

Міністерство освіти та науки України
Харківський національний університет радіоелектроніки
Кафедра програмної інженерії

Практична робота №2
з дисципліни: «Безпека програм та даних»
на тему: «Одноразовий блокнот»

Виконав

ст. гр. ПЗПІ-20-1

Бабанін А.К.

Перевірив

доцент кафедри ПІ

Турута О.О.

2023

Мета роботи: Ознайомити студентів з шифром «одноразовий блокнот», відпрацювати навички використання цього шифру для кодування та декодування тексту.

[Посилання на Google Collab.](#)

Індивідуальне завдання:

Завдання 1 Розшифрувати повідомлення прикладеним ключем. Варіант вибирається за номером в журналі (таблиця 6.1).

№	Повідомлення	Ключ
2	3652576465291928550126959788	сол

№	Текст завдання
2	Зашифровать любое шестизначное число сегодняшним днём недели

Таблиця кодування (словник) має наступний вигляд:

```
compression_table = {  
    '': ["А", "И", "Т", "Е", "С", "Н", "О"],  
    '8': ["Б", "В", "Г", "Д", "Ж", "З", "К", "Л", "М", "П"],  
    '9': ["Р", "У", "Ф", "Х", "Ц", "Ч", "Ш", "Щ", "Ъ", "Ы"],  
    '0': ["Ь", "Э", "Ю", "Я", "Й"]  
}  
  
require_extra_int = ['8', '9', '0']
```

Рисунок 1 – Таблиця стиснення у вигляді словника

Далі наведені два методи для перетворення символу у числове представлення та навпаки. Наприклад А – ‘1’, Г – ‘83’.

```
def getIntFromChar(ch):  
    # special case due to task limitations...  
    if ch == ' ':  
        return '00'  
  
    for (key, value) in compression_table.items():  
  
        if ch in value:  
            chIndex = value.index(ch)  
  
            concatInt = ((chIndex + 10) % 10) + 1  
  
            return key + str(concatInt)  
  
def getCharFromIntStr(intStr):  
  
    # special case due to task limitations...  
    if intStr == '00':  
        return ' '  
  
    k1 = ''  
    k2 = ''  
  
    if len(intStr) == 1:  
        k2 = intStr[0]  
    else:  
        k1 = intStr[0]  
        k2 = intStr[1]  
  
    row_key = int(k2) - 1  
  
    return compression_table[k1][row_key]
```

Рисунок 2 – Методи для перетворення символу в числове представлення та навпаки.

При шифрування ми перетворюємо вхідний текст та ключ в числовий вигляд. Наступним кроком є прохід по повідомленню в числовому вигляді та проведення операції суми та взяття залишку від ділення на 10 чим відповідно і кодуємо наше повідомлення, повторно використовуючи ключ як в випадку з шифром віженера.

```
def encryptDigest(plainText, keyStr):
    encodedChars = list(map(lambda x: getIntFromChar(x), plainText))

    print("Encoded chars: " + str(encodedChars))

    encodedText = ''.join(encodedChars)

    print("Encoded chars: " + str(encodedText))

    parsedKey = parseStrKey(keyStr)

    print(str(parsedKey))

    encrypted = ''

    for idx, x in enumerate(encodedText):
        strPart = int(x)
        keyPartStr = parsedKey[(idx) % len(parsedKey)]
        keyPart = int(keyPartStr)

        result = (strPart + keyPart) % 10

        encrypted = encrypted + str(result)

    tokenized_decrypted = parseIntDigest(encrypted)

    print("Tokenized encrypted: " + str(tokenized_decrypted))

    plain_text = map(lambda x: str(x), tokenized_decrypted)

    return ''.join(plain_text)
```

Рисунок 3 – Кодування тексту

Дешифрування проходить дещо складніше, адже нам потрібно перетворити отримані числові значення назад в символи. Тут як і в випадку з кодуванням перетворюємо ключ в числовий вигляд і проводимо стандартні математичні операції попарно з кожним елементом повідомлення і ключа тільки замість додавання віднімаємо ключ від повідомлення і беремо остаток від ділення. Далі нам необхідно перетворити розкодоване повідомлення в символний вигляд, Для цього ми проходимося по всьому повідомленню, і якщо зустрічаємо цифру від 1 до 7 спокійно замінюємо її на символи з першого рядка таблиці стиснення.

Якщо зустрічаються символи 0, 9 або 8 то ми їх додаємо до акумулятору і на наступному кроці циклу комбінуємо їх з новою цифрою і отримуємо відповідний символ з таблиці.

```
def decryptDigest(encrypted, keyStr):
    parsedKey = parseStrKey(keyStr)

    decrypted = ''

    for idx, x in enumerate(encrypted):
        strPart = int(x)
        keyPartStr = parsedKey[(idx) % len(parsedKey)]
        keyPart = int(keyPartStr)

        result = (10 + strPart - keyPart) % 10

        decrypted = decrypted + str(result)

    tokenized_decrypted = parseIntDigest(decrypted)

    print("Tokenized decrypted: " + str(tokenized_decrypted))

    plain_text = map(lambda x: getCharFromIntStr(str(x)), tokenized_decrypted)

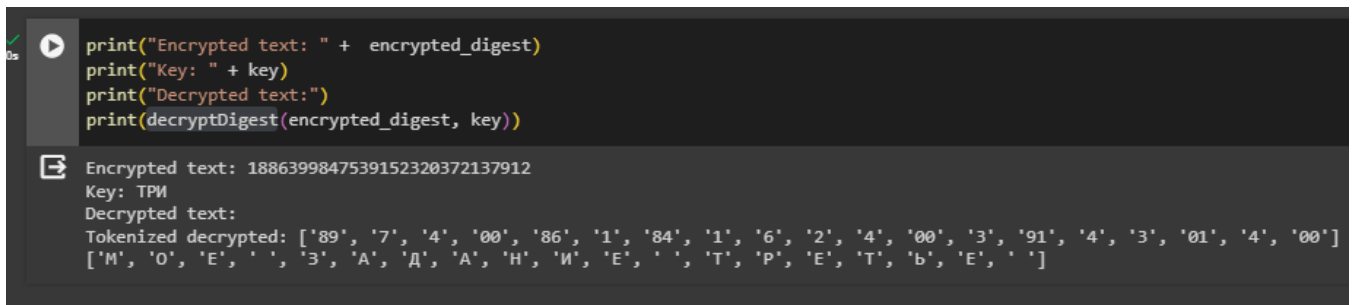
    return list(plain_text)
```

Рисунок 4 – Декодування тексту

Результати виконання

Індивідуальне завдання №1

Результат виконання індивідуального завдання №1 можна побачити на рисунку 5.



```
print("Encrypted text: " + encrypted_digest)
print("Key: " + key)
print("Decrypted text:")
print(decryptDigest(encrypted_digest, key))

Encrypted text: 1886399847539152320372137912
Key: ТРИ
Decrypted text:
Tokenized decrypted: ['89', '7', '4', '00', '86', '1', '84', '1', '6', '2', '4', '00', '3', '91', '4', '3', '01', '4', '00']
['М', 'О', 'Е', ' ', 'З', 'А', 'Д', 'А', 'Н', 'И', 'Е', ' ', 'Т', 'Р', 'Е', 'Т', 'Ь', 'Е', ' ']
```

Рисунок 5 – Результати виконання першого завдання

Розшифрований текст: 'МОЕ ЗАДАНИЕ ТРЕТЬЕ'.

Індивідуальне завдання №2

Текст команд та результат виконання індивідуального завдання №2 можна побачити нижче:

```
plainText = "МИЛЛИОН ДВЕСТИ ТЫСЯЧ ДВАДЦАТЬ ДВА"
```

```
key = "ВОСКРЕСЕНЬЕ"
```

```
encrypted = encryptDigest(plainText, key)
```

```
print("Encrypted: " + str(encrypted))
```

```
decrypted = decryptDigest(encrypted, key)
```

```
print("Decrypted: " + str(decrypted))
```

Encoded chars: ['89', '2', '88', '88', '2', '7', '6', '00', '84', '82', '4', '5', '3', '2', '00', '3', '910', '5', '04', '96', '00', '84', '82', '1', '84', '95', '1', '3', '01', '00', '84', '82', '1']

Encoded chars: 89288882760084824532003910504960084821849513010084821

827587914546014

Tokenized encrypted: ['6', '1', '93', '6', '5', '7', '3', '1', '1', '4', '6', '85', '2', '06', '2', '80', '7', '94', '3', '6', '4', '1', '05', '3', '4', '2', '7', '3', '2', '5', '1', '2', '2', '93', '1', '1', '4', '4', '92', '7', '3', '2', '5', '1', '2']

Encrypted: 61936573114685206280794364105342732512293114492732512

Tokenized decrypted: ['89', '2', '88', '88', '2', '7', '6', '00', '84', '82', '4', '5', '3', '2', '00', '3', '91', '05', '04', '96', '00', '84', '82', '1', '84', '95', '1', '3', '01', '00', '84', '82', '1']

Decrypted: ['М', 'И', 'Л', 'Л', 'И', 'О', 'Н', ' ', 'Д', 'В', 'Е', 'С', 'Т', 'И', ' ', 'Т', 'Р', 'Й', 'Я', 'Ч', ' ', 'Д', 'В', 'А', 'Д', 'Ц', 'А', 'Т', 'Ь', ' ', 'Д', 'В', 'А']