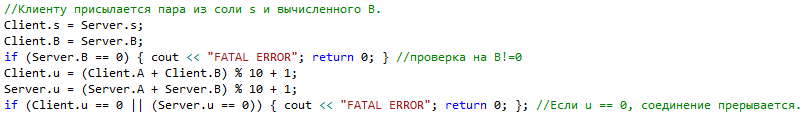
Затем запускается метод Autotification сервера



Считываются данные из файла Server.txt и создается число B по формуле B=k\*v+g^b %N;

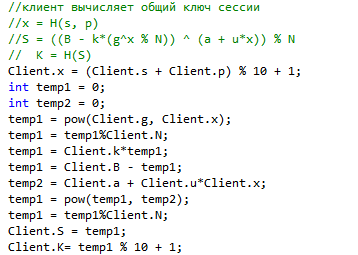
После этого клиенту присылается пара из соли и вычисленного B.

Убедившись что B!=0 обе стороны вычисляют скремблер u = H(A, B)и если они не равны то соединение прерывается

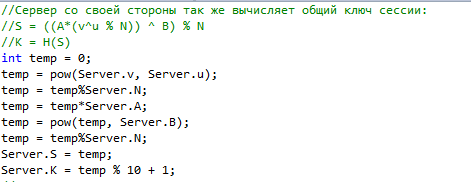


Далее клиент вводит свой пароль p, на основе которого вычисляется общий ключ сессии

x = H(s, p)  
S = ((B - k\*(g^x % N)) ^ (a + u\*x)) % N  
K = H(S)



Сервер со своей стороны так же вычисляет общий ключ сессии:  
S = ((A\*(v^u % N)) ^ B) % N  
K = H(S)

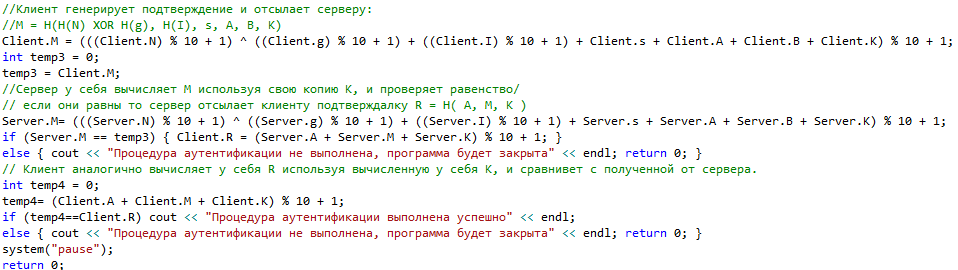


Начиная с этого момента, если клиент и сервер оба действительно знают друг друга, есть идентичное число K у обоих.

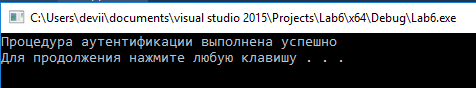
Клиент генерирует подтверждение:  
M = H( H(N) XOR H(g), H(I), s, A, B, K)  
и отсылает серверу.

Сервер у себя вычисляет M используя свою копию K, и проверяет равенство. Если они равны, значит у клиента точно такой же K, и, следовательно, клиент тот, за кого себя выдаёт. Если не равны — соединение прерывается, БЕЗ отсылки клиенту R.

Если же они равны, то чтобы доказать теперь клиенту, что он не верблюд, он отсылает ему в ответ свою подтверждалку:  
R = H( A, M, K )  
Клиент аналогично вычисляет у себя R используя вычисленную у себя K, и сравнивает с полученной от сервера.



Если всё прошло успешно, то программа выдаст сообщение об успешной аутентификации



**Листинг программы**

#include <iostream>

#include <fstream>

#include <time.h>

#include <stdlib.h>

#include <string>

#include <vector>

#include <ctime>

#include <map>

#include <math.h>

using namespace std;

class Client

{

public:

int q = 0; // простое

int N = 0; // "безопасное простое"

int g = 0; //g = генератор по модулю N

int k = 3; // параметр-множитель в SRP-6 это 3

int I = 0; // числовой идентификатор пользователя - login

int p = 0; //числовой пароль

int s = 0; // соль

int x = 0; // x=H(s,p)

int v = 0; // password verifier

int A = 0;

int B = 0;

int u = 0; //скремблер

int S = 0;

int K = 0;//общий ключ сессии

int a = 0;

int M = 0;

int R = 0; //подтверждение от сервера

void Registration() //регистрация клиента

{

int a = 0, b = 1, sch = 0, tg = 2, j = 2;

cout << "Введите простое число q" << endl;

cin >> q;

for (int n = 2; n <= q / 2; n++) //проверка, является ли введенное q простым числом

{

if (!(q % n)) { cout << "число q - не простое, удачи при в вводе в следующий раз" << endl; return; }

}

N = 2 \* q + 1; //"безопасное простое"

for (int n = 2; n <= N / 2; n++) //проверка, является ли вычесленное N простым числом

{

if (!(N % n)) { cout << "число N - не простое, выберите другое q, при котором N=2\*q+1 - простое" << endl; return; }

}

while (sch != 1 && tg<N) //находим g

{

while (a != b && j <= N) {

a = N%j;

b = tg%j;

j++;

}

if (a == b&&a == 0) {

tg++;

}

else {

sch = 1;

}

}

g = tg;

cout << "Введите числовой идентификатор пользователя - login" << endl;

cin >> I;

cout << "Введите числовой пароль" << endl;

cin >> p;

s= rand() % 15 + 1; //генерация соли

x=(p + s) % 10 + 1; //функция хэширования

v = pow(g, x); // v = g^x % N

v = v%N;

cout << "Пользователь создан" << endl;

}

void Autotification() //аутентификации клиента

{

int aa = 0;

int temp1, temp2 = 0;

aa = rand() % 15 + 1; //выбирается произвольное а (будет учавствовать в формуле A = g^a % N)

//получаем из файла Client.txt g и N

ifstream CLIENTread("Client.txt");

CLIENTread >> g;

CLIENTread.seekg(4, ios::cur);

CLIENTread >> N;

CLIENTread.seekg(4, ios::cur);

CLIENTread >> I;

CLIENTread.seekg(26, ios::cur);

CLIENTread >> s;

CLIENTread.close();

A = pow(g, aa); //A = g^a % N

A = A%N;

a = aa;

}

};

class Server

{

public:

int g = 0; //g = генератор по модулю N

int N = 0; // "безопасное простое"

int I = 0; // числовой идентификатор пользователя - login

int s = 0; // соль

int v = 0; // password verifier

int k = 3; // параметр-множитель в SRP-6 это 3

int A = 0;

int B = 0;

int u = 0; // скремблер

int S = 0;

int K = 0; //общий ключ сессии

int M = 0;

void Registration()

{

ofstream Serverwrite("Server.txt");

Serverwrite << g << " - g " << endl;

Serverwrite << N << " - N " << endl;

Serverwrite << I << " - ID пользователя (login)" << endl;

Serverwrite << s << " - Соль" << endl;

Serverwrite << v << " - password verifier" << endl;

Serverwrite.close();

cout << "Пользователь зарегистрирован на сервере" << endl;

}

void Autotification()

{

int b = 0;

ifstream ServerRead("Server.txt"); //считываем из файла Server.txt

ServerRead >> g;

ServerRead.seekg(4, ios::cur);

ServerRead >> N;

ServerRead.seekg(4, ios::cur);

ServerRead >> I;

ServerRead.seekg(26, ios::cur);

ServerRead >> s;

ServerRead.seekg(7, ios::cur);

ServerRead >> v;

ServerRead.close();

b = rand() % 15 + 1; //выбирается произвольное b (будет участвовать в формуле B=k\*v + g^b % N)

B = k\*v; //B=k\*v + g^b % N

B = B + pow(g, b);

}

};

int main(int args, const char \* argv[])

{

Client Client;

Server Server;

setlocale(LC\_ALL, "rus");

srand(time(0));

int id = 0;

rand();

cout << " [Меню] " << endl;

cout << "[1] Регистрация пользователя" << endl;

cout << "[2] Аутентификация пользователя" << endl;

cout << "Выберите пункт " << endl;

cin >> id;

if (id == 1) {

system("cls");

cout << "Вы в меню регистрации"<< endl;

Client.Registration(); //регистрация на стороне клиента

ofstream CLIENTwrite("Client.txt");

CLIENTwrite << Client.g << " - g " << endl;

CLIENTwrite << Client.N << " - N " << endl;

CLIENTwrite << Client.I << " - ID пользователя (login)" << endl;

CLIENTwrite << Client.s << " - Соль" << endl;

CLIENTwrite.close();

/// OTPRAVKA NA SERVER

Server.g = Client.g;

Server.N = Client.N;

Server.I = Client.I;

Server.s = Client.s;

Server.v = Client.v;

Server.Registration(); //регистрация на стороне сервера

system("pause");

return 0;

}

else if (id == 2) {

system("cls");

Client.Autotification();

Server.A = Client.A;

if (Client.A == 0) { cout << "FATAL ERROR"; return 0; } //проверка на А!=0

Server.A = Client.A;

Server.Autotification();

//Клиенту присылается пара из соли s и вычисленного B.

Client.s = Server.s;

Client.B = Server.B;

if (Server.B == 0) { cout << "FATAL ERROR"; return 0; } //проверка на B!=0

Client.u = (Client.A + Client.B) % 10 + 1;

Server.u = (Server.A + Server.B) % 10 + 1;

if (Client.u == 0 || (Server.u == 0)) { cout << "FATAL ERROR"; return 0; }; //Если u == 0, соединение прерывается.

//клиент вычисляет общий ключ сессии

//x = H(s, p)

//S = ((B - k\*(g^x % N)) ^ (a + u\*x)) % N

// K = H(S)

Client.x = (Client.s + Client.p) % 10 + 1;

int temp1 = 0;

int temp2 = 0;

temp1 = pow(Client.g, Client.x);

temp1 = temp1%Client.N;

temp1 = Client.k\*temp1;

temp1 = Client.B - temp1;

temp2 = Client.a + Client.u\*Client.x;

temp1 = pow(temp1, temp2);

temp1 = temp1%Client.N;

Client.S = temp1;

Client.K= temp1 % 10 + 1;

//Сервер со своей стороны так же вычисляет общий ключ сессии:

//S = ((A\*(v^u % N)) ^ B) % N

//K = H(S)

int temp = 0;

temp = pow(Server.v, Server.u);

temp = temp%Server.N;

temp = temp\*Server.A;

temp = pow(temp, Server.B);

temp = temp%Server.N;

Server.S = temp;

Server.K = temp % 10 + 1;

//Клиент генерирует подтверждение и отсылает серверу:

//M = H(H(N) XOR H(g), H(I), s, A, B, K)

Client.M = (((Client.N) % 10 + 1) ^ ((Client.g) % 10 + 1) + ((Client.I) % 10 + 1) + Client.s + Client.A + Client.B + Client.K) % 10 + 1;

int temp3 = 0;

temp3 = Client.M;

//Сервер у себя вычисляет M используя свою копию K, и проверяет равенство/

// если они равны то сервер отсылает клиенту подтверждалку R = H( A, M, K )

Server.M= (((Server.N) % 10 + 1) ^ ((Server.g) % 10 + 1) + ((Server.I) % 10 + 1) + Server.s + Server.A + Server.B + Server.K) % 10 + 1;

if (Server.M == temp3) { Client.R = (Server.A + Server.M + Server.K) % 10 + 1; }

else { cout << "Процедура аутентификации не выполнена, программа будет закрыта" << endl; return 0; }

// Клиент аналогично вычисляет у себя R используя вычисленную у себя K, и сравнивет с полученной от сервера.

int temp4 = 0;

temp4= (Client.A + Client.M + Client.K) % 10 + 1;

if (temp4==Client.R) cout << "Процедура аутентификации выполнена успешно" << endl;

else { cout << "Процедура аутентификации не выполнена, программа будет закрыта" << endl; return 0; }

system("pause");

return 0;

}

else { cout << "Программа будет закрыта" << endl;

system("pause");

return 0;

}

}