Отчёт по лабораторной работе №8

Шифр гаммирования

Савченко Елизавета НБИ-01-20

Содержание

[1 Цель работы 1](#_Toc149414470)

[2 Теоретические сведения 1](#_Toc149414471)

[2.1 Шифр гаммирования 1](#_Toc149414472)

[2.2 Идея взлома 2](#_Toc149414473)

[3 Выполнение работы 3](#_Toc149414474)

[3.1 Реализация взломщика, шифратора и дешифратора на Python 3](#_Toc149414475)

[3.2 Контрольный пример 3](#_Toc149414476)

[4 Выводы 4](#_Toc149414477)

[Список литературы 4](#_Toc149414478)

# 1 Цель работы

Освоить на практике применение режима однократного гаммирования на примере кодирования различных исходных текстов одним ключом.

# 2 Теоретические сведения

## 2.1 Шифр гаммирования

Гаммирование – это наложение (снятие) на открытые (зашифрованные) данные криптографической гаммы, т.е. последовательности элементов данных, вырабатываемых с помощью некоторого криптографического алгоритма, для получения зашифрованных (открытых) данных.

Принцип шифрования гаммированием заключается в генерации гаммы шифра с помощью датчика псевдослучайных чисел и наложении полученной гаммы шифра на открытые данные обратимым образом (например, используя операцию сложения по модулю 2). Процесс дешифрования сводится к повторной генерации гаммы шифра при известном ключе и наложении такой же гаммы на зашифрованные данные. Полученный зашифрованный текст является достаточно трудным для раскрытия в том случае, если гамма шифра не содержит повторяющихся битовых последовательностей и изменяется случайным образом для каждого шифруемого слова. Если период гаммы превышает длину всего зашифрованного текста и неизвестна никакая часть исходного текста, то шифр можно раскрыть только прямым перебором (подбором ключа). В этом случае криптостойкость определяется размером ключа.

Метод гаммирования становится бессильным, если известен фрагмент исходного текста и соответствующая ему шифрограмма. В этом случае простым вычитанием по модулю 2 получается отрезок псевдослучайной последовательности и по нему восстанавливается вся эта последовательность.

Метод гаммирования с обратной связью заключается в том, что для получения сегмента гаммы используется контрольная сумма определенного участка шифруемых данных. Например, если рассматривать гамму шифра как объединение непересекающихся множеств H(j), то процесс шифрования можно пердставить следующими шагами:

1. Генерация сегмента гаммы H(1) и наложение его на соответствующий участок шифруемых данных.
2. Подсчет контрольной суммы участка, соответствующего сегменту гаммы H(1).
3. Генерация с учетом контрольной суммы уже зашифрованного участка данных следующего сегмента гамм H(2).
4. Подсчет контрольной суммы участка данных, соответствующего сегменту данных H(2) и т.д.

## 2.2 Идея взлома

Шифротексты обеих телеграмм можно получить по формулам режима однократного гаммирования:

Открытый текст можно найти, зная шифротекст двух телеграмм, зашифрованных одним ключом. Для это оба равенства складываются по модулю 2. Тогда с учётом свойства операции XOR получаем:

Предположим, что одна из телеграмм является шаблоном — т.е. имеет текст фиксированный формат, в который вписываются значения полей. Допустим, что злоумышленнику этот формат известен. Тогда он получает достаточно много пар (известен вид обеих шифровок). Тогда зная имеем:

Таким образом, злоумышленник получает возможность определить те символы сообщения , которые находятся на позициях известного шаблона сообщения . В соответствии с логикой сообщения , злоумышленник имеет реальный шанс узнать ещё некоторое количество символов сообщения . Затем вновь используется равенство с подстановкой вместо полученных на предыдущем шаге новых символов сообщения . И так далее. Действуя подобным образом, злоумышленник даже если не прочитает оба сообщения, то значительно уменьшит пространство их поиска.

# 3 Выполнение работы

## 3.1 Реализация взломщика, шифратора и дешифратора на Python

import string  
import random  
  
def hexx(text):  
 return ' '.join(hex(ord(i))[2:] for i in text)  
def gen\_key(size):  
 return ''.join(random.choice(string.ascii\_letters+string.digits) for \_ in range(size))  
def encrypted(firstText,secondText):  
 first\_text=[ord(i) for i in firstText]  
 second\_text=[ord(i) for i in secondText]  
 return ''.join(chr (a^b) for a,b in zip(first\_text,second\_text))  
  
p1 = "Восходящийот1222"  
p2 = "Верныйфайл"  
  
key=gen\_key(len(p1))  
print(key)  
hex\_key=hexx(key)  
print("Ключ в шестнадцатиричном виде: ",hex\_key)  
  
c1 = encrypted(p1,key)  
c2 = encrypted(p2,key)  
  
print("Зашифрованный текст: ",c1)  
print("Зашифрованный текст: ",c2)  
  
decrypt=encrypted(c1,c2)  
print("Расшифрованный текст: ",encrypted(decrypt,p2) )  
print("Расшифрованный текст: ",encrypted(decrypt,p1) )

## 3.2 Контрольный пример

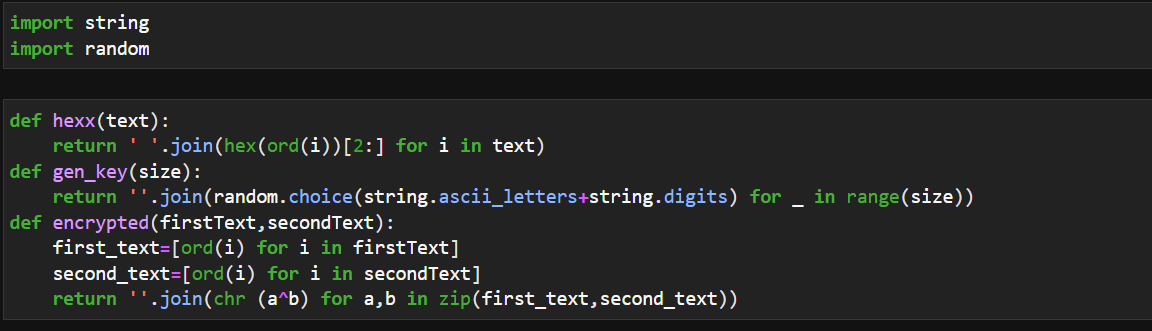


Figure 1: алгоритм работы

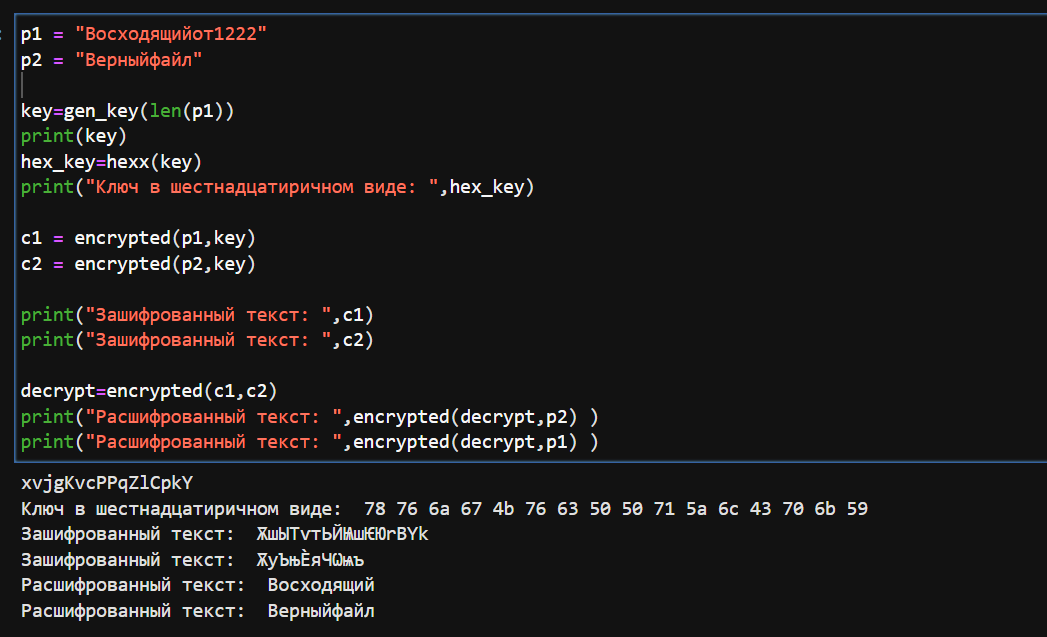


Figure 2: алгоритм работы взлома ключа

# 4 Выводы

В ходе выполнения лабораторной работы было разработано приложение, позволяющее шифровать тексты в режиме однократного гаммирования.

# Список литературы

1. [Шифрование методом гаммирования](http://altaev-aa.narod.ru/security/XOR.html)
2. [Режим гаммирования в блочном алгоритме шифрования](https://kabinfo.ucoz.ru/index/shifr_reshetka_kardano/0-374)