Шифр гаммирования

Боровиков Даниил Александрович НПИбд-01-22

11 мая, 2024, Москва, Россия

Российский Университет Дружбы Народов

Цели и задачи

Цель лабораторной работы

Изучение алгоритма шифрования гаммированием

Выполнение лабораторной работы

Гаммирование

Гаммирование – это наложение (снятие) на открытые (зашифрованные) данные криптографической гаммы, т.е. последовательности элементов данных, вырабатываемых с помощью некоторого криптографического алгоритма, для получения зашифрованных (открытых) данных.

Гаммирование

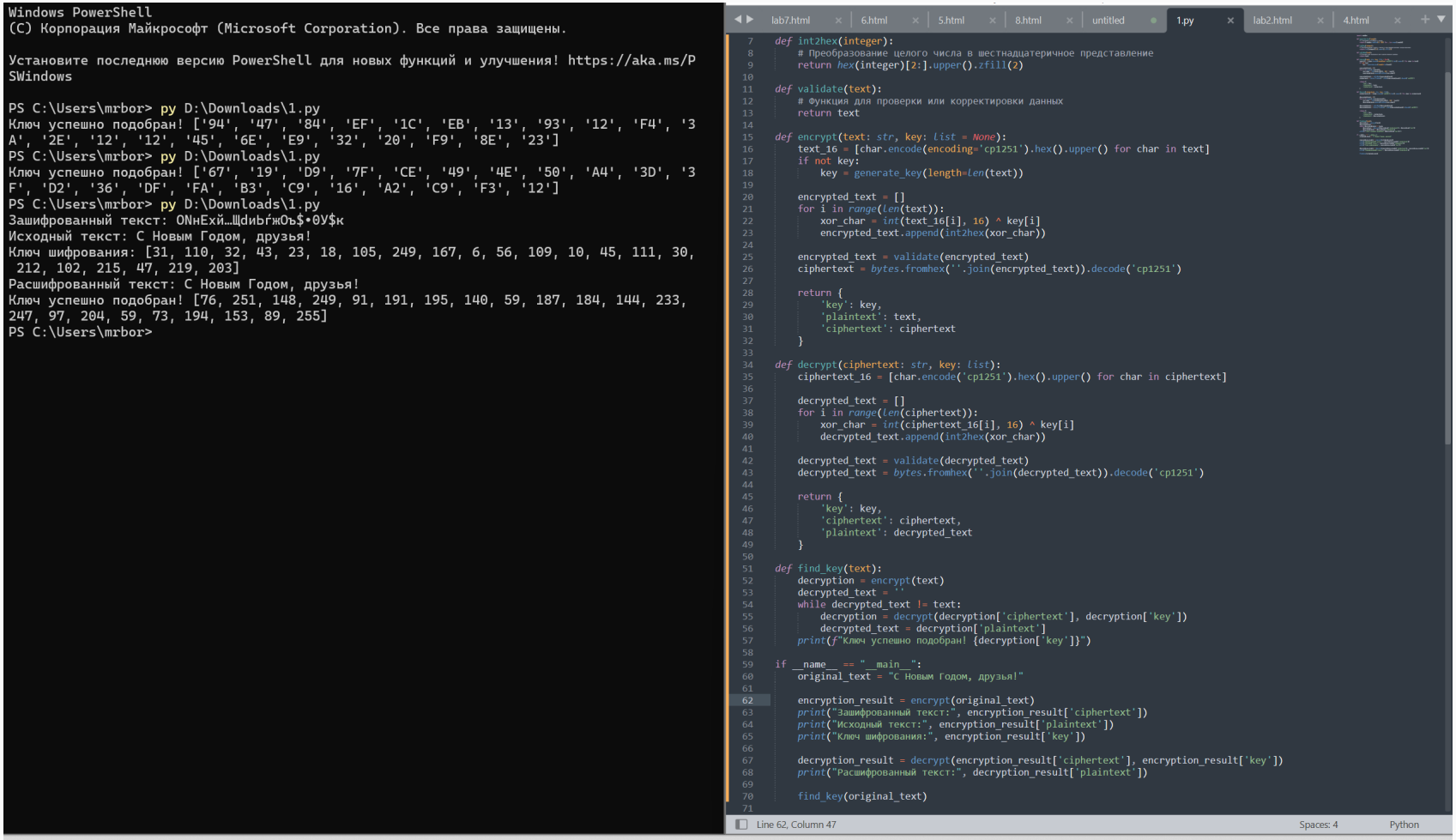
Наложение (или снятие) гаммы на блок сообщения в рассматриваемом нами стандарте реализуется с помощью операции побитного сложения по модулю 2 (XOR). То есть при шифровании сообщений каждый блок открытого сообщения ксорится с блоком криптографической гаммы, длина которого должна соответствовать длине блоков открытого сообщения. При этом, если размер блока исходного текста меньше, чем размер блока гаммы, блок гаммы обрезается до размера блока исходного текста (выполняется процедура усечения гаммы).

Формула

В аддитивных шифрах символы исходного сообщения заменяются числами, которые складываются по модулю с числами гаммы. Ключом шифра является гамма, символы которой последовательно повторяются. Перед шифрованием символы сообщения и гаммы заменяются их номерами в алфавите и само кодирование выполняется по формуле

$$Ci = (Ti + Gi) mod N$$

Пример работы алгоритма



```
Windows PowerShell
(C) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

Установите последнюю версию PowerShell для новых функций и улучшения! https://aka.ms/P
SWindows

PS C:\Users\mrbor> py D:\Downloads\1.py
Ключ успешно подобран! ['94', '47', '84', 'EF', '1C', 'EB', '13', '93', '12', 'F4', '3
A', '2E', '12', '12', '45', '6E', 'E9', '32', '20', 'F9', '8E', '23']
PS C:\Users\mrbor> py D:\Downloads\1.py
Ключ успешно подобран! ['67', '19', 'D9', '7F', 'CE', '49', '4E', '50', 'A4', '3D', '3
F', 'D2', '36', 'DF', 'FA', 'B3', 'C9', '16', 'A2', 'C9', 'F3', '12']
PS C:\Users\mrbor> py D:\Downloads\1.py
Зашифрованный текст: ONнExй…ЩdиЬѓжОъ$•0У$к
Исходный текст: С Новым Годом, друзья!
Ключ шифрования: [31, 110, 32, 43, 23, 18, 105, 249, 167, 6, 56, 109, 10, 45, 111, 30,
 212, 102, 215, 47, 219, 203]
Расшифрованный текст: С Новым Годом, друзья!
Ключ успешно подобран! [76, 251, 148, 249, 91, 191, 195, 140, 59, 187, 184, 144, 233,
247, 97, 204, 59, 73, 194, 153, 89, 255]
PS C:\Users\mrbor>
```

```python
def int2hex(integer):
    # Преобразование целого числа в шестнадцатеричное представление
    return hex(integer)[2:].upper().zfill(2)

def validate(text):
    # Функция для проверки или корректировки данных
    return text

def encrypt(text: str, key: list = None):
    text_16 = [char.encode(encoding='cp1251').hex().upper() for char in text]
    if not key:
        key = generate_key(length=len(text))

    encrypted_text = []
    for i in range(len(text)):
        xor_char = int(text_16[i], 16) ^ key[i]
        encrypted_text.append(int2hex(xor_char))

    encrypted_text = validate(encrypted_text)
    ciphertext = bytes.fromhex(''.join(encrypted_text)).decode('cp1251')

    return {
        'key': key,
        'plaintext': text,
        'ciphertext': ciphertext
    }

def decrypt(ciphertext: str, key: list):
    ciphertext_16 = [char.encode('cp1251').hex().upper() for char in ciphertext]

    decrypted_text = []
    for i in range(len(ciphertext)):
        xor_char = int(ciphertext_16[i], 16) ^ key[i]
        decrypted_text.append(int2hex(xor_char))

    decrypted_text = validate(decrypted_text)
    decrypted_text = bytes.fromhex(''.join(decrypted_text)).decode('cp1251')

    return {
        'key': key,
        'ciphertext': ciphertext,
        'plaintext': decrypted_text
    }

def find_key(text):
    decryption = encrypt(text)
    decrypted_text = ''
    while decrypted_text != text:
        decryption = decrypt(decryption['ciphertext'], decryption['key'])
        decrypted_text = decryption['plaintext']
    print(f"Ключ успешно подобран! {decryption['key']}")

if __name__ == "__main__":
    original_text = "С Новым Годом, друзья!"

    encryption_result = encrypt(original_text)
    print("Зашифрованный текст:", encryption_result['ciphertext'])
    print("Исходный текст:", encryption_result['plaintext'])
    print("Ключ шифрования:", encryption_result['key'])

    decryption_result = decrypt(encryption_result['ciphertext'], encryption_result['key'])
    print("Расшифрованный текст:", decryption_result['plaintext'])

    find_key(original_text)
```

Figure 1: Работа алгоритма гаммирования

Выводы

Результаты выполнения лабораторной работы

Изучили алгоритм шифрования с помощью гаммирования