Лабораторная работа №7

Информационная безопасность

Николаев Дмитрий Иванович

Содержание

# 1 Цель работы

Освоить на практике применение режима однократного гаммирования.

# 2 Теоретическое введение

*Гаммирование* представляет собой наложение (снятие) на открытые (зашифрованные) данные последовательности элементов других данных, полученной с помощью некоторого криптографического алгоритма, для получения зашифрованных (открытых) данных. Иными словами, наложение гаммы — это сложение её элементов с элементами открытого (закрытого) текста по некоторому фиксированному модулю, значение которого представляет собой известную часть алгоритма шифрования.

В соответствии с теорией криптоанализа, если в методе шифрования используется однократная вероятностная гамма (однократное гаммирование) той же длины, что и подлежащий сокрытию текст, то текст нельзя раскрыть. Даже при раскрытии части последовательности гаммы нельзя получить информацию о всём скрываемом тексте.

Наложение гаммы по сути представляет собой выполнение операции сложения по модулю 2 (XOR) (обозначаемая знаком ) между элементами гаммы и элементами подлежащего сокрытию текста.

Такой метод шифрования является симметричным, так как двойное прибавление одной и той же величины по модулю 2 восстанавливает исходное значение, а шифрование и расшифрование выполняется одной и той же программой.

Если известны ключ и открытый текст, то задача нахождения шифротекста заключается в применении к каждому символу открытого текста следующего правила:

где — i-й символ получившегося зашифрованного послания, — i-й символ открытого текста, — i-й символ ключа, . Размерности открытого текста и ключа должны совпадать, и полученный шифротекст будет такой же длины.

Если известны шифротекст и открытый текст, то задача нахождения ключа решается также в соответствии с , а именно, обе части равенства необходимо сложить по модулю 2 с :

Открытый текст имеет символьный вид, а ключ — шестнадцатеричное представление. Ключ также можно представить в символьном виде, воспользовавшись таблицей ASCII-кодов.

К. Шеннон доказал абсолютную стойкость шифра в случае, когда однократно используемый ключ, длиной, равной длине исходного сообщения, является фрагментом истинно случайной двоичной последовательности с равномерным законом распределения. Криптоалгоритм не даёт никакой информации об открытом тексте: при известном зашифрованном сообщении все различные ключевые последовательности возможны и равновероятны, а значит, возможны и любые сообщения .

Необходимые и достаточные условия абсолютной стойкости шифра:

* полная случайность ключа;
* равенство длин ключа и открытого текста;
* однократное использование ключа.[1].

# 3 Выполнение лабораторной работы

Подберем ключ, чтобы получить сообщение «С Новым Годом, друзья!». Разработаем приложение, позволяющее шифровать и дешифровать данные в режиме однократного гаммирования. Приложение должно:

1. Определить вид шифротекста при известном ключе и известном открытом тексте.
2. Определить ключ, с помощью которого шифротекст может быть преобразован в некоторый фрагмент текста, представляющий собой один из возможных вариантов прочтения открытого текста.

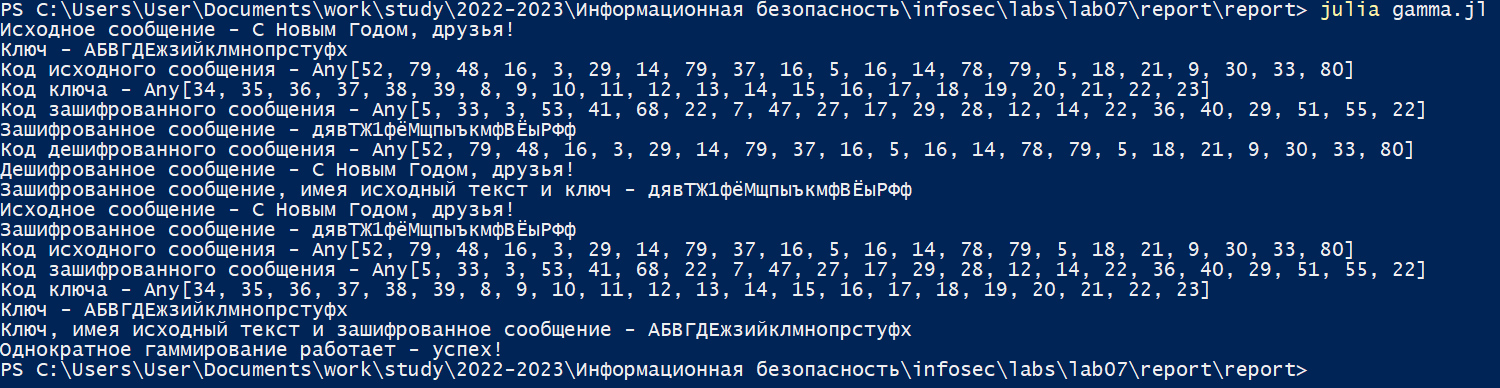
Реализуем приложение с помощью языка Julia, где первая функция будет выдавать вид шифротекста при известном ключе и известном открытом тексте (также проведем проверку проведя дешифровку сообщения), а вторая — вид ключа при известном открытом тексте и шифротексте. Результат работы программы представлен на ([??]).

Так как в программе реализован собственный словарь (длины 81), то рассматривается не операция исключающего ИЛИ, а остатки от деления на длину словаря. Так в случае получения шифротекста вместо имеем где — длина словаря, — код ключа, — код исходного сообщения, — код зашифрованного сообщения; остаток 0 означает последний элемент словаря.

А в случае получения ключа вместо имеем

Ниже представлен код реализации на Julia:

const S = """абвгдеёжзийклмнопрстуфхцчшщъыьэюяАБВГДЕЁЖЗИЙКЛМНОПР  
СТУФХЦЧШЩЪЫЬЭЮЯ0123456789., !-"""  
const N = length(S)  
  
Dictionary = Dict(zip(S, 1:length(S)))  
# Сделаем словарь с ключом и значением наоборот  
Dictionary2 = Dict(zip(values(Dictionary), keys(Dictionary)))  
  
function Gamma\_Find\_Encrypted\_Text(Source\_Message::String, Key::String)::String  
 n = length(Source\_Message) # Длина исходного сообщения  
 println("Исходное сообщение - ", Source\_Message)  
 println("Ключ - ", Key)  
 n != length(Key) ? println("Размерности ключа и сообщения не равны") : skip  
 Source\_Code = []  
 Key\_Code = []  
 for i in Source\_Message  
 push!(Source\_Code, Dictionary[i])  
 end  
 for i in Key  
 push!(Key\_Code, Dictionary[i])  
 end  
 println("Код исходного сообщения - ", Source\_Code)  
 println("Код ключа - ", Key\_Code)  
 Encrypted\_Code = [] # Код зашифрованного сообщения  
 for i in range(1, n)  
 a = Source\_Code[i] + Key\_Code[i]  
 a > N ? a %= N : skip  
 push!(Encrypted\_Code, a)  
 end  
 println("Код зашифрованного сообщения - ", Encrypted\_Code)  
 Encrypted\_Message = ""  
 for i in Encrypted\_Code  
 Encrypted\_Message \*= Dictionary2[i]  
 end  
 println("Зашифрованное сообщение - ", Encrypted\_Message)  
  
 Decrypted\_Code = [] # Код зашифрованного сообщения  
 for i in range(1, n)  
 a = Encrypted\_Code[i] - Key\_Code[i]  
 a <= 0 ? a += N : skip  
 push!(Decrypted\_Code, a)  
 end  
 println("Код дешифрованного сообщения - ", Decrypted\_Code)  
 Decrypted\_Message = ""  
 for i in Decrypted\_Code  
 Decrypted\_Message \*= Dictionary2[i]  
 end  
 println("Дешифрованное сообщение - ", Decrypted\_Message)  
 return Encrypted\_Message  
end  
  
function Gamma\_Find\_Key\_Text(Source\_Message::String, Encrypted\_Message::String)::String  
 n = length(Source\_Message) # Длина исходного сообщения  
 println("Исходное сообщение - ", Source\_Message)  
 println("Зашифрованное сообщение - ", Encrypted\_Message)  
 n != length(Encrypted\_Message) ? println("Несоответсвие размерности исходного и зашифрованного сообщений") : skip  
 Source\_Code = []  
 Encrypted\_Code = []  
 for i in Source\_Message  
 push!(Source\_Code, Dictionary[i])  
 end  
 for i in Encrypted\_Message  
 push!(Encrypted\_Code, Dictionary[i])  
 end  
 println("Код исходного сообщения - ", Source\_Code)  
 println("Код зашифрованного сообщения - ", Encrypted\_Code)  
 Key\_Code = [] # Код ключа  
 for i in range(1, n)  
 a = Encrypted\_Code[i] - Source\_Code[i]  
 a <= 0 ? a += N : skip  
 push!(Key\_Code, a)  
 end  
 println("Код ключа - ", Key\_Code)  
 Key = ""  
 for i in Key\_Code  
 Key \*= Dictionary2[i]  
 end  
 println("Ключ - ", Key)  
 return Key  
end  
  
Source\_Text = "С Новым Годом, друзья!"  
Given\_Key = "АБВГДЕжзийклмнопрстуфх"  
  
Result\_Encrypted\_Message = Gamma\_Find\_Encrypted\_Text(Source\_Text, Given\_Key)   
println("Зашифрованное сообщение, имея исходный текст и ключ - ", Result\_Encrypted\_Message)  
  
Result\_Key = Gamma\_Find\_Key\_Text(Source\_Text, Result\_Encrypted\_Message)  
println("Ключ, имея исходный текст и зашифрованное сообщение - ", Result\_Key)  
  
if Given\_Key == Result\_Key  
 println("Однократное гаммирование работает - успех!")  
else  
 println("Неудача")  
end



Реализация однократного гаммирования

# 4 Ответы на вопросы

1. Поясните смысл однократного гаммирования.

* Гаммирование — выполнение операции XOR (исключающее ИЛИ) между элементами гаммы и элементами подлежащего сокрытию текста. Если в методе шифрования используется однократная вероятностная гамма (однократное гаммирование) той же длины, что и подлежащий сокрытию текст, то текст нельзя раскрыть, так как даже при раскрытии части последовательности гаммы нельзя получить информацию о всём скрываемом тексте.

1. Перечислите недостатки однократного гаммирования.

* Абсолютная стойкость шифра доказана только для случаев, где:
  + однократно используемый ключ;
  + обладает длиной, равной длине исходного сообщения;
  + является фрагментом истинно случайной двоичной последовательности с равномерным законом распределения.

1. Перечислите преимущества однократного гаммирования.
   * Такой способ симметричен, т.е. двойное прибавление одной и той же величины по модулю 2 восстанавливает исходное значение.
   * Шифрование и расшифрование может быть выполнено одной и той же программой.
   * Криптоалгоритм не даёт никакой информации об открытом тексте: при известном зашифрованном сообщении все различные ключевые последовательности возможны и равновероятны, а значит, возможны и любые сообщения .
2. Почему длина открытого текста должна совпадать с длиной ключа?

* Если ключ короче текста, то операция XOR будет применена не ко всем элементам и конец сообщения будет не закодирован. Если ключ будет длиннее, то появится неоднозначность декодирования.

1. Какая операция используется в режиме однократного гаммирования, назовите её особенности?

* Наложение гаммы по сути представляет собой выполнение операции исключающего ИЛИ (XOR), т.е. мы должны сложить каждый элемент гаммы с соответствующим элементом ключа. Данная операция является симметричной, так как прибавление одной и той же величины по модулю 2 восстанавливает исходное значение.

1. Как по открытому тексту и ключу получить шифротекст?

* В таком случае задача сводится к правилу: , т.е. мы поэлементно получаем символы зашифрованного сообщения, применяя операцию исключающего или к соответствующим элементам ключа и открытого текста.

1. Как по открытому тексту и шифротексту получить ключ?

* Подобная задача решается путем применения операции исключающего или к последовательностям символов зашифрованного и открытого сообщений: .

1. В чем заключаются необходимые и достаточные условия абсолютной стойкости шифра?

* Необходимые и достаточные условия абсолютной стойкости шифра:
  + полная случайность ключа;
  + равенство длин ключа и открытого текста;
  + однократное использование ключа.

# 5 Выводы

В ходе выполнения лабораторной работы я освоил на практике применение режима однократного гаммирования.

# Список литературы

1. Кулябов Д. С., Королькова А. В., Геворкян М. Н Лабораторная работа №7 [Электронный ресурс]. RUDN, 2023. URL: <https://esystem.rudn.ru/pluginfile.php/2090212/mod_resource/content/2/007-lab_crypto-gamma.pdf>.