

**Міністерство освіти і науки України**  
**Національний технічний університет України**  
**"Київський політехнічний інститут імені Ігоря Сікорського"**  
**Фізико-технічний інститут**

**Криптографія**

Комп'ютерний практикум №2  
Криптоаналіз шифру Віженера

Виконали:  
Студенти групи ФБ-22  
Дажук Павло, Копилов Сергій

Київ – 2024

## Варіант 8

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

### 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:

Обираємо текст з першої лабораторної роботи меншої довжини

```
text = read_text("anna-karenina.txt")[0:5000].replace(" ", "")
cleaned_text = clean_text(text)
```

Генеруємо 20 ключів довжиною від 1 до 20

```
keys = [''.join(random.choice(alphabet) for _ in range(i)) for i in range(1, 21)]
```

Обчислюємо індекс відповідності для відкритого тексту

```
ioc_results = {}
ioc_cleaned_text = index_of_coincidence(cleaned_text)
ioc_results[0] = round(ioc_cleaned_text, 4)
```

Шифруємо текст згенерованими ключами та рахуємо індекси відповідності всіх одержаних шифртекстів

```
for key in keys:
    encrypted_text = vigenere(cleaned_text, key, alphabet, encrypt=True)
    ioc = index_of_coincidence(encrypted_text)
    ioc_results[len(key)] = round(ioc, 4)
```

Довжина ключа	Індекс відповідності
0	0.0567
1	0.0567
2	0.0449
3	0.0373
4	0.0385
5	0.0385
6	0.0332
7	0.0342
8	0.0328
9	0.033
10	0.0327

11	0.0338
12	0.0328
13	0.0343
14	0.0332
15	0.0322
16	0.0323
17	0.0322
18	0.0329
19	0.032
20	0.0323



Зчитуємо даний нам шифротекст (варіант 8)

```
cipher_text = clean_text(read_text("cipher_text.txt"))
```

Знаходимо значення індексів відповідності для кожної довжини ключа в шифротексті

```
def find_optimal_r(cipher_text, max_r=30):
    ioc_values = {}

    for r in range(1, max_r + 1):
        blocks = [''.join(cipher_text[i] for i in range(j, len(cipher_text),
r)) for j in range(r)]
        ioc_per_block = [index_of_coincidence(block) for block in blocks]
        avg_ic = sum(ioc_per_block) / r
        ioc_values[r] = round(avg_ic, 4)

    return ioc_values
```

Довжина ключа	Індекс відповідності
1	0.0332

2	0.0348
3	0.0332
4	0.0362
5	0.0398
6	0.0349
7	0.0332
8	0.0361
9	0.0333
10	0.0462
11	0.0332
12	0.0362
13	0.033
14	0.0347
15	0.0397
16	0.0363
17	0.0333
18	0.0348
19	0.0331
20	0.0557
21	0.0331
22	0.0347
23	0.0332
24	0.0361
25	0.04
26	0.0349
27	0.0332
28	0.0358
29	0.0331
30	0.046



Найближче значення до теоритичного отримано для ключа довжиною 20

```
optimal_r = 20
```

Визначемо ключ для кожного блоку обравши за букву «о» найчастішу букву в шифротексті, а також букви наближені до найчастішої, та рахуємо зсув відносно «о». З отриманих результатів формуємо всі можливі комбінації ключа.

```
def find_letter_key(cipher_text, r, alphabet):
    blocks = [''.join(cipher_text[i] for i in range(j, len(cipher_text), r))
    for j in range(r)]
    key = []

    # Індекс найімовірнішої літери мови (для російської це "о")
    x_index = alphabet.index('о')

    for block_num, block in enumerate(blocks, start=1):
        letter_counts = Counter(block)
        total_letters = sum(letter_counts.values())
        frequencies = {letter: letter_counts.get(letter, 0) / total_letters
        for letter in alphabet}

        # Сортуємо за частотою
        sorted_frequencies = sorted(frequencies.items(), key=lambda x: x[1],
        reverse=True)

        # Знаходимо букви, частоти яких близькі до максимальної частоти
        max_frequency = sorted_frequencies[0][1]
        close_frequencies = [letter for letter, freq in sorted_frequencies if
        abs(freq - max_frequency) <= 0.015]

        # Обчислюємо можливі значення ключа для кожного блоку
        block_keys = [(alphabet.index(letter) - x_index) % len(alphabet) for
        letter in close_frequencies]
        key.append(block_keys)

    keys = [''.join(alphabet[k] for k in key_comb) for key_comb in
    list(product(*key))]

    return keys
```

```
keys = find_letter_key(cipher_text, optimal_r, alphabet)
with open("keys.txt", "w", encoding="utf-8") as file:
    for key in keys:
        file.write(f"{key}\n")
```

Знаходимо ключ з найбільш зрозумілим змістом

>	Q- серебряныепули	×
295	<u>улановсеребзцныепуя</u>	
296	<u>улановсеребзцныепуи</u>	
297	<u>улановсеребряныепуля</u>	
298	<u>улановсеребряныепули</u>	
299	<u>улановсеребряныепуя</u>	

## Перевіряємо

```
key = "улановсеребряныепули"  
  
plaintext = vigenere(cipher_text, key, alphabet, encrypt=False)  
  
print(cipher_text[0:40])  
print(key*2)  
print(plaintext[0:40])
```

рэаюцугкьелаяюиутбхигцичопщпюиермтгсфюлх – шифротекст

улановсеребряныепулиулановсеребряныепули – ключ

этасистемакрасногокарликаникогданеимелан – відкритий текст

або

эта система красного карлика никогда не имела н... – відкритий текст

## Висновки

У роботі було засвоєно метод частотного криптоаналізу шифру Віженера, ми навчилися використовувати його для шифрування та розшифрування. Досліджено поняття індексу відповідності та спосіб знаходження довжини ключа для шифротексту за його допомогою. За результатами було визначено, що зі збільшенням розміру ключа та вибором випадкових символів для нього значення для індексу відповідності зменшується.