Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт
Кафедра Информатики
кафедра

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ

Основы криптоанализа шифров полиалфавитной подстановки
тема

Преподаватель _____ М.М. Кучеров
подпись, дата | инициалы, фамилия

Студент КИ19-17/1б 031940142 _____ С.Е. Говорухина
номер группы, зачетной книжки | подпись, дата | инициалы, фамилия

Студент КИ19-17/1б 031940427 _____ М.А. Луцив
номер группы, зачетной книжки | подпись, дата | инициалы, фамилия

Студент КИ19-17/1б 031940142 _____ А.К. Никитин
номер группы, зачетной книжки | подпись, дата | инициалы, фамилия

Красноярск 2021

1 Цель

- введение в криптосистемы полиалфавитной подстановки на примере шифра Виженера;

- изучение методов криптоанализа шифров полиалфавитной подстановки на примере шифра Виженера.

2 Описание варианта задания

Используя любой из способов криптоанализа (рассмотренных ранее) шифров полиалфавитной подстановки расшифруйте текст сообщения (согласно вашему персональному варианту), а также определите секретный ключ, которым оно было зашифровано.

Известные сведения о шифрограмме следующие:

- для зашифрования сообщения, которое представляет собой отрывок из литературного произведения, использовался шифр Виженера, с алфавитом, представленным в таблице;

- знаки препинания, а также знак пробела не входят в алфавит шифрования и, соответственно, не является частью шифротекста, т.е. шифротекст с расставленными знаками препинания и пробелами эквивалентен шифротексту без них;

- весь шифротекст представляет собой одну строку;

- в качестве ключа шифрования использовалось одно слово (из букв алфавита), длина которого не более 10 символов.

Для осуществления статистического криптоанализа необходимо разработать программу для дешифрования со следующими возможностями:

- автоматическое приближённое нахождение периода шифрования (числа μ) с помощью теста Касиски;

- автоматическая проверка правильности выбора μ с помощью нахождения индексов соответствия (для всех подгрупп шифротекста);

- в случаях, когда тест Касиски не удаётся применить или значение μ, найденное с его помощью, не подтверждается после определения индексов соответствия, μ определяется подбором (такой случай возможен в случаях, когда длина ключа более 7 знаков и/или зашифрованный текст слишком короткий);

- автоматическое вычисление всех потенциально возможных вариантов ключа шифрования, и вывод их пользователю для принятия решения;

- расшифровка шифротекста, по выбранному пользователем ключу шифрования.

3 Ход выполнения

Для проверки работы программы был выбран вариант 15.

3.1 Теоретическая часть

В общем, алгоритм теста можно описать следующим образом:

1. В шифротексте ищутся повторяющиеся последовательности букв длиной не менее трёх (триграммы) и расстояния между ними (например, в шифрограмме рхцэооэцг**БРЬ**цмйфктъъюьмшэсяцпунуящэйтаьэдкци**БРЬ**цгбр расстояние между триграммами равно 35);

2. Находится наибольший общий делитель (НОД) полученных чисел (расстояний);

3. Если он больше 1 и меньше 10 (в нашем случае известно, что ключ не более 10 символов) то выдвигаем гипотезу, что НОД является значением μ и приступаем к проверке этой гипотезы с помощью индексов соответствия;

4. Если гипотеза подтвердилась, то переходим к расшифровке, в противном случае снова переходим к п.п. 1 и осуществляем поиск других триграмм.

3.2 Программная реализация

На листингах 1-4 представлены реализацияя алгоритмов для дешифрования текста, зашифрованного шифром Виженера.

Листинг 1 – Реализация теста Касиски

```
def sort_dict_by_values(x: dict, reverse=False) -> dict:
    """Сортирует словарь по значениям"""
    return {k: v for k, v in sorted(x.items(), key=lambda item: item[1],
reverse=reverse)}


def find_repetitions_pos(message: str, length=3) -> dict:
    """Ищет повторения заданной длинны в массиве текста и возвращает словарь с
их позициями"""
    repetition_dict = {}
    for pos in range(len(message) - 2):
        section = message[pos: pos + length]

        if section not in repetition_dict:
            repetition_dict[section] = [m.start() for m in re.finditer(section,
message)]

    return dict((k, v) for k, v in repetition_dict.items() if len(v) > 1)


def compute_gcd(*args):
    """Находит НОД нескольких чисел"""
    x = args[0]

    for y in args[1:]:
        while y:
            x, y = y, x % y

    return x


def get_intervals_list(repetitions: dict) -> list:
    """Получает список интервалов из списка позиций повторений"""
    intervals = []
    for rep_list in repetitions.values():
        intervals.extend([rep_list[i + 1] - rep_list[i] for i in
range(len(rep_list) - 1)])
    return intervals


def calculate_intervals_gcd(repetitions: dict) -> dict:
    """Считает НОД для всех пар растояний и возвращает словарь {пара: НОД}"""
    intervals = get_intervals_list(repetitions)

    print('Список с интервалами повторениями:\n')
    print(intervals, '\n')

    gcd_list = []
    for i in range(len(intervals)):
        for j in range(i + 1, len(intervals)):

            if (gcd := compute_gcd(intervals[i], intervals[j])) != 1:
```

Окончание листинга 1

```
                gcd_list.append(gcd)

    gcd_frequencies = Counter(gcd_list)
    return sort_dict_by_values(dict((k, v) for k, v in gcd_frequencies.items() if v > 1), reverse=True)


def kasiski_test(cipher: str) -> int:
    """Находит длину ключа при помощи теста Касиски"""
    repetitions = find_repetitions_pos(cipher)
    print('Словарь с повторениями последовательности символов длиной больше 3:\n')
    print(repetitions, '\n')

    gcd_freq = calculate_intervals_gcd(repetitions)
    print('Словарь с количеством повторений НОД между каждыми интервалами интервала:\n')
    print(gcd_freq, '\n')

    return max(gcd_freq, key=gcd_freq.get)
```

Листинг 2 – Реализация проверки правильности длины ключа

```
def validate_cipher_period(cipher: str, period: int):
    """Проверяет, действительно ли длина ключа такая или нет"""
    n = len(cipher)
    freq = Counter(cipher)
    affinity_index = reduce(lambda prev, cur: prev + cur * (cur - 1), freq.values(), 0) / (n * (n - 1))

    borders = AFFINITY_INDEXES.loc[[period]]
    return float(borders['min']) <= affinity_index <= float(borders['max'])
```

Листинг 3 – Реализация алгоритма подбора ключа заданной длины

```
def form_frequency_dict(text: str):
    """Создает словарь, который хранит частоту употребления каждой буквы в тексте относительно всех букв"""
    total = len(text)
    absolute = Counter(text)
    relative = dict([(k, v/total) for k, v in absolute.items()])
    return relative


def calculate_deviation(freq1: dict, freq2: dict):
    """Считает, насколько частота употребления букв одного текста расходится с другим"""
    deviation = 0
    for letter in RUSSIAN_SYMBOLS:
        deviation += abs(freq1.get(letter, 0.035) - freq2.get(letter, 0.035))
    return deviation


def _decrypt_1(cipher: str, caesar_key: pd.Series):
    """Дешифрует часть текста у одной из буквы ключа по данному шифру Цезаря"""
    decrypted_message = ''
    for symbol in cipher:
```

Окончание листинга 3

```
        decrypted_message += caesar_key[symbol]
    return decrypted_message


def guess_key(cipher: str, period: int, stringency=0.6) -> list:
    """Угадывает ключ для шифротекста, анализируя частотность букв. Текст
разбивается на period частей для каждой буквы
    ключа, затем угадывается каждая буква по отдельности. Возвращает список
подходящих под условие ключей"""
    key_letter_variations = ['']
    for pos in range(period):
        pos_letter_message = cipher[pos::period]
        pos_key_letters = []

        for letter, caesar_cipher in VIGENERE_TABLE.iterrows():
            decrypted_cipher_part        =        _decrypt_1(pos_letter_message,
caesar_cipher)
            frequency = form_frequency_dict(decrypted_cipher_part)
            deviation              =              calculate_deviation(frequency,
RUSSIAN_SYMBOLS_FREQUENCY)

            if deviation < stringency:
                pos_key_letters.append(letter)

        new_keys = key_letter_variations.copy()
        for i, key in enumerate(key_letter_variations):
            for letter in pos_key_letters:
                new_keys.append(key + str(letter))
            del new_keys[0]

        key_letter_variations = new_keys.copy()

    return key_letter_variations
```

Листинг 4 – Реализация алгоритма дешифрования

```
def decrypt_cipher(cipher: str, key: str) -> str:
    """Дешифрует текст по ключу"""
    key = repeat_key(cipher, key)
    source_message = ''
    for pos, cipher_letter, key_letter in zip(range(len(cipher)), cipher, key):
        source_message += VIGENERE_TABLE.at[cipher_letter, key_letter]
    return source_message


def restore_signs(source, decrypted):
    """Восстанавливает исходные символы, которые были удалены"""
    decrypted_list = list(decrypted)
    for m in re.finditer('[^А-Я]', source):
        pos = m.start()
        decrypted_list.insert(pos, source[pos])
    return ''.join(decrypted_list)
```

4 Пример работы программы

На рисунках 1-3 представлен пример работы программы на тексте варианта 15 лабораторной работы.

```
D:\System\Desktop\Learning\ИБ\lab3>python main.py
Словарь с повторениями последовательности символов длиной больше 3:

{'ДЪМ': [1, 663], 'ЪМС': [2, 664], 'ЗЖЯ': [19, 681], 'ЙЮН': [24, 479], 'НЫМ': [40, 264, 985], 'ЪЧО': [48, 1021], 'ЧОЮ': [49, 947], 'ЕПЙ': [57, 2
15], 'ЯЖЖ': [60, 375], 'ЖЖС': [61, 376], 'ЖСН': [62, 377], 'СНЬ': [63, 378], 'НЬЭ': [64, 379], 'ЬЭЦ': [65, 380, 723], 'ЭЦТ': [66, 381], 'ЮЬД': [
73, 780], 'ЬДП': [74, 781], 'ДПМ': [75, 782], 'ОЮК': [94, 948], 'ШФЫ': [125, 356], 'ФЫХ': [126, 357], 'ЧЕХ': [133, 522], 'МНФ': [168, 504], 'ЪЛМ
': [176, 933], 'ЕЫА': [183, 407], 'ЫАБ': [184, 408], 'АБК': [185, 409], 'ЦОР': [189, 812, 868], 'ЪМГ': [198, 1038], 'МГЕ': [199, 1039], 'ГЕШ': [
200, 1040], 'ЕШХ': [201, 1041], 'ШХВ': [202, 1042], 'ХВФ': [203, 1043], 'ВФЖ': [204, 1044], 'ФЖС': [205, 1045], 'ЖСЦ': [206, 1046], 'СЦЗ': [207,
 1047], 'ЭЙО': [223, 440], 'ЗКБ': [233, 646], 'МЯИ': [241, 260, 981], 'ОХС': [251, 720], 'ХСЫ': [252, 364], 'ЙМЯ': [259, 980], 'ЯИХ': [261, 982]
, 'ИХН': [262, 983], 'ХНЫ': [263, 984], 'ЧШП': [285, 726], 'ЕСП': [290, 864], 'ПЩЙ': [292, 320], 'ЬМР': [297, 346, 626, 745], 'РШТ': [307, 790],
 'НМЖ': [337, 953], 'ИЧЩ': [340, 473], 'ЪЬМ': [345, 744], 'УАЕ': [391, 654], 'ЫШЕ': [395, 430], 'ЕЦХ': [397, 642], 'ЦИЩ': [424, 494], 'ИЩЙ': [42
5, 495], 'ЬЭХ': [446, 922], 'ССЩ': [452, 914], 'ЫЩА': [482, 713], 'ПЛЧ': [509, 999], 'ЛЧЕ': [510, 1000], 'ЧЕЮ': [511, 1001], 'ХЗК': [524, 902],
'ЩЪЩ': [529, 970], 'ЦЮУ': [549, 852], 'ЧЩУ': [586, 593], 'ЙБУ': [612, 1067], 'ЛФЕ': [622, 1028], 'ФЕШ': [623, 959], 'АЕО': [655, 938], 'ХБЗ': [6
79, 924], 'ЮТЪ': [693, 749], 'ТЬЙ': [694, 750], 'ЬЙЮ': [695, 751], 'КАЗ': [754, 824], 'АЗЛ': [755, 825], 'ЯЖР': [788, 991], 'ТЪЛ': [792, 932], '
ТЦХ': [808, 941], 'ЛИЮ': [966, 973], 'ИЮЧ': [967, 974]}

Список с интервалами повторениями:

[662, 662, 662, 455, 224, 721, 973, 898, 158, 315, 315, 315, 315, 315, 315, 343, 315, 707, 707, 707, 854, 231, 231, 389, 336, 757, 224, 224, 224
, 623, 56, 840, 840, 840, 840, 840, 840, 840, 840, 840, 840, 217, 413, 19, 721, 469, 112, 721, 721, 721, 721, 441, 574, 28, 49, 280, 119, 483, 6
16, 133, 399, 263, 35, 245, 70, 70, 476, 462, 231, 490, 490, 490, 378, 441, 303, 7, 455, 406, 336, 283, 245, 56, 56, 56, 70, 70, 203, 140, 133,
7, 7]

Словарь с количеством повторений НОД между каждыми интервалами интервала:

{7: 1916, 14: 247, 2: 197, 35: 195, 21: 166, 56: 157, 70: 102, 105: 70, 28: 61, 840: 45, 3: 28, 49: 25, 42: 25, 63: 23, 315: 21, 168: 20, 721: 1
5, 112: 14, 140: 11, 280: 10, 245: 7, 224: 6, 231: 6, 662: 3, 707: 3, 101: 3, 77: 3, 19: 3, 133: 3, 490: 3}

Период ключа: 7
```

Рисунок 1 – Нахождение периода ключа

```
Нажмите Enter для продолжения...

Период ключа валиден.
```

Рисунок 2 – Валидация ключа

```
Нажмите Enter для продолжения...

Список возможных ключей:
1. ЩАФЕПЫХ
Введите номер ключа:
1
Расшифрованное сообщение:

С ДОСАДОЮ ЗАКУСИВ ГУБЫ, ВЫШЕЛ ОН ИЗ КОНДИТЕРСКОЙ И РЕШИЛСЯ, ПРОТИВ СВОЕГО
ОБЫКНОВЕНИЯ, НЕ ГЛЯДЕТЬ НИ НА КОГО И НИКОМУ НЕ УЛЫБАТЬСЯ. ВДРУГ ОН СТАЛ КАК ВКОПАННЫЙ
У ДВЕРЕЙ ОДНОГО ДОМА; В ГЛАЗАХ ЕГО ПРОИЗОШЛО ЯВЛЕНИЕ НЕИЗЪЯСНИМОЕ: ПЕРЕД ПОДЪЕЗДОМ
ОСТАНОВИЛАСЬ КАРЕТА; ДВЕРЦЫ ОТВОРИЛИСЬ; ВЫПРЫГНУЛ, СОГНУВШИСЬ, ГОСПОДИН В МУНДИРЕ
И ПОБЕЖАЛ ВВЕРХ ПО ЛЕСТНИЦЕ. КАКОВ ЖЕ БЫЛ УЖАС И ВМЕСТЕ ИЗУМЛЕНИЕ КОВАЛЕВА, КОГДА
ОН УЗНАЛ, ЧТО ЭТО БЫЛ СОБСТВЕННЫЙ ЕГО НОС! ПРИ ЭТОМ НЕОБЫКНОВЕННОМ ЗРЕЛИЩЕ, КАЗАЛОСЬ
ЕМУ, ВСЕ ПЕРЕВОРОТИЛОСЬ У НЕГО В ГЛАЗАХ; ОН ЧУВСТВОВАЛ, ЧТО ЕДВА МОГ СТОЯТЬ; НО
РЕШИЛСЯ ВО ЧТО БЫ ТО НИ СТАЛО ОЖИДАТЬ ЕГО ВОЗВРАЩЕНИЯ В КАРЕТУ, ВЕСЬ ДРОЖА, КАК В
ЛИХОРАДКЕ. ЧРЕЗ ДВЕ МИНУТЫ НОС ДЕЙСТВИТЕЛЬНО ВЫШЕЛ. ОН БЫЛ В МУНДИРЕ, ШИТОМ ЗОЛОТОМ,
С БОЛЬШИМ СТОЯЧИМ
ВОРОТНИКОМ; НА НЕМ БЫЛИ ЗАМШЕВЫЕ ПАНТАЛОНЫ; ПРИ БОКУ ШПАГА. ПО ШЛЯПЕ С ПЛЮМАЖЕМ
МОЖНО БЫЛО ЗАКЛЮЧИТЬ, ЧТО ОН СЧИТАЕТСЯ В РАНГЕ СТАТСКОГО СОВЕТНИКА. ПО ВСЕМУ ЗАМЕТНО
БЫЛО, ЧТО ОН ЕХАЛ КУДА-НИБУДЬ С ВИЗИТОМ. ОН ПОГЛЯДЕЛ НА ОБЕ СТОРОНЫ, ЗАКРИЧАЛ
КУЧЕРУ: "ПОДАВАЙ!" - СЕЛ И УЕХАЛ. БЕДНЫЙ КОВАЛЕВ ЧУТЬ НЕ СОШЕЛ С УМА. ОН НЕ ЗНАЛ,
КАК И ПОДУМАТЬ О ТАКОМ СТРАННОМ ПРОИСШЕСТВИИ. КАК ЖЕ МОЖНО, В САМОМ ДЕЛЕ, ЧТОБЫ
НОС, КОТОРЫЙ ЕЩЕ ВЧЕРА БЫЛ У НЕГО НА ЛИЦЕ, НЕ МОГ ЕЗДИТЬ И ХОДИТЬ, - БЫЛ В МУНДИРЕ!
ОН ПОБЕЖАЛ ЗА КАРЕТОЮ, КОТОРАЯ, К СЧАСТИЮ, ПРОЕХАЛА НЕДАЛЕКО И ОСТАНОВИЛАСЬ ПЕРЕД
КАЗАНСКИМ СОБОРОМ.
```

Рисунок 3 – Расшифровка текста по заданному ключу

5 Выводы

В результате работы были приобретены умения:

а) нахождения длины ключа сообщения, зашифрованного шифром Виженера;

б) нахождения ключа при известной длине;

в) дешифровки шифра Виженера по известному ключу;

г) анализа криптостойкости системы и использования ее уязвимостей.