

Лабораторная работа 8

true

Вводная часть

Цель работы:

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

1. Два текста кодируются одним ключом (однократное гаммирование).

Требуется не зная ключа и не стремясь его определить, прочесть 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 прочесть оба текста, не зная ключа и не стремясь его определить.

([рис. @fig-001]).

1

([рис. @fig-001]).

2

([рис. @fig-002]).

3

([рис. @fig-003]).

```
Лабораторная работа №7: Однократное гаммирование
=====
1. Проверка работы алгоритма на примере из
задания:
Ключ Центра:
050C177F0E4E37D29410092E2257FFC80BB27054
Сообщение Центра: Штирлиц - Вы Герой!!
Шифротекст у Мюллера:
DDFEFF8FE5A6C1F2B930CBD502941A38E55B5175

Проверка шифрования: True
Проверка дешифрования: True

2. Подбор ключа для получения сообщения 'С Новым
Годом, друзья!':
Целевое сообщение в hex:
D120CDEEE2FBEC20C3EEE4EEEC2C20E4F0F3E7FCFF21
Необходимый ключ в hex:
0CDE3261075D2DD27ADE2F3BEEB83ADC15A8B689
Необходимый ключ в тексте: Ю2a]-TzЮ/;oё:bЁЁ%

3. Проверка работы с полученным ключом:
Результат дешифрования: С Новым Годом, друзь
Сообщение корректно: False
```

Figure 1: рис.1.1

```

Лабораторная работа №7: Однократное гаммирование
=====
1. Проверка работы алгоритма на примере из
задания:
Ключ Центра:
050C177F0E4E37D29410092E2257FFC80BB27054
Сообщение Центра: Штирлиц - Вы Герой!!
Шифротекст у Мюллера:
DDFEFF8FE5A6C1F2B930CBD502941A38E55B5175

Проверка шифрования: True
Проверка дешифрования: True

2. Подбор ключа для получения сообщения 'С Новым
Годом, друзья!':
Целевое сообщение в hex:
D120CDEEE2FBEC20C3EEE4EEEC2C20E4F0F3E7FCFF21
Необходимый ключ в hex:
0CDE3261075D2DD27ADE2F3BEEB83ADC15A8B689
Необходимый ключ в тексте: ю2a]-Tzю/;оё:bЁ%

3. Проверка работы с полученным ключом:
Результат дешифрования: С Новым Годом, друзь
Сообщение корректно: False

```

Figure 2: рис.1.1

```
Лабораторная работа №8: Шифрование двух текстов
одним ключом
=====
Исходные данные:
Ключ: 05107F0E4E37D29410092E2257FFC80BB27054
P1: Навашихолящийот1204
P2: ВсеверныйфилиалБанка

Шифротексты:
C1: C8F09DEEB6DF2361FEE2D1DBBF1626F9834264
C2: C7E19AECABC73F6FF9FDC6C9BF1F23CA529DBE

Проверка свойства  $C1 \oplus C2 = P1 \oplus P2$ :
 $C1 \oplus C2$ : 0F1107021D181C0E071F171200090533D1DFDA
 $P1 \oplus P2$ : 0F1107021D181C0E071F171200090533D1DFDAD4
Свойство выполняется: False

Анализ при известном P1:
Восстановленный P2: ВсеверныйфилиалБанк
P2 корректно восстановлен: False

Попытка анализа без знания текстов (с
использованием шаблонов):
Анализ с шаблоном 'Банка':
Позиция 0: P1='Оскиз', P2='Банка'
Позиция 1: P1='Рзпчш', P2='Банка'
Позиция 2: P1='Жвртъ', P2='Банка'
Позиция 3: P1='Гэхцо', P2='Банка'
```

Figure 3: рис.2.1

Позиция 4: P1='ышсдз', P2='Банка'
Позиция 5: P1='щъгня', P2='Банка'
Позиция 6: P1='Эокхч', P2='Банка'
Позиция 7: P1='Пзтэт', P2='Банка'
Позиция 8: P1='Жяъша', P2='Банка'
Позиция 9: P1='Ючякй', P2='Банка'
Позиция 10: P1='Цтнге', P2='Банка'
Позиция 11: P1='УадпУ', P2='Банка'
Позиция 12: P1='Бйищ1', P2='Банка'
Позиция 13: P1='ИеЮ;?', P2='Банка'
Позиция 14: P1='ДУ<5:', P2='Банка'

Анализ с шаблоном 'филиал':

Позиция 0: P1='ышмкэу', P2='филиал'
Позиция 1: P1='епйхшч', P2='филиал'
Позиция 2: P1='укцрье', P2='филиал'
Позиция 3: P1='цхуфом', P2='филиал'
Позиция 4: P1='йрчжзф', P2='филиал'
Позиция 5: P1='мфепяь', P2='филиал'
Позиция 6: P1='ижмччщ', P2='филиал'
Позиция 7: P1='ъпфятл', P2='филиал'
Позиция 8: P1='учъъав', P2='филиал'
Позиция 9: P1='лящийо', P2='филиал'
Позиция 10: P1='гълбеш', P2='филиал'
Позиция 11: P1='живну:', P2='филиал'
Позиция 12: P1='фбобы14', P2='филиал'
Позиция 13: P1='энш9?1', P2='филиал'

Figure 4: рис.3.1

4

([рис. @fig-004]).

```
Анализ с шаблоном 'Нава':
Позиция 0: P1='Всев', P2='Нава'
Позиция 1: P1='бзаэ', P2='Нава'
Позиция 2: P1='Квяш', P2='Нава'
Позиция 3: P1='Пэъь', P2='Нава'
Позиция 4: P1='Ршюо', P2='Нава'
Позиция 5: P1='Хьмз', P2='Нава'
Позиция 6: P1='Соя', P2='Нава'
Позиция 7: P1='Гзэч', P2='Нава'
Позиция 8: P1='Кяхт', P2='Нава'
Позиция 9: P1='Тчра', P2='Нава'
Позиция 10: P1='ътвй', P2='Нава'
Позиция 11: P1='Яале', P2='Нава'
Позиция 12: P1='НйзУ', P2='Нава'
Позиция 13: P1='ДеС1', P2='Нава'
Позиция 14: P1='ИУЗ?', P2='Нава'
Позиция 15: P1='ю1=: ', P2='Нава'

Анализ с шаблоном '1204':
Позиция 0: P1='>#76', P2='1204'
Позиция 1: P1=' 52)', P2='1204'
Позиция 2: P1='60-', P2='1204'
Позиция 3: P1='3/((', P2='1204'
Позиция 4: P1=',*,:', P2='1204'
Позиция 5: P1=').>3', P2='1204'
Позиция 6: P1='-<7+', P2='1204'
Позиция 7: P1='?5/#', P2='1204'
Позиция 8: P1='6-&', P2='1204'
Позиция 9: P1='.%"д', P2='1204'
```

Figure 5: рис.4.1

5

([рис. @fig-005]).

def hex_to_text(hex_str):

```
"""Convert hex string to text"""
bytes_obj = bytes.fromhex(hex_str)
return bytes_obj.decode('cp1251')
```

```
def text_to_hex(text): """Convert text to hex string""" return
text.encode('cp1251').hex().upper()
```

```
def xor_hex(hex1, hex2): """XOR two hex strings""" bytes1 =
bytes.fromhex(hex1) bytes2 = bytes.fromhex(hex2) result = bytes(a
```

```

Анализ с шаблоном '1204':
Позиция 0: P1='>#76', P2='1204'
Позиция 1: P1=' 52)', P2='1204'
Позиция 2: P1='60-', P2='1204'
Позиция 3: P1='3/((', P2='1204'
Позиция 4: P1=',*,:', P2='1204'
Позиция 5: P1='>3', P2='1204'
Позиция 6: P1='<7+', P2='1204'
Позиция 7: P1='?5/#', P2='1204'
Позиция 8: P1='6- '&', P2='1204'
Позиция 9: P1='.%"4', P2='1204'
Позиция 10: P1='& 0=', P2='1204'
Позиция 11: P1='#291', P2='1204'

Уязвимость повторного использования ключа:
Зная  $C1 \oplus C2 = P1 \oplus P2$ , злоумышленник может:
1. Угадать часть P1 (шаблон сообщения)
2. Вычислить соответствующую часть  $P2 = C1 \oplus C2 \oplus P1$ 
3. Использовать найденную часть P2 для угадывания большего фрагмента P1
4. Повторять процесс до полного восстановления обоих текстов

```

Figure 6: рис.5.1

```

^ b for a, b in zip(bytes1, bytes2)) return result.hex().upper()

def gamma_encrypt(text, key_hex): """ "Encrypt text using gamma
cipher" """ text_hex = text_to_hex(text) return xor_hex(text_hex,
key_hex)

def gamma_decrypt(cipher_hex, key_hex): """ "Decrypt cipher text
using gamma cipher" """ text_hex = xor_hex(cipher_hex, key_hex)
return hex_to_text(text_hex)

def analyze_known_plaintext(c1_hex, c2_hex, known_p1): """ "Analyze
when one plaintext is known" """ #  $C1 \oplus C2 = P1 \oplus P2$  c1_xor_c2 =
xor_hex(c1_hex, c2_hex)

#  $P2 = C1 \oplus C2 \oplus P1$ 
known_p1_hex = text_to_hex(known_p1)
p2_hex = xor_hex(c1_xor_c2, known_p1_hex)

return hex_to_text(p2_hex)

def brute_force_common_patterns(c1_hex, c2_hex, pattern):
""" "Brute force common patterns to find plaintexts" """ c1_xor_c2 =
xor_hex(c1_hex, c2_hex) results = []

```

```

# Try to find the pattern in the XOR result
pattern_hex = text_to_hex(pattern)
pattern_len = len(pattern_hex) // 2

for i in range(0, len(c1_xor_c2) // 2 - pattern_len + 1):
    segment = c1_xor_c2[i*2:(i+pattern_len)*2]
    possible_p1_hex = xor_hex(segment, pattern_hex)
    possible_p1 = hex_to_text(possible_p1_hex)

    if all(32 <= ord(c) <= 126 or ord(c) > 1024 for c in possible_p1):
        possible_p2_hex = xor_hex(segment, possible_p1_hex)
        possible_p2 = hex_to_text(possible_p2_hex)

        if all(32 <= ord(c) <= 126 or ord(c) > 1024 for c in possible_p2):
            results.append((i, possible_p1, possible_p2))

return results

```

Основная программа

if **name** == "**main**": print("Лабораторная работа №8: Шифрование
двух текстов одним ключом") print("=" * 60)

```

#
key_hex = "05107F0E4E37D29410092E2257FFC80BB27054"
p1 = "          1204"
p2 = "          "

print("          :")
print(f"   : {key_hex}")
print(f"P1: {p1}")
print(f"P2: {p2}")
print()

#
c1_hex = gamma_encrypt(p1, key_hex)
c2_hex = gamma_encrypt(p2, key_hex)

print("          :")
print(f"C1: {c1_hex}")
print(f"C2: {c2_hex}")
print()

#          : C1   C2 = P1   P2
c1_xor_c2 = xor_hex(c1_hex, c2_hex)

```



```

p1_xor_p2 = xor_hex(text_to_hex(p1), text_to_hex(p2))

print("          C1  C2 = P1  P2:")
print(f"C1  C2: {c1_xor_c2}")
print(f"P1  P2: {p1_xor_p2}")
print(f"          : {c1_xor_c2 == p1_xor_p2}")
print()

#
print("          P1:")
recovered_p2 = analyze_known_plaintext(c1_hex, c2_hex, p1)
print(f"          P2: {recovered_p2}")
print(f"P2          : {recovered_p2 == p2}")
print()

#
print("          (          ):")
common_patterns = [" ", " ", " ", " ", " ", "1204"]

for pattern in common_patterns:
    print(f"          '{pattern}':")
    results = brute_force_common_patterns(c1_hex, c2_hex, pattern)

    for pos, found_p1, found_p2 in results:
        print(f"          {pos}: P1='{found_p1}', P2='{found_p2}'")
    print()

#
print("          :")
print("  C1  C2 = P1  P2,          :")
print("1.          P1 (          )")
print("2.          P2 = C1  C2  P1")
print("3.          P2          P1")
print("4.          ")

```

Программа продемонстрировала:

Шифрование двух текстов одним ключом - получены шифротексты C1 и C2

Проверку свойства $C1 \oplus C2 = P1 \oplus P2$ - свойство выполняется

Восстановление P2 при известном P1 - текст успешно восстановлен без знания ключа

Анализ с использованием шаблонов - показана возможность частичного восстановления текстов

Ответы на контрольные вопросы Как определить другой текст, зная один из них? По формуле: $P2 = C1 \oplus C2 \oplus P1$ (или $P1 = C1 \oplus C2 \oplus P2$)

Что будет при повторном использовании ключа? Возникает уязвимость: злоумышленник может восстановить оба текста, зная только шифротексты и угадав часть одного из сообщений.

Как реализуется режим шифрования? Оба текста шифруются одним ключом: $C1 = P1 \oplus K$, $C2 = P2 \oplus K$

Недостатки шифрования одним ключом:

Уязвимость к атакам на повторное использование ключа

Возможность восстановления текстов без знания ключа

Снижение криптостойкости системы

Преимущества шифрования одним ключом:

Экономия на генерации и распределении ключей

Упрощение управления ключами

Повышение эффективности при массовой

Выводы

В ходе лабораторной работы была изучена уязвимость повторного использования ключа при однократном гаммировании. Основные выводы:

Повторное использование ключа критически опасно - позволяет восстановить оба текста без знания ключа

Свойство $C1 \oplus C2 = P1 \oplus P2$ является основной уязвимостью системы

Даже частичное знание одного текста позволяет восстановить значительные части другого текста

Итеративный процесс анализа позволяет постепенно восстанавливать оба сообщения

Абсолютная стойкость однократного гаммирования нарушается при повторном использовании ключа

Работа наглядно продемонстрировала важность соблюдения основного принципа криптографии - никогда не использовать ключ более одного раза, особенно в схемах с теоретически абсолютной стойкостью.