Національний технійчний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Фізико-технічний інститут

«Криптографія»

Комп’ютерний практикум №3

Криптоаналіз афінної біграмної підстановки

Виконали:

ФБ-21 Жиговець Олександр

ФБ-21 Альгішиєв Дмитро

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

Варіант: 3

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
def extended_gcd(a, b):

    if b == 0:
        return (a, 1, 0)
    else:
        gcd, x1, y1 = extended_gcd(b, a % b)
        x = y1
        y = x1 - (a // b) * y1
        return (gcd, x, y)

def modular_inverse(a, m):

    gcd, x, _ = extended_gcd(a, m)
    if gcd != 1:
        return None
    else:
        return x % m
```

```python
def solve_congruence(a_coeff, b_coeff, modulus):

    divisor, bezout_x, _ = extended_gcd(a_coeff, modulus)

    if b_coeff % divisor != 0:
        return []

    simplified_a = a_coeff // divisor
    simplified_b = b_coeff // divisor
    simplified_mod = modulus // divisor

    inverse_a = modular_inverse(simplified_a, simplified_mod)

    if inverse_a is None:
        return []

    base_sol = (inverse_a * simplified_b) % simplified_mod

    solutions = []
    for i in range(divisor):
        solution = (base_sol + i * simplified_mod) % modulus
        solutions.append(solution)

    return solutions
```

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
def compute_bigram_frequencies(text: str, step: int = 2) -> Counter:

    bigrams = []
    for i in range(0, len(text) - 1, step):
        bg = text[i:i+2]
        if len(bg) == 2:
            bigrams.append(bg)
    return Counter(bigrams)

def read_ciphertext(file_path: str) -> str:

    with open(file_path, 'r', encoding='utf-8') as file:
        text = file.read()
        text = ''.join(filter(str.isalpha, text.lower()))
        return text


file_path = 'TEXT.txt'

ciphertext = read_ciphertext(file_path)

freq = compute_bigram_frequencies(ciphertext, step=2)

top_5 = [bg for bg, _ in freq.most_common(5)]
print("Топ5 біграм :", top_5)
```

✓ 0.0s

```
Топ5 біграм : ['тд', 'рб', 'во', 'що', 'кд']
```

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
COMMON_RUS_BIGRAMS = ["ст", "но", "то", "на", "ен"]
```

```python
common_pairs = list(itertools.combinations(COMMON_RUS_BIGRAMS, 2))
cipher_pairs = list(itertools.combinations(top_5, 2))

for (bg_rus_1, bg_rus_2) in common_pairs:
    x1 = bigram_to_num(bg_rus_1)
    x2 = bigram_to_num(bg_rus_2)

    for (bg_ciph_1, bg_ciph_2) in cipher_pairs:
        y1 = bigram_to_num(bg_ciph_1)
        y2 = bigram_to_num(bg_ciph_2)

        candidates = solve_affine_bigram(x1, y1, x2, y2, modulus=M_SQUARED)
        if not candidates:
            continue
```

```python
def solve_affine_bigram(x1, y1, x2, y2, modulus=M_SQUARED):

    # (y1 - y2) = a*(x1 - x2) (mod m^2)

    diff_y = (y1 - y2) % modulus
    diff_x = (x1 - x2) % modulus

    g, _, _ = extended_gcd(diff_x, modulus)

    if diff_y % g != 0:
        return []

    d = g
    diff_x_prim = diff_x // d
    diff_y_prim = diff_y // d
    mod_prim = modulus // d

    inv_diff_x_prim = modular_inverse(diff_x_prim, mod_prim)
    if inv_diff_x_prim is None:
        return []

    a0 = (diff_y_prim * inv_diff_x_prim) % mod_prim

    solutions = []
    for k in range(d):
        a_candidate = (a0 + k * mod_prim) % modulus
        b_candidate = (y1 - a_candidate*x1) % modulus # b = y1 - a_candidate*x1 (mod m)
        solutions.append((a_candidate, b_candidate))

    return solutions
```

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.

```python
def decrypt_affine_bigram(ciphertext, a, b):

    plain = []
    for i in range(0, len(ciphertext) - 1, 2):
        bg = ciphertext[i:i+2]
        Y = bigram_to_num(bg)
        if Y is None:
            plain.append("??")
            continue

        g, _, _ = extended_gcd(a, M_SQUARED)
        if g != 1:
            return None
        inv_a = modular_inverse(a, M_SQUARED)
        if inv_a is None:
            return None

        X = (inv_a * ((Y - b) % M_SQUARED)) % M_SQUARED
        plain_bg = num_to_bigram(X)
        plain.append(plain_bg)
    return "".join(plain)
```

Перевірка на змістовність

```python
def is_russian_text(text, threshold_common=0.02, threshold_rare=0.005, entropy_threshold=4.4):

    # 1 Частота частих літер
    common_letters = ['о', 'а', 'е']
    count_common = sum(text.count(ch) for ch in common_letters)
    ratio_common = count_common / len(text)

    if ratio_common < threshold_common:
        return False

    # 2 Частота рідкісних літер
    rare_letters = ['ф', 'щ', 'ь']
    count_rare = sum(text.count(ch) for ch in rare_letters)
    ratio_rare = count_rare / len(text)

    if ratio_rare < threshold_rare:
        return False

    # 3 ентропія
    entropy = calculate_entropy(text)
    if entropy > entropy_threshold:
        return False

    return True

def calculate_entropy(text):

    if not text:
        return 0.0
    freq = Counter(text)
    total = len(text)
    entropy = -sum((count / total) * math.log2(count / total) for count in freq.values())
    return entropy
```

Висновки: У ході виконання компютерного практикуму я вивчив зміст поняття аінної біграмної підстановки, випробував та вивчив різні критерії змістовності тексту, попрацював з ентропією та налаштуванням параметрів. Зрозумів як атакувати цю підстановку