НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

КРИПТОГРАФІЯ

КОМП’ЮТЕРНИЙ ПРАКТИКУМ №2

Криптоаналіз шифру Віженера

Виконали:
студенти групи ФБ-22
Мартинюк Артем
Шеїна Еліна

Мета роботи:

Засвоєння методів частотного криптоаналізу. Здобуття навичок роботи та аналізу потокових 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 шифротекстів і порівняти їх значення.

3. Використовуючи наведені теоретичні відомості, розшифрувати наданий шифротекст (згідно свого номеру варіанта).

Хід роботи

Для першого завдання ми скористаємося текстом із попередньої роботи. Маємо список із ключів:

```
keys_list = ["да", "кто", "литр", "слово", "дальспщнаб", "чиназесчина", "саунтресчина", "детиголодаютт", "зачемяэтоделаю", "помогитеявзалож"]
```

У коді ми проходимося по всім ключам із цього списку. Функція шифрування відкритого тексту за допомогою шифру Віженера:

```
def enc_vig(plain_text, cipher_key):
    result = []
    cipher_key = cipher_key.lower()
    key_size = len(cipher_key)
    key_pos = 0

    for symbol in plain_text:
        if 'а' <= symbol <= 'я':
            offset = ord(cipher_key[key_pos % key_size]) - ord('а')
            encrypted_symbol = chr((ord(symbol) - ord('а') + offset) % 32 + ord('а'))
            result.append(encrypted_symbol)
            key_pos += 1

    return ''.join(result)
```

Для кожної букви ключа послідовно визначається величина зсуву. Потім, використовуючи цей зсув, розраховується номер зашифрованої букви у відкритому тексті, а результат береться за модулем 32, щоб залишитися в межах алфавіту. Особливість у тому, що довжина ключа може не збігатися з

довжиною відкритого тексту, і це враховується завдяки використанню індексу букви ключа за модулем.
Функція дешифрування:

```
def dec_vig(encrypted_text, cipher_key):
    result = []
    cipher_key = cipher_key.lower()
    key_size = len(cipher_key)
    key_pos = 0

    for symbol in encrypted_text:
        if 'а' <= symbol <= 'я':
            offset = ord(cipher_key[key_pos % key_size]) - ord('а')
            decrypted_symbol = chr((ord(symbol) - ord('а') - offset) % 32 + ord('а'))
            result.append(decrypted_symbol)
            key_pos += 1

    return ''.join(result)
```

Виконуються ті ж дії, але зсув виконується в інший бік.

Індекс відповідності - метод криптоаналізу шифру Віженера. Обраховується за формулою:

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

Якщо текст не зашифрований, то індекс відповідності набуватиме значень випадковим чином. Частота появи різних букв у тексті буде відповідати їхній загальній імовірності в появі у відкритому тексті. У такому випадку індекс відповідності дорівнюватиме:

$$MI(Y) = \sum_{t \in Z_m} p_t^2$$

де t p – імовірність появи літери t в мові.

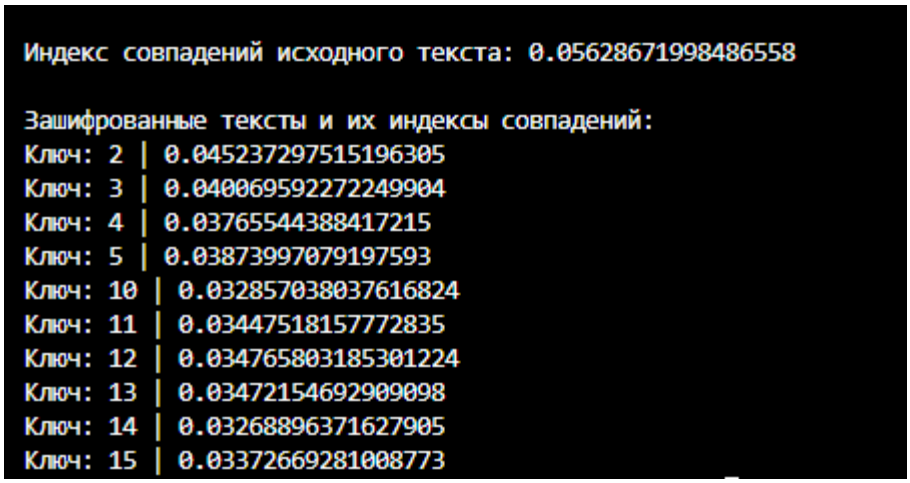
Обрахунок індексу відповідності реалізований наступним чином:

```
def calculate_ic(input_text):
    total_chars = len(input_text)
    if total_chars == 0:
        return 0

    char_frequencies = Counter(input_text)
    ic_value = sum(freq * (freq - 1) for freq in char_frequencies.values()) / (total_chars * (total_chars - 1))
    return ic_value
```

Виконується проходження по всіх буквах відкритого тексту з розрахунком їхніх індексів відповідності. Частота появи кожної букви ділиться на загальну кількість букв у тексті. Потім індекси відповідності всіх букв підсумовуються, утворюючи загальний індекс відповідності для всього

тексту.
Після застосування цієї функції до відкритого тексту отримано результат:



У наступному завданні нам надано шифротекст, і потрібно експериментально визначити ймовірну довжину ключа, використовуючи аналіз на основі індексів відповідності. З попереднього завдання відомо, що індекс відповідності для відкритого тексту російською мовою приблизно дорівнює 0.05628.

Оскільки ми розглядаємо шифр Віженера, де ключ повторюється до тих пір, поки його довжина не дорівнює довжині відкритого тексту, через певний період, рівний довжині ключа, можна помітити, що зсув букв відбувається на один і той самий індекс. Тому доцільно розбити шифротекст на блоки, що відповідають цьому періоду. Наприклад, якщо період дорівнює 2, слово "калейдоскоп" буде розбито на блоки "клйдскп" і "аеооо".

Обчисливши індекс відповідності для кожного блоку, можна визначити довжину ключа. Це стає можливим, оскільки для блоків, які містять букви, до яких застосовано однаковий зсув, індекс відповідності наближатиметься до нормального значення, тобто приблизно до 0.05628.

Такий принцип знаходження ключа було продемонстровано у трьох функціях нижче:

```python
def divide_text_into_blocks(input_text, block_size):
    divided_blocks = [''] * block_size
    for index, character in enumerate(input_text):
        divided_blocks[index % block_size] += character
    return divided_blocks

def average_ic_for_block_size(input_text, block_size):
    blocks = divide_text_into_blocks(input_text, block_size)
    ic_values = [calculate_ic(block) for block in blocks]
    return sum(ic_values) / len(ic_values)

def identify_key_length(input_text, max_block_size=30):
    ic_results = {}
    for block_size in range(2, max_block_size + 1):
        ic_value = average_ic_for_block_size(input_text, block_size)
        ic_results[block_size] = ic_value
        print(f"Размер блока: {block_size}, IC: {ic_value}")

    return ic_results
```

Таким чином, ми проаналізували ключі довжини від 2 до 30:

```
Размер блока: 2, IC: 0.03432921421542369
Размер блока: 3, IC: 0.03734839112182639
Размер блока: 4, IC: 0.03846786795894798
Размер блока: 5, IC: 0.032753684507439526
Размер блока: 6, IC: 0.04242249836150345
Размер блока: 7, IC: 0.032845671625834745
Размер блока: 8, IC: 0.038394305262087654
Размер блока: 9, IC: 0.037406913486166676
Размер блока: 10, IC: 0.034343106655826135
Размер блока: 11, IC: 0.03282596004503103
Размер блока: 12, IC: 0.05436955673586635
Размер блока: 13, IC: 0.032807635112857336
Размер блока: 14, IC: 0.034253133094361496
Размер блока: 15, IC: 0.03741441107403287
Размер блока: 16, IC: 0.03846816039387033
Размер блока: 17, IC: 0.0326076877752591
Размер блока: 18, IC: 0.042619239781400246
Размер блока: 19, IC: 0.03299852287693898
Размер блока: 20, IC: 0.03839407833306634
Размер блока: 21, IC: 0.03734596917614833
Размер блока: 22, IC: 0.03436346417856434
Размер блока: 23, IC: 0.03248823743567128
Размер блока: 24, IC: 0.05435416649918132
Размер блока: 25, IC: 0.032517536103743
Размер блока: 26, IC: 0.03434857665414954
Размер блока: 27, IC: 0.03762500312229972
Размер блока: 28, IC: 0.0383860390427654
Размер блока: 29, IC: 0.033132183908045974
Размер блока: 30, IC: 0.04250450051229374

Наиболее вероятная длинна ключа: [12]
```

Тепер спробуємо знайти сам ключ:

```python
def derive_key(cipher_text, key_size):
    groups = divide_into_groups(cipher_text, key_size)
    derived_key = []

    for group in groups:
        char_freqs = Counter(group)
        most_frequent_char = char_freqs.most_common(1)[0][0]
        offset = (ord(most_frequent_char) - ord('о')) % 32
        derived_key.append(chr(ord('а') + offset))

    return ''.join(derived_key)
```

Розбиваємо зашифрований текст на блоки відповідно до періоду, у цьому випадку для довжин 12 та 24. В кожному блоці визначаємо букву, яка з'являється найчастіше. Припускаємо, що ця буква відповідає найпоширенішій букві відкритого тексту, тобто "о". Потім обчислюємо зсув між найчастіше зустрічаючою буквою блоку шифротексту та буквою "о" у відкритому тексті, переводимо його в індекс і визначаємо відповідну букву ключа для кожного блоку. Таким чином, проходимо по всіх блоках. Спробуємо застосувати цей підхід для ключів довжиною 12 та 24:

```
Найденный ключ для длины 12: вшебспирбуря
```

```
Найденный ключ для длины 24: вшебипизшурявшеьспирбтря
```

Як бачимо, ключ довжиною 24 є просто подвоєним ключем довжиною 12, тому надалі ми будемо використовувати лише ключ 12. Ураховуючи, що ключ має бути змістовним, а також результат, який ми отримали при спробах знайти ключі довжиною 12 та 24, шебспир занадто сильно схоже на шекспир. Тому ключ буде "вшекспирбуря".

Спробуємо розшифрувати текст, використовуючи цей ключ:

Найденный ключ для длины 12: вшебспирбуря

Расшифрованный текст с ключом длиной 12:

действующиелицаалонзокорольнеаполитанскийсебастьянегобратпросперозаконныйгерцогмиланскийантониоегобратнезаконнозахватившийвластьвмиланскомгерцогствефердинандсынкоролянеаполитанскогогонзалостарыйчестныйсоветниккоролянеаполитанскогоадрианфрансископридворныекалибанрабуродливыйдикарьтринкулошутстефанодворецкийпьяницакапитанкораблябоцманматросымирандадочьпроспероариэльдухвоздухаиридацереraюнонанимфыжнец

ыдухидругиедухипокорныепроспероместодействиякорабльвмореостровкораб
львморебурягромимолниявходяткапитанкорабляибоцманкапитанбоцманбоцм
анслушаюкапитанкапитанзовикомандунаверхживейзаделонетомыналетимнар
ифыскорейскорейкапитануходитпоявляютсяматросыбоцманэймолодцывеселе
йребятавеселейживоубратьмарсельслушайкапитанскийсвистокнутеперьветер
тебепросторнодуйпоканелопнешьвходяталонзосебастьянантониоферди нандго
нзалоидругиеалонзодобрыйбоцманмыполагаемсянатебяагдекапитанмужайтес
ьдрузьябоцмананукаотправляйтесьвнизантониобоцмангдекапитанбоцманавам
егонеслышночтоливынаммешаетеотправляйтесьвкаютывидитештормразыгра
лсяатутещевыгонзалополегчелюбезныйусмирисьбоцманкогдаусмиритсяморе у
бирайтесьэтимревущимваламнетделадокоролеймаршпокаютаммолчатьнемеш
айтегонзаловсетакипомнилюбезныйктоутебянабортубоцманаяпомнючтонетни
когочьяшкура…

Текст вдалося розшифрувати.

Висновки:

У процесі цієї роботи ми ознайомилися з методами частотного криптоаналізу. Експериментально визначили індекс відповідності для відкритого тексту російською мовою, порівняли його з індексами відповідності зашифрованих текстів та виявили залежність цього індексу від довжини ключа.

Використовуючи отримані знання, ми змогли визначити довжину ключа для конкретного зашифрованого тексту, а також відновити сам ключ, що дозволило успішно дешифрувати текст.