**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**Институт информационных технологий и управления**

**Кафедра компьютерных систем и программных технологий**

**ОТЧЕТ**

**по курсовой работе на тему**

**«QT. GUI-приложение. Шифрование данных с использованием алгоритма RSA»**

**Дисциплина "Программирование"**

**Студент гр.13501/4 Ларенкова А.В.**

**Преподаватель Глухих М.И.**

**Санкт-Петербург**

**2015**

**Оглавление**

Описание задачи……………………………………………………………….……..2

Текст и описание программы………………………………………………….…….3

Испытание программы(тестирование)………………………………………….….6

Заключение…………………………………………………………………….……..8

**Описание задачи**

Разработать программу, позволяющую шифровать и дешифровать тексты по алгоритму RSA. Для начала необходимо сгенерировать открытый и секретный ключи:

Возьмем два больших простых числа p и q.

Определим n, как результат умножения p на q (n= p\*q).

Выберем случайное число, которое назовем d. Это число должно быть взаимно простым (не иметь ни одного общего делителя, кроме 1) с результатом умножения (p-1)\*(q-1).

Определим такое число е, для которого является истинным следующее соотношение (e\*d) mod ((p-1)\*(q-1))=1.

Hазовем открытым ключем числа e и n, а секретным - d и n.

Для того, чтобы зашифровать данные по открытому ключу {e,n}, необходимо следующее:

разбить шифруемый текст на блоки, каждый из которых может быть представлен в виде числа M(i)=0,1,2..., n-1( т.е. только до n-1).

зашифровать текст, рассматриваемый как последовательность чисел M(i) по формуле C(i)=(M(I)^e)mod n.

Чтобы расшифровать эти данные, используя секретный ключ {d,n}, необходимо выполнить следующие вычисления: M(i) = (C(i)^d) mod n. В результате будет получено множество чисел M(i), которые представляют собой исходный текст.

**Текст и описание программы**

**RSA.h:**

#pragma once

#include<vector>

#include<iostream>

#include <QString>

using namespace std;

typedef vector<unsigned char> data\_type;

class BigInteger

{

public:

data\_type Value;

BigInteger();

~BigInteger();

BigInteger(const BigInteger&);

BigInteger(int);

//переводит BigInteger в string

friend QString BigIntegerToStr(BigInteger number);

BigInteger operator=(const BigInteger&);

//переводит string в BigInteger

BigInteger(std::string &x);

int toInt();

//делаем перегрузку оператора << для вывода BigInteger

friend ostream& operator<<(ostream&, const BigInteger&);

//логические операторы

friend bool operator==(const BigInteger&, const BigInteger&);

friend bool operator!=(const BigInteger&, const BigInteger&);

friend bool operator>(const BigInteger&, const BigInteger&);

friend bool operator<(const BigInteger&, const BigInteger&);

//математические операторы

friend BigInteger operator+(const BigInteger&, const BigInteger&);

friend BigInteger operator-(const BigInteger&, const BigInteger&);

friend BigInteger operator\*(const BigInteger&, const BigInteger&);

friend BigInteger operator/(const BigInteger&, const BigInteger&);

friend BigInteger operator%(const BigInteger&, const BigInteger&);

//(a\*b) % mod умножение по модулю

friend BigInteger modul\_mult(BigInteger&, BigInteger&, BigInteger&);

//(a^b) % mod возведение в степень по модулю

friend BigInteger fast\_power(BigInteger&, BigInteger&, BigInteger&);

protected:

bool Sign;

};

const BigInteger zero(0);

const BigInteger one(1);

const BigInteger two(2);

const int LENGTH(8);

const int PROBABILITY(5); // вероятность

class RSA

{

public:

RSA();

~RSA();

//шифрование (e и n берутся из программы)

void encryption(BigInteger&, BigInteger&);

//шифрование (e и n вводятся пользователем в соотвествующие окна)

void encryption(BigInteger &, BigInteger &,BigInteger & exp,BigInteger & mod);

//расшифровка

void decryption(BigInteger&, BigInteger&);

//публичный ключ

void publish\_keys(BigInteger&, BigInteger&);

private:

//алгоритм Евклида

BigInteger euclid(const BigInteger&, const BigInteger&);

//генерирует случайные (любые) числа

void Randomiser(BigInteger &, int);

//генерирует случайное число меньшее сгенерированных случайно ранее(для теста Рабина-Миллера)

void RangeRandomiser(BigInteger &, const BigInteger &);

//тест Рабина-Миллера

bool IsPrime(BigInteger &);

//генерирует случайные числа нужной длинны, потом проверяет, является ли число простым

void PrimeGenerator(BigInteger &);

BigInteger e; // Открытая экспонента

BigInteger d; // Секртеная экспонента

BigInteger n; // p\*q, где p и q простые числа (BigInteger)

};

**mainwindow.h:**

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include "RSA.h"

namespace Ui {

class MainWindow;

}

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit MainWindow(QWidget \*parent = 0);

QString s;

~*MainWindow*();

private slots:

//считывает введенное сообщение

void on\_actionRead\_triggered();

//зашифровывает сообщение

void on\_actionEncode\_triggered();

//расшифровывает сообщение

void on\_actionDecode\_triggered();

//кнопка выхода

void on\_actionQuit\_triggered();

//кнопка "зашифровать", используется при передаче сообщений между двумя компьютерами,

//позволяет вводить e и n вручную

void on\_radioButton\_toggled(bool checked);

//кнопка "расшифровать", используется при передаче сообщений между двумя компьютерами,

//генерирует e и n

void on\_radioButton\_2\_toggled(bool checked);

private:

Ui::MainWindow \*ui;

RSA tryhard;

BigInteger crypt, e, n;

bool flag;

};

#endif

Класс BigInteger создан для работы с большими числами, длина которых не помещается в стандартные типы данных. Поэтому необходимо отдельно прописать функции, которые выполняют математические операции с «длинными» числами.

Класс RSA необходим для непосредственного шифрования, дешифрации, генерации ключей и проверки сгенерированных чисел на простоту.

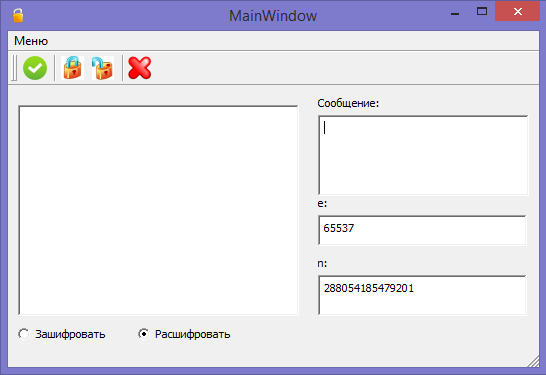
При создании графического интерфейса пользователя были созданы 4 основные кнопки: считывание введенного сообщения, шифрование, дешифровка и выход. Для обмена зашифрованной информацией между двумя компьютерами были созданы радиокнопки.

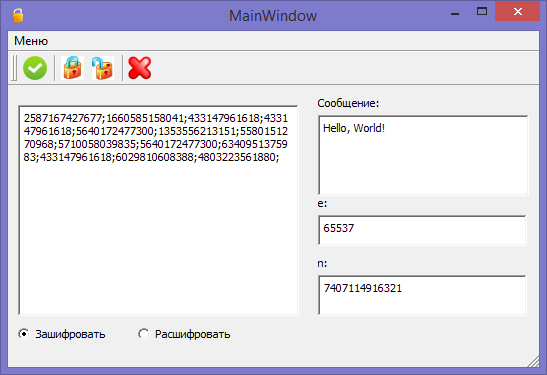
Схема передачи сообщения между двумя компьютерами:

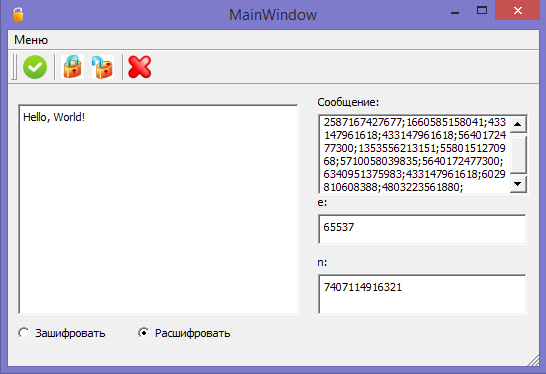
1. Пользователь1 выбирает опцию «Расшифровать», генерируется публичный ключ, который передается Пользователю2.
2. Пользователь2 выбирает опцию «Зашифровать», вводит с клавиатуры значения e и n (публичного ключа), затем вводит сообщение в соответствующее окно и шифрует сообщение.
3. Пользователь2 передает зашифрованное сообщение Пользователю1.
4. Пользователь1 копирует зашифрованное сообщение в окно «Сообщение» и нажимает кнопку расшифровать.

Пользователь1 не должен закрывать программу во время выполнения этой схемы, т.к. закрытый ключ генерируется при каждом открытии программы, в противном случае, он просто не сможет расшифровать сообщение от Пользователя2.

**Тестирование**



****



**Заключение**

Несмотря на то, что программа является целостным проектом, она имеет перспективы для дальнейшего развития. Например: увеличение скорости работы программы путем перехода к битовым операциям, создание сервера для обмена открытыми ключами, разработка удобного пользовательского интерфейса.