Лабораторная работа №8

Дисциплина: Информационная безопасность

Губина Ольга Вячеслвовна

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

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

2 Задание

- Разработать приложение, позволяющее шифровать и дешифровать два текста в режиме однократного гаммирования.

3 Теоретическое введение

Гаммирование, или Шифр XOR, — метод симметричного шифрования, заключающийся в «наложении» последовательности, состоящей из случайных чисел, на открытый текст. Последовательность случайных чисел называется гамма-последовательностью и используется для зашифровывания и расшифровывания данных. Суммирование обычно выполняется в каком-либо конечном поле. Например, в поле Галуа GF(2) суммирование принимает вид операции «исключающее ИЛИ (XOR)».[1]

Шифры гаммирования (аддитивные шифры) являются самыми эффективными с точки зрения стойкости и скорости преобразований (процедур зашифрования и дешифрования). По стойкости данные шифры относятся к классу совершенных. Для зашифрования и дешифрования используются элементарные арифметические операции – открытое / зашифрованное сообщение и гамма, представленные в числовом виде, складываются друг с другом по модулю (mod). Напомним, что результатом сложения двух целых чисел по модулю является остаток от деления (например, 5+10 mod 4 = 15 mod 4 = 3).

В литературе шифры этого класса часто называют потоковыми, хотя к потоковым относятся и другие разновидности шифров. В шифрах гаммирования может использоваться сложение по модулю N (общий случай) и по модулю 2 (частный случай, ориентированный на программно-аппаратную реализацию).[2]

В этом способе шифрование выполняется путем сложения символов исходного текста и ключа по модулю, равному числу букв в алфавите. Если в исходном алфавите, например, 33 символа, то сложение производится по модулю 33. Та-

кой процесс сложения исходного текста и ключа называется в криптографии наложением гаммы.

Пусть символам исходного алфавита соответствуют числа от 0 (А) до 32 (Я). Если обозначить число, соответствующее исходному символу, x, а символу ключа – k, то можно записать правило гаммирования следующим образом:

`z = x + k (mod N)`, где `z` – закодированный символ, N - количество символов в алфавите, а сложение по модулю N - операция, аналогичная обычному сложению, с тем отличием, что если обычное суммирование дает результат, больший или равный N, то значением суммы считается остаток от деления его на N. Например, пусть сложим по модулю 33 символы Г (3) и Ю (31):

`3 + 31 (mod 33) = 1`, то есть в результате получаем символ Б, соответствующий числу 1.[3]

`ord()` - встроенная в Python функция. Принимает только один символ(иначе возникнет ошибка) и возвращает целое число - номер из таблицы символов Unicode, представляющий позицию данного символа. Функция имеет свою функцию-антипод `chr()`.[4]

```
ord('a') # 99
ord('h') # 104
```

Функция `chr()` вернет строку, представляющую символ, соответствующий переданному в качестве аргумента целому числу из таблицы символов Unicode. Например, chr(97) возвращает строку a, а `chr(8364)` возвращает строку €. Функция `chr()` - обратная функции `ord()`.[5]

4 Выполнение лабораторной работы

Два текста кодируются одним ключом (однократное гаммирование). Требуется не зная ключа и не стремясь его определить, прочитать оба текста. Необходимо разработать приложение, позволяющее шифровать и дешифровать тексты P_1 и P_2 в режиме однократного гаммирования. Приложение должно определить вид шифротекстов C_1 и C_2 обоих текстов P_1 и P_2 при известном ключе ; Необходимо определить и выразить аналитически способ, при котором злоумышленник может прочитать оба текста, не зная ключа и не стремясь его определить.

Был написан код на языке Python, предоставленный на рис. 4.1 - 4.3.

```
import string # импортируем библиотеки для работы
import random

# задаем сообщения, которые мы хотим закодировать
message_1 = "Хочу плакать(" # P1
message_2 = "Божи поможи((" # P2

key = '' # ключ K пока неизвестен, т.к не подобран

# создаем ключ для кодирования обоих сообщений
for i in range(len(message_1)):
  key = key + random.choice(string.ascii_letters + string.digits)
print("Ключ:", key)
```

Рис. 4.1: Сообщения для кодирования и ключ

```python
# функция кодирования/декодирования текста по ключу
def encryption(text, key):
  en_text = ''
  for i in range(len(key)):
    en_text_symbol_xor = ord(text[i]) ^ ord(key[i])
    en_text = en_text + chr(en_text_symbol_xor)
  return en_text

# кодируем сообщения в соответствие с ключом
encrypted_message_1 = encryption(message_1, key) # C1
encrypted_message_2 = encryption(message_2, key) # C2

print("Зашифрованные тексты:", encrypted_message_1, ",", encrypted_message_2)

# декодируем зашифрованные ранее сообщения
decrypted_message_1 = encryption(encrypted_message_1, key)
decrypted_message_2 = encryption(encrypted_message_2, key)

print("Первое расшифрованное сообщение: ", decrypted_message_1)
print("Второе расшифрованное сообщение: ", decrypted_message_2)
```

```
Ключ: 3oPUlieI2nVsb
Зашифрованные тексты: ЖёЗЖLiўѳJўДпJ , TёѦѫLiћvќjѯ[J
Первое расшифрованное сообщение:  Хочу плакать(
Второе расшифрованное сообщение:  Божи поможи((
```

Рис. 4.2: Функция декодировния и декодирование сообщений с результатом

```python
# не знаем ключ

new_key = encryption(encrypted_message_1, encrypted_message_2)
print("Текст для расшифровки:", new_key)

print("Расшифрованный текст 1:", encryption(new_key, message_2))
print("Расшифрованный текст 2:", encryption(new_key, message_1))
```

```
Текст для расшифровки: 4q{zK
Расшифрованный текст 1: Хочу плакать(
Расшифрованный текст 2: Божи поможи((
```

Рис. 4.3: Декодирование без ключа

На рис. 4.2 видим, что сообщения P_1 и P_2 были раскодированы корректно при использовании созданного ключа. Также мы имеем возможность раскодировать сообщения, не имея ключа вовсе, если применять функцию декодирвоания с вариативными параметрами как показано на рис. 4.3.

5 Выводы

Освоила на практике применение режима однократного гаммирования для дешифровки двух текстов одним ключом.

Список литературы

1. Гаммирование [Электронный ресурс]. 2023. URL: https://ru.wikipedia.org/wiki/%D0%93%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5.
2. Основы шифрования [Электронный ресурс]. 2023. URL: https://sites.google.com/site/anisimovkhv/learning/kripto/lecture/tema6.
3. Простейшие методы шифрования с закрытым ключом [Электронный ресурс]. 2023. URL: https://intuit.ru/studies/mini_mba/5398/courses/547/lecture/12373?page=4.
4. Что делает метод ord python? [Электронный ресурс]. 2023. URL: https://ru.hexlet.io/qna/python/questions/chto-delaet-metod-ord-python.
5. Функция chr() в Python, число в символ Юникода [Электронный ресурс]. 2023. URL: https://docs-python.ru/tutorial/vstroennye-funktsii-interpretatora-python/funktsija-chr/.