МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. ІГОРЯ СІКОРСЬКОГО”

#### «ОСНОВИ ПРОГРАМУВАННЯ – 3»

**Курсова робота на тему**

**«Розробка програми для шифрування даних методом Ель-Гамаля»**

Дата “\_\_\_”\_\_\_\_\_\_ Виконав: студент 1 курсу ТЕФ

Кафедра АПЕПС

Гр. ТВ-61

Миколюк Андрій Сергійович

Захищена з оцінкою:

Київ – 2017

ЗМІСТ

Вступ……………………………………………………………………...……..3

1 Теоретичні відомості 4

1.1 Генерація ключів 4

1.2 Шифрування 4

1.3 Дешифрування 4

2 Програмна реалізація алгоритму 6

2.1 Середовище розробки проекту 6

2.2 Мова програмування проекту 7

2.3 Опис функцій 7

3 Огляд результатів 14

Висновки 15

Список використаної літератури 16

Додаток1 17

Додаток2 19

ВСТУП

Схема Ель-Гамаля (Elgamal) - Криптосистема з відкритим ключем, заснована на труднощі обчислення дискретних логарифмів в кінцевому полі. Криптосистема включає в себе алгоритм шифрування і алгоритм цифрового підпису. Схема Ель-Гамаля лежить в основі колишніх стандартів електронного цифрового підпису в США (DSA) і Росії (ГОСТ Р 34.10-94).

Схема була запропонована Тахера Ель-Гамаль в 1985 у. Ель-Гамаль розробив один з варіантів алгоритму Діффі-Хеллмана. Він удосконалив систему Діффі-Хеллмана і отримав два алгоритми, які використовувалися для шифрування і для забезпечення автентифікації. На відміну від RSA алгоритм Ель-Гамаля не запатентований і, тому, став більш дешевою альтернативою, оскільки не була потрібна оплата внесків за ліцензію. Вважається, що алгоритм потрапляє під дію патенту Діффі-Хеллмана.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Схема Ель-Гамаля містить три основні етапи:

* Генерація ключів
* Шифрування
* Дешифрування

Розглянемо їх по порядку.

**Генерація ключів**

1. Генерується випадкове просте число {\displaystyle ~p}**p**.
2. Вибирається випадковий первісний корінь {\displaystyle ~g}**g** поля **Zp** {\displaystyle \mathbb {Z} \_{p}}.
3. Вибирається випадкове ціле число {\displaystyle ~x}**x** таке, що {\displaystyle ~1<x<p-1}**1< x < p-1**.
4. Обчислюється {\displaystyle ~y=g^{x}\,{\bmod {\,}}p}**y = gx mod p**.
5. Відкритим ключем є трійка {\displaystyle \left(p,g,y\right)}**(p,g,y)**, закритим ключем — число {\displaystyle ~x}**x**.

**Шифрування**

Повідомлення   **M** шифрується таким чином:

1.Вибирається сесійний ключ - випадкове ціле число   **k** таке, що   **1 < k < p – 1**.

2.Обчислюються числа **a = gk mod p** і **b = yk mod p**.

3.Пара чисел ( a , b ) є шифротекстом.

Неважко бачити, що довжина шифротекста у схемі Ель-Гамаля довше вихідного повідомлення **M** вдвічі.

**Розшифрування**

Знаючи закритий ключ   **x** , вихідне повідомлення можна обчислити з шифротекста **( a , b)** за формулою:

**M = b (ax)−1 mod p.**

При цьому неважко перевірити, що

**(ax)−1 ≡ g−kx (mod p)**

і тому

**b (ax)−1 ≡ (ykM)g−xk ≡ (gxkM)g−xk ≡ M (mod p)**

Для практичних обчислень більше підходить наступна формула:

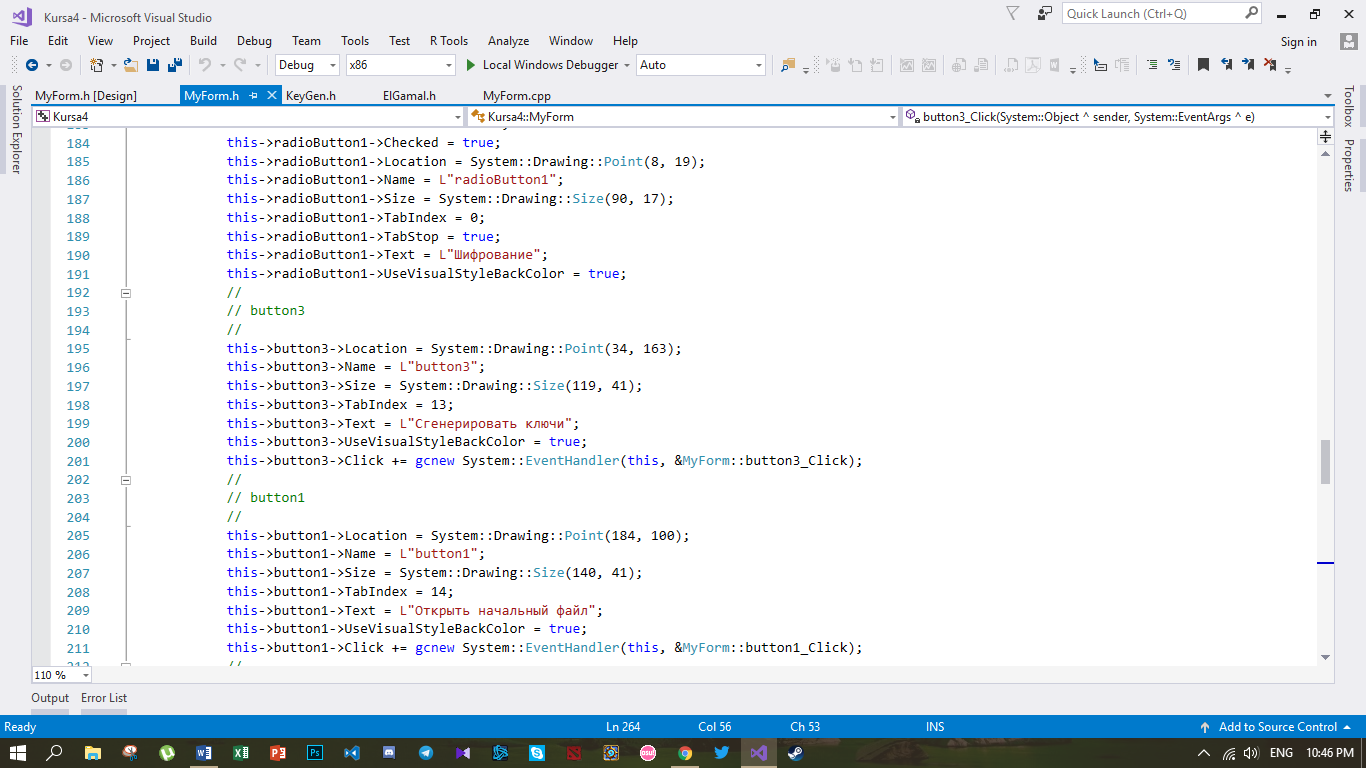
**M = b(ax)-1 mod p = b\*a(p-1-x) mod p.**

Так як в схему Ель-Гамаля вводиться випадкова величина k, То шифр Ель-Гамаля можна назвати шифром багатозначною заміни. Через випадковості вибору числа k таку схему ще називають схемою імовірнісного шифрування. Імовірнісний характер шифрування є перевагою для схеми Ель-Гамаля, так як у схем імовірнісного шифрування спостерігається велика стійкість у порівнянні зі схемами з певним процесом шифрування. Недоліком схеми шифрування Ель-Гамаля є подвоєння довжини зашифрованого тексту в порівнянні з початковим текстом. Для схеми імовірнісного шифрування саме повідомлення Mі ключ не визначають шифротекст однозначно. У схемі Ель-Гамаля необхідно використовувати різні значення випадкової величини k для шифровки різних повідомлень **M**i**Mꞌ** . Якщо використовувати однакові k, то для відповідних шифротекстів **(a, b)** і **(aꞌ, bꞌ)** виконується співвідношення **b(bꞌ)-1 = M(Mꞌ)-1**. З цього виразу можна легко обчислити **Mꞌ**, якщо відомо **M**.

ПРОГРАМНА РЕАЛІЗАЦІЯ АЛГОРИТМУ

**Середовище розробки проекту**

Розробка програмного забезпечення виконувалась у середовищі Microsoft Visual Studio 2017 (рис.1 ). Visual Studio – це лінійка продуктів компанії Microsoft, що включають інтегроване середовище, розробки програмного забезпечення та ряду інших інструментальних засобів. Цей продукт дозволяє створювати як консольні програми, так і програми з графічним інтерфейсом, у тому числі технології з підтримкою Windows Forms, а також веб-сайти, веб-додатки, веб-служби для всіх платформ, які підтримують Microsoft Windows, Windows Mobile, Windows CE, .NET Framework, Microsoft Silverlight.

Рис.1

**Мова програмування проекту**

Програма бла розроблена на мові програмування С++ з використанням функцій середовища програмування .NET для побудови графічного інтерфейсу програми для операційної системи Windows.

C++ — мова програмування високого рівня з підтримкою декількох парадигм програмування: об'єктно-орієнтованої, узагальненої та процедурної. Розроблена Б'ярном Страуструпом в AT&T Bell Laboratories (Мюррей-Хілл, Нью-Джерсі) 1979 року та початково отримала назву «Сі з класами». Згодом Страуструп перейменував мову на C++ у 1983 р. Базується на мові С. Вперше описана стандартом ISO/IEC 14882:1998, найбільш актуальним же є стандарт ISO/IEC 14882:2014.

С++/СLI – специфікація мови програмування С++ для середовища програмування .NET фірми Microsoft. Вона інтегрує С++ стандарту ISO з Об’єднаною системою типів (Unified Type System, UTS), що розглядається як частина Загальної мовної інфраструктури (Common Language Infrastructure, CLI). Вона підтримує ф сирцевий рівень, і функціональну сумісність виконуваних файлів, скомпільованих із рідного і керованого C++. C++/CLI стандартизований в ECMA як ECMA-372.

**Функціональні можливості проекту.**

Програмне забезпечення повинно реалізовувати такі ключові можливості:

* Зчитування файлу;
* Шифрування файлу схемою Ель-Гамаля;
* Розшифрування файлу;
* Запис файлу;
* Генерація ключів.

Проаналізувати завдання, можна виділити три основні напрямки розробки, а саме:

* Розробка графічного інтерфейсу;
* Реалізація криптографічної системи;
* Генерація ключів для криптографічної системи.

На основі цих досліджень програмний код був поділений на три логічні частини, які були розбиті по різним файлам: «MyForm.h», «ElGamal.h», «KeyGen.h» відповідно. Давайте розглянемо реалізацію кожного з них.

Для початку розглянемо програмний код, розроблений в файлу «MyForm.h», так як саме він делегує виконання криптографічних робіт з файлами і генерацію ключів.

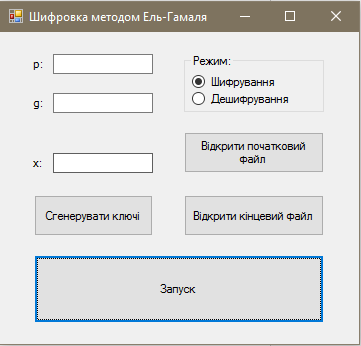
На рис. 2 зображено вигляд графічного інтерфейсу при запуску програми. 

Рис. 2

При натисненні на кнопок “Відкрити початковий файл” та “Відкрити кінцевий файл” виникає діалогове вікно вибору текстового файлу. При натисненні на кнопку “Сгенерувати ключі” програма автоматично генерує ключі.

Після вибору початкового та кінцевого файлів можна починати роботу. При натисненні на кнопку “Запуск” викликається функція шифрування або дешифрування, в залежності від вибраного пункту в правому верхньому куті програми.

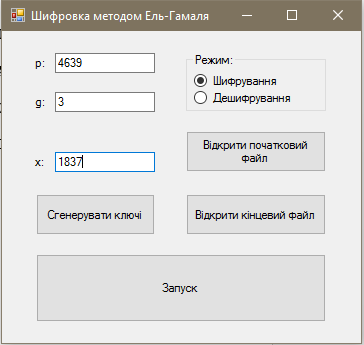
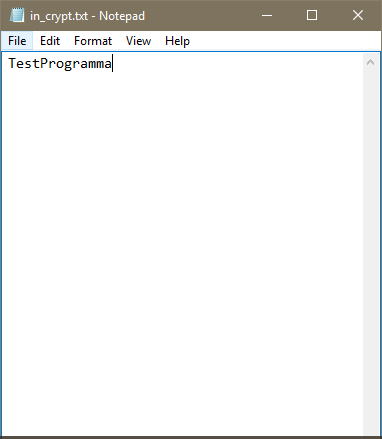
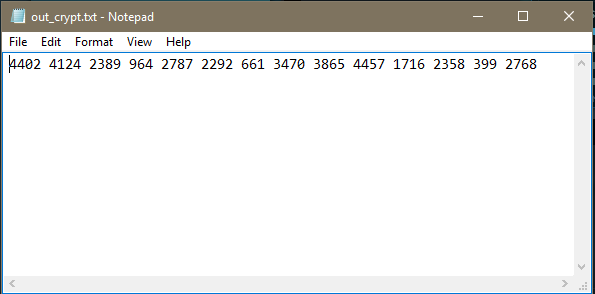
Для успішного шифрування/дешифрування текстового файлу також повинні бути введені ключі **(p, g)** та **х**. Їх можна як ввести самому, так і згенерувати випадкові ключі автоматично. Приклад роботи програми зображений на рис. 3, рис. 4 та рис. 5. 

Рис.3

 Рис.4

 Рис.5

Розглянемо детальніше проблему, яка виникла при реалізації з’єднанні текстового файлу з програмою шифрування/дешифрування.

Діалог вибору файлу входить в стандартну реалізацію Windows Forms і представлений класом System.Windows.Forms.OpenFileDialog.

Windows Forms – інтерфейс програмування додатків (АРІ), що відповідає за графічний інтерфейс користувача і є частиною Microsoft .NET Framework. Даний інтерфейс спрощує доступ до елементів інтерфейсу Microsoft Windows за рахунок створення обгортки для вже існуючого Win32 API в керованому коді. Причому керований код – класи, що реалізують АРІ для Windows Forms, що не залежать від мови розробки. Тобто програміст однаково може використовувати Windows Forms як про написанні ПЗ на С #, C ++, так і на VB.Net, J # і ін.

Усередині .NET Framework, Windows Forms реалізується в рамках простору імен System.Windows.Forms.

Суть виникнувшої проблеми в тому, шо інформацію про шлях до вибраного файлу клас System.Windows.Forms.OpenFileDialog зберігає в властивості FileName, записаній в структуру даних System::String^, в той час як для запуску файлового потоку засобами С ++ інформацію необхідно конвертувати в std::string. Спеціально для цих цілей Microsoft випустили бібліотеку підтримки msclr\marshal\_cppstd.h. Реалізація маршалінгу виглядає наступним чином:

#include <msclr\marshal\_cppstd.h>

//

// . . .

//

String^ FilePath1 = openFileDialog1->FileName;

filepath1 = msclr::interop::marshal\_as< string >(FilePath1);

Давайте тепер розглянемо реалізацію генерування ключів для шифрування / дешифрування схемою Ель-Гамаля.

Даний процес реалізований класом KeyGen, код якого представлений в файлі KeyGen.h (див. Додаток 2).

Генерацію простого числа **р** реалізовують функції KeyGen::primeGen та KeyGen::isPrime. Основна складність реалізації генератора заключається в генерації первісного кореня **g** для простого числа **р**.

Давайте розглянемо теоретичні відомості.

Первісний корінь за модулем m – ціле число g таке, що



та



де ― [функція Ейлера](https://uk.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D1%96%D1%8F_%D0%95%D0%B9%D0%BB%D0%B5%D1%80%D0%B0). Іншими словами, первісний корінь — це [породжуючий елемент](https://uk.wikipedia.org/wiki/%D0%9F%D0%BE%D1%80%D0%BE%D0%B4%D0%B6%D1%83%D1%8E%D1%87%D0%B0_%D0%BC%D0%BD%D0%BE%D0%B6%D0%B8%D0%BD%D0%B0_%D0%B3%D1%80%D1%83%D0%BF%D0%B8) мультиплікативної [групи](https://uk.wikipedia.org/wiki/%D0%93%D1%80%D1%83%D0%BF%D0%B0_(%D0%B0%D0%BB%D0%B3%D0%B5%D0%B1%D1%80%D0%B0)) [кільця лишків](https://uk.wikipedia.org/wiki/%D0%9A%D1%96%D0%BB%D1%8C%D1%86%D0%B5_%D0%BB%D0%B8%D1%88%D0%BA%D1%96%D0%B2) за модулем **m**. Для первісного кореня **g** його степені непорівнювані між собою за модулем  **m** і породжують приведену систему лишків за модулем **m**.

Тому для кожного числа **a** [взаємно простого](https://uk.wikipedia.org/wiki/%D0%92%D0%B7%D0%B0%D1%94%D0%BC%D0%BD%D0%BE_%D0%BF%D1%80%D0%BE%D1%81%D1%82%D1%96_%D1%87%D0%B8%D1%81%D0%BB%D0%B0) з **m** знайдеться показник такий, що 

Таке число **l** називається індексом числа  **a** за основою **g**. Первісні корені існують не для всіх модулів, а тільки для модулів **m** виду 

де  ― [просте число](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%81%D1%82%D0%B5_%D1%87%D0%B8%D1%81%D0%BB%D0%BE). Тільки в цих випадках мультиплікативна група кільця лишків за модулем **m** є [циклічною групою](https://uk.wikipedia.org/wiki/%D0%A6%D0%B8%D0%BA%D0%BB%D1%96%D1%87%D0%BD%D0%B0_%D0%B3%D1%80%D1%83%D0%BF%D0%B0) порядку .

З теореми Лагранжа слідує, що показник будь-якого числа по модулю m є дільником . Таким чином, досить перевірити, що для всіх власних дільників d виконується .

Таким чином, алгоритм знаходження первісної кореня такої. Знаходимо , факторизуємо її. Тепер перебираємо всі числа **g = 1..n**, і для кожного рахуємо всі велечини . Якщо для поточного g всі ці числа виявилися відмінними від 1, то це g і є шуканим первісним контролем.

Реалізація алгоритму представлена функціями KeyGen::PrimeRootGen та KeyGen::Pow.

Ну і нарешті розглянемо реалізацію шифрування та дешифрування текстового файлу схемою Ель-Гамаля, представлений в файлі «ElGamal.h»

(див. Додаток 1).

Функція crypt отримує в якості параметрів ключі р, х, а також строкові велечини inFileName та outFileName з інформацією про вихідний файл та зашифрований файл відповідно.

Вихідний файл перебирається посимвольно, символ char конвертується в число int, шифрується в шифротекст з двійки чисел a i b, і вони записуються в зашифрований файл.

Аналогічно працює функція decrypt, тільки в зворотному порядку. Єдине що, в якості параметрів крім інформації про файли функція отримує тільки ключі р та х, так як для дешифрування первісний корінь g не потрібен.

Допоміжними функціями для обчислень реалізовані функції Power та Multiply.

ОГЛЯД РЕЗУЛЬТАТІВ

Результатом виконання курсової роботи став програмний продукт, який реалізовує шифрування та розшифрування текстового файлу за криптографічною схемою Ель-Гамаля з відкритим ключем. Програмний продукт був реалізований під операційну систему Windows за допомогою середовища розробки Visual Studio 2017 Community Edition. Мова програмування, яка використовувалась при створенні програми: С ++. Також для побудови графічного інтерфейсу використовувались функції платформи .NET.

Список використаної літератури

1. Керніган Б. Мова C / Б. Керніган Д. Річі . – М.:Вільямс ,2009 − 292с.

**2. Рябко Б. Я., Фионов А. Н. Криптографические методы защиты информации: Учебное пособие для вузов. — М.: Горячая линия-Телеком, 2005. — 229 с.: ил**.

3. Основи програмування та алгоритмічні мови – 2. Спеціальні засоби мови програмування. Методичні вказівки до виконання комп’ютерних практикумів (частини 1, 2) для студентів напрямку підготовки 6.050103 – «Програмна інженерія» денної форми навчання/Укладачі: Крячок О.С., Кузьменко І.М., Гурін А.Л., Круш О.Є. – К.: НТУУ «КПІ», 2012.

4. [http://cppstudio.com/uk/post/439/](http://cppstudio.com/uk/post/439/%20) -робота з класами.

5. <https://www.youtube.com/> - робота з Microsoft Forms.

8. <http://replace.org.ua/forum/6/> - теоретичні відомості.

9. <https://ru.scribd.com/doc/18748945/> - теоретичні відомості.

10. <http://cppstudio.com/post/7216/> - робота з масивами.

ВИСНОВКИ

В даний час криптосистеми з відкритим ключем вважаються найбільш перспективними. До них належить і схема Ель-Гамаля, криптостійкість якої заснована на обчислювальній складності проблеми дискретного логарифмування, де за відомими p, g та у потрібно обчислити х, що задовольняє рівнянню:



ГОСТ У34.10-1994, прийнятий в Україні, що регламентував процедури формування та перевірки електронного цифрового підпису, був заснований на схемі Ель-Гамаля. С 2001р. використовується новий ГОСТ У34.10-2001, що використовує арифметику еліптичних кривих, визначених над простими полями Галуа. Існує велика кількість алгоритмів, заснованих на схемі Ель-Гамаля: це алгоритми DSA, ECDSA, KCDSA, схема Шнорра.

**Додаток 1**

#pragma once

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <fstream>

#include <cstdlib>

#include <ctime>

#include <string>

using namespace std;

int Power(int a, int b, int n) // Функция Power вычисляет и возвращает a^b mod n

{

int tmp = a;

int sum = tmp;

for (int i = 1; i < b; i++)

{

for (int j = 1; j < a; j++)

{

sum += tmp;

if (sum >= n)

{

sum -= n;

}

}

tmp = sum;

}

return tmp;

}

int Multiply(int a, int b, int n) // Функция Multiply вычисляет и возвращает a\*b mod n

{

int sum = 0;

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

{

sum += a;

if (sum >= n)

{

sum -= n;

}

}

return sum;

}

void crypt(int p, int g, int x, string inFileName, string outFileName)

{

ifstream inf(inFileName.c\_str());

ofstream outf(outFileName.c\_str());

int y = Power(g, x, p);

while (inf.good())

{

int m = inf.get();

if (m > 0)

{

int k = rand() % (p - 2) + 1;

int a = Power(g, k, p);

int b = Multiply(Power(y, k, p), m, p);

outf << a << " " << b << " ";

}

}

inf.close();

outf.close();

}

void decrypt(int p, int x, string inFileName, string outFileName)

{

ifstream inf(inFileName.c\_str());

ofstream outf(outFileName.c\_str());

while (inf.good())

{

int a = 0;

int b = 0;

inf >> a;

inf >> b;

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

{

int Num = Multiply(b, Power(a, p - 1 - x, p), p);

char m = static\_cast<char>(Num);

outf << m;

}

}

inf.close();

outf.close();

}

Додаток 2

#pragma once

#include <iostream>

#include <vector>

#include <cmath>

using namespace std;

class KeyGen

{

int p;

int g;

int y;

int x;

bool isPrime(int);

int PrimeGen();

int Pow(int, int, int);

int PrimitiveRootGen(int);

public:

KeyGen();

~KeyGen() {};

void run();

int get\_p();

int get\_g();

int get\_x();

};

KeyGen::KeyGen() {};

bool KeyGen::isPrime(int number)

{

int i;

int wall = (int)sqrt((double)number);

for (i = 2; i <= wall; i++)

{

if (number % i == 0) return false;

}

return true;

};

int KeyGen::PrimeGen()

{

while (true)

{

int r = rand();

if (isPrime(r)) return r;

}

}

int KeyGen::Pow(int a, int b, int n)

{

long long x = 1, y = a;

while (b > 0)

{

if (b % 2 == 1)

{

x = ( x \* y ) % n;

}

y = ( y \* y ) % n;

b /= 2;

}

return x % n;

}

int KeyGen::PrimitiveRootGen(int p)

{

vector<int> fact;

int phi = p - 1, n = phi;

for (int i = 2; i\*i <= n; ++i)

if (n % i == 0)

{

fact.push\_back(i);

while (n % i == 0)

n /= i;

}

if (n > 1)

fact.push\_back(n);

for (int res = 2; res < p; ++res)

{

bool ok = true;

for (size\_t i = 0; i < fact.size() && ok; ++i)

ok &= Pow(res, phi / fact[i], p) != 1;

if (ok)

return res;

}

return -1;

}

void KeyGen::run()

{

g = 0;

while (true)

{

p = PrimeGen();

g = PrimitiveRootGen(p);

if ((p > 300) && (g > 0))

break;

}

x = rand() % (p - 2) + 1;

}

int KeyGen::get\_p() { return p; }

int KeyGen::get\_g() { return g; }

int KeyGen::get\_x() { return x; }