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

# Криптологія
## Комп’ютерний практикум Робота №3

Виконав стдент групи ФБ-14
Шовкун Богдан

Київ-2023

**Мета роботи**

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

Варіант 10

Фрагмент шифротексту.

```
юммутмкйсйумтцбишчоцхйнкхйяхклзкугргтвдньлмгсбтмейашрэлшогэгклсмтцзлжфбтдлычтфыляунгфищйэзргччсбьцжм
ьижнулхщьлкюэклксыямямбйжцтпогсбэищмзмхшсчмддуилойэйугюцдтруцвдуампзэйбуззцюжнвкбхгвргфбчишчжпэгкнрш
зццлбгвптмвннггшргэмбхогрирумчилцнвцвпжэбтцтвпеэлкжээйуццлкшцбоцюццнзлмчяуфбяцтбжэеэсйлкдмеццатмбцца
ймумгхьцнгюцьцдхшзеэпнсбжэящбгилнгтзяунивпеэсмямешуйэйшйэсозгкшйяментэьхрхюзгкхййзгкнйфцгзбйаьыцрхрг
ычксдяьцзллтлгтмбхьэецшзюхщйххшзтбьцтмтэеэьхрхжэпмтэсмжкбйаьыцкирийзсмеивфзэсглйллжмцкэсткяжюццтдлкш
укикейяржэййнгзлахрхмйщмймммнзиьцтмнипйхуфуббьлвфоцлйямярсмюмгчжюфбфмтмжжкэекчэзмхйейаклккддуилридйщй
укнйриеэцукссмтцбгопямржшйцлритбмкэбьцогфгебээяждивфбшквдусгбгвццнтчойцкщбдярхтфьлжнммжэцксмхуоцяффу
ббкияйэншцфуклщмяжллфгргяэувшчбубиьлцнммейцюклримвнйяэзвьлыщебрвщйзиычелцнфихйщнзимйтэзцяфебнгтммйсг
гнзиьцтмугклемшйхбщюржгэыцлдргзлафжтзэфкцрагахрхтбилдтвпшчейоэбкагебэеыццнеэдэьэюцоцрхюцфщшйдшрцшмжэ
жэшушмямшиьлйзюцтмыцщюгэдшрцтввфцтргогкнзкдуешюказдмзипйбшрээпямршгэвнтэикцрхктмеэыгыцхчфужмсчмйтцюх
днейщйжнсмвксйбхмйвпхкгнлкймвчбгфипйагригйсгнийзхкхпзлриэхюмишсэнгцлатшйэнзкбхиянггэгэагмкгнцргпцнцр
гнвпыцогфгрхксдктчксгйуэцкгшбшрэзммпямпшбхзаопзлриэхгвумэмулыццюкклктмнгэфчэилмйщмошяэнгфйзнюццкйэжм
ьиычвпяйэнлиюмсгзлоиястмгржэсчклнгвушеьйщмнчыфэгдэомбжбипйзэщцргдмпбэеыцфуцнейошгпцнцрэхрхтэрэюксйбх
шзечявыцеэймлкугжтумэйжэдмчйоцоияптмумцуяйэнклрюжтзэгкьэвнсбычниймззтмпчцнзиычщгюцшчцнекмнбткинйдйшх
цлкектжкрмвчгнгктэгшцбэеыцявтнцрсбфмзингогжэагргдмсчжныййжибщчпшсгммзингогкйбрщмумпмывтнцрргкйгкояьц
ьлыццношэйицфбкляэьэнкпзузпмьнцктцрхтпзлйдмйсбкляэцкцюриычилизяцлйвньхмдллринйргчяцшэмяхдэсчахыццлщм
змхшдншшфэяхрхсиуьнгкйейыилшсгцалксгкйэмлшсйщмдздэшйдэржчэнюриыычилюцпйфйсыямгезлгнблднцрниямбйдшбйоу
зззиойибэеыцыпшчяфбкдшршкйзэоушмпшсгзлгвтнбжэхтпзлыцпвдуикрцтвриднлкмгсбтмейхйбшшхкшйждццлйздмяцемэв
впэисгдмчэтыршцлатгшруоиилйзуарвумяццлйзюмнйоээцсбтмщмсчдншшфэеньхпйфйрвюцелрггшгэюцбирзьэбрдтимдшлл
жеьлшйугщюычтфпмомнйнииятмьцфязлрвьцмйжгхщилкезлгнлбклдуцалксгжжеэцуращмеццнршсэгуьцдхщюгэюцдмзгрхсч
жнмвьлйзхягэтыямщмшимхрхргычравфжэяэбпямейдйлрямдянйямжэяэзмбрюмжэсгизэйошюцюцтпыцьлжнфиргмгрхойэгум
смхурхямцкашзрафпмэмяэящниилчюжнейжвнйэзицюхлжцклкщцмхжжкамукэцьлдязагшюмхйщмахтмбкгйтэйлапгнхкьцчякк
увхщщхешоэафкйтзюмбшексгцанйфцуатчбипвчкнйрцимфйнэлклйюмечшзтццннишчжэдмсгахжэяюжэдэсэпмвклжчэлипйгй
крекйкгйкрекэхчэгчбгщхжнилеэшиюцсмсгщйсмтэьэафюмецчяуавчыфнэлкщмзмыхюхнйкйзэщйзхмйрмзмбжйкбхххждщзлни
екквжннйзтычшмбшчэмпюхриошзэыррвьлвпхкеэгйшхжмсгбгхксчмезмщмеццютвзэмутмчэцкэпнгкйыгщюмкплгмаюычзлий
```

**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.

Спочатку визначаю алфавіт та створюю функцію для  перетворення біграм у числове представлення для подальшої роботи.

```
def bigrams_to_numbers(bigrams, alphabet):
    return [alphabet.index(bigram[0]) * len(alphabet) + alphabet.index(bigram[1]) for bigram in bigrams]
```

```
alphabet = 'абвгдежзийклмнопрстуфхцчшщьыэюя'
mod = len(alphabet)
rus_bigram = ["ст", "но", "то", "на", "ен"]
```

Реалізував обчислення оберненого елемента за модулем із використанням розширеного алгоритму евкліда.

```
def mod_inverse(a, m):
    def gcd_extended(a, m):
        if m == 0:
            return a, 1, 0
        else:
            gcd, x, y = gcd_extended(m, a % m)
            return gcd, y, x - (a // m) * y

    gcd, x, y = gcd_extended(a, m)
    if gcd != 1:
        return None
    else:
        return (x % m + m) % m
```

Реалізував також лінійні рівняння в функції find_keys().

```
def find_keys():
    keys = []
    for bigram1 in rus_bigram:
        for bigram2 in text_bigram:
            for bigram3 in rus_bigram:
                for bigram4 in text_bigram:
                    if bigram1 != bigram3 and bigram2 != bigram4:
                        x1, y1 = bigrams_to_numbers( bigrams: [bigram1], alphabet)[0], bigrams_to_numbers( bigrams: [bigram2], alphabet)[0]
                        x2, y2 = bigrams_to_numbers( bigrams: [bigram3], alphabet)[0], bigrams_to_numbers( bigrams: [bigram4], alphabet)[0]
                        if mod_inverse(x1 - x2, mod**2):
                            a = (y1 - y2) * mod_inverse(x1 - x2, mod**2) % mod**2
                            b = (y1 - a * x1) % mod**2
                            keys.append((a, b))
    print(f'All keys: {keys}')
    return keys
```

```
a = (y1 - y2) * mod_inverse(x1 - x2, mod**2) % mod**2
b = (y1 - a * x1) % mod**2
```

2. За допомогою програми обчислення частот біграм, яка написана в ході виконання комп’ютерного практикуму №1, знайти 5 найчастіших біграм запропонованого шифртексту (за варіантом).

Знаходжу кількість біграм та іх частоту. Виводжу 5 найчастіших .

```
def calculate_bigram_and_frequency(text):
    bigrams_list = []
    i = 0
    while i < len(text) - 1:
        bigram = text[i:i+2]
        bigrams_list.append(bigram)
        i += 2

    bigram_frequency = Counter(bigrams_list)
    most_common_bigrams = [bigram for bigram, frequency in bigram_frequency.most_common(5)]
    print(f"text_bigrams = {most_common_bigrams}")
    return most_common_bigrams, bigrams_list
```

Ось результат:

```
text_bigrams = ['сг', 'жэ', 'ям', 'нг', 'тм']
```

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.

Я використовував чотири вкладені цикли:

```
for bigram1 in rus_bigram:
    for bigram2 in text_bigram:
        for bigram3 in rus_bigram:
            for bigram4 in text_bigram:
```

Вкладені цикли потрібні для перебору найчастіших біграм  російської мови та найчастіших біграм з тексту. Ці чотири вкладені цикли створюють всі можливі комбінації пар біграм.
bigram1,  bigram3 беруться з rus_bigram, a bigram2, bigram4 з text_bigram.

Далі йде умова: `if bigram1 != bigram3 and bigram2 != bigram4:`

Вона забезпечує, що bigram1 та bigram3 не є однаковими, а bigram2 та bigram4 також. Це робиться для того, щоб уникнути дублювання ключів.

Пілся цього для кожної пари обчислюються потенційні ключі шляхом розв’язання систем лінійних рівнянь:

```
x1, y1 = bigrams_to_numbers([bigram1], alphabet)[0],
bigrams_to_numbers([bigram2], alphabet)[0]
x2, y2 = bigrams_to_numbers([bigram3], alphabet)[0],
bigrams_to_numbers([bigram4], alphabet)[0]
if mod_inverse(x1 - x2, mod**2):
    a = (y1 - y2) * mod_inverse(x1 - x2, mod**2) % mod**2
    b = (y1 - a * x1) % mod**2
    keys.append((a, b))
```

Тут x1, x2, y1, y2 є числовими представленнями біграм, a mod_inverse(x1 - x2, mod**2) це обернене значення різниці між x1 та x2 за модулем mod**2.
Якщо обернене значення існує то обчислюються потенційні ключі (a, b) та додаються до спику ключів Keys = [].

Таким чином вкладені цикли дозвлдяють перебрати всі можливі комбінації біграм та знайти всі можливі ключі для дешифрування.

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

```
usage
def decode(keys, bigrams, alphabet):
    for key in keys:
        a_inv = mod_inverse(key[0], mod**2)
        if a_inv:
            b = key[1]
            decrypted_bigrams = [(bigram - b) * a_inv % mod**2 for bigram in bigrams]
            decrypted_text = "".join([alphabet[bigram // mod] + alphabet[bigram % mod] for bigram in decrypted_bigrams])
            if all(l_grams not in decrypted_text for l_grams in ['уь', 'юь', 'еь', 'эь', 'эы', 'фй', 'ьь', 'юы', 'оь', 'дй', 'яь']):
                print(f"Successful key: {key}")
                return decrypted_text, key
    return None, None
```

Для кожного ключа функція обчислює обернене значення a, за умови, якщо воно існує, та використовує його для розшифрування біграм. Розшифрований текст перевіряється на наявність l_gram (заборонених біграм). Якщо іх нема в тексті, то ключ вважається успішним і функція повертає розшифрований текст та успішний ключ. Якщо жоден ключ не був успішним - то функція повертає None для тексту а ключа.

```
text_bigrams = ['сг', 'жэ', 'ям', 'нг', 'тм']
All keys: [(513, 596), (207, 152), (31, 933), (300, 400), (451, 751),
Successful key: (300, 400)

Process finished with exit code 0
```

Розшифрований текст:

```
поздновечеромнаверандесиделколяичтотописалвтемнотебумагуитутолкомнельзябылоразглядетьвремяотвре
нулаточновсеткуотмоскитовудариласьночнаябабочкалинашепнулауфманонаселарядомснимнакачеливоднойно
уюуженелюбятноскладнаяикрепкаяименнотакаякакнадотаковыженщинывовсякомвозрастееслионилюбимыонабы
еовкаждуюкомнатуучтобынеуловимоизменитьтамсамыйвоздухподстатьнастроениюмужаказалосьонаникогданез
енночтолеонесмогбыдаинехотелизобразитьэтокакимилибочертежамиэтамашинасказалаонанаконецненужнаон
еоскопическиеочкиеслисобратьвсеэтовместевсякийчеловекпощупаетулыбнетсяискажетдадаэтоиестьсчасть
аетсявпостеливсюночьнапролетидушуегогрызутзаботыавсеравнотвоямашинадастемусчастьекактамагическа
изобреститакуюмашинупустьемуответитнаэтотвопросцелыймирпустьответитвесьгородокпустьответитженал
```

## Висновки

В цій роботі я проаналізував та зламав шифр афінної біграмної підстановки. Також засвоїв прийому модулярної арифметики на практиці. За допомогої найчастіших біграм шифротексту та найуживаніших біграм російської мови склав систему рівнянь за допомогою якої отримав потенційні ключі шифрування. Таких ключів було багато, тому за допомогою методу відсіювання я знайшов правильний ключ.