Лабораторная 8

Отчет

Карпачев Ярослав

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Освоить на практике режим однократного гаммирования (одноразового шифрования) на примере кодирования двух различных телеграмм одним ключом и продемонстрировать уязвимость повторного использования ключа.

1.1 Выполнение

Шифрование и получение шифротекстов

Кодирование двух исходных телеграмм P_1 и P_2 одним ключом K с помощью операции XOR.

Скрипт otp.py:

```
#!/usr/bin/env python3
# otp.py
import binascii
# Ключ (20 байт)
K = bytes.fromhex('05 0C 17 7F 0E 4E 37 D2 94 10 09 2E 22 57 FF C8 0B B2 70 54')
# Исходные тексты
P1 = 'НаВашисходящийот1204'.encode('cp1251')  # 20 байт
P2 = 'ВСеверныйфилиалБанка'.encode('cp1251')  # 20 байт
# Шифрование
C1 = bytes(a ^ b for a, b in zip(P1, K))
C2 = bytes(a ^ b for a, b in zip(P2, K))
# Вывод результатов
```

```
print('C1 =', binascii.hexlify(C1).decode().upper())
print('C2 =', binascii.hexlify(C2).decode().upper())
```

Результаты выполнения:

C1 = C8ECD59FF6A6C6277AF4F6D7CABE113A3A804060 C2 = C7DDF29DEBBEDA297DE4E1C5CAB

2. Демонстрация уязвимости: получение двух открытых текстов без знания ключа

Повторное использование ключа при шифровании P_1 и P_2 позволяет злоумышленнику, зная C_1 и C_2, получить $P_1 \oplus P_2$:

$$C1 \oplus C2 = P1 \oplus K \oplus P2 \oplus K = P1 \oplus P2.$$

Если P_1 известен (например, шаблонный текст), то:

$$P2 = (C1 \oplus C2) \oplus P1.$$

Вычисление C1⊕C2 и получение P_2 при известном P_1.

Скрипт otp_vuln.py:

```
#!/usr/bin/env python3
import binascii
# Двоичная операция XOR для шифротекстов и известного P1
def xor_bytes(a, b): return bytes(x ^ y for x, y in zip(a, b))
C1 = bytes.fromhex('C8ECD59FF6A6C6277AF4F6D7CABE113A3A804060')
C2 = bytes.fromhex('C7DDF29DEBBEDA297DE4E1C5CAB71409EB5F9AB4')
# Вычисляем P1⊕P2
X = xor_bytes(C1, C2)
# Известный шаблон P1
P1 = 'НаВашисходящийот1204'.encode('cp1251')
# Восстанавливаем P2
P2 = xor_bytes(X, P1)
print('C1⊕C2 =', binascii.hexlify(X).decode().upper())
print('Recovered P2 =', P2.decode('cp1251'))
```

Результаты выполнения:

C1⊠C2 = 0F3127021D181C0E0710171200090533D1DFDAD4 Recovered P2 = BCe-верныйфилиалБанка

2 Выводы

1. Повторное использование одного и того же ключа при режиме гаммирования приводит к опасной уязвимости: злоумышленник, имея два шифротекста, может получить XOR двух открытых текстов.
2. Зная один из открытых текстов (шаблон), можно полностью восстановить второй без знания ключа.
3. Ключ в режиме одноразовой гаммы должен использоваться лишь один раз; повторное использование делает шифрование небезопасным.