

# **Отчет по лабораторной работе 8**

Даниил Генералов, 1032212280

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
<b>3</b>	<b>Контрольные вопросы</b>	<b>10</b>
<b>4</b>	<b>Выводы</b>	<b>12</b>

## Список иллюстраций

2.1	plaintext, template и ключ . . . . .	6
2.2	xor шифртекстов . . . . .	7
2.3	изменение угаданного текста . . . . .	8
2.4	определение исходного текста . . . . .	9

## **Список таблиц**

# 1 Цель работы

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



Затем, сделав XOR между этим и шаблоном, можно найти те слова, которые могут быть частью исходного текста (рис. fig. 2.2).

```
cipher1 = xor_bytes(plaintx1, key)
cipher2 = xor_bytes(plaintx2, key)
print(hex(cipher1))
print(hex(cipher2))

[6] ✓ 0.0s
... f2 d3 fd 15 1b 90 0d 90 01 c2 08 37 ca ef c7 cb 34 a2 51 2e 06 32 23 b9 4a 81 43 1a
f4 ee d0 28 64 be 3c a5 09 eb 0a 3e 86 e9 94 aa 02 8f 7b 15 5f 15 38 ae 42 da 5d 58

both_ciphers = xor_bytes(cipher1, cipher2)
print(hex(both_ciphers))

[7] ✓ 0.0s
... 06 3d 2d 3d 7f 2e 31 35 08 29 02 09 4c 06 53 61 36 2d 2a 3b 59 27 1b 17 08 5b 1e 42

both_plus_template = xor_bytes(both_ciphers, template)
print(hex(both_plus_template))

[8] ✓ 0.0s
... 54 68 69 73 20 6d 65 73 73 09 22 29 6c 26 73 41 16 0d 0a 1b 79 07 3b 37 28 7b 3e 3f

def ascii_filter_unprintable(b):
    text = ''
    for c in b:
        if c < 32 or c > 126:
            text += '~'
        else:
            text += chr(c)
    return text

print(ascii_filter_unprintable(both_plus_template))

[9] ✓ 0.0s
... This mess~")l&sA~::~~y~;7({>?
```

Рис. 2.2: хог шифртекстов

Можно затем попробовать этот процесс несколько раз, чтобы отгадывать больше частей исходного сообщения: подставлять угаданные куски, затем делать XOR, и смотреть какие части остаются.

```

guess2 = b'This message &sA~~~~y~;7({>?'
[10] ✓ 0.0s

def textdiff(a, b):
    print(ascii_filter_unprintable(a))
    chars = ''
    for q,w in zip(a,b):
        if q != w:
            chars += '|'
        else:
            chars += ' '
    print(chars)
    print(ascii_filter_unprintable(b))

textdiff(template, guess2)
[11] ✓ 0.0s

...
RUDN_CTF{
|||||
This message &sA~~~~y~;7({>?

mix = xor_bytes(cipher1, cipher2)

both_plus_guess = xor_bytes(mix, guess2)

textdiff(both_plus_guess, both_plus_template)
[14] ✓ 0.0s

...
RUDN_CTF{Hell  HSTE Y  }
|||||
This mess~"}&sA~~~~y~;7({>?

guess1 = b'RUDN_CTF{Hello  HSTE Y  }'
both_plus_guess = xor_bytes(mix, guess1)
ascii_filter_unprintable(both_plus_guess)
[15] ✓ 0.0s

...
'This message isA~~~~y~;7({>?'

```

Рис. 2.3: изменение угаданного текста

Постепенно, смешивая шифртекст со своим предсказанием, мы можем получить исходные тексты обоих сообщений на рис. fig. 2.4. Зная исходные тексты и шифртексты, мы можем определить ключ, и затем расшифровывать любые другие сообщения, которые зашифрованы этим ключом (а также генерировать собственные).



```
Code | Markdown | Run All | Restart | Clear All Outputs | Variables | Outline |
guess1 = b'RUDN_CTF{Hello ASCII Y   }'
both_plus_guess = xor_bytes(mix, guess1)
ascii_filter_unprintable(both_plus_guess)
[17] ✓ 0.0s
... 'This message is encry-;7({>?'

guess2 = b'This message is encrypted(>?'
both_plus_guess = xor_bytes(mix, guess2)
ascii_filter_unprintable(both_plus_guess)
[19] ✓ 0.0s
... 'RUDN_CTF{Hello ASCII Worl  }'

guess1 = b'RUDN_CTF{Hello ASCII World }'
both_plus_guess = xor_bytes(mix, guess1)
ascii_filter_unprintable(both_plus_guess)
[20] ✓ 0.0s
... 'This message is encrypted?>>'

guess2 = b'This message is encrypted?!?'
both_plus_guess = xor_bytes(mix, guess2)
ascii_filter_unprintable(both_plus_guess)
[22] ✓ 0.0s
... 'RUDN_CTF{Hello ASCII World?}'

guess1 = b'RUDN_CTF{Hello ASCII World!}'
both_plus_guess = xor_bytes(mix, guess1)
ascii_filter_unprintable(both_plus_guess)
[23] ✓ 0.0s
... 'This message is encrypted???'
```

Рис. 2.4: определение исходного текста

### 3 Контрольные вопросы

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

Если известен один шифртекст и его связанный исходный текст, то можно сделать XOR между ними, чтобы определить ключ. После этого можно использовать этот ключ, чтобы расшифровать другой шифртекст.

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

Злоумышленник, наблюдающий оба шифртекста, может определить части исходного текста этих сообщений.

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

После обычного шифрования используемый ключ не выбрасывается, а используется для шифрования второго текста.

4. Перечислите недостатки шифрования одним ключом двух открытых текстов.

Как мы показали в этой лабораторной работе, самым большим недостатком этой схемы является то, что злоумышленник, наблюдающий оба шифртекста, может определить исходный текст сообщений.

5. Перечислите преимущества шифрования одним ключом двух открытых текстов.

Для этого шифрования можно использовать один и тот же ключ. Поскольку в режиме однократного гаммирования длина ключа равна длине сообщения, то использование одного и того же ключа позволяет минимизировать количество общих данных.

## **4 Выводы**

В этой лабораторной работе мы рассмотрели алгоритм однократного гаммирования и показали его слабость в случае, когда один и тот же ключ используется больше одного раза.