

Отчёт по лабораторной работе №8

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

Ким Реачна

Содержание

1	Цель работы	4
2	Теоретические сведения	5
2.1	Шифр гаммирования	5
2.2	Идея взлома	6
3	Выполнение работы	8
3.1	Реализация взломщика, шифратора и дешифратора на Python . .	8
3.2	Контрольный пример	11
4	Выводы	13
	Список литературы	14

Список иллюстраций

3.1	Работа алгоритма взлома ключа	11
3.2	Работа алгоритма шифрования и дешифровки	12

1 Цель работы

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

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 представить 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 гаммы $H(2)$.
4. Подсчет контрольной суммы участка данных, соответствующего сегменту данных $H(2)$ и т.д.

2.2 Идея взлома

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 это оба равенства складываются по модулю 2. Тогда с учётом свойства 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. 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 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 равенство 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 Выполнение работы

3.1 Реализация взломщика, шифратора и дешифратора на Python

```
a = ord("a")
litters = [chr(i) for i in range(a, a + 32)]
a = ord("0")
for i in range(a, a + 10):
    litters.append(chr(i))

a = ord("A")
for i in range(1040, 1104):
    litters.append(chr(i))

P1 = "КодофаяФраза1"
P2 = "Безопасность2"

def vzlom(P1, P2):
    code = []
    for i in range(len(P1)):
        code.append(litters[(litters.index(P1[i]) + litters.index(P2[i]))
                           % len(litters)])
    print(code)
```



```
pr = "".join(code)
print(pr)
```

```
def shifr(P1, gamma):
```

```
dicts = {"a": 1, "б": 2, "в": 3, "г": 4, "д": 5, "е": 6, "ё": 7, "ж": 8,
        "з": 9, "и": 10, "й": 11, "к": 12, "л": 13, "м": 14, "н": 15,
        "о": 16, "п": 17, "р": 18, "с": 19, "т": 20, "у": 21, "ф": 22,
        "х": 23, "ц": 24, "ч": 25, "ш": 26, "щ": 27, "ъ": 28, "ы": 29,
        "ь": 30, "э": 31, "ю": 32, "я": 32, "А":33 , "Б": 34, "В": 35 ,
        "Г":36, "Д":37 , "Е":38 , "Ё":39 , "Ж":40 , "З":41, "И":42,"Й":43 ,
        "К":44 , "Л":45 , "М":46 , "Н":47 , "О":48 , "П":49 , "Р":50 ,
        "С":51 , "Т":52 , "У":53 , "Ф":54 , "Х":55 , "Ц":56 , "Ч":57 ,
        "Ш":58,"Щ":59 , "Ъ":60 , "Ы":61 , "Ь":62 , "Э":63 , "Ю":64 ,
        "Я":65 , "1":66 , "2":67 , "3":68 , "4":69 , "5":70 , "6":71 ,
        "7": 72, "8":73 , "9":74 , "0":75
    }
```

```
dicts2 = {v: k for k, v in dicts.items()}
```

```
text = P1
```

```
digits_text = []
```

```
digits_gamma = []
```

```
for i in text:
```

```
    digits_text.append(dicts[i])
```

```
print("Числа текста ", digits_text)
```

```
for i in gamma:
```

```
    digits_gamma.append(dicts[i])
```

```
print("Числа гаммы ", digits_gamma)
```

```

digits_result = []
ch = 0
for i in text:
    try:
        a = dicts[i] + digits_gamma[ch]

    except:
        ch = 0
        a = dicts[i] + digits_gamma[ch]
    if a > 75:
        a = a % 75
        print(a)
    ch += 1
    digits_result.append(a)
print("Числа шифротекста ", digits_result)

text_cr = ""
for i in digits_result:
    text_cr += dicts2[i]
print("Шифротекст ", text_cr)

digits = []
for i in text_cr:
    digits.append(dicts[i])
ch = 0
digits1 = []
for i in digits:
    try:
        a = i - digits_gamma[ch]

```

```

except:
    ch = 0
    a = i - digits_gamma[ch]
    if a < 1:
        a = 75 + a
    digits1.append(a)
    ch += 1

text_decr = ""
for i in digits1:
    text_decr += dicts2[i]
print("Расшифрованный текст: ", text_decr)

```

3.2 Контрольный пример

```

14 def vzlom(P1, P2):
15     code = []
16     for i in range(len(P1)):
17         code.append(liters[(liters.index(P1[i]) + liters.index(P2[i])) % len(liters)])
18     print(code)
19     pr = "".join(code)
20     print(pr)

```

✓ [2] 1 len(P1)
13

✓ [3] 1 len(P2)
13

✓ [4] 1 vzlom(P1, P2)
['х', 'у', 'л', 'б', 'г', 'а', 'р', 'б', 'ю', 'с', 'щ', 'б', 'щ']
хУлбгАрбЮсЩбЩ

Рис. 3.1: Работа алгоритма взлома ключа

```
✓ [26] 1 P1 = "КодофаяФраза1"
      2 gamma = "хУльгАрБЮСЦЬЩ"

✓ [27] 1 shifr(P1, gamma)

      Числа текста [44, 16, 5, 16, 22, 1, 32, 54, 18, 1, 9, 1, 66]
      Числа гаммы [23, 53, 45, 62, 4, 33, 18, 34, 64, 51, 59, 62, 59]
      3
      13
      7
      50
      Числа шифротекста [67, 69, 50, 3, 26, 34, 50, 13, 7, 52, 68, 63, 50]
      Шифротекст 24РвшБРлётЗЭР
      Расшифрованный текст: КодофаяФраза1
```

Рис. 3.2: Работа алгоритма шифрования и дешифровки

4 Выводы

В ходе выполнения лабораторной работы было разработано приложение, позволяющее шифровать тексты в режиме однократного гаммирования.

Список литературы

1. Шифрование методом гаммирования
2. Режим гаммирования в блочном алгоритме шифрования