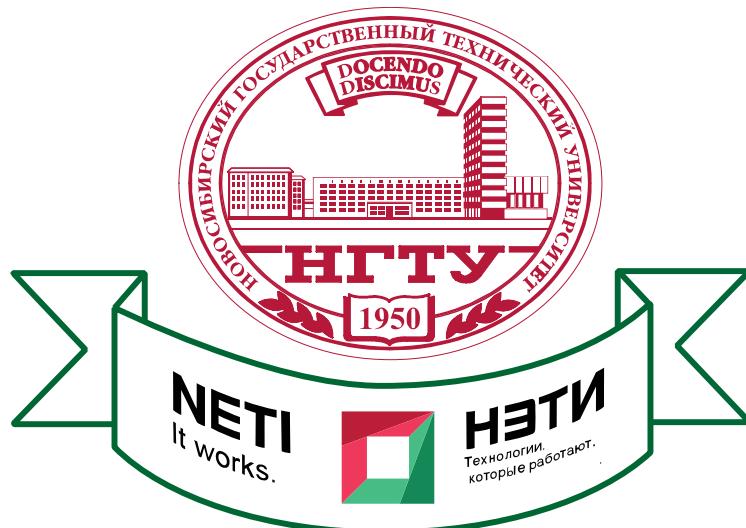


Министерство науки и высшего образования
Российской Федерации

Федеральное государственное бюджетное
образовательное учреждение высшего образования

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

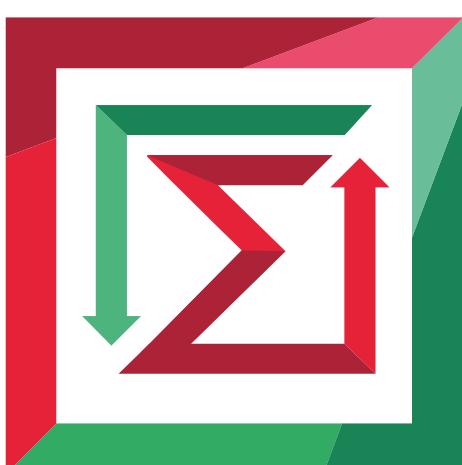


Теоретической и прикладной математики

Лабораторная работа № 4

по дисциплине «ОСНОВЫ ТЕОРИИ ИНФОРМАЦИИ КРИПТОГРАФИИ»

ШИФРЫ ПОДСТАНОВКИ И ЗАМЕНЫ



Факультет: ПМИ
Группа: ПМИ-02
Вариант: 7
Студент: Сидоров Даниил,
Дюков Богдан
Преподаватель: Авдеенко Татьяна Владимировна,
Сивак Мария Алексеевна.

Новосибирск

2026

1. Цель работы

Освоить основные алгоритмы шифрования перестановки и замены.

2. Задача

1. Реализовать приложение для шифрования:
 - a. Шифруемый текст должен храниться в одном файле, а ключ шифрования (если есть) – в другом.
 - b. Шифрование производится согласно заданному в варианте алгоритму. Конкретную реализацию алгоритма нужно выбрать самостоятельно. В алфавит шифруемых сообщений, который задан в варианте, нужно добавить символ '_', который является разделителем слов.
 - c. Зашифрованный текст должен сохраняться в файл.
2. Реализовать приложение для дешифрования:
 - a. Зашифрованный текст должен храниться в одном файле, ключ (если есть) – в другом.
 - b. Расшифрованный текст должен сохраняться в файл.
3. С помощью реализованных приложений выполнить следующие задания:
 - a. Протестировать правильность работы разработанных приложений при различных сообщениях и ключах.
 - b. Выполнить шифрование нескольких сообщений аналитически (вручную) и сравнить полученные шифротексты с результатами работы шифрующего приложения.
 - c. Проанализировать шифр на приведённые в лекции типы криptoаналитического вскрытия.
 - d. Проанализировать шифр с точки зрения совершенной криптостойкости, т.е. проверить выполнение условий теорем (о совершенной криптостойкости симметричной крипtosистемы). Ответить на вопрос об идеальной стойкости данного шифра.
 - e. Сделать выводы о проделанной работе.

Вариант	Алгоритм шифрования	Алфавит сообщения	Дополнительная информация
7	Шифр Вернама	* , -	

3. Разработанное программное средство

Программа шифр Вернама:

1. Окно шифрование:

На вход программе может подаваться файл с шифруемой последовательностью, для которой имеется возможность сгенерировать случайный ключ шифрования той же длины (его можно сохранить в файл). Поля с шифруемой последовательностью и ключом шифрования можно редактировать вручную.

В случае успеха программа выводит в заданное поле зашифрованную последовательность, которую можно сохранить в файл.

В случае неудачного ввода (длина ключа не равна длине сообщения или ввод символов не из алфавита) программа выводит соответствующее сообщение.

2. Окно дешифрование:

На вход программе может подаваться файл с зашифрованной последовательностью, а также файл с ключом, полученным при шифровании. Поля с зашифрованной последовательностью и ключом шифрования можно редактировать вручную.

В случае успеха программа выводит в заданное поле Расшифрованную последовательность, которую можно сохранить в файл.

В случае неудачного ввода (длина ключа не равна длине сообщения или ввод символов не из алфавита) программа выводит соответствующее сообщение.

4. Исследования

1) Анализ шифра с использованием приведённых в лекции типов криptoаналитического вскрытия

Вскрытие с использованием выбранного открытого текста.

Дано: $M_1, C_1 = E_k(M_1), \dots, M_i, C_i = E_k(M_i)$.

Найти: k или алгоритм, как получить M_{i+1} из $C_{i+1} = E_k(M_{i+1})$.

Анализ: по открытому тексту нельзя получить ни ключ, ни алгоритм.

Вскрытие с использованием открытого текста.

Дано: $M_1, C_1 = E_k(M_1), \dots, M_i, C_i = E_k(M_i)$, при этом $M_1 \div M_i$ можно выбирать любыми.

Найти: k или алгоритм, как получить M_{i+1} из $C_{i+1} = E_k(M_{i+1})$.

Анализ: по открытому тексту нельзя получить ни ключ, ни алгоритм.

Вскрытие с использованием выбранного ключа (у криптоаналитика имеется некоторая информация о связи между различными ключами).

Анализ: по ключу можно понять, как происходит шифрование и дешифрование.

Вскрытие с использованием выбранного шифротекста (обычно применяется к алгоритмам с открытым ключом).

Дано: $C_1, M_1 = D_k(C_1), \dots, C_i, M_i = D_k(C_i)$, при этом $C_1 \div C_i$ можно выбирать любыми.

Найти: k

Анализ: имея шифротекст и алгоритм, нельзя получить ключ, так как он создается случайным образом для каждого шифруемого сообщения.

Вскрытие только с использованием шифротекста (у криptoаналитика есть несколько зашифрованных одним и тем же алгоритмом сообщений).

Дано: $C_1, M_1 = D_k(C_1), \dots, C_i, M_i = D_k(C_i)$.

Найти: $M_1 \div M_i, k$ или алгоритм, как получить M_{i+1} из $C_{i+1} = E_k(M_{i+1})$.

Анализ: имея шифротекст нельзя получить ни ключ, ни алгоритм.

Бандитский криптоанализ (угрозы, шантаж, и т. д. Это наилучший путь взлома крипtosистемы).

Анализ: рабочий метод.

2) Анализ шифра с точки зрения совершенной криптостойкости (выполнение теорем):

Шифр Вернама обладает абсолютной криптографической стойкостью.

Необходим и достаточным условием абсолютной криптографической стойкости является:

1. Полная случайность ключа;
2. Равенство длин ключа и открытого текста;
3. Однократное использование ключа.

Докажем, что ключи равновероятны:

$$P(Y_j|X_i) = P(y^n|x^n) = P(k_1 = y_1 \oplus x_1, \dots, k_n = y_n \oplus x_n) = P(k_1, k_2, \dots, k_n) = 2^{-n}$$

В выражении использовано предположение о равновероятности ключей.

Найдем $P(Y_j)$. По формуле полной вероятности:

$$P(Y_j) = \sum_{i=1}^{2^n} P(X_i)P(Y_j|X_i). \text{ Учитывая, что } P(Y_j|X_i) = 2^{-n}, \text{ получаем}$$

$$P(Y_j) = 2^{-n} \sum_{i=1}^{2^n} P(X_i) = 2^{-n}, \text{ при } \sum_{i=1}^{2^n} P(X_i) = 1$$

5. Код программы

Заголовочный файл VernamCipher.h:

```
#pragma once
#ifndef VernamCipher_VernamCipher_H
#define VernamCipher_VernamCipher_H

#include <string>
#include <fstream>
#include <time.h>

// Получение зашифрованной последовательности
std::string GetVernamEncryptedText(std::string key, std::string sourceText, std::string alphabet);

// Получение расшифрованной последовательности
std::string GetTheVernamDecryptedText(std::string key, std::string encryptedText, std::string alphabet);

// Генерация случайного ключа шифрования
std::string GeneratingKey(int textLength);

// Получить содержимое файла файла
std::string GetTextFromFile(std::string fileName);

// Записать текст в файл
void WriteTextToFile(std::string fileName, std::string text);

#endif
```

Файл VernamCipher.cpp

```
#include <iostream>
#include "VernamCipher.h"

// Получение зашифрованной последовательности
std::string GetVernamEncryptedText(std::string key, std::string sourceText, std::string alphabet)
{
    std::string encryptedText;

    for (int i = 0; i < sourceText.length(); i++)
    {
        encryptedText += alphabet[(alphabet.find(sourceText[i]) + alphabet.find(key[i])) % alphabet.length()];
    }

    return encryptedText;
}

// Получение расшифрованной последовательности
std::string GetTheVernamDecryptedText(std::string key, std::string encryptedText, std::string alphabet)
{
    std::string decryptedText;
    auto alphabetLength = alphabet.length();

    for (int i = 0; i < encryptedText.length(); i++)
    {
        auto diff = alphabet.find(encryptedText[i]) - alphabet.find(key[i]);
```

```

        decryptedText += diff < 0 ? alphabet[alphabetLength + diff] : alphabet[diff % alphabetLength];
    }

    return decryptedText;
}

// Генерация случайного ключа шифрования
std::string GeneratingKey(int textLength)
{
    const char* alphabet[] = { "*", "-", ",", "_" };

    std::string key;

    for (int i = 0; i < textLength; i++)
    {
        key += static_cast<char>(*alphabet[rand() % (sizeof alphabet / sizeof(char*))]);
    }

    return key;
}

// Получить содержимое файла файла
std::string GetTextFromFile(std::string fileName)
{
    std::ifstream file(fileName);

    std::string result;
    file >> result;
    file.close();

    return result;
}

```

```

// Записать текст в файл
void WriteTextToFile(std::string fileName, std::string text)
{
    std::ofstream file(fileName);
    file << text;
    file.close();
}

```

Файл EncodingForm.h (только обработчики событий):

```

// Нажатие на кнопку "Вставить из файла" для шифруемой последовательности
private: System::Void button8_Click(System::Object^ sender, System::EventArgs^ e)
{
    textBox6->Text = gcnew System::String(GetTextFromFile("OriginalText.txt").c_str());
}

// Нажатие на кнопку "Сгенерировать" для случайной генерации ключа шифрования
private: System::Void button5_Click(System::Object^ sender, System::EventArgs^ e)
{
    srand(time(nullptr));
    textBox4->Text = gcnew System::String(GeneratingKey((textBox6->Text)->Length).c_str());
}

// Нажатие на кнопку "Зашифровать"
private: System::Void button7_Click(System::Object^ sender, System::EventArgs^ e)
{

    // Проверка на равенство длин ключа и сообщения
    if(((textBox6->Text)->Length != (textBox4->Text)->Length))
    {

```

```

    MessageBox::Show("Длина ключа не равна длине сообщения!", "Ошибка");
    return;
}

std::string alphabet = GetTextFromFile("Alphabet.txt");

// Проверка на символ, не принадлежащий алфавиту
for(int i = 0; i < (textBox6->Text)->Length; i++)
{
    if(alphabet.find(msclr::interop::marshal_as<std::string>(textBox6->Text)[i]) ==
    std::string::npos
        || alphabet.find(msclr::interop::marshal_as<std::string>(textBox4->Text)[i]) ==
    std::string::npos)
    {
        MessageBox::Show("Ввод символа, не принадлежащего заданному алфавиту!",
        "Ошибка");
        return;
    }
}

// Запись в файл ключа шифрования, если в чекбоксе галочка
if(checkBox1->Checked)
{
    WriteTextToFile("EncryptionKey.txt", msclr::interop::marshal_as<std::string>(textBox4-
>Text));
}

// Получаем из текстбоксов ключ шифрования и шифруемую последовательность
std::string encryptionKey = msclr::interop::marshal_as<std::string>(textBox4->Text);
std::string originalText = msclr::interop::marshal_as<std::string>(textBox6->Text);

// Запишем в текстбокс Зашифрованную последовательность
textBox5->Text = gcnew System::String(GetVernamEncryptedText(encryptionKey, originalText,
alphabet).c_str());
}

// Нажатие на кнопку "Сохранить в файл" для зашифрованной последовательности
private: System::Void button6_Click(System::Object^ sender, System::EventArgs^ e)
{
    WriteTextToFile("EncryptedText.txt", msclr::interop::marshal_as<std::string>(textBox5->Text));
}

```

Файл DecodingForm.h (только обработчики событий):

```

// Нажатие на кнопку "Вставить из файла" для зашифрованной последовательности
private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e)
{
    textBox2->Text = gcnew System::String(GetTextFromFile("EncryptedText.txt").c_str());
}

// Нажатие на кнопку "Вставить из файла" для ключа шифрования
private: System::Void button4_Click(System::Object^ sender, System::EventArgs^ e)
{
    textBox9->Text = gcnew System::String(GetTextFromFile("EncryptionKey.txt").c_str());
}

// Нажатие на кнопку "Дешифровать"
private: System::Void button2_Click(System::Object^ sender, System::EventArgs^ e)

```

```

{

// Проверка на равенство длин ключа и сообщения
if (((textBox2->Text)->Length != (textBox9->Text)->Length))
{
    MessageBox::Show("Длина ключа не равна длине сообщения!", "Ошибка");
    return;
}

std::string alphabet = GetTextFromFile("Alphabet.txt");

// Проверка на символ, не принадлежащий алфавиту
for (int i = 0; i < (textBox2->Text)->Length; i++)
{
    if (alphabet.find(msclr::interop::marshal_as<std::string>(textBox2->Text)[i]) ==
        std::string::npos
        || alphabet.find(msclr::interop::marshal_as<std::string>(textBox9->Text)[i]) ==
        std::string::npos)
    {
        MessageBox::Show("Ввод символа, не принадлежащего заданному алфавиту!",
        "Ошибка");
        return;
    }
}

// Получаем из текстбоксов зашифрованную последовательность и ключ шифрования
std::string encryptedText = msclr::interop::marshal_as<std::string>(textBox2->Text);
std::string key = msclr::interop::marshal_as<std::string>(textBox9->Text);

// Запишем в текстбокс расшифрованную последовательность
textBox3->Text = gcnew System::String(GetTheVernamDecryptedText(key, encryptedText,
alphabet).c_str());
}

// Нажатие на кнопку "Сохранить в файл" для расшифрованной последовательности
private: System::Void button3_Click(System::Object^ sender, System::EventArgs^ e)
{
    WriteTextToFile("DecryptedText.txt", msclr::interop::marshal_as<std::string>(textBox3->Text));
}

```

Файл MainForm.h (только обработчики событий):

```

private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e)
{
    EncodingForm^ encodingForm = gcnew EncodingForm();
    encodingForm->ShowDialog();
}

private: System::Void button2_Click(System::Object^ sender, System::EventArgs^ e)
{
    DecodingForm^ decodingForm = gcnew DecodingForm();
    decodingForm->ShowDialog();
}

private: System::Void button3_Click(System::Object^ sender, System::EventArgs^ e)
{
    System::Windows::Forms::Application::Exit();
}

```

6. Тесты

№	Ключ	Шифруемая последовательность	Зашифрованная последовательность	Расшифрованная последовательность	Комментарии
1	* __,-	* __,-	** __,-	* __,-	Простой тест
2	****	****	****	****	Ключ и сообщение совпадают
3	-- **- - -	** __-* *	-----	** __-* *	Взятие сообщения из файла и генерация для него случайного ключа
4	- __, * __,-	-,* *	” __,	’ __-	Для шифрования и дешифрования были использованы разные ключи
5	, - * - * -	- * __-,	Сообщение об ошибке		Длина ключа не равна длине сообщения!
6	- ___ ,	***-3,__			Ввод символа, не принадлежащего заданному алфавиту!

Писем № 3. Акравит: * → -

Сообщение M = ** - - **

Ключ K = - - * * - -

1) Присвоим каждой букве акравита числовое значение:

* -
0 1 2 3

2) Для декодирования сообщения столбки по модулю сообщности акравита соответствующие числовые значения ключа и сообщения:

$$\begin{array}{ccccccc} 0(*) & 0(*) & 3(-) & 3(-) & 0(*) & 0(*) & \text{- сообщение} \\ + & & 3(-) & 3(-) & 0(*) & 0(*) & \text{- ключ} \\ \hline & & 3(-) & 3(-) & 3(-) & 3(-) & 3(-) \end{array}$$

3(-) 3(-) 3(-) 3(-) 3(-) 3(-) - (сообщение + ключ) % 4

Зашифрованное сообщение: - - - -

3) Для демодификации сообщений получим соответствующие числовые значения ключа и зашифрованного сообщения:

① Если разность < 0 , то результатом будет символ с числовым значением, равным сумме разностей символов и начальной разности;

② Иначе результатом будет символ символ с числовым значением, равным начальной разности по модулю сообщности акравита.

- 3(-) 3(-) 3(-) 3(-) 3(-) 3(-) - зашифрованное сообщение

- 3(-) 3(-) 0(*) 0(*) 3(-) 3(-) - ключ

0(*) 0(*) 3(-) 3(-) 0(*) 0(*) - (сообщение - ключ) % 4

Расшифрованное сообщение: ** - - * *