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: «Автоматизированные системы обработки информации и управления»

Семестр 7

ОТЧЕТ

по лабораторной работе №8

Тема: «Методы гаммироваиня»

Выполнил: студент группы АСУ-17-1б

Хохряков Денис

Проверил: доцент кафедры ИТАС

Шереметьев В.Г.\_\_\_\_\_\_\_\_\_\_

Дата\_\_\_\_\_\_

# ЦЕЛЬ РАБОТЫ

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

# ЗАДАНИЕ

Реализовать шифрование файла методом однократного гаммирования, используя блоки открытого текста длиной 48 бита и используя в алгоритме шифрования операцию исключающее или (XOR).

# КРАТКИЕ ТЕОРИТИЧЕСКИЕ СВЕДЕНИЯ

Гаммирование – способ шифрования информации, при котором каждый символ текста складываются по модулю с каждым символом «гаммы» - набора случайных символов. Зашифрованный текст абсолютно невозможно расшифровать, не зная гаммы, так как таких способов бесконечно много. Так что метод гаммирования является абсолютно криптостойким.

Но есть один нюанс. Где хранить эти наборы последовательностей? Можно их передавать по защищенному каналу, но тогда отпадает необходимость дополнительного шифрования как таковой. Поэтому метод гаммирования практически никто не использует, хотя он весьма прост.

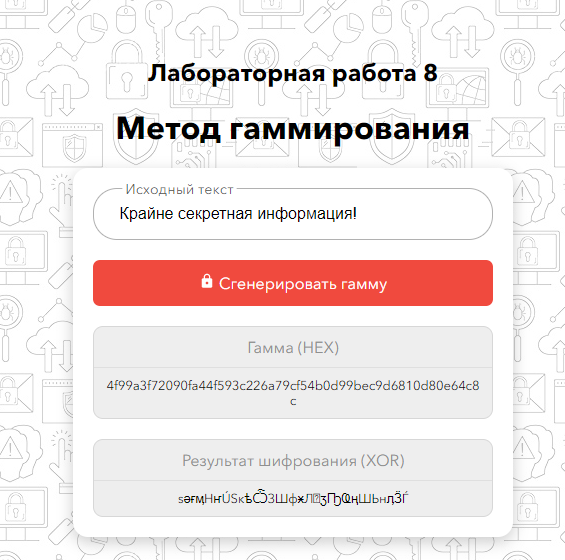
# ХОД РАБОТЫ

1. На основе входной строки генерируем гамму такой же длины, как и строка при помощи датчика случайных чисел.
2. Складываем по модулю посимвольно каждый символ строки с каждым символом гаммы. Если гамма меньше по длине чем строка, то берем символы из гаммы опять из начала. Однако, если гамма меньше, чем исходная строка, то данное шифрование теряет свойство абсолютной криптостойкости
3. При операции гаммирования шифрование и дешифрование – абсолютно идентичные функции. Т.е. для расшифровки текста нужно также посимвольно сложить по модулю зашифрованный текст с гаммой

# РАБОТА ПРОГРАММЫ

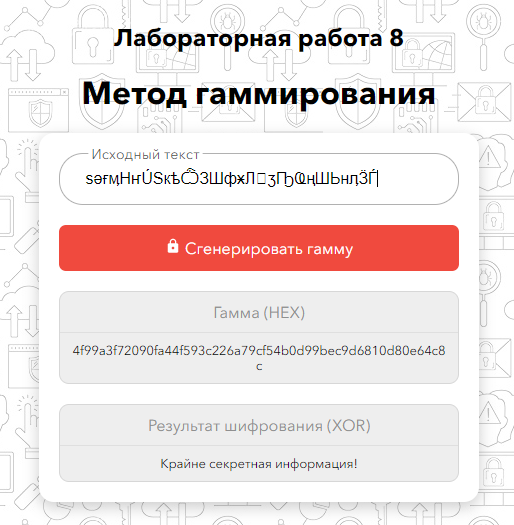
Вводим исходную строку и нажимаем на кнопку «Сгенерировать гамму». Программа генерирует гамму длиной с исходную строку (при этом каждый символ для удобства представлен в 16ричном формате)

Дальше программа на ходу выводит результат шифрования, так как XOR – очень быстрая операция (заложена еще в самом процессоре)



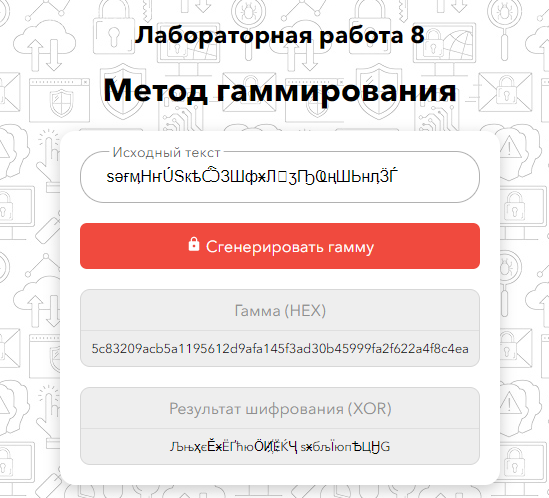
1. Сгенерированная гамма и результат шифрования

Если мы захотим расшифровать данную строку, то нужно будет скопировать полученные кракозябры в поле ввода исходной строки



2. Расшифровка текста

При этом, если мы будем использовать неверную гамму, то получим, по сути, такие же бессмысленные кракозябры.



3. Неправильное расшифрование

# ВЫВОД

Данный алгоритм является самым быстрым из всех существующих алгоритмов шифрования, и при этом обладает абсолютной криптостойкостью – ну разве не чудо? И при этом, как ни странно, его никто не использует. Всему виной его проблемная гамма – каким образом ее передавать, где хранить и т.д. – остается таким же большим вопросом для человечества.

# КОД ПРОГРАММЫ (ФУНКЦИЯ СЖАТИЯ)

# (Язык программирования – JavaScript)

//Функция шифровки текста

const onChange = (obj) => {

const newVal = Object.assign({}, values, obj);

if(newVal.text && newVal.gamma && newVal.gamma.length > 0){

let ciphered = "";

for(let i = 0; i < newVal.text.length; i++){

ciphered = ciphered + String.fromCharCode(newVal.text.charCodeAt(i) ^ newVal.gamma[round(i, newVal.gamma.length)]);

}

newVal.ciphered = ciphered;

}

setValues(newVal);

}

//Функция генерации новой гаммы на основе датчика случайных чисел

const generateGamma = () => {

const str = values.text;

const arr = [];

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

arr.push(\_.random(0, 256, false));

onChange({gamma: arr, gammaStr: arr.map(i => i.toString(16)).join('') });

}