

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

імені ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики

Кафедра системного програмування і спеціалізованих комп’ютерних систем

**Лабораторна робота №** 4

з дисципліни «Захист інформації в комп’ютерних системах»

Виконав: Стецюренко І. С,

Студент групи КВ-03

Перевірив(ла):\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Київ 2022**Лабораторна робота**

**Розробка програмних засобів шифрування інформації**

**Теоретичні відомості**

**Концепція криптосистеми з відкритим ключем**

Ефективними системами криптографічного захисту є криптосистеми з відкритим ключем, звані також асиметричними криптосистемами. У таких системах для шифрування даних використовується один ключ, а для розшифрування - інший (звідси і назва - асиметричні).

Перший ключ є відкритим і може бути опублікований для використання всіма користувачами системи, які зашифровують дані. Розшифрування даних за допомогою відкритого ключа неможливо. Для розшифрування даних одержувач зашифрованої інформації використовує другий ключ, який є секретним. Зрозуміло, ключ розшифрування не може бути визначений з ключа шифрування.

**Однонаправлені функції**

Вся концепція криптосистем з відкритим ключем заснована на застосуванні однонаправлених функцій (one way functions). Однак, точне визначення цього класу функцій з математичної точки зору дати досить складно. Неформально односпрямовану функцію можна визначити наступним чином.

Нехай X і Y - довільні множини. Функція

f(X) -> Y,

є однонаправленою, якщо для всіх х, що входять в Х, легко обчислити функцію f(x), і в той же час для більшості y, що входять в Y, отримати будь-яке значення x, що входить в X, таке що f(x) = y досить складно (при цьому вважають, що існує, принаймні, одне таке значення x).

На жаль, зараз математика не в змозі дати нам відповідь на питання, чи існують такі функції взагалі або ж це тільки красива гіпотеза. Проте вдалося виявити кілька залежностей, які можуть бути використані (і використовуються!) В якості односпрямованих. Основний критерій зарахування функції до класу односпрямованих - відсутність ефективних алгоритмів зворотного перетворення. Найпростіший приклад

односпрямованої функції - множення цілих чисел.

**Алгоритм RSA**

В даний час найбільш розвинутим методом криптографічного захисту інформації з відомим ключем є RSA, названий так за початковими літерами прізвищ його винахідників (Rivest, Shamir і Adleman). Перед тим як приступити до викладу концепції методу RSA, необхідно визначити деякі терміни. Під простим числом будемо розуміти таке число, яке ділиться тільки на 1 і саме на себе. Взаємно простими числами будемо називати такі числа, які не мають жодного спільного дільника, крім 1.

Криптостійкість алгоритму RSA грунтується на припущенні, що так важко визначити секретний ключ за відомим, оскільки для цього необхідно вирішити задачу про існування дільників цілого числа. Дане завдання є NP-повною. Відомі точні алгоритми для вирішення даного завдання мають експонентну оцінку обчислювальної складності, наслідком чого є неможливість отримання точних рішень для задач великої і навіть

середньої розмірності. Більш того, саме питання існування ефективних алгоритмів розв'язання NP - повних задач є до теперішнього часу відкритим. У зв'язку з цим для чисел, що складаються з 200 цифр (а саме такі числа рекомендується використовувати), традиційні методи вимагають виконання величезного числа операцій (близько 1023).

**ЗАВДАННЯ**

1. Розробити програмний комплекс, який реалізує приклади 1, 2, за умови - р і q прості числа.
2. За допомогою стандартних засобів MS Office захистити документ MS Word цифровим підписом.

**Приклад 1.**

Щоб використовувати алгоритм RSA треба спочатку згенерувати відкритий і секретний ключі, виконавши наступні кроки:

1. Виберемо два дуже великих простих числа p і q.
2. Визначимо n як результат множення p на q (n = p \* q).
3. Виберемо велике випадкове число, яке назвемо d. Це число повинне бути взаємно простим з результатом множення (p-1) \* (q-1).
4. Визначимо таке число е, для якого є істинним наступне співвідношення: (e \* d) mod ((p-1) \* (q-1)) = 1.
5. Назвемо відкритим ключем числа е і n, а секретним ключем числа d і n.

Тепер, щоб зашифрувати дані по відомому ключу {e, n}, необхідно робити наступне:

1. розбити текст, що необхідно зашифрувати, на блоки, кожен з яких може бути представлений у вигляді числа

M(i)=0, 1,..., n-1;

1. зашифрувати текст, що розглядається як послідовність чисел M(i), за формулою:

С(i)=(M(i)^e) mod n.

Щоб розшифрувати ці дані використовуючи секретний ключ {d, n}, необхідно виконати наступні обчислення: M (i) = (C (i) ^ d) mod n. В результаті буде отримано множина чисел M (i), які представляють собою вихідний текст.

**Хід роботи**

1. Мною було розроблено програмний комплекс на мові програмування Python. Я використав такі числа: p = 3, q = 11. А отже n = 55, тому я вирішив зашифрувати літери англійського алфавіту, цифри 0-9, та деякі деякі символи, а саме “ . ” , “ , ” , “< ” , “ > ” , “ ? ” , “ ’ ” , “ : ” , “ ; ” , “ { ” , “ } ” , “ [ ” , “ ] ” , “ ! ” , “ @ ” , “ № ” , “ # ” , “ ^ ” , “ & ” .
2. Реалізовано такі функції:

* range\_char(start, stop) – заповнює масив літерами.
* zap()- проходить заповнення масиву 55 символами .
* encrypt() – зашифровує вхідний текст.
* decipher() – розшифровує заданий код.
* q\_exit() – організовує процес продовження роботи програми або виходу з неї.
* main() – вибір потрібного функціоналу.

1. Код програми:
3. **def** range\_char(start, stop):
4. **return** (chr(n) **for** n **in** range(ord(start), ord(stop) + 1))

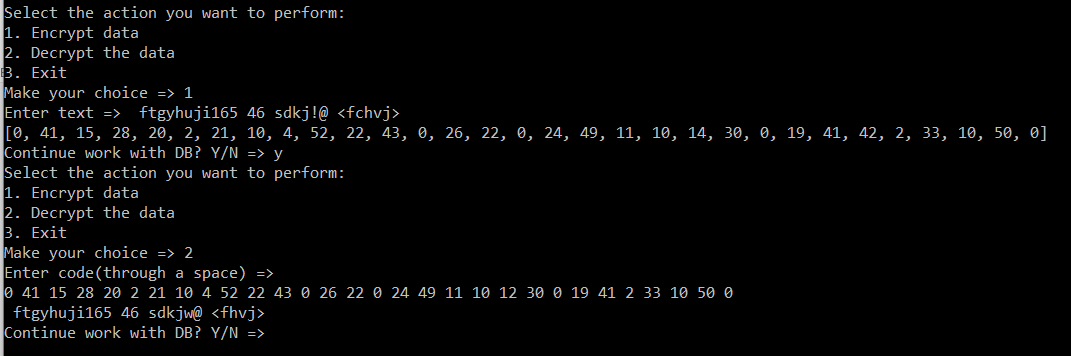
7. **def** zap():
8. mas = []
9. mas.insert(0, " ")
11. **i = 1**
12. **for** character **in** range\_char("a", "z"):
13. mas.insert(i, character)
14. i = i + 1
16. **for j in range(0, 10):**
17. mas.insert(i, j)
18. i = i + 1
20. mas.insert(i, ".")
21. **i = i + 1**
22. mas.insert(i, ",")
23. i = i + 1
24. mas.insert(i, "<")
25. i = i + 1
26. **mas.insert(i, ">")**
27. i = i + 1
28. mas.insert(i, "?")
29. i = i + 1
30. mas.insert(i, "'")
31. **i = i + 1**
32. mas.insert(i, ":")
33. i = i + 1
34. mas.insert(i, ";")
35. i = i + 1
36. **mas.insert(i, "{")**
37. i = i + 1
38. mas.insert(i, "}")
39. i = i + 1
40. mas.insert(i, "[")
41. **i = i + 1**
42. mas.insert(i, "]")
43. i = i + 1
44. mas.insert(i, "!")
45. i = i + 1
46. **mas.insert(i, "@")**
47. i = i + 1
48. mas.insert(i, "#")
49. i = i + 1
50. mas.insert(i, "№")
51. **i = i + 1**
52. mas.insert(i, "^")
53. i = i + 1
54. mas.insert(i, "&")
55. **return** mas

58. **def** encrypt():
59. *# p = 5, q = 11, n = p \* q = 55,*
60. d = 3
61. **e = 27**
63. ABC = zap()
65. *# print(ABC)*
66. ***# print(len(ABC))***
67. **print**("Enter text =>", end=' ')
69. line = input()
70. line = line.lower()
71. ***# print(line)***
72. result = []
73. j = 0
74. **for** char **in** line:
75. **for** i **in** range(0, len(ABC)):
76. **if str(ABC[i]) == str(char):**
77. *# print(char)*
78. result.insert(j, (i \*\* e) % 55)
79. j = j + 1
81. **print(result)**
82. **return** q\_exit()

85. **def** decipher():
86. ***# p = 5, q = 11, n = p \* q = 55,***
87. d = 3
88. e = 27
90. ABC = zap()
92. **print**("Enter code(through a space) =>")
93. code = input()
94. code = code
95. mas = []
96. **j = 0**
97. **for** i **in** range(0, len(code) - 1):
98. **if** code[i + 1] == ' ':
99. **if** i == 0:
100. mas.insert(j, int(f"{code[0]}"))
101. **j = j + 1**
102. **elif** i == 1:
103. mas.insert(j, int(f"{code[0]}{code[1]}"))
104. j = j + 1
105. **elif** code[i - 2] == ' ':
106. **mas.insert(j, int(f"{code[i - 1]}{code[i]}"))**
107. j = j + 1
108. **elif** code[i - 1]:
109. mas.insert(j, int(f"{code[i]}"))
110. j = j + 1
111. **if code[len(code) - 3] == ' ':**
112. mas.insert(j, int(f"{code[len(code) - 2]}{code[len(code) - 1]}"))
113. j = j + 1
114. **if** code[len(code) - 2] == ' ':
115. mas.insert(j, int(f"{code[len(code) - 1]}"))
116. **j = j + 1**
117. *# print(mas)*
118. result = []
119. j = 0
120. **for** record **in** mas:
121. **result.insert(j, ABC[((record \*\* d) % 55)])**
122. j = j + 1
123. *# print(result)*
124. transform\_result = ""
125. **for** record **in** result:
126. **transform\_result = str(transform\_result) + str(record)**
127. **print**(transform\_result)
128. **return** q\_exit()

131. **def q\_exit():**
132. **print**('Continue work with DB? Y/N =>', end=' ')
133. **while** 1:
134. c = input()
135. **if** c == 'Y' **or** c == 'y':
136. **return 1**
137. **elif** c == 'N' **or** c == 'n':
138. **return** 0
139. **else**:
140. **print**("The wrong value was entered, it will be entered again. Y/N =>", end=' ')

143. **def** main():
144. flag = 1
145. **while** flag:
146. **print("Select the action you want to perform:")**
147. **print**("1. Encrypt data")
148. **print**("2. Decrypt the data")
149. **print**("3. Exit")
150. **print**("Make your choice =>", end=' ')
152. num = input()
153. num = int(num)
154. **if** num == 1:
155. flag = encrypt()
156. **if num == 2:**
157. flag = decipher()
158. **if** num == 3:
159. **return** 0

162. main()
163. Демонстрація роботи програми
164. Потрібно захистити документ MS Word цифровим підписом. Тому я вирішив захистити цей файл.

