Лабораторная 7

Отчет

Карпачев Ярослав

Содержание

# 1 Цель работы

Освоить на практике применение режима однократного гаммирования

# 2 Задание

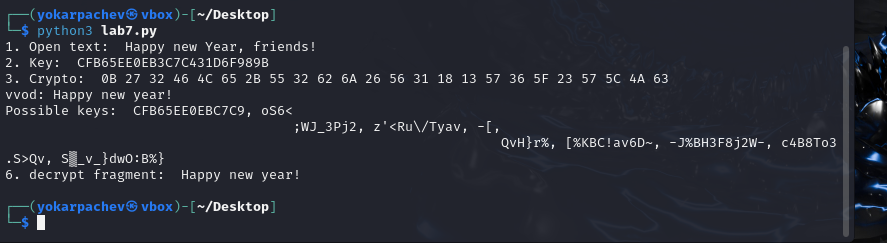
Нужно подобрать ключ, чтобы получить сообщение «С Новым Годом, друзья!». Требуется разработать приложение, позволяющее шифровать и дешифровать данные в режиме однократного гаммирования. Приложение должно: 1. Определить вид шифротекста при известном ключе и известном откры- том тексте. 2. Определить ключ, с помощью которого шифротекст может быть преоб- разован в некоторый фрагмент текста, представляющий собой один из возможных вариантов прочтения открытого текста.

# 3 Выполнение лабораторной работы

1. Пишим скрипт

#!/usr/bin/env python3  
# -\*- coding: utf-8 -\*-  
import random  
import string  
from typing import List, Tuple  
  
EXCHARS = "0123456789ABCDEF"  
  
  
def generate\_key(length: int) -> str:  
 """Return pseudo‑random printable HEX‑like string of \*length\* symbols."""  
 return ''.join(random.choice(HEXCHARS) for \_ in range(length))  
  
  
def xor\_strings(a: str, b: str) -> str:  
 """XOR two equal‑length strings, return new string of chars."""  
 return ''.join(chr(ord(x) ^ ord(y)) for x, y in zip(a, b))  
  
  
def to\_hex(s: str) -> str:  
 """Convert string → 'AA BB CC' spaced HEX representation."""  
 return ' '.join(f"{ord(ch):02X}" for ch in s)  
  
  
def find\_possible\_keys(cipher: str, fragment: str) -> List[Tuple[int, str]]:  
 frag\_len = len(fragment)  
 out: List[Tuple[int, str]] = []  
 for pos in range(len(cipher) - frag\_len + 1):  
 key\_candidate = xor\_strings(cipher[pos:pos + frag\_len], fragment)  
 out.append((pos, key\_candidate))  
 return out  
  
  
def main() -> None:  
 plain = "С Новым Годом, друзья!"  
  
 key = generate\_key(len(plain))  
  
 cipher = xor\_strings(plain, key)  
  
 print("1. Open text :", plain)  
 print("2. Key :", key)  
 print("3. Crypto text :", to\_hex(cipher))  
  
 fragment = input("4. Input fragment : ")  
  
 cand = find\_possible\_keys(cipher, fragment)  
 if not cand:  
 print("5. Possible keys : — (фрагмент не найден)")  
 return  
  
 keys\_only = [k for \_, k in cand]  
 print("5. Possible keys :", ', '.join(keys\_only))  
  
 pos0, key0 = cand[0]  
 decrypted = xor\_strings(cipher[pos0:pos0 + len(fragment)], key0)  
 print("6. Decrypted frag.:", decrypted)  
  
  
if name == "\_\_main\_\_":  
 main()

1. проверяем работу программы



Работа программы

1. Контрольные вопросы
2. Смысл однократного гаммирования – сложение (XOR) текста с однократной случайной гаммой той же длины.
3. Недостатки: нужна истинно случайная гамма; ключ хранить/передавать так же долго, как сообщение; ключ нельзя переиспользовать.
4. Преимущества: абсолютная криптостойкость; простота реализации; симметричность (шифр = дешифр).
5. Длины равны, чтобы каждый символ текста «прикрывался» одним символом гаммы; иначе остаётся статистическая избыточность.
6. Операция – XOR (сложение по модулю 2); даёт ‑– при повторном применении тем же ключом восстанавливается исходник.
7. Шифротекст: C\_i = P\_i K\_i.
8. Ключ: K\_i = C\_i P\_i.
9. Условия абсолютной стойкости: (a) гамма истинно случайна; (b) длина ключа = длина сообщения; (c) ключ используется лишь однажды.

# 4 Выводы

Я освоить на практике применение режима однократного гаммирования