Міністерство освіти і науки України

Національний технічний університет України

"Київський політехнічний інститут імені Ігоря Сікорського"

Фізико-технічний інститут

Криптографія

Комп’ютерний практикум №3

Криптоаналіз афінної біграмної підстановки

Варіант 4

Виконали:

Студенти 3 курсу

Загородній Я.М, Венгер П.Ю.

Перевірив:

Київ – 2024

Мета роботи: Набуття навичок частотного аналізу на прикладі розкриття моноалфавітної підстановки; опанування прийомами роботи в модулярній арифметиці.

Постановка задачі:

1. Реалізувати підпрограми із необхідними математичними операціями: обчисленням оберненого елементу за модулем із використанням розширеного алгоритму Евкліда, розв'язуванням лінійних порівнянь. При розв'язуванні порівнянь потрібно коректно обробляти випадок із декількома розв'язками, повертаючи їх усі.

2. За допомогою програми обчислення частот біграм, яка написана в ході виконання комп'ютерного практикуму No1, знайти 5 найчастіших біграм запропонованого шифртексту (за варіантом).

3. Перебрати можливі варіанти співставлення частих біграм мови та частих біграм шифртексту (розглядаючи пари біграм із п'яти найчастіших). Для кожного співставлення знайти можливі кандидати на ключ (a,b) шляхом розв'язання системи:

$$\begin{cases} Y^{*} \equiv aX^{*} + b \pmod{m^2} \\ Y^{**} \equiv aX^{**} + b \pmod{m^2} \end{cases},$$

4. Для кожного кандидата на ключ дешифрувати шифртекст. Якщо шифртекст не є змістовним текстом російською мовою, відкинути цього кандидата.

5. Повторювати дії 3-4 доти, доки дешифрований текст не буде змістовним.

Хід роботи:

1. Реалізувати підпрограми із необхідними математичними операціями: обчисленням оберненого елементу за модулем із використанням розширеного алгоритму Евкліда, розв'язуванням лінійних порівнянь. При розв'язуванні порівнянь потрібно коректно обробляти випадок із декількома розв'язками, повертаючи їх усі.

```python
from math import gcd

# Функція реалізації розширеного алгоритму Евкліда
def extended_euclid(a, b):
    if a == 0:
        return b, 0, 1
    gcd_val, x, y = extended_euclid(b % a, a)
    return gcd_val, y - (b // a) * x, x

# функція знаходить обернений елемент числа a за модулем m
def modulo_inverse(a, m):
    if gcd(a, m) != 1:
        return "Can't find a^(-1)"
    else:
        u = extended_euclid(a, m)[1]
        return u % m

# Функція для вирішення лінійних конгруентних рівнянь
def modular_equation(a, b, mod):
    d = gcd(a, mod)
    if d == 1:
        x = (modulo_inverse(a, mod) * b) % mod
        return [x]
    else:
        if b % d != 0:
            return ["no solutions"]
        else:
            results = []
            x = modular_equation(a // d, b // d, mod // d)[0]
            for i in range(d):
                results.append((x + (mod // d) * i) % mod)
            return results
```

1. Функція extended_euclid(a, b):

Ця функція реалізує розширений алгоритм Евкліда, який:

- Рекурсивно обчислюється НСД для чисел a та b, доки a не стане нулем.
- У зворотному напрямку розраховуються коефіцієнти x і y.

2. Функція modulo_inverse(a, m):

Ця функція знаходить обернений елемент числа a за модулем m

- Перевіряється, чи a та m взаємно прості (якщо ні, обернений елемент знайти неможливо)
- Використовується розширений алгоритм Евкліда, щоб знайти коефіцієнт u, який є оберненим елементом.

3. Функція modular_equation(a, b, mod):

Ця функція розв’язує лінійне конгруентне рівняння вигляду: $a \cdot x \equiv b \pmod{}$

- Якщо d=1, рівняння має єдиний розв’язок (використовується обернений елемент a для знаходження x)
- Якщо d>1 і b не кратне d (Розраховується один розв’язок і потім доповнюється іншими, враховуючи періодичність). Інакше розв’язків немає

2. За допомогою програми обчислення частот біграм, яка написана в ході виконання комп’ютерного практикуму No1, знайти 5 найчастіших біграм запропонованого шифртексту (за варіантом).

```python
from collections import Counter

# Функція для отримання всіх біграм у тексті
def get_bigrams(text):
    bigrams = [text[i:i+2] for i in range(len(text)-1)]
    return bigrams

# Функція для пошуку найпоширеніших біграм у тексті
def find_most_common_bigrams(text, top_n=5):
    bigrams = get_bigrams(text)
    bigram_counts = Counter(bigrams)
    total_big = sum(bigram_counts.values())
    most_common_bigrams = bigram_counts.most_common(top_n)
    return most_common_bigrams, total_big

chipter_text = open('04.txt', 'r', encoding='utf-8').read().replace("\n", "")
most_common_bigrams, total_big = find_most_common_bigrams(chipter_text) # list

chipher_bigrams = []
for bigram, count in most_common_bigrams:
    chipher_bigrams.append(bigram)
    freq_bigr = count/total_big
    print(f"Біграм: {bigram}, Кількість: {count}, Частота {freq_bigr:.3%}")

print(chipher_bigrams)
```

Результат:

```
Біграм: еш, Кількість: 68, Частота 1.163%
Біграм: шя, Кількість: 52, Частота 0.890%
Біграм: еы, Кількість: 50, Частота 0.855%
Біграм: до, Кількість: 49, Частота 0.838%
Біграм: зо, Кількість: 48, Частота 0.821%
['еш', 'шя', 'еы', 'до', 'зо']
```

3. Перебрати можливі варіанти співставлення частих біграм мови та частих біграм шифртексту (розглядаючи пари біграм із п’яти найчастіших). Для кожного співставлення знайти можливі кандидати на ключ (a,b) шляхом розв’язання системи:

```
rus_birgams = ["ст", "но", "то", "на", "ен"]
alphabet_str = 'абвгдежзийклмнопрстуфхцчшщьыэюя'

# Формуємо всі можливі пари біграм (російські -> зашифровані)
pairs_list_bigram = [(rus_bigr, cipher_bigr) for rus_bigr in rus_birgams for cipher_bigr in chipher_bigrams]
print("Pairs:")
print(pairs_list_bigram)

# Створення словника для перетворення літер у їх числове представлення
lettertonum = {letter: index for index, letter in enumerate(alphabet_str)}

# Перетворення кожної біграми в числове представлення
pairs_num_bigram = [
    (len(alphabet_str) * lettertonum[pair[0][0]] + lettertonum[pair[0][1]],
     len(alphabet_str) * lettertonum[pair[1][0]] + lettertonum[pair[1][1]])
    for pair in pairs_list_bigram]
print("Pairs num:")
print(pairs_num_bigram)

alphabet_squared = len(alphabet_str) ** 2
possible_key = []

# Функція для знаходження різниці двох значень
def find_delta(first_el, second_el):
    delta = first_el - second_el
    return delta
```

```python
# Пошук можливих ключів
for first in range(len(pairs_num_bigram)):
    X1, Y1 = pairs_num_bigram[first]
    for second in range(first + 1, len(pairs_num_bigram)):
        X2, Y2 = pairs_num_bigram[second]
        delta_X = find_delta(X1, X2)
        delta_Y = find_delta(Y1, Y2)
        if delta_X == 0:
            continue
        try:
            result = modulo_inverse(delta_X, alphabet_squared)
            # Якщо обернений елемент знайдено
            if isinstance(result, int):
                num_delta_X = result
                a = (delta_Y * num_delta_X) % alphabet_squared
                b = (Y1 - a * X1) % alphabet_squared
                answer = modular_equation(a, b, alphabet_squared)
                if answer is not None:
                    possible_key.append((a, b))
        except ValueError as e:
            print("ValueError:", e)
            continue

print('Maybe key:')

chunk_size = 10
for i in range(0, len(possible_key), chunk_size):
    chunk = possible_key[i:i + chunk_size]
    print(chunk)
```

Результат:

```
Pairs:
[('ст', 'еш'), ('ст', 'шя'), ('ст', 'ем'), ('ст', 'до'), ('ст', 'зо'), ('но', 'еш'), ('но', 'шя'), ('но', 'ем'), ('но', 'до'), ('но', 'зо'), ('то', 'еш'), ('то', 'шя'), ('то', 'ем'), ('то', 'до'), ('то', 'зо'), ('на',
Pairs num:
[(545, 179), (545, 774), (545, 182), (545, 138), (545, 231), (417, 179), (417, 774), (417, 182), (417, 138), (417, 231), (572, 179), (572, 774), (572, 182), (572, 138), (572, 231), (403, 179), (403, 774), (403, 182),
Maybe key:
[(0, 179), (138, 888), (503, 890), (173, 72), (390, 10), (0, 179), (761, 586), (427, 26), (571, 348), (354, 410)]
[(0, 179), (165, 588), (873, 89), (562, 448), (717, 541), (0, 179), (470, 616), (390, 10), (436, 887), (33, 453)]
[(823, 65), (0, 774), (365, 776), (35, 919), (252, 857), (200, 367), (0, 774), (627, 214), (771, 536), (554, 598)]
[(796, 365), (0, 774), (708, 275), (397, 634), (552, 727), (491, 337), (0, 774), (881, 168), (927, 84), (524, 611)]
[(458, 432), (596, 180), (0, 182), (631, 325), (848, 263), (534, 335), (334, 742), (0, 182), (144, 504), (888, 566)]
[(88, 272), (253, 681), (0, 182), (650, 541), (805, 634), (571, 351), (80, 788), (0, 182), (46, 98), (604, 625)]
[(788, 245), (926, 954), (330, 956), (0, 138), (217, 76), (390, 930), (190, 376), (817, 777), (0, 138), (744, 200)]
[(399, 830), (564, 278), (311, 740), (0, 138), (155, 231), (525, 391), (34, 828), (915, 222), (0, 138), (558, 665)]
[(571, 400), (709, 148), (113, 150), (744, 293), (0, 231), (607, 0), (407, 407), (73, 808), (217, 169), (0, 231)]
[(244, 830), (409, 278), (156, 740), (806, 138), (0, 231), (928, 918), (437, 394), (357, 749), (403, 665), (0, 231)]
[(0, 179), (438, 123), (343, 337), (758, 262), (820, 355), (0, 179), (893, 666), (741, 624), (444, 504), (351, 845)]
[(523, 830), (0, 774), (866, 27), (320, 913), (382, 45), (68, 287), (0, 774), (809, 732), (512, 612), (419, 953)]
[(618, 24), (95, 929), (0, 182), (415, 107), (477, 200), (220, 698), (152, 224), (0, 182), (664, 62), (571, 403)]
[(203, 55), (641, 960), (546, 213), (0, 138), (62, 231), (517, 774), (449, 300), (297, 258), (0, 138), (868, 479)]
[(141, 55), (579, 960), (484, 213), (899, 138), (0, 231), (610, 526), (542, 52), (390, 10), (93, 851), (0, 231)]
[(0, 179), (531, 123), (870, 337), (603, 262), (665, 355), (0, 179), (862, 188), (245, 345), (816, 473), (723, 814)]
[(430, 830), (0, 774), (339, 27), (72, 913), (134, 45), (99, 845), (0, 774), (344, 50), (915, 178), (822, 519)]
[(91, 24), (622, 929), (0, 182), (694, 107), (756, 200), (716, 16), (617, 906), (0, 182), (571, 310), (478, 651)]
[(358, 55), (889, 960), (267, 213), (0, 138), (62, 231), (145, 805), (46, 734), (390, 10), (0, 138), (868, 479)]
[(296, 55), (827, 960), (205, 213), (899, 138), (0, 231), (238, 557), (139, 486), (483, 723), (93, 851), (0, 231)]
[(0, 179), (754, 954), (687, 86), (221, 489), (376, 489), (207, 960), (0, 774), (894, 867), (428, 309), (583, 309)]
[(274, 275), (67, 89), (0, 182), (495, 585), (650, 585), (740, 789), (533, 603), (466, 696), (0, 138), (155, 138)]
[(585, 882), (378, 696), (311, 789), (806, 231), (0, 231)]
```

4. Для кожного кандидата на ключ дешифрувати шифртекст. Якщо шифртекст не є змістовним текстом російською мовою, відкинути цього кандидата.

5. Повторювати дії 3-4 доти, доки дешифрований текст не буде змістовним.

```python
# Створення словників для перетворення символів у числові індекси і навпаки
char_to_index = {char: i for i, char in enumerate(alphabet_str)}
index_to_char = {i: char for i, char in enumerate(alphabet_str)}

encrypted_text = chipter_text.replace('\r', '')

# Перетворюємо зашифрований текст у числовий список, замінюючи кожен символ його індексом
encrypted_numbers = [char_to_index[char] for char in encrypted_text]

# Функція для дешифрування афінного шифру
def decrypt_affine_cipher(numbers, key_a, key_b, modulus_squared):
    decrypted_message = []
    try:
        inverse_a = modulo_inverse(key_a, modulus_squared)
    except ValueError:
        print(f"Оберненого не існує для key_a = {key_a}. Пропускаємо ключ.")
        return None
    for index in range(0, len(numbers), 2):
        first_num = numbers[index]
        second_num = numbers[index + 1] if index < len(numbers) - 1 else 0
        bigram_index = first_num * 31 + second_num
        if not all(isinstance(value, int) for value in [inverse_a, bigram_index, key_b, modulus_squared]):
            return None
        original_index = (inverse_a * (bigram_index - key_b)) % modulus_squared
        first_original = original_index // 31
        second_original = original_index % 31
        decrypted_message.append(index_to_char[first_original % len(alphabet_str)])
        decrypted_message.append(index_to_char[second_original % len(alphabet_str)])
    return ''.join(decrypted_message)

decrypted_variants = {}
for key_a, key_b in possible_key:
    decrypted_result = decrypt_affine_cipher(encrypted_numbers, key_a, key_b, alphabet_squared)
    if decrypted_result:
        decrypted_variants[(key_a, key_b)] = decrypted_result
```

✓ 1.5s

Функція decrypt_affine_cipher дешифрує текст, зашифрований афінним шифром, використовуючи задані ключі keya і kayb

1. Знаходження оберненого елемента для

    - Використовується функція modulo_inverse.
    - Якщо обернений елемент не існує, дешифрування неможливе.

2. Обробка біграм:

    - Текст перетворюється в числові біграми (двійки чисел) за індексами символів.
    - Для кожної біграми обчислюється її зворотне значення за формулою:

$$Y^* - Y^{**} \equiv a(X^* - X^{**}) \pmod{m^2}. \qquad b = (Y^* - aX^*) \bmod m^2.$$

- Розрахунок індексів символів для дешифрованого тексту.

```python
# Список рідкісних біграм
uncommon_bigrams = ["щт", "ьо", "ьж", "юв", "яы", "аы", "бй", "гй", "дй", "еы", "шщ", "шя", "щб", "щд", "цк", "ьы", "ыа", "ыь", "ыы", "ыэ"]

# Функція для розбиття тексту на біграми
def generate_bigrams(text):
    return [text[index:index + 2] for index in range(0, len(text), 2)]

# Функція для підрахунку рідкісних біграм у тексті
def calculate_uncommon_bigrams(text, uncommon_list):
    bigrams_in_text = generate_bigrams(text)
    uncommon_count = sum(1 for bigram in bigrams_in_text if bigram in uncommon_list)
    return uncommon_count

uncommon_bigram_counts = {}

for decryption_key, decoded_text in decrypted_variants.items():
    uncommon_count = calculate_uncommon_bigrams(decoded_text, uncommon_bigrams)
    uncommon_bigram_counts[decryption_key] = uncommon_count


sorted_uncommon_counts = sorted(uncommon_bigram_counts.items(), key=lambda item: item[1])
print("Топ-10 ключів із найменшою кількістю рідкісних біграм:")
for rank, (key, count) in enumerate(sorted_uncommon_counts[:10], start=1):
    print(f"{rank}. Ключ {key} - Кількість рідкісних біграм: {count}")
```

Результат:

```
Топ-10 ключів із найменшою кількістю рідкісних біграм:
1. Ключ (390, 10) - Кількість рідкісних біграм: 5
2. Ключ (554, 598) - Кількість рідкісних біграм: 5
3. Ключ (862, 108) - Кількість рідкісних біграм: 7
4. Ключ (503, 890) - Кількість рідкісних біграм: 12
5. Ключ (641, 960) - Кількість рідкісних біграм: 14
6. Ключ (376, 489) - Кількість рідкісних біграм: 14
7. Ключ (221, 489) - Кількість рідкісних біграм: 16
8. Ключ (80, 788) - Кількість рідкісних біграм: 17
9. Ключ (771, 536) - Кількість рідкісних біграм: 18
10. Ключ (571, 351) - Кількість рідкісних біграм: 18
```

В даній частині коді перевіряємо кожний декодований текст на наявність рідкісних біграм і сортуємо їх за зростанням і вибираємо топ-10

Перевіремо перший ключ і записуємо в файл:

```python
final_key = (390, 10)

# Отримуємо розшифрований текст за фінальним ключем
decrypted_text = decrypted_variants[final_key]
output_file_name = "decrypted_text.txt"
with open(output_file_name, "w", encoding="utf-8") as output_file:
    output_file.write(decrypted_text)

print(decrypted_text)

print(f"Розшифрований текст збережено у файл: {output_file_name}")
```

Відкритий текст:

еслиправдачтодостоевскийвсибиринебылподверженприпадкамтоэтолишьподтв
ерждаетточтоегоприпадкибылиегокаройонболеевнихненуждалсякогдабылкарае
минымобразомнодоказатьэтоневозможноскорееэтойнеобходимостьювнаказани
идляпсихическойэкономиидостоевскогообясняетсяточтоонпрошелнесломленны
мчерезэтигодыбедствийиунижений осуждениедостоевскоговкачествеполитичес
когопреступникабылонесправедливымиондолженбылэтознатьноонпринялэтоне
заслуженноенаказаниеотбатюшкицарякакзаменунаказаниязаслуженногоимзасв
ойгрехпоотношениюксвоемусобственномуотцувместосамонаказанияондалсебя
наказатьзаместителюотцаэтодаетнамнекотороепредставлениеопсихологическом
оправданиинаказанийприсуждаемыхобществомэтонасамомделетакмногиеизпре
ступниковжаждутнаказанияеготребуетихсверхяизбавляясебятакимобразомотса
монаказаниятотктознаетсложноеиизменчивоезначениеистерическихсимптомов
пойметчтомыздесьнепытаемсядобитьсясмыслаприпадковдостоевскогововсейпо
лнотедостаточнотогочтоможнопредположитьчтоихпервоначальнаясущностьост
аласьнеизменнойнесморянавсепоследующиенаслоенияможносказатьчтодостоев
скийтакникогдаинеосвободилсяотугрызенийсовестивсвязиснамерениемубитьот
цаэтолежащеенасовестибремяопределилотакжеегоотношениекдвумдругимсфер
ампокоющимсянаотношениикотцукгосударственномуавторитетуикверевбогавп
ервойонпришелкполномуподчинениюбатюшкецарюоднаждыразыгравшемусни
мкомедиюубийствавдействительностинаходившуюстолькоразотражениевегопр
ипадкахздесьверхвзялопокаяниебольшесвободыоставалосьунеговобластирелиг
иознойпонедопускающимсомненийсведениямондопоследнейминутысвоейжизн
ивсеколебалсямеждуверойибезбожиемеговысокийумнепозволялемунезамечатьт
етрудностиосмысливаниякоторымприводитверавиндивидуальномповторениим
ировогоисторическогоразвитияоннадеялсявидеалехристанайтивыходиосвобожд
ениеотгреховииспользоватьсвоисобственныестраданиячтобыпритязатьнарольхр
истаеслионвконечномсчетенепришелксвободеисталреакционеромтоэтообясняет
сятемчтообщечеловеческаясыновняявинанакоторойстроитсярелигиозноечувств
одостиглаунегосверхиндивидуальнойсилыинемоглабытьпреодоленадажееговыс
окойинтеллектуальностьюздесьнасказалосьбыможноупрекнутьвтомчтомыотказ
ываемсяотбеспристрастностипсихоанализаиподвергаемдостоевскогооценкеиме
ющейправонасуществованиелишьспристрастнойточкизренияопределенногомир
овоззренияконсерваторсталбынаточкузренияявеликогоинквизитораиоценивалбы
достоевскогоиначеупрекправедливдляегосмягченияможнолишьсказатьчтореш
ениедостоевскоговызваноочевиднозатрудненностьюегомышленияявследствиене
врозаедвалипростойслучайностьюможнообяснитьчтотришедеврамировойлитер
атурывсехвременктрактуютоднуитужетемутемуотцеубийствацарьэдипсофоклага
млетшекспираибратьякарамазовыдостоевскогововсехтрехраскрываетсяимотивд
еяниясексуальноесоперничествоиззаженщиныпрямеевсегоконечноэтопредставл
еновдрамеоснованнойнагреческомсказаниииздесьдеяниесовершаетсяещесамимге
роемнобезсмягченияизавуалированияпоэтическаяобработканевозможнаоткрове

нноепризнаниевнамерениииубитьотцакакогомыдобиваемсяприпсихоанализекаж
етсянепереносимымбезаналитическойподготовкивгреческойдраменеобходимое
смягчениеприсохраненииисущностимастерскидостигаетсятемчтобессознательны
ймотивгерояпроецируетсявдействительностькакчуждоеемупринуждениенавяза
нноесудьбойгеройсовершаетдеяниенепреднамеренноиповсейвидимостибезвлия
нияженщиныивсежеэтостечениеобстоятельствпринимаетсяврасчеттаккаконмож
етзавоеватьцарицуматьтолькопослеповторениятогожедействиявотношениичудо
вищасимволизирующегоотцапослетогокакобнаруживаетсяиоглашаетсяеговинан
еделаетсяникакихпопытокснятьеессебявзвалитьеенапринуждениесосторонысуд
ьбынаоборотвинапризнаетсяикаквсецелаявинанаказываетсячторассудкуможетп
оказатьсянесправедливымнопсихологическиабсолютноправильнованглийскойд
рамеэтоизображеноболеекосвеннопоступоксовершаетсянесамимгероемадругим
длякоторогоэтотпоступокнеявляетсяотцеубийствомпоэтомупредосудительныйм
отивсексуальногосоперничествауженщиныненуждаетсявзавуалированииравнои
эдиповкомплексгероямывидимкакбывотраженномсвететаккакмывидимлишьток
акоедействиепроизводитнагерояпоступокдругогоондолженбылбызаэтотпоступо
котомститьностраннымобразомневсилахэтосделатьмызнаемчтоегорасслабляетс
обственноечувствовинывсоответствиисхарактеромневротическихявленийпроис
ходитсдвигичувствовиныпереходитвосознаниесвоейнеспособностивыполнитьэт
озаданиепоявляютсяпризнакитогочтогеройвоспринимаетэтувинукаксверхиндив
ндуальнуюонпрезираетдругихнеменеечемсебяеслиобходитьсяскаждымпозаслуг
амктоуйдетотпоркивэтомнаправлениироманрусскогописателяуходитнашагдаль
шеиздесьубийствосовершенодругимчеловекомоднакочеловекомсвязаннымсуби
тымтакимижесыновнимиотношениямикакигеройдмитрийукоторогомотивсексуа
льногосоперничествоткровеннопризнаетсясовершенодругимбратомкоторомук
акинтереснозаметитьдостоевскийпередалсвоюсобственнуюболезньякобыэпиле
псиютемсамымкакбыжелаясделатьпризнаниечтомолэпилептикневротиквомнеот
цеубийцаивотвречизащитниканасудетажеизвестнаянасмешканадпсихологиейон
амолпалкаодвухконцахзавуалировановеликолепнотаккакстоитвсеэтоперевернут
ьинаходишьглубочайшуюсущностьвосприятиядостоевскогозаслуживаетнасмеш
киотнюдьнепсихологияасудебныйпроцессдознаниясовершеннобезразличноктоэ
тотпоступоксовершилнасамомделепсихологияинтересуетсялишьтемктоеговсво
емсердцежелаликтопоегосовершенииегоприветствовалипоэтомувплотьдоконтр
астнойфигурыалешивсебратьяравновиновныдвижимыйпервичнымипозывамиис
кательнаслажденийполныйскепсисациникиэпилептическийпреступниквбратьях
карамазовыхестьсценаввысшейстепенихарактернаядлядостоевскогоизразговора
сдмитриемстарецпостигаетчтодмитрийноситвсебеготовностькотцеубийствуибр
осаетсяпереднимнаколениэтонеможетвлятьсявыражениемвосхищенияадолжно
означатьчтосвятойотстраняетотсебяискушениеисполнитьсяпрезрениемкубийце
илиимпогнушатьсяипоэтомупереднимсмиряетсясимпатиядостоевскогокпреступ
никудействительнобезграничнаонадалековыходитзапределысостраданиянакото
роенесчастныйимеетправоонанапоминаетблагоговениескоторымвдревностиотн

осилиськэпилептикуидушевнобольномупреступникдлянегопочтиспасительвзяв шийнасебявинукоторуювдругомслучаенеслибыдругиеаа

Висновок:

Ми здобули практичні навички частотного аналізу на прикладі моноалфавітних підстановок, а також отримали досвід з засобами роботи в модулярній арифметиці. Змогли розшифрувати такст і відкинути неправильно розшифровані тексти за допомогою всіх кандидатів на ключ.