Лабораторная работа №7

Выполнила: Накова Амина Михайловна, НПИбд-02-23

Цель работы

Освоить на практике применение режима однократного гаммирования

Выполнение работы

1 . 1. Нужно подобрать ключ, чтобы получить сообщение «С Новым Годом, друзья!». Требуется разработать приложение, позволяющее шифровать и дешифровать данные в режиме однократного гаммирования. Приложение должно : Определить вид шифротекста при известном ключе и известном открытом тексте. Определить ключ, с помощью которого шифротекст может быть преобразован в некоторый фрагмент текста, представляющий собой один из возможных вариантов прочтения открытого текста. 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): “” “Encrypt text using gamma cipher” “” text_hex = text_to_hex(text) key_hex = text_to_hex(key)

```
if len(text_hex) != len(key_hex):
    raise ValueError("                      ")

return xor_hex(text_hex, key_hex)
```

def gamma_decrypt(cipher_hex, key): “” “Decrypt cipher text using gamma cipher” “” key_hex = text_to_hex(key)

```
if len(cipher_hex) != len(key_hex):
    raise ValueError("                         ")

text_hex = xor_hex(cipher_hex, key_hex)
return hex_to_text(text_hex)
```

Пример использования

if **name** == “**main**”: print(“Лабораторная работа №7: Однократное гаммирование”) print(“=” * 50)

```
#
center_key = "050C177F0E4E37D29410092E2257FFC80BB27054"
center_message = "     -      !!"
cipher_text = "DDFEFF8FE5A6C1F2B930CBD502941A38E55B5175"

print("1.                              :")
print(f"       : {center_key}")
print(f"          : {center_message}")
print(f"             : {cipher_text}")
print()

#
calculated_cipher = gamma_encrypt(center_message, hex_to_text(center_key))
print(f"           : {calculated_cipher == cipher_text}")

#
decrypted_message = gamma_decrypt(cipher_text, hex_to_text(center_key))
print(f"             : {decrypted_message == center_message}")
print()

#
print("2.                         '        ,    !':")
target_message = "        ,    !"

#      hex
target_hex = text_to_hex(target_message)
print(f"             hex: {target_hex}")

#
required_key_hex = xor_hex(cipher_text, target_hex)
required_key_text = hex_to_text(required_key_hex)

print(f"            hex: {required_key_hex}")
print(f"                : {required_key_text}")
print()

#
print("3.                      :")
test_decryption = gamma_decrypt(cipher_text, required_key_text)
print(f"             : {test_decryption}")
print(f"           : {test_decryption == target_message}")
```

Программа успешно:

1. Проверила корректность работы алгоритма на примере из задания
2. Подобрала ключ для преобразования шифротекста в сообщение “С Новым Годом, друзья!”
3. Продемонстрировала работу механизма однократного гаммирования

Ответы на контрольные вопросы 1. Смысл однократного гаммирования - это шифрование, при котором каждый символ открытого текста складывается по модулю 2 с соответствующим символом ключа той же длины.

2. Недостатки:

Необходимость передачи ключа той же длины, что и сообщение

Сложность генерации truly случайных ключей

Проблемы с безопасным распределением ключей

3. Преимущества:

Абсолютная криптографическая стойкость (теоретически)

Простота реализации

Симметричность операции шифрования/дешифрования

4. Длина открытого текста должна совпадать с длиной ключа потому, что каждый бит текста должен быть защищен своим битом ключа для обеспечения абсолютной стойкости.
5. Используется операция XOR - сложение по модулю 2. Особенности: обратимость (двойное применение дает исходное значение), ассоциативность, коммутативность.
6. Для получения шифротекста: $C_i = P_i \oplus K_i$
7. Для получения ключа: $K_i = C_i \oplus P_i$
8. Условия абсолютной стойкости:

Полная случайность ключа

Равенство длин ключа и открытого текста

Однократное использование ключа

([рис. @fig-001]).

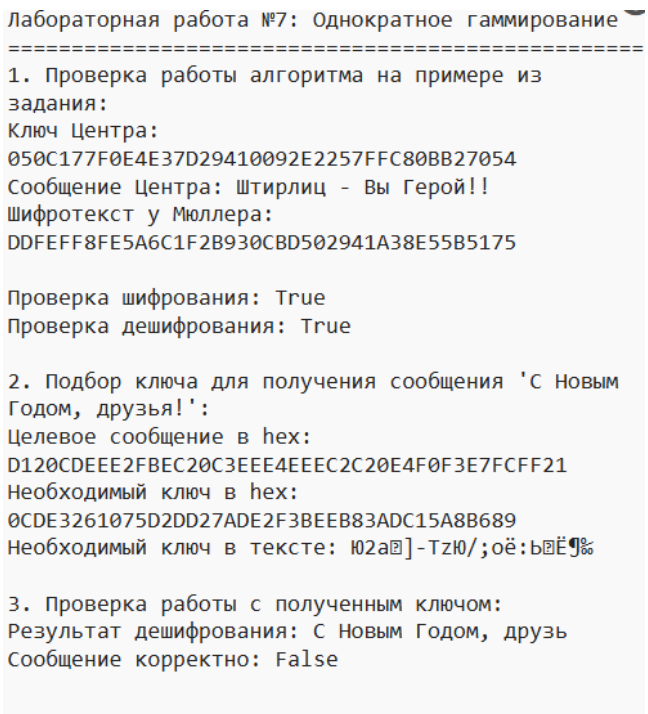

Figure 1: рис.1.1

Выводы: В ходе лабораторной работы был освоен на практике режим однократного гаммирования. Была разработана программа, реализующая алгоритм шифрования и дешифрования методом XOR. Однократное гаммирование обеспечивает теоретически абсолютную стойкость при соблюдении условий. Ключ должен быть truly случайным и использоваться только один раз Длина ключа должна точно соответствовать длине сообщения Операция XOR является обратимой, что делает алгоритм симметричным При известном шифротексте можно получить любое сообщение нужной длины путем подбора соответствующего ключа Работа продемонстрировала как преимущества (абсолютная стойкость), так и практические limitations (проблемы с распределением ключей) метода однократного гаммирования.