Bezpieczeństwo systemów i usług informatycznych 2

Laboratorium 2

Paul Paczyński 254307

GRUPA PON 14:30
08.11.2023

Wprowadzenie

Zadanie laboratoryjne polegało na implementacji mechanizmu szyfrującego wiadomość dowolną metodą. Szyfrowanie wiadomości ma na celu zapewnienie bezpieczeństwa przesyłanych danych i zapewnienie możliwości ich odczytania upoważnionym osobom.

Rozróżniamy aktualnie głównie szyfrowanie symetryczne i szyfrowanie asymetryczne. W przypadku kluczy symetrycznych, używany jest jeden klucz, dzięki któremu można zakodować i odkodować wiadomość. W szyfrach asymetrycznych wykorzystujemy więcej niż jeden klucz, jeden do zakodowania wiadomości i inny do jej odkodowania. W programie na laboratorium zastosowano metodę szyfrowania symetrycznego.

Jako bazę programu wykorzystano szyfr bramką XOR. Generowany jest klucz o podanej długości bajtów. W procesie szyfrowania kolejne bajty wchodzą do bramki XOR z kolejnymi bajtami klucza, wynikiem jest zaszyfrowana wiadomość. W przypadku klucza mniejszego od wiadomości, zaczynamy podstawiać do bramki XOR bajty klucza od początku. Deszyfrowanie odbywa się na tej samej zasadzie. Sposób ten zapewnia ten sam rozmiar wiadomości zatajonej, co jawnej.

Implementacja

Klucz generowany jest prostą metodą generująca losowe znaki:

```
2  def generate_key(size):
3      key = ""
4      for i in range(size):
5          key +=chr(random.randrange(33,127))
6      return key
```

Do szyfrowania wiadomości używane są stringi reprezentujące poszczególne bity, jest to najwygodniejszy, choć mało optymalny sposób pracy z bitami w języku Python. Warunek sprawdzający długość key_index pozwala na szyfrowanie wiadomości kluczem krótszym od samej wiadomości.

```
 8  def encrypt(text,key):
 9      encrypted = ""
10      key_index = 0
11      for i in range(len(text)):
12          x = format(ord(text[i]),'08b')
13          if (key_index == len(key)):
14              key_index = 0
15          y = format(ord(key[key_index]),'08b')
16          encrypted += xor(x,y)
17          key_index += 1
18      return bytes_to_chr(encrypted)
19
```

Xor to metoda wprowadzająca logikę bramki XOR dla jednego bajta.

```
20  def xor(x,y):
21      encrypted_byte = ""
22      for i in range(8):
23          if x[i] == y[i]:
24              encrypted_byte += "0"
25          else:
26              encrypted_byte += "1"
27
28      return encrypted_byte
29
```

Bytes_to_chr zamienia poszczególne bajty na typ char, dając końcową zaszyfrowaną wiadomość.

```
30 def bytes_to_chr(text):
31     message = ""
32     for i in range(0,len(text),8):
33         byte = text[i:i+8]
34         message += chr(int(byte,2))
35     return message
36
```

Przykład działania programu przedstawiono poniżej.

```
WIADOMOSC:
Szyfrowanie i odszyfrowywanie wiadomosci

Skompresowana wiadomsoc:
►Szyfrowanie dsmc $h¬ñ5sk$h¬Lñ5m/Ë¿?

Dekompresowana wiadomosc:
Szyfrowanie i odszyfrowywanie wiadomosci

Klucz: Ny[_vDfqG)%(rI7:0>SR+j/D_4#Glr~t
Zakodowana wiadomość: ^*!&►6    ♠&GLMR-DWS→74⁰/À>A#VÐÉ{
Odkodowana wiadomość ►Szyfrowanie dsmc $h¬ñ5sk$h¬Lñ5m/Ë¿?
Dekompresowana wiadomość po odkodowaniu: Szyfrowanie i odszyfrowywanie wiadomosci
```