Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Пермский национальный исследовательский политехнический университет»
Электротехнический факультет
Кафедра «Информационные технологии и автоматизированные системы»

Дисциплина: «Защита информации»

Профиль: «Автоматизированные системы обработки информации и управления»

Семестр 5

ОТЧЕТ
по лабораторной работе №8

Тема: «Методы Гаммирования»

Выполнил: студент группы РИС-19-1б
Миннахметов Э.Ю. _____
Проверил: доцент кафедры ИТАС
Шереметьев В. Г. _____
Дата _____

Пермь, 2021

ЦЕЛЬ РАБОТЫ

Получить практические навыки по применению метода шифрования однократной случайной равновероятной гаммой – однократного гаммирования.

ЗАДАНИЕ

Вариант №14. Реализовать шифрование файла методом однократного гаммирования, используя блоки открытого текста длиной 48 бита и используя в алгоритме шифрования операцию исключающее или.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Шифрование методом гаммирования

Под гаммированием понимают процесс наложения по определенному закону гаммы шифра на открытые данные. Гамма шифра — это псевдослучайная последовательность, выработанная по заданному алгоритму для зашифрования открытых данных и расшифрования зашифрованных данных.

Процесс зашифрования заключается в генерации гаммы шифра и наложении полученной гаммы на исходный открытый текст обратимым образом, например с использованием операции сложения по модулю 2.

Следует отметить, что перед зашифрованием открытые данные разбивают на блоки T_0^i одинаковой длины, обычно по 64 бита. Гамма шифра вырабатывается в виде последовательности блоков $Г_{ш}^i$ аналогичной длины.

Уравнение зашифрования можно записать в виде

$$Т_{ш}^i = Г_{ш}^i \oplus Т_0^i, i = 1...M$$

где $Т_{ш}^i$ - i-й блок шифртекста; $Г_{ш}^i$- i-й блок гаммы шифра; $Т_0^i$- i-й блок открытого текста; M - количество блоков открытого текста.

Процесс расшифрования сводится к повторной генерации гаммы шифра и наложению этой гаммы на зашифрованные данные. Уравнение расшифрования имеет вид обратный вид.

Получаемый этим методом шифртекст достаточно труден для раскрытия, поскольку теперь ключ является переменным. По сути дела гамма шифра должна изменяться случайным образом для каждого шифруемого блока. Если период гаммы превышает длину всего шифруемого текста и злоумышленнику неизвестна никакая часть исходного текста, то такой шифр можно раскрыть только прямым перебором всех вариантов ключа. В этом случае криптостойкость шифра определяется длиной ключа.

«Разоблачение» гаммирования.

С точки зрения теории криптоанализа метод шифрования однократной случайной равновероятной гаммой той же длины, что и открытый текст, является невскрываемым (далее для краткости авторы будут употреблять термин «однократное гаммирование», держа в уме все вышесказанное). Обоснование, которое привел Шеннон, основываясь на введенном им же понятии информации, не дает возможности усомниться в этом - из-за равных априорных вероятностей криптоаналитик не может сказать о дешифровке, верна она или нет. Кроме того, даже раскрыв часть сообщения, дешифровщик не сможет хоть сколько-нибудь поправить положение - информация о вскрытом участке гаммы не дает информации об остальных ее частях.

Логично было бы предположить, что для организации канала конфиденциальной связи в открытых сетях следовало бы воспользоваться именно схемой шифрования однократного гаммирования. Ее преимущества вроде бы очевидны. Есть, правда, один весомый недостаток, который сразу бросается в глаза, - это необходимость иметь огромные объемы данных, которые можно было бы использовать в качестве гаммы. Для этих целей обычно пользуются датчиками настоящих случайных чисел (в западной литературе аналогичный термин носит название True Random Number Generator или TRNG). Это уже аппаратные устройства, которые по запросу выдают набор случайных чисел, генерируя их с помощью очень большого количества физических параметров окружающей среды. Статистические характеристики таких наборов весьма близки к характеристикам "белого шума", что означает равновероятное появление каждого следующего числа в наборе. А это, в свою очередь, означает для нас действительно равновероятную гамму.

К сожалению, для того чтобы организовать конфиденциальный канал передачи данных, потребуется записать довольно большое количество этих данных и обменяться ими по секретному каналу. Уже одно это условие делает однократное гаммирование во многих случаях неприемлемым. В самом деле, зачем передавать что-то по открытому незащищенному каналу, когда есть возможность передать все это по секретному защищенному? И хотя на простой вопрос, является ли метод использования однократной случайной равновероятной гаммы стойким к взлому, существует положительный ответ, его использование может оказаться попросту невозможным.

Да и к тому же метод однократного гаммирования криптостоек только в определенных, можно даже сказать, тепличных условиях. Что же касается общего случая, то все не так просто.

Показать слабости шифра однократного гаммирования можно, говоря наукообразно, с помощью примера или, что называется, "на пальцах". Представим следующую ситуацию.

Допустим, в тайной деловой переписке используется метод однократного наложения гаммы на открытый текст.

Чтобы дальнейшие рассуждения были как можно более понятны, рассмотрим следующее свойство шифротекста. Предположим, что мы знаем часть гаммы, которая была использована для шифрования текста «Приветствую, мой ненаглядный сосед!» в формате ASCII, кодировка WIN-1251.

П р и в е т с т в у ю , _ м о й

CF F0 E8 E2 E5 F2 F1 F2 E2 F3 FE 2C 20 EC EE E9

_ н е н а г л я д н ы й _ с о с

20 ED E5 ED E0 E3 EB FF E4 ED FB E9 20 F1 EE F1

е д !

E5 E4 21

ХОД РАБОТЫ

На рисунке 1 представлена главная форма программы.

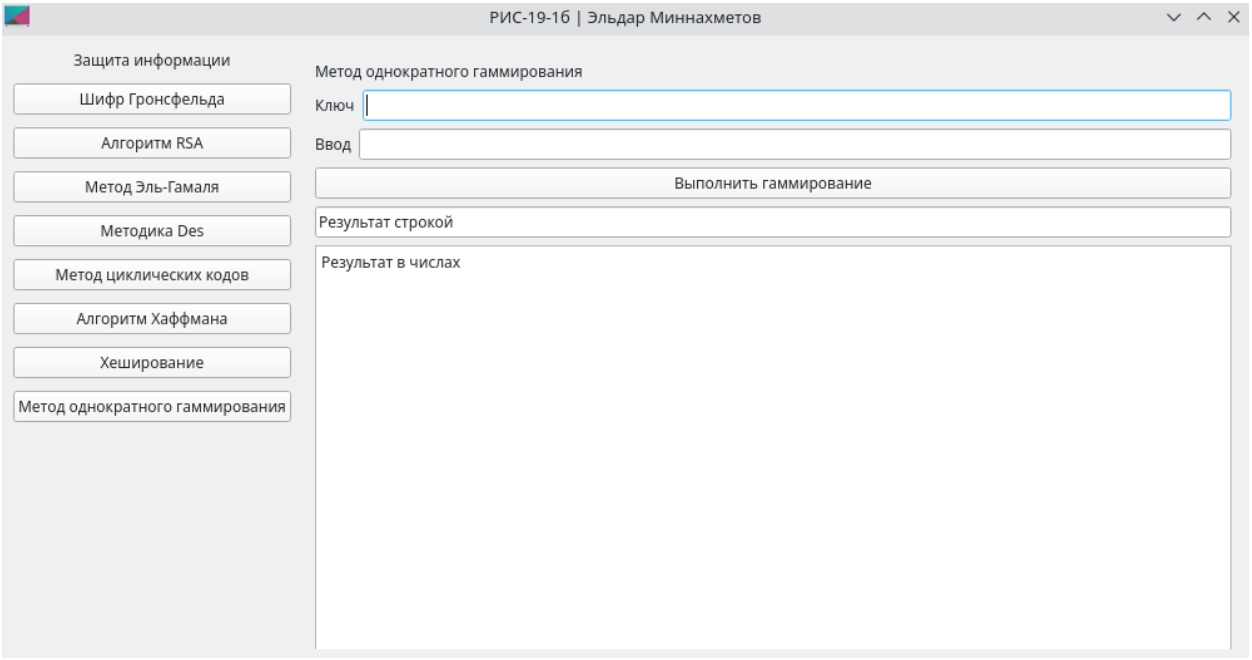

Рисунок 1 – Главная форма программы.

Пример работы программы представлен на рисунке 2.

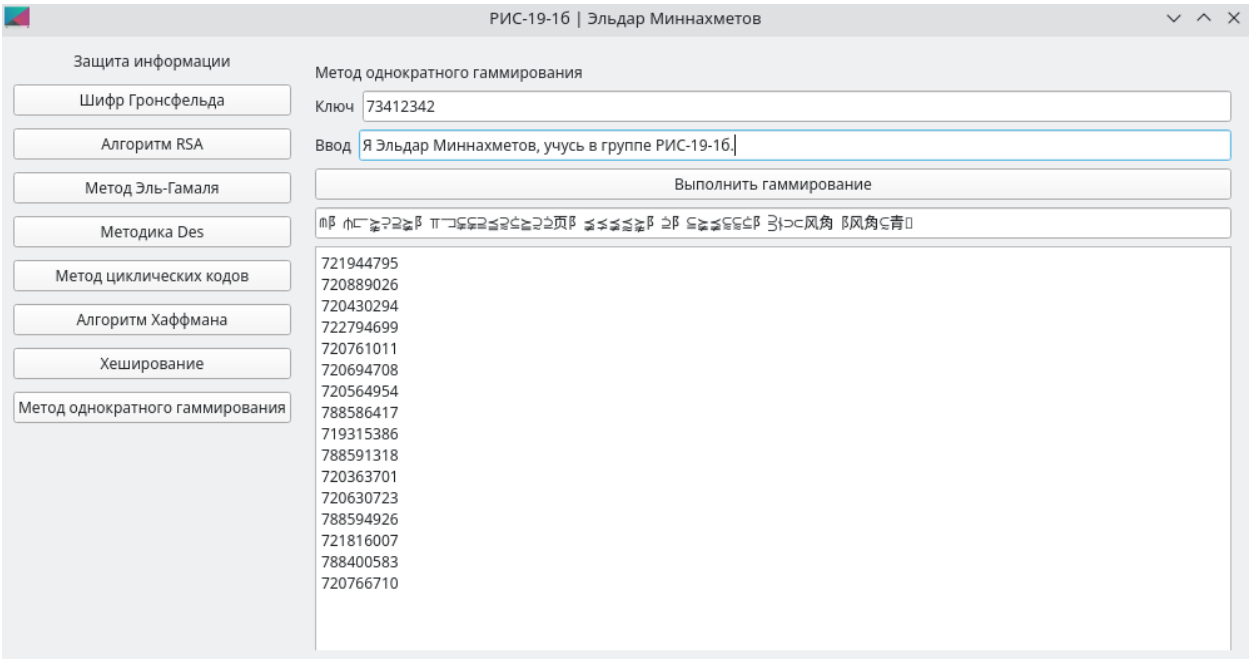

Рисунок 2 – Пример работы программы.

ПРИЛОЖЕНИЕ А

Листинг класса GammaTask.h

```
#pragma once

#include <QWidget>

#include "Task.h"

class QVBoxLayout;
class QHBoxLayout;
class QLabel;
class QLineEdit;
class QPushButton;
class QTextEdit;

struct u48 {
    ushort s[3];
};

void getBlocks(QString text, u48 *&out, int &n);

QString getString(u48 *&in, const int &n);

class GammaTask: public QWidget, public Task {
Q_OBJECT

private:
    QVBoxLayout *lytMain;
    QHBoxLayout *lytKey;
    QHBoxLayout *lytIn;
    QLabel *lblName;
    QLabel *lblKey;
    QLabel *lblIn;
    QLineEdit *leKey;
    QLineEdit *leIn;
    QPushButton *btn;
    QLineEdit *leResult;
    QTextEdit *txtResult;

public:
    GammaTask() : Task("Метод однократного гаммирования") {}

    void initWidget(QWidget *wgt) override;

    void run() const override {}

private slots:
    void generate();

};
```

ПРИЛОЖЕНИЕ Б

Листинг класса GammaTask.cpp

```
#include "GammaTask.h"

#include <QWidget>
#include <QVBoxLayout>
#include <QLabel>
#include <QLineEdit>
#include <QPushButton>
#include <QTextEdit>

void getBlocks(QString text, u48 *&out, int &n) {
   int N = text.size();
   int m = N + (N % 3 ? 3 - N % 3 : 0);
   n = m / 3;
   out = new u48[n];
   ushort *o = (ushort *) out;
   for (int i = 0; i <= n; ++i) {
      for (int j = 0; j < 3; ++j) {
         int k = i * 3 + j;
         if  (k < N) {
            o[k] = text[k].unicode();
         } else {
            o[k] = 0;
         }
      }
   }
}

void hash(u48 *&in, const int &n, const unsigned long long &key) {
   for(int i = 0; i < n * 3; ++i) {
      ((ushort*)in)[i] ^= key;
   }
}

QString getString(u48 *&in, const int &n) {
   QString out;
   int N = n * 3;
   ushort *o = (ushort *) in;
   for(int i = 0; i < N; ++i) {
      out += QChar(o[i]);
   }
   return out;
}

QString getNumbers(u48 *&in, const int &n) {
   QString out;
   int N = n * 3;
   ushort *o = (ushort *) in;
   for(int i = 0; i < N; i += 3) {
      out += QString::asprintf("%llu\n", ((unsigned long long)o[i] * 256 + o[i + 1]) * 256 + o[i + 2]);
   }
   return out;
}

void GammaTask::initWidget(QWidget *wgt) {
   lytMain = new QVBoxLayout;
   lytKey = new QHBoxLayout;
```

```
    lytIn = new QHBoxLayout;
    lblName = new QLabel("Метод однократного гаммирования");
    lblKey = new QLabel("Ключ");
    lblIn = new QLabel("Ввод");
    leResult = new QLineEdit("Результат строкой");
    txtResult = new QTextEdit("Результат в числах");
    leKey = new QLineEdit;
    leIn = new QLineEdit;
    btn = new QPushButton("Выполнить гаммирование");

    wgt->setLayout(lytMain);
    lytMain->addWidget(lblName);
    lytMain->addLayout(lytKey);
    lytKey->addWidget(lblKey);
    lytKey->addWidget(leKey);
    lytMain->addLayout(lytIn);
    lytIn->addWidget(lblIn);
    lytIn->addWidget(leIn);
    lytMain->addWidget(btn);
    lytMain->addWidget(leResult);
    lytMain->addWidget(txtResult);

    lytMain->setAlignment(Qt::Alignment::enum_type::AlignTop);
    connect(btn, SIGNAL(released()), SLOT(generate()));
}

void GammaTask::generate() {
    int n;
    u48 *bytes;
    getBlocks(leIn->text(), bytes, n);
    unsigned long long key = leKey->text().toULongLong();
    hash(bytes, n, key);
    QString str = getString(bytes, n);
    QString nums = getNumbers(bytes, n);
    delete[] bytes;
    leResult->setText(str);
    txtResult->setText(nums);
}
```