ФГБОУ ВО ‘‘Чувашский государственный университет им. И. Н. Ульянова’’.

Факультет: Информатики и вычислительной техники.

Кафедра: Вычислительной техники.

Средства защиты информации

Лабораторная работа №3

Выполнил: студент группы ИВТ-41-18

Сергеев Давид Евгеньевич

Проверил: доцент Ковалев Сергей Васильевич

Чебоксары 2021 г.

Задание:

Целью работы является знакомство с классическим криптографическим алгоритмом - алгоритмом линейного шифрования данных (шифрования гаммированием).

Решение:

Шифры гаммирования относятся к к шифрам замены, но выделяются в собственный класс в связи со своими характерными свойствами и особенностями. В алфавите, заданном для решения криптографических задач каждому символу соответствует его порядковый номер. Это позволяет каждую букву открытого сообщения заменить её естественным порядковым номером в рассматриваемом алфавите, т.о. преобразование числового сообщения в буквенное позволяет однозначно восстановить исходное открытое сообщение. Мой алфавит задан следующим множеством:

{0..9, пробел, A..Z, a..z, A..Я, a..я, знаки препинания}

В соответствии с символами заданы индексы:

{1..161}.

Что касается алгоритма генерации псевдослучайных чисел, он задан следующей ф-ей:

$$T(i+1) = (A * T(i) + C) \bmod B, \qquad \text{где параметры: } \begin{cases} A = 13 \\ B = 161 \\ C = 43 \\ T(0) = 37 \end{cases}$$

Хотя линейный смешанный($C \neq 0$) конгруэнтный метод порождает статистически хорошую псевдослучайную последовательность чисел, он не является криптографически стойким. Генераторы на основе линейного конгруэнтного метода являются предсказуемыми, поэтому их нельзя использовать в криптографии.

Дональд Кнут в главе 3.2.2 книги «Искусство программирования» называет рандомизацию перемешиванием, открытую Картером Бейсом и С.Д.Дархамом, неожиданно лучшим способом генерации случайной последовательности, чем даже алгоритм совмещающий две случайные последовательности, разработанный М.Д. Мак-Ларен и Дж. Марсалья, несмотря на то, что все эти алгоритмы построены на одной и той же идее. Более точной эту оценку можно сделать используя статистические методы, например: критерий промежутков между днями рождений. Так можно выяснить, будет ли генератор вырабатывать приемлемые случайные промежутки между днями рождений.

В качестве примера работы алгоритма шифрования гаммированием предлагаю использовать сообщение: «Современная информатика широко использует псевдослучайные числа.».

1. Перевод сообщения в список порядковых номеров символов алфавита.

[81, 111, 98, 113, 101, 109, 101, 110, 110, 96, 128, 10, 105, 110, 117, 111, 113, 109, 96, 115, 105, 107, 96, 10, 121, 105, 113, 111, 107, 111, 10, 105, 114, 112, 111, 108, 125, 104, 116, 101, 115, 10, 112, 114, 101, 98, 100, 111, 114, 108, 116, 120, 96, 106, 110, 124, 101, 10, 120, 105, 114, 108, 96, 142]

2. Генерация псевдослучайной последовательности (гаммы)

«аеЭьv]дscМВУ`+Пзк<9~ТоаеЭьv]дscМВУ`+Пзк<9~ТоаеЭьv]дscМВУ`+Пзк<9~»

3. Шифрование методом гаммирования

«x\TH}ed4AAVЭдШYrwЮис'u%oqЁ жj5ыJHXйЦgkzЦы9.Б*+VЛАдsDmKDja?blxЭи-» или

[118, 152, 30, 77, 159, 101, 40, 4, 63, 11, 32, 93, 100, 88, 35, 54, 59, 94, 105, 114, 135, 57, 133, 51, 53, 69, 10, 103, 46, 5, 124, 20, 18, 34, 106, 86, 43, 47, 62, 86, 124, 9, 142, 64, 138, 139, 32, 75, 11, 100, 55, 14, 49, 21, 14, 46, 96, 149, 38, 48, 60, 93, 105, 141]

4. Дешифрование

«Современная информатика широко использует псевдослучайные числа.» или [81, 111, 98, 113, 101, 109, 101, 110, 110, 96, 128, 10, 105, 110, 117, 111, 113, 109, 96, 115, 105, 107, 96, 10, 121, 105, 113, 111, 107, 111, 10, 105, 114, 112, 111, 108, 125, 104, 116, 101, 115, 10, 112, 114, 101, 98, 100, 111, 114, 108, 116, 120, 96, 106, 110, 124, 101, 10, 120, 105, 114, 108, 96, 142]

Текст программы:

```
A = 13
B = 161
C = 43
T_0 = 37

STD_INPUT = 'Эту глупую улыбку он не мог простить себе. Увидав эту улыбку,
Долли вздрогнула, как от ' \
            'физической боли, разразилась, со свойственною ей горячностью,
потоком жестоких слов и ' \
            'выбежала из комнаты. С тех пор она не хотела видеть мужа.'  #
Анна Каренина
ALPHABET = [
    '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', ' ', 'A', 'B', 'C',
'D', 'E', 'F', 'G', 'H', 'I',
    'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W',
'X', 'Y', 'Z', 'a', 'b', 'c',
    'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q',
'r', 's', 't', 'u', 'v', 'w',
    'x', 'y', 'z', 'А', 'Б', 'В', 'Г', 'Д', 'Е', 'Ё', 'Ж', 'З', 'И', 'Й',
'К', 'Л', 'М', 'Н', 'О', 'П',
    'Р', 'С', 'Т', 'У', 'Ф', 'Х', 'Ц', 'Ч', 'Ш', 'Щ', 'Ъ', 'Ы', 'Ь', 'Э',
'Ю', 'Я', 'а', 'б', 'в', 'г',
    'д', 'е', 'ё', 'ж', 'з', 'и', 'й', 'к', 'л', 'м', 'н', 'о', 'п', 'р',
'с', 'т', 'у', 'ф', 'х', 'ц',
    'ч', 'ш', 'щ', 'ъ', 'ы', 'ь', 'э', 'ю', 'я', '!', '"', '#', '$', '%',
'&', "'", '(', ')', '*', '+',
    ',', '-', '.', '/', ':', ';', '<', '=', '>', '?', '@', '[', '\\', ']',
'^', '_', '`', '{', '|', '}',
    '~'
]
ALPHABET_LEN = len(ALPHABET)


def alphabet_msg_sequence(msg: str) -> list:
    # msg -> list of char positions in alphabet
    alphabet_pos_sequence = []
    for char in msg:
        alphabet_pos_sequence.append(ALPHABET.index(char))
    return alphabet_pos_sequence


def pseudorandom_number_generator(msg_len: int) -> list:
    # pseudorandom number generator for gamma
    gamma = [T_0]
    for idx in range(1, msg_len):
        # t(i+1) = (A * t(i) + C) mod B
        num = (A * gamma[idx - 1] + C) % B
        gamma.append(num)
    return gamma
```

```
def gamma_to_str(gamma: list) -> str:
    gamma_ = ''
    for idx in gamma:
        gamma_ += ALPHABET[idx]
    return gamma_


def encode(msg_pos_sequence: list, gamma: list):
    encoded_str = ''
    encoded_list = []
    for idx in range(len(gamma)):
        value = (msg_pos_sequence[idx] + gamma[idx]) % ALPHABET_LEN
        encoded_list.append(value)
    for idx in encoded_list:
        encoded_str += ALPHABET[idx]

    return [encoded_str, encoded_list]


def decode(encoded_str: str, gamma: list):
    decoded_str = ''
    decoded_list = []
    for idx in range(len(gamma)):
        value = ALPHABET.index(encoded_str[idx]) - gamma[idx]
        if value < 0:
            value += ALPHABET_LEN
        decoded_list.append(value)
    for idx in decoded_list:
        decoded_str += ALPHABET[idx]

    return [decoded_str, decoded_list]


def main():
    msg = STD_INPUT
    print(f'\n\nMessage: {STD_INPUT}')
    # Перевод сообщения в список позиций символов алфавита
    msg_alphabet_pos_sequence = alphabet_msg_sequence(msg)
    # Генерация псевдослучайной гаммы длиной сообщения msg
    gamma = pseudorandom_number_generator(len(msg))
    gamma_str = gamma_to_str(gamma)
    print(f'\nGamma: {gamma_str}')
    # Шифрование строки
    encoded_str, encoded_list = encode(msg_alphabet_pos_sequence, gamma)
    print(f'\nencoded: {encoded_str}\n')
    # Дешифрование строки
    decoded_str, decoded_list = decode(encoded_str, gamma)
    print(f'decoded: {decoded_str}\n')


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

Пример работы программы:

```
Message: Эту глупую улыбку он не мог простить себе. Увидав эту улыбку, Долли вздр
огнула, как от физической боли, разразилась, со свойственною ей горячностью, по
током жестоких слов и выбежала из комнаты. С тех пор она не хотела видеть мужа.

Gamma: aeЭьv]dscMBУ`+Пзк<9~ToaeЭьv]dscMBУ`+Пзк<9~ToaeЭьv]dscMBУ`+Пзк<9~ToaeЭьv]d
scMBУ`+Пзк<9~ToaeЭьv]dscMBУ`+Пзк<9~ToaeЭьv]dscMBУ`+Пзк<9~ToaeЭьv]dscMBУ`+Пзк<9~T
oaeЭьv]dscMBУ`+Пзк<9~ToaeЭьv]dscMBУ`+Пзк<9~ToaeЭьv]dscMBУ`+Пзк<9~ToaeЭьv]dscM

encoded: "`l'{ds6ЁfЛbжёEnz`чmdx*oeЛ{2o7БSIQнжЩukTн-dW'<Vx`2B9ЁЦJTцЛOwЦ`Мн*v.oTE|
ип^AUBH'?Ocq`чcdД.;aФ}йj5wЦ1WжУvcwСрпэг.?ROL$н8БЦНЙфVvhЦцм-Hk.b'{жpLЙOEZнжiУфбч
c-u>@жД}йr5xJLЭмЦSefЪIб^k*:RИ^2h}ыLEUиKWДШ`Ъ9;o_ohЛ 2n4mЦDM5aSvkЭи9яs).kЩEes|mu

decoded: Эту глупую улыбку он не мог простить себе. Увидав эту улыбку, Долли вздр
огнула, как от физической боли, разразилась, со свойственною ей горячностью, по
током жестоких слов и выбежала из комнаты. С тех пор она не хотела видеть мужа.
```