УДК 004.72.056.52:003.27:004.438

ПРОГРАМНА РЕАЛІЗАЦІЯ ПРИХОВУВАННЯ СЕКРЕТНОГО ПОВІДОМЛЕННЯ У ЗОБРАЖЕННІ МЕТОДОМ PVD

Микола Головін, Дмитро Гузачов, Дмитро Приступа

SOFTWARE IMPLEMENTATION OF HIDDEN SECRET MESSAGE IN AN IMAGE USING THE PVD METHOD

Mykola Holovin, Dmytro Huzachov, Dmytro Prystupa

Волинський національний університет імені Лесі Українки, просп. Волі, 13,Луцьк, 43025, Україна

Abstract. *The paper presents the code of a training program for steganographic hiding of text information in graphic files. Hiding is implemented using the PVD (Pixel Value Difference) method in Python. The program can be used for both educational and practical purposes. The Pillow library was used to implement the program. This graphic library is not specialized for steganographic needs. A positive quality of the library is the ability to visualize the information hiding mechanism. This is interesting for educational purposes.*

Постановка наукової проблеми. Глобальна інформаційна мережа Інтернет робить передачу інформації відкритими каналами зв'язку легкою в будь-яку точку Земної кулі. Однак, на жаль, при цьому існує широкий спектр можливостей із перехоплення повідомлень. Тому, існує необхідність передачі важливої інформації в зашифрованому або прихованому вигляді. При вивченні криптографії та стеганографії цікаво мати лаконічні, прозорі аплікації програмних кодів механізмів приховування інформації у медійних файлах.

Метою цієї роботи є реалізація простого лаконічного програмного коду приховування текстової інформації в графічних файлах методом PVD, засобами бібліотеки Pillow мови Python, для використання цього коду в навчальних і прикладних цілях.

PVD (Pixel Value Difference) – це алгоритм стеганографії, який використовує різницю між значеннями пікселів для закодування та приховування інформації в цифрових зображеннях. Алгоритм PVD є одним з кращих алгоритмів, який використовується в стеганографії поряд з іншими ефективними методами (LSB та DCT) [1].

Виклад основного матеріалу дослідження. Код, представлений нижче, реалізує приховування секретного повідомлення у зображенні та його подальше вилучення за допомогою методу, що базується на зміні різниці колірних каналів пікселів.

Пояснення механізму PVD, що впроваджує текст в зображення складається з двох основних функцій: **pvd_encode**, що вставляє повідомлення у зображення та **pvd_decode,** що вилучає повідомлення із зображення. Процес роботи програми із впровадження зображення у повідомлення має наступні етапи. 1. Відкриття вхідного зображення (порожній контейнер) та перетворення його на двомірний масив значень пікселів, що має три складових Red, Green, Blue. 2. Перетворення повідомлення на двійковий формат. 3. Визначення вимірів зображення за шириною width і висотою height. 4. Відкриття відповідних циклів сканування зображення за висотою і шириною та перебір за пікселями парами. 5. Розрахунок різниці між червоними компонентами двох сусідніх пікселів (diff). 6. Пошук та підбір пар пікселів для яких ця різниця була б така, щоб число бітів (n_bits), які можна було б закодувати, не перевищувало трьох. 7. Вирізка із двійкового повідомлення сегменту довжиною n_bits. 8. Перетворення цього двійкового сегменту назад у десяткове число і зміна червоних компонент пікселів так, щоб відображати нову різницю (new_diff).

Таке приховування повідомлення, не надто змінює зображення, тому що зміни сфокусовані на невеликих змінах різниці між пікселями, що мають близьку кольорову гаму червоного.

```
from PIL import Image
def pvd_encode(cover_image_path, secret_message, stego_image_path):
    img = Image.open(cover_image_path).convert('RGB')
    width, height = img.size
    binary_message = ''.join(format(ord(char), '08b') for char in secret_message)
    message_len = len(binary_message)
    data_index = 0
    for i in range(0, height - 1, 2):
        for j in range(0, width - 1, 2):
            if data_index < message_len:
                p1, p2 = img.getpixel((j, i))[0], img.getpixel((j + 1, i))[0]
                diff = abs(p1 - p2)
                if   diff <= 7:
                    n_bits= 3
                    if data_index +n_bits> message_len: n_bits= message_len - data_index
                    binary_chunk = binary_message[data_index:data_index + n_bits]
                    data_index +=n_bits
                    new_diff = int(binary_chunk, 2)
                    sp=(p1+p2)//2
                    if p1 < p2:
                        p1_new = sp-new_diff//2
                        p2_new = sp+(new_diff-new_diff//2)
                    else:
                        p1_new = sp+(new_diff-new_diff//2)
                        p2_new = sp-new_diff//2
                    img.putpixel((j,i),(p1_new,img.getpixel((j,i))[1],img.getpixel((j,i))[2]))
                    img.putpixel((j+1,i),(p2_new,img.getpixel((j+1,i))[1],img.getpixel((j+1,i))[2]))
    img.save(stego_image_path)
```

Пояснення механізму PVD, що вилучає текст із зображення. 1. Відкриття, сканування зображення із прихованим повідомленням та визначення різниці між червоним кольором у пар пікселів. 2. Відновлення двійкового, а далі і звичайного представлення тексту.

```
def pvd_decode(stego_image_path):
    img = Image.open(stego_image_path).convert('RGB')
    width, height = img.size
    binary_message = ''
    for i in range(0, height - 1, 2):
        for j in range(0, width - 1, 2):
            p1, p2 = img.getpixel((j, i))[0], img.getpixel((j + 1, i))[0]
            diff = abs(p1 - p2)
            if diff <= 7:
                n_bits= 3
                binary_chunk = bin(diff)[2:].zfill(n_bits)
                binary_message += binary_chunk
    decoded_message = ''
    for i in range(0, len(binary_message), 8):
        if i + 8 > len(binary_message): break
        byte = binary_message[i:i + 8]
        if '#'!=chr(int(byte, 2)):
            decoded_message += chr(int(byte, 2))
        else: break
    return decoded_message
if __name__ == '__main__':
    cover_image_path = 'cover.png' stego_image_path = 'stego_image.png'
    secret_message = 'This is a secret message! #'
    pvd_encode(cover_image_path, secret_message, stego_image_path)
    decoded_message = pvd_decode(stego_image_path); print("Decoded message:", decoded_message)
```

Висновки та перспективи подальшого дослідження

- Реалізовано просту програму, що дозволяє приховувати текстову інформацію в графічному файлі. Контроль заповненого і не заповненого графічного контейнера не виявляє відмінностей.
- Програма цікава для навчальних цілей завдяки лаконічності і прозорості програмного коду. Механізми приховування інформації в графічному файлі тут легко візуалізуються.

Бібліографія

1. Mykola Holovin, Nina Holovina Educational example of masking textual information in a photographic signal. Journal «ScienceRise: Pedagogical Education» No4(49)2022. P. 24-28 (Index Copernicus) http://journals.uran.ua/sr_edu/article/view/261051/258566