Отчёт по лабораторной работе №7

Шифр гаммирования

Боровиков Даниил Александрович НПИбд-01-22

Содержание

# 1 Цель работы

Изучение алгоритма шифрования гаммированием

# 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.

# 3 Выполнение работы

## 3.1 Реализация шифратора и дешифратора Python

import random  
  
def generate\_key(length):  
 # Генерация случайного ключа  
 return [random.randint(0, 255) for \_ in range(length)]  
  
def int2hex(integer):  
 # Преобразование целого числа в шестнадцатеричное представление  
 return hex(integer)[2:].upper().zfill(2)  
  
def validate(text):  
 # Функция для проверки или корректировки данных  
 return text  
  
def encrypt(text: str, key: list = None):  
 text\_16 = [char.encode(encoding='cp1251').hex().upper() for char in text]  
 if not key:  
 key = generate\_key(length=len(text))  
  
 encrypted\_text = []  
 for i in range(len(text)):  
 xor\_char = int(text\_16[i], 16) ^ key[i]  
 encrypted\_text.append(int2hex(xor\_char))  
  
 encrypted\_text = validate(encrypted\_text)  
 ciphertext = bytes.fromhex(''.join(encrypted\_text)).decode('cp1251')  
  
 return {  
 'key': key,  
 'plaintext': text,  
 'ciphertext': ciphertext  
 }  
  
def decrypt(ciphertext: str, key: list):  
 ciphertext\_16 = [char.encode('cp1251').hex().upper() for char in ciphertext]  
  
 decrypted\_text = []  
 for i in range(len(ciphertext)):  
 xor\_char = int(ciphertext\_16[i], 16) ^ key[i]  
 decrypted\_text.append(int2hex(xor\_char))  
  
 decrypted\_text = validate(decrypted\_text)  
 decrypted\_text = bytes.fromhex(''.join(decrypted\_text)).decode('cp1251')  
  
 return {  
 'key': key,  
 'ciphertext': ciphertext,  
 'plaintext': decrypted\_text  
 }  
  
def find\_key(text):  
 decryption = encrypt(text)  
 decrypted\_text = ''  
 while decrypted\_text != text:  
 decryption = decrypt(decryption['ciphertext'], decryption['key'])  
 decrypted\_text = decryption['plaintext']  
 print(f"Ключ успешно подобран! {decryption['key']}")  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 original\_text = "С Новым Годом, друзья!"  
  
 encryption\_result = encrypt(original\_text)  
 print("Зашифрованный текст:", encryption\_result['ciphertext'])  
 print("Исходный текст:", encryption\_result['plaintext'])  
 print("Ключ шифрования:", encryption\_result['key'])  
  
 decryption\_result = decrypt(encryption\_result['ciphertext'], encryption\_result['key'])  
 print("Расшифрованный текст:", decryption\_result['plaintext'])  
  
 find\_key(original\_text)

## 3.2 Контрольный пример

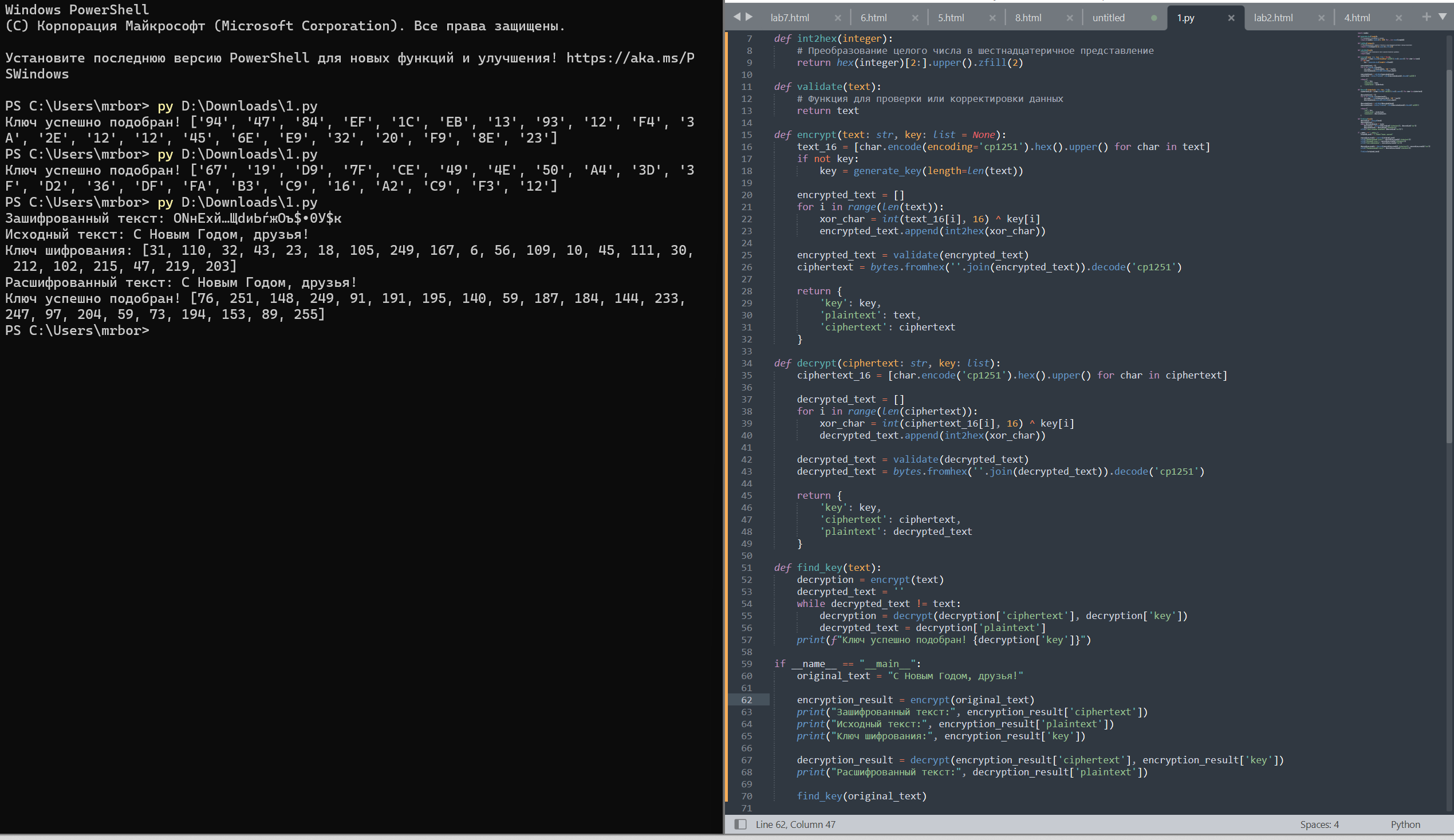


Figure 1: Работа алгоритма гаммирования

# 4 Выводы

Изучили алгоритмы шифрования на основе гаммирования

# 5 Контрольные вопросы

1. Поясните смысл однократного гаммирования Гаммирование – выполнение операции XOR 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. Даже при раскрытии части последовательности гаммы нельзя получить информацию о всём скрываемом тексте.
2. Перечислите недостатки однократного гаммирования Шифр абсолютно стойкий только тогда, когда ключ сгенерирован из случайной двоичной последовательности
3. Перечислите преимущества однократного гаммирования Это симметричный способ шифрования; алгоритм не дает никакой информации об исходном сообщении; шифрование/дешифрование может быть применено одной программой (в обратном порядке)
4. Почему длина открытого текста должна совпадать с длиной ключа? Если ключ длиннее, то часть текста (разница между длиной ключа и открытого текста) не будет зашифрована. Если же ключ короче, то однозначное дешифрование невозможно
5. Какая операция используется в режиме однократного гаммирования, назовите её особенности? операция XOR (сложение по модулю 2), ее особенность - симметричность, т.к. если ее применить 2 раза, то вернется исходное значение
6. Как по открытому тексту и ключу получить шифротекст? Сначала исходный текст и ключ шифрования преобразуются в 16-ную СС, затем, применяется операция XOR для каждого элемента ключа и текста. Полученный шифротекст декодируется из 16-ной СС и получается набор из символов.
7. Как по открытому тексту и шифротексту получить ключ? Применить операцию XOR для каждого элемента шифротекста и открытого текста: key[i] = crypted[i] XOR text[i]
8. 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

# 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)