DOI: https://doi.org/10.36910/6775-2524-0560-2026-63-13
УДК 004.056.5:004.932
Головін Микола Борисович, к.фіз.-мат.н., доцент
https://orcid.org/0000-0003-4516-4677
Головіна Ніна Анатоліївна, к.фіз.-мат.н., доцент
https://orcid.org/0000-0002-1152-1536
Волинський національний університет імені Лесі Українки, м. Луцьк, Україна

ПРОГРАМНА РЕАЛІЗАЦІЯ НАВЧАЛЬНОГО ПРИКЛАДУ СТЕГАНОГРАФІЧНОГО ПРИХОВУВАННЯ ТЕКСТІВ В ГРАФІЧНИХ ФАЙЛАХ МЕТОДОМ PVD

Головін М.Б., Головіна Н.А. Програмна реалізація навчального прикладу стеганографічного приховування текстів в графічних файлах методом PVD. Робота присвячена розробці навчального програмного застосунку, що має покращити проведення занять зі стеганографії за темою приховування текстів в графічних файлах методом PVD. Пропонована програма має відкритий код, написаний максимально прозоро та лаконічно з використанням засобів лише структурного та функціонального програмування. Це ключовий момент для зниження бар'єру сприйняття коду для студентів, які не є спеціалістами з програмування і вивчають стеганографію. Наочності сприяє використана при реалізації програми бібліотека Pillow. Ця графічна бібліотека не є спеціалізованою для стеганографічних потреб. Позитивною якістю бібліотеки є можливість візуалізації механізму приховування інформації на рівні двійкових кодів. Проведено випробування програми з текстами різної довжини та з графічними контейнерами (фотографіями) різного ґатунку. Експеримент показав коректне впровадження текстів у графічний файл та відповідне коректне вилучення текстів з цих файлів. Програма пройшла апробацію, використовується на лекційних та лабораторних заняттях з криптографії та стеганографії. Додаток дозволяє приховувати великі масиви текстів, що є цікавим для відпрацювання дій з стеганоаналізу на предмет закладки в фотографії та інші графічні зображення секретних текстових повідомлень. Програма може бути використана, як для навчальних, так і практичних цілей.
Ключові слова: стеганографія, захист інформації, приховування інформації, маскування інформації, зображення, фотографії, мова Python, бібліотека Pillow.

Holovin M., Holovina N.A. Software implementation of an educational example of steganographic text hiding in image files using the PVD method. The work is devoted to the development of an educational software application intended to improve the teaching of steganography classes on the topic of hiding text in image files using the PVD method. The proposed program is open source, written as transparently and concisely as possible using only structural and functional programming techniques. This is a key factor in lowering the barrier to understanding the code for students who are not programming specialists but study steganography. The use of the Pillow library contributes to the clarity of the implementation. This graphics library is not specialized for steganographic purposes; however, its advantage lies in the ability to visualize the information-hiding mechanism at the level of binary codes. The program was tested with texts of different lengths and with graphical containers (photographs) of various types. The experiments demonstrated correct embedding of texts into image files as well as accurate extraction of texts from these files. The program has undergone approbation and is used in lectures and laboratory classes on cryptography and steganography. The application allows hiding large volumes of text, which is useful for practicing steganalysis aimed at detecting hidden secret textual messages embedded in photographs and other graphic images. The program can be used for both educational and practical purposes.
Keywords: steganography, information security, information hiding, information masking, images, photographs, Python language, Pillow library.

Постановка наукової проблеми. В умовах постійного зростання обсягів цифрової інформації зростає і потреба в забезпеченні її конфіденційності, достовірності та автентичності. Захист інформації стає важливою задачею інформаційних технологій. Викладання сучасних підходів у сфері захисту інформації є актуальною задачею університетської освіти. Іншим важливим моментом сучасних освітніх курсів є створення ефективних комп'ютерно орієнтованих програмних застосунків, що підвищують ефективність навчання.

Ефективним способом захисту інформації методом її приховування, зокрема і в медійних файлах, є стеганографія. На відміну від криптографії, що шифрує зміст повідомлення, стеганографія маскує сам факт передавання або збереження інформації. Одним з ефективних методів стеганографічного приховування інформації у графічних файлах є метод PVD (Pixel Value Differencing), який використовує різницю значень сусідніх пікселів зображення для вбудовування бітів повідомлення. Цей метод забезпечує добрий баланс між непомітністю для людського ока та ємністю вбудовування даних. Розтлумачення роботи подібних методів приховування інформації традиційними методами є недостатніми. Існує проблема легкої ілюстрації технологій, робота яких базується на маніпуляції з двійковими величинами, як на етапі лекційних занять, так і впродовж проведення відповідних лабораторних.

Актуальність методу PVD. Метод PVD, що застосовує зображення, як контейнери, привертає особливу увагу завдяки здатності забезпечувати високу ємність вбудовування інформації при збереженні візуальної якості зображення. Цей метод використовує малу чутливість людського зору до невеликих змін значень сусідніх пікселів. Це дає змогу більш ефективно використовувати області зображення з високою деталізацією.

Актуальність методу PVD зумовлена також такими факторами.

- Баланс між непомітністю та ємністю. На відміну від більш простих методів, таких як LSB (Least Significant Bit), PVD забезпечує краще співвідношення між обсягом прихованої інформації та якістю зображення.
- Стійкість до виявлення. PVD-метод ускладнює виявлення фактів приховування даних за допомогою простого статистичного аналізу, що підвищує його безпечність.
- Придатність до гібридизації. Метод може бути поєднаний з іншими підходами (наприклад, блоковою сегментацією або адаптивними алгоритмами), що дозволяє адаптувати його під конкретні задачі чи носії.

У зв'язку з цим дослідження, пов'язані з розробкою, оптимізацією та застосуванням PVD-методу у стеганографії, залишаються надзвичайно актуальними, як в прикладному сенсі, так і в навчальних цілях.

Актуальність методу PVD в прикладних задачах породжує необхідність розгляду цього методу у відповідних університетських курсах. Однак, при лекційному розгляді і, надалі, при використанні готового програмного забезпечення на лабораторних заняттях, утворюється деякий розрив між теоретичними знаннями і практичними навичками, які важливі при стеганоаналізі повідомлень. Цей розрив можна подолати використовуючи для процесу приховування інформації відповідний відкритий програмний код. Зрозуміло, що цей програмний код має бути максимально лаконічним і простим, а здобувачі освіти мають бути ознайомлені з основами програмування. У цих умовах стає доступним для роздруківки і відповідно для аналізу сам процес приховування інформації на рівні двійкових кодів тексту, що впроваджується, та двійкових кодів окремих сусідніх пікселів зображення, куди впроваджується повідомлення.

Літературний огляд реалізації методу PVD.

Метод Pixel Value Differencing (PVD) є однією з найвідоміших технік стеганографії, яка дозволяє ефективно приховувати інформацію в цифрових медіа. Його було запропоновано Wu і Tsai у 2003 році як поліпшену альтернативу класичним методам Least Significant Bit (LSB), що часто мали проблеми з візуальною помітністю змін у зображеннях [1]. З часу публікації початкової ідеї з'явилося багато вдосконалень методу. Adnan Abdul-Aziz Gutub (2008) [2] запропонував використання PVD у кольорових зображеннях, де дані приховуються в одному або кількох каналах RGB, збільшуючи стійкість до виявлення. У цій роботі вони пропонують алгоритм стеганографії для RGB-зображень із адаптивною кількістю біт, що вбудовуються в кожен колірний канал (R, G або B) залежно від інтенсивності пікселя. Thiyagarajan et al. (2011) [3] реалізували гібридний підхід, комбінуючи PVD із технікою LSB Matching, щоб забезпечити кращий компроміс між якістю і ємністю. C.M. Wang (2013) [4] розробив метод на основі PVD з використанням Histogram Shifting, щоб покращити зворотне вилучення і зменшити спотворення. У цій статті представлено двовимірну схему для зворотного (reversible) приховування даних, яка використовує зміщення гістограми (histogram shifting) на основі помилки прогнозування (prediction errors). Автори показують, що такий підхід забезпечує вищу пропускну здатність і менше спотворення в порівнянні з одновимірними методами, які базуються лише на PVD.

Реалізація навчальних технологій, що базуються на застосуванні лаконічного відкритого програмного коду, для більш ефективного ознайомлення з методами стеганографії і криптографії представлена в роботі [5].

Мета дослідження. Метою даного дослідження є розробка простого програмного застосунку, що може бути використаний в навчальних цілях у курсі криптографії та стеганографії, для стеганографічного приховування текстової інформації методом PVD в цифрових зображеннях.

Завдання: розробка простого програмного застосунку мовою програмування Python, в межах структурного і функціонального програмування, механізм роботи якого може бути зрозумілим здобувачам освіти, для яких програмування не є спеціальністю; розгляд роботи додатку у контексті застосування при проведенні занять з криптографії та стеганографії, підготовка завдань

для навчального середовища moodle за тематикою приховування текстової інформації методом PVD. Обговорення апробації програмного застосунку.

Результати дослідження. Процес роботи програми із впровадження повідомлення у зображення має наступні етапи. Відкриття вхідного зображення (порожній контейнер) та перетворення його на двомірний масив значень пікселів, що має три складових Red, Green, Blue. Визначення ширини і висоти зображення. Перетворення текстового повідомлення в бітовий масив. Визначення довжини бітового масиву.

Фрагмент програми, що реалізує ці дії можна вважати підготовчим в сенсі приховування тексту в електронному зображенні. Відповідний фрагмент представлений нижче в тексті коду програми 1.

```
from PIL import Image                                   # Імпорт бібліотеки Pillow
img = Image.open('cover.png').convert('RGB')            # Завантаження фотографії
# Завантаження тексту, що приховується
with open('TextIn.txt','r', encoding="utf-8") as f: secret_message=f.read()
width, height = img.size                                # Визначення ширини і висоти зображення
# Перетворення текстового повідомлення в бітовий масив
binary_message = ''.join(format(ord(char), '08b') for char in secret_message)
message_len = len(binary_message)                       # Визначення довжини бітового масиву
data_index = 0                                          # Довжина прихованої частини бітового масиву =0
```

Код програми 1. Фрагмент підготовчої частини програми приховування тексту в електронному зображенні

Безпосереднє приховування представлено в коді програми 2 і починається з сканування зображення за висотою і шириною. Це реалізується роботою двох циклів де один цикл вкладений в інший. Така алгоритмічна конструкція реалізує перебір пар пікселів зображення. У кожній парі сусідніх пікселів визначається різниця між червоними складовими кольору (diff). У процесі сканування зображення відбувається пошук пар пікселів для яких ця різниця була б така, щоб число поточних бітів тексту, що приховується (n_bits), не перевищувало трьох. Адже, $111_2=7_{10}$. Визначення такої пари пікселів реалізується розгалуженням if diff <= 7:.

Якщо між бітовими значеннями червоного кольору знайдена різниця, що може бути представлена трьома розрядами двійкового числа, вирізаємо відповідний бінарний шматок тексту довжиною три біти і впроваджуємо його у зображення, замінюючи ним реальну різницю в червоній складовій кольору двох сусідніх пікселів. Оскільки різниця в значеннях кольору між сусідніми пікселями може бути, як додатна так і від'ємна, існує два варіанти впровадження нової різниці в зображення.

Якщо значення червоності пікселів p1 < p2. Обчислюємо нові значення червоності пікселів так:

p1_new = sp-new_diff//2; p2_new=sp+(new_diff-new_diff//2).

Якщо значення червоності пікселів p1 > p2. Обчислюємо нові значення червоності сусідніх пікселів наступним чином:

p1_new = sp+(new_diff-new_diff//2) ; p2_new = sp-new_diff//2.

Необхідно відмітити, що кожна з трьох складових кольору пікселя, зокрема і червоного, зберігається в байтовій комірці пам'яті, тому різниця між значеннями кольору двох сусідніх пікселів може бути значно більше, ніж трьох бітове двійкове число. Тобто, впроваджувати текст можна було б і більшими до того ще й різновеликими шматками бітового тексту.

Однак, існує проблема, яка може ускладнити вилучення такого варіативно впровадженого тексту. Так, при впровадженні в різницю кольорів чергового бітового шматка тексту може виявитись, що черговий простір для букви має діапазон, скажімо, 6 бітів двійкового числа, а поточний шматок тексту, що впроваджується виглядає ось так: 000101_2.

Після впровадження такої різниці зазор між кольорами пікселів перетвориться з шести бітового в трьох бітовий. Це відіб'ється, з одного боку, на вилученні тексту, адже, замість чергового

шматка тексту в бітовому представленні 000101_2 ми вилучимо тільки 101_2. З іншого боку, трьох бітовий зазор в кольорах сусідніх пікселів відрізняється від шести бітового і тим більше, можливого восьмибітного зазору в значеннях кольору. Останнє може призвести до вад зображення.

```
for i in range(0, height - 1, 2):                    # Перебір рядків пікселів
        for j in range(0, width - 1, 2):             # Перебір пікселів в рядку
                if data_index < message_len:         # Перевірка чи не завершився текст в бітовому масиві
                        # Визначення міри червоності двох сусідніх пікселів p1, p2
                        p1, p2 = img.getpixel((j, i))[0], img.getpixel((j + 1, i))[0]
                        diff = abs(p1 - p2)          # Визначення різниці міри червоності двох сусідніх пікселів
                        if diff <= 7:                # Якщо різниця менше рівна 7 (3х розрядного  бітового числа)
                                # Вирізка  бінарного 3х розрядного шматку коду для впровадження
                                binary_chunk = binary_message[data_index:data_index + 3]
                                data_index +=3  # Збільшення довжини прихованої частини бітового масиву
                                # Визначення нової різниці червоності двох сусідніх пікселів в десятковому коді
                                new_diff = int(binary_chunk, 2)
                                sp=(p1+p2)//2   # Знаходження середнього значення червоності 2х пікселей
                                if p1 < p2:          # Якщо значення червоності пікселя p1 < p2
                                        # Знаходження нових значень кольорів двох сусідніх пікселів
                                        p1_new = sp-new_diff//2; p2_new = sp+(new_diff-new_diff//2)
                                else:                # Якщо значення червоності пікселя p1 >= p2
                                        # Знаходження нових значень кольорів двох сусідніх пікселів
                                        p1_new = sp+(new_diff-new_diff//2); p2_new = sp-new_diff//2
                                # Впровадження нових значень червоності пікселів в зображення
                                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.png')                          # Запис заповненого тестом графічного контейнеру в файл
print('Текст успішно приховано в графічному файлі ')    #  Друк повідомлення
```

Код програми 2. Фрагмент програми, в якій безпосередньо відбувається приховування тексту в електронному зображенні

Нижче представлена найбільш проста реалізація методу PVD, у якій текст закладається в графічний контейнер виключно тільки порціями рівними або меншими трьох бітового двійкового коду. Цей підхід, сприяє зменшенню помітності втручання. Так, якщо різниця в кольорі між двома пікселями була 101_2, а ми впровадили туди шматок тексту 111_2, то це утворить зміну кольору з 5_{10} одиниць до 7_{10}. Зрозуміло, що найбільша різниця в кольорах, при трьохбітовому впровадженні тексту в контейнер, буде тоді, коли в найбільший можливий проміжок між значеннями кольорів 111_2 (7_{10}) ми вставили шматок коду тексту 000_2 (0_{10}) або, коли в найменшу різницю між кольорами 000_2 (0_{10}) ми вставили шматок коду тексту 111_2 (7_{10}). На фоні можливої різниці значень кольорів від 00000000_2 (0_{10}) до 11111111_2 (256_{10}) це є не суттєвим і малопомітним.

Інше спрощення програми стосується кольорів. Закладка тексту відбувається тільки в один колір, червоний, хоча невеликим ускладнення коду можна досягнути впровадження і в інші два кольори (в зелений і блакитний). Зрозуміло, що це кратно підвищить ємність контейнера.

Аналіз роботи програми приховування тексту в картинці за технологією PVD

Проведемо деякі оцінки стосовно місткості графічних контейнерів, зокрема фотографій, для впровадження туди текстів методом PVD.

Фотографія HD формату 1280×720 містить 921 600 пікселів. Впровадження методом PVD відбувається в пари сусідніх пікселів. Тобто потенційно можна використати 460 800 бітових мікроконтейнерів. Однак, впровадження в нашій програмі відбувається тільки шматками тексту по три біти на фоні восьми можливих бітів у деяких випадках. Якщо, вважати, що третина пар пікселів підходить для трьох бітового впровадження, то можна бачити, що заявлений вище контейнер, має не менше 150 тисяч пар пікселів, здатних прийняти текст. Враховуючи, що кожна пара пікселів може прийняти тільки три біти тексту, видно, що згаданий графічний контейнер може містити не менше 50 кілобайт, а це кілька сторінок тексту в форматі txt. Приведені міркування були підтверджені випробуванням програми.

Стеганоаналіз медійних файлів на предмет впровадження в них секретного повідомлення передбачає розгляд цих файлів у двійковому форматі. Тому, з навчальної точки зору, цікавим є

розгляд динаміки впровадження тексту в бінарному представленні в графічний файл, що теж розглядається як сукупність пікселів у бітовому форматі.

Для ілюстрації того, як може виглядати на занятті зі стеганографії така динаміка впровадження тексту в графічний контейнер, розглянемо приховування в картинці короткого тексту «ACE». Знизу представлена таблиця, зроблена за роздруківкою роботи програми, що приховує інформацію в різницях значень червоного кольору сусідніх пікселів картинки. Приховування відбувається тільки в тих різницях значень пікселів, які не перевищують десяткове 7 або двійкове 111. Тобто в пікселях, які мають малу різницю значень червоного кольору і різниця вкладається в трьох розрядне двійкове число. Якщо різниця перевищує зазор в значеннях кольорів, змін не відбувається.

Перетворення тексту «ACE» в двійковий код дає наступне бінарне значення 010000010100001101000101 (нагадаємо, код A - 65 або 01000001, алф. A,B,C,D,E,F,…). В таблиці 1 приведена трансформована в таблицю роздруківка роботи програми. На екран виводяться: значення RED кольорів двох сусідніх пікселів; різниця значень двох сусідніх пікселів; бінарний фрагмент тексту, що впроваджується; нова різниця значень двох сусідніх пікселів; нові значення кольорів пікселів.

Таблиця 1. Похідна від роздруківки програми приховування тексту методом PVD

Значення кольору двох сусідніх пікселів	Різниця значень кольору сусідніх пікселів	Бінарний текст, що впроваджується	Нова різниця значень сусідніх пікселів	Нові значення кольорів сусідніх пікселів
24, 29	5	010	2	25, 27
51, 42	9	-	9	51, 42
40, 43	3	000	0	41, 41
52, 33	19	-	19	52, 33
33, 73	40	-	40	33, 73
70, 49	21	-	21	70, 49
51, 60	9	-	9	51, 60
87, 64	23	-	23	87, 64
62, 65	3	010	2	62, 64
97, 105	8	-	8	97, 105
77, 49	28	-	28	77, 49
35, 29	6	100	4	34, 30
28, 24	4	001	1	27, 26
20, 27	7	101	5	21, 26
16, 19	3	000	0	1, 17
8, 16	8	-	8	8, 16
21, 26	5	101	5	21, 26

З роздруківки видно, що впровадження бінарного коду відбувається не в усі поруч розташовані пікселі, а лише в ті, де різниця червоної складової менше або рівна 7. У випадку, якщо впровадження не відбувалось - виводяться прочерки. Видно, що програма працює коректно.

Аналіз роботи програми з вилучення тексту з картинки контейнера, що був впроваджений за технологією PVD

Вилучення з картинки тексту реалізується програмним кодом 3. Процес роботи програми із вилучення повідомлення із зображення має наступні етапи. Відкриття вхідного зображення, що є заповненим контейнером. Визначення ширини і висоти зображення. Перетворення його в двомірний масив значень пікселів. Сканування масиву на предмет пошуку пар пікселів, що мають зазор значень в три біти між червоними складовими кольору пікселів. Саме в цих зазорах знаходяться бінарні шматки тексту, який приховується. Далі, відбувається послідовна склейка цих шматків у цілісність. Це формує текст в бінарному представленні. Фіналізує роботу програми

перетворення цього тексту в бінарному представленні у звичайний байтовий формат. Останнє дає можливість прочитати повідомлення.

```
from PIL import Image                                  # Імпорт бібліотеки Pillow
img = Image.open('stego_image.png').convert('RGB') # Завантаження фотографії
width, height = img.size                               # Визначення ширини і висоти зображення
binary_message = ''                                    # Резервування строкової змінної для бінарного коду
for i in range(0, height - 1, 2):                      # Переключення рядків пікселів
    for j in range(0, width - 1, 2):                   # Переключення пікселів в рядку
        p1, p2 = img.getpixel((j, i))[0], img.getpixel((j + 1, i))[0]  # Визначення міри червоності двох сусідніх пікселів p1, p2
        diff = abs(p1 - p2)                            # Визначення різниці міри червоності двох сусідніх пікселів
        if diff <= 7:                                  # Якщо різниця менше рівна 7
            binary_chunk = bin(diff)[2:].zfill(3)      # Перетворення diff в двійкове три розрядне число
            binary_message += binary_chunk             # Формування тексту в бінарному форматі
decoded_message = ''                                   # Резервування строкової змінної для вилученого тексту
for i in range(0, len(binary_message), 8):             # Розбиття повідомлення в бінарному форматі на байтові фрагменти
    if i + 8 > len(binary_message): break              # Перевірка на завершення
    byte = binary_message[i:i + 8]                     # Вирізка чергового байта - коду букви
    if '#'!=chr(int(byte, 2)):                         # Перевірка на знак завершення повідомлення '#'
        decoded_message += chr(int(byte, 2))           # Підклейка чергової букви до повідомлення, що вилучається
    else: break                                        #  Вихід з циклу
print("Decoded message:", decoded_message)             #  Друк вилученого повідомлення
```

Код програми 3. Програма в якій відбувається вилучення тексту з електронного зображення

Таблиця 2 побудована за роздруківкою роботи програми, що вилучає текстову інформацію з різниць значень червоного кольору сусідніх пікселів картинки. Вилучення відбувається тільки з тих різниць значень пікселів, які не перевищують десяткове 7 або двійкове 111. Тобто вилучення відбувається тільки з тих пікселів, які мають малу різницю значень червоного кольору і ця різниця вкладається в трьох розрядне двійкове число. Якщо різниця перевищує зазор в значеннях кольорів, вилучення не відбувається.

Таблиця 2. Похідна від роздруківки програми вилучення тексту методом PVD

Значення RED кольорів двох сусідніх пікселів	Різниця значень сусідніх пікселів	Бінарний фрагмент тексту, що впроваджується
25, 27	2	010
51, 42	9	-
41, 41	0	000
52, 33	19	-
33, 73	40	-
70, 49	21	-
51, 60	9	-
87, 64	23	-
62, 64	2	010
97, 105	8	-
77, 49	28	-
34, 30	4	100
27, 26	1	001
21, 26	5	101
1, 17	0	000
8, 16	8	-
21, 26	5	101

У випадку, якщо вилучення не відбувалось виводяться прочерки «-». Вино, що програма працює коректно.

Апробація. Проведена практична апробація реалізованої програми, як на лекційних, так і на лабораторних заняттях з криптографії у Волинському національному університеті імені Лесі Українки. Також, ця програма застосовувалась для генерації завдань в середовищі moodle.

На лекційних заняттях програма використовувалась для демонстрації роботи методу приховування текстової інформації за методом PVD. Програма в реальному часі приховувала текстову інформацію та роздруковувала значення кольорів окремих пікселів графічного зображення, куди відбувалось впровадження тексту, та чергових бітів тексту, що впроваджувались.

Код програми використовувався для швидкої генерації великої кількості заготовок для завдань в середовищі moodle. Один з таких шаблонів завдання представлений на рис.1.

Приховування в картинці тексту «ACE» по технології PVD

1. Перетворити текст «ACE» в двійковий код.

2. Впровадити текст в картинку

Значення RED кольорів двох сусідніх пікселів	Різниця значень кольорів	Бінарний фрагмент тексту, що приховується	Нова різниця значень кольорів	Нові значення кольорів сусідніх піксклів
24,29	5	010	2	25,27
51,42	9	-	9	51,42
40,43				
52,33				
33,73				
70,49				
51,60				

Рис.1. Приклад фрагменту завдання в середовищі moodle.

Зазначимо, що між лекційними і лабораторними завданнями в умовах експрес генерації протоколу приховування тексту, може бути безпосередній зв'язок. Лекційна задача і завдання наступної лабораторної дуже важливий для активізації діяльності на лекції в умовах, коли відомо, що лекційна задача продукує завдання лабораторної.

Відкритий програмний код використовувався для закріплення і усвідомлення студентами механізму приховування. Для цього автори роботи використовували свій оригінальний застосунок, який виводив ключову частину логічно завершеного готового робочого коду програми, представленого у статті, разом з відповідними ремарками в інше вікно. Там цей код і ремарки цілісними рядками перемішувався. Деякі лаконічні ключові фрагменти коду, могли вилучатись повністю. Завдання студентам полягало у відновленні роботи програми приховування в цьому конструкторі правильно розташувавши рядки і надрукувавши відповідний короткий зв'язуючий код. Далі, автоматизована перевірка коду і його виконання.

Висновки та перспективи подальших досліджень

У роботі представлено відкритий програмний код для стеганографічного приховування в цифрових зображеннях текстової інформації методом PVD. Представлено також і код відповідної програми для вилучення тексту з цифрових зображень, що використовуються як контейнери.

Проведена апробація цього коду для використання його в навчальних цілях в курсі криптографії та стеганографії. На думку авторів, можливість приховувати тексти і при цьому бачити цей процес на рівні перетворень двійкових кодів, дозволяє краще підготуватись до стеганоаналізу файлів.

Програмний код успішно використовувався для швидкої генерації великої кількості завдань за темою стеганографічного приховування текстової інформації методом PVD для середовища moodle.

Важливою особливістю методу PVD є те, що він легко трансформується для приховування інформації в звукових файлах. У випадку приховування інформації в звукових файлах, можна, так само, спостерігати на рівні двійкових кодів, як текст в бінарному форматі впроваджується в оцифровані миттєві значення звукової хвилі. Автори роботи найближчим часом планують провести дослідження на предмет можливості застосування подібних програмних застосунків, що працюють зі звуком, для курсів стеганографії та криптографії.

Список бібліографічного опису

1. Da Chun Wu, Wen Hsiang Tsai. A steganographic method for images by pixel value differencing. Pattern Recognition Letters, Volume 24, Issue 9–10 (June 2003), P. 1613–1626. DOI: 10.1016/S0167-8655(02)00402-6.
2. Mohammad Tanvir Parvez та Adnan Abdul Aziz Gutub. RGB Intensity Based Variable Bits Image Steganography. Proceedings of the 3rd IEEE Asia Pacific Services Computing Conference, Yilan, Taiwan. P. 1322-1327, 2008 DOI:10.1109/APSCC.2008.105
3. Azzat A. Al-Sadi & El Sayed M. El-Alfy. Thiyagarajan et al. An Adaptive Steganographic Method for Color Images Based on LSB Substitution and Pixel Value Differencing. Advances in Computing and Communications (ACC) 2011. P. 535–544 https://doi.org/10.1007/978-3-642-22714-1_55
4. Chun Ming Wang et al. Reversible data hiding based on two dimensional prediction errors. IET Image Processing, Volume 7, Issue 9, 2013, P. 805 816 https://doi.org/10.1049/iet-ipr.2012.0521
5. Mykola Holovin, Nina Holovina Educational example of masking textual information in a photographic signal. Journal «ScienceRise: Pedagogical Education» No4 (49) 2022. P. 24-28 http://journals.uran.ua/sr_edu/article/view/261051/258566

References

1. Da Chun Wu, Wen Hsiang Tsai. A steganographic method for images by pixel value differencing. Pattern Recognition Letters, Volume 24, Issue 9–10 (June 2003), P. 1613–1626. DOI: 10.1016/S0167-8655(02)00402-6.
2. Mohammad Tanvir Parvez та Adnan Abdul Aziz Gutub. RGB Intensity Based Variable Bits Image Steganography. Proceedings of the 3rd IEEE Asia Pacific Services Computing Conference, Yilan, Taiwan. P. 1322-1327, 2008 DOI:10.1109/APSCC.2008.105
3. Azzat A. Al-Sadi & El Sayed M. El-Alfy. Thiyagarajan et al. An Adaptive Steganographic Method for Color Images Based on LSB Substitution and Pixel Value Differencing. Advances in Computing and Communications (ACC) 2011. P. 535–544 https://doi.org/10.1007/978-3-642-22714-1_55
4. Chun Ming Wang et al. Reversible data hiding based on two dimensional prediction errors. IET Image Processing, Volume 7, Issue 9, 2013, P. 805 816 https://doi.org/10.1049/iet-ipr.2012.0521
5. Mykola Holovin, Nina Holovina Educational example of masking textual information in a photographic signal. Journal «ScienceRise: Pedagogical Education» No4 (49) 2022. P. 24-28 http://journals.uran.ua/sr_edu/article/view/261051/258566

Історія статті:
Отримано: 10.05.2026 Доопрацьовано: 10.05.2026 Прийнято до друку: 23.05.2026 Опубліковано: 29.05.2026