Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт Космических и информационных технологий

институт

Кафедра «Информатика»

кафедра

**ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ №1**

«Симметричная криптография. Простые симметричные шифры»

тема

Преподаватель \_\_\_\_\_\_\_\_\_ А.А. Сидарас

подпись, дата инициалы, фамилия

Студент КИ18-17/1б 031830504 \_\_\_\_\_\_\_\_\_ Е.В. Железкин

номер группы, зачетной книжки подпись, дата инициалы, фамилия

Красноярск 2020

Содержание

[Содержание 2](#_Toc18956750)

[1 Цель работы 3](#_Toc18956751)

[2 Задача работы 3](#_Toc18956752)

[3 Ход работы 3](#_Toc18956753)

[4 Вывод 10](#_Toc18956754)

# Цель работы

1. Ознакомиться с основами симметричного шифрования
2. Ознакомиться с простыми симметричными криптографическими шифрами на основе методов подстановок, перестановок и гаммирования
3. Освоить основные этапы проектирования и реализации симметричных шифров.

# Задача работы

Согласно персональному варианту разработать и составить в виде блок-схемы алгоритм шифрования и дешифрования текста. Убедиться в правильности составления алгоритма и затем на языке программирования составьте программу, которая реализует данный алгоритм.

На ряде контрольных примеров (не менее 10) открытого текста, состоящего из различного количества символов, проверить правильность работы алгоритма шифрования и дешифрования.

Самостоятельно или с помощью преподавателя придумать оригинальный

способ модификации шифра с целью повышения его криптостойкости. Внести изменения в исходный алгоритм и программу. Проверить работоспособность алгоритма на тестовых примерах.

Доказать, что предложенный способ модификации действительно повышает криптостокость.

Разработанная программа должна содержать графический интерфейс пользователя.

**Вариант 7** - Шифр Трисемуса.

# Краткие теоретические сведения

Шифр Тритемиуса — система шифрования, разработанная Иоганном Тритемием. Представляет собой усовершенствованный шифр Цезаря, то есть шифр подстановки. По алгоритму шифрования, каждый символ сообщения смещается на символ, отстающий от данного на некоторый шаг. Здесь шаг смещения делается переменным, то есть зависящим от каких-либо дополнительных факторов. Например, можно задать закон смещения в виде линейной функции (уравнения зашифрования) позиции шифруемой буквы. Сама функция должна гарантировать целочисленное значение. Прямая функция шифрования должна иметь обратную функцию шифрования, тоже целочисленную.

Уравнением зашифрования называется соотношение, описывающее процесс образования зашифрованных данных из открытых данных в результате преобразований, заданных алгоритмом криптографического преобразования.

Уравнение зашифрования для шифра Тритемиуса имеет следующий вид:



Рисунок 1 – Уравнение шифра Трисемуса

где L- номер зашифрованной буквы в алфавите; m — номер позиции буквы шифруемого текста в алфавите; k — шаг смещения (функциональная зависимость от позиции буквы в сообщении); N — число букв алфавита.

Некоторые варианты вычисления шага смещения k:



Рисунок 2 – Варианты вычисления k

# Ход работы

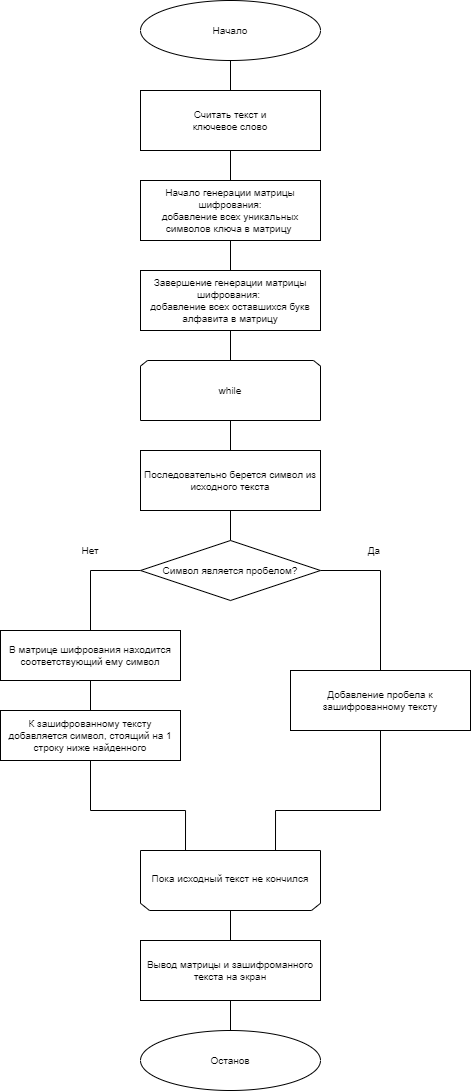


Рисунок 3 – Блок-схема алгоритма

Листинг 1 – файл main.cpp

#include "ssc.h"

#include <QApplication>

int main(int argc, char \*argv[])

{

QApplication a(*argc*, argv);

SSC w;

w.show();

return a.exec();

}

Листинг 2 – заголовочный файл класса SSC ssc.h

#ifndef SSC\_H

#define SSC\_H

#include <QMainWindow>

QT\_BEGIN\_NAMESPACE

namespace Ui { class SSC; }

QT\_END\_NAMESPACE

class SSC : public QMainWindow

{

Q\_OBJECT

public:

SSC(QWidget \*parent = nullptr);

~SSC();

private slots:

void Encryption();

void strEdit(QString);

void keyEdit(QString);

QString getTextFromLine1();

QString getTextFromLine2();

void setTextToTB1(QString);

void setTextToTB2(QString);

int getShiftFromRB();

private:

Ui::SSC \*ui;

QString alfabet = "abcdefghijklmnopqrstuvwxyz ";

};

#endif // SSC\_H

Листинг 3 – файл класса SSC ssc.cpp

#include "ssc.h"

#include "ui\_ssc.h"

#include "func.cpp"

SSC::**SSC**(QWidget \*parent)

: QMainWindow(parent)

, ui(new Ui::SSC)

{

ui->setupUi(this);

connect(ui->pushButton, SIGNAL(clicked()), this, SLOT(Encryption()));

connect(ui->lineEdit, SIGNAL(textEdited(QString)), this, SLOT(strEdit(QString)));

connect(ui->lineEdit\_2, SIGNAL(textEdited(QString)), this, SLOT(keyEdit(QString)));

}

void SSC::**strEdit**(QString change)

{

QString newStr;

for (int i = 0; i < change.length(); i++)

if (alfabet.indexOf(change[i], 0, Qt::CaseInsensitive) != -1) newStr = newStr + change[i];

ui->lineEdit->setText(newStr);

}

void SSC::**keyEdit**(QString change)

{

QString newStr;

for (int i = 0; i < change.length(); i++)

if (alfabet.indexOf(change[i], 0, Qt::CaseInsensitive) != -1 &&

alfabet.indexOf(change[i], 0, Qt::CaseInsensitive) != 26) newStr = newStr + change[i];

ui->lineEdit\_2->setText(newStr);

}

QString SSC::**getTextFromLine1**(){

return ui->lineEdit->text();

}

QString SSC::**getTextFromLine2**(){

return ui->lineEdit\_2->text();

}

void SSC::**setTextToTB1**(QString temp){

return ui->textBrowser->setText(temp);

}

void SSC::**setTextToTB2**(QString temp){

return ui->textBrowser\_2->setText(temp);

}

int SSC::**getShiftFromRB**(){

return (ui->radioButton->isChecked() \* 1) + (ui->radioButton\_2->isChecked() \* 2) +

(ui->radioButton\_3->isChecked() \* 3) + (ui->radioButton\_4->isChecked() \* 4);

}

SSC::~***SSC***()

{

delete ui;

}

Листинг 4 – файл, содержащий функцию шифрования, func.cpp

#include "ssc.h"

void SSC::**Encryption**(){

QString str;

QString key;

QString newKey;

QString newStr;

str = SSC::getTextFromLine1();

key = SSC::getTextFromLine2();

int flag = 0;

for (int i = 0; i < key.length(); i++){

flag = 0;

for (int j = 0; j < newKey.length(); j++)

if (newKey[j] == key[i]) flag++;

if (flag == 0)

newKey = newKey + key[i];

}

int pos = 0;

for (int i = 0; i < alfabet.length(); i++)

if (newKey.indexOf(alfabet[i], 0, Qt::CaseInsensitive) == -1) newKey = newKey + alfabet[i];

for (int i = 0; i < str.length(); i++){

pos = newKey.indexOf(str[i], 0, Qt::CaseInsensitive);

if (pos > 19 && pos < 24) pos = pos % 18;

else

if (pos > 23) pos = pos % 24;

else

pos = pos + SHIFT\_FOR\_STR;

newStr = newStr + newKey[pos];

}

QString out;

for (int i = 0; i < newKey.length(); i++){

if (i % 6 == 0 && i != 0) out = out + "\n";

out = out + newKey[i] + " ";

}

out = out + "\* \* \* \*";

SSC::setTextToTB2(out);

SSC::setTextToTB1(newStr);

}

Алгоритм Трисемуса был усовершенствован с помощью добавления возможности выбирать сдвиг в матрице шифрования (только в вертикальной плоскости). Если предположить, что злоумышленник знает шифр и ключ, то данная модификация теоретически должна усложнить сложность расшифровки в 4 раза (т. к. в матрице 4 полные строки и соответственно сдвиг можно выполнять на 4 позиции).

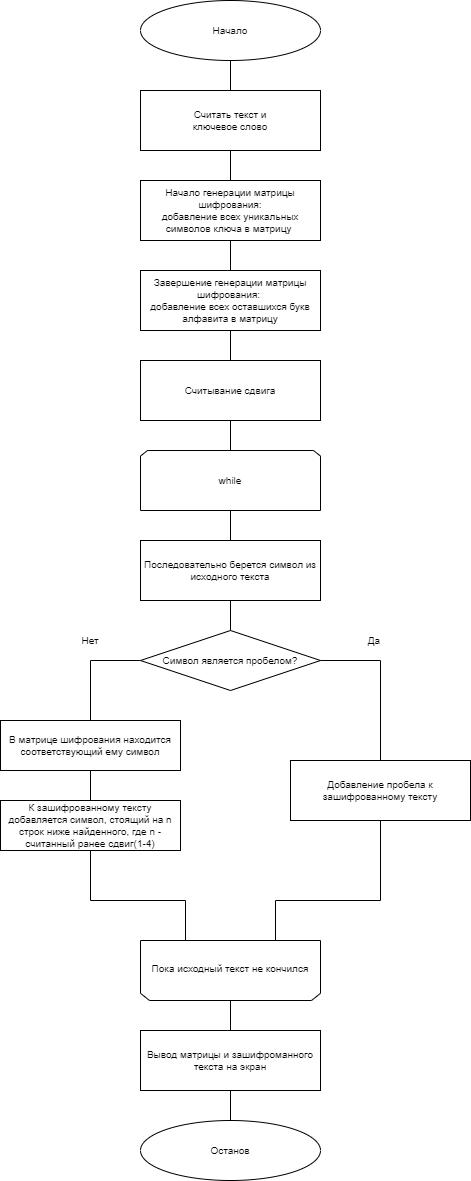


Рисунок 4 - Блок-схема модифицированного алгоритма

Листинг 5 – файл, содержащий модернизированную функцию шифрования, func.cpp

#include "ssc.h"

void SSC::**Encryption**(){

QString str;

QString key;

QString newKey;

QString newStr;

// Получение исходного текста и ключа из формы

str = SSC::getTextFromLine1();

key = SSC::getTextFromLine2();

int flag = 0;

// Создание первой части матрицы шифрования из ключа

for (int i = 0; i < key.length(); i++){

flag = 0;

for (int j = 0; j < newKey.length(); j++)

if (newKey[j] == key[i]) flag++;

if (flag == 0)

newKey = newKey + key[i];

}

int pos = 0;

int shift = SSC::getShiftFromRB();

// Создание второй части матрицы шифрования из букв, которые не вошли в ключ

for (int i = 0; i < alfabet.length() - 1; i++)

if (newKey.indexOf(alfabet[i], 0, Qt::CaseInsensitive) == -1) newKey = newKey + alfabet[i];

// Шифрование исходного текста

for (int i = 0; i < str.length(); i++){

if (str[i] == ' ') {

newStr += str[i];

continue;

}

pos = newKey.indexOf(str[i], 0, Qt::CaseInsensitive) + shift \* SHIFT\_FOR\_STR;

if (pos > 25 && pos < 30) pos += SHIFT\_FOR\_STR;

if (pos >= 30) pos %= 30;

newStr = newStr + newKey[pos];

}

QString out;

// Подготовка матрицы шифрования(пеобразование из строки в таблицу)

for (int i = 0; i < newKey.length(); i++){

if (i % SHIFT\_FOR\_STR == 0 && i != 0) out = out + "\n";

out = out + newKey[i] + " ";

}

out = out + "\* \* \* \*";

// Вывод зашифрованного текста и матрицы

SSC::setTextToTB2(out);

SSC::setTextToTB1(newStr);

}

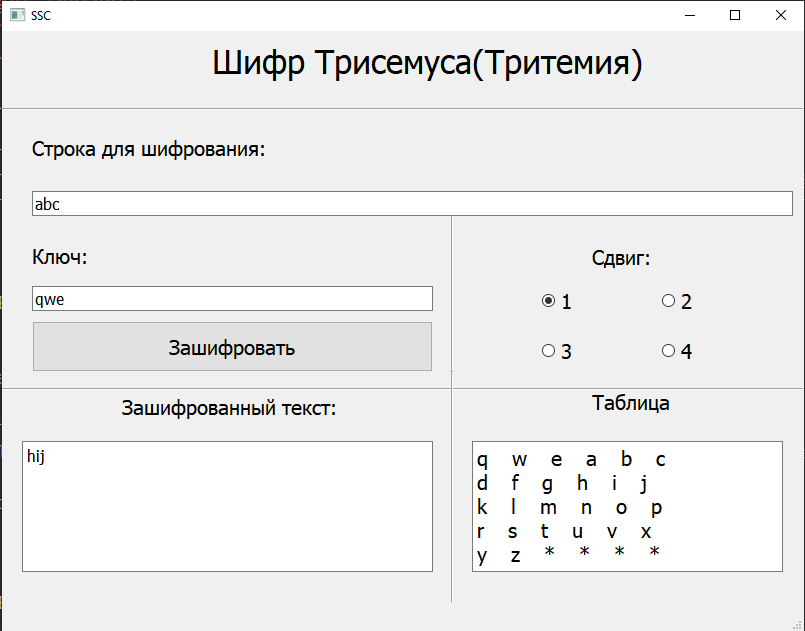


Рисунок 5 – Пример работы 1

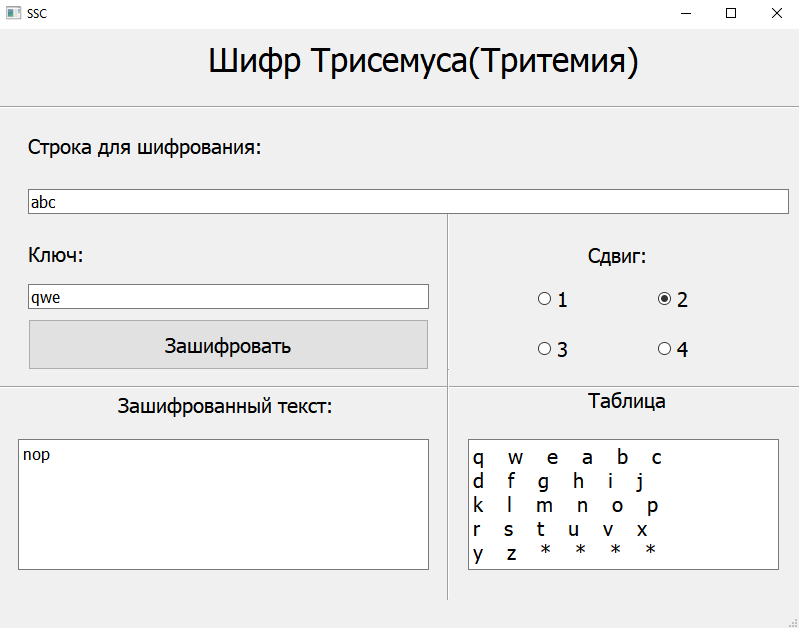


Рисунок 6 – Пример работы 2

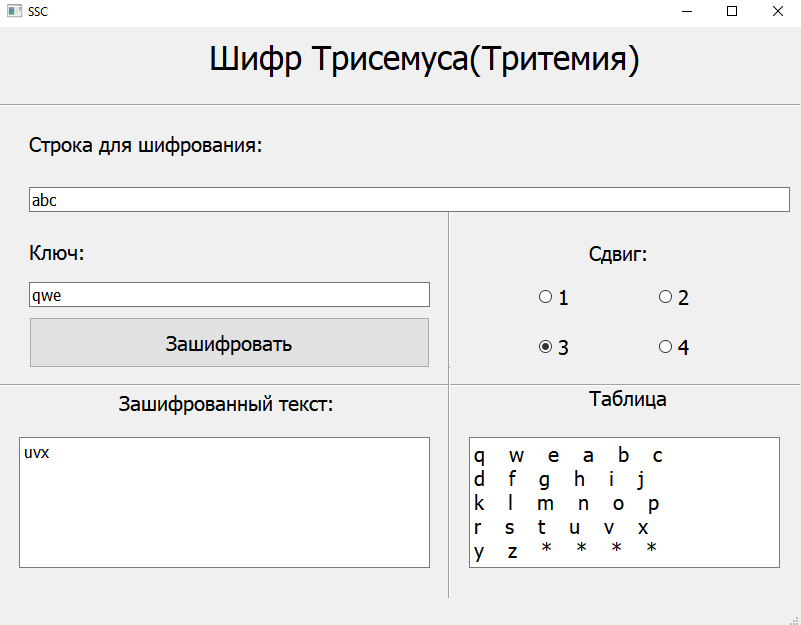


Рисунок 7 – Пример работы 3

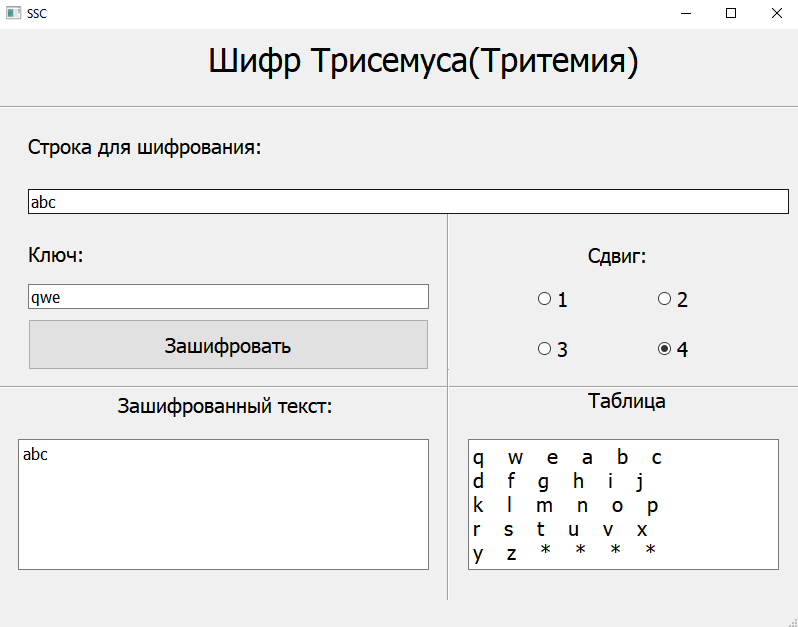


Рисунок 8 – Пример работы 4

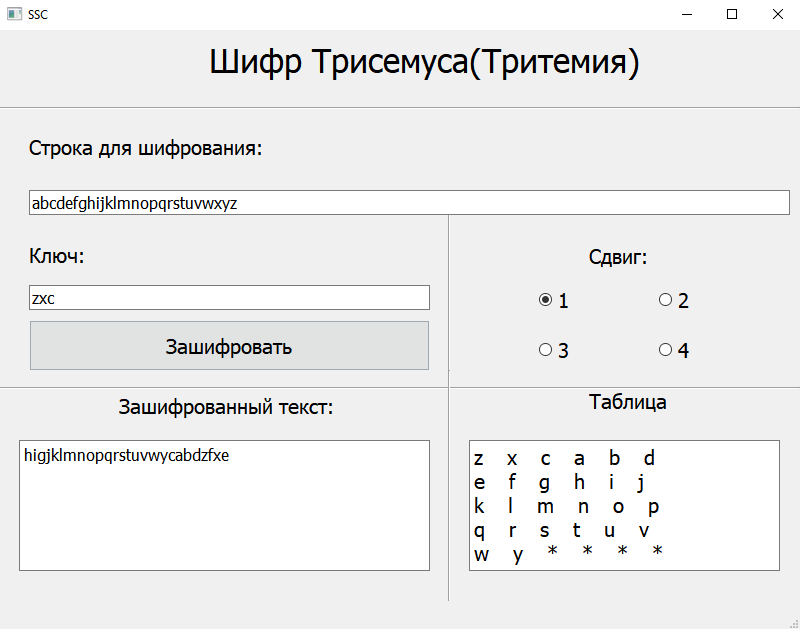


Рисунок 9 – Пример работы 5

# Вывод

В ходе выполнения данной практической работы были изучены основы симметричного шифрования на основе методов подстановок, перестановок и гаммирования. Реализован алгоритм на основе шифра Трисемуса, а также сделаны первые шаги в сфере криптостойкости.