Ковкель Никита, ФИТ 3-4

Информационная безопасность

Отчет по лабораторной работе № 9.

**Цель:** изучение и приобретение практических навыков разработки и использования приложений для реализации асимметричных шифров.

1. **Практическая часть**

Функция принимает три параметра: start (начальное значение), step (шаг прогрессии) и length (длина последовательности). Создается пустой массив arr, который будет содержать сгенерированную последовательность. В цикле for происходит генерация последовательности. Итерации выполняются от 1 до length - 1, так как первый элемент уже задан в массиве arr (равен step). Внутри цикла происходит вычисление следующего элемента последовательности. Создается переменная sum и инициализируется нулевым значением. Вложенный цикл for проходит по всем элементам массива arr и выполняет суммирование их значений. Результат суммирования сохраняется в переменной sum. После завершения вложенного цикла, к значению sum прибавляется значение step, что формирует новый элемент прогрессии.

|  |
| --- |
| def Sverhvozr(start, step, length):  arr = [step]  for i in range(1, length):  sum = 0  for el in arr:  sum += el  arr.append(sum + step)  return arr |

Листинг 9.1 – Функция генерации сверхвозрастающей последовательности



Рисунок 9.1 – Результат сверхвозрастающей последовательности

Вычисление нормальной последовательности (открытого ключа);

В начале определен список secretKey, который содержит некоторые числа. Функция Normvozr принимает secretKey в качестве параметра. Создается переменная sum и инициализируется нулевым значением. Она будет использоваться для вычисления суммы всех элементов в secretKey. В цикле for происходит итерация по каждому элементу secretKey. Внутри цикла каждый элемент el суммируется с текущим значением переменной sum. После завершения цикла полученная сумма сохраняется в переменной n, увеличенной на 5. Создается переменная a, которая равна n - 1. Создается новый список arr, в котором каждый элемент element из secretKey умножается на a и берется остаток от деления на n. Это выполняется для каждого элемента в списке secretKey с использованием генератора списка. Возвращается список arr, но в обратном порядке с помощью среза [::-1]. Это означает, что порядок элементов возвращаемого списка будет обратным по сравнению с исходным списком arr.

|  |
| --- |
| secretKey = [2, 3, 7, 13, 27, 52, 105, 210]  def Normvozr(secretKey):  sum = 0  for el in secretKey:  sum += el  n = sum + 5  a = n - 1  arr = [(element \* a) % n for element in secretKey]  return arr[::-1] |

Листинг 9.2 – Функция генерации нормальной последовательности



Рисунок 9.2 – Результат нормальной последовательности

Зашифрование и расшифрование сообщения, состоящего из собственных фамилии, имени и отчества;

В начале кода определены несколько переменных и списков, включая сообщение (message), список w, числа q, r, а также другие переменные, используемые в процессе. Создается пустой список b, который будет использоваться для вычисления шифра. Выполняется цикл, в котором каждый элемент element из списка w умножается на r, а затем берется остаток от деления на q. Полученные значения добавляются в список b с использованием генератора списка. Затем следует цикл, который обрабатывает каждый символ char в сообщении message. Для каждого символа char выполняется цикл, который проходит по каждому биту символа и вычисляет сумму S путем умножения текущего бита на соответствующий элемент из списка b. Сумма S добавляется в список cipher, после чего S сбрасывается в ноль. Время выполнения шифрования записывается в переменную executionTime. Зашифрованный текст формируется путем объединения элементов списка cipher в строку, разделенных пробелами. Затем следует блок кода для расшифровки. Вычисляется обратное значение n для числа r по модулю q с помощью функции mod\_inverse(). В списке S1 каждый элемент из списка cipher умножается на n и берется остаток от деления на q. Создается список S2, который содержит элементы списка w в отсортированном порядке по убыванию. Для каждого элемента element в списке S1 выполняется цикл, в котором происходит разбор числа element на биты. Используя значения из списка S2, определяется, какие биты имеют значение 1 или 0, и формируется бинарный код m. Затем бинарный код преобразуется в символ с помощью chr() и добавляется к строке text.

|  |
| --- |
| message = "KovkelNikitaVictorovich"  w = [2, 7, 11, 21, 42, 89, 180, 354]  q = 881  r = 588  b = []  S = 0  cipher = []  S1 = []  text = ""  start = time.time()  b = [(element \* r) % q for element in w]  for i in range(len(message)):  char = list(format(ord(message[i]), '08b'))  for i in range(len(char)):  S += int(char[i]) \* b[i]  cipher.append(S)  S = 0  end = time.time()  executionTime = end - start  encrypted\_text = ' '.join(map(str, cipher))  start = time.time()  n = mod\_inverse(r, q)  S1 = [(element \* n) % q for element in cipher]  S2 = sorted(w, reverse=True)  for element in S1:  n = element  m = [0, 0, 0, 0, 0, 0, 0, 0]  while n != 0:  i = 0  f = S2[i]  while n < f:  f = S2[i]  i += 1  m[w.index(f)] = 1  n = n - f  binCode = [chr(int(''.join(map(str, m)), 2))]  text += ''.join(binCode)  end = time.time()  executionTime = end - start |

Листинг 9.3 – Программный код шифрования и расшифровывания сообщения

**Оценка времени:**

Разработанный алгоритм быстро шифрует, но время расшифрования больше времени зашифрования.





Рисунок 9.4 – Время зашифрования и расшифрования

**Вывод:** на данной лабораторной работе мыизучили и приобрели практические навыки разработки и использования приложений для реализации асимметричных шифров.