

Отчёт по лабораторной работе №8

Шифр гаммирования

Турсунов Баходурхон Азимджонович

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	9
4	Выводы	10
	Список литературы	11

List of Figures

3.1	Работа алгоритма взлома ключа	8
3.2	Работа алгоритма шифрования и дешифровки	9

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

```
# создаем алфавит из русских букв и цифр
# он нужен для гаммирования
a = ord("a")
alphabeth = [chr(i) for i in range(a, a + 32)]
a = ord("0")
for i in range(a, a+10):
    alphabeth.append(chr(i))

a = ord("А")
for i in range(1040, 1072):
    alphabeth.append(chr(i))
print(alphabeth)
P1 = "Навашиксходящийот1204"
P2 = "ВСеверныйфилиалБанка"
# длина ключа 20
key = "05 0C 17 7F 0E 4E 37 D2 94 10 09 2E 22 57 FF C8 0B B2 70 54"

def vzlom(P1, P2):
    code = []
    for i in range(20):
        code.append(alphabeth[(alphabeth.index(P1[i]) + alphabeth.index(P2[i])) % len(alphabeth)])
    #получили известные символы в шаблоне
    print(code)
    print(code[16], " и ", code[19])
    p3 = "".join(code)
    print(p3)

vzlom(P1, P2)
```

['а', 'о', 'в', 'г', 'д', 'е', 'ж', 'з', 'и', 'й', 'к', 'л', 'м', 'н', 'о', 'п', 'р', 'с', 'т', 'у', 'ф', 'х', 'ц', 'ч', 'ш', 'щ', 'ь', 'ы', 'ъ', 'э', 'ю', 'я', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'А', 'Б', 'В', 'Г', 'Д', 'Е', 'Ж', 'З', 'И', 'Й', 'К', 'Л', 'М', 'Н', 'О', 'П', 'Р', 'С', 'Т', 'У', 'Ф', 'Х', 'Ц', 'Ч', 'Ш', 'Щ', 'Ь', 'Ы', ' ', 'б', 'в', 'г', 'д', 'е', 'ж', 'з', 'и', 'й', 'к', 'л', 'м', 'н', 'о', 'п', 'р', 'с', 'т', 'у', 'ф', 'х', 'ц', 'ч', 'ш', 'щ', 'ь', 'ы', '1 и 4
щС3вэшюЖчш74рйщУ1ЕА4

Figure 3.1: Работа алгоритма взлома ключа

3.2 Контрольный пример

```
~ ~ ~
    a = i - listofdigitsofgamma[ch]
    if a < 1:
        a = 75 + a
    listofdigits1.append(a)
    ch += 1
textdecrypted = ""
for i in listofdigits1:
    textdecrypted += dict2[i]
print("Расшифрованный текст", textdecrypted)

shifr(P1)

Введите гамму(на русском языке! Да и пробелы тоже нельзя! Короче, только символы из dict7Щаф0рРпплфыв
Числа текста [47, 1, 35, 1, 26, 10, 19, 23, 16, 5, 32, 27, 10, 11, 16, 20, 66, 67, 75, 69]
числа гаммы [72, 59, 3, 1, 22, 48, 18, 50, 4, 17, 13, 22, 29, 3]
44
13
4
22
42
Числа зашифрованного текста [44, 60, 38, 2, 48, 58, 37, 73, 20, 22, 45, 49, 39, 14, 13, 4, 69, 68, 22, 42]
Зашифрованный текст: КЪЕ60ЩД8тфПЕмл43фИ
Расшифрованный текст НаВашисходящийот1204
```

Figure 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