

Міністерство освіти і науки України

Національний технічний університет України
"Київський політехнічний інститут імені Ігоря
Сікорського"

Фізико-технічний інститут

Криптографія

Лабораторна робота №3

Виконав студент групи ФБ-13

Лагно Костянтин

Криптоаналіз афінної біграмної підстановки

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

Порядок виконання роботи:

0. Уважно прочитати методичні вказівки до виконання комп'ютерного практикуму.
1. Реалізувати підпрограми із необхідними математичними операціями: обчисленням оберненого елемента 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 №1, 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 (1).
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. Функції для математичних операцій: пошук НСД, розширений алгоритм Евкліда, пошук оберненого та розв'язування лінійних порівнянь:

```
def gcd(a, b):
    while b:
        a, b = b, a % b
    return a

def extended_gcd(a, b):
    if a == 0:
        return b, 0, 1
    else:
        g, x, y = extended_gcd(b % a, a)
        return g, y - (b // a) * x, x

def mod_inverse(a, m):
    g, x, _ = extended_gcd(a, m)
    return x % m if g == 1 else None

def solve_linear_congruence(a, b, m):
    roots = []
    a, b = a % m, b % m
    g = gcd(a, m)
    if g == 1:
        a_inv = mod_inverse(a, m)
        roots.append((a_inv * b) % m)
        return roots
    elif g > 1 and b % g == 0:
        a1 = a // g
        b1 = b // g
        m1 = m // g
        roots = solve_linear_congruence(a1, b1, m1)
```

```

    roots.extend((roots[0] + m1 * i) % m for i in range(g))
    return roots
else:
    return None

```

Результати виконання:

```

37     else:
38         return None
39
40
41     print(gcd(17, 18))
42     print(mod_inverse(56, 221))
43     solve_linear_congruence(5, 16, 22)

```

test x

C:\Users\ACER\AppData\Local\Microsoft\WindowsApps\python3.10.exe C:\Users\ACER\Desktop\Крипта\crypto-23-24\cp3\cp3_fb-13_lahno\test.py

1
75
[12]

Process finished with exit code 0

Перевірка:

Знайти НСД і НСК двох чисел

і

Найменше спільне кратне (НСК)	Найбільший спільний дільник (НСД)
306	1

$5x = 16 \pmod{22}$

NATURAL LANGUAGE
 MATH INPUT
 EXTEND

Input interpretation

solve

5 x ≡ 16 (mod 22)

Solution in the least residue system

$x \equiv 12 \pmod{22}$

Input

$56^{-1} \pmod{221}$

Result

75

2. Для виконання цієї частини завдання, я використав частини коду з першої лабораторної, а саме форматування тексту (хоч він вже і був без пробілів і лишніх символів, все одно залишились перенесення рядків), і пошук і підрахунок біграм без перетину зашифрованого тексту. Ці функції я трохи модифікував, підігнавши під основне завдання, тому функція підрахунку біграм видає перші 10 біграм з найбільшою частотою. Програму прикріплю окремо файлом з назвою bigrams.py.

В результаті виконання я отримав список найчастіших біграм мого тексту:

```
if __name__ == "__main__":
    bigrams.py
C:\Users\ACER\AppData\Local\Microsoft\WindowsApps\p
Введіть ім'я файлу для читання: 12.txt
Біграма 2 'хк': 0.017722954771019424
Біграма 2 'ек': 0.01446193109315185
Біграма 2 'вю': 0.012051609244293209
Біграма 2 'пн': 0.012051609244293209
Біграма 2 'вх': 0.012051609244293209
Біграма 2 'дп': 0.011626258329788742
Біграма 2 'лю': 0.010917340138947965
Біграма 2 'ок': 0.009783071033602722
Біграма 2 'юз': 0.009357720119098255
Біграма 2 'ху': 0.0092159364809301

Process finished with exit code 0
```

Їх я потім і використовуватиму в процесі розшифрування. Сам же оброблений текст записав в окремий файл var12.txt.

3-5. Тут починається найцікавіше. Розібравшись з завданням, я написав більш-менш робочий і зв'язний код для пошуку ключа і розшифрування:

```
def find_possible_keys(mcrb, mccb, alphabet):
    possible_keys = []
    for i in range(len(mcrb)):
        for j in range(len(mcrb)):
            if i != j:
                for k in range(len(mccb)):
                    for v in range(len(mccb)):
                        if k != v:
                            x1 = bigram_to_numeric(mcrb[i], alphabet)
                            y1 = bigram_to_numeric(mccb[k], alphabet)
                            x2 = bigram_to_numeric(mcrb[j], alphabet)
                            y2 = bigram_to_numeric(mccb[v], alphabet)
                            x = x1 - x2
                            y = y1 - y2
                            if gcd(x, alph_sq) == 1 and mod_inverse(x, alph_sq):
                                c = solve_linear_congruence(x, y, alph_sq)
                                if c:
                                    for a in c:
                                        if a > 0 and gcd(a, alph_sq) == 1 and
mod_inverse(a, alph_sq):
                                            b = (y1 - a * x1) % alph_sq
                                            possible_keys.append((a, b))
    return possible_keys
```

```

def bigram_to_numeric(bigram, alphabet):
    index1 = alphabet.index(bigram[0])
    index2 = alphabet.index(bigram[1])
    numeric = index1 * len(alphabet) + index2
    return numeric

def numeric_to_bigram(int, alphabet):
    bigram = ""
    letter1 = int // 31
    letter2 = int % 31
    bigram += alphabet[letter1]
    bigram += alphabet[letter2]
    return bigram

def decrypt_text_from_file(text, keys, alphabet):
    decrypted_texts = []
    for key in keys:
        decrypted = ""
        for letter in range(0, len(text) - 1, 2):
            bigram = text[letter: letter + 2]
            y = bigram_to_numeric(bigram, alphabet)
            c, b = key
            x = mod_inverse(c, alph_sq) * (y - b) % alph_sq
            decrypted += numeric_to_bigram(x, alphabet)
        decrypted_texts.append([key, decrypted])

    return decrypted_texts

```

Для розшифрування використовував наступні вхідні дані:

```

alphabet = "абвгдежзийклмнопрстуфхцчшщъыэя"
alph = 31
alph_sq = alph**2
mcrrb = ['ст', 'но', 'то', 'на', 'ен'] # most common russian bigrams
mccb = ['хк', 'ек', 'вю', 'пн', 'вх'] # most common ciphered bigrams
mcrl = ['о', 'е', 'а'] # most common russian letters
mrll = ['ф', 'щ', 'ь'] # most rare russian letters

```

Біграми відкритого тексту взяв з методички, так само як і найчастіші і найрідкісніші букви. Перейшовши до самого розшифрування, почались проблеми. Все ніяк не виходило розшифрувати текст, а в результаті, після довгого дебагу і перебору усіх ключів (їх на диво було багато), виявилось що я просто в функції `numeric_to_bigram` переплутав місцями букви. Тому в мене виходило розшифрувати текст, але він був нечитабельним, бо букви в біграмах були поміняні місцями. Відповідно, потрібно було дописати ще функцію для перевірки тексту на його умовну змістовність. З допомогою, мені таки вдалось написати таку функцію, в якій рахуються найчастіші і найрідкісніші букви кожного тексту і порівнюються з заданими такими буквами. При певній кількості співпадінь, виводиться правильно розшифрований текст. В моєму випадку при > 4 співпадіннях зразу вивело правильний текст.

Сам код функції:

```

def find_readable_text(text):
    counter = {char: text.count(char) for char in set(text) if char.isalpha() or

```

```

char.isspace()})
    sorted_letters = dict(sorted(counter.items(), key=lambda item: item[1],
reverse=True))
    mcl = list(sorted_letters.keys())[:3]
    mrl = list(sorted_letters.keys())[-3:]
    return sum(i in mcl for i in mcl) + sum(j in mrrl for j in mrl)

def final_text_checking(texts):
    for key, text in texts:
        if find_readable_text(text) > 4:
            return [key, text]

```

Результат виконання повної програми:

```

a = 398 , b = 571
a = 750 , b = 943
a = 840 , b = 230
a = 705 , b = 424
a = 519 , b = 424
a = 871 , b = 796
a = 121 , b = 331
Введіть ім'я файлу для читання: var12.txt
Введіть ім'я файлу для запису: decrypted.txt

Текст розшифровано з ключем (555, 331)

Process finished with exit code 0
|

```

Зашифрований текст (уривок):

Оклязогтдхвоэштжсэллыэежяхбчехеквюхуашайейллокийбятйвндкпйзмйхшкшэхкиккптжп
нуафйнрцэзкурлюхжыгэвхбкдкшхяиожцбтхлцймтдцгхуюклцбьшнцмешврчцчьтакугэвжйулн
екчвоцмахухкпюукгхупнфйллрзгдзкдохксавхцляплэенцяхкпюлюбйчдгцякбуейяждохксавхц
ляяхкпюлюбштжхквгейчдгцхнцмыжшжхжэьехтшфхййвнхкйеокгхйэоцмахутжджюхбжяежвб
урзещежяхбчехжфлююющчэюхуевхвюйшыьоржлвнкбгхвюдшчжуастхжыгэжвашцэежяюэра
нмафйфквхябчжцжшхкктхзюяжябтмюхййвнгцдпгфздьлпнзюеябзюгцщдэтлюзжжэпппшт

Розшифрований текст (уривок):

Когда пожарные соседи ушли леоауфманосталсясдешушкойсполдингомдугласомитомомвсеон
изадумчивосмотрелинадогорающиеостаткигаражалеоткнулногойвмокрууюзолуимедленновыс
казалточтолежалонадушепервоечтоузнаешьвжизниэточтотыдуракпоследнеечтоузнаешьэточто
отывсетотжедуракмногоепередумалязодинтолькочасисказалсебедаведьтыслепойлеоауфман
хотитеувидатьнастоящую

Виглядає наче уривок з книги Рея Бредбері - Вино з кульбабок.

Всі вихідні файли і результати окремо надішлю.

Висновок: в ході виконання даної лабораторної роботи, я опанував навички використання частотного аналізу, модулярної арифметики, розширеного алгоритму Евкліда і алгоритму розв'язку лінійних порівнянь для аналізу, пошуку ключа та розшифрування тексту, зашифрованого афінним шифром.