МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Лабораторная работа №2

по дисциплине

«Информатика и программирование»

Классы и шифры

Группа: **АБс-822**

Студент: **Соленый Н.Е.**

Преподаватель:

НОВОСИБИРСК

2019

**Задача:**

Реализовать интерфейс для теста алгоритмов шифрования на выбор пользователя. Реализовать и сами алгоритмы на C++, применив ООП. Для секретной информации использовать атрибуты private. Задачу выполнить для следующих алгоритмов: RSA, шифр Шамира, шифр Эль-Гамаля, система Диффи-Хеллмана.

UPD: В ходе работы помимо основной задачи была решена задача обмена данными через канал двойной связи (прямой канал и обратный), реализованный элементарно через два бинарных файла, задача не оказалась тривиальной: блок-схема и описание подводных камней прилагаются.

**Структура проекта**

**Интерфейс взаимодействия с пользователем**

#include <iostream>

#include <typeinfo>

#include "shifres.hpp"

using namespace std;

enum class Type {SHAMIR=1, DIFFI\_HELLMAN, RSA, EL\_GHAMAL};

template <typename Enumeration>

auto as\_integer(Enumeration const value) -> typename std::underlying\_type<Enumeration>::type {

return static\_cast<typename std::underlying\_type<Enumeration>::type>(value);

}

template<>

constexpr auto as\_integer<Type>(Type const value) -> typename std::underlying\_type<Type>::type {

int n = static\_cast<typename std::underlying\_type<Type>::type>(value);

if (n < 1 || n > 4)

throw std::runtime\_error("Incorrect input");

return n;

}

Type chooseEncode() {

int t;

inputerror:

cout << "1 - Shamir`s\n"

"2 - Diffi-Hellman`s\n"

"3 - RSA\n"

"4 - El-Ghamal`s\n"

"Enter type of simulated encryption algorythm: ";

cin >> t;

cout << endl;

try {

Type encode = static\_cast<Type>(t);

cout << as\_integer(encode) << endl;

return encode;

}

catch (std::exception &e) {

cerr << e.what() << endl;

goto inputerror;

}

}

void emulateRecipient(Type type) {

init("rec", "dis"); /// this is a crutch. it shouldn't be here, but we have it

switch (type) {

case Type::DIFFI\_HELLMAN: {

Diffi\_Hellman session;

cout << "\nMy shared key is " << session.sharedKey <<

". Just reporting, nothing more. I won`t tell you my secret key." << endl;

cout << "processing..." << endl;

session.recipient\_protocol();

cout << "\nI`m abonent B. My secret number is " << session.getEvaluatedNumber() <<

"; abonent A has equal secret number" << endl;

break;

}

case Type::EL\_GHAMAL: {

El\_Ghamal session;

session.recipient\_protocol();

break;

}

case Type::SHAMIR: {

Shamir session;

session.recipient\_protocol();

break;

}

case Type::RSA: {

RSA session;

session.recipient\_protocol();

break;

}

}

}

void emulateDispatcher(Type type) {

init("dis", "rec"); /// this is a crutch. it shouldn't be here, but we have it

switch (type) {

case Type::DIFFI\_HELLMAN: {

Diffi\_Hellman session;

cout << "\nMy shared key is " << session.sharedKey <<

". Just reporting, nothing more. I won`t tell you my secret key." << endl;

cout << "processing..." << endl;

session.dispatcher\_protocol();

cout << "\nI`m abonent A. My secret number is " << session.getEvaluatedNumber() <<

"; abonent B has equal secret number" << endl;

break;

}

case Type::EL\_GHAMAL: {

El\_Ghamal session;

session.dispatcher\_protocol();

break;

}

case Type::SHAMIR: {

Shamir session;

session.dispatcher\_protocol();

break;

}

case Type::RSA: {

RSA session;

session.dispatcher\_protocol();

break;

}

}

}

void chooseMode(Type type) {

int m;

err:

cout << "1 - Emulate recipient\n"

"2 - Emulate dispatcher to encrypt file (shared key of any recipient required)\n"

"Choose mode: ";

cin >> m;

cout << endl;

switch(m) {

case DISPATCHER:

emulateDispatcher(type);

break;

case RECIPIENT:

emulateRecipient(type);

break;

default:

goto err;

}

};

int main() {

Type type = chooseEncode();

chooseMode(type);

system("pause");

return 0;

}

Выбор, какого клиента эмулировать, отправителя или получателя. Выбор алгоритма для теста. Защита ввода. Пользователю нужно запустить два экземпляра /\*\*TODO сделат через терминал вз-ие \*\*/ программы и дождаться выполнения (передачи файла или удачного соединения).

**Описание классов алгоритмов через наследование**

#include "sh.h"

#include "locker.h"

#include <fstream>

#include <iostream>

#ifdef EXIT\_SUCCESS

# undef EXIT\_SUCCESS

#endif

enum Mode {RECIPIENT=1, DISPATCHER};

using namespace std;

class Environment {

public:

int encrMaxNumber, encrSuppNumber;

string input, output;

static Environment& Instance() { static Environment s; return s; }

private:

Environment() { encrMaxNumber = 1000667, encrSuppNumber = 500333; input = "input.txt", output = "output.txt"; }

~Environment() { }

Environment(Environment const&) = delete;

Environment& operator= (Environment const&) = delete;

};

class Encoded\_Structure {

public:

virtual void recipient\_protocol() = 0;

virtual void dispatcher\_protocol() = 0;

enum class FileStat{ EXIT\_SUCCESS, FAILED, UNNAMED\_ERROR };

protected:

virtual void decode();// = 0;

virtual void encode();// = 0;

virtual void takeSharedKey();// = 0;

virtual void giveSharedKey();// = 0;

void waitTilReady(WaitMode m);

uint8\_t takeByte();

void sendByte(uint8\_t data);

void sendEOF();

void write(int byte);

int read();

};

class Diffi\_Hellman : public Encoded\_Structure {

public:

int sharedKey;

Diffi\_Hellman();

void recipient\_protocol() override final;

void dispatcher\_protocol() override final;

int getEvaluatedNumber() { return evaluatedNumber; };

protected:

void takeSharedKey() override final;

void giveSharedKey() override final;

void decode() override final;

void encode() override final;

private:

int hiddenKey, takenSharedKey, evaluatedNumber;

};

class Shamir : public Encoded\_Structure {

public:

void recipient\_protocol() override final{};

void dispatcher\_protocol() override final{};

};

class El\_Ghamal : public Encoded\_Structure {

public:

void recipient\_protocol() override final{};

void dispatcher\_protocol() override final{};

};

class RSA : public Encoded\_Structure {

public:

void recipient\_protocol() override final{};

void dispatcher\_protocol() override final{};

};

Классы алгоритмов наследуются от основного класса, реализующего некоторые общие методы (чтения, записи данных, реализацию каналов их передачи). При этом, наследуясь от базового класса, потомки должны реализовать некоторые виртуальные методы родителя.

**Реализация алгоритмов**

1. Система Диффи-Хеллмана

Здесь абоненты просто установят связь и, вычислив секретное число, покажут пользователю.

2. Шифр Шамира

3. Шифр Эль-Гамаля

4. Алгоритм RSA

**Реализация канала двойной связи**

Если провести мысленный эксперимент, нетрудно догадаться, что одного файла не хватит, чтобы провести передачу данных **синхронно**. Потребуется как минимум два файла-канала, первый будет выступать в роли прямого канала, второй – обратного. Для абонента с другой стороны линии наоборот.

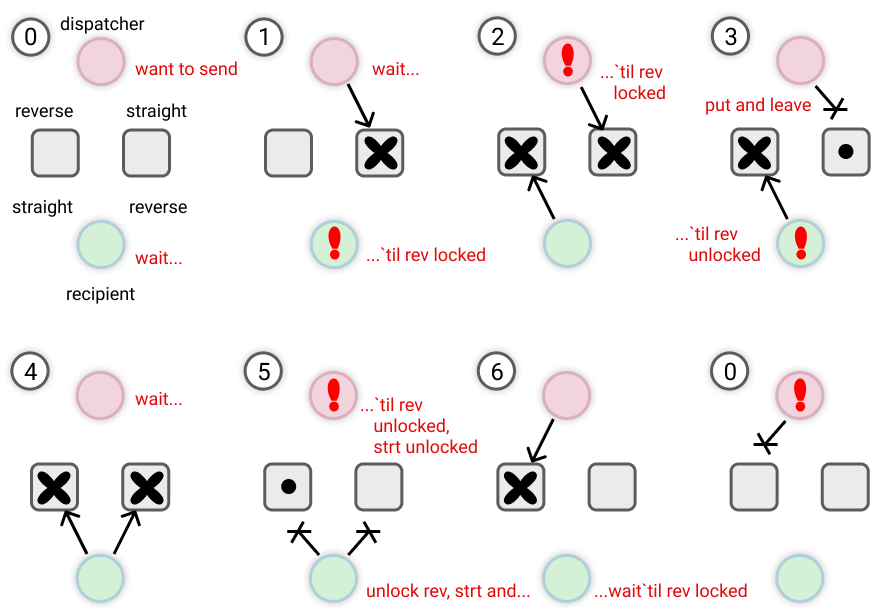
****

Рис.1 Единственный способ синхронной передачи данных через двойной канал

Имея возможности (реализованные в виде функций из библиотеки ниже) блокирования/разблокирования файла, чтения/записи(с блокировкой соответственно) и проверки, заблокирован ли файл, передачу организовать можно только изображенным на схеме образом.

UPD: Зачем передавать через файлы, ведь это можно сделать через localhost или разделяемую память? Это была первая идея. К тому же функции отправки можно заменить на использующие localhost, не трогая остальной код. Почти.

**Реализация блокировки файловых потоков**

#ifdef BUILD\_DLL

#define DLL\_EXPORT \_\_declspec(dllexport)

#else

#define DLL\_EXPORT \_\_declspec(dllimport)

#endif

#include <stdint-gcc.h>

#define EOF (-1)

enum WaitMode { OREV\_LOCKED=1, REV\_LOCKED=2, REV\_UNLOCKED=3, REV\_STRT\_UNLOCKED=5 };

#ifdef \_\_cplusplus

extern "C"

{

#endif

void DLL\_EXPORT init(const char\* straightChannel, const char\* inverseChannel);

int DLL\_EXPORT readByte(); /// only from reverse channel /// !

void DLL\_EXPORT putByte(int data); /// only to straight channel /// !

void DLL\_EXPORT unlock\_straight\_channel(); /// theese six methods should be used with caution (inf-loop)

void DLL\_EXPORT unlock\_reverse\_channel(); /// !

void DLL\_EXPORT lock\_reverse\_channel(); /// !

void DLL\_EXPORT lock\_straight\_channel(); /// !

void DLL\_EXPORT trylock\_reverse\_channel();

void DLL\_EXPORT trylock\_straight\_channel();

#ifdef \_\_cplusplus

}

#endif

#include "locker.h"

#include <stdio.h>

#include <share.h>

#include <string.h>

#include <errno.h>

static FILE \*stFile, \*invFile;

static char straightFilename[40] = {0}, inverseFilename[40] = {0};

void DLL\_EXPORT init(const char\* straight, const char\* inverse) {

int i = 0;

while(straight[i++]);

memcpy(straightFilename, straight, i);

i = 0;

while(inverse[i++]);

memcpy(inverseFilename, inverse, i);

fclose(fopen(straightFilename, "wb")), fclose(fopen(inverseFilename, "wb"));

}

void DLL\_EXPORT unlock\_straight\_channel() { fclose(stFile); }

void DLL\_EXPORT unlock\_reverse\_channel() { fclose(invFile); }

void DLL\_EXPORT lock\_reverse\_channel() {

do {

errno=0;

invFile = \_fsopen(inverseFilename, "rb", \_SH\_DENYRW);

} while(errno!=0);

}

void DLL\_EXPORT lock\_straight\_channel() {

do {

errno=0;

stFile = \_fsopen(straightFilename, "wb", \_SH\_DENYRW);

} while(errno!=0);

}

void DLL\_EXPORT trylock\_reverse\_channel() {

FILE \*f = \_fsopen(inverseFilename, "rb", \_SH\_DENYNO);

if(f) fclose(f), errno=0; /// anyway, fclose locks file

else errno=EACCES;

}

void DLL\_EXPORT trylock\_straight\_channel() {

FILE \*f = \_fsopen(straightFilename, "rb", \_SH\_DENYNO);

if(f) fclose(f), errno=0;

else errno=EACCES;

}

int DLL\_EXPORT readByte() {

do{ errno=0;

invFile = \_fsopen(inverseFilename, "rb", \_SH\_DENYRW);

}while(errno!=0);

int n = fgetc(invFile);

fgetc(invFile);

return feof(invFile) ? n : EOF;

}

void DLL\_EXPORT putByte(int data) {

do{ errno=0;

stFile = \_fsopen(straightFilename, "wb", \_SH\_DENYRW);

}while(errno!=0);

if(0 <= data && data < 256)

fputc(data, stFile);

else if (data == EOF) {

fputc(0xff, stFile);

fputc(0x01, stFile);

}

}

int \_\_attribute\_\_((\_\_stdcall\_\_)) DllMainCRTStartup(int hinstDLL, int fdwReason, void\* lpvReserved){

return 1;

}

**Результат работы программы**

**/\*\*TODO показать shell\*\*/**