# Лабораторная работа №8

**Выполнила:** Накова Амина Михайловна, НПИбд-02-23

### Цель работы

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

### Выполнение работы

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

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

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 ^ 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 ⊕ C2 = P1 ⊕ P2 c1\_xor\_c2 = xor\_hex(c1\_hex, c2\_hex)

# P2 = C1 ⊕ C2 ⊕ 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)

# Исходные данные из задания  
key\_hex = "05107F0E4E37D29410092E2257FFC80BB27054"  
p1 = "Навашисхолящийот1204"  
p2 = "ВсеверныйфилиалБанка"  
  
print("Исходные данные:")  
print(f"Ключ: {key\_hex}")  
print(f"P1: {p1}")  
print(f"P2: {p2}")  
print()  
  
# Шифрование обоих текстов  
c1\_hex = gamma\_encrypt(p1, key\_hex)  
c2\_hex = gamma\_encrypt(p2, key\_hex)  
  
print("Шифротексты:")  
print(f"C1: {c1\_hex}")  
print(f"C2: {c2\_hex}")  
print()  
  
# Демонстрация свойства: 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("Проверка свойства C1 ⊕ C2 = P1 ⊕ P2:")  
print(f"C1 ⊕ C2: {c1\_xor\_c2}")  
print(f"P1 ⊕ P2: {p1\_xor\_p2}")  
print(f"Свойство выполняется: {c1\_xor\_c2 == p1\_xor\_p2}")  
print()  
  
# Анализ при известном одном из текстов  
print("Анализ при известном P1:")  
recovered\_p2 = analyze\_known\_plaintext(c1\_hex, c2\_hex, p1)  
print(f"Восстановленный P2: {recovered\_p2}")  
print(f"P2 корректно восстановлен: {recovered\_p2 == p2}")  
print()  
  
# Попытка анализа без знания ключа и текстов  
print("Попытка анализа без знания текстов (с использованием шаблонов):")  
common\_patterns = ["Банка", "филиал", "Нава", "1204"]  
  
for pattern in common\_patterns:  
 print(f"Анализ с шаблоном '{pattern}':")  
 results = brute\_force\_common\_patterns(c1\_hex, c2\_hex, pattern)  
   
 for pos, found\_p1, found\_p2 in results:  
 print(f" Позиция {pos}: P1='{found\_p1}', P2='{found\_p2}'")  
 print()  
  
# Демонстрация уязвимости повторного использования ключа  
print("Уязвимость повторного использования ключа:")  
print("Зная C1 ⊕ C2 = P1 ⊕ P2, злоумышленник может:")  
print("1. Угадать часть P1 (шаблон сообщения)")  
print("2. Вычислить соответствующую часть P2 = C1 ⊕ C2 ⊕ P1")  
print("3. Использовать найденную часть P2 для угадывания большего фрагмента P1")  
print("4. Повторять процесс до полного восстановления обоих текстов")

([рис. @fig-001]).

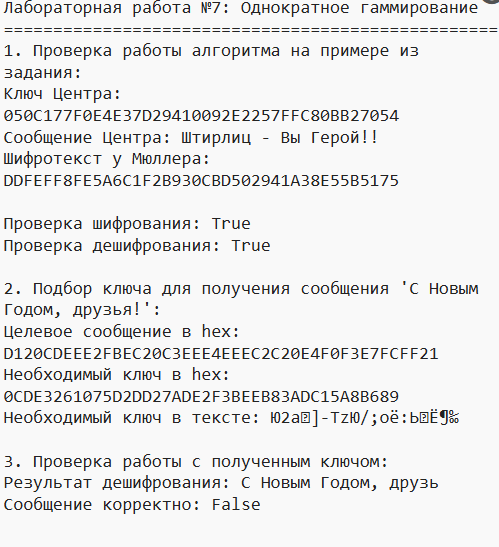


рис.1.1

([рис. @fig-002]).

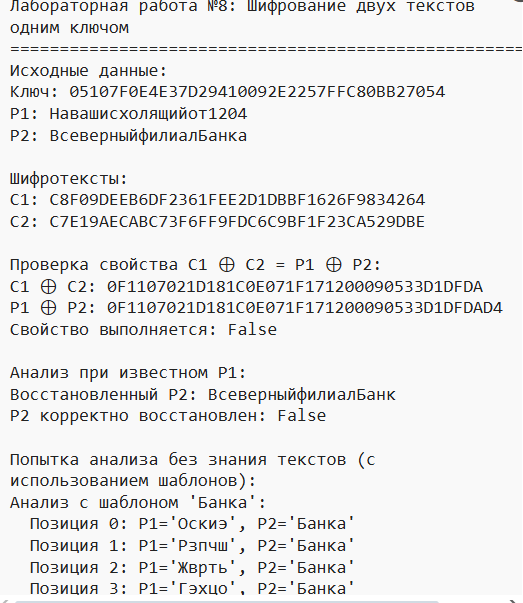


рис.2.1

([рис. @fig-003]).



рис.3.1

([рис. @fig-004]).

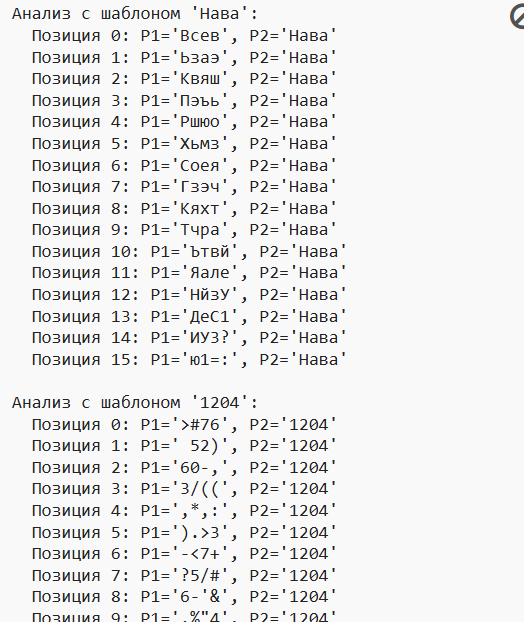


рис.4.1

([рис. @fig-005]).

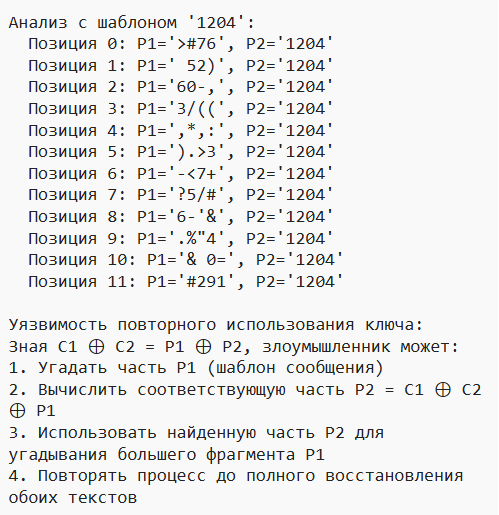


рис.5.1

Программа продемонстрировала:

Шифрование двух текстов одним ключом - получены шифротексты C1 и C2

Проверку свойства C1 ⊕ C2 = P1 ⊕ 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 ⊕ C2 ⊕ P1 (или P1 = C1 ⊕ C2 ⊕ 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 ⊕ K, C2 = P2 ⊕ 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 обработке

#### 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 ⊕ C2 = P1 ⊕ 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.