



МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. ІГОРЯ
СІКОРСЬКОГО”

КРИПТОГРАФІЯ
КОМП'ЮТЕРНИЙ ПРАКТИКУМ №2
Криптоаналіз шифру Віженера

Виконали:
Студенти групи ФБ-22
Орлов Антон, Ялбуган Федір
(бригада 7)

КИЇВ 2024

Мета роботи

Засвоєння методів частотного криптоаналізу. Здобуття навичок роботи та аналізу поточкових шифрів гамування адитивного типу на прикладі шифру Віженера.



Порядок виконання роботи

0. Уважно прочитати методичні вказівки до виконання комп'ютерного практикуму.
1. Самостійно підібрати текст для шифрування (2-3 кб) та ключі довжини $r = 2, 3, 4, 5$, а також довжини 10-20 знаків. Зашифрувати обраний відкритий текст шифром Віженера з цими ключами.
2. Підрахувати індекси відповідності для відкритого тексту та всіх одержаних шифртекстів і порівняти їх значення.
3. Використовуючи наведені теоретичні відомості, розшифрувати наданий шифртекст (згідно свого номеру варіанта).

Хід роботи

1

Ми взяли шматок тексту з попередньої лабораторної роботи розміром 3КБ.

Имя	Дата изменения	Тип	Размер
 cleaned_bible_no_spaces.txt	22.10.2024 16:09	Текстовый докум...	3 КБ
 ...	22.10.2024 16:10	Текстовый докум...	3 КБ

Далі підібрали ключі відповідних довжин:

```
keys = ["ад", "рай", "вера", "иисус", "судныйдень", "триединство",  
"апостолиоанн", "игорьандрущак", "архангелмихаил"]
```

Далі ми зашифрували текст з цими ключами і створили окремий файл для кожного ШТ:

```
def text_to_shifts(key, alphabet):
    return [alphabet.index(char) for char in key if char in alphabet]

def vigenere_encrypt(text, key_shifts, alphabet):
    encrypted_text = []
    key_length = len(key_shifts)

    for i, char in enumerate(text):
        if char in alphabet:
            shift = key_shifts[i % key_length]
            char_index = alphabet.index(char)
            new_index = (char_index + shift) % len(alphabet)
            encrypted_text.append(alphabet[new_index])
        else:
            encrypted_text.append(char)

    return ''.join(encrypted_text)

def encrypt_text_with_keys(input_file, keys, alphabet):

    with open(input_file, 'r', encoding='utf-8') as file:
        text = file.read().strip()

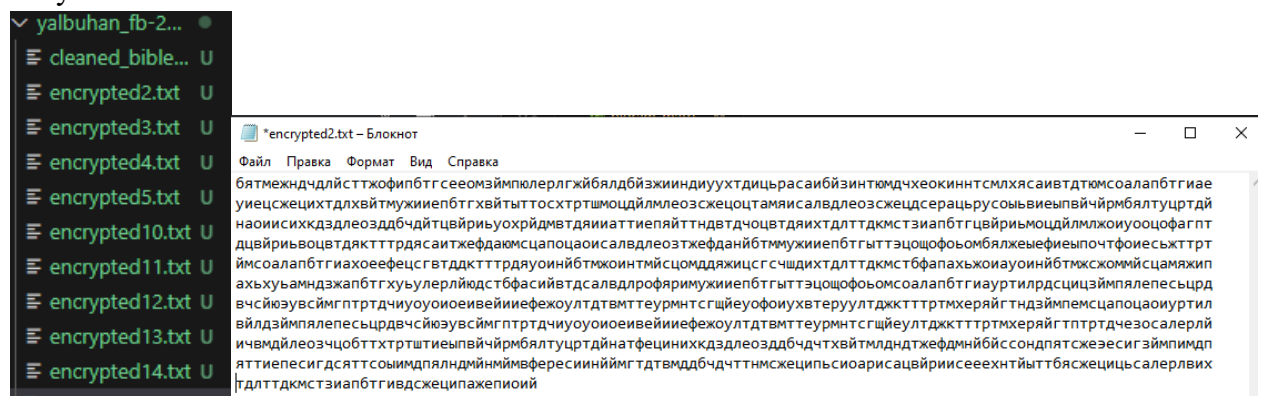
    for key in keys:
        key_shifts = text_to_shifts(key, alphabet)
        encrypted_text = vigenere_encrypt(text, key_shifts, alphabet)

        output_filename = f'encrypted{len(key)}.txt'
        with open(output_filename, 'w', encoding='utf-8') as output_file:
            output_file.write(encrypted_text)

alphabet = 'абвгдезийклмнопрстуфхцшщъыьэя'
keys = ["ад", "рай", "вера", "иисус", "судныйдень", "триединство",
        "апостолиоанн", "игорьяндрущак", "архангелмихаил"]
input_file = 'cleaned_bible_no_spaces.txt'

encrypt_text_with_keys(input_file, keys, alphabet)
```

Результат:



Підрахований індекс ВТ (IC: 0.060727478203206356):

```
def calculate_ic_from_text(input_file):
    with open(input_file, 'r', encoding='utf-8') as file:
        text = file.read()

    symbol_counts = Counter(text)

    n = len(text)

    numerator = sum(count * (count - 1) for count in symbol_counts.values())
    denominator = n * (n - 1)
    IC = numerator / denominator if denominator != 0 else 0

    return IC

IC = calculate_ic_from_text(input_file)
print(f"IC: {IC}")
```

Python

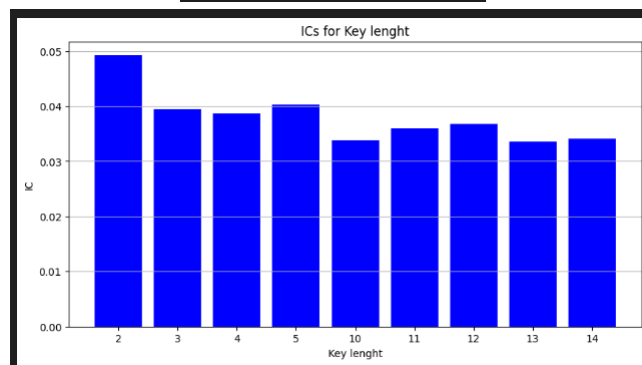
IC: 0.060727478203206356

Обраховували по формулі:

$$I(Y) = \frac{1}{n(n-1)} \sum_{i \in Z_m} N_i(Y)(N_i(Y)-1),$$

Після застосування цієї функції до шифротекстів, отримали такі Індeksi Відповідності:

```
IC2: 0.04927063826740202
IC3: 0.03942180317584848
IC4: 0.03864504188128784
IC5: 0.04024370691037358
IC10: 0.03387550960366494
IC11: 0.03604079008609753
IC12: 0.03677707884827626
IC13: 0.03350970017636684
IC14: 0.034123014705538975
```



Беремо ШТ свого варіанту. Використовуємо цей алгоритм для знаходження довжини ключа:

- 1) Для кожного кандидата $r = 2, 3, \dots$ розбити шифртекст Y на блоки Y_1, Y_2, \dots, Y_r .
- 2) Обчислити значення індексу відповідності для кожного блоку.
- 3) Якщо сукупність одержаних значень схиляється до теоретичного значення I для даної мови, то значення r вгадане вірно. Якщо сукупність значень схиляється до значення $I_0 = \frac{1}{m}$, що відповідає мові із рівноімовірним алфавітом, то значення r вгадане неправильно.

В кодї це реалізовано так:

```
def calculate_single_IC(text_block):

    freaks = Counter(text_block)
    n = len(text_block)

    if n == 0:
        return 0

    index = sum(f * (f - 1) for f in freaks.values()) / (n * (n - 1))
    return index

def splitting_func(text, key_length):
    blocks = ['' for _ in range(key_length)]
    for i, char in enumerate(text):
        blocks[i % key_length] += char
    return blocks

def calculate_ICs(file_path, min_key_length=2, max_key_length=30):
    with open(file_path, 'r', encoding='utf-8') as file:
        text = file.read().replace('\n', '').replace(' ', '')

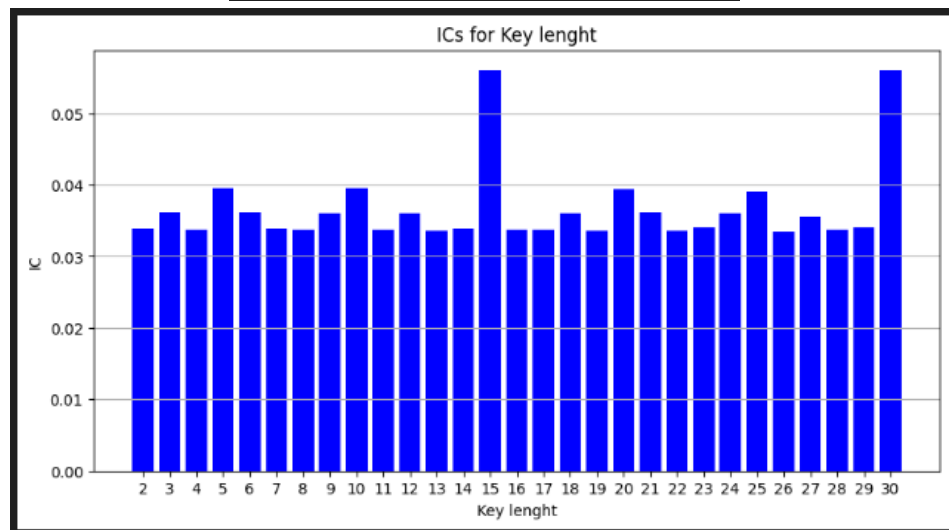
    for key_length in range(min_key_length, max_key_length + 1):
        blocks = splitting_func(text, key_length)
        indices = [calculate_single_IC(block) for block in blocks]

        average_index = sum(indices) / len(indices)
        print(f"Key length: {key_length}, IC: {average_index:.4f}")

calculate_ICs('var_text.txt')
```

Отримуємо такі значення IC, вибираємо ті, які найближчі до індексу відповідності ВТ:

```
... Key length: 2, IC: 0.0339  
Key length: 3, IC: 0.0362  
Key length: 4, IC: 0.0337  
Key length: 5, IC: 0.0395  
Key length: 6, IC: 0.0361  
Key length: 7, IC: 0.0338  
Key length: 8, IC: 0.0337  
Key length: 9, IC: 0.0361  
Key length: 10, IC: 0.0395  
Key length: 11, IC: 0.0337  
Key length: 12, IC: 0.0360  
Key length: 13, IC: 0.0336  
Key length: 14, IC: 0.0339  
Key length: 15, IC: 0.0561  
Key length: 16, IC: 0.0337  
Key length: 17, IC: 0.0337  
Key length: 18, IC: 0.0361  
Key length: 19, IC: 0.0335  
Key length: 20, IC: 0.0394  
Key length: 21, IC: 0.0362  
Key length: 22, IC: 0.0336  
Key length: 23, IC: 0.0341  
Key length: 24, IC: 0.0361  
Key length: 25, IC: 0.0390  
Key length: 26, IC: 0.0335  
Key length: 27, IC: 0.0356  
Key length: 28, IC: 0.0337  
Key length: 29, IC: 0.0340  
Key length: 30, IC: 0.0560
```



Звідси випливає, що наш ключ скоріш за все буде довжиною 15 або 30. Перевіримо це за допомогою наступного коду. Ділимо на блоки для довжини ключа 15, обраховуємо зсув між найпоширенішою буквою кожного блоку та буквою «о» (найпоширенішою буквою відкритого тексту російською мовою). Ці числа, що виходять, будуть індексами букв нашого ключа.

```
def key_finder(file_path, key_length):
    with open(file_path, 'r', encoding='utf-8') as file:
        text = file.read().replace('\n', '').replace(' ', '')
        blocks = splitting_func(text, key_length)
        key = [
            chr((ord(Counter(block).most_common(1)[0][0]) - ord('a')) % 32 + ord('a'))
            for block in blocks
        ]

    return ''.join(key)

key = key_finder('var_text.txt', 15)
print(key)
```

✓ 0.0s

Python

арудазевархимаг

Отримали такий ключ: **арудазевархимаг**

Одразу бачимо слово «архимаг», яке стане основою для пошуку справжнього ключа. Ми вирішили здогадатися яким буде повний ключ, і знайшли книгу Олександра Рудазова «Архімаг», тому повним ключем нашого шифротексту буде **арудазовархимаг** (майже те саме, що видав код)).

Перевіримо це інвертованою до *vigenere_encrypt()* функції з першого пункту завдання:

```
def vigenere_decrypt(ciphertext, key):
    alphabet = 'абвгдежзийклмнопрстуфхцщъьэя'
    decrypted_text = []
    key = key.lower()
    key_length = len(key)

    ciphertext = ''.join([c for c in ciphertext.lower() if c in alphabet])

    for i, char in enumerate(ciphertext):
        if char in alphabet:
            text_index = alphabet.index(char)
            key_index = alphabet.index(key[i % key_length])
            decrypted_index = (text_index - key_index) % len(alphabet)
            decrypted_text.append(alphabet[decrypted_index])
        else:
            decrypted_text.append(char)

    return ''.join(decrypted_text)

with open('var_text.txt', 'r', encoding='utf-8') as file:
    ciphertext = file.read().replace('\n', '').replace(' ', '')
key = "арудазовархимаг"
```

```
decrypted_message = vigenere_decrypt(ciphertext, key)
print("Decrypted text:", decrypted_message)
```

У виводі бачимо знайдений ВТ:

```
Decrypted text: прошлопятнадцатьднейистарыйдомпостепенноначаложиватьсороклетвнемниктонеж
```

Можна навіть знайти, що цей текст було взято з 8 глави з цієї книги:

Александр Рудазов Архимар

...и, разумеется... и просто оторву ему еще пару палцев...

Креол кивнул на все еще пылающую в углу пентаграмму и мрачно усмехнулся.

ГЛАВА 8

Мы наш, мы новый мир построим!

Рамзес II

Прошло пятнадцать дней, и старый дом постепенно начал оживать. Сорок лет в нем никто не жил по-настоящему. За это время он сменил одиннадцать хозяев, но никто из них не выдерживал в подобном месте больше трех месяцев. Креол и Ванесса стали двенадцатыми.

Маг полностью погрузился в работу. Он отрывался только затем, чтобы поесть, а от сна избавлялся заклятием Бессонницы. Но для Креола это явно не проходило безнаказанно — глаза у него покраснели, а веки набрякли и отвисли. Ванесса всячески старалась убедить его в том, что ему следует прекратить издевательства над организмом и хоть разок выспаться по-настоящему, но маг только огрызался. Занимался он двумя делами: неутомимо писал магическую книгу и окутывал особняк магической защитой. И то и другое требовало уймы времени, а Креол никак не мог решить, что

Висновки: Підрахунок індексів відповідності для відкритого тексту та всіх шифртекстів дав змогу зробити висновки щодо того, як збільшення довжини ключа впливає на ентропію та структуру шифртексту. Зазвичай, чим довший ключ, тим більше шифртекст наближається до випадковості, що ускладнює його розшифрування.

Використовуючи теоретичні методи криптоаналізу, вдалося успішно розшифрувати шифртекст, отриманий за зашифрованим методом Віженера. Це продемонструвало ефективність індексу відповідності у визначенні довжини ключа та подальшого

розшифрування тексту. Завдання сприяло розумінню принципів роботи шифру Віженера та методів криптоаналізу для його зламу.