

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Навчально-науковий фізико-технічний інститут
Кафедра інформаційної безпеки

Дисципліна «Криптографія»

Комп'ютерний практикум

Робота №2

Криптоаналіз шифру Віженера

Виконали: студенти гр. ФБ-12 Головка М. С. і Марчук І. С.

Київ – 2023

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. Почали з того, що створили текстовий файл розміром 4 кб з якимось випадковим текстом (файл під назвою `input_to_cipher.txt`). Саме цей текст будемо шифрувати у першому завданні.

Написали в Python (пишемо цією мовою і далі) функцію шифрування шифром Віженера (у коді вона називається `vigenere_cipher`), яка приймає два аргументи – текст, який треба зашифрувати, і ключ, яким треба шифрувати. Якщо коротко, ця функція створює два масиви – один з букв відкритого тексту, другий масив складається з букв ключа (масив `key_repeated`, який до цього був стрінгом, в якому ключ записано стільки разів, скільки треба, щоб довжина стрінгу відкритого тексту була така сама, як довжина стрінгу повторюванного ключа). І потім кожна буква з масиву шифрується буквою з масиву ключа за відповідною формулою:

$$y_i = (x_i + k_{i \bmod r}) \bmod m, i = \overline{0, n}.$$

(функції `ord` і `chr` взаємнообернені, `ord` отримує букву і віддає ID цієї букви, а `chr` навпаки, отримує ID букви і віддає букву, в якій відповідне ID)

Тобто всього в нас вийшло 5 зашифрованих текстів:

- 1) $r=2$, ключ – «ма»

[illegible]

- 2) $r=3$, ключ – «МЯУ»

приймає як аргументи словник (той самий словник з кількостями появи букв) і кількість усіх букв у тексті (у нашому випадку всюди: і в відкритому тексті, і у всіх шифртекстах кількість всіх букв буде однакова).

Ось такі значення індексів відповідності у нас вийшли (перше значення – для відкритого тексту, а далі показані індекси відповідності відповідно для $r=2$, $r=3$, $r=4$, $r=5$, $r=16$):

```
0.054240485313873966
*****
0.04034963513678555
*****
0.035914791463116755
*****
0.035084212414509035
*****
0.03409142953102101
*****
0.03318070270517831
*****
```

Також ми порахували значення математичного очікування індексів відповідності за наступною формулою:

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

, де p_t – імовірність появи літери t у мові.

Для цього створили функцію `theoretical_ic`, яка приймає як аргументи словник кількостей появи букв у тексті і кількість всіх букв у тексті.

На наступному екрані показані спочатку значення індексів відповідності (які ми вже порахували, з попереднього екрана), а після кожного з них записане їхнє математичне очікування:

```
0.054240485313873966
*****
0.04034963513678555
0.040818214416503906
*****
0.035914791463116755
0.036385536193847656
*****
0.035084212414509035
0.035555362701416016
*****
0.03409142953102101
0.03456306457519531
*****
0.03318070270517831
0.03365278244018555
*****
```

Для відкритого тексту математичне очікування індексу відповідності ми порахували, користуючись текстом з попередньої лаби (той великий текст без пробілів) і це теоретичне значення (MI) співпало з практичним значенням (I). Тобто ми в наш код з першої лаби закинули формулу `theoretical_ic` і порахували його там (потім видалили цю функцію і взагалі зайві рядки):

```
def theoretical_ic(letter_freq, text_length):
    result=float()
    for i in letter_freq.values():
        result+=(float(i)/text_length)**2
    return result
```

```
our_freq=find_freq(output_without_spaces)
print(theoretical_ic(our_freq,len(output_without_spaces)))
```

Ось таке значення MI в нас вийшло для більшого тексту:

```
0.05492834774381558
PS D:\uni_year_2\aks_labi\1
```

А таке значення індексу відповідності в нас було:

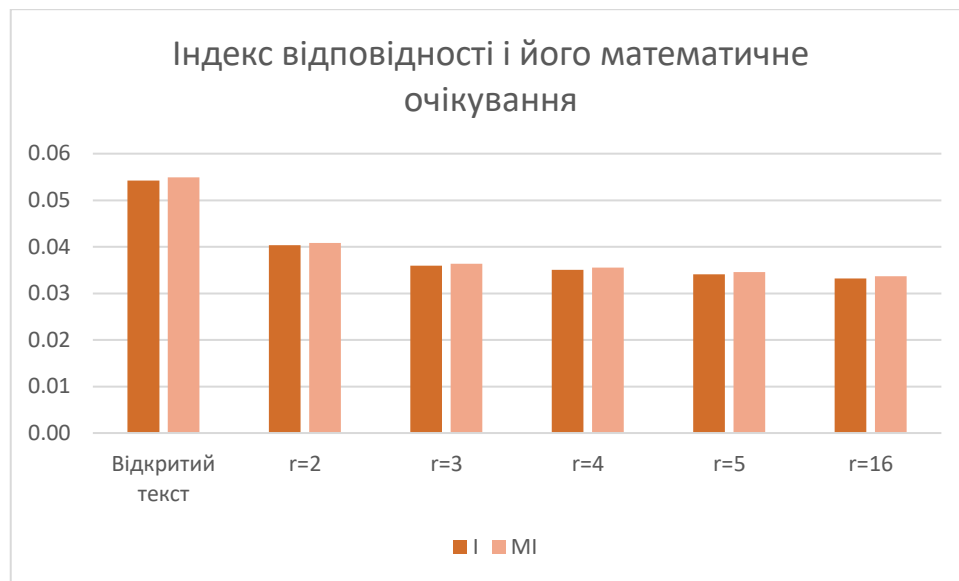
```
0.054240485313873966
PS D:\uni_year_2\aks_labi\1
```

Тобто теоретичне значення індексу відповідності співпало з практичним.

Для зручності, занесемо отримані дані в таблицю і діаграму:

	Відкритий текст	r=2	r=3
Значення I	0.054240485313873966	0.04034963513678555	0.035914791463116755
Значення MI	0.05492834774381558	0.040818214416503906	0.036385536193847656
	r=4	r=5	r=16
Значення I	0.035084212414509035	0.03409142953102101	0.03318070270517831
Значення MI	0.035555362701416016	0.03456306457519531	0.03365278244018555

Таблиця 1. Значення індексів відповідності і їхніх математичних очікувань



Діаграма 1. Індеси відповідності і їхні математичні очікування для різних r

Можна побачити, що значення індексу відповідності і його математичного очікування спадає зі збільшенням r . Це зумовлено тим, що зі збільшенням довжини ключа, по-перше, сам ключ стає більш складним (відповідно, і сам шифр стає складнішим теж), а по-друге, коли ключ стає довшим, відбувається менше повторень цього ключа, тому шаблон шукати важче.

3. Отже, в нас перший варіант, беремо зашифрований текст з цього варіанту і закинемо його у файл test.txt. Для знаходження довжини ключа шифрування, яким був зашифрований цей текст, ми обрали перший алгоритм, що був наданий у теоретичних відомостях:

- 1) Для кожного кандидата $r = 2, 3, \dots$ розбити шифртекст Y на блоки $Y_1, Y_2, Y_3, \dots, Y_r$.
- 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 (у нашому випадку це $\frac{1}{32} = 0,03125$), то значення r вгадане неправильно.

Отже, зробили функцію `calc_IC_block`, яка приймає як аргументи зашифрований текст і значення r . Значення r надається для того, щоб розуміти, на які блоки треба розділяти текст (тобто, якщо наприклад $r=5$, то треба у блок додавати букви із кроком у 5). У методичних вказівках зазначено, що треба перевіряти довжини щонайменше до $r=30$, тому так і зробимо:

```
for i in range (2,31):
    print(i, "=", calc_IC_block(text, i))
```

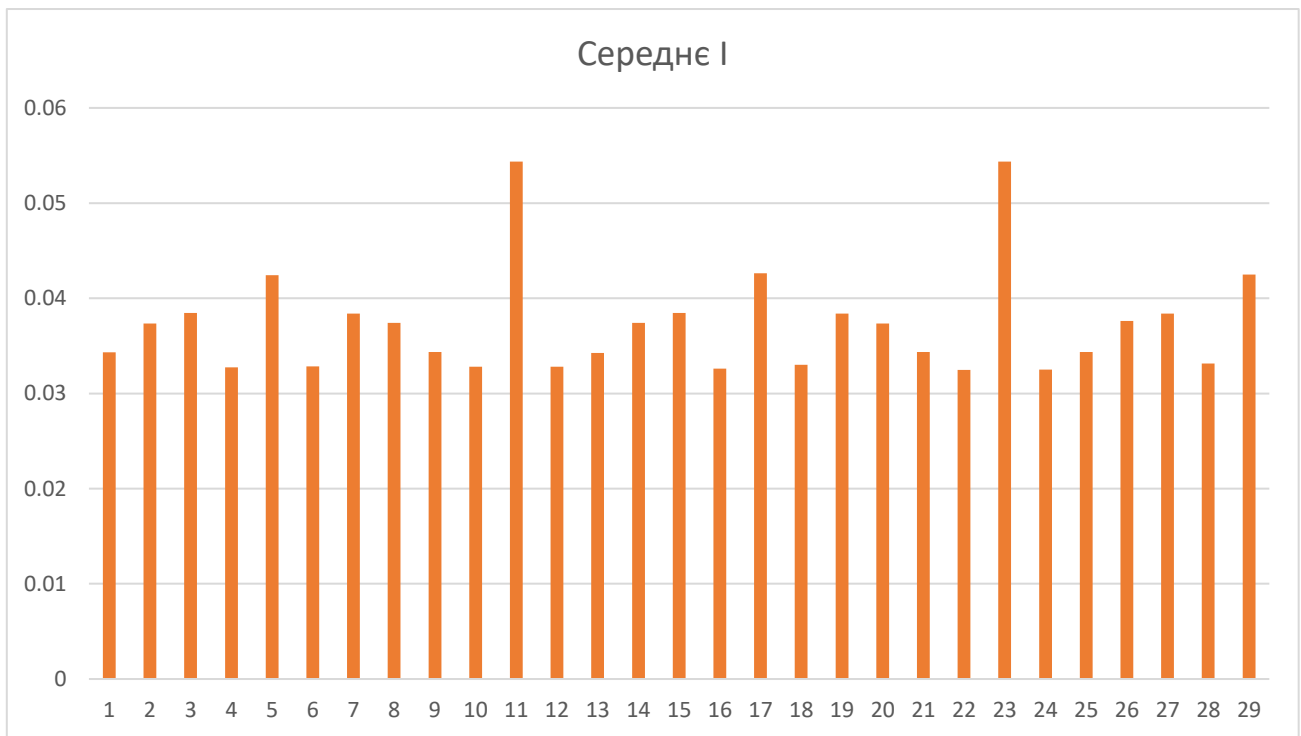

2 = 0.03432921421542369
 3 = 0.03734839112182639
 4 = 0.03846786795894798
 5 = 0.032753684507439526
 6 = 0.04242249836150345
 7 = 0.03284567162583475
 8 = 0.03839430526208765
 9 = 0.03740691348616666
 10 = 0.034343106655826135
 11 = 0.03282596004503103
 12 = 0.054369556735866346
 13 = 0.032807635112857336
 14 = 0.03425313309436149
 15 = 0.03741441107403287
 16 = 0.03846816039387033
 17 = 0.0326076877752591
 18 = 0.04261923978140026
 19 = 0.03299852287693897
 20 = 0.038394078333066343
 21 = 0.03734596917614833
 22 = 0.03436346417856435
 23 = 0.03248823743567128
 24 = 0.05435416649918132
 25 = 0.032517536103743
 26 = 0.034348576654149546
 27 = 0.03762500312229973
 28 = 0.038386039042765406
 29 = 0.03313218390804597
 30 = 0.042504500512293736

Нас цікавлять значення, які схиляються до теоретичного значення I для даної мови, тобтодесь 0.054240485313873966. На екрані вже можна побачити, що ці значення I приймає при $r=12$ та $r=24$. Занесемо отримані дані в таблицю і діаграму:

r	Середнє I
2	0.03432921421542369
3	0.03734839112182639
4	0.03846786795894798
5	0.032753684507439526
6	0.04242249836150345
7	0.03284567162583475
8	0.03839430526208765
9	0.03740691348616666
10	0.034343106655826135
11	0.03282596004503103
12	0.054369556735866346
13	0.032807635112857336
14	0.03425313309436149
15	0.03741441107403287
16	0.03846816039387033
17	0.0326076877752591
18	0.04261923978140026
19	0.03299852287693897

20	0.038394078333066343
21	0.03734596917614833
22	0.03436346417856435
23	0.03248823743567128
24	0.05435416649918132
25	0.032517536103743
26	0.034348576654149546
27	0.03762500312229973
28	0.038386039042765406
29	0.03313218390804597
30	0.042504500512293736

Таблиця 2. Середні I для блоків зашифрованого тексту



Діаграма 2. Середні I для блоків зашифрованого тексту

Тобто, можемо припустити, що довжина ключа шифрування дорівнює або 12, або 24 (примітка: ми перевіряли цей код на наших інших шифртекстах, які ми до цього самі зашифрували, знаючи ключ, і воно працює. Наприклад, коли $r=5$, теж показується теоретичне значення I:

```

2 = 0.03415773154936462
3 = 0.033733205908058304
4 = 0.03433257399706457
5 = 0.05400379318046021

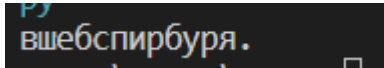
```

Тепер будемо намагатися дізнатися, що саме за ключ. Його розшифрування зводиться до серії розшифрувань шифрів Цезаря. Тобто ми беремо кожний блок тексту, дивимось, яка буква там

частіше всього зустрічається, і припускаємо, що це буква «о» (тому що вона частіше всього зустрічається в тексті російською). Після цього знаходимо ключ за формулою:

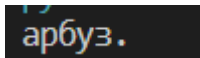
$$k = (y^* - x^*) \bmod m$$

, де y^* - це буква, яка частіше всього зустрічається у шифртексті, а x^* - це буква, яка частіше всього зустрічається в мові. Для цього написали функцію `get_key`, яка приймає як аргументи зашифрований текст і значення r . Проте цей код видав отаке:



Щось схоже на слово, але не дуже (адже в методичних вказівках було зазначено, що ключ повинен бути змістовним). Знову ж таки, ми перевіряли цю функцію на нашому іншому тексті, який ми зашифрували самі ($r=5$, наприклад):

```
print(get_key(crypto5, 5))
```



Скоріш за все проблема в тому, що ті букви, які ми вважали за «о», виявилися не «о». Тоді беремо наступну за популярністю букву – «е». Зробили таку ж саму функцію, як з «о», тільки з «е», назва функції – `get_key2`.

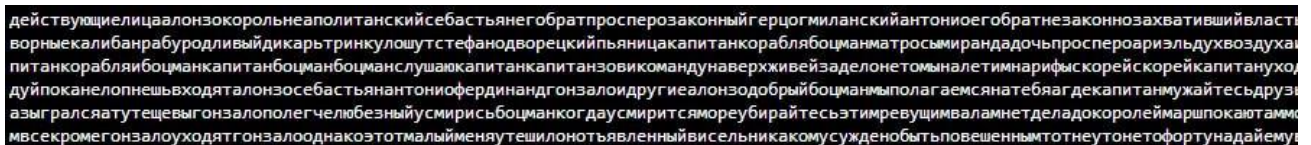
Вийшло отаке:



Після години роздумів зрозуміли, що ключ – «вшекспирбуря», тобто твір В. Шекспіра – «Буря». (треба було в попередньому ключі просто четверту букву поміняти на «к», яка у нас у другій спробі знайти ключ) (припускаємо, що у розшифрованому тексті повинен бути уривок з цього твору)

Тепер залишилося тільки розшифрувати цей текст, вже знаючи ключ. Для цього написали функцію `vigenere_decrypt`, яка робить протилежне функції `vigenere_cipher`.

Застосували цю функцію для зашифрованого тексту і ось, що вийшло:



Це справді уривок з твору В. Шекспіра «Буря».

Висновки: у ході комп'ютерного практикуму було розглянуто методи шифрування і розшифрування тексту шифром Віженера. Ми навчилися шифрувати текст шифром Віженера, а також знаходити для нього індекс відповідності I і його математичне очікування MI . За результатами обрахувань, які було занесено у таблицю 1 і візуалізовано на діаграмі 1, можна зробити висновок, що індекс відповідності I і його математичне очікування спадає зі збільшенням періоду шифру, що зумовлено збільшенням складності шифру. Також ми

навчилися розшифровувати текст методом знаходження індексів відповідності для блоків шифртексту. У нашому випадку довжина ключа дорівнює 12, що можна побачити у таблиці 2 і на діаграмі 2 (значення I схиляється до теоретичного). Також ми дізналися, що для розшифрування великого тексту зручно застосовувати частотний аналіз: можна порівнювати частоту букв, які частіше всього зустрічаються у шифртексті, з частотою букв, які найчастіше використовуються у мові в цілому.