ОТЧЕТ К ЛАБОРАТОРНОЙ РАБОТЕ № 2

Лабораторная работа № 2

Криптоанализ аффинного шифра

Вариант №5

Ф. И. О. студента: Гюнтер Тимофей Вячеславович

Группа: ФИТ-221

Проверил: Дата:

Основные сведения

Формула для зашифрования текста: Формула для расшифрования текста:

Результаты

ШИФР-ТЕКСТ (ШТ):

Результаты частотного анализа ШТ:

буква	а	б	в	г	д	е/ё	ж	з	и	й
частота	0	0.021	0.014	0.034	0.021	0	0.027	0.062	0.116	0
буква	к	л	м	н	о	п	р	с	т	у
частота	0.021	0.082	0.014	0.041	0.014	0	0.054	0.014	0.027	0
буква	ф	х	ц	ч	ш	щ	ъ	ы	ь	э
частота	0.021	0.027	0.021	0	0.014	0.11	0.021	0.082	0	0.068
буква	ю	я								
частота	0.048	0.027								

Наиболее часто встречающиеся символы:

и, щ

Система уравнения верного ключа:

$$\begin{cases} 5 \cdot a + b = 8 \ (mod\ 32) \\ 14 \cdot a + b = 25 \ (mod\ 32) \end{cases}$$

Решения систем уравнений: a = 9, b = 27

ВЕРНЫЙ КЛЮЧ: (9, 27)

РАСШИФРОВАННЫЙ ТЕКСТ (ОТ):

тамкоролевичмимоходомпленяетгрозногоцарятамвоблакахпереднародомчерез лесачерезморяколдуннесетбогатырявтемницетамцаревнатужитабурыйволкейв ернослужит

Код программы

```
def euclid_extended(m, a):
    if a == 0: return m, 1, 0
    d, x, y = euclid_extended(a, m%a)
    x, y = y, x - m//a*y
    return d, x, y


def inverse_element_interface():
     user_input = input("Введите элемент и модуль (через пробел), в кольце которого желаете найти обратный элемент: \n")
    a, m = map(int, user_input.split())
    d, _, inverse_a = euclid_extended(m, a%m)
    inverse_a %= m
    if d != 1:
        print("Приносим свои извинения обратного элемента не существует...")
    print(f'Обратный элемент найден: {inverse_a}\n')
# inverse_element_interface()
```

```
def comparison_solver(a, b, m):
    gcd, _, y = euclid_extended(m, a )

    if gcd != 1 and b % gcd == 0:
        # print("Обратного элемента не существует!")
        # print(f"Решение имеет в точности {gcd} решений!")
        b //= gcd
        a //= gcd
        new_m = m // gcd
        new_gcd, _, y = euclid_extended(new_m, a % new_m)
        inverse_a = y

        pair_ans = []
        for k in range(gcd):
            # print(new_betta * inverse_a, k)
            a = (b * inverse_a % new_m) + new_m*k
            # b = (betta_1 - alpha_1*a) % m
            pair_ans.append(a)
        return pair_ans
    if gcd != 1 and b % gcd != 0:
        # print("Обратного элемента не существует!")
        # print("Сравнение неразрешимо!")
```

```
        return None

    # print("Решение единственно!")
    inverse_a = y
    a = b * inverse_a % m
    # b = (betta_1 - alpha_1*a) % m
    return a

def comparison_solver_interface():
    user_input = input("Введите a, b и m (через пробел): \n")
    a, b, m = map(int, user_input.split())
    answer = comparison_solver(a, b, m)
    if not answer:
        print("Приносим свои извинения решения не существует...")

    print(f'Решение найдено: {answer}\n')

def comparison_system_solver_interface():
    user_input = input("Введите a, b, c, d и m (через пробел): \n")
    a, b, c, d, m = map(int, user_input.split())

    answer = comparison_system_solver(a, b, c, d, m)
    if not answer:
```

```
        print("Приносим свои извинения решения не существует...")


    print(f'Решение найдено: {answer}\n')


# print(comparison_solver(9, 6, 12))


def comparison_system_solver(alpha, betta, alpha_1, betta_1, m=32):
    new_alpha = (alpha_1 - alpha) % m
    new_betta = (betta_1 - betta) % m
    gcd, _, y = euclid_extended(m, new_alpha )

    if gcd != 1 and new_betta % gcd == 0:
        # print("Обратного элемента не существует!")
        # print(f"Решение имеет в точности {gcd} решений!")
        new_betta //= gcd
        new_alpha //= gcd
        new_m = m // gcd
        new_gcd, _, y = euclid_extended(new_m, new_alpha % new_m)
        inverse_a = y

        pair_ans = []
        for k in range(gcd):
            # print(new_betta * inverse_a, k)
```

```
            a = (new_betta * inverse_a % new_m) + new_m*k
            b = (betta_1 - alpha_1*a) % m
            pair_ans.append((a,b))
        return pair_ans
    if gcd != 1 and new_betta % gcd != 0:
        # print("Обратного элемента не существует!")
        # print("Сравнение неразрешимо!")
        return None

    # print("Решение единственно!")
    inverse_a = y
    a = new_betta * inverse_a % m
    b = (betta_1 - alpha_1*a) % m

    return a, b

print('------------------------------------------------------------')
from lab1 import Caesar, Encoding

# from menu import Menu

preprocessing = lambda x: Caesar.preprocessing(x)
```

```
def create_frequent_dict(message):
    preproc_message = preprocessing(message)
    n = len(message)
    alphabet = set(preproc_message)
    freq_dict = {letter:0 for letter in alphabet}
    for letter in preproc_message:
        freq_dict[letter] += preproc_message.count(letter)/n

    return freq_dict


def decrypt(message):
    encoding = Encoding(message)

    preproc_message = preprocessing(message)

    frequent_dict = create_frequent_dict(message)
    m = 32

    # two assumptions: и (5), о (14)
    # let's say that again but in terms of comparisons system:
    # zero assumption
```

```
def find_two_most_frequent_letters(frequent_dict):
    copy_dict = {key: value for key, value in frequent_dict.items()}

    first_max = max(copy_dict, key=copy_dict.get)

    copy_dict.pop(first_max)
    second_max = max(copy_dict, key=copy_dict.get)
    first_max_code = encoding.letters_to_code(first_max)[0]
    second_max_code = encoding.letters_to_code(second_max)[0]

    return first_max_code, second_max_code

first_max_code, second_max_code = find_two_most_frequent_letters(frequent_dict)

def alpha_guesser(alpha, alpha_1):
    betta, betta_1 = first_max_code, second_max_code

    solution = comparison_system_solver(alpha, betta, alpha_1, betta_1)
    if solution:
        a, b = solution
    else:
        # print("Ищите другие буквы!")
```

```
        return 0
    d, _, inverse_a = euclid_extended(32, a)
    assert d == 1
    coded_message = encoding.letters_to_code(preproc_message)
    decrypted_message = [0 for _ in preproc_message]
    answer = [el for el in preproc_message]
    for index in range(len(coded_message)):
        decrypted_message[index] = inverse_a*(coded_message[index] - b ) % m
       answer[index] = encoding.code_to_letters(index, decrypted_message[index])

    return ''.join(answer)

from itertools import product

frequency_table = {
    'о': 0.090, 'е': 0.072, 'а': 0.062,
    'и': 0.062, 'т': 0.053, 'н': 0.053, 'с': 0.045,
    'р': 0.040, 'в': 0.038, 'л': 0.035, 'к': 0.028,
    'м': 0.026, 'д': 0.025, 'п': 0.023, 'у': 0.021,
    'я': 0.018, 'ы': 0.016, 'з': 0.016, 'ъ': 0.014,
    'б': 0.014, 'г': 0.013, 'ч': 0.012, 'й': 0.010,
    'ч': 0.009, 'ж': 0.007, 'ю': 0.006, 'ш': 0.006,
    'ц': 0.004, 'щ': 0.003, 'э': 0.003, 'ф': 0.002,
```

```
    }


                # alpha, alpha_1 = encoding.letters_to_code('e')[0], encoding.letters_to_code("и")[0]
    # betta, betta_1 = first_max_code, second_max_code
    # result = alpha_guesser(alpha, alpha_1)
    # print(result)


    def write_to_file(file, letter, letter_1, result):
        file.write(f'key: ({letter}, {letter_1}); text: {result}\n')
        file.write('--------------------------------------------------\n')


    c = 0
    answer = ''
    # !! ANSWER: e, o
    file = open("files/lab2_decryption_tries.txt", 'w', encoding='utf-8')
    for letter, letter_1 in product(frequency_table.keys(), repeat=2):
        # print(letter, letter_1)
        c += 1
                        alpha, alpha_1 = encoding.letters_to_code(letter)[0], encoding.letters_to_code(letter_1)[0]
        betta, betta_1 = first_max_code, second_max_code
        result = alpha_guesser(alpha, alpha_1)
```

```
        if result != 0:
            print(f'{c}: {letter}, {letter_1}, {result}')
            write_to_file(file, letter, letter_1, result)

            answer = input("Вывод имеет смысл? (да/нет): \n")
            print('-----------------------------------')
        if answer.lower() == "да": break

        if letter != letter_1:
                        alpha, alpha_1 = encoding.letters_to_code(letter_1)[0], encoding.letters_to_code(letter)[0]
            # betta, betta_1 = first_max_code, second_max_code
            result = alpha_guesser(alpha, alpha_1)

            if result != 0:
                print(f'{c}: {letter_1}, {letter}, {result}')
                write_to_file(file, letter_1, letter, result)

                answer = input("Вывод имеет смысл? (да/нет): \n")
                print('-----------------------------------')
            if c == 100 or answer.lower() == "да": break
```

```
    file.close()


def variant_case_decryption():
# message = input("Введите фразу: ")
                                                    message                                =
'эызхщлщюингкзгзщшщящзвюиртиэцлщърщцщбылтэызнщдюыхышви\
лиярылщящзкилиъюифыкилиъзщлтхщюяжррифиэдщцыэолтнэизргбиэыз\
былинрыэжсгэыджломнщюхимнилрщфюжсгэ'
    decrypt(message)


def input_message_decryption():
    message = input("Введите сообщения для расшифрования: ")
    decrypt(message)


# Menu()
```