Отчёт по лабораторной работе №3.  
Шифрование гаммированием

Студент: Аронова Юлия Вадимовна, 1032212303

Группа: НФИмд-01-21

Преподаватель: Кулябов Дмитрий Сергеевич,

д-р.ф.-м.н., проф.

Москва 2021

Содержание

[1 Цель работы 1](#_Toc88681489)

[2 Задание 1](#_Toc88681490)

[3 Теоретическое введение 1](#_Toc88681491)

[4 Выполнение лабораторной работы 4](#_Toc88681492)

[5 Выводы 5](#_Toc88681493)

[Список литературы 6](#_Toc88681494)

# 1 Цель работы

Целью данной лабораторной работы является ознакомление с методом шифрования гаммированием, а также его последующая программная реализация для случая конечной гаммы.

# 2 Задание

Рассмотреть и реализовать на языке программирования Python алгоритм шифрования гаммированием на основе конечной гаммы.

# 3 Теоретическое введение

*Шифры гаммирования* (или *аддитивные шифры*) являются самыми эффективными с точки зрения стойкости и скорости преобразований (процедур зашифрования и дешифрования) [1]. По стойкости данные шифры относятся к классу *совершенных*, т.е. при правильном использовании они заведомо не поддаются вскрытию, а дешифрование секретного сообщения приводит к нескольким осмысленным равновероятным открытым сообщениям.

Гаммирование представляет собой частный случай многоалфавитной подстановки, шифрование в котором осуществляется путем сложения символов исходного текста и ключа по модулю, равному числу букв в алфавите [2]. Такой процесс сложения исходного текста и ключа называется в криптографии *наложением гаммы*.

**Сложение по модулю N**. В 1888 г. француз маркиз де Виари в одной из своих научных статей, посвященных криптографии, доказал, что при замене букв исходного сообщения и ключа на числа справедливы формулы:

где – -ый символ открытого и шифрованного сообщения, – количество символов в алфавите, – -ый символ гаммы (ключа) [1].

Так, пусть символам исходного алфавита соответствуют числа от 0 (А) до 31 (Я), как показано в Табл. 1.

Table 1: Таблица кодирования символов русского алфавита

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| А | 0 |  | И | 8 |  | Р | 16 |  | Ш | 24 |
| Б | 1 |  | Й | 9 |  | С | 17 |  | Щ | 25 |
| В | 2 |  | К | 10 |  | Т | 18 |  | Ъ | 26 |
| Г | 3 |  | Л | 11 |  | У | 19 |  | Ы | 27 |
| Д | 4 |  | М | 12 |  | Ф | 20 |  | Ь | 28 |
| Е | 5 |  | Н | 13 |  | Х | 21 |  | Э | 29 |
| Ж | 6 |  | О | 14 |  | Ц | 22 |  | Ю | 30 |
| З | 7 |  | П | 15 |  | Ч | 23 |  | Я | 31 |

*Пример 1*. Зашифуем слово “ПРИКАЗ” гаммой “ГАММА”, используя операцию сложения по модулю 32, и получим криптограмму ТРФЦАК (см. Табл. 2). Так как гамма в данном случае конечна и короче, чем сообщение, то она повторяется требуемое число раз.

Table 2: Пример аддитивного шифрования по модулю 32

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 15 (П) | 16 (Р) | 8 (И) | 10 (К) | 0 (А) | 7 (З) |
|  | 3 (Г) | 0 (А) | 12 (М) | 12 (М) | 0 (А) | 3 (Г) |
|  | 18 (Т) | 16 (Р) | 20 (Ф) | 22 (Ц) | 0 (А) | 10 (К) |

Отметим, что результат шифрования отличается от приведённого в задании к лабораторной работе по двум причинам:

* Нумерация букв в алфавите начинается с 0, а не с 1. Это изменение необходимо, чтобы выполнялось выражение (1). Если же нумеровать символы с единицы, то при нулевом вычете (остатке от деления) у нас не будет символа для сопоставления.
* В качестве модуля сложения используется 32, а не 33, поскольку в используемом алфавите отсутствует буква Ё.

**Сложение по модулю 2**. Наиболее часто на практике встречается двоичное гаммирование [2]. При этом используется двоичный алфавит и сложение по модулю два, обозначаемое знаком . В алгебре логики данная операция также называется *исключающее ИЛИ* или *XOR*.

При данном способе шифрования символы текста и гаммы представляются в двоичном виде, а затем каждая пара двоичных разрядов складывается по модулю 2. Поскольку операция является обратимой (), процедуры шифрования и дешифрования выполняются следующим образом:

Стойкость аддитивных шифров определяется, главным образом, качеством гаммы, которое зависит от длины периода (минимального количества символов, после которого последовательность начинает повторяться) и случайности распределения по периоду. Для обеспечения абсолютной стойкости необходимо, чтобы последовательность символов в пределах периода гаммы обладала следующими свойствами:

* была случайной (должна отсутствовать закономерность в появлении символов гаммы);
* символы алфавита гаммы были распределены нормально (равновероятно);
* совпадала по размеру или была больше исходного открытого текста;
* применялась только один раз.

Так, могут использоваться *истинно случайные гаммы* (полученные путем оцифровки случайных физических или антропогенных процессов) или *псевдослучайные гаммы* – последовательности чисел, вычисленные по определённой процедуре (рекуррентной формуле или полноценному алгоритму), но имеющие все свойства случайной последовательности чисел в рамках решаемой задачи. При этом отсутствие истинной случайности не мешает получать криптографически стойкие последовательности, в том числе и с бесконечным периодом. Схема гаммирования с использованием генератора псевдослучайных чисел показана на Рис. 1.

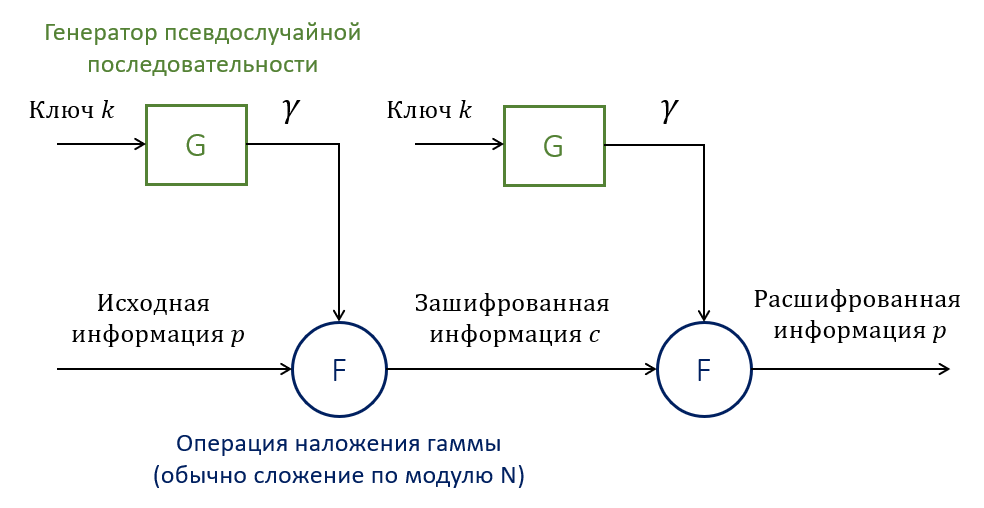


Figure 1: Схема гаммирования с использованием генератора псевдослучайных чисел

В качестве простейшего генератора, например, может использоваться линейный конгруэнтный генератор, в котором каждый новый член последовательности рассчитывается на базе предыдущего через линейную зависимость:

где и – некоторые коэффициенты. Получаемая последовательность периодична с максимальным периодом .

# 4 Выполнение лабораторной работы

Реализуем вышеописанный метод шифрования конечной гаммой на языке **Python** в среде Jupyter Notebook. Создадим список с алфавитом и словарь “буква-порядковый номер” для русского и английского языка, а также реализуем функцию gamma\_cipher(message, key, language):

# русский алфавит  
abc\_rus = [chr(code) for code in range(ord('а'), ord('я') + 1)]  
# английский алфавит  
abc\_eng = [chr(code) for code in range(ord('a'), ord('z') + 1)]  
  
# словарь вида {буква : её порядковый номер в алфавите}  
letter2number\_rus = {abc\_rus[i] : i for i in range(len(abc\_rus))} # (русском)  
letter2number\_eng = {abc\_eng[i] : i for i in range(len(abc\_eng))} # (англ-ом)  
  
abc = {  
 "rus" : abc\_rus,  
 "eng" : abc\_eng  
}  
  
letter2number = {  
 "rus" : letter2number\_rus,  
 "eng" : letter2number\_eng  
}  
  
def gamma\_cipher(message, key, language):  
 """  
 Шифрует сообщение message на языке language конечной гаммой key  
 """  
 mes = message.lower() # приводим сообщение к нижнему регистру  
 n = len(abc[language]) # размерность алфавита  
  
 gamma = key.lower() # приводим гамму к нижнему регистру  
 while len(gamma) < len(mes): # пока она короче сообщения..  
 gamma += gamma[len(gamma) - len(key)] # дополняем её повторениями  
  
 message\_encrypted = "" # криптограмма  
  
 for i in range(len(mes)): # для каждого символа в сообщении..  
 m = letter2number[language][mes[i]] # получаем его порядковый номер  
 # и номер соответствующего символа гаммы  
 g = letter2number[language][gamma[i]]  
  
 # зашифровываем символ и добавляем его к криптограмме  
 message\_encrypted += abc[language][(m + g) % n]  
  
 return message\_encrypted

Теперь с помощью данной функции зашифруем два сообщения: на русском и английском языке (см. Рис. 2). Результат шифрования первого сообщения можно сравнить с примером, описанным ранее (см. Табл. 2), и убедиться, что шифрование произведено корректно.

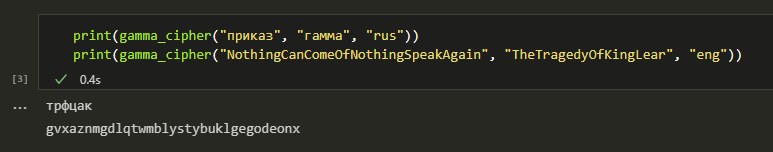


Figure 2: Пример шифрования гаммированием на основе конечной гаммы

# 5 Выводы

Таким образом, была достигнута цель, поставленная в начале лабораторной работы: было проведено краткое знакомство с шифрованием гаммированием, а его вариация для конечной гаммы была успешно реализована на языке программирования **Python**.

# Список литературы

1. Анисимов В.В. Лекции по курсу Криптографические методы защиты информации. Лекция 6. Шифры гаммирования. <https://sites.google.com/site/anisimovkhv/learning/kripto/lecture/tema6>, 2012.

2. Басалова Г. Основы криптографии. Лекция 3. Простейшие методы шифрования с закрытым ключом. Тульский государственный университет <https://intuit.ru/studies/courses/691/547/lecture/12373?page=4>, 2011.