

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

**КРИПТОГРАФІЯ
КОМП'ЮТЕРНИЙ ПРАКТИКУМ №2**

Криптоаналіз шифру Віженера

Виконали роботу:
студент ФБ-23 Гнидюк Данііл
студент ФБ-23 Жушман Ілля

Київ 2024

Мета роботи

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

Варіант: 5

Порядок виконання роботи

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

Спочатку ми очистили текст, використовуючи функцію *clean_text*. Ця функція дозволила залишити лише російські літери та перевести їх у нижній регістр. Для цього ми зчитали текст із вхідного файлу, видалили зайві символи за допомогою регулярного виразу та зберегли результат у новий файл.

```
lab2.1.py

def clean_text(file_path, output_path):
    """
    Очищує текст у файлі, залишаючи лише російські літери та перетворюючи
    їх у нижній регістр.
    """
    file_encoding = detect_file_encoding(file_path)

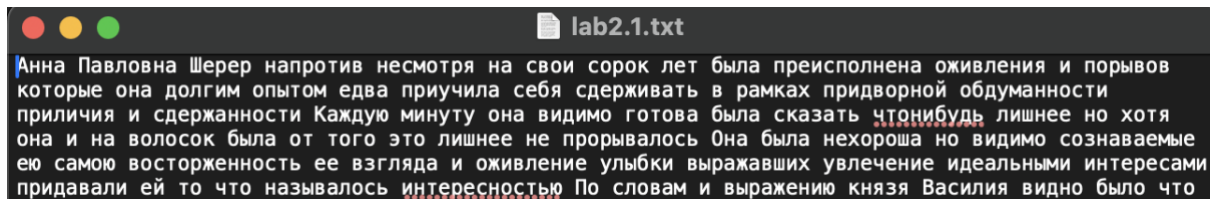
    with open(file_path, 'r', encoding=file_encoding) as file:
        text = file.read()

    cleaned_text = re.sub(r'^[а-яё]', '', text.lower())

    with open(output_path, 'w', encoding='utf-8') as file:
        file.write(cleaned_text)

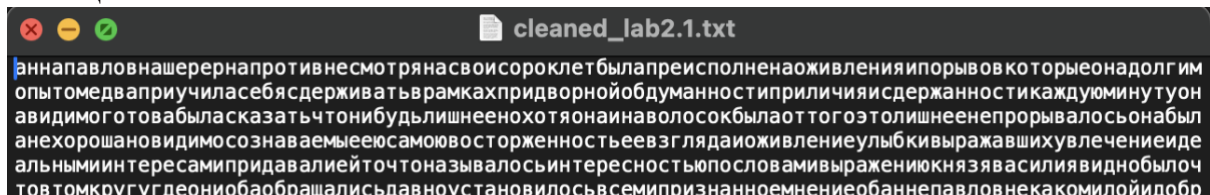
    print(f"Очищений текст збережено у файл: {output_path}")
    return cleaned_text
```

Початковий текст:



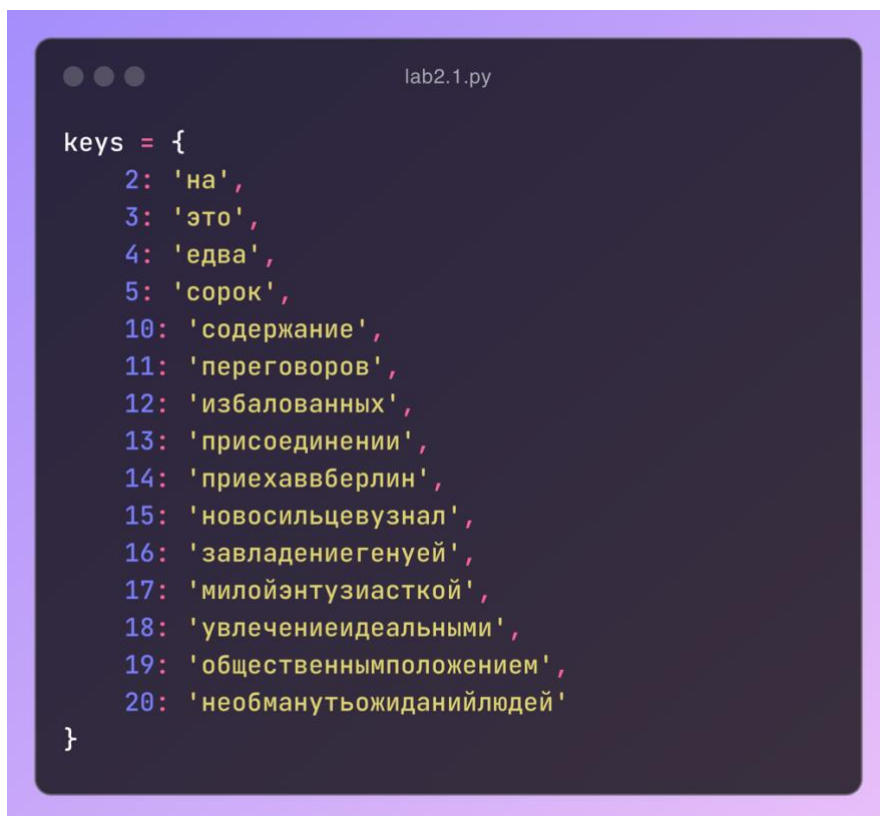
```
lab2.1.txt
Анна Павловна Шерер напротив несмотря на свои сорок лет была преисполнена оживления и порывов
которые она долгим опытом едва приучила себя сдерживать в рамках придворной обдуманности
приличия и сдержанности Каждую минуту она видимо готова была сказать чтонибудь лишнее но хотя
она и на волосок была от того это лишнее не прорывалось Она была нехороша но видимо сознаваемые
ею самую восторженность ее взгляда и оживление улыбки выражавших увлечение идеальными интересами
придавали ей то что называлось интересностью По словам и выражению князя Василия видно было что
```

Очищенный текст:



```
cleaned_lab2.1.txt
АннаПавловнаШерернапротивнесмотрянасвоисороклетбылапреисполненаоживленияипорывовкоторыеонадолгим
опытомедваприучиласебасдерживатьврамкахпридворнойобдуманностиприличияисдержанностикаждуюминутуона
авидимогоготовабиласказатьчтонибудьлишнеенохотяонаинаволосокбылаоттогоэтолишнеенепрывалосьонабыл
анехорошановидимосознаваемыееюсамуювосторженностьеевзглядаиоживлениеулыбкивыражавшихувлечениеиде
альнымиинтересамипридавалиейточноназывалосьинтересностьюпословамивыражениюкнязясасилиявиднобылоч
товтомкругугдеониобаобращалисьдавноустановилосьвсемипризнанноемнениеобаннепавловнекакомилويدобр
```

Далі ми обрали ключі для шифрування. Ми використовували осмислені слова різної довжини: від коротких ключів (2, 3, 4, 5 символів) до довгих (10–20 символів). Ключі були підібрані так, щоб продемонструвати, як довжина ключа впливає на результат шифрування



```
lab2.1.py
keys = {
    2: 'на',
    3: 'это',
    4: 'едва',
    5: 'сорок',
    10: 'содержание',
    11: 'переговоров',
    12: 'избалованных',
    13: 'присоединении',
    14: 'приехаввберлин',
    15: 'новосильцевузнал',
    16: 'завладениегенуей',
    17: 'милойэнтузиасткой',
    18: 'увлечениеидеальными',
    19: 'общественнымположением',
    20: 'необманутьожиданийлюдей'
}
```

Після підготовки тексту та вибору ключів ми перейшли до процесу шифрування. Для цього використали функцію `vigenere_encrypt`, яка реалізує шифр Віженера. В основі цієї функції лежить ідея зсуву символів алфавіту відповідно до індексів символів ключа. Функція по черзі

обробляє кожен символ тексту та здійснює шифрування залежно від ключа.

```
lab2.1.py

def vigenere_encrypt(text, key):
    """
    Шифрує текст за допомогою шифру Віженера
    """
    alphabet = 'абвгдеёжзийклмнопрстуфхцчщъыьэюя'
    alphabet_size = len(alphabet)
    encrypted_text = []

    key_indices = [alphabet.index(k) for k in key]
    key_length = len(key)

    for i, char in enumerate(text):
        if char in alphabet:
            text_index = alphabet.index(char)
            shift = key_indices[i % key_length]
            encrypted_char = alphabet[(text_index + shift) %
alphabet_size]
            encrypted_text.append(encrypted_char)
        else:
            encrypted_text.append(char)

    return ''.join(encrypted_text)
```

Ця функція дозволила зашифрувати очищений текст за кожним із вибраних ключів. Результат шифрування для кожного ключа ми зберегли у окремі файли, що дали змогу проаналізувати залежність шифру від довжини ключа.

Результат:

```
encrypted_with_key_length_2.txt

нныаэаплъываёеюеннпюоаипнтсьоармннспоцсьрькшеабилнпюецсэоштнтнофилтнцяцпъривъвшоаюытоыасощгцм
ьпитъмтдпаэрцуеищаяеоядтर्फипааьпрнмшаг пюисвърьочоодбмнныоатцпюишиимиядтर्फаыньсаишафдбюиыуауьн
нвцдцмьгътъвнбилнсшахааьетьнцббдйлцшыетнхътмоыацннвъльськоыщаьтаороктьлцшыетнтпюоюыпашоаяьнбил
ннтхърьшннвъдцмьсьзыапатмиетюяаьолвьсаоюжтныоатйетвхгцясацофилтнцбблибишпюафапшцхбвъеееыитисе
```

Для перевірки правильності шифрування ми можемо скористатися онлайн-ресурсами

Текст

аннапавловнашерернапротивнесмотрянасоисороклетбылапреисполне
наоживленияипорывовкоторыеонадолгимопытомедваприучиласебясде
рживатьврамкахпридворнойобдуманностиприличияисдержанностикак
дуюминутуонавидимоготовабыласказатьчтонибудьлишнеенохотяонаина
волосокбылаоттогоэтолишнеенепрорывалосьонабыланехорошановиди
мосознаваемыееюсамоювосторженностьюеевзглядаиюживлениенулыбкив

Ключ
на

Преобразование
☒ Зашифровать
☐ Расшифровать

Алфавит
Русский

РАССЧИТАТЬ

Преобразованный текст

нныазаплываёеюеюннпюоаипнтсьоармннспоцсьрькщеабилнпюецсэощтнн
офиплтнцяцпъривъвшоаюутоыасощгцмьпйтмтдпаэрцуеищаяеоядтрфипаа
ьпрнмшагпюисвьрыочоодбмнныоятцпюищиеимиядтрфаыньсаишафдбюиыу
ауьннвдцмьгтьвнбилнсшахааьетьнцббдйлцшыетньхътмоыацннвьльськоыщ

Підрахувати індекси відповідності для відкритого тексту та всіх одержаних шифртекстів

Функція *index_of_coincidence* обчислює індекс відповідності тексту, який показує ймовірність випадкового співпадіння символів у ньому. Для цього вона підраховує частоти символів у тексті, використовуючи Counter, і застосовує формулу, що враховує кількість пар однакових символів відносно загальної кількості пар у тексті. Якщо текст занадто короткий ($n \leq 1$), індекс встановлюється рівним нулю.

```

lab2.2.py

def index_of_coincidence(text):
    """
    Обчислює індекс відповідності за заданою формулою.
    """
    n = len(text)
    if n <= 1:
        return 0

    frequencies = Counter(text)

    # Розрахунок IC за формулою
    numerator = sum(f * (f - 1) for f in frequencies.values())
    denominator = n * (n - 1)
    ic = numerator / denominator

    return ic

```

Результат:

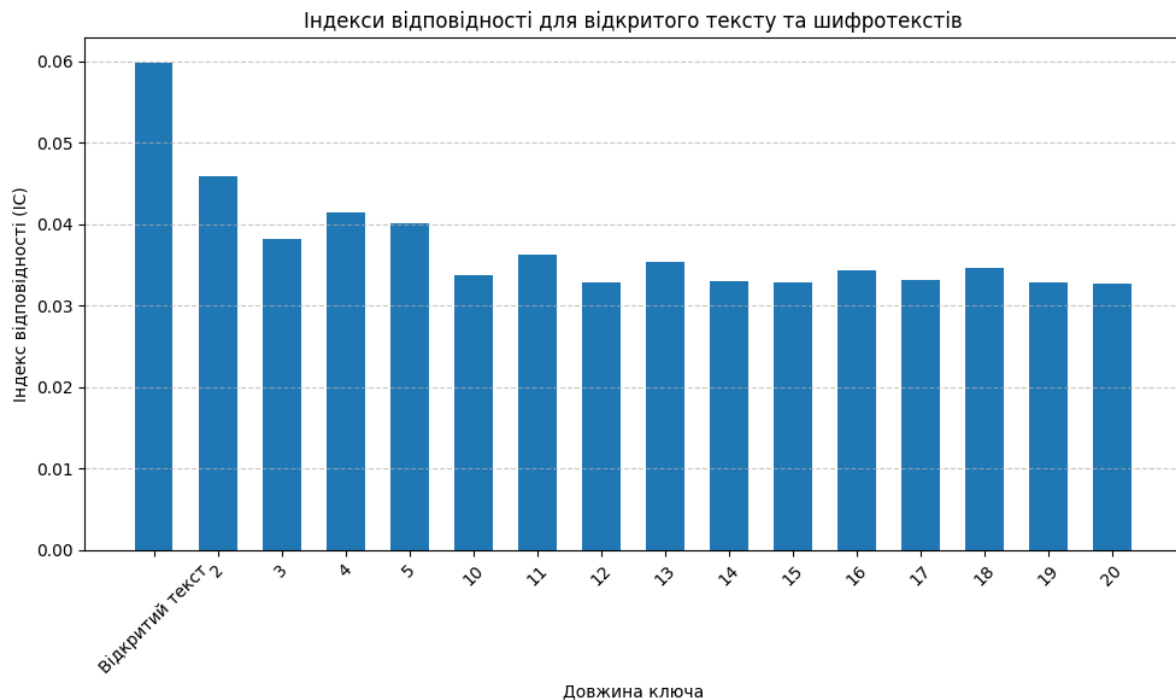
```

[daniil05@MacBook-Pro-DAniil05 lab2 % python3 lab2.2.py
Індекс відповідності для відкритого тексту: 0.05988
Індекс відповідності для шифротексту (ключ довжини 2): 0.04589
Індекс відповідності для шифротексту (ключ довжини 3): 0.03816
Індекс відповідності для шифротексту (ключ довжини 4): 0.04145
Індекс відповідності для шифротексту (ключ довжини 5): 0.04015
Індекс відповідності для шифротексту (ключ довжини 10): 0.03376
Індекс відповідності для шифротексту (ключ довжини 11): 0.03631
Індекс відповідності для шифротексту (ключ довжини 12): 0.03287
Індекс відповідності для шифротексту (ключ довжини 13): 0.03535
Індекс відповідності для шифротексту (ключ довжини 14): 0.03293
Індекс відповідності для шифротексту (ключ довжини 15): 0.03287
Індекс відповідності для шифротексту (ключ довжини 16): 0.03440
Індекс відповідності для шифротексту (ключ довжини 17): 0.03308
Індекс відповідності для шифротексту (ключ довжини 18): 0.03459
Індекс відповідності для шифротексту (ключ довжини 19): 0.03292
Індекс відповідності для шифротексту (ключ довжини 20): 0.03265

```

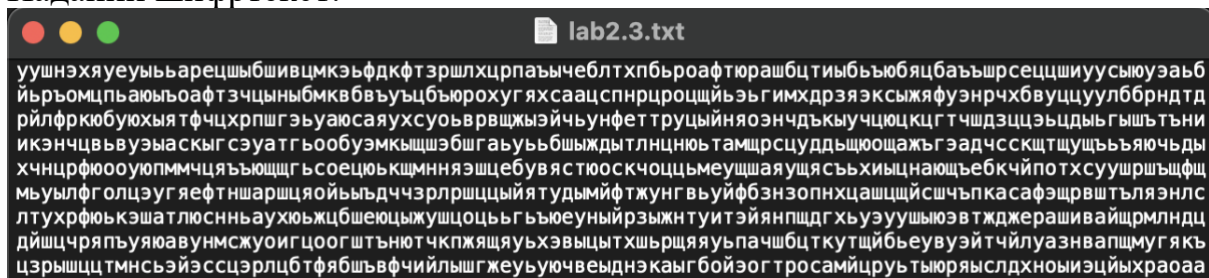
Порівняння індексів відповідності показує, що для відкритого тексту ІС становить 0.05988, що близько до теоретичного значення для природного тексту (~0.055). Для шифротекстів із короткими ключами (2-5 символів) ІС поступово зменшується (0.04589–0.04015), але залишається вищим за випадковий рівень (~0.033), що свідчить про залишкову структуру через

повторення ключа. Для довгих ключів (10-20 символів) ІС наближається до рівня випадкового тексту (0.03376–0.03265), що вказує на ефективніше приховування закономірностей тексту. Отже, довші ключі забезпечують кращу криптографічну стійкість, ніж короткі.



Далі ми розшифровували наданий шифртекст

Наданий шифртекст:



Функція *determine_key_length* використовується для визначення оптимальної довжини ключа у шифрі Віженера. Вона аналізує індекси відповідності (ІС) для тексту, розбитого на блоки різної довжини (від 2 до максимального значення), і обчислює середній ІС для кожної довжини ключа. Отримані значення порівнюються з теоретичним ІС природного тексту (0.0557), а довжина, для якої різниця є найменшою, вважається оптимальною.

```

lab2.3.py

def determine_key_length(text, max_key_length=30):
    """
    Знаходить оптимальну довжину ключа шляхом аналізу індексів
    відповідності.
    """
    optimal_length = 0
    minimal_difference = float('inf')
    ic_values = {}
    theoretical_ic = 0.0557 # Теоретичний IC для російського тексту

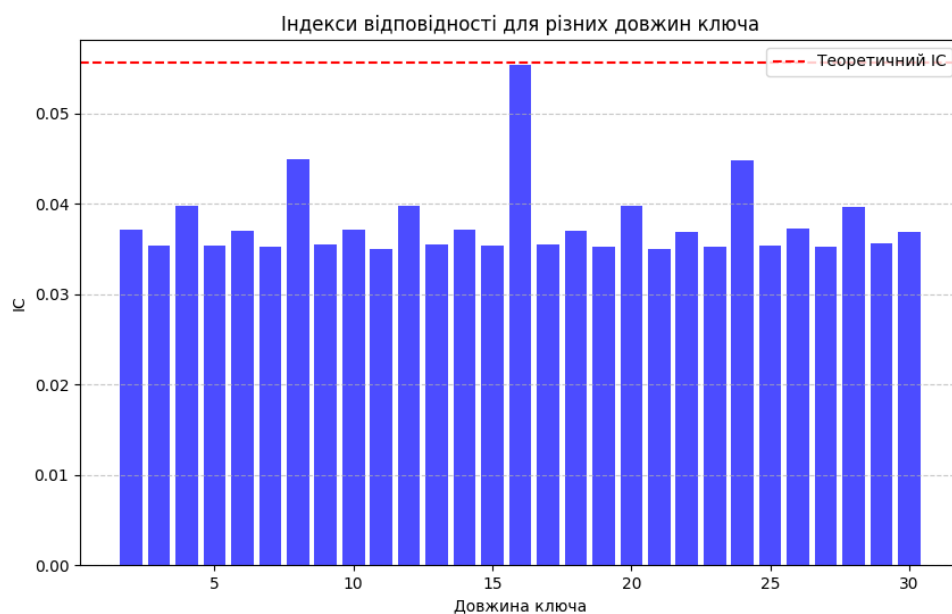
    for length in range(2, max_key_length + 1):
        average_ic = sum(index_of_coincidence(text[i::length]) for i in
range(length)) / length
        ic_values[length] = average_ic
        difference = abs(average_ic - theoretical_ic)

        if difference < minimal_difference:
            minimal_difference = difference
            optimal_length = length

    return optimal_length, ic_values

```

Результат:



Оптимальна довжина ключа: 16
Знайдений ключ: декелисоборойдей


```
lab2.3.py

def recover_key(text, key_length):
    """
    Знаходить ключ за допомогою частотного аналізу.
    """
    alphabet = 'абвгдежзийклмнопрстуфхцшщъыьэюя'
    most_common_letter = "о"
    key = []

    for i in range(key_length):
        block = text[i::key_length]
        most_common_char = Counter(block).most_common(1)[0][0]
        key_char = (alphabet.index(most_common_char) -
                    alphabet.index(most_common_letter)) % len(alphabet)
        key.append(alphabet[key_char])

    return ''.join(key)
```

recover_key: Ця функція знаходить ключ для розшифрування, використовуючи частотний аналіз. Вона розбиває текст на блоки, довжина яких відповідає довжині ключа, і визначає найчастотніший символ у кожному блоці. Потім, шляхом порівняння цього символу з найпоширенішою літерою природного тексту ("о"), обчислюється відповідний символ ключа. Отриманий ключ повертається у вигляді рядка. Цей метод базується на припущенні, що найпоширеніша літера в зашифрованому тексті відповідає найпоширенішій літері вихідного тексту.

```
lab2.3.py

def vigenere_decrypt(cipher_text, key):
    """
    Розшифровує текст за допомогою шифру Віженера.
    """
    alphabet = 'абвгдежзийклмнопрстуфхцшщъыьэюя'
    decrypted_text = []
    key_length = len(key)

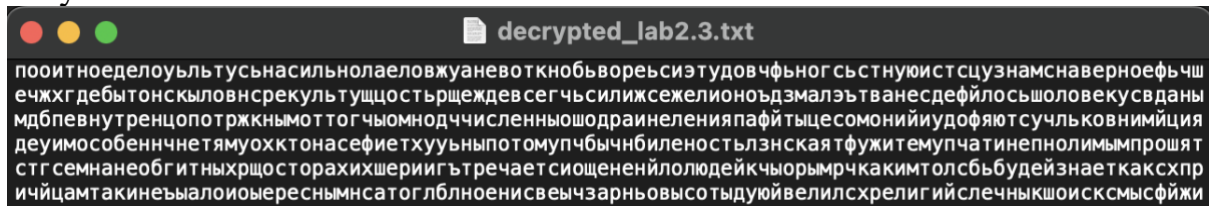
    for i, char in enumerate(cipher_text):
        if char in alphabet:
            shift = alphabet.index(key[i % key_length])
            decrypted_char = alphabet[(alphabet.index(char) - shift) %
                                      len(alphabet)]
            decrypted_text.append(decrypted_char)
        else:
            decrypted_text.append(char)

    return ''.join(decrypted_text)
```

vigenere_decrypt: Ця функція використовує знайдений ключ для розшифрування тексту. Для кожного символу шифротексту вона обчислює зворотний зсув за алфавітом відповідно до символу ключа. Якщо символ не входить до алфавіту (наприклад, це пробіл або знак пунктуації), він додається до результату без змін. У результаті формується розшифрований текст, який повертається як рядок.

Використовуємо знайдений ключ «*декелисоборйдей*», щоб розшифрувати текст

Результат:



На основі результатів програми та графіка, можна побачити, що оптимальна довжина ключа становить 16. У програмному виводі було отримано ймовірне значення ключа: «*декелисоборйдей*». Однак цей ключ не повністю відповідає тексту і потребує додаткового уточнення.

Розглядаючи розшифрований текст із цим ключем, ми помічаємо деякі знайомі слова, але частина тексту все ще виглядає неправильно. Для кращого аналізу розбиваємо текст на рядки по 16 символів, щоб відстежити, на яких позиціях ключа можуть бути помилки.

Пооитноеделоуль | тусьнасиьнолаел | овжуаневоткнобь
ореьсиэтудовчфьн | огсьстнуюистецуз | намснаверноефьчше

Спостереження:

У першому блоці: «Пооитноедело» виглядає як «дело» (позиції 9-12).

У другому блоці: «тусьнасиьнолаел» схоже на «насиьно» (позиції 5-12).

У шостому блоці: «намснаверноефьчше» схоже на «наверно» (позиції 5-12).

Отже, можна зробити висновок, що у ключі правильно визначені позиції з 5 по 12, а саме: «*декелисоборйдей*». Окрім того, у першому фрагменті слово «Пооитноедело» має очевидну схожість із фразою «Понятное дело», що дозволяє уточнити 3-ю та 4-ту позиції ключа.

Використовуємо формулу:

$$k = (y^* - x^*) \bmod m$$

y^* — індекс розшифрованої літери (очікуваний текст).

x^* — індекс зашифрованої літери (шифротекст).

k — індекс літери ключа.

$m=32$ — кількість літер у російському алфавіті без «ё»

Ми аналізуємо перші чотири літери шифротексту, щоб уточнити ключ. Ми знаходимо, що літеру «ш» в шифротексті було отримано в результаті зсуву літери «н» у відкритому тексті. Для обчислення цього зсуву застосовуємо формулу модульної арифметики:

$$(24 - 13) \bmod 32 = 11, \text{ що відповідає літері «л»}.$$

Це означає, що з літери «ш» було отримано літеру «л».

Далі, для іншого символу: літеру «н» у шифротексті було отримано в результаті зсуву літери «я» у відкритому тексті. Розрахунок виглядає так:

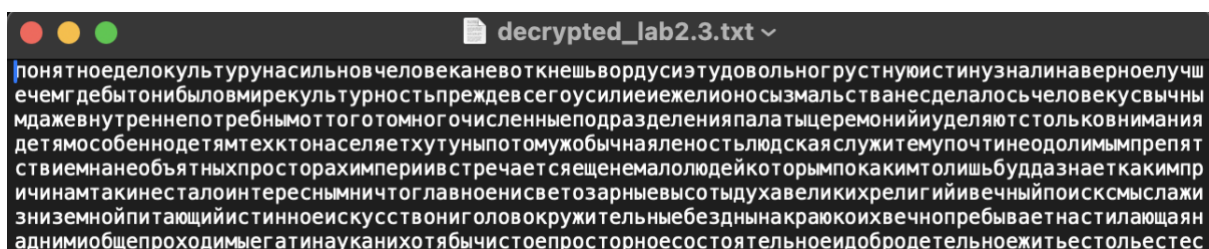
$$(13 - 31) \bmod 32 = (-18) \bmod 32 = 14, \text{ що відповідає літері «о»}.$$

Тобто з літери «н» було отримано літеру «о».

На цьому етапі ми уточнюємо ключ, який тепер виглядає як «делолисоборойдей».

І такими спробами ми дійшли до ключа який має вигляд «делолисоборотней»

Розшифрований текст:



Висновки:

У процесі виконання роботи з шифром Віженера було встановлено, що стійкість шифру значною мірою залежить від довжини та складності

ключа. Виконані дослідження продемонстрували, як методи криптоаналізу, зокрема частотний аналіз і використання індексу відповідності, дозволяють ефективно аналізувати шифротексти та відновлювати ключ.