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»

Направление 09.03.01 – «Информатика и вычислительная техника»

Дисциплина: «Защита информации»

Профиль: «Разработка программно-информационных систем»

Семестр 5

ОТЧЕТ

по лабораторной работе №8

Тема: «Методы гаммирования»

Выполнил: студент группы РИС 20-2б

Уржумов В.И. \_\_\_\_\_\_\_\_\_\_\_

Проверил: доцент кафедры ИТАС

Шереметьев В. Г. \_\_\_\_\_\_\_\_\_\_\_

Дата \_\_\_\_\_\_

Пермь, 2022

**ЦЕЛЬ РАБОТЫ**

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

**ЗАДАНИЕ**

Реализовать шифрование сообщения методом однократного гаммирования, используя блоки открытого текста длиной 32 бита и используя в алгоритме шифрования операцию сложения.

**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  одинаковой длины, обычно по 64 бита. Гамма шифра вырабатывается в виде последовательности блоков  аналогичной длины.

Уравнение зашифрования можно записать в виде



где  - *i*-й блок шифртекста; - *i*-й блок гаммы шифра; - *i*-й блок открытого текста; *М* - 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.

**ХОД РАБОТЫ**

На рисунке 1 представлена главная форма программы. Пример работы программы представлен на рисунке 2.

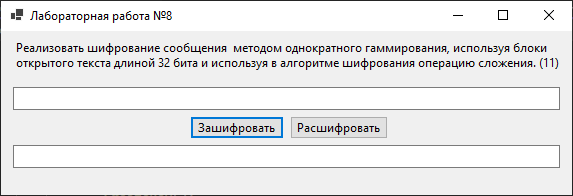
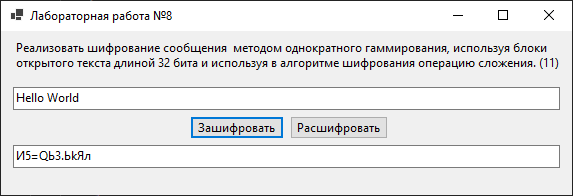


Рисунок 1- Главная форма



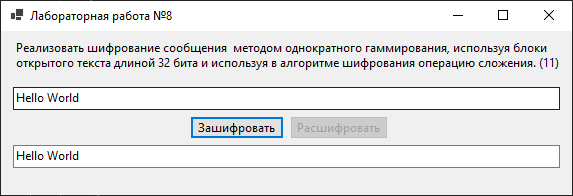


Рисунок 2-Пример работы программы

**ПРИЛОЖЕНИЕ А**

**Листинг класса Labs8**

public partial class Labs8 : Form

{

static char[] alphabet =

{

'0','1','2','3','4','5','6','7','8','9','.',',','?','!','/','\*','-','+','=',' ',

'а','б','в','г','д','е','ё','ж','з','и','й','к','л','м','н','о','п','р','с',

'т','у','ф','х','ц','ч','ш','щ','ъ','ы','ь','э','ю','я',

'А','Б','В','Г','Д','Е','Ё','Ж','З','И','Й','К','Л','М','Н','О','П','Р','С',

'Т','У','Ф','Х','Ц','Ч','Ш','Щ','Ъ','Ы','Ь','Э','Ю','Я',

'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z',

'A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z'

};

int N = alphabet.Length;

string key = "";

public Labs8()

{

InitializeComponent();

}

private void CreateKey()

{

key = "";

Random random = new Random();

for(int i = 0; i < InputText.Text.Length; i++)

key += alphabet[random.Next(0, alphabet.Length)];

}

private void Labs8\_FormClosed(object sender, FormClosedEventArgs e)

{

MainForm mf = new MainForm();

mf.Show();

}

private string Encrypt(string input)

{

string res = "";

int key\_index = 0;

foreach (char symbol in input)

{

int c = (Array.IndexOf(alphabet, symbol) +

Array.IndexOf(alphabet, key[key\_index])) % N;

res += alphabet[c];

key\_index++;

if ((key\_index + 1) == key.Length)

key\_index = 0;

}

return res;

}

private void button2\_Click(object sender, EventArgs e)

{

button2.Enabled = false;

if (outputText.Text != "")

outputText.Text = Decrypt(CheckString(outputText.Text));

}

private string Decrypt(string input)

{

string res = "";

int key\_index = 0;

foreach (char symbol in input)

{

int p = (Array.IndexOf(alphabet, symbol) + N -

Array.IndexOf(alphabet, key[key\_index])) % N;

res += alphabet[p];

key\_index++;

if ((key\_index + 1) == key.Length)

key\_index = 0;

}

return res;

}

private string CheckString(string res)

{

int input = InputText.Text.Length;

while (res.Length != input)

{

res=res.Remove(0, 1);

}

return res;

}

private void button1\_Click(object sender, EventArgs e)

{

if (InputText.Text.Length > 0)

{

CreateKey();

outputText.Text = Encrypt(InputText.Text);

button2.Enabled = true;

}

else

{

MessageBox.Show("Исходное сообщение иммет не верный формат\nиспользуйте русские буквы или цифры");

InputText.Text = "";

}

}

}