Пшенко Артем ФИТ 3-4

Отчет по лабораторной работе № 14

Исследование методов текстовой стеганографии (алгоритм модификации числа пробелов)

Информационная безопасность

Вариант 9

Цель данного алгоритма текстовой стеганографии - скрыть секретное сообщение в текстовом документе (.docx) без вызывающего подозрений изменения его внешнего вида. То есть, мы хотим внедрить информацию таким образом, чтобы обычный пользователь не заметил ничего подозрительного, а сообщение осталось доступным только тем, кто знает, что оно там есть и как его извлечь.

Описание алгоритма.

Шифрование сообщения:

1. Сообщение, которое мы хотим скрыть, конвертируется в последовательность битов. Каждый символ сообщения представляется в виде 8-битного двоичного числа.
2. Для обозначения конца сообщения добавляется пробел.

Внедрение сообщения:

1. Оригинальный документ открывается, и каждый параграф разбивается на слова.
2. Для каждого слова из оригинального текста:
   1. Если есть еще биты сообщения для внедрения:
      1. Берется следующий бит сообщения.
      2. Если бит равен 1, к слову добавляется двойной пробел, иначе - одинарный.
   2. Если биты закончились, добавляется одинарный пробел.

Сохранение изменений: измененный текст сохраняется в новом документе.

Листинг 1. Исходный код алгоритма модификации числа пробелов

|  |
| --- |
| import docx  def encode\_message(original\_doc\_path, message, output\_doc\_path):  doc = docx.Document(original\_doc\_path)  message += ' ' # Добавляем пробел в конец сообщения для обозначения конца  message\_bin = ''.join(format(ord(i), '08b') for i in message) # 01110011 01100101 01100011 01110010 01100101 01110100 + 00100000 (это пробел выше)  # print(message\_bin)  for paragraph in doc.paragraphs:  words = paragraph.text.split()  # print(words)  encoded\_words = []  for word in words:  if message\_bin:  print(message\_bin)  bit = message\_bin[0]  print('bit: ', bit)  message\_bin = message\_bin[1:]  if bit == '1':  encoded\_words.append(word + ' ') # Добавляем двойной пробел после слова  print(encoded\_words)  else:  encoded\_words.append(word + ' ') # Добавляем одинарный пробел после слова  print(encoded\_words)  else:  encoded\_words.append(word + ' ') # Если сообщение закончилось, добавляем одинарный пробел  paragraph.text = ''.join(encoded\_words)  doc.save(output\_doc\_path)  def decode\_message(encoded\_doc\_path):  doc = docx.Document(encoded\_doc\_path)  message\_bin = ''  for paragraph in doc.paragraphs:  words = paragraph.text.split(' ')  for i in range(len(words) - 1): # Исключаем последнее слово, так как оно может не содержать пробела  if words[i+1] == '': # Если следующее слово пустое, это был двойной пробел  message\_bin += '1'  elif words[i]: # Если текущее слово не пустое, это был одинарный пробел  message\_bin += '0'  message = ''  for i in range(0, len(message\_bin), 8): # Разбиваем двоичное сообщение на байты  byte = message\_bin[i:i+8]  message += chr(int(byte, 2)) # Преобразуем каждый байт в символ  return message.rstrip() # Удаляем пробел в конце сообщения  encode\_message('original.docx', 'secret', 'encoded.docx')  message = decode\_message('encoded.docx')  print(message) |

Таким образом, в данной лабораторной работе был реализован алгоритм модификации числа пробелов для сокрытия сообщения. Данный алгоритм позволяет скрыть текстовое сообщение в текстовом документе без его заметного изменения. Скрытое сообщение сохраняется в виде последовательности пробелов между словами. При извлечении сообщения из измененного документа, пробелы интерпретируются как биты, а затем преобразуются обратно в символы, составляющие исходное сообщение.