# Комп’ютерний практикум №2

## Криптоаналіз шифру Віженера

**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 відповідної довжини:

```
keys = ["ар", "дос", "метр", "текст", "деньгинесм", "йщомпывояьп", "агенствонедв", "гурманынеедят", "журеавыофчцкыю", "славаукраинегер"]
```

У коді ми проходимося по всім ключам із цього списку.

Функція шифрування відкритого тексту за допомогою шифру Віженера:

```python
def vigenere_encrypt(text, key):
    encrypted_text = []
    key = key.lower()
    key_length = len(key)
    key_index = 0

    for char in text:
        if 'а' <= char <= 'я':
            shift = ord(key[key_index % key_length]) - ord('а')
            encrypted_char = chr((ord(char) - ord('а') + shift) % 32 + ord('а'))
            encrypted_text.append(encrypted_char)
            key_index += 1

    return ''.join(encrypted_text)
```

Для кожної літери ключа почергово обраховується зсув. Потім на основі цього зсуву обраховується номер зашифрованої букви із відкритого тексту та береться по модулю 32, щоб ми не вийшли за межі алфавіту. Те, що довжина ключа може не дорівнювати довжині відкритого тексту, також враховується, оскільки індекс літери ключа береться по модулю.

Функція дешифрування:

```python
def vigenere_decrypt(ciphertext, key):
    decrypted_text = []
    key = key.lower()
    key_length = len(key)
    key_index = 0

    for char in ciphertext:
        if 'а' <= char <= 'я':
            shift = ord(key[key_index % key_length]) - ord('а')
            decrypted_char = chr((ord(char) - ord('а') - shift) % 32 + ord('а'))
            decrypted_text.append(decrypted_char)
            key_index += 1

    return ''.join(decrypted_text)
```

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

Застосовуємо ці функції до всіх ключів у списку, при цьому шифротексти зберігаємо у словнику.

**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 є зашифрованим, то індекс відповідності буде випадковою функцією. Відповідно і частота появи тих чи інших букв у тексті буде рівна загальній імовірності їх появи у відкритому тексті. Індекс відповідності у такому випадку буде рівним

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

де t p – імовірність появи літери t в мові.

Обрахунок індексу відповідності реалізований наступним чином:

```
def index_of_coincidence(text):
    n = len(text)
    if n == 0:
        return 0

    letter_counts = Counter(text)
    ic = sum(count * (count - 1) for count in letter_counts.values()) / (n * (n - 1))
    return ic
```

Відбувається прохід по всім буквам відкритого тексту і обрахунок для них індексу відповідності. Частота появи букви ділиться на загальну кількість букв у тексті. Індекси відповідності всіх букв сумуються і ми отримуємо індекс відповідності для всього відкритого тексту.

Після застосування цієї функції до відкритого тексту, ми отримали результат:

```
Індекс відповідності відкритого тексту: 0.05598299566372803
```

Після використання функції для шифротекстів, ми отримали наступні результати:

| Довжина ключа | Індекс відповідності |
|---|---|
| 2 | 0.03953757513676814 |
| 3 | 0.04005597468464893 |
| 4 | 0.0368105234069343 |
| 5 | 0.03759071992087914 |
| 10 | 0.034519799953873866 |
| 11 | 0.0334559900600263 |
| 12 | 0.033389501941660835 |
| 13 | 0.03337287991206946 |
| 14 | 0.03183118666747006 |
| 15 | 0.034650698436905894 |

Можна прослідкувати, як по мірі збільшення довжини ключа зменшується індекс відповідності.

Для наступного завдання нам даний шифротекст. Експериментальним шляхом необхідно визначити імовірну довжину ключа на основі аналізу за допомогою індексів відповідності. Із попереднього завдання ми знаємо, що індекс відповідності відкритого тексту для російської мови приблизно рівний 0.5598.

Мінімальна довжина ключа - 2, максимальна - 30.

Оскільки ми розглядаємо шифр Віженера, де ключ незалежно від його довжини дублюється до тих пір, поки не досягне розміру відкритого тексту, за певний період, який рівний довжині ключа, можна прослідкувати, що зсув літер буде на той же індекс. Тому має сенс розбити шифротекст на блоки відповідно до періоду. Якщо, до прикладу, період буде рівним 2, то слово абракадабра буде розбито на блоки аркдба та баааp. Порахувавши індекс відповідності для кожного з блоків, ми таким чином зможемо знайти довжину ключа, оскільки для блоків, у яких вибрано літери, до яких теоретично був застосований однаковий зсув, індекс відповідності наблизиться до нормального значення, тобто приблизно 0.5598.

Такий принцип знаходження ключа було продемонстровано у трьох функціях нижче:

```
def split_into_blocks(ciphertext, period):
    blocks = [''] * period
    for i, char in enumerate(ciphertext):
        blocks[i % period] += char
    return blocks

def average_ic_for_period(ciphertext, period):
    blocks = split_into_blocks(ciphertext, period)
    ic_values = [index_of_coincidence(block) for block in blocks]
    return sum(ic_values) / len(ic_values)

def find_key_length(ciphertext, max_period=30):
    ics = {}
    for period in range(2, max_period + 1):
        ic = average_ic_for_period(ciphertext, period)
        ics[period] = ic
        print(f"Період: {period}, IC: {ic}")

    return ics
```

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

| Довжина ключа | Індекс відповідності |
|---|---|
| 2 | 0.03626820548217928 |
| 3 | 0.03482768182988512 |
| 4 | 0.03636814063472606 |
| 5 | 0.034922938298912444 |

| | |
|---|---|
| 6 | 0.03625749982568957 |
| 7 | 0.04462598000292351 |
| 8 | 0.03642261436036939 |
| 9 | 0.0347582318469512 |
| 10 | 0.03622973929007978 |
| 11 | 0.03487267099671423 |
| 12 | 0.0362869754225611 |
| 13 | 0.03488086337545442 |
| 14 | 0.05528168514213951 |
| 15 | 0.03490504821126325 |
| 16 | 0.036369597622619515 |
| 17 | 0.034842144619378235 |
| 18 | 0.03623606945107959 |
| 19 | 0.0348722467083866 |
| 20 | 0.03610046641142232 |
| 21 | 0.044469842267784 |
| 22 | 0.03629161557882429 |
| 23 | 0.034620481881822346 |
| 24 | 0.03624160801795553 |
| 25 | 0.03499992207832686 |
| 26 | 0.03635045767437681 |
| 27 | 0.034663226443448165 |
| 28 | 0.05536082691255105 |
| 29 | 0.03475382830824632 |
| 30 | 0.03623560278864295 |

Можна побачити, що для періодів 14 та 28 індекс відповідності збігся із індексом відповідності відкритого тексту.

Таким чином, можемо зробити припущення, що імовірні довжини ключа є 14 та 28.

Тепер спробуємо знайти сам ключ.

```
def find_key(ciphertext, key_length):
    blocks = split_into_blocks(ciphertext, key_length)
    key = []

    for block in blocks:
        freqs = Counter(block)
        most_common_letter = freqs.most_common(1)[0][0]
        shift = (ord(most_common_letter) - ord('o')) % 32
        key.append(chr(ord('a') + shift))

    return ''.join(key)
```

Так само розбиваємо зашифрований текст на блоки відповідно до періоду. У цьому випадку це буде 14 та 28. У кожному блоці знаходиться буква, яка зустрічається найчастіше. Припускаємо, що ця буква відповідає букві, яка зустрічається найчастіше у алфавіті відкритого тексту. Далі обраховуємо зсув між найпоширенішою буквою блоку шифротексту та найпоширенішою буквою відкритого тексту російською мовою (це буква о) і переводимо його у індекс, щоб знайти букву ключа для кожного блоку. Таким чином, відбувається прохід по всіх блоках.

Спробуємо застосувати її для ключів довжиною 14 та 28:

```
Знайдений ключ для довжини 14: жосвеыдиадозор

Знайдений ключ для довжини 28: жесвиднийдозорпослеызиаделор
```

Як бачимо, ключ довжини 28 є лише подвоєним ключем довжини 14, тому надалі ми будемо використовувати лише ключ довжини 14.

Враховуючи те, що ключ має бути змістовним, і те, який результат ми отримали при спробі знайти ключ довжини 14 та 28, можемо спробувати здогадатися яким же є фінальний ключ.

Це буде последнийдозор.

Спробуємо розшифрувати текст, використовуючи такий ключ:

Розшифрований текст з ключем довжиною 14:

Какясмогэтосделатьспросилгесерипочемуэтогонесмогсделатьтымыстоялипосредибескрайнейсеройравнинывзгляднефиксировалаяркихкрасоквцелойкартинеостоиловсмотретьсявотдельнуюпесчинкуитавспыхивалазолотомбагрянцемлазурьюзеленьюнадголовойзастылобелоесрозовымбудтомолочнуюрекуперемешалискисельнымиберегамидаивыплеснуливнебесааещедулветерибылохолодномневсегдахолоднаначетвертомслоесумраканоэтоиндивидуальнаяреакциягесерунапротивбыложарколицораскраснелосьполбустекаликапелькипотамненехватаетсилысказалялицогесерасовсемпобагровелоответнеправильныйтывысшиймагтакполучилосьслучайнонотывысшийпочемувысшихмаговтакженазываютмагамивнекатегорийпотомучторазницавсилемеждуниминастольконезначительначтонеможетбытьисчисленаиневозможноопределитьктосильнееактослабеепробормоталяборисигнатьевичяпонимаюномненехватаетсилыянемогупройтинапятыйслойгесерпосмотрелсебеподногиподделноскомботинкапесокподбросилввоздух(...)

Як бачимо, текст вдалося розшифрувати.

**Висновки:**

У цій роботі ми ознайомилися з методами частотного криптоаналізу. Експериментально визначили індекс відповідності для відкритого тексту російською мовою, порівняли його із індексами відповідності зашифрованих текстів і знайшли залежність індекса відповідності від довжини ключа.

На основі отриманих знань, ми вгадали довжину ключа певного зашифрованого тексту, а потім змогли розгадати і сам ключ, щоб дешифрувати текст.