Санкт-Петербургский Политехнический Университет

Петра Великого

Институт информационных технологий и управления

Кафедра компьютерных сетей и программных технологий

**ОТЧЁТ**

ПО КУРСОВОЙ РАБОТЕ

«GUI приложение. Шифрование данных

с использованием алгоритма RSA»

Дисциплина «Программирование»

Студент Данилов А.Д.

Гр. 13501/4

Преподаватель Глухих М.И.

Санкт-Петербург

2015

Оглавление

Описание задачи………………………………………………………2

Описание алгоритма.…………………………………………………3

Исходный код….………………………………………………………4

Тесты.…………………………………………………………………8

**Описание задачи**

Написать приложение, выполняющее шифрование чисел, используя алгоритм RSA и поддерживающее ключи длиной 2048 бит.

**Описание алгоритма**

**Алгоритм создания открытого и секретного ключей**

RSA-ключи генерируются следующим образом:

1 Выбираются два различных случайных простых числа  и  заданного размера.

2 Вычисляется их произведение , которое называется *модулем*.

3 Вычисляется значение функции Эйлера от числа : 

4 Выбирается целое число  (), взаимно простое со значением функции .

• Число  называется *открытой экспонентой.*

5 Вычисляется число , мультипликативно обратное к числу  по модулю , то есть число, удовлетворяющее сравнению: 

• Число  называется *секретной экспонентой*.

6 Пара  публикуется в качестве *открытого ключа RSA*

7 Пара  играет роль *закрытого ключа RSA* и держится в секрете.

**Шифрование и расшифрование**

Предположим, Боб хочет послать Алисе сообщение .

Сообщениями являются целые числа в интервале от  до , т.е .

**Алгоритм шифрования**:

• Взять *открытый ключ*  Алисы

• Взять *открытый текст* 

• Зашифровать сообщение с использованием открытого ключа Алисы: 

**Алгоритм расшифрования**:

• Принять зашифрованное сообщение 

• Взять свой *закрытый ключ* 

• Применить закрытый ключ для расшифрования сообщения: 

**Исходный код**

1. bigint.h работа с большими целыми числами неограниченной длины

#pragma once

#include <vector>

#include <iterator>

#include <string>

#include <iostream>

#include <QString>

const unsigned int BASE = 10000;

const unsigned int RAZRYAD = 4;

class BigInt {

public:

std::vector< int > data;

void mul10();

void div10();

BigInt();

~BigInt();

BigInt(unsigned long x);

BigInt(const BigInt &x);

BigInt(std::string x);

friend std::ostream &operator << (std::ostream &os, const BigInt &num);

friend std::istream &operator >> (std::istream &is, BigInt &num);

friend BigInt operator + (const BigInt &a, const BigInt &b);

friend BigInt operator - (const BigInt &a, const BigInt &b);

friend BigInt operator \* (const BigInt &a, const BigInt &b);

friend BigInt operator / (BigInt a, BigInt b);

friend BigInt operator % (BigInt a, BigInt b);

friend QString biginttostr(BigInt number);

int toint();

BigInt& operator=(const BigInt &x);

friend BigInt power(const BigInt &a, const BigInt &b);

friend BigInt fast\_power(BigInt &num, BigInt &power, BigInt &modulus);

friend BigInt modul\_mult(BigInt &a, BigInt &b, BigInt &modul);

friend int cmp(const BigInt &a, const BigInt &b);

friend bool operator < (const BigInt &a, const BigInt &b);

friend bool operator > (const BigInt &a, const BigInt &b);

friend bool operator <= (const BigInt &a, const BigInt &b);

friend bool operator >= (const BigInt &a, const BigInt &b);

friend bool operator == (const BigInt &a, const BigInt &b);

friend bool operator != (const BigInt &a, const BigInt &b);

};

const BigInt one(1);

const BigInt two(2);

const BigInt zero(0);

class SBigInt

{

public:

SBigInt();

~SBigInt();

SBigInt(const BigInt &x);

SBigInt(const SBigInt &x);

friend std::ostream &operator << (std::ostream &os, const SBigInt &num);

friend SBigInt operator + (const SBigInt &a, const SBigInt &b);

friend SBigInt operator - (const SBigInt &a, const SBigInt &b);

friend SBigInt operator \* (const SBigInt &a, const SBigInt &b);

friend SBigInt operator / (SBigInt a, SBigInt b);

friend SBigInt operator % (SBigInt a, SBigInt b);

SBigInt& operator=(const SBigInt &x);

BigInt data;

bool sign;

};

2) Keys.h генерирование ключей

#ifndef KEYS\_H

#define KEYS\_H

#include "bigint.h"

#include<algorithm>

#include <string>

#include <time.h>

const unsigned int LENGTH = 12;

const unsigned int PRIME\_ACCURACY = 5; //Rabin-Miller test accuracy

class Keys

{

public:

Keys();

~Keys();

BigInt n, e, d;

private:

void Randomiser(BigInt &num, int Didgits);

void PrimeGenerator(BigInt &num);

bool IsPrime(BigInt &num);

void RangeRandomiser(BigInt &num, const BigInt &top);

BigInt euclid(const BigInt &e,const BigInt &modul);

};

#endif // KEYS\_H

3) rsa.h шифрование и расшифрованние данных

#ifndef RSA\_H

#define RSA\_H

#include"bigint.h"

#include"keys.h"

class RSA

{

public:

RSA();

~RSA();

void encrypt(BigInt& cryptmes, BigInt& message, BigInt& e, BigInt& n);

void encrypt(BigInt& cryptmes, BigInt& message);

void decrypt(BigInt& cryptmes, BigInt& message);

void PublicKey(BigInt &exp, BigInt& modulus);

private:

Keys dog;

};

#endif // RSA\_H

4) mainwindows.h главное окно приложения

#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);

~MainWindow();

private slots:

void on\_pushButton\_3\_clicked();

void on\_pushButton\_2\_clicked();

void on\_pushButton\_4\_clicked();

void on\_pushButton\_clicked();

void on\_pushButton\_5\_clicked();

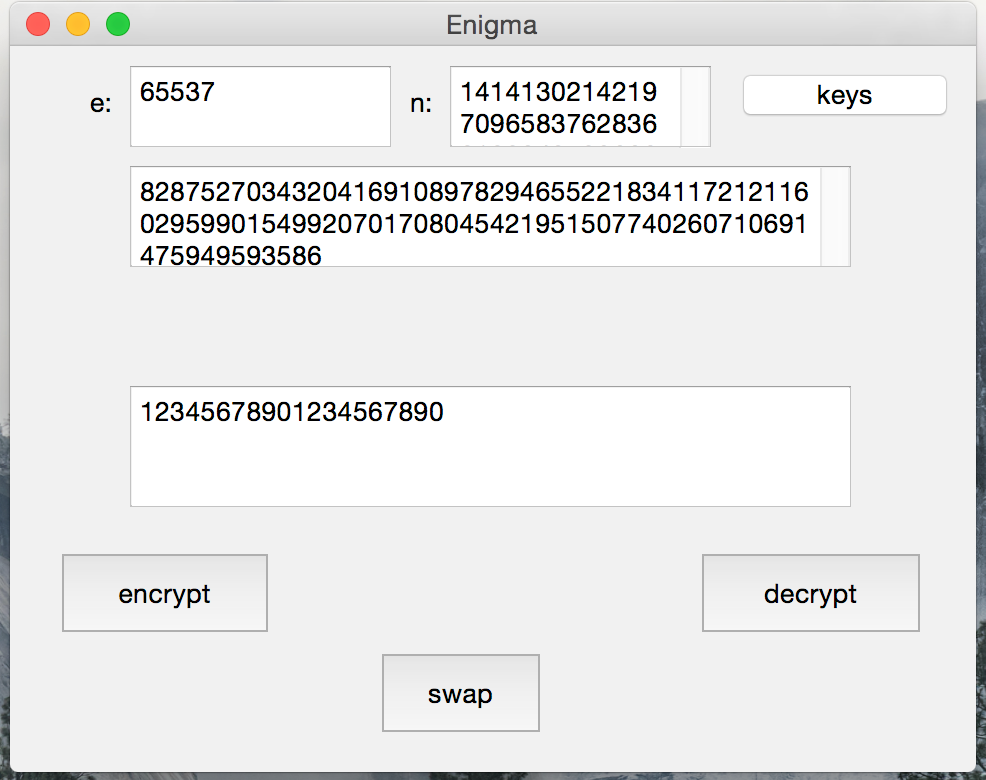
private:

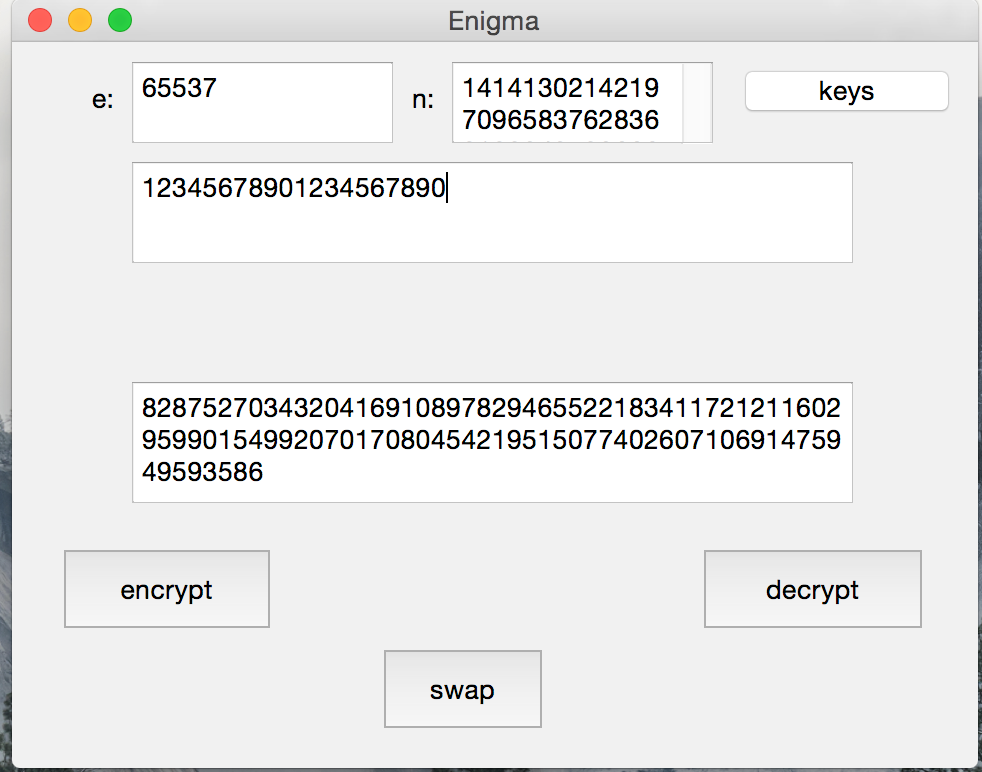
Ui::MainWindow \*ui;

RSA\* worker;

};

#endif // MAINWINDOW\_H

**Тест**

****

Содержание входного сообщения: *12345678901234567890*

Содержание зашифрованного файла: 8287527034320416910897829465522183411721211602959901549920701708045421951507740260710691475949593586

Содержание расшифрованного файла: *12345678901234567890*

*Длина ключа около 192 бит.*