Отчет по лабораторной работе №8

Элементы криптографии. Шифрование (кодирование) различных исходных текстов одним ключом

Бурдина Ксения Павловна

2022 Oct 19th

Содержание

Список иллюстраций

1. Цель работы

Целью данной работы является освоение на практике применение режима однократного гаммирования на примере кодирования различных исходных текстов одним ключом.

2. Теоретическое введение

Режим шифрования однократного гаммирования одним ключом двух видов открытого текста реализуется в соответствии со схемой:

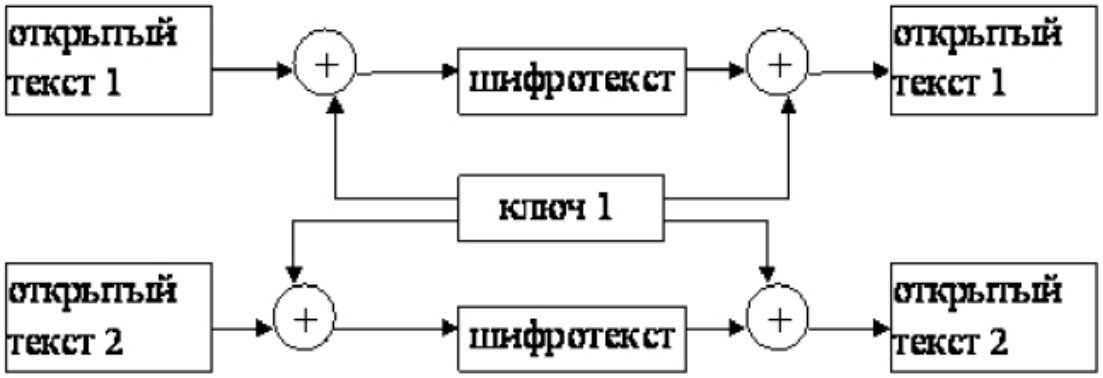

Рис. 2.1.: Схема шифрования одним ключом

Шифротексты обеих телеграмм можно получить по формулам режима однократного гаммирования:

$$C_1 = P_1 \oplus K$$

$$C_2 = P_2 \oplus K$$

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

$$C_1 \oplus C_2 = P_1 \oplus K \oplus P_2 \oplus K = P_1 \oplus P_2$$

Предположим, что одна из телеграмм является шаблоном, т.е. имеет текст фиксированный формат, в который вписываются значения полей [1]. Допустим, что злоумышленнику этот формат известен. Тогда он получает достаточно много пар $C_1 \oplus C_2$ (известен вид обеих шифровок). Тогда зная P_1 и учитывая свойства операции XOR, имеем:

$$C_1 \oplus C_2 \oplus P_1 = P_1 \oplus P_2 \oplus P_1 = P_2$$

Таким образом, злоумышленник получает возможность определить те символы сообщения P_2, которые находятся на позициях известного шаблона сообщения P_1. В соответствии с логикой сообщения P_2, злоумышленник имеет реальный шанс узнать ещё некоторое количество символов сообщения P_2. Затем вновь используется формула с подстановкой вместо P_1 полученных на предыдущем шаге новых символов сообщения P_2. И так далее. Действуя подобным образом, злоумышленник даже если не прочитает оба сообщения, то значительно уменьшит пространство их поиска.

3. Ход выполнения лабораторной работы

Задача ставится следующая: 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.

Для написания приложения будем использовать среду Jupyter. Выполним необходимую задачу.

1. Начнем с того, что введем необходимые нам для работы импорты:

```
import string
import random
```

Рис. 3.1.: Ввод необходимых импортов

2. Далее определим функции, которые мы будем использовать:

```python
def fun_1(text):
    return ' '.join(hex(ord(i))[2:] for i in text)

def fun_2(size):
    return ''.join(random.choice(string.ascii_letters+string.digits) for _ in range(size))

def fun_3(text1, text2):
    text_1 = [ord(i) for i in text1]
    text_2 = [ord(i) for i in text2]
    return ''.join(chr(a^b) for a, b in zip(text_1, text_2))
```

Рис. 3.2.: Функции для работы

Первая функция у нас преобразует символьное представление текста в шестнадцатиричный вид.

Вторая отвечает за назначение каждому символу введенного текста определенного значения из символов Unicode.

А в третьей функции мы прописываем нахождение шифров обоих текстов по одному ключу и далее их расшифровка.

3. Теперь пропишем листинг нашей программы.

Вводим два текста и выводим для проверки их длину (потому что нам требуются тексты с одинаковой длиной для подбора шаблонного ключа):

```python
p1 = "_Звездопад столетия_"
p2 = "Уставший программист"

print(len(p1))
print(len(p2))
```

```
20
20
```

Рис. 3.3.: Ввод текстов и вывод их длины

4. Пропишем нахождение ключа для одного из текстов, чтобы потом использовать его в качестве шаблона:

```
key = fun_2(len(p1))
print("Ключ в символьном виде: ", key)
key_16 = fun_1(key)
print("Ключ в шестнадцатиричном виде: ", key_16)
```

Рис. 3.4.: Нахождение ключа для текста 1

При нахождении ключа используем функцию 2 для составления символьного шифра, после чего преобразуем каждый выявленный символ в шестнадцатиричный вид с помощью функции 1 и выводим рещультат на экран:

```
Ключ в символьном виде:  leXZe5d7I8Jgj2Oj8hys
Ключ в шестнадцатиричном виде:  6c 65 58 5a 65 35 64 37 49 38 4a 67 6a 32 4f 6a 38 68 79 73
```

Рис. 3.5.: Результат нахождения ключа

5. Далее найдем шифр для каждого из текстов:

```
c1 = fun_3(p1, key)
c2 = fun_3(p2, key)
print("Символьный вид текста 1: ", c1)
print("Символьный вид текста 2: ", c2)

c1_16 = fun_1(c1)
c2_16 = fun_1(c2)
print("Текст 1 в виде шифра: ", c1_16)
print("Текст 2 в виде шифра: ", c2_16)
```

Рис. 3.6.: Определение видов шифровок для тектов

Используем функцию 3, которая помогает сделать шифровку для наших текстов. Сначала выводим тексты в символьном виде, после чего преобразуем их в шифр шестнадцатиричного вида и выводим на экран результат:

```
Символьный вид текста 1:  3ΘЖ₴ђЁњJщЌjЦШЌVцOѐж,
Символьный вид текста 2:  яΦKЖїѿќЎiЇћљьΘЩi€ѐиб
Текст 1 в виде шифра:  33 472 46a 46f 452 401 45a 408 479 40c 6a 426 428 40c 474 45f 47a 450 436 2c
Текст 2 в виде шифра:  44f 424 41a 46a 457 47d 45c 40e 69 407 40a 459 459 472 47f 456 404 450 438 431
```

Рис. 3.7.: Результат шифрования текстов

6. Теперь, используя опять же функцию 3, сделаем преобразование шифра в символы необходимого алфавита. При этом заменяем текст сообщения 1 на текст сообщения 2, так как при подстановке шаблона ключа у нас изменился порядок текстов:

```
decr = fun_3(c1, c2)
print("Расшифрованный текст 1: ", fun_3(decr, p2))
print("Расшифрованный текст 2: ", fun_3(decr, p1))
```

Рис. 3.8.: Расшифровка текстов

И в итоге получим наши изначальные тексты:

```
Расшифрованный текст 1:  _Звездопад столетия_
Расшифрованный текст 2:  Уставший программист
```

Рис. 3.9.: Результат расшифровки текстов

4. Листинг программы

```
import string
import random

def fun_1(text):
  return ' '.join(hex(ord(i))[2:] for i in text)
def fun_2(size):
  return ''.join(random.choice(string.ascii_letters+string.digits) for _ in range
def fun_3(text1, text2):
  text_1 = [ord(i) for i in text1]
  text_2 = [ord(i) for i in text2]
  return ''.join(chr(a^b) for a, b in zip(text_1, text_2))

p1 = "_Звездопад столетия_"
p2 = "Уставший программист"
print(len(p1))
print(len(p2))

key = fun_2(len(p1))
print("Ключ в символьном виде: ", key)
key_16 = fun_1(key)
print("Ключ в шестнадцатиричном виде: ", key_16)
```

```
c1 = fun_3(p1, key)
c2 = fun_3(p2, key)
print("Символьный вид текста 1: ", c1)
print("Символьный вид текста 2: ", c2)

c1_16 = fun_1(c1)
c2_16 = fun_1(c2)
print("Текст 1 в виде шифра: ", c1_16)
print("Текст 2 в виде шифра: ", c2_16)

decr = fun_3(c1, c2)
print("Расшифрованный текст 1: ", fun_3(decr, p2))
print("Расшифрованный текст 2: ", fun_3(decr, p1))
```

5. Выводы

В ходе работы мы освоили на практике применение режима однократного гаммирования на примере кодирования различных исходных текстов одним ключом.

6. Контрольные вопросы

1. Как, зная один из текстов (P_1 или P_2), определить другой, не зная при этом ключа?

Необходимо попытаться определить длину ключа, а затем разделить сообщение на несколько частей. Если правильно определить ключ, то перед пользователем будет вариант из определенного количества простых криптограмм с заменой Цезаря, которые нужно будет разгадать.

3. Как реализуется режим шифрования однократного гаммирования одним ключом двух открытых текстов?

Режим шифрования однократного гаммирования одним ключом двух видов открытого текста реализуется в соответствии со схемой:

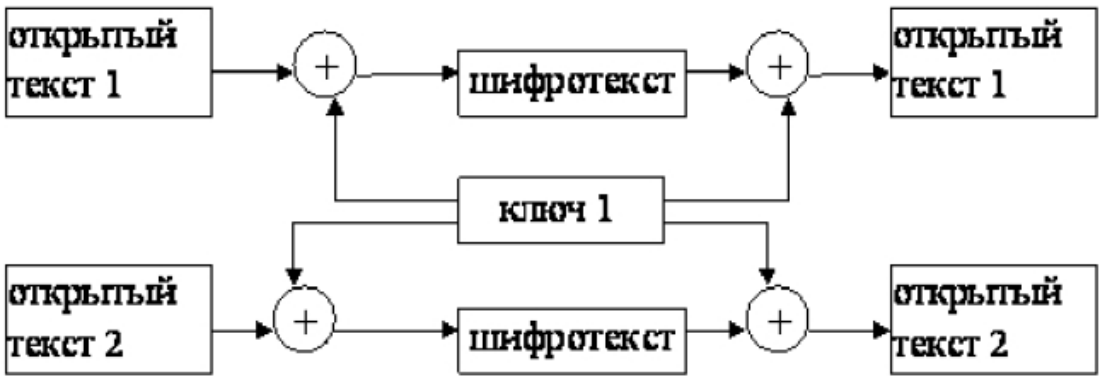

Рис. 6.1.: Схема шифрования одним ключом

Шифротексты обеих телеграмм можно получить по формулам режима однократного гаммирования:

$$C_1 = P_1 \oplus K$$

$$C_2 = P_2 \oplus K$$

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

$$C_1 \oplus C_2 = P_1 \oplus K \oplus P_2 \oplus K = P_1 \oplus P_2$$

4. Перечислите недостатки шифрования одним ключом двух открытых текстов.

Преимущества открытых тектов:

- Не нужно предварительно передавать секретный ключ по надёжному каналу;
- Только одной стороне известен ключ дешифрования, который нужно держать в секрете;
- В больших сетях число ключей в открытой криптосистеме значительно меньше, чем в симметричной.

5. Перечислите преимущества шифрования одним ключом двух открытых текстов.

Недостатки алгоритма открытого шифрования:

- В алгоритм сложнно внести изменения;
- При открытом алгоритме используются порой слишком длинные ключи, которые тяжело подобрать;

- Шифрование-расшифровывание с использованием пары ключей проходит на два-три порядка медленнее, чем шифрование-расшифрование того же текста симметричным алгоритмом;
- Требуются существенно большие вычислительные ресурсы, поэтому на практике открытые криптосистемы нужно использовать в сочетании с другими алгоритмами.