Федеральное государственное автономное образовательное учреждение высшего образования
«Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Лабораторная работа 1

«Основы шифрования данных»

Вариант № 10

Группа: P34102

Выполнил: Лапин А.А.

Проверил:
Рыбаков С.Д.

Санкт-Петербург
2024г.

Оглавление

Введение

Целью данной лабораторной работы является изучение основных принципов шифрования информации, знакомство с широко известными алгоритмами шифрования, приобретение навыков их программной реализации.

Глава 1 Текст задания

Необходимо реализовать программу, выполняющую шифрование и дешифрование текстовых файлов по методу Цезаря. Дополнительно требуется провести частотный анализ зашифрованного текста и использовать файл с ключевыми словами для определения исходного сдвига шифра.

Были реализованы основные функции для шифрования и дешифрования. Проведен частотный анализ, позволяющий определить вероятность сдвига шифра как для английского, так и для русского алфавитов. Также применен метод сопоставления с ключевыми словами для уточнения результатов дешифрования.

Глава 2 Структура проекта

Структура проекта представлена следующим образом:

```
Cryptography/
└── lab1.1/
    ├── src/
    │   ├── caesar_cipher.py
    │   ├── frequency_analysis.py
    │   └── main.py
    ├── keywords.txt
    └── tests/
        ├── test_caesar_cipher.py
        ├── test_frequency_analysis.py
        └── test_main.py
```

2.1 Описание основных компонентов

2.1.1 caesar_cipher.py

Реализует класс `CaesarCipher`, отвечающий за шифрование и дешифрование текста с использованием метода Цезаря. Поддерживает как английский, так и русский алфавиты.

2.1.2 frequency_analysis.py

Содержит класс `FrequencyAnalyzer`, выполняющий анализ частотности символов в тексте для определения сдвига шифра.

2.1.3 main.py

Основной исполняемый файл программы, предоставляющий интерфейс командной строки для шифрования, дешифрования и анализа текста.

2.1.4 keywords.txt

Файл, содержащий набор ключевых слов, используемых для повышения точности дешифрования методом сопоставления.

2.1.5 tests/

Директория с тестовыми скриптами для проверки корректности работы основных компонентов программы.

Глава 3 Листинги разработанной программы с комментариями

3.1 caesar_cipher.py

```
import string

class CaesarCipher:
    def __init__(self, shift):
        self.shift = shift
        # Define English and Russian alphabets
        self.english_lower = string.ascii_lowercase
        self.english_upper = string.ascii_uppercase
        self.russian_lower = 'абвгдеёжзийклмнопрстуфхцчшщъыьэюя'
        self.russian_upper = 'АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ'

    def encrypt(self, text):
        return self._shift_text(text, self.shift)

    def decrypt(self, text):
        return self._shift_text(text, -self.shift)

    def _shift_text(self, text, shift):
        shifted_text = []
        for char in text:
            shifted_char = char
            if char in self.english_lower:
                idx = self.english_lower.find(char)
                shifted_idx = (idx + shift) % 26
                shifted_char = self.english_lower[shifted_idx]
            elif char in self.english_upper:
                idx = self.english_upper.find(char)
                shifted_idx = (idx + shift) % 26
                shifted_char = self.english_upper[shifted_idx]
            elif char in self.russian_lower:
                idx = self.russian_lower.find(char)
                shifted_idx = (idx + shift) % len(self.russian_lower)
                shifted_char = self.russian_lower[shifted_idx]
            elif char in self.russian_upper:
                idx = self.russian_upper.find(char)
                shifted_idx = (idx + shift) % len(self.russian_upper)
                shifted_char = self.russian_upper[shifted_idx]
            shifted_text.append(shifted_char)
        return ''.join(shifted_text)
```

3.2 frequency_analysis.py

```
import string
from collections import Counter
```

```
from typing import Optional

class FrequencyAnalyzer:
    def __init__(self, keywords_file: Optional[str] = None):
        self.keywords = self._load_keywords(keywords_file)
        # Define English and Russian alphabets
        self.english_lower = string.ascii_lowercase
        self.russian_lower = 'абвгдеёжзийклмнопрстуфхцчшщъыьэюя'
        self.alphabet = self.english_lower + self.russian_lower

        # Define expected frequency distributions for English and Russian
        self.english_freq = {
            'a': 8.167, 'b': 1.492, 'c': 2.782, 'd': 4.253,
            'e': 12.702, 'f': 2.228, 'g': 2.015, 'h': 6.094,
            'i': 6.966, 'j': 0.153, 'k': 0.772, 'l': 4.025,
            'm': 2.406, 'n': 6.749, 'o': 7.507, 'p': 1.929,
            'q': 0.095, 'r': 5.987, 's': 6.327, 't': 9.056,
            'u': 2.758, 'v': 0.978, 'w': 2.360, 'x': 0.150,
            'y': 1.974, 'z': 0.074
        }

        self.russian_freq = {
            'а': 8.01, 'б': 1.59, 'в': 4.54, 'г': 1.70,
            'д': 2.98, 'е': 8.45, 'ё': 0.04, 'ж': 0.94,
            'з': 1.65, 'и': 7.35, 'й': 1.21, 'к': 3.49,
            'л': 4.40, 'м': 3.21, 'н': 6.70, 'о': 10.97,
            'п': 2.81, 'р': 4.73, 'с': 5.47, 'т': 6.26,
            'у': 2.62, 'ф': 0.26, 'х': 0.97, 'ц': 0.48,
            'ч': 1.44, 'ш': 0.73, 'щ': 0.36, 'ъ': 0.04,
            'ы': 1.90, 'ь': 1.74, 'э': 0.32, 'ю': 0.64,
            'я': 2.01
        }

    def _load_keywords(self, filepath: Optional[str] = None):
        if filepath is None:
            return []
        with open(filepath, 'r', encoding='utf-8') as file:
            return [line.strip().lower() for line in file]

    def analyze(self, text):
        text = text.lower()
        filtered_text = [char for char in text if char in self.alphabet]
        freq = Counter(filtered_text)
        return freq

    def matches_keywords(self, text):
        text = text.lower()
        matches = 0
        for keyword in self.keywords:
            if keyword in text:
                matches += 1
        return matches

    def detect_shift(self, text):
        freq = self.analyze(text)

        # Normalize frequencies
```

```
        total = sum(freq.values())
        freq_percent = {char: (count / total) * 100 for char, count in freq.items()}

        # Detect English shift
        english_shift_scores = []
        for shift in range(26):
            score = 0.0
            for char in self.english_lower:
                shifted_char = self.english_lower[(self.english_lower.index(char) -
                    shift) % 26]
                score += abs(freq_percent.get(char, 0) - self.english_freq[
                    shifted_char])
            english_shift_scores.append((shift, score))
        # Select shift with minimum score
        english_shift = min(english_shift_scores, key=lambda x: x[1])[0]

        # Detect Russian shift
        russian_shift_scores = []
        for shift in range(33):
            score = 0.0
            for char in self.russian_lower:
                shifted_char = self.russian_lower[(self.russian_lower.index(char) -
                    shift) % 33]
                score += abs(freq_percent.get(char, 0) - self.russian_freq[
                    shifted_char])
            russian_shift_scores.append((shift, score))
        # Select shift with minimum score
        russian_shift = min(russian_shift_scores, key=lambda x: x[1])[0]

        return english_shift, russian_shift
```

3.3 main.py

```
import argparse
from collections import Counter
from caesar_cipher import CaesarCipher
from frequency_analysis import FrequencyAnalyzer
from tabulate import tabulate


def read_file(filepath):
    with open(filepath, 'r', encoding='utf-8') as file:
        return file.read()


def write_file(filepath, content):
    with open(filepath, 'w', encoding='utf-8') as file:
        file.write(content)


def encrypt_file(input_path, output_path, shift):
    cipher = CaesarCipher(shift)
    plaintext = read_file(input_path)
    ciphertext = cipher.encrypt(plaintext)
    write_file(output_path, ciphertext)
    print(f"Файл зашифрован и сохранён как {output_path}")
```

```
24
25
26  def decrypt_file(input_path, output_path, shift):
27      cipher = CaesarCipher(shift)
28      ciphertext = read_file(input_path)
29      plaintext = cipher.decrypt(ciphertext)
30      write_file(output_path, plaintext)
31      print(f"Файл дешифрован и сохранён как {output_path}")
32
33
34  def print_frequency_table(freq: Counter):
35      print("Частотный анализ зашифрованного текста:")
36      # Normalize frequencies
37      total = sum(freq.values())
38      freq_percent = {char: round((count / total) * 100, 3) for char, count in freq.
            items()}
39      sorted_freq = freq.most_common()
40      print(tabulate([(char, count, freq_percent[char]) for char, count in sorted_freq
            ],
41            headers=['Символ', 'Количество', 'Частота'], tablefmt='grid'))
42
43
44  def frequency_detect_shift(input_path):
45      analyzer = FrequencyAnalyzer()
46      ciphertext = read_file(input_path)
47      freq = analyzer.analyze(ciphertext)
48      print_frequency_table(freq)
49
50      # Попытка дешифровки методом частотного анализа
51      english_shift, russian_shift = analyzer.detect_shift(ciphertext)
52      print("\nПопытка дешифровки с использованием частотного анализа:")
53      print(f"Наиболее вероятный сдвиг для английского алфавита: {english_shift}")
54      print(f"Наиболее вероятный сдвиг для русского алфавита: {russian_shift}")
55      # Дешифровка с найденным сдвигом для каждого алфавита
56      cipher_english = CaesarCipher(english_shift)
57      cipher_russian = CaesarCipher(russian_shift)
58      decrypted_text_english = cipher_english.decrypt(ciphertext)
59      decrypted_text_russian = cipher_russian.decrypt(ciphertext)
60      print(f"Расшифрованный текст (английский сдвиг):\n{decrypted_text_english}")
61      print(f"Расшифрованный текст (русский сдвиг):\n{decrypted_text_russian}")
62
63
64  def frequency_analysis(input_path, keywords_file):
65      analyzer = FrequencyAnalyzer(keywords_file)
66      ciphertext = read_file(input_path)
67      freq = analyzer.analyze(ciphertext)
68      print_frequency_table(freq)
69
70      # Попытка дешифровки путем сопоставления ключевых слов
71      best_shift = None
72      max_matches = -1
73      for shift in range(100):
74          cipher = CaesarCipher(shift)
75          decrypted_text = cipher.decrypt(ciphertext)
76          matches = analyzer.matches_keywords(decrypted_text)
77          if matches > max_matches:
78              max_matches = matches
```

```
            best_shift = shift

    if best_shift is not None:
        print(f"Наиболее вероятный сдвиг для дешифровки: {best_shift}")
        print(f"Количество совпадений ключевых слов: {max_matches}")
    else:
        print("Не удалось определить сдвиг для дешифровки.")


def main():
    parser = argparse.ArgumentParser(
        description="Шифр Цезаря: шифрование, дешифровка и частотный анализ.")
    subparsers = parser.add_subparsers(dest='command', required=True)

    # Шифрование
    encrypt_parser = subparsers.add_parser('encrypt', help='Зашифровать файл')
    encrypt_parser.add_argument('input', help='Путь к входному файлу')
    encrypt_parser.add_argument('output', help='Путь к выходному файлу')
    encrypt_parser.add_argument('shift', type=int, help='Сдвиг')

    # Дешифровка
    decrypt_parser = subparsers.add_parser('decrypt', help='Дешифровать файл')
    decrypt_parser.add_argument('input', help='Путь к зашифрованному файлу')
    decrypt_parser.add_argument('output', help='Путь к выходному файлу')
    decrypt_parser.add_argument('shift', type=int, help='Сдвиг')

    # Поиск сдвига методом ключевых слов
    analyze_parser = subparsers.add_parser(
        'analyze', help='Провести частотный анализ зашифрованного файла')
    analyze_parser.add_argument('input', help='Путь к зашифрованному файлу')
    analyze_parser.add_argument(
        'keywords', help='Путь к файлу с ключевыми словами')

    # Поиск сдвига методом частотного анализа
    freq_parser = subparsers.add_parser(
        'freq', help='Поиск сдвига методом частотного анализа')
    freq_parser.add_argument('input', help='Путь к зашифрованному файлу')
    args = parser.parse_args()

    if args.command == 'encrypt':
        encrypt_file(args.input, args.output, args.shift)
    elif args.command == 'decrypt':
        decrypt_file(args.input, args.output, args.shift)
    elif args.command == 'analyze':
        frequency_analysis(args.input, args.keywords)
    elif args.command == 'freq':
        frequency_detect_shift(args.input)


if __name__ == "__main__":
    main()
```

Глава 4 Результаты работы программы

4.1 Пример 1: Шифрование

4.1.1 Исходный текст (input.txt)

```
Lorem Ipsum - это текст-"рыба", часто используемый в печати и вэб-дизайне. Lorem Ipsum является стандартной "рыбой" для текстов на латинице с начала XVI века. В то время некий безымянный печатник создал большую коллекцию размеров и форм шрифтов, используя Lorem Ipsum для распечатки образцов. Lorem Ipsum не только успешно пережил без заметных изменений пять веков, но и перешагнул в электронный дизайн. Его популяризации в новое время послужили публикация листов Letraset с образцами Lorem Ipsum в 60-х годах и, в более недавнее время, программы электронной вёрстки типа Aldus PageMaker, в шаблонах которых используется Lorem Ipsum.
```

4.1.2 Команда для шифрования

```
python src/main.py encrypt input.txt encrypted.txt 5
```

4.1.3 Результат

Вывод в консоль:

```
Файл зашифрован и сохранён как encrypted.txt
```

4.1.4 Файл encrypted.txt

```
Qtwjr Nuxzr - вчу чйпцч-"хаёе", ьецчу нцфурбмшйсао ж фйьечн н жвё-инмеотй. Qtwjr Nuxzr джрдйчцд цчетиехчтуо "хаёуо" ирд чйпцчуж те речнтный ц теьере CAN жйпе. Ж чу жхйсд тйпно ёймасдттао фйьечтнп цумиер ёурбэшг пуррйпынг хемсйхуж н щухс эхнщчуж, нцфурбмшд Qtwjr Nuxzr ирд хецфйьечпн уёхемыуж. Qtwjr Nuxzr тй чурбпу шцфйэту фйхйлнр ёйм месйчтаъ нмсйтйтно фдчб жйпуж, ту н фйхйэезтшр ж врйпчхуттао инмеот. Йзу фуфшрдхнмеынн ж тужуй жхйсд фуцршлнрн фшёрнпеынд рнцчуж Qjywfxjy ц уёхемыесн Qtwjr Nuxzr ж 60-ъ зуиеъ н, ж ёурйй тйиежтйй жхйсд, фхузхесса врйпчхуттуо жкхцчпн чнфе Fqizx UfljRfpjw, ж эеёрутеъ пучухаъ нцфурбмшйчцд Qtwjr Nuxzr.
```

4.2 Пример 2: Дешифровка

4.2.1 Исходный текст (encrypted.txt)

```
Qtwjr Nuxzr - вчу чйпцч-"хаёе", ьецчу нцфурбмшйсао ж фйьечн н жвё-инмеотй. Qtwjr Nuxzr джрдйчцд цчетиехчтуо "хаёуо" ирд чйпцчуж те речнтный ц теьере CAN жйпе. Ж чу жхйсд тйпно ёймасдттао фйьечтнп цумиер ёурбэшг пуррйпынг хемсйхуж н щухс эхнщчуж, нцфурбмшд Qtwjr Nuxzr ирд хецфйьечпн уёхемыуж. Qtwjr Nuxzr тй чурбпу шцфйэту фйхйлнр ёйм месйчтаъ нмсйтйтно фдчб жйпуж, ту н фйхйэезтшр ж врйпчхуттао
```

```
 инмеот. Йзу фуфшрдхнмеынн ж тужуй жхйсд фуцршлнрн фшёрнпеынд рнцчуж Qjywfxjy ц
уёхемыесн Qtwjr Nuxzr ж 60-ъ зуиеъ н, ж ёурйй тйиежтйй жхйсд, фхузхесса
врйпчхуттуо жкхцчпн чнфе Fqizx UfljRfpjw, ж эеёрутеъ пучухаъ нцфурбмшйчцд Qtwjr
Nuxzr.
```

4.2.2 Команда для дешифровки

```
python src/main.py decrypt encrypted.txt decrypted.txt 5
```

4.2.3 Результат

Вывод в консоль:

```
Файл дешифрован и сохранён как decrypted.txt
```

4.2.4 Файл decrypted.txt

```
Lorem Ipsum - это текст-"рыба", часто используемый в печати и вэб-дизайне. Lorem
Ipsum является стандартной "рыбой" для текстов на латинице с начала XVI века. В
то время некий безымянный печатник создал большую коллекцию размеров и форм
шрифтов, используя Lorem Ipsum для распечатки образцов. Lorem Ipsum не только
успешно пережил без заметных изменений пять веков, но и перешагнул в электронный
 дизайн. Его популяризации в новое время послужили публикация листов Letraset с
образцами Lorem Ipsum в 60-х годах и, в более недавнее время, программы
электронной вёрстки типа Aldus PageMaker, в шаблонах которых используется Lorem
Ipsum.
```

4.3 Пример 3: Поиск сдвига методом частотного анализа

4.3.1 Исходный текст (encrypted.txt)

```
Qtwjr Nuxzr - вчу чйпцч-"хаёе", ьецчу нцфурбмшйсао ж фйьечн н жвё-инмеотй. Qtwjr
Nuxzr джрдйчцд цчетиехчтуо "хаёуо" ирд чйпцчуж те речнтный ц теьере CAN жйпе. Ж
чу жхйсд тйпно ёймасдттао фйьечтнп цумиер ёурбэшг пуррйпынг хемсйхуж н щухс
эхнщчуж, нцфурбмшд Qtwjr Nuxzr ирд хецфйьечпн уёхемыуж. Qtwjr Nuxzr тй чурбпу
шцфйэту фйхйлнр ёйм месйчтаъ нмсйтйтно фдчб жйпуж, ту н фйхйэезтшр ж врйпчхуттао
 инмеот. Йзу фуфшрдхнмеынн ж тужуй жхйсд фуцршлнрн фшёрнпеынд рнцчуж Qjywfxjy ц
уёхемыесн Qtwjr Nuxzr ж 60-ъ зуиеъ н, ж ёурйй тйиежтйй жхйсд, фхузхесса
врйпчхуттуо жкхцчпн чнфе Fqizx UfljRfpjw, ж эеёрутеъ пучухаъ нцфурбмшйчцд Qtwjr
Nuxzr.
```

4.3.2 Команда для частотного анализа

```
python src/main.py freq encrypted.txt
```

4.3.3 Результат

Вывод в консоль:

```
Частотный анализ зашифрованного текста:
+----------+--------------+-----------+
| Символ   |   Количество |   Частота |
+==========+==============+===========+
| у        |           38 |     7.308 |
```

й	38	7.308
е	31	5.962
н	31	5.962
т	26	5
ч	25	4.808
р	24	4.615
ж	23	4.423
х	22	4.231
ц	17	3.269
ф	16	3.077
п	15	2.885
м	14	2.692
д	14	2.692
r	13	2.5
с	12	2.308
ё	11	2.115
j	10	1.923
о	10	1.923
а	9	1.731
ш	9	1.731
q	8	1.538
w	8	1.538
x	8	1.538
и	8	1.538
n	7	1.346
u	7	1.346
z	7	1.346
t	6	1.154

```
| б         |             6 |     1.154 |
+-----------+---------------+-----------+
| ы         |             6 |     1.154 |
+-----------+---------------+-----------+
| ь         |             5 |     0.962 |
+-----------+---------------+-----------+
| э         |             5 |     0.962 |
+-----------+---------------+-----------+
| ъ         |             5 |     0.962 |
+-----------+---------------+-----------+
| в         |             4 |     0.769 |
+-----------+---------------+-----------+
| з         |             4 |     0.769 |
+-----------+---------------+-----------+
| f         |             4 |     0.769 |
+-----------+---------------+-----------+
| г         |             2 |     0.385 |
+-----------+---------------+-----------+
| щ         |             2 |     0.385 |
+-----------+---------------+-----------+
| л         |             2 |     0.385 |
+-----------+---------------+-----------+
| у         |             2 |     0.385 |
+-----------+---------------+-----------+
| с         |             1 |     0.192 |
+-----------+---------------+-----------+
| а         |             1 |     0.192 |
+-----------+---------------+-----------+
| к         |             1 |     0.192 |
+-----------+---------------+-----------+
| i         |             1 |     0.192 |
+-----------+---------------+-----------+
| l         |             1 |     0.192 |
+-----------+---------------+-----------+
| р         |             1 |     0.192 |
+-----------+---------------+-----------+

Попытка дешифровки с использованием частотного анализа:
Наиболее вероятный сдвиг для английского алфавита: 5
Наиболее вероятный сдвиг для русского алфавита: 5
Расшифрованный текст (английский сдвиг):
Lorem Ipsum - это текст-"рыба", часто используемый в печати и вэб-дизайне. Lorem Ipsum является стандартной "рыбой" для текстов на латинице с начала XVI века. В то время некий безымянный печатник создал большую коллекцию размеров и форм шрифтов, используя Lorem Ipsum для распечатки образцов. Lorem Ipsum не только успешно пережил без заметных изменений пять веков, но и перешагнул в электронный дизайн. Его популяризации в новое время послужили публикация листов Letraset с образцами Lorem Ipsum в 60-х годах и, в более недавнее время, программы электронной вёрстки типа Aldus PageMaker, в шаблонах которых используется Lorem Ipsum.
Расшифрованный текст (русский сдвиг):
Lorem Ipsum - это текст-"рыба", часто используемый в печати и вэб-дизайне. Lorem Ipsum является стандартной "рыбой" для текстов на латинице с начала XVI века. В то время некий безымянный печатник создал большую коллекцию размеров и форм шрифтов, используя Lorem Ipsum для распечатки образцов. Lorem Ipsum не только успешно пережил без заметных изменений пять веков, но и перешагнул в электронный дизайн. Его популяризации в новое время послужили публикация листов Letraset с
```

образцами Lorem Ipsum в 60-х годах и, в более недавнее время, программы электронной вёрстки типа Aldus PageMaker, в шаблонах которых используется Lorem Ipsum.

4.4 Пример 4: Поиск сдвига методом ключевых слов

4.4.1 Исходный текст (encrypted.txt)

```
Qtwjr Nuxzr - вчу чйпцч-"хаёе", ьецчу нцфурбмшйсао ж фйьечн н жвё-инмеотй. Qtwjr Nuxzr джрдйчцд цчетиехчтуо "хаёуо" ирд чйпцчуж те речнтный ц теьере CAN жйпе. Ж чу жхйсд тйпно ёймасдттао фйьечтнп цумиер ёурбэшг пуррйпынг хемсйхуж н щухс эхнщчуж, нцфурбмшд Qtwjr Nuxzr ирд хецфйьечпн уёхемыуж. Qtwjr Nuxzr тй чурбпу шцфйэту фйхйлнр ёйм месйчтаъ нмсйтйтно фдчб жйпуж, ту н фйхйэезтшр ж врйпчхуттао инмеот. Йзу фуфшрдхнмеынн ж тужуй жхйсд фуцршлнрн фшёрнпеынд рнцчуж Qjywfxjy ц уёхемыесн Qtwjr Nuxzr ж 60-ъ зуиеъ н, ж ёурйй тйиежтйй жхйсд, фхузхесса врйпчхуттуо жкхцчпн чнфе Fqizx UfljRfpjw, ж эеёрутеъ пучухаъ нцфурбмшйчцд Qtwjr Nuxzr.
```

4.4.2 Файл с ключевыми словами (keywords.txt)

```
Lorem
ipsum
dolor
```

4.4.3 Команда для поиска сдвига методом ключевых слов

```
python src/main.py analyze encrypted.txt keywords.txt
```

4.4.4 Результат

```
Частотный анализ зашифрованного текста:
+----------+--------------+-----------+
| Символ   |   Количество |   Частота |
+==========+==============+===========+
| у        |           38 |     7.308 |
+----------+--------------+-----------+
| й        |           38 |     7.308 |
+----------+--------------+-----------+
| е        |           31 |     5.962 |
+----------+--------------+-----------+
| н        |           31 |     5.962 |
+----------+--------------+-----------+
| т        |           26 |     5     |
+----------+--------------+-----------+
| ч        |           25 |     4.808 |
+----------+--------------+-----------+
| р        |           24 |     4.615 |
+----------+--------------+-----------+
| ж        |           23 |     4.423 |
+----------+--------------+-----------+
| х        |           22 |     4.231 |
+----------+--------------+-----------+
| ц        |           17 |     3.269 |
```

ф	16	3.077
п	15	2.885
м	14	2.692
д	14	2.692
r	13	2.5
с	12	2.308
ё	11	2.115
j	10	1.923
о	10	1.923
а	9	1.731
ш	9	1.731
q	8	1.538
w	8	1.538
х	8	1.538
и	8	1.538
n	7	1.346
u	7	1.346
z	7	1.346
t	6	1.154
б	6	1.154
ы	6	1.154
ь	5	0.962
э	5	0.962
ъ	5	0.962
в	4	0.769
з	4	0.769

f	4	0.769
г	2	0.385
щ	2	0.385
л	2	0.385
у	2	0.385
с	1	0.192
а	1	0.192
к	1	0.192
i	1	0.192
l	1	0.192
р	1	0.192

Наиболее вероятный сдвиг для дешифровки: 5
Количество совпадений ключевых слов: 2