

Отчет по лабораторной работе 8

Основы Информационной безопасности

Хрусталеv Влад Николаевич

Содержание

1	Цель работы	5
2	Ход работы	6
2.1	Выполнение лабораторной	6
2.2	Ответы на контрольные вопросы	10
3	Вывод	12

Список иллюстраций

Список таблиц

1 Цель работы

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

2 Ход работы

2.1 Выполнение лабораторной

Нампишем программу эмулирующую задачу

```
def xor_bytes(a, b):  
    return bytes(x ^ y for x, y in zip(a, b))  
  
def extend_key(key, length):  
    return (key * (length // len(key) + 1))[:length]  
  
def encrypt(plaintext, key):  
    plaintext_bytes = plaintext.encode('utf-8')  
    extended_key = extend_key(key, len(plaintext_bytes))  
    ciphertext = xor_bytes(plaintext_bytes, extended_key)  
    return ciphertext.hex()  
  
def decrypt(ciphertext_hex, key):  
    ciphertext = bytes.fromhex(ciphertext_hex)  
    extended_key = extend_key(key, len(ciphertext))  
    plaintext_bytes = xor_bytes(ciphertext, extended_key)
```

```
return plaintext_bytes.decode('utf-8')
```

```
def attacker_decipher(C1_hex, C2_hex, known_P1):
```

```
    C1 = bytes.fromhex(C1_hex)
```

```
    C2 = bytes.fromhex(C2_hex)
```

```
    known_P1_bytes = known_P1.encode('utf-8')
```

```
    P1_xor_P2 = xor_bytes(C1, C2)
```

```
    P2_bytes = xor_bytes(P1_xor_P2, known_P1_bytes)
```

```
    return P2_bytes.decode('utf-8')
```

```
def main():
```

```
    P1 = "НаВашисходящийот1204"
```

```
    P2 = "ВСеверныйфилиалБанка"
```

```
    k_hex = "05 0C 17 7F 0E 4E 37 D2 94 10 09 2E 22 57 FF C8 0B B2 70 54"
```

```
    K = bytes.fromhex(k_hex)
```

```
    C1_hex = encrypt(P1, K)
```

```
    C2_hex = encrypt(P2, K)
```

```
    decrypted_C1 = decrypt(C1_hex, K)
```

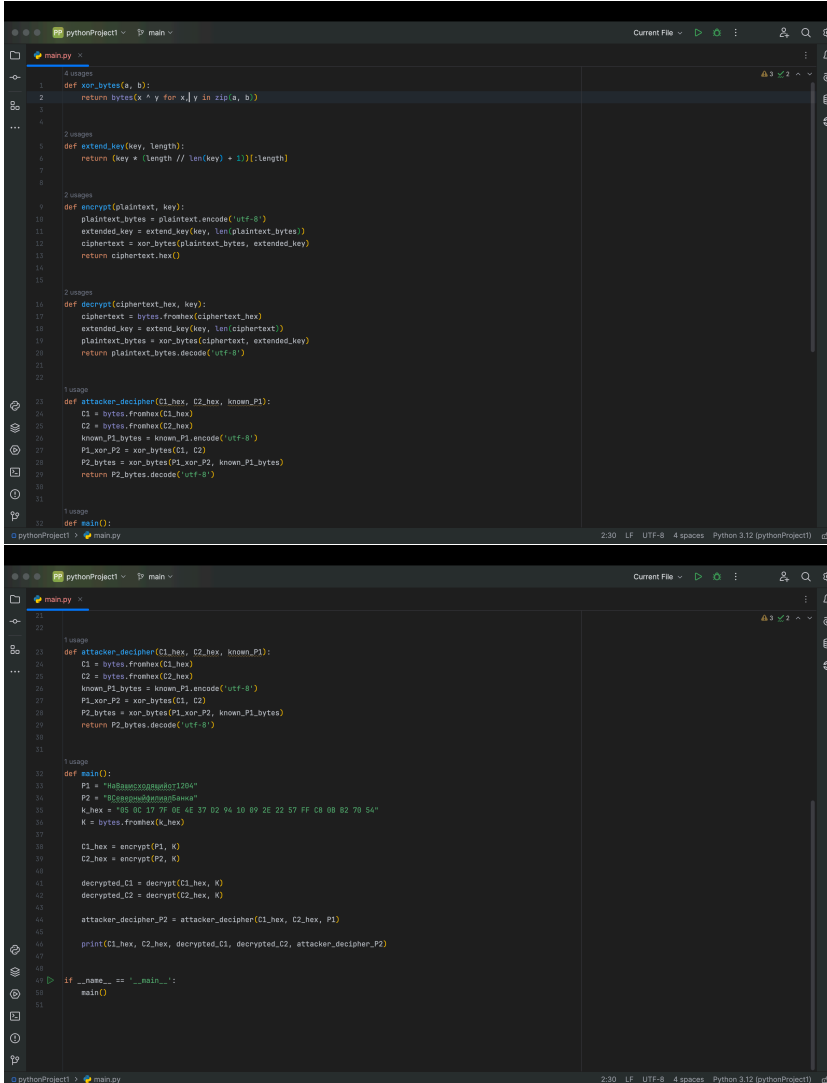
```
    decrypted_C2 = decrypt(C2_hex, K)
```

```
    attacker_decipher_P2 = attacker_decipher(C1_hex, C2_hex, P1)
```

```
    print(C1_hex, C2_hex, decrypted_C1, decrypted_C2, attacker_decipher_P2)
```

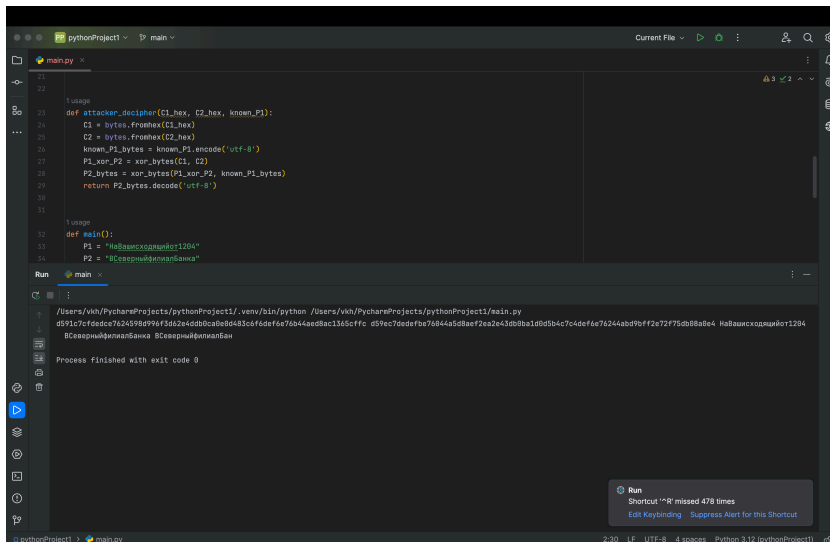
```
if __name__ == '__main__':  
    main()
```

Выше сам код для большей наглядности (рис. ?? , рис. ??)



Весь код не включая функции “attacker_decipher” эмулирует работу шифровальщика в одном ключе за счёт чего становится крайне не безопасным. Из-за свойств функции хог злоумышленник может расшифровать закодированные значения не зная ключи шифрования. Для этого ему надо пару Расшифрованное значение и Зашифрованное и другое сообщение которое хочет расшифровать. То есть нужно знать C1_hex C2_hex и P1_расшифрованный.

Посмотрим на вывод программы с имитацией атаки (рис. ??)



```
21
22
23 def attacker_decipher(C1_hex, C2_hex, known_P1):
24     C1 = bytes.fromhex(C1_hex)
25     C2 = bytes.fromhex(C2_hex)
26     known_P1_bytes = known_P1.encode('utf-8')
27     P1_xor_P2 = xor_bytes(C1, C2)
28     P2_bytes = xor_bytes(P1_xor_P2, known_P1_bytes)
29     return P2_bytes.decode('utf-8')
30
31
32 if __name__ == '__main__':
33     P1 = "Всем привет!"
34     P2 = "Всем привет!"
35
36     C1_hex = "d51c7c70d0c7624598099af3d62e4dd8bca8e0483c4fdef6e76b44e08ac1365cffe"
37     C2_hex = "d51c7c70d0c7624598099af3d62e4dd8bca8e0483c4fdef6e76b44e08ac1365cffe"
38
39     P2_decoded = attacker_decipher(C1_hex, C2_hex, P1)
40     print(P2_decoded)
```

Как мы видим злоумышленник успешно смог расшифровать значение P2 при том что не узнавал ключ шифрования. Почему это так?

В режиме однократного гаммирования (one-time pad), если один ключ используется для шифрования двух разных сообщений, возникает уязвимость. Это происходит из-за свойства операции XOR.

Для двух шифротекстов (C1) и (C2), зашифрованных с использованием одного ключа (K):

$$C1 = P1 \text{ xor } K$$

$$C2 = P2 \text{ xor } K$$

Злоумышленник может вычислить:

$$C1 \text{ xor } C2 = (P1 \text{ xor } K) \text{ xor } (P2 \text{ xor } K)$$

Из свойства операции XOR:

$$(P1 \text{ xor } K) \text{ xor } (P2 \text{ xor } K) = P1 \text{ xor } P2$$

Таким образом, злоумышленник получает:

$$P1 \text{ xor } P2 = C1 \text{ xor } C2$$

Предположим, что злоумышленник знает текст (P1) (или его формат). Тогда он может использовать известный текст (P1) для вычисления (P2).

1.Злоумышленник вычисляет (C1 xor C2):

$$P1 \text{ xor } P2 = C1 \text{ xor } C2$$

2.Зная (P1), злоумышленник может вычислить (P2):

$$P2 = (C1 \text{ xor } C2) \text{ xor } P1$$

2.2 Ответы на контрольные вопросы

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

Если злоумышленник знает один из текстов (P1) или (P2) и имеет доступ к шифротекстам (C1) и (C2), он может вычислить другой текст, используя следующее свойство XOR:

1.Вычислите (C1 xor C2):

$$P1 \text{ xor } P2 = C1 \text{ xor } C2$$

2.Используя известный текст (P1), вычислите (P2):

$$P2 = (C1 \text{ xor } C2) \text{ xor } P1$$

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

При повторном использовании ключа для шифрования двух разных текстов возникает уязвимость, так как злоумышленник может вычислить XOR шифротекстов:

$$C1 \text{ xor } C2 = P1 \text{ xor } P2$$

Это позволяет злоумышленнику, зная один из открытых текстов, определить второй текст, не зная ключа.

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

Режим однократного гаммирования с одним ключом для двух текстов реализуется следующим образом:

1.Задается ключ (K).

2.Каждый текст шифруется с использованием ключа K и операции XOR:

$$C1 = P1 \text{ xor } K$$

$$C2 = P2 \text{ xor } K$$

4.Перечислите недостатки шифрования одним ключом двух открытых текстов.

Недостатки шифрования одним ключом двух открытых текстов включают:

1.Уязвимость к атаке на основе известного текста: Если злоумышленник знает один из текстов, он может вычислить второй текст.

2.Повторное использование ключа: Приводит к уязвимости, позволяющей вычислить XOR двух текстов.

3.Отсутствие безопасности: Повторное использование ключа нарушает принцип безопасности однократного гаммирования.

4.Перечислите преимущества шифрования одним ключом двух открытых текстов.

Преимущества шифрования одним ключом двух открытых текстов включают:

1.Простота реализации: Легко реализовать с использованием операции XOR.

2.Эффективность: Быстрое шифрование и дешифрование, так как операция XOR выполняется быстро.

3.Минимальное использование ресурсов: Не требует сложных вычислений или большого объема памяти.

Однако эти преимущества применимы только в контексте простоты и эффективности реализации, и не гарантируют безопасности, если ключ используется более одного раза.

3 Вывод

Используя этот метод, злоумышленник может прочитать оба текста, не зная ключа и не стремясь его определить. Достаточно иметь доступ к одному из шифротекстов и знать или предполагать формат одного из открытых текстов.

Этот подход демонстрирует уязвимость однократного гаммирования при повторном использовании ключа для шифрования разных сообщений.