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

Криптографія

Комп’ютерний практикум №3
Криптоаналіз афінної біграмної підстановки

Виконали:
Студенти групи ФБ-22
Дажук Павло, Копилов Сергій

Київ – 2024

Варіант 8

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

Порядок виконання 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.

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

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 (1).

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

5. Повторювати дії 3-4 доти, доки дешифрований текст не буде змістовним

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.

```python
# Розширений алгоритм Евкліда
def extended_euclid(a, b):
    """
    Розширений алгоритм Евкліда для знаходження найбільшого спільного
дільника та коефіцієнтів
    рівняння Безу для ax + by = gcd(a, b)
    Повертає кортеж (gcd, x, y), де gcd — найбільший спільний дільник a та b,
    x і y — коефіцієнти рівняння Безу.
    """
    if b == 0:
        return a, 1, 0
    gcd, x1, y1 = extended_euclid(b, a % b)
    x = y1
    y = x1 - (a // b) * y1
    return gcd, x, y


# Обернений елемент за модулем
def mod_inverse(a, m):
    """
    Знаходить обернений елемент a за модулем m.
    Повертає обернений елемент, якщо він існує (тобто a і m взаємно прості),
    інакше повертає None.
    """
    gcd, x, _ = extended_euclid(a, m)
    if gcd != 1:
        return None
    return x % m
```

```python
# Розв'язання лінійного рівняння
def solve_linear_congruence(a, b, m):
    """
    Розв'язує лінійне порівняння a * x ≡ b (mod m).
    Повертає список усіх розв'язків.
    """
    gcd, x, y = extended_euclid(a, m)

    # Перевірка, чи існує розв'язок
    if b % gcd != 0:
        return None

    # Знаходимо часткове рішення
    a = a // gcd
    b = b // gcd
    m = m // gcd

    solution = mod_inverse(a, m) * b % m

    # Повертаємо всі можливі розв'язки
    return [solution + i * m for i in range(gcd)]
```

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

```python
# Знаходження найчастіших біграм
def most_frequent_bigrams(text, top_n=5):
    bigrams = [text[i:i + 2] for i in range(0, len(text) - 1, 2)]
    bigram_counts = Counter(bigrams)
    return [bigram for bigram, _ in bigram_counts.most_common(top_n)]
```

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 (1).

```python
# Знаходження всіх можливих варіантів зіставлення біграм
def find_bigram_combinations(bigrams1, bigrams2):
    bigram_combinations = []
    bigram_combinations1 = list(itertools.combinations(bigrams1, 2))
    bigram_combinations2 = list(itertools.combinations(bigrams2, 2))
    for bigram in bigram_combinations1:
        for cipher in bigram_combinations2:
            bigram_combinations.append(list(zip(bigram, cipher)))

    return bigram_combinations


# Індекс біграми
def find_index_bigram(bigram, alphabet):
    letter1 = alphabet.index(bigram[0])
    letter2 = alphabet.index(bigram[1])
    return letter1 * len(alphabet) + letter2


# Пошук a і b
def find_a_b(combo, alphabet):
    m = len(alphabet) ** 2

    x1, y1 = find_index_bigram(combo[0][0], alphabet),
find_index_bigram(combo[0][1], alphabet)
```

```
    x2, y2 = find_index_bigram(combo[1][0], alphabet),
find_index_bigram(combo[1][1], alphabet)

    # Різниці X і Y
    delta_x = (x1 - x2) % m
    delta_y = (y1 - y2) % m

    a_values = solve_linear_congruence(delta_x, delta_y, m)
    if a_values is None:
        return None

    b_values = []
    for a in a_values:
        b_values.append((y1 - a * x1) % m)

    return a_values, b_values
```

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

```
# Розшифрування тексту
def decrypt(text, all_a_solutions, all_b_solutions, alphabet):
    m = len(alphabet) ** 2
    plaintexts = []

    # Перебираємо всі можливі комбінації для a та b
    for a_solutions, b_solutions in zip(all_a_solutions, all_b_solutions):
        for a, b in zip(a_solutions, b_solutions):
            a_inv = mod_inverse(a, m)
            if a_inv is None:
                continue

            plaintext = ''
            for i in range(0, len(text), 2):
                bigram = text[i:i + 2]
                if len(bigram) != 2:
                    continue

                y = find_index_bigram(bigram, alphabet)
                x = (a_inv * (y - b)) % m

                first_letter = alphabet[x // len(alphabet)]
                second_letter = alphabet[x % len(alphabet)]
                plaintext += first_letter + second_letter

            plaintexts.append(plaintext)

    return plaintexts


# Перевірка змістовності тексту
def is_meaningful_text(text, alphabet):
    letter_freqs = letter_frequency(text, alphabet)
    common_letters = ['о', 'е', 'а']
    rare_letters = ['щ', 'ф', 'ц']

    # Перевірка частоти частих літер
    if sum(letter_freqs[letter] for letter in common_letters if letter in
letter_freqs) < 0.2:
        return False

    # Перевірка частоти рідкісних літер
    if sum(letter_freqs[letter] for letter in rare_letters if letter in
```

```
letter_freqs) > 0.05:
        return False

    # Перевірка ентропії
    if entropy(letter_freqs) > 4.6:
        return False

    return True
```

5. Повторювати дії 3-4 доти, доки дешифрований текст не буде змістовним

```
if __name__ == "__main__":
    # Алфавіт
    alphabet = list('абвгдежзийклмнопрстуфхцчшщьыэюя')

    # Шифротекст
    cipher_text = clean_text(read_text("cipher_text.txt"))

    # Знаходимо 5 найчастіших біграм в мові та шифротексті
    most_frequent_bigrams_language = ['ст', 'но', 'то', 'на', 'ен']
    most_frequent_bigrams_text = most_frequent_bigrams(cipher_text, 5)

    # Знаходимо всі можливі варіанти зіставлення біграм
    bigram_combinations =
find_bigram_combinations(most_frequent_bigrams_language,
most_frequent_bigrams_text)

    # Знаходимо всі можливі кандидати на ключ (a, b)
    all_a_solutions, all_b_solutions = [], []
    for bigram_pairs in bigram_combinations:
        result = find_a_b(bigram_pairs, alphabet)
        if result is not None:
            a_solutions, b_solutions = result
            all_a_solutions.append(a_solutions)
            all_b_solutions.append(b_solutions)

    # Розшифровуємо текст знайденими ключами та записуємо змістовний текст у
файл
    plaintexts = decrypt(cipher_text, all_a_solutions, all_b_solutions,
alphabet)
    with open("decrypted_texts.txt", 'w', encoding='utf-8') as f:
        for plaintext in plaintexts:
            # Перевіряємо на змістовність тексту російською мовою
            if is_meaningful_text(plaintext, alphabet):
                f.write(plaintext + '\n'*2)
```

Результат:

Мальчикизаулыбалисьисжаромвзялисьзаделоонирвализолотистыецветыцветычтонаводняю твесьмир…

Мальчики заулыбались и с жаром взялись за дело они рвали золотистые цветы цветы что наводняют весь мир…

Висновки

У роботі реалізовано математичні підпрограми для обчислення оберненого елемента за модулем (розширений алгоритм Евкліда) та розв’язання лінійних порівнянь з урахуванням

усіх можливих розв’язків. Виконано частотний аналіз шифртексту для визначення 5 найчастіших біграм, які зіставлено з найбільш частими біграмами російської мови. Для кожного варіанта зіставлення обчислено можливі ключі (a, b) шляхом розв’язання систем лінійних порівнянь. Шифртекст дешифровано для кожного кандидата на ключ, і перевірено змістовність тексту автоматичним розпізнавачем російської мови, який базується на аналізі частот літер та ентропії.

У результаті отримано змістовний текст шляхом підбору правильного ключа. Реалізовані алгоритми продемонстрували високу ефективність у виконанні криптоаналізу афінної біграмної підстановки.