Отчёт по лабораторной работе №8

Шифр гаммирования

Эттеев Сулейман

Содержание

[1 Цель работы 1](#_Toc151367703)

[2 Теоретические сведения 1](#_Toc151367704)

[2.1 Шифр гаммирования 1](#_Toc151367705)

[2.2 Идея взлома 2](#_Toc151367706)

[3 Выполнение работы 3](#_Toc151367707)

[3.1 Реализация взломщика, шифратора и дешифратора на Python 3](#_Toc151367708)

[3.2 Контрольный пример 5](#_Toc151367709)

[4 Выводы 5](#_Toc151367710)

[Список литературы 5](#_Toc151367711)

# 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

# создаем алфавит из русских букв и цифр  
# он нужен для гаммирования  
a = ord("а")  
alphabeth = [chr(i) for i in range(a, a + 32)]  
a = ord("0")  
for i in range(a, a+10):  
 alphabeth.append(chr(i))  
  
a = ord("А")  
for i in range(1040, 1072):  
 alphabeth.append(chr(i))  
print(alphabeth)  
P1 = "НаВашисходящийот1204"  
P2 = "ВСеверныйфилиалБанка"  
# длина ключа 20  
key = "05 0C 17 7F 0E 4E 37 D2 94 10 09 2E 22 57 FF C8 0B B2 70 54"  
  
  
def vzlom(P1, P2):  
 code = []  
 for i in range(20):  
 code.append(alphabeth[(alphabeth.index(P1[i]) + alphabeth.index(P2[i])) % len(alphabeth)])  
 #получили известные символы в шаблоне  
 print(code)  
 print(code[16], " и ", code[19])  
 p3 = "".join(code)  
 print(p3)  
  
vzlom(P1, P2)

def shifr(P1):  
 # создаем алфавит  
 dicts = {"а": 1, "б": 2, "в": 3, "г": 4, "д": 5, "е": 6, "ё": 7, "ж": 8, "з": 9, "и": 10, "й": 11, "к": 12, "л": 13,  
 "м": 14, "н": 15, "о": 16, "п": 17,  
 "р": 18, "с": 19, "т": 20, "у": 21, "ф": 22, "х": 23, "ц": 24, "ч": 25, "ш": 26, "щ": 27, "ъ": 28,  
 "ы": 29, "ь": 30, "э": 31, "ю": 32, "я": 32, "А":33 , "Б": 34, "В": 35 , "Г":36 , "Д":37 , "Е":38 , "Ё":39 , "Ж":40 , "З":41 ,  
 "И":42,"Й":43 , "К":44 , "Л":45 , "М":46 , "Н":47 , "О":48 , "П":49 , "Р":50 , "С":51 , "Т":52 , "У":53 , "Ф":54 , "Х":55 , "Ц":56 , "Ч":57 ,  
 "Ш":58,"Щ":59 , "Ъ":60 , "Ы":61 , "Ь":62 , "Э":63 , "Ю":64 , "Я":65 , "1":66 , "2":67 , "3":68 , "4":69 , "5":70 , "6":71 , "7": 72, "8":73 , "9":74 , "0":75  
 }  
 # меняем местами ключ и значение, такой словарь понадобится в будущем  
 dict2 = {v: k for k, v in dicts.items()}  
 text = P1  
 gamma = input("Введите гамму(на русском языке! Да и пробелы тоже нельзя! Короче, только символы из dict")  
 listofdigitsoftext = list() # сюда будем записывать числа букв из текста  
 listofdigitsofgamma = list() # для гаммы  
 # запишем числа в список  
 for i in text:  
 listofdigitsoftext.append(dicts[i])  
 print("Числа текста", listofdigitsoftext)  
 # то же самое сделаем с гаммой  
 for i in gamma:  
 listofdigitsofgamma.append(dicts[i])  
 print("числа гаммы", listofdigitsofgamma)  
 listofdigitsresult = list() # сюда будем записывать результат  
 ch = 0  
 for i in text:  
 try:  
 a = dicts[i] + listofdigitsofgamma[ch]  
 except:  
 ch = 0  
 a = dicts[i] + listofdigitsofgamma[ch]  
 if a > 75:  
 a = a%75  
 print(a)  
 ch += 1  
 listofdigitsresult.append(a)  
 print("Числа зашифрованного текста", listofdigitsresult)  
 # теперь обратно числа представим в виде букв  
 textencrypted = ""  
 for i in listofdigitsresult:  
 textencrypted += dict2[i]  
 print("Зашифрованный текст: ", textencrypted)  
 # теперь приступим к реализации алгоритма дешифровки  
 listofdigits = list()  
 for i in textencrypted:  
 listofdigits.append(dicts[i])  
 ch = 0  
 listofdigits1 = list()  
 for i in listofdigits:  
 try:  
 a = i - listofdigitsofgamma[ch]  
 except:  
 ch=0  
 a = i - listofdigitsofgamma[ch]  
 if a < 1:  
 a = 75 + a  
 listofdigits1.append(a)  
 ch += 1  
 textdecrypted = ""  
 for i in listofdigits1:  
 textdecrypted += dict2[i]  
 print("Расшифрованный текст", textdecrypted)  
  
shifr(P1)

## 3.2 Контрольный пример

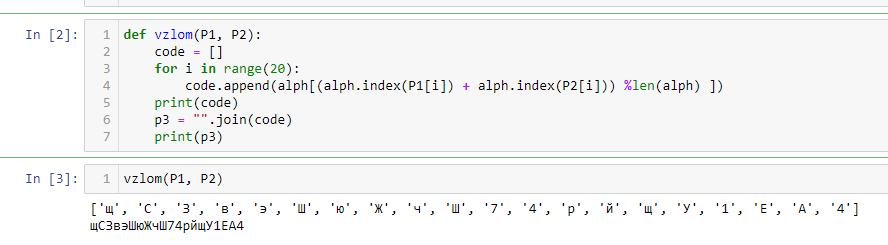


Figure 1: Работа алгоритма взлома ключа

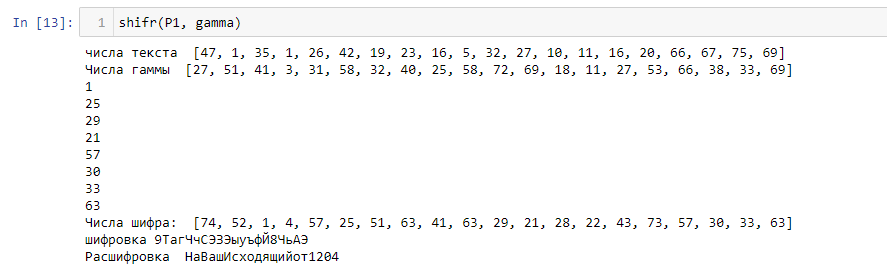


Figure 2: Работа алгоритма шифрования и дешивровки

# 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)