Отчёт по лабораторной работе №7

Шифргаммирования

Назармамадов Умед Джамшедович

Содержание

1	Цельработы 4	
2	Теоретическиесведения 5 2.1 Шифр гаммирования	5
3	Выполнениеработы 7 3.1 Реализация шифратора и дешифратора Python 3.2 Контрольный пример	
4	Выводы 10	J
Сп	исоклитературы 1	.1

List of Figures

1 Цель работы

Изучение алгоритма шифрования гаммированием

2 Теоретические сведения

2.1 Шифр гаммирования

Гаммирование — это наложение (снятие) на открытые (зашифрованные) данные криптографической гаммы,т.е. последовательности элементов данных, вырабатываемыхспомощьюнекоторогокриптографическогоалгоритма,дляполучения 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 пердставить 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 гамм H(2).
- 4. Подсчет контрольной суммы участка данных, соответствующего сегменту данных H(2) и т.д.

3 Выполнение работы

3.1 Реализация шифратора и дешифратора Python

```
def main(text, gamma):
      dict = {"a" :1, "б" :2 , "в" :3 ,"г" :4 ,"д" :5 ,"е" :6 ,"ё" :7 ,"ж": 8, "з":
                  "M": 14, "H": 15, "o": 16, "п": 17,
                                                           "р": 18, "с": 19, "т": 20, "у": 21, "ф": 22, "х": 23, "ц": 24, "ч": 2
                  "ы": 29, "ь": 30, "э": 31, "ю": 32, "я": 32
                  } dict2 = {v: k for k, v in dict.items()}
      digits_text = list() digits_gamma = list()
      for i in text:
            digits_text.append(dict[i]) print("Числа текста: ",
      digits_text)
      for i in gamma:
            digits_gamma.append(dict[i]) print("Числа
      гаммы: ", digits_gamma)
      digits_res = list()
      ch = 0
      for i in text:
            try: a = dict[i] + digits_gamma[ch] except:
                  ch = 0
                  a = dict[i] + digits_gamma[ch] if a>=33:
```

```
a = a%33 ch
      += 1
      digits_res.append(a) print("Числа шифровки:
", digits_res)
text_enc = "" for i in
digits_text:
     text_enc += dict2[i] print("Шифровка: ", text_enc)
digits = list() for i in
text_enc:
      digits.append(dict[i])
ch = 0
digits1 = list() for i in digits: a = i -
digits_gamma[ch] if a < 1:
            a = 33 + a
      digits1.append(a)
      ch += 1 text_dec = ""
for i in digits1:
     text_dec += dict2[i] print("Рассшифровка: ",
```

text_dec)

3.2 Контрольный пример

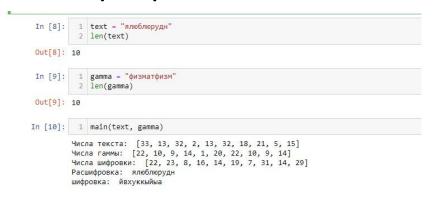


Figure 3.1: Работа алгоритма гаммирования

4 Выводы

Изучили алгоритмы шифрования на основе гаммирования

Список литературы

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