

Правительство Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»
(НИУ ВШЭ)

Московский институт электроники и математики им. А.Н.Тихонова

ОТЧЁТ
О ПРАКТИЧЕСКОЙ РАБОТЕ №1
по дисциплине «Основы криптографии и стеганографии»
ПОДСТАНОВОЧНЫЕ ШИФРЫ

Студент БИБ252

Мельников В.К.
«___» 2026 г.

Руководитель
Заведующий кафедрой информационной
безопасности киберфизических систем
канд. техн. наук, доцент

О.О. Евсютин
«___» 2026 г.

СОДЕРЖАНИЕ

1 Задание на практическую работу	3
2 Краткая теоретическая часть	4
2.1 Описание шифров	4
2.2 Методы криptoанализа шифров	5
3 Примеры шифрования	7
4 Программная реализация шифров	10
5 Примеры криptoанализа	14
6 Выводы о проделанной работе	15
7 Список использованных источников	16
Приложение А Формулы	17
Приложение А.1 Таблицы	17

1 Задание на практическую работу

Цель данной работы заключается в освоении навыков программной реализации и криптоанализа простых подстановочных шифров.

В рамках практической работы необходимо выполнить следующее:

- Написать программную реализацию аффинного шифра, аффинного рекуррентного шифра, а также же шифра простой подстановки.
- Изучить методы криптоанализа моноалфавитных шифров, а так же провести криптоанализ данных шифров.
- Подготовить отчёт о проделанной работе.

2 Краткая теоретическая часть

2.1 Описание шифров

Шифр простой замены — простейший пример подстановочного шифра, в котором каждому символу открытого текста соответствует один символ зашифрованного текста. Сообщение рассматривается как последовательность символов алфавита A , мощности m , а ключ k формируется как произвольная перестановка алфавита.

$$k = \begin{pmatrix} x_1 & x_2 & \dots & x_m \\ y_1 & y_2 & \dots & y_m \end{pmatrix} \quad (1)$$

В данном случае x_1, x_2, \dots, x_m — это символы открытого текста, а y_1, y_2, \dots, y_m — символы зашифрованного текста.

Зашифрование осуществляется заменой каждого символа открытого текста на соответствующий ему символ зашифрованного текста и может быть записано как

$$E_k(x) = (k(x_1), \dots, k(x_m)) \quad (2)$$

Расшифрование осуществляется обратной заменой и может быть записано как

$$D_k(y) = (k^{-1}(y_1), \dots, k^{-1}(y_m)), \quad (3)$$

$$k^{-1} = \begin{pmatrix} y_1 & y_2 & \dots & y_m \\ x_1 & x_2 & \dots & x_m \end{pmatrix} \quad (4)$$

Аффинный шифр — это частный случай шифра замены, который использует математическую функцию для преобразования символов, называемую аффинное преобразование.

Данный шифр реализует замену символов открытого текста с использованием операций в арифметике остатков. Символы алфавита A мощностью m представляются элементами множества Z_m , а ключ k состоит из двух чисел a и b , принадлежащих Z_m , таких что a и m взаимно простые.

Ключ может быть представлен как перестановка (1), в котором

$$y_i = (a \cdot x_i + b) \pmod{m}, i \in \{1, 2, \dots, m\} \quad (5)$$

Расшифрование осуществляется с помощью обратного аффинного преобразования, которое может быть записано как

$$x_i = (a^{-1} \cdot (y_i - b)) \pmod{m}, i \in \{1, 2, \dots, m\} \quad (6)$$

Аффинный рекуррентный шифр — это усиление аффинного шифра, который использует рекуррентные функции для генерации ключей.

Алгоритм шифрования основан на вычислении нового ключа для каждого символа открытого текста на основе предыдущих символов и ключей. Для этого необходимо задать

два ключевые пары: $k_1 = (a_1, b_1)$ и $k_2 = (a_2, b_2)$, а последующие ключевые пары будут вычисляться по формуле:

$$k_i = ((a_{i-1} \cdot a_{i-2}) \pmod{m}, (b_{i-1} + b_{i-2}) \pmod{m}), i \in \{3, 4, \dots, m\} \quad (7)$$

Алгоритм зашифрования, расшифрования и формулы для вычисления ключей аналогично аффинному шифру, но с использованием рекуррентных ключей.

2.2 Методы криptoанализа шифров

Криptoанализ шифра простой замены. Для каждого алфавита мощности m может быть подобрано $m!$ различных ключей, что делает полный перебор ключей неэффективным и нецелесообразным. Однако, шифр простой замены является моноалфавитным шифром, что означает, что каждый символ открытого текста всегда заменяется одним и тем же символом зашифрованного текста. Это делает его уязвимым к частотному анализу, который заключается в изучении частоты появления символов в зашифрованном тексте и сопоставлении их с частотой появления символов в языке, на котором написан открытый текст. Например, в русском языке буква «о» встречается чаще всего, следовательно, символ, который встречается чаще всего в зашифрованном тексте, вероятно, соответствует букве «о». Аналогично, можно сопоставить другие часто встречающиеся символы с наиболее частыми буквами в языке.

Криptoанализ аффинного шифра. Для каждого алфавита мощности m необходимо подобрать два числа a и b , где $\gcd(a, m) = 1$, а $b \in \{0, 1, \dots, m - 1\}$. Чтобы найти количество взаимно простых чисел можно воспользоваться функцией Эйлера $\varphi(m)$, которая определяет количество чисел от 1 до $m - 1$, которые являются взаимно простыми с m .

$$\varphi(n) = n \prod_{p|n} \left(1 - \frac{1}{p}\right), \quad n > 1 \quad (8)$$

где p — простое число и пробегает все значения, участвующие в разложении n на простые множители

Так, для русского алфавита мощности $33 = 3^1 \cdot 11^1$

$$\varphi(33) = 33 \cdot \left(1 - \frac{1}{3}\right) \cdot \left(1 - \frac{1}{11}\right) = 20 \quad (9)$$

В таком случае количество всех возможных ключей равно $33 \cdot 20 = 660$. Такое количество различных ключей позволяет сделать полный перебор, все современные компьютеры способны выполнить его меньше чем за секунду. Однако, существуют и другие способы криptoанализа. Аффинный шифр является моноалфавитным шифром, что делает его уязвимым к частотному анализу, аналогично шифру простой замены.

Криptoанализ аффинного рекуррентного шифра. Аффинный рекуррентный шифр является более сложным, чем аффинный шифр, из-за использования рекуррентных ключей, что делает его полиалфавитным шифром, а следовательно, более устойчивым к частотному

анализу. Так, разные символы открытого текста могут быть зашифрованы одними и теми же символами зашифрованного текста и наоборот, что затрудняет использование частотного анализа.

Разберем сложность полного перебора. Для этого нужно верно подобрать четыре числа: a_1 , b_1 , a_2 и b_2 . Числа a_1 и a_2 должны быть взаимно простыми с мощностью алфавита m , а числа b_1 и b_2 могут принимать любые значения от 0 до $m - 1$. Таким образом, количество возможных ключей равно $\varphi(m)^2 \cdot m^2$. Для русского алфавита это будет равно $20^2 \cdot 33^2 = 435600$. Такое количество ключей делает полный перебор сложным, но не невозможным для современных компьютеров.

3 Примеры шифрования

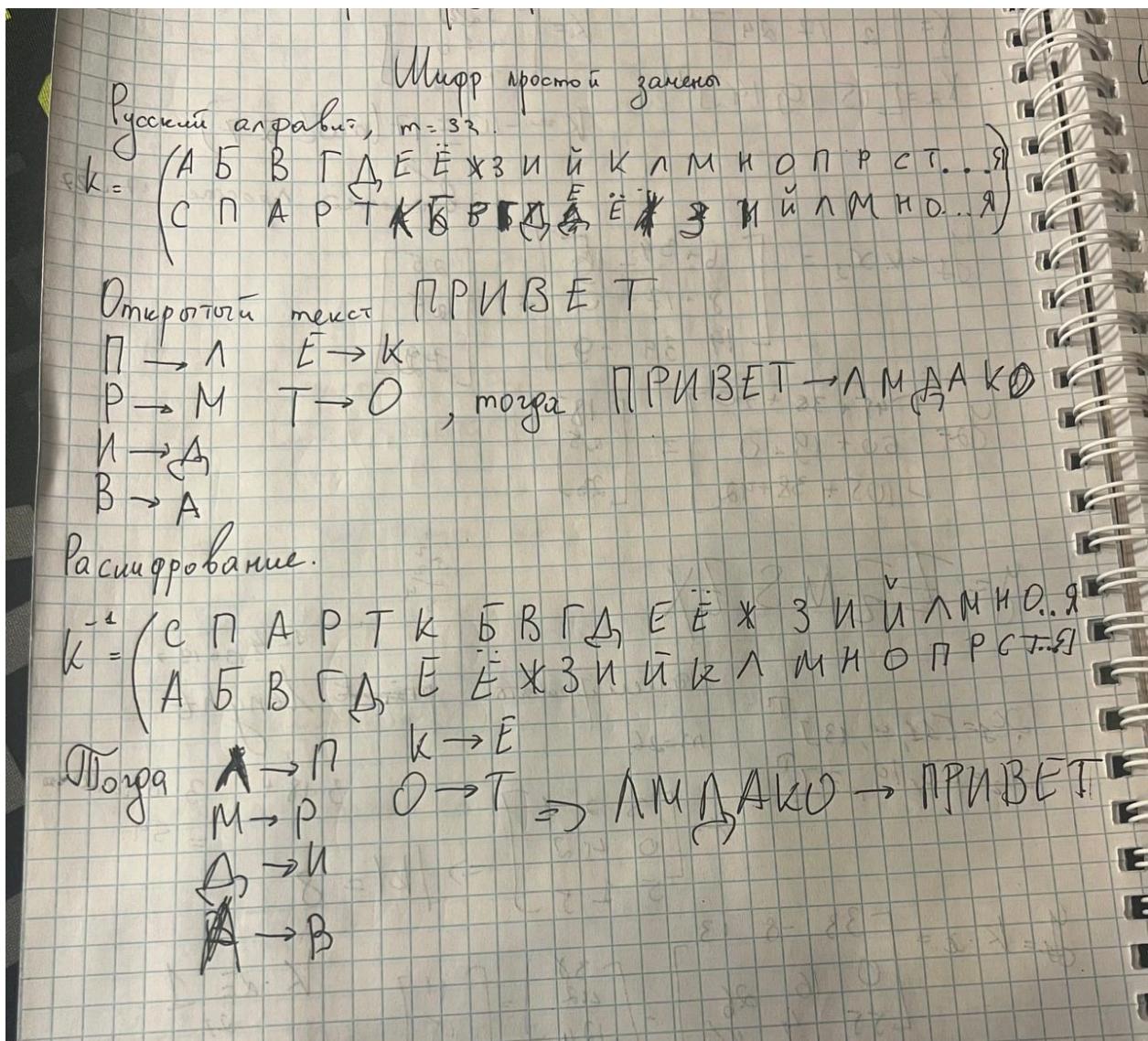


Рисунок 1 — Пример шифрования и расшифрования текста с помощью шифра простой замены

Русский алфавит, $m = 33$

$$d = (4, 7), \text{ НОД}(4, 33) = 1$$

Формула $y_i = (a \cdot x_i + b) \bmod 33$
Шифруем ПРИВЕТ

$$a = 16$$

$$b = 7$$

$$x_1 = 9$$

$$x_2 = 2$$

$$x_3 = 5$$

$$x_4 = 10$$

$$y_1 = (4 \cdot 16 + 7) \bmod 33 = 71 \bmod 33 = 5$$

$$y_2 = (4 \cdot 17 + 7) \bmod 33 = 75 \bmod 33 = 9.$$

$$y_3 = (4 \cdot 8 + 7) \bmod 33 = 43 \bmod 33 = 10$$

$$y_4 = (2 \cdot 9 + 7) \bmod 33 = 15$$

$$y_5 = (5 \cdot 2 + 7) \bmod 33 = 27.$$

$$y_6 = (13 \cdot 4 + 7) \bmod 33 = 83 \bmod 33 = 17$$

$$5 = e; 9 = u; 10 = y; 0 = i; 15 = r; 27 = t; 17 = p.$$

ПРИВЕТ. ЕНИЙ ОГРН

Расшифрование:

$$\text{Формула } x_i = (a^{-1} \cdot (y_i - b)) \bmod 33$$

$$a^{-1} = 25$$

по расч. алг. Евклида

$$\begin{array}{r} 33 \\ 25 \overline{) 8} \\ 25 \overline{) 3} \\ 25 \overline{) 3} \\ 25 \overline{) 3} \end{array}$$

$$1 \equiv -32 \bmod 33$$

$$-8 + 33 = 25.$$

$$x_1 = (25 \cdot (5 - 7)) \bmod 33 = -50 \bmod 33 = 16 = n$$

$$x_2 = (25 \cdot (9 - 7)) \bmod 33 = 50 \bmod 33 = 17 = p$$

$$x_3 = (25 \cdot (10 - 7)) \bmod 33 = 75 \bmod 33 = 9 = u$$

$$x_4 = (25 \cdot (15 - 7)) \bmod 33 = 200 \bmod 33 = 2 = b$$

$$x_5 = (25 \cdot (27 - 7)) \bmod 33 = 500 \bmod 33 = 5 = e$$

$$x_6 = (25 \cdot (17 - 7)) \bmod 33 = 250 \bmod 33 = 17 = t$$

Расшифровка
ПРИВЕТ

Рисунок 2 — Пример шифрования и расшифрования текста с помощью аффинного шифра

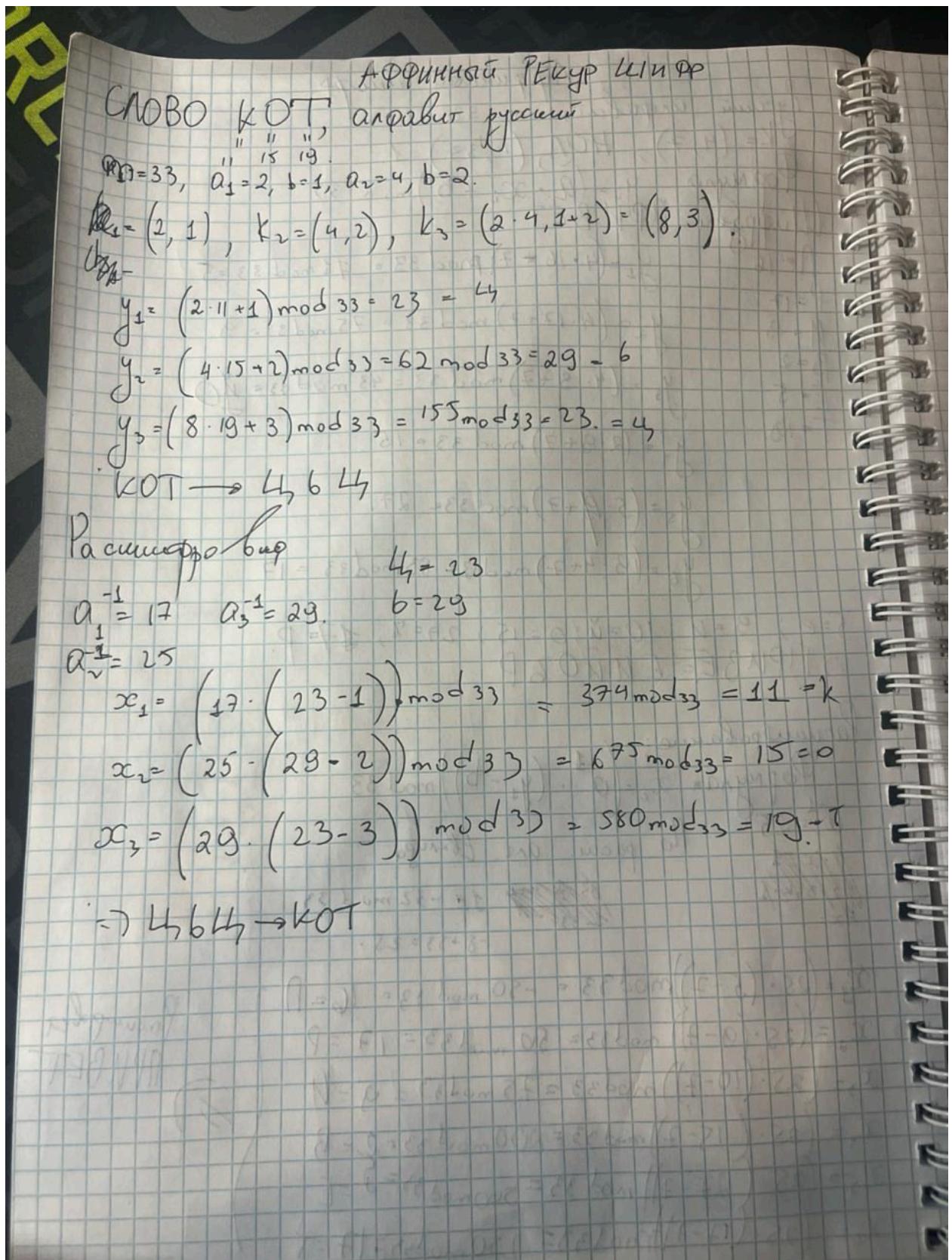


Рисунок 3 — Пример шифрования и расшифрования текста с помощью аффинного рекуррентного шифра

На этом примере (см. Рисунок 3) видно, что только аффинный рекуррентный шифр обеспечивает шифрование разных символов открытого текста в одни и те же символы зашифрованного текста, что делает его более устойчивым к частотному анализу.

4 Программная реализация шифров

Программная реализация шифров была выполнена на языке Python. Реализованы функции записи шифротекста или расшифрованного текста в отдельный файл (см. Рисунок 4), а так же исключение для ошибки записи файла. Написана функция вычисления НОД для двух чисел (см. Рисунок 5) для проверки валидности ключа.

Реализовано 6 функций для расшифровки и шифровки текста тремя способами: простой замены, аффинного шифра и аффинного рекуррентного шифра. Каждая функция принимает на вход открытый текст, ключ и алфавит, а возвращает зашифрованный текст или расшифрованный текст в зависимости от функции. Учтены некоторые исключения, такие как несоответствие мощности алфавита и некорректный ключ (см. Рисунки 6,7,8,9).

Пользователь из консоли может выбрать способ шифрования, ввести открытый текст, ключ и алфавит, а также указать имя файла для сохранения результата. Результат будет записан в указанный файл, в случае если файл с таким именем существует, он будет перезаписан (см. Рисунки 10, 11).

```
Explain Code | Generate Tests | Ask Sourcery | You, 5 minutes ago | 1 author (You)
class SaveFileException(Exception):
    """Ошибка сохранения файла"""
    pass

Explain Code | Generate Tests | Generate Docstrings | Ask Sourcery
def save_in_file(text):
    ans = input("Сохранить содержимое вывода в отдельный файл? [y/n]: ")
    if ans == 'y':
        filename = input("Введите название файла: ")
        if not filename.endswith(".txt"):
            filename += ".txt"
        try:
            with open(filename, "w") as f:
                f.write(text)
        except:
            raise SaveFileException("Ошибка записи файла")
```

Рисунок 4 — Реализация функции записи в файл и ее исключения

```
Explain Code | Generate Tests | Ask Sourcery
@staticmethod
def gcd(a, b):
    """Алгоритм Евклида"""
    while b != 0:
        a, b = b, a % b
    return a
```

Рисунок 5 — Реализация функции вычисления НОД

```

Explain Code | Generate Tests | Generate Docstrings | Ask Sourcery
def inverse(self, a):
    return pow(a, -1, self.m)

Explain Code | Generate Tests | Generate Docstrings | Ask Sourcery
def simple_substitution_encrypt(self, text, key):
    if len(key) != self.m:
        raise KeyValidityError(f'Длина ключа ({key}) не соответствует мощности алфавита ({self.m})')
    if sorted(key) != sorted(self.alphabet):
        raise CharacterMismatch("Один или более символов не указаны в алфавите или наоборот")
    result = ''
    for char in text:
        if char in [" ", ".", ","]:
            result += char
            continue
        index = self.alphabet.find(char)
        result += key[index]
    return result

Explain Code | Generate Tests | Generate Docstrings | Ask Sourcery
def simple_substitution_decrypt(self, text, key):
    if len(key) != self.m:
        raise KeyValidityError(f'Длина ключа ({key}) не соответствует мощности алфавита ({self.m})')
    if sorted(key) != sorted(self.alphabet):
        raise CharacterMismatch("Один или более символов не указаны в алфавите или наоборот")
    result = ''
    for char in text:
        if char in [" ", ".", ","]:
            result += char
            continue
        index = key.find(char)
        result += self.alphabet[index]
    return result

```

Рисунок 6 — Реализация функций шифрования и расшифрования текста с помощью шифра простой замены

```

Explain Code | Generate Tests | Generate Docstrings | Ask Sourcery
def affine_cipher_encrypt(self, text, key_a, key_b):
    key_a = int(key_a)
    key_b = int(key_b)
    if self.gcd(key_a, self.m) != 1:
        raise KeyValidityError("Первое число ключа и мощность алфавита не взаимно просты")
    result = ''
    for char in text:
        if char in [" ", ".", ","]:
            result += char
            continue
        x = self.alphabet.find(char)
        y = key_a * x + key_b
        result += self.alphabet[y % self.m]
    return result

Explain Code | Generate Tests | Generate Docstrings | Ask Sourcery
def affine_cipher_decrypt(self, text, key_a, key_b):
    key_a = int(key_a)
    key_b = int(key_b)
    if self.gcd(key_a, self.m) != 1:
        raise KeyValidityError("Первое число ключа и мощность алфавита не взаимно просты")
    result = ''
    for char in text:
        if char in [" ", ".", ","]:
            result += char
            continue
        y = self.alphabet.find(char)
        x = (y - key_b) * self.inverse(key_a)
        result += self.alphabet[x % self.m]
    return result

```

Рисунок 7 — Реализация функций шифрования и расшифрования текста с помощью аффинного шифра

```

Explain Code | Generate Tests | Generate Docstrings | Ask Sourcery
def affine_recurrent_cipher_encrypt(self, text, key_1a, key_1b, key_2a, key_2b):
    key_1a, key_1b, key_2a, key_2b = int(key_1a), int(key_1b), int(key_2a), int(key_2b)

    if self.gcd(key_1a, self.m) != 1 or self.gcd(key_2a, self.m) != 1:
        raise KeyValidityError("Первое число ключа и мощность алфавита не взаимно просты")
    result = ''

    x = self.alphabet.find(text[0])
    y = key_1a * x + key_1b
    result += self.alphabet[y % self.m]
    x = self.alphabet.find(text[1])
    y = key_2a * x + key_2b
    result += self.alphabet[y % self.m]

    prev2_a, prev2_b = key_1a, key_1b #i - 2 элемент
    prev1_a, prev1_b = key_2a, key_2b #i - 1 элемент

    for i in range(2, len(text)):
        if text[i] in [" ", ".", ","]:
            result += text[i]
            continue
        current_a = (prev1_a * prev2_a) % self.m
        current_b = (prev1_b + prev2_b) % self.m
        x = self.alphabet.find(text[i])
        y = current_a * x + current_b
        result += self.alphabet[y % self.m]
        prev2_a, prev2_b = prev1_a, prev1_b
        prev1_a, prev1_b = current_a, current_b

    return result

```

Рисунок 8 — Реализация функции шифрования текста с помощью аффинного рекуррентного шифра

```

Explain Code | Generate Tests | Generate Docstrings | Ask Sourcery
def affine_recurrent_cipher_decrypt(self, text, key_1a, key_1b, key_2a, key_2b):
    key_1a, key_1b, key_2a, key_2b = int(key_1a), int(key_1b), int(key_2a), int(key_2b)

    if self.gcd(key_1a, self.m) != 1 or self.gcd(key_2a, self.m) != 1:
        raise KeyValidityError("Первое число ключа и мощность алфавита не взаимно просты")
    result = ''
    result += self.alphabet[((self.alphabet.find(text[0]) - key_1b) * self.inverse(key_1a)) % self.m]
    result += self.alphabet[((self.alphabet.find(text[1]) - key_2b) * self.inverse(key_2a)) % self.m]

    prev2_a, prev2_b = key_1a, key_1b #i - 2 элемент
    prev1_a, prev1_b = key_2a, key_2b #i - 1 элемент

    for i in range(2, len(text)):
        if text[i] in [" ", ".", ","]:
            result += text[i]
            continue

        current_a = (prev1_a * prev2_a) % self.m
        current_b = (prev1_b + prev2_b) % self.m
        y = self.alphabet.find(text[i])
        x = (y - current_b) * self.inverse(current_a)
        result += self.alphabet[x % self.m]
        prev2_a, prev2_b = prev1_a, prev1_b
        prev1_a, prev1_b = current_a, current_b

    return result

```

Рисунок 9 — Реализация функции расшифрования текста с помощью аффинного рекуррентного шифра

```

if __name__ == "__main__":
    encrypt_or_decrypt = input("Вы хотите зашифровать или расшифровать? 1 - зашифровка, 2 - расшифровка \n")
    res = ''
    if encrypt_or_decrypt == "1":
        encrypt_way = input("Каким способом вы хотите зашифровать текст? \n 1 - шифр простой замены \n 2 - аффинный шифр \n 3 - аффинный рекурентный шифр \n")
        if encrypt_way == "1":
            user_alphabet = input("Введите алфавит: ")
            app = SimpleCiphers(user_alphabet)
            text = input("Текст, который вы хотите зашифровать: ")
            key = input("Введите ключ: ")
            res = app.simple_substitution_encrypt(text, key)
        if encrypt_way == "2":
            user_alphabet = input("Введите алфавит: ")
            app = SimpleCiphers(user_alphabet)
            text = input("Введите текст, который хотите зашифровать ")
            key_a = input("Введите первое число ключа. Оно должно быть взаимно простым с мощностью алфавита ")
            key_b = input("Введите второе число ключа ")
            res = app.affine_cipher_encrypt(text, key_a, key_b)
        if encrypt_way == "3":
            user_alphabet = input("Введите алфавит: ")
            app = SimpleCiphers(user_alphabet)
            text = input("Введите текст, который хотите зашифровать ")
            key_1a = input("Введите первое число первого ключа. Оно должно быть взаимно простым с мощностью алфавита ")
            key_1b = input("Введите второе число первого ключа ")
            key_2a = input("Введите первое число второго ключа. Оно должно быть взаимно простым с мощностью алфавита ")
            key_2b = input("Введите второе число второго ключа ")
            res = app.affine_recurrent_cipher_encrypt(text, key_1a, key_1b, key_2a, key_2b)

```

Рисунок 10 — Реализация главной функции, которая позволяет пользователю выбрать способ шифрования или расшифрования, ввести открытый текст, ключ и алфавит, а также указать имя файла для сохранения результата [часть 1]

```

elif encrypt_or_decrypt == '2':
    encrypt_way = input("Каким способом вы хотите расшифровать текст? \n 1 - шифр простой замены \n 2 - аффинный шифр \n 3 - аффинный рекурентный шифр \n")
    if encrypt_way == "1":
        user_alphabet = input("Введите алфавит: ")
        app = SimpleCiphers(user_alphabet)
        text = input("Шифртекст, который вы хотите расшифровать: ")
        key = input("Введите ключ: ")
        res = app.simple_substitution_decrypt(text, key)
    if encrypt_way == "2":
        user_alphabet = input("Введите алфавит: ")
        app = SimpleCiphers(user_alphabet)
        text = input("Шифртекст, который хотите расшифровать: ")
        key_a = input("Введите первое число ключа. Оно должно быть взаимно простым с мощностью алфавита: ")
        key_b = input("Введите второе число ключа: ")
        res = app.affine_cipher_decrypt(text, key_a, key_b)
    if encrypt_way == "3":
        user_alphabet = input("Введите алфавит: ")
        app = SimpleCiphers(user.alphabet)
        text = input("Шифртекст, который хотите расшифровать: ")
        key_1a = input("Введите первое число первого ключа. Оно должно быть взаимно простым с мощностью алфавита ")
        key_1b = input("Введите второе число первого ключа ")
        key_2a = input("Введите первое число второго ключа. Оно должно быть взаимно простым с мощностью алфавита ")
        key_2b = input("Введите второе число второго ключа ")
        res = app.affine_recurrent_cipher_decrypt(text, key_1a, key_1b, key_2a, key_2b)

if res:
    print(f'{res}')
    save_in_file(res)

```

Рисунок 11 — Реализация главной функции, которая позволяет пользователю выбрать способ шифрования или расшифрования, ввести открытый текст, ключ и алфавит, а также указать имя файла для сохранения результата [часть 2]

5 Примеры криptoанализа

Пример криptoанализа шифра простой замены. Из-за вычислительной сложности полного перебора, наиболее эффективным методом криptoанализа шифра простой замены является частотный анализ. Этот метод позволяет сопоставить наиболее часто встречающиеся символы в зашифрованном тексте с наиболее частыми буквами в языке, на котором написан открытый текст. Для этого нужно определить какие буквы в естественном языке встречаются часто, а какие нет. Можно использовать внешние источники информации, но можно и проанализировать большой объем текста на исходном языке. Для этого я взял первый том «Войны и мир» Л.Н. Толстого и проанализировал частоту появления каждой буквы.

6 Выводы о проделанной работе

Краткие выводы о проделанной работе: достоинства и недостатки исследуемых шифров, ограничения выбранных методов криптоанализа, наиболее эффективные сценарии криптоанализа.

7 Список использованных источников

ПРИЛОЖЕНИЕ А

Формулы

Чтобы оформить формулы в документе, можно использовать синтаксис `typst-math`. Пример такого использования:

$$\sum_{k=0}^n k = 1 + \dots + n = \frac{n(n+1)}{2} \quad (\text{A.1})$$

Как оформлять таблицы сказано в приложении А.1.

ПРИЛОЖЕНИЕ А.1

Таблицы

Для создания таблиц используется функция `table()`, обёрнутая в макрос `#figure` для добавления подписи. Пример показан на таблице А.1.

Таблица А.1 — Пример таблицы с данными

Заголовок 1	Заголовок 2	Заголовок 3	Заголовок 4
Проверка	Проверка	Проверка	Проверка
Проверка	Проверка	Проверка	Проверка