Отчет по лабораторной работе №7

Элементы криптографии. Однократное гаммирование

Бурдина Ксения Павловна

2022 Oct 19th

Содержание

1. **Цель работы** 4
2. **Теоретическое введение** 5
3. **Ход выполнения лабораторной работы** 7
4. **Листинг программы** 11
5. **Выводы** 13
6. **Контрольные вопросы** 14
7. **Список литературы** 18

Список иллюстраций

2.1. Схема однократного гаммирования 5

3.1. Ввод необходимых импортов 7
3.2. Функции для работы 8
3.3. Листинг задания 1 8
3.4. Результат выполнения программы 1 9
3.5. Листинг задания 2 9
3.6. Результат выполнения программы 2 10

1. Цель работы

Целью данной работы является освоение на практике применения режима однократного гаммирования.

2. Теоретическое введение

Гаммирование представляет собой наложение (снятие) на открытые (зашифрованные) данные последовательности элементов других данных, полученной с помощью некоторого криптографического алгоритма, для получения зашифрованных (открытых) данных. Иными словами, наложение гаммы — это сложение её элементов с элементами открытого (закрытого) текста по некоторому фиксированному модулю, значение которого представляет собой известную часть алгоритма шифрования.

В соответствии с теорией криптоанализа, 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 [1].

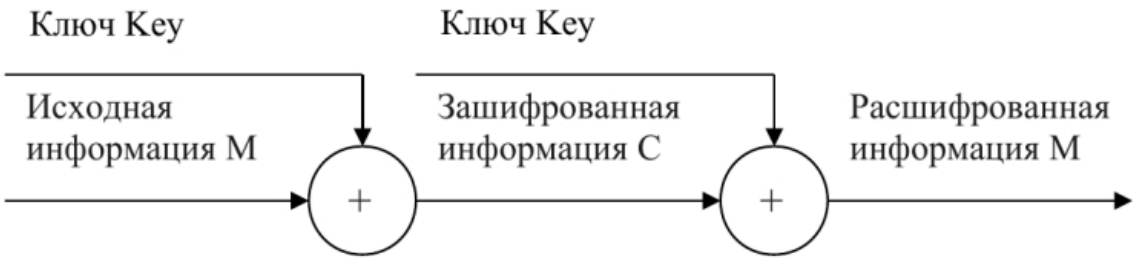

Рис. 2.1.: Схема однократного гаммирования

Наложение гаммы по сути представляет собой выполнение операции сложения по модулю 2 (XOR) (обозначаемая знаком $\oplus$) между элементами гаммы и

элементами подлежащего сокрытию текста. Такой метод шифрования является симметричным, так как двойное прибавление одной и той же величины по модулю 2 восстанавливает исходное значение, а шифрование и расшифрование выполняется одной и той же программой. Если известны ключ и открытый текст, то задача нахождения шифротекста заключается в применении к каждому символу открытого текста следующего правила:

$$C_i = P_i \oplus K_i$$

где C_i - это i-й символ получившегося зашифрованного послания, P_i - i-й символ открытого текста, K_i - i-й символ ключа, $i = \overline{1, m}$. Размерности открытого текста и ключа должны совпадать, и полученный шифротекст будет такой же длины.

Если известны шифротекст и открытый текст, то задача нахождения ключа решается так, что обе части равенства необходимо сложить по модулю 2 с P_i:

$$C_i \oplus P_i = P_i \oplus K_i \oplus P_i = K_i$$

$$K_i = C_i \oplus P_i$$

Открытый текст имеет символьный вид, а ключ — шестнадцатеричное представление. Ключ также можно представить в символьном виде, воспользовавшись таблицей ASCII-кодов.

При известном зашифрованном сообщении C все различные ключевые последовательности K возможны и равновероятны, а значит, возможны и любые сообщения P. Необходимые и достаточные условия абсолютной стойкости шифра: - полная случайность ключа; - равенство длин ключа и открытого текста; - однократное использование ключа.

3. Ход выполнения лабораторной работы

Нам нужно подобрать ключ, чтобы получить сообщение «С Новым Годом, друзья!». Требуется разработать приложение, позволяющее шифровать и дешифровать данные в режиме однократного гаммирования.

Для написания приложения будем использовать среду Jupyter. Выполним необходимые задания.

1. Определим вид шифротекста при известном ключе и известном открытом тексте.

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

```
import string
import random
```

Рис. 3.1.: Ввод необходимых импортов

Далее определим функции, которые мы будем использовать:

```
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(text, key):
    return ''.join(chr(a^b) for a, b in zip(text, key))

def fun_4(text, encr):
    return ''.join(chr(a^b) for a, b in zip(text, encr))
```

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.

С помощью третьей функции мы будем выводить зашифрованный и раскодированный виды нашего текста по имеющемуся ключу.

А четвертая функция нам понадобиться в следующем задании для преобразования шифротекста в один из возможных вариантов расшифровки.

Теперь пропишем листинг нашей программы:

```
message = 'С Новым Годом, друзья!'
print("Исходное сообщение: ", message)

key = fun_2(len(message))
key_16 = fun_1(key)
print("Сгенерированный ключ: ", key)
print("Ключ в шестнадцатиричном виде: ", key_16)

encr = fun_3([ord(i) for i in message], [ord(i) for i in key])
encr_16 = fun_1(encr)
print("Текст в зашифрованном виде: ", encr_16)
decr = fun_3([ord(i) for i in encr], [ord(i) for i in key])
print("Расшифрованное сообщение: ", decr)
```

Рис. 3.3.: Листинг задания 1

Мы вводим имеющееся сообщение. Далее составляем ключ для него с помощью функции 2. Определяем длину нашего сообщения, после чего преобразуем

каждый выявленный символ в шестнадцатиричный вид с помощью функции 1. Далее выводим результат на экран.

Далее, используя функцию 3, сделаем шифровку для нашего текста. Преобразуем ее в шестнадцатиричный вид и выведем на экран полученное зашифрованное сообщение.

После этого, используя опять же функцию 3, сделаем преобразование шифра в символы необходимого алфавита. И в итоге получим наше изначальное сообщение:

```
Исходное сообщение:  С Новым Годом, друзья!
Сгенерированный ключ:  kBs1gtkXKjvGD1MxT7Wwuh
Ключ в шестнадцатиричном виде:  6b 42 73 31 67 74 6b 58 4b 6a 76 47 44 31 4d 78 54 37 57 77 75 68
Текст в зашифрованном виде:  44a 62 46e 40f 455 43f 457 78 458 454 442 479 478 1d 6d 44c 414 474 460 43b 43a 49
Расшифрованное сообщение:  С Новым Годом, друзья!
```

Рис. 3.4.: Результат выполнения программы 1

2. Теперь определим ключ, с помощью которого шифротекст может быть преобразован в некоторый фрагмент текста, представляющий собой один из возможных вариантов прочтения открытого текста.

В данном случае будем использовать нашу функцию 4. С помощью нее, зная шифрованный вид нашего сообщения, попробуем подобрать ключ шифрования в символьном виде. После этого, с помощью функции 3 расшифруем получившийся код в символы используемого алфавита:

```
ident_key = fun_4([ord(i) for i in message], [ord(i) for i in encr])
decr_ident_key = fun_3([ord(i) for i in encr], [ord(i) for i in key])
print("Подобранный ключ: ", ident_key)
print("Вариант прочтения открытого текста: ", decr_ident_key)
```

Рис. 3.5.: Листинг задания 2

В результате при выполнении команд мы получим следующие результаты подобранного ключа и расшифрованного сообщения:

```
Подобранный ключ:  kBs1gtkXKjvGD1MxT7Wwuh
Вариант прочтения открытого текста:  С Новым Годом, друзья!
```

Рис. 3.6.: Результат выполнения программы 2

Видим, что мы получили наше изначальное сообщение с помощью зашифрованных данных и подобранного ключа расшифровки.

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(text, key):
  return ''.join(chr(a^b) for a, b in zip(text, key))
def fun_4(text, encr):
  return ''.join(chr(a^b) for a, b in zip(text, encr))

message = 'С Новым Годом, друзья!'
print("Исходное сообщение: ", message)

key = fun_2(len(message))
key_16 = fun_1(key)
print("Сгенерированный ключ: ", key)
print("Ключ в шестнадцатиричном виде: ", key_16)

encr = fun_3([ord(i) for i in message], [ord(i) for i in key])
encr_16 = fun_1(encr)
```

```
print("Текст в зашифрованном виде: ", encr_16)
decr = fun_3([ord(i) for i in encr], [ord(i) for i in key])
print("Расшифрованное сообщение: ", decr)

ident_key = fun_4([ord(i) for i in message], [ord(i) for i in encr])
decr_ident_key = fun_3([ord(i) for i in encr], [ord(i) for i in key])
print("Подобранный ключ: ", ident_key)
print("Вариант прочтения открытого текста: ", decr_ident_key)
```

5. Выводы

В ходе работы мы освоили на практике применение режима однократного гаммирования.

6. Контрольные вопросы

1. Поясните смысл однократного гаммирования.

В теории криптоанализа метод шифрования однократной случайной равновероятной гаммой той же длины (“однократное гаммирование”), что и открытый текст, является невскрываемым. Обоснование, которое привел Шеннон, основываясь на введенном им же понятии информации, не дает возможности усомниться в этом - из-за равных априорных вероятностей криптоаналитик не может сказать о дешифровке, верна она или нет. Кроме того, даже раскрыв часть сообщения, дешифровщик не сможет поправить положение - информация о вскрытом участке гаммы не дает информации об остальных ее частях. Для организации метода защиты от прослушивания второго рода следовало бы воспользоваться именно схемой шифрования однократного гаммирования.

2. Перечислите недостатки однократного гаммирования.

Главный недостаток - это необходимость иметь огромные объемы данных, которые можно было бы использовать в качестве гаммы. Для этих целей обычно пользуются датчиками настоящих случайных чисел. Статистические характеристики таких наборов весьма близки к характеристикам “белого шума”, что означает равновероятное появление каждого следующего числа в наборе. К сожалению, для того чтобы организовать метод защиты от прослушивания второго рода, воспользовавшись схемой шифрования однократного гаммирования, потребуется записать довольно большое количество этих данных и обменяться ими

по каналу связи. Одно это условие делает однократное гаммирование неприемлемым.

3. Перечислите преимущества однократного гаммирования.

С точки зрения теории криптоанализа, метод шифрования случайной однократной равновероятной гаммой той же длины, что и открытый текст, является невскрываемым. Кроме того, даже раскрыв часть сообщения, дешифровщик не сможет хоть сколько-нибудь поправить положение - информация о вскрытом участке гаммы не дает информации об остальных ее частях.

4. Почему длина открытого текста должна совпадать с длиной ключа?

Например, если длина зашифрованного текста составляет 128 бит (что означает, что открытый текст также равен 128 битам), существует 2 ^ 128 возможных зашифрованных текстов. Следовательно, с точки зрения злоумышленника должно быть 2 ^ 128 возможных открытых текстов. Но если существует менее 2 ^ 128 возможных ключей, злоумышленник может исключить некоторые открытые тексты. Например, если ключ имеет только 64 бита, злоумышленник может определить 264 возможных открытых текста и исключить подавляющее большинство 128-битных строк. Злоумышленник не узнает, что такое открытый текст, но он узнает, чем открытый текст не является, что делает секретность шифрования несовершенной.

5. Какая операция используется в режиме однократного гаммирования? Назовите её особенности.

Наложение гаммы по сути представляет собой выполнение операции сложения по модулю 2 (XOR) (обозначаемая знаком $\oplus$) между элементами гаммы и элементами подлежащего сокрытию текста. Такой метод шифрования является симметричным, так как двойное прибавление одной и той же величины по модулю 2 восстанавливает исходное значение, а шифрование и расшифрование выполняется одной и той же программой. Если известны ключ и открытый

текст, то задача нахождения шифротекста заключается в применении к каждому символу открытого текста следующего правила:

$$C_i = P_i \oplus K_i$$

6. Как по открытому тексту и ключу получить шифротекст?

Для шифрования и дешифрования используются разные ключи. Это свойство, которое отличает эту схему от схемы симметричного шифрования. Каждый получатель обладает уникальным ключом дешифрования и ему необходимо опубликовать ключ шифрования, называемый его открытым ключом. В этой схеме необходима некоторая уверенность в подлинности открытого ключа, чтобы избежать подделки злоумышленником в качестве получателя.

Алгоритм шифрования достаточно сложен, чтобы запретить злоумышленнику выводить открытый текст из зашифрованного текста и открытого ключа шифрования. Хотя закрытый и открытый ключи связаны математически, вычислить закрытый ключ из открытого ключа невозможно. Интеллектуальная часть любой криптосистемы с открытым ключом заключается в разработке взаимосвязи между двумя ключами.

7. Как по открытому тексту и шифротексту получить ключ?

Для получения ключа, зная данные в открытом и шифрованном виде, необходимо определить совпадающие символы по имеющимся. Пространство ключей шифра простой замены огромно и равно количеству перестановок используемого алфавита. Подбор ключа производится с помощью перебора возможных вариантов последовательности. Шифртекст является опорой для подбора ключа по возможным переборам значений из таблици ASCII-кодов, главное, чтобы длина подбираемого ключа совпадала с длиной имеющегося сообщения.

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.

7. Список литературы

1. Методические материалы курса [1]