Лабораторная работа №8

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

Тасыбаева Наталья Сергеевна

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

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 P1 и P2 в режиме однократного гаммирования. Приложение должно определить вид шифротекстов C1 и C2 обоих текстов P1 и P2 при известном ключе ; Необходимо определить и выразить аналитически способ, при котором злоумышленник может прочитать оба текста, не зная ключа и не стремясь его определить.

3 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. Наложение гаммы по сути представляет собой выполнение операции сложения по модулю 2 (XOR) (обозначаемая знаком ⊠) между элементами гаммы и элементами подлежащего сокрытию текста. Напомним, как работает операция XOR над битами:

$$0 \oplus 0 = 0, 0 \oplus 1 = 1, 1 \oplus 0 = 1, 1 \oplus 1 = 0.$$

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[**1?**].

4 Выполнение лабораторной работы

Написала программу, реализующее задание (рис. 4.1, 4.2, 4.3).

```python
def encode_val(word):
    d = form_dict()
    return [k for c in word for k, v in d.items() if v == c]
# Разбираем слово и ключ из ASCII
4 usages
def decode_val(value):
    d = form_dict()
    decode_v = ''
    for i in value:
        decode_v = decode_v + d[i]
    return decode_v
1 usage
def comparator(value, key):
    return dict([(index, list(character))
                 for index, character in enumerate(zip(value, itertools.cycle(key)))])
# сложение по модулю
2 usages
def full_encode(value, key):
    d = comparator(value, key)
    l = len(form_dict())
    return [(v[0] ^ v[1]) % l for v in d.values()]

2 usages
def full_encode3_text(C1, C2, Pn):
    ans = []
    for i in range(len(Pn)):
        ans.append((Pn[i] ^ C1[i] ^ C2[i]))
    return ans
```

Рис. 4.1: Название рисунка

```
P1 = 'S novim godom, druzia!'
P2 = 'Schactlivogo Rozdestva'
key = 'pvnquervjttjvhkunuziiu'
encode_p1 = encode_val(P1)
encode_p2 = encode_val(P2)
# кодируем начальные тексты с помощью ключа
encode_c1 = full_encode(encode_p1, encode_val(key))
encode_c2 = full_encode(encode_p2, encode_val(key))
# переводим в текстовый вид
C1 = decode_val(encode_c1)
C2 = decode_val(encode_c2)
print('P1: ', "\n", P1, "; len:", len(P1))
print('C1: ', "\n", C1, "; len:", len(C1))
print('P2: ', "\n", P2, "; len:", len(P2))
print('C2: ', "\n", C2, "; len:", len(C2))
# декодируем сообщение без ключа
print("декодируем сообщение без ключа")
P1_decoded_no_key = full_encode3_text(encode_c1, encode_c2, encode_p2)
P2_decoded_no_key = full_encode3_text(encode_c1, encode_c2, encode_p1)
# переводим в текстовый вид
P1_1 = decode_val(P1_decoded_no_key)
P2_1 = decode_val(P2_decoded_no_key)
print('P1_1: ', "\n", P1_1, "; len:", len(P1_1))
print('P2_1: ', "\n", P2_1, "; len:", len(P2_1))
```

Рис. 4.2: Название рисунка

```
C:\Users\maksi\PycharmProjects\lab7_infoBez\venv\Scripts\python.exe C:\Users\maksi\PycharmProjects\lab7_infoBez\main.py
P1:
 S novim godom, druzia! ; len: 22
C1:
[illegible] ; len: 22
P2:
 Schactlivogo Rozdestva ; len: 22
C2:
[illegible] ; len: 22
декодируем сообщение без ключа
P1_1:
 S novim godom, druzia! ; len: 22
P2_1:
 Schactlivogo Rozdestva ; len: 22

Process finished with exit code 0
```

Рис. 4.3: Название рисунка

5 Выводы

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

6 Ответы на контрольные вопросы

Однократное гаммирование (также известное как шифр Вернама или шифр по модулю 2) является методом шифрования, который использует одноразовый ключ, который должен быть такой же длины, что и шифруемый текст. Ответим на ваши вопросы:

1. Как, зная один из текстов (P1 или P2), определить другой, не зная при этом ключа?

Если вы знаете один из открытых текстов (P1 или P2) и соответствующий ему зашифрованный текст (C1 или C2), и у вас нет ключа, вы не сможете определить другой открытый текст. Однократное гаммирование обеспечивает сильную криптографическую защиту, если ключ используется только один раз.

2. Что будет при повторном использовании ключа при шифровании текста?

Повторное использование ключа при однократном гаммировании полностью компрометирует шифрование. Если ключ используется повторно, злоумышленный анализатор может просто выполнить операцию XOR между зашифрованным текстом и известным текстом, чтобы получить оригинальный ключ, и затем использовать этот ключ для расшифровки других сообщений, зашифрованных с тем же ключом.

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

В однократном гаммировании каждому биту открытого текста соответствует бит ключа. Операция XOR выполняется между открытым текстом и ключом, чтобы получить зашифрованный текст. Если вы хотите зашифровать два разных открытых текста (P1 и P2) с использованием одного и того же ключа, вы просто выполняете операцию XOR между каждым открытым текстом и ключом по отдельности.

4. Перечислите недостатки шифрования одним ключом двух открытых текстов.

Основными недостатками шифрования одним ключом двух открытых текстов (повторное использование ключа) являются: - Уязвимость к криптоанализу при многократном использовании ключа. - Невозможность безопасного обмена ключами для секретной передачи. - Ограничение на длину ключа, который должен быть такой же длины, что и открытый текст.

5. Перечислите преимущества шифрования одним ключом двух открытых текстов.

Преимущества такого шифрования ограничены и обычно не рекомендуются. Однако можно выделить следующие преимущества: Простота реализации: Однократное гаммирование легко реализовать с использованием операции XOR. Высокая скорость шифрования и дешифрования: Поскольку операция XOR быстрая, шифрование и дешифрование могут выполняться быстро.

Список литературы