Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ДОПУСТИТИ ДО ЗАХИСТУ:
В.о. завідувача кафедри
кібербезпеки та захисту інформації
_____ Іван ПАРХОМЕНКО
«____» червня 2023 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи

галузь знань 12 Інформаційні технології
(шифр і назва галузі знань)
спеціальність 125 Кібербезпека
(код і назва спеціальності)
освітній ступень бакалавр
(назва освітньої програми)
освітня програма Кібербезпека
(назва освітньо-професійної програми)

на тему: «Засоби приховування інформації в графічних та мультимедійних об'єктах з використанням стеганографічних і криптографічних технологій»

Виконавець: студент IV курсу, групи КБ-42

_____ Олександр ШИНКАРЕНКО
(підпис) (ім'я прізвище)

	Прізвище, ініціали	**Підпис**
Керівник роботи	Лариса МИРУТЕНКО	

Нормоконтроль	Андрій ФЕСЕНКО	

Київ 2023

Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ЗАТВЕРДЖЕНО:
В.о. завідувача кафедри кібербезпеки
та захисту інформації
_____ Сергій ТОЛЮПА
«24» жовтня 2022 р.

ЗАВДАННЯ
на виконання дипломної роботи

спеціальност 125 Кібербезпека
(код і назва спеціальності)

освітньої програми Кібербезпека
(назва освітньої програми)

Студенту **КБ-42** **Шинкаренку Олександру Михайловичу**
(група) (прізвище ім'я по-батькові)

Тема дипломної роботи Засоби приховування інформації в графічних та мультимедійних об'єктах з використанням стеганографічних і криптографічних технологій

1. ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Тема кваліфікаційної роботи затверджена на засіданні кафедри кібербезпеки та захисту інформації протокол №3 від 20.10.2022 р.

2. ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Структури, архітектури, засоби приховування інформаціяї в графічних та мультимедійних об'єктах, алгоритми стеганографічних і криптографічних криптографічних технологій

3. ЗМІСТ РОЗРАХУНКОВО-ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ

Стеганографія й стеганографічні системи, моделі комп'ютерної стеганографії для графічних і мультимедійних об'єктів, Форматні й Неформатні методи приховування інормації, методи приховування інформації в частотній і просторовій області, алгоритми LSB і JSteg, програмний застосунок

4. ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Практична цінність Програмна реалізація застосунку для приховування даних в графічних об’єктах на основі алгоритму LSB

5. ДАТА ВИДАЧІ ЗАВДАННЯ

Дата видачі завдання: 24 жовтня 2022 року

Завдання видала _____ (підпис) Лариса МИРУТЕНКО (ім’я, прізвище)

Завдання прийняв до виконання _____ (підпис) Олександр ШИНКАРЕНКО (ім’я, прізвище)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Найменування етапів робіт	Строки виконання робіт (початок-кінець)	Відмітка про виконання
1	Уточнення постановки задачі	24.10.2022 – 22.01.2023	*виконано*
2	Аналіз відкритих джерел	29.01.2023 – 11.02.2023	*виконано*
3	Обґрунтування вибору рішення	12.02.2023 – 15.02.2023	*виконано*
4	Дослідження концепцІЇ стегосистеми	16.02.2023 – 04.03.2023	*виконано*
5	Аналіз проблем інформаційної безпеки в стеганографічних системах	05.03.2023 – 21.03.2023	*виконано*
6	Дослідження особливостей і вразливостей алгоритму LSB	22.03.2023 – 08.04.2023	*виконано*
7	Практична реалізація програми на основі вибраного алгоритму	09.04.2023 – 10.05.2023	*виконано*
8	Оформлення пояснювальної записки	11.05.2023 – 27.05.2023	*виконано*
9	Підготовка до захисту кваліфікаційної роботи	28.05.2023 – 12.06.2023	*виконано*

Завдання видала _____ (підпис) Лариса МИРУТЕНКО (ім’я, прізвище)

Завдання прийняв до виконання _____ (підпис) Олександр ШИНКАРЕНКО (ім’я, прізвище)

Термін подання дипломної роботи до ЕК 12 червня 2023 року

РЕФЕРАТ

Кваліфікаційна робота складається зі вступу, трьох розділів, загальних висновків, списку використаних джерел, додатків, має 56 сторінки основного тексту, 26 рисунків та 4 формули. Список використаних джерел містить 21 найменування і займає 2 сторінки. Крім того, робота містить 2 додатки із загальною кількістю сторінок 10.

Метою роботи є створення застосунку для приховування інформаціїї в графічних об’єктах з використанням стеганографічних і криптографічних технологій.

Об'єктом дослідження є процес захисту інформації в графічних об’єктах.

Предметом дослідження є моделі та методи комп’ютерної стеганографії.

Методи дослідження кваліфікаційної роботи:

- аналіз;
- опис;
- порівняння.

В даній роботі проаналізовані вітчизняні та іноземні джерела з теорії стеганографічних систем, виконаний аналіз документів, порівняння, вивчення та узагальнення вітчизняної і зарубіжної практики з теми стеганографічні технології, розроблено рекомендації з вибору моделі стеганографії.

Практична цінність: розроблено застосунок, який виконує приховування інформації в фото на основі алгоритму LSB та може використовуватися користувачами для наочного представлення шифрування даних в графічних об’єктах.

Ключові слова: кібербезпека, стеганографічні і криптографічні технології, стегоаналіз, алгоритм LSB,приховування інформації.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ОЗНАЧЕНЬ ТА СКОРОЧЕННЬ

НЗБ	–	Найменш Значущий Біт
НЗБ	–	Дискретне Косинусне Перетворення
JPEG	–	Joint Photographic Experts Group
LSB	–	Least Significant Bit
КС	–	Комп’ютерна Стеганографія
ЦВДЗн	–	Цифровий Водяний Знак
FFT	–	Швидке Перетворення Фур'є
БМЕЮ	–	Метод Бенгама-Мемон-Ео-Юнга

ВСТУП

Протягом усієї історії людства проблема інформаційної безпеки була актуальною і для її вирішення виникли два основних напрямки: криптографія та стеганографія. Криптографія забезпечує захист даних шляхом їх шифрування та блокування несанкціонованого доступу. З іншого боку, стеганографія метою якої є приховування інформації в публічному носії з метою забезпечення конфіденційності та захисту цілісності даних. Цей термін походить від грецького слова "стеганос", що означає "прихований", і "графія", що означає "писати" або "записувати".

Ідея стеганографії полягає в тому, щоб внести секретну інформацію в беззаперечний носій, такий як зображення, звуковий файл або текстовий документ, таким чином, щоб ця інформація залишалася прихованою від сторонніх спостерігачів. Основна відмінність стеганографії від криптографії полягає в тому, що вона не лише шифрує інформацію, але й маскує її наявність, що робить процес приховування більш ефективним.

Стеганографія має широке застосування в різних сферах, включаючи комунікації, безпеку даних, криміналістику та військову розвідку. Вона може бути використана для передачі конфіденційної інформації, забезпечення анонімності, а також для виявлення та вивчення прихованих даних.

Завдяки постійному розвитку технологій, стеганографія продовжує еволюціонувати, стаючи все більш складною і ефективною. Вона є важливим інструментом для забезпечення безпеки та захисту інформації в сучасному цифровому світі.

Сучасний розвиток стеганографії базується на широкому спектрі наукових дисциплін і методологій. Для досягнення прогресу в цій галузі використовуються методи теорії ймовірностей та математичної статистики, теорії швидких ортогональних перетворень, теорії апроксимації, теорії кодування, теорії складності, теорії похибок, цифрової обробки сигналів та зображень, і багато інших.

Це свідчить про те, що стеганографія є науковою та технологічно складною дисципліною. Незважаючи на молодість комп'ютерної стеганографії, її основні

принципи та поняття відрізняються від традиційної стеганографії, яка використовувала фізичні методи приховування інформації. Так у роботах [1-5] були представлені основні визначення та математичні моделі стеганографічних систем.

Серед видатних вчених, які внесли значний вклад у розвиток стеганофонії та стеганографії, можна виділити наступних дослідників:: Задірака В.К., Кошкіна Н.В., Олексюк О.С., а також польські вчені Wojciech Mazurczyk, Krzysztof Szczypiorski, Zbigniew Kotulski.

Тому метою роботи є створення застосунку для приховування інформаціїї в графічних об’єктах з використанням стеганографічних і криптографічних технологій.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- Розглянути засоби приховування інформації з використанням стеганографічних технологій.
- Дослідити проблематику використання різних методів стеганографії.
- Розглянути існуючі алгоритми втілення методів стеганографії.
- Створити на основі вибраного алгоритму застосунок для приховування інформації графічних об’єктах з використанням стеганографічних і криптографічних технологій.

Об’єктом дослідження в даній роботі є процес приховування інформації в графічний та мультимедійний об’єкт.

Предметом дослідження є методи та алгоритми комп’ютерної стеганографії для зображень.

Методи дослідження кваліфікаційної роботи бакалавра:

- аналіз вітчизняних та іноземних джерел;
- опис методів приховування інформаціїї в графічних об’єктах з використанням стеганографічних і криптографічних технологій;
- порівняння описаних методів.

РОЗДІЛ 1.
АНАЛІЗ СТЕГАНОГРАФІЇ І СТЕГАНОГРАФІЧНОЇ СИСТЕМИ

1.1 Етапи розвитку та формування поняття "Стеганографія"

Розвиток та формування поняття "Стеганографія" має свої етапи, які охоплюють історію використання цієї техніки для приховування інформації. Основні етапи розвитку стеганографії включають наступне:

- Давні часи: стеганографія має дуже давнє походження і використовувалася ще у давніх цивілізаціях, таких як Стародавній Єгипет та Стародавня Греція. Техніки стеганографії того часу включали використання прихованих повідомлень на папірусі або проникнення інформації до зображень шляхом нанесення невидимих чорнил.
- Середньовіччя: у середньовіччі стеганографія продовжувала розвиватися. Один із прикладів використання стеганографії був описаний Джованні Батіста Портіа у 1563 році у його книзі "De Furtivis Literarum Notis". Він описував методи приховування інформації за допомогою різних шифрів та схем, включаючи використання нотних значків та багатомовних текстів.
- Період двох світових воєн: у цей період стеганографія отримала значний розквіт, оскільки шпигуни використовували її для передачі та отримання секретної інформації. Під час Другої світової війни стеганографія широко використовувалася різними розвідувальними агентствами для приховування секретної інформації у повідомленнях, зображеннях, радіопередачах тощо.
- Розвиток комп'ютерної стеганографії: з появою комп'ютерів і цифрових зображень стеганографія перейшла в нову еру. У 1990-х роках з'явилися нові методи та алгоритми для приховування інформації у цифрових медіа, таких як зображення, аудіо та відео. Ці методи використовують особливості цифрових даних та психовізуальні аномалії для вбудовування прихованої інформації без помітних змін у носіях.

- Сучасність: стеганографія залишається актуальною й поширеною технологією в сучасному світі. Вона знайшла застосування в сферах кібербезпеки, захисту даних, цифрового медіа, телекомунікацій, а також у правоохоронних та розвідувальних сферах.

На сьогоднішній день стеганографія продовжує розвиватися, з'являються нові методи та алгоритми, які дозволяють більш ефективно та безпечно приховувати інформацію у різних типах носіїв даних.

1.2 Стенографія: поняття, особливості, сутність

Стеганографія - це методика приховування інформації шляхом вбудовування її в інший носій, такий як зображення, аудіофайл, текстовий документ тощо, з метою збереження конфіденційності і недоступності для сторонніх спостерігачів. У відмінність від криптографії, яка займається шифруванням повідомлень, стеганографія спрямована на забезпечення тахтовості самого факту наявності прихованої інформації.

Основна мета стеганографії полягає в тому, щоб зробити приховану інформацію якомога менш помітною для непрофесійних спостерігачів. Інформація може бути вбудована у носій даних шляхом зміни певних параметрів або властивостей носія, таких як малий рівень шуму у зображеннях, слабка зміна акустичних характеристик у аудіофайлах або використання непомітних модифікацій текстових документів.

Стеганографія може бути застосована в різних сферах, включаючи кібербезпеку, розвідку, антивірусні системи, захист авторських прав, цифрове медіа та інші. Вона використовується як додатковий інструмент до криптографії, а не як її заміна. Використання стеганографії для приховування повідомлення значно зменшує ймовірність виявлення самого факту передачі цього повідомлення. Якщо до цього повідомлення також застосовується шифрування, то це надає ще один додатковий рівень захисту [6-8].

У зв'язку з швидким розвитком обчислювальної техніки та нових каналів передачі інформації, стеганографія отримала нові методи, які базуються на

особливостях представлення інформації в комп'ютерних файлах, мережах тощо. Це призвело до зародження нового напрямку - комп'ютерної стеганографії.

Комп'ютерна стеганографія - це галузь стеганографії, яка використовує комп'ютерні технології для приховування інформації у цифрових медіафайлах. Вона використовує можливості комп'ютерних систем для вбудовування додаткової інформації у файли різних форматів, таких як зображення (наприклад, JPEG, BMP), аудіофайли (наприклад, MP3, WAV) або відеофайли (наприклад, AVI, MPEG).

Комп'ютерна стеганографія використовує різні методики та алгоритми для вбудовування інформації у носії. Одним з поширених методів є метод найменших значущих бітів (LSB), при якому інформація вбудовується шляхом зміни найменш значущих бітів пікселів у зображенні або аудіосемплів у звуковому файлі.

Сучасна комп'ютерна стеганографія використовує два основних типи файлів: повідомлення-файл і контейнер-файл.

Повідомлення-файл є файлом, який призначений для приховування конфіденційної інформації. Це може бути текстовий файл, зображення, аудіо-або відеофайл.

Контейнер-файл, у свою чергу, використовується для занесення прихованого повідомлення. Це може бути звичайний файл будь-якого формату, який не містить прихованої інформації. Контейнер-файл слугує основою, в яку вбудовується повідомлення, і його вміст змінюється таким чином, щоб прихована інформація стала непомітною для зовнішнього спостерігача.

Контейнери поділяються на два варіанти. Перший - це контейнер-оригінал, який не містить прихованої інформації і є початковим файлом. Другий - це контейнер-результат, який містить приховане повідомлення після проведення процесу вбудовування.

Для забезпечення безпеки і доступу до прихованої інформації використовується ключ. Ключ є секретним елементом, який визначає правила та порядок занесення повідомлення в контейнер. Це дозволяє лише особам, які мають відповідний ключ, виконувати процес вилучення прихованої інформації з контейнера. Основні положення сучасної комп'ютерної стеганографії [9- 13]:

- Збереження автентичності: методи стеганографії повинні забезпечувати, що після внесення прихованого повідомлення файл залишається недоторканим і не втрачає своїх основних властивостей. Іншими словами, стеганографічні зміни повинні бути непомітними для сторонніх спостерігачів.
- Відомість методів: припускається, що потенційному противнику відомі можливі методи стеганографії. Це вимагає використання більш складних і тонких технік, які можуть ускладнити виявлення прихованої інформації.
- Безпека та ключ: безпека методів стеганографії ґрунтується на збереженні основних властивостей вихідного файлу після внесення секретного повідомлення і використанні невідомого противнику ключа. Тільки особи з правильним ключем можуть вилучити приховану інформацію.
- Складність відновлення: навіть якщо факт приховування повідомлення став відомим противнику через спільника, отримання самого секретного повідомлення вимагатиме складних обчислень або аналізу, що робить процес відновлення більш складним і часо- та ресурсозатратним.

Стеганографія містить у собі такі напрями:

- Вбудовування інформації з метою прихованої передачі.
- Використання цифрових водяних знаків (ЦВДЗн) для вбудовування ідентифікаційних слідів.
- Вбудовування ідентифікаційних номерів для відстеження та ідентифікації вмісту (fingerprinting).
- Вбудовування заголовків для позначення та опису зображень (captioning).

В процесі приховування великого обсягу даних в контейнері, ставиться серйозна вимога щодо розміру контейнера. Розмір контейнера повинен бути значно більшим за розмір вбудованих даних, зазвичай в кілька разів більшим.

Цифрові водяні знаки використовуються для захисту авторських або майнових прав на цифрові зображення, фотографії або інші цифрові твори мистецтва. При використанні цифрових водяних знаків, важливими вимогами є надійність і стійкість до спотворень вбудованих даних.

Впроваджувані заголовки мають обмежений обсяг і їм ставляться низькі вимоги. Заголовки повинні внести мінімальні спотворення і бути стійкими до основних геометричних перетворень.

Наразі найпоширенішим методом стеганографічного приховування є метод заміни найменш значущих бітів. Цей метод використовується для растрових зображень, зокрема у форматі BMP, який є популярним контейнером через його високу якість та простоту. Ідея полягає в заміні молодших бітів пікселів зображення на біти повідомлення, що приховуються. Це можливо завдяки наявності структурної надлишковості в зображеннях.

Інший метод стеганографічного перетворення використовується для стиснутих файлів з втратами даних, зокрема для графічного формату JPEG. У цьому методі інформація приховується не в значеннях окремих пікселів, а в бітах квантованих дискретних косинусних коефіцієнтів. Файли у форматі JPEG забезпечують можливість приховування значних обсягів інформації з точки зору стеганографії.

1.3 Стеганографічна система або стегосистеми

Стеганографічна система, також відома як стегосистема, є комплексом методів, протоколів та алгоритмів, які застосовуються для приховування та передачі конфіденційної інформації у недоступний для спостереження спосіб. Стегосистеми мають на меті забезпечити конфіденційність та захист інформації від небажаних сторонніх осіб. При розробці стегосистеми необхідно враховувати наступні принципи:

Противник повністю знайомий зі стеганографічною системою і її реалізацією. Єдиною невідомою для потенційного супротивника залишається ключ, який використовується для розпізнавання і витягування прихованого повідомлення. Тільки власник ключа може встановити факт наявності і зміст прихованої інформації.

Якщо противник дізнається про наявність прихованого повідомлення, це не повинно дозволити йому отримати подібні повідомлення з інших даних, поки ключ залишається секретним.

Потенційний противник повинен бути позбавлений будь-яких технічних або інших переваг в розпізнаванні або розкритті змісту таємних повідомлень.

Такі перепрограмовані речення можуть забезпечити ефективну та надійну стеганографічну комунікацію, зберігаючи важливі аспекти конфіденційності та безпеки.

Узагальнена модель стегосистеми представлена на рисунку 1.1.

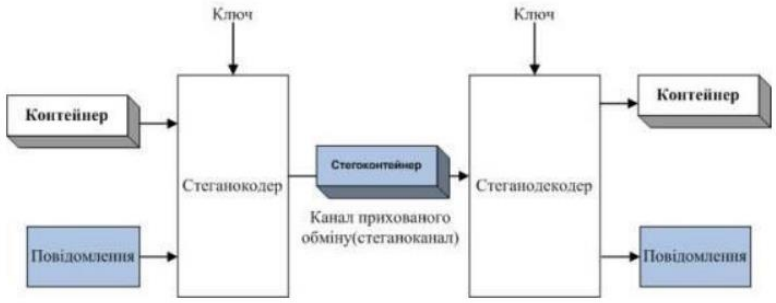

Рисунок 1.1 – Узагальнена модель стегосистеми

Стеганографічна система складається з наступних компонентів:

Повідомлення: Це конфіденційна інформація, яку необхідно приховати в носії (контейнері).

Контейнер: Це файл або носій інформації, який використовується для занесення прихованого повідомлення. Контейнер може бути будь-яким типом файлу, наприклад, зображенням, звуковим файлом, відео тощо.

Алгоритми стеганографії: Це методи та процедури, які використовуються для приховування повідомлення в контейнері. Алгоритми стеганографії можуть використовувати різні методи, такі як зміна найменших значущих бітів, частотна модуляція, перетворення дискретного косинусного, перекодування і т. д.

Алгоритми дешифрування: Це методи та процедури, які використовуються для вилучення прихованого повідомлення з контейнера за допомогою правильного ключа.

Стеганографічний канал або стеганоканал: Це канал передачі стеганоконтейнера, де приховане повідомлення може бути передане без спричинення підозр.

Стеганоключ або просто ключ - це секретний ключ, що використовується для приховування інформації. В стеганосистемі може бути один або кілька стеганоключів, залежно від рівня захисту (наприклад, вбудовування попередньо зашифрованого повідомлення).

В залежності від типу ключа, стегосистеми можна розділити на дві основні категорії: стегосистеми з секретним ключем і стегосистеми з відкритим ключем.

У стегосистемах з секретним ключем використовується один спільний ключ, який повинен бути заздалегідь відомий і переданий між спільниками, що займаються обміном повідомленнями. Цей ключ використовується для вбудовування та вилучення прихованого повідомлення.

У стегосистемах з відкритим ключем використовуються два різних ключі: публічний ключ і приватний ключ. Публічний ключ може бути переданий відкритим способом, тоді як приватний ключ залишається секретним. Вбудовування та вилучення повідомлення здійснюються за допомогою цих двох ключів. Ця схема дозволяє ефективно працювати навіть у випадку взаємної недовіри між сторонами, оскільки приватний ключ залишається секретним і не може бути виведений з публічного ключа.

Стеганосистеми повинні задовольняти такі основні чотири вимоги[14-17]:

1. Непомітність вбудовування: Властивості контейнера повинні бути змінені таким чином, що зміни не можна виявити візуально. Це забезпечує безперешкодний процес передачі стеганоповідомлення через канал зв'язку, не привертаючи увагу атакуючого.

2. Робастність до спотворень: Стеганоповідомлення повинно бути стійким до різних трансформацій, які можуть відбуватися з контейнером під час передачі, таких як зміна розмірів, зміна формату, стиснення даних та інше.

3. Забезпечення цілісності: Використання коду з виправленням помилок допомагає зберегти цілісність вбудованого повідомлення, щоб уникнути пошкоджень під час передачі.

4. Надійність через дублювання: Для підвищення надійності вбудованого повідомлення, його можна продублювати, тобто вбудувати декілька копій

повідомлення, що забезпечує можливість відновлення повідомлення при його частковій втраті.

Отже, стеганосистеми повинні забезпечувати непомітність, робастність до спотворень, цілісність повідомлення та надійність через дублювання для успішного приховування і передачі повідомлень.

Більшість сучасних методів приховування повідомлення в цифрових контейнерах показують залежність надійності системи від обсягу вбудованих даних, що можна побачити на рисунку 1.2.

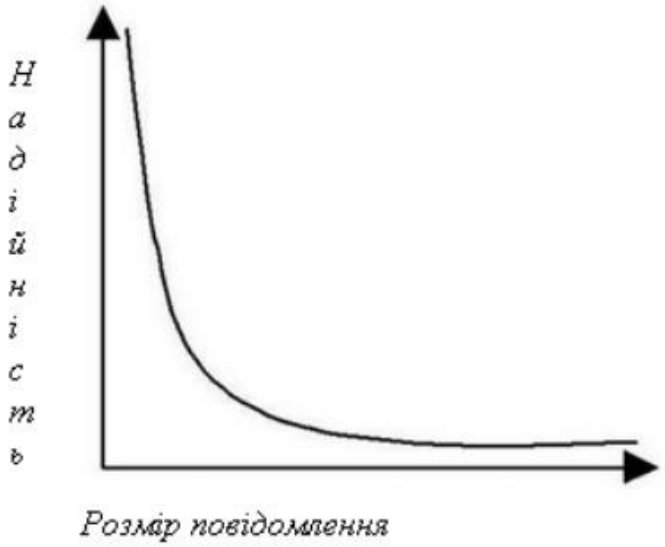

Рисунок 1.2 – Залежність надійності системи від обсягу вбудованих даних

Представлена залежність показує, що збільшення обсягу вбудованих даних при незмінному розмірі контейнера призводить до зниження надійності системи. Отже, контейнер, що використовується в стеганосистемі, накладає обмеження на розмір вбудованих даних.

1.4 Контейнери

Вибір контейнера має значний вплив на надійність стеганосистеми та можливість виявлення прихованого повідомлення. Наприклад, навіть непрофесійному спостерігачу, особливо з художньою освітою, може бути легко помітна зміна колірної гами, якщо повідомлення вбудовано в репродукцію таких відомих картин, як "Мадонна" Рафаеля або "Чорний квадрат" Малевича. Такі

візуальні зміни можуть викликати підозри і спричинити виявлення прихованого повідомлення. Тому вибір контейнера повинен бути обдуманим і враховувати його властивості, щоб забезпечити високу стійкість та непомітність стеганографічного вбудовування.

За протяжністю, контейнери в стеганографії можна розділити на два типи: фіксованої довжини і змінної довжини.

Контейнери змінної довжини, навпаки, мають змінний розмір, що дозволяє адаптувати їх до розміру прихованого повідомлення. У цьому випадку, контейнер може бути збільшений або скорочений для вміщення потрібного обсягу інформації. Контейнери змінної довжини забезпечують більшу гнучкість при вбудовуванні повідомлень різних розмірів, але можуть бути більш вразливі до виявлення атаками стеганалізу.

Контейнери фіксованої довжини мають сталу кількість байтів або бітів, які можуть бути використані для вбудовування прихованого повідомлення. Це означає, що розмір контейнера залишається незмінним незалежно від розміру прихованого повідомлення. При використанні контейнерів фіксованої довжини, може виникнути проблема недостатньої місткості для вбудовування великого повідомлення. Інший недолік контейнерів фіксованої довжини полягає в тому, що відстані між бітами, які використовуються для приховування інформації, розподіляються рівномірно. Однак справжній випадковий шум має експоненціальний розподіл довжини між бітами.

Це означає, що у контейнерах фіксованої довжини відстані між бітами можуть бути прогнозовані або виділені шляхом статистичного аналізу. У порівнянні з експоненціальним розподілом, рівномірний розподіл відстаней може викликати підозру і вказувати на наявність стеганографічного приховування. Отже, цей недолік може вплинути на безпеку стеганографічної системи, оскільки може збільшити шанси успішного виявлення інформації, яку намагаються приховати.

Проте на практиці найбільш поширеними і доступними є контейнери фіксованої довжини. Варіанти таких контейнерів можуть бути наступними:

- Контейнер, що генерується самою стеганосистемою, наприклад, за допомогою програми MandelSteg, де використовується фрактал Мандельброта як

контейнер для вбудовування повідомлення. Цей підхід можна назвати конструюючою стеганографією.

- Контейнер обирається зі значною кількістю доступних варіантів контейнерів. Велика кількість альтернативних контейнерів генерується, а потім обирається найбільш підходящий для приховування повідомлення. Цей підхід можна назвати селективною стеганографією. При виборі оптимального контейнера найважливішим критерієм є природність контейнера. Однак, навіть найкращий контейнер має обмежену потужність для приховування даних при великому розмірі самого контейнера.
- Контейнер надходить ззовні, і в цьому випадку немає можливості вибору контейнера. Використовується перший підходящий контейнер, який не завжди ідеально підходить для вбудованого повідомлення. Цей підхід можна назвати безальтернативною стеганографією.

Висновки за розділом 1

У першому розділі дипломної роботи досліджуються теоретичні аспекти стеганографії, включаючи її визначення, особливості, сферу застосування та сутність. Було проведено огляд еволюції поняття стеганографії та досліджуються основні складові компоненти методів комп'ютерної стеганографії. Окрему увагу приділяється дослідженню різних типів стеганографічних систем і їх складових, таких як стеганографічний канал, стеганографічний ключ. Також була наведена узагальнена модель стегосистеми. Крім того, досліджується поняття контейнера, наводяться різновиди контейнерів і розглядається їх вплив на надійність стеганосистеми.

РОЗДІЛ 2
МОДЕЛІ ТА МЕТОДИ КОМП'ЮТЕРНОЇ СТЕГАНОГРАФІЇ ДЛЯ ГРАФІЧНИХ І МУЛЬТИМЕДІЙНИХ ОБ'ЄКТІВ.

2.1 Методи приховування інформації для графічних об’єктів

Методи приховування даних можна класифікувати на форматні і неформатні , залежно від принципів, що лежать в їх основі.

Форматні методи стеганографії використовують можливості внесення змін у формати файлів, такі як зображення (наприклад, BMP або JPEG), звукові файли (наприклад, WAV або MP3) або відео (наприклад, AVI або MPEG). Ці методи використовують особливості структури файлу і його кодування для приховування додаткової інформації. Наприклад, внесення змін у менш значущі біти зображення або використання шуму в звукових доріжках для занесення прихованого повідомлення.

Неформатні методи стеганографії не залежать від конкретного формату файлу і можуть застосовуватися до будь-якого типу даних. Ці методи зазвичай базуються на аналізі статистичних властивостей даних, таких як розподіл символів або функції автокореляції. Приховане повідомлення може бути вбудоване в даний потік за допомогою внесення мінімальних змін, щоб зберегти статистичну незмінність даних.

Вибір між форматними і неформатними методами залежить від конкретного контексту і вимог стеганографічної системи. Кожен з цих підходів має свої переваги і обмеження, і їх використання залежить від типу даних, які потрібно приховати, і вимог до безпеки та незрозумілості приховуваної інформації.

2.1.1 Форматні методи приховування інформації для графічних об’єктів

Метод приховування в палітрі (palette-based steganography) є одним з форматних методів стеганографії, який використовується для приховування даних в зображеннях з обмеженою палітрою кольорів.

У зображеннях з обмеженою палітрою кольорів, кожен піксель представлений відповідним індексом або кодом кольору, що вказує на значення з палітри. Метод приховування в палітрі використовує непомітні зміни в цих індексах кольорів для внесення прихованої інформації.

Процес внесення прихованого повідомлення в зображення полягає в тому, щоб вибрати пікселі зображення, в яких можна змінити значення індексу кольору без помітних змін візуального сприйняття зображення. Зміни в індексах кольорів можуть бути виконані, наприклад, шляхом заміни найменш значущих бітів індексу на біти прихованого повідомлення.

При отриманні зображення з внесеним прихованим повідомленням, процес вилучення полягає в аналізі змінених індексів кольорів і відновленні прихованої інформації.

Метод приховування в палітрі є одним зі способів стеганографічного застосування до зображень з обмеженою палітрою кольорів, таких як зображення у форматі GIF. Він дозволяє вбудовувати приховану інформацію без помітних змін в зовнішньому вигляді зображення, що робить його використання в стеганографії привабливим для певних сценаріїв та вимог.

Метод дописування даних в кінець BMP-файлу використовує особливість структури цього формату, де кінець даних зображення визначається на основі заголовка, який знаходиться в підрядку знизу-вгору.

Зазвичай BMP-файл складається з заголовка, який містить метадані про зображення, і самого растрового зображення, яке йде після заголовка. Однак, через особливість формату, програми, що обробляють BMP-файли, визначають кінець зображення на основі інформації з заголовка, а не шляхом фактичного перевіряння всього файлу до кінця.

Тому, використовуючи цей метод, можна приховати додаткову інформацію, дописуючи її в кінець BMP-файлу після завершення фактичного зображення, але перед заголовком. Оскільки програми, які читають BMP-файли, ігнорують дані після визначеного кінця зображення в заголовку, ці додаткові дані можуть бути приховані.

Цей метод вимагає знання структури BMP-файлу та правильного форматування додаткових даних, щоб забезпечити їх правильне внесення та вилучення. Варто враховувати, що при використанні цього методу можуть виникнути певні виклики, такі як відображення змін у вигляді зображення та можливість виявлення прихованої інформації шляхом детального аналізу файлу.

2.1.2 Неформатні методи приховування інформації для графічних об’єктів

Методи приховування в графічних зображеннях з палітрою кольорів.

Використання палітри (або іншими словами - відображення кольорів) в графічних форматах пов'язане зі спробою зменшити розмір зображень. Палітра була введена в графічні адаптери з метою спрощення їхньої структури та забезпечення більшого дозволу при меншому обсязі оперативної пам'яті. Після цього з'явилися формати зберігання растрових графічних зображень, які використовують палітру. Деякі з цих форматів є активно використовуваними й нині. Наприклад, формат GIF, який став популярним в Інтернеті і є невід'ємною частиною дизайну сучасних веб-сторінок і реклами в Інтернеті. Кількість переданих по мережі файлів у форматі GIF значно перевищує кількість переданих сторінок і листів. У світі веб-дизайну також використовується формат PNG, який дозволяє використовувати палітру кольорів, але не набув такого ж широкого поширення, як GIF.

Коли ми використовуємо зображення з палітрою, кожна точка зображення може мати лише один колір, який вибирається з наявних кольорів палітри. Палітра кольорів представляє собою набір елементів, кожен з яких визначає інтенсивність кольорових компонентів в певному колірному просторі (зазвичай RGB). Таким чином, кожна точка зображення містить лише номер кольору з палітри, а не пряму інформацію про її кольорові властивості у колірному просторі.

Нижче наведений приклад 8-бітного RGB-зображення розміром 4x4 точки. Зображення складається з точок зеленого (G) і синього (B) кольору, які чергуються в шаховому порядку. Щоб представити це зображення, використовується матриця, де кожен елемент містить значення кольорових компонентів (R, G, B).

(0,255,0) (0,0,255) (0,255,0) (0,0,255)

(0,0,255) (0,255,0) (0,0,255) (0,255,0)

(0,255,0) (0,0,255) (0,255,0) (0,0,255)

(0,0,255) (0,255,0) (0,0,255) (0,255,0)

Для зберігання такої матриці без використання палітри потрібно 384 біти пам'яті. Однак, якщо ми використовуємо зображення з палітрою, то нам потрібна лише палітра, що складається з двох кольорів.

$$0 \rightarrow (0,255,0);\ 1 \rightarrow (0,0,255)$$

Замість зберігання кольорових значень для кожної точки, ми можемо просто зберігати номери кольорів з палітри для кожної точки.

0 10 1

1 0 1 0

0 1 0 1

1 0 1 0

Таким чином, для даного зображення з палітрою нам потрібно лише 48 біт про палітру і 16 біт для зобрадення.

Метод приховування з використанням молодших біт елементів палітри.

У всіх форматах, які використовують палітру кольорів, сама палітра зберігається разом із зображенням у його файлі. Тому для приховування повідомлення можна використовувати метод приховування в молодших бітах елементів палітри. Адже формат зберігання елемента палітри аналогічний формату зберігання точки звичайного зображення без палітри.

Однак, розмір палітри обмежений до 256 елементів, і в кожному з них можна приховати не більше 3 бітів. Тому даним методом можна приховати повідомлення розміром не більше 768 бітів, що значно менше розміру самого зображення.

Крім того, після приховування в палітрі можуть з'явитися елементи, які кодують однакові кольори. Це може бути використано як критерій для визначення наявності прихованого повідомлення в молодших бітах палітри. Якщо в палітрі з'являються декілька елементів з однаковими кольорами, можна припустити, що відбулося приховування даних.

Метод приховування, заснований на наявності однакових елементів палітри. Даний метод використовується якщо палітра містить два або більше однакових елементів, то немає значення, якому з них буде призначено певну точку зображення. Так як при перегляді зображення ці точки будуть виглядати однаково, ця особливість може бути використана для приховування даних без видимих змін у самому зображенні. Важливо враховувати, що використання однакових елементів палітри не має смислу з точки зору кодування графічних зображень, і призводить до збільшення розміру зображення.

У цьому методі спочатку знаходяться кілька елементів палітри, які найчастіше зустрічаються в графічному зображенні. Для цих елементів створюються їх "двійники" в палітрі. Потім проходить послідовний перегляд кожної точки зображення. Якщо точка посилається на елемент, який має "двійника" в палітрі, то ця точка використовується для приховування наступного біта повідомлення. Наприклад, якщо біт повідомлення має значення 1, то значення точки замінюється на "двійника" цього елемента палітри.

Розглянемо приклад даного методу. Припустим, що ми маємо повідомленя А яке дорівнює 10110101, малітра має два коліри(0 → (0,255,0); 1 → (0,0,255)).

Тоді зображення має вигляд:

0 1 0 1

1 0 1 0

0 1 0 1

1 0 1 0

Після додавання в палітру елемента 2→(0,0,255) ідентичний кольору 1, приховане повідомлення буде мати наступний вигляд:

0 2 0 1

1 0 2 0

0 1 0 2

1 0 2 0

Якщо використовувати декілька однакових кольорів в палітрі, то метод приховування можна розширити. Однак важливо зазначити, що це призведе до зменшення стійкості прихованого повідомлення.

Метод приховування шляхом перестановки елементів палітри.

Цей метод стеганографії використовує порядок елементів палітри зображення для приховування інформації. Припускається, що палітра фіксованого зображення містить n різних елементів, тобто немає жодних пар однакових елементів. Враховуючи комбінаторику, кількість перестановок n різних елементів дорівнює n!. Ясно, що максимальна довжина прихованого повідомлення, використовуючи перестановки, становить приблизно log_2 (n!) біт. Загалом, метод приховування шляхом перестановки елементів палітри полягає в тому, що задається відображення, яке при фіксованому ключі встановлює однозначний відповідний зв'язок між будь-яким повідомленням допустимої довжини та певною перестановкою елементів палітри контейнера.

Наведено приклад простого методу перестановки елементів палітри. Повідомлення m є цілим числом від 0 до n! - 1, де n - кількість різних елементів палітри. Елементи палітри впорядковані за зростанням ваги, яка визначається формулою (2.1), що наведено нижче.

$$(65536R + 256G + B) \qquad (2.1)$$

В новій палітрі, отриманій після приховування, є порожні місця, пронумеровані від 0 до n-1. Перша позиція для елемента палітри обчислюється як залишок від ділення m на n. Друга позиція визначається як залишок від ділення m без остачі на n, а результат ділення без остачі на n-1. Аналогічно обчислюються позиції для решти елементів палітри, що відповідають повідомленню m. Після отримання нової палітри необхідно відповідним чином змінити значення всіх точок зображення.

Припустимо, що палітра складається з трьох елементів, a, b і c, впорядкованих за алфавітом. Для приховування повідомлення використовується максимально можливе значення m, яке дорівнює 3! - 1 = 5. При діленні 5 на 3 отримуємо залишок 2, що означає, що елемент a займе останнє місце в новій палітрі. Потім, при діленні 5 на 3 без остачі і діленні результату на 2 отримуємо залишок 1. Це означає, що елемент

b залишиться на своєму місці. І, нарешті, залишок від ділення 5 на 2 дорівнює 1, що означає, що елемент c займе перше порожнє місце. Отже, після приховування палітра матиме вигляд cba.

Для вилучення повідомлення, процес виконується в зворотному порядку. Перший елемент a займає останнє місце, яке має номер 2, тому залишок від ділення m на 3 дорівнює 2, і m не може бути 0. Другий елемент b займає місце з номером 1, тому залишок від ділення m на 2 дорівнює 1. Таким чином, m = 1 + 1 * 2 + 3 = 5.

2.2 Вбудовування інформації в зображення

Мультимедійні об'єкти зазвичай мають багато доступного простору, який можна використовувати для приховування додаткової інформації за допомогою стеганографічних методів. Різні рівні доступного простору використовуються для розробки різних способів внесення інформації в зображення.

2.2.1 Методи приховування інформації в частотній області

Методи заміни, які нестійкі до стиснення з втратами, майже повністю спотворюють всю впроваджену інформацію. Це означає, що під час стиснення зображення або іншого мультимедійного об'єкта, впроваджена інформація втрачається або стає непридатною для витягування. На відміну від цього, методи приховування в частотній області зазвичай мають більшу стійкість до стиснення з втратами.

Ортогональні перетворення, такі як дискретно-косинусне перетворення (DCT), швидке перетворення Фур'є (FFT) і вейвлет-перетворення, широко використовуються в стеганографії. Це пояснюється їхнім використанням у алгоритмах стиснення зображень, оскільки ці перетворення дозволяють ефективно представляти і компактно зберігати сигнал в частотному домені. Ці перетворення можуть бути застосовані до всього мультимедійного контейнера або до його окремих частин, що дає можливість ефективно впроваджувати інформацію в мультимедійні об'єкти з мінімальними змінами в їх вигляді та якості.

Існує багато методів стеганографії, які базуються на ортогональних перетвореннях. Проте, при виборі конкретного методу для впровадження інформації в зображення необхідно враховувати, до якого типу стиску зображення можуть піддаватися з часом.

Наприклад, алгоритм дискретного косинусного перетворення (DCT) є базовим для стандарту стиснення зображень JPEG. Він ефективно компресує зображення, використовуючи коефіцієнти DCT, але при цьому може впливати на впроваджену стеганографічну інформацію. У разі повторних стиснень JPEG-зображення може втрачати додаткову інформацію, яка була впроваджена за допомогою методів, заснованих на DCT.

У випадку JPEG2000, який використовує вейвлет-перетворення, цей стандарт забезпечує кращу стійкість до стиску зображення. Вейвлет-перетворення може забезпечити більшу резервну ємність для впровадження інформації, і зображення, що пройшло стиснення за допомогою JPEG2000, може зберігати впроваджену інформацію краще, ніж у випадку JPEG.

Метод відносної заміни величин коефіцієнтів дискретно-косинусного перетворення (ДКП), також відомий як метод Коха-Жао є ефективним, оскільки зміни в коефіцієнтах ДКП не спричиняють значних змін у вигляді та якості зображення. Він забезпечує певний рівень стійкості до різних видів атак і стиску зображень. Тому цей метод є популярним у стеганографії для приховування інформації в частотній області зображень.

Перш за все, зображення розбивається на блоки, і для кожного блоку обчислюються ДКП-коефіцієнти. Зазвичай використовуються два низькочастотних коефіцієнти, які відображають загальну яскравість або низькочастотну складову блоку. Різниця між цими коефіцієнтами порівнюється з певним пороговим значенням.

Для вбудовування інформації в блок використовуються зміни значень коефіцієнтів. Зазвичай це полягає в заміні одного з коефіцієнтів на нуль або одиницю, що представляє приховану бітову інформацію. Вибір конкретного коефіцієнта для заміни може залежати від розміру різниці між низькочастотними коефіцієнтами та від самої прихованої інформації.

Метод Бенгама-Мемон-Ео-Юнга (БМЕЮ) є оптимізованою версією методу Коха-Жао, де зміни вносяться не в усі блоки зображення, а лише в певні блоки, які краще підходять за своїми властивостями. Вибір цих блоків здійснюється на підставі їхніх властивостей, зокрема, відсутності різких переходів яскравості та монотонності.

В методі БМЕЮ використовуються два параметри: P L і P H, які встановлюють граничні значення для першої та другої властивостей відповідно. Блоки вважаються придатними для обробки, якщо їхні властивості не перевищують значення P L та не падають нижче значення P H.

При впровадженні інформації, нуль вводиться, коли третій коефіцієнт менше будь-якого з перших двох, а одиниця вводиться шляхом збільшення третього коефіцієнта відносно перших двох. Якщо такі перетворення призводять до значного спотворення блоку, то такий блок не використовується для впровадження інформації.

Метод Фрідріх використовує комбінацію двох алгоритмів для впровадження інформації в зображення. Перший алгоритм впроваджує інформацію в низькочастотні коефіцієнти дискретного косинусного перетворення, тоді як другий алгоритм впроваджує інформацію в середньочастотні коефіцієнти дискретного косинусного перетворення.

Впровадження інформації в низькочастотні коефіцієнти дозволяє забезпечити хорошу незамітність інформації, оскільки низькочастотні компоненти мають значний вплив на сприйняття зображення людиною. Впровадження інформації в середньочастотні коефіцієнти доповнює цей процес, дозволяючи розподілити інформацію по ширшому діапазону частот і збільшити стійкість до аналізу з боку стеганалізаторів.

Таким чином, застосування методу Фрідріх, який поєднує впровадження інформації в низькочастотні та середньочастотні коефіцієнти дискретного косинусного перетворення, допомагає досягти високого рівня стійкості до стеганалізу та забезпечити незамітність впровадженої інформації.

2.2.2 Методи заміни біт в просторової області

Ці методи засновані на принципі заміни біт надлишкової і малозначимої інформації зображення на біти впроваджуваного повідомлення. Метод заміни найменш значущих бітів (LSB) є найпоширеніший методом заміни бітів для впровадження інформації в зображення. У цьому методі, менш значущі біти пікселів зображення замінюються на біти впроваджуваної інформації.

Однак, використання цього методу має свої обмеження. Зазвичай довжина впроваджуваної інформації менша, ніж кількість біт в зображенні, тому після впровадження виникають області з різними статистичними властивостями, які можуть бути виявлені статистичними тестами. Це може підказати про наявність впровадженої інформації. Щоб уникнути цього, до впроваджуваної інформації додаються випадкові біти, щоб збільшити її довжину до кількості пікселів у зображенні, яке використовується для впровадження. Цей процес відомий як додавання інформаційного сміття.

Метод заміни найменш значущих бітів має перевагу в простоті реалізації та високій корисній ємності контейнера, оскільки кількість пікселів у зображенні значно перевищує кількість біт впроваджуваної інформації. Однак, будь-яке спотворення зображення також призводить до спотворення впровадженої інформації, що є недоліком цього метод.

Для визначення корисної ємності контейнера при реалізації методу заміни найменш значущих біт можна скористатися формулою(2.2):

$$Q = H * W * V * D \tag{2.2}$$

де H - це висота зображення в пікселях,

W - це ширина зображення в пікселях,

V - це число компонент кольору,

D - це кількість найменш значущих біт в кожній компоненті,

Q - це ємність контейнера, яка вимірюється в бітах.

Метод блочного приховування використовує розбиття вихідного зображення на блоки і впровадження інформації в кожен блок за допомогою біту парності та

впроваджуваного біту. Зміна в контейнері відбувається тільки в тих блоках, де біт парності не збігається з впроваджуваним бітом.

Цей метод має переваги, такі як можливість вибору розміру блоку для оптимального впровадження інформації та можливість модифікації тільки в одному блоку, що зменшує спотворення контейнера. Однак, він має недоліки, пов'язані зі стійкістю до спотворень контейнера.

Метод блочного приховування має низьку стійкість до спотворень контейнера, оскільки зміна впроваджуваного біта впливає на біт парності і може призвести до спотворення зображення в тому блоку. Це може бути помітним при аналізі статистичних властивостей зображення або за допомогою візуального спостереження.

Метод заміни палітри є ще одним способом впровадження даних в зображення. У цьому методі кожен піксель зображення відповідає індексу палітри, а саме значенню з палітри кольорів. Інформація впроваджується шляхом зміни послідовності кольорів в палітрі.

Оскільки інформація кодується у послідовностях кольорів, кількість можливих послідовностей залежить від кількості доступних кольорів у палітрі. Якщо палітра містить N кольорів, то можливих послідовностей буде N! (факторіал N).

Метод заміни палітри є простим у використанні, але має свої обмеження. Цей метод підходить для малих повідомлень, оскільки його потужність обмежена кількістю доступних кольорів у палітрі, і кількість можливих послідовностей зростає швидко з кожним новим кольором.

Однак, метод заміни палітри не є стійким до стеганоатак, пов'язаних зі зміною або перегрупуванням палітри зображення. Якщо палітра змінюється, це може призвести до втрати вкрапленої інформації або некоректного її відновлення.

Ще одним методом впровадження інформації в просторовій області зображення є метод зміни яскравості. У цьому методі створюється маска, яка має таку ж розмірність, як і контейнер (зображення), і складається з псевдовипадкових елементів, які можуть бути нулями або одиницями.

Контейнер (зображення) розбивається на матриці розміром 8x8 пікселів. Кожна матриця поділяється на два масиви - B1 і B2. Для кожного масиву обчислюється середнє значення яскравості, позначене як λ1 і λ2 відповідно.

Метод зміни яскравості використовує різницю між середніми значеннями яскравості λ1 і λ2 для впровадження інформації. Залежно від значення різниці і впроваджуваного біту, один з масивів B1 або B2 може бути змінений шляхом зміни яскравості пікселів. Порядок вбудовування біта повідомлення проводиться згідно з наступною формулою(2.3):

$$S(x,y)=\begin{cases}1, & \text{при } \boldsymbol{\lambda 1 - \lambda 2} > E \\ 0, & \text{при } \boldsymbol{\lambda 1 - \lambda 2} < -E\end{cases} \quad (2.3)$$

де E - це параметр, який відповідає за значення порогу, що є необхідною різницею між середніми значеннями яскравості.

Якщо умова не виконується, то одне із значень яскравості λ 1 або λ 2 модифікується, щоб умова виконувалася. Цей метод дозволяє впроваджувати інформацію в просторовій області зображення, використовуючи властивості яскравості пікселів. Проте, метод зміни яскравості може бути вразливим до різних видів атак, таких як атаки на маску, розміщення та модифікація пікселів, що можуть призвести до втрати впровадженої інформації або її некоректного відновлення.

2.3 Алгоритми стиснення зображень

В стеганографії алгоритми стиснення зображень можуть бути використані для ефективного вбудовування прихованої інформації в контейнер-зображення. Основною метою використання стиснення в стеганографії є зменшення розміру прихованого повідомлення для зручності передачі та збереження. Два широко використовуваних методів стиснення в стеганографії це JPEG і PNG.

Одним з популярних алгоритмів стиснення, який може бути використаний в стеганографії, є алгоритм JPEG (Joint Photographic Experts Group). JPEG використовує метод стиснення з втратами, який базується на видаленні непомітної для людського сприйняття інформації зображення. Вбудовування прихованого повідомлення здійснюється шляхом заміни найменш значущих бітів (LSB) коефіцієнтів ДКП

(дискретного косинусного перетворення), що дозволяє впливати на менш помітні частини зображення. Основними кроками алгоритму стиснення JPEG є:

1.Підготовка зображення:

Конвертування кольорової моделі: Зображення, яке може бути використане в різних кольорових моделях (наприклад, RGB), перетворюється в іншу модель (наприклад, YCbCr), яка підходить для подальшої обробки та стиснення. Формулой переведення моделі RGB в YCbCr (2.4) наведено ниже:

$$\begin{aligned} Y &= R * 0{,}299 + G * 0{,}587 + B * 0{,}114 \\ Cb &= R * (-0{,}169) + G * (-0{,}332) + B * 0{,}5 + 128 \\ Cr &= R * 0{,}5 + G * (-0{,}419) + B * (-0{,}0813) + 128 \end{aligned} \quad (2.4)$$

Наступним кроком є "проріджування" (subsampling). Зрозуміти це просто: береться 2x2 масив пікселів, далі беруться Cb і Cr — середні значення кожного компонента YCbCr цих 4 пікселів. І так ми виграли 6 байт, замість 4 Y, 4 Cb, 4 Cr ми отримали 4 Y і однакові для кожного з них Cb і Cr (4 + 4 + 4 = 12; 4 + 1 + 1 = 6; 12 - 6 = 6). У масштабах навіть 2x2 стиск із втратою з коефіцієнтом стиснення 2:1 звучить солідно. Це стосується всього зображення. І так скинули половину розміру. А такий прийом ми можемо використовувати завдяки нашому сприйняттю кольору. Людина з легкістю помітить різницю в яскравості, але не в кольорі, якщо вона усереднена в маленькому піскелевому блоці. Також проріджування може виконуватися в лінію, 4 пікселі по горизонталі та вертикалі. Перший варіант використовується найчастіше. Якщо важлива якість зображення, то проріджування не виконується взагалі.

Розбиття зображення на блоки: Зображення розбивається на неперекриваючіся блоки пікселів. Зазвичай використовуються блоки розміром 8x8 пікселів.

2. Блочне перетворення:

Застосування Дискретного Косинусного Перетворення (ДКП): Кожен блок 8x8 пікселів піддається ДКП, що перетворює його з просторової області в частотну область. У цій частині з картинки виймають усе зайве. Використовуючи ДКП треба зрозуміти, чи описує даний блок (8x8) якусь монотонну частину зображення: неба, стіни; або він містить складну структуру (волосся, символи і т.д.). Не дивно, що 64

схожих за кольором пікселів можна описати всього одним, т.к. розмір блоку вже відомий. Ось вам і стиск: 64 до 1.

Дискретне косинусне перетворення (ДКП) перетворює блок у спектр, і там, де показники різко змінюються, коефіцієнт стає позитивним, і чим різкіший перехід, тим вище буде вихід. Там, де коефіцієнт вищий, зображені чіткі переходи в кольорі та яскравості, а де він нижчий - слабкі (плавні) зміни величин компонентів YCbCr у блоку.

3. Квантування:

Квантування коефіцієнтів відбувається шляхом поділу кожного коефіцієнта Дискретного Косинусного Перетворення на певне значення квантування. Значення квантування вибирається в залежності від потреб: якщо необхідно зберегти якість зображення без додаткової втрати, то коефіцієнт квантування встановлюється на одиницю; якщо важливо зекономити пам'ять і розмір даних, то використовується значення квантування більше одиниці, яке округлюється вниз. В результаті округлення можуть утворитися нульові коефіцієнти, які можуть бути видалені на наступному етапі. Квантування допомагає видалити менш значущу інформацію та зменшити обсяг даних. Приклад як квантування графіку y = sin(x) наведено на рисунку 2.1.

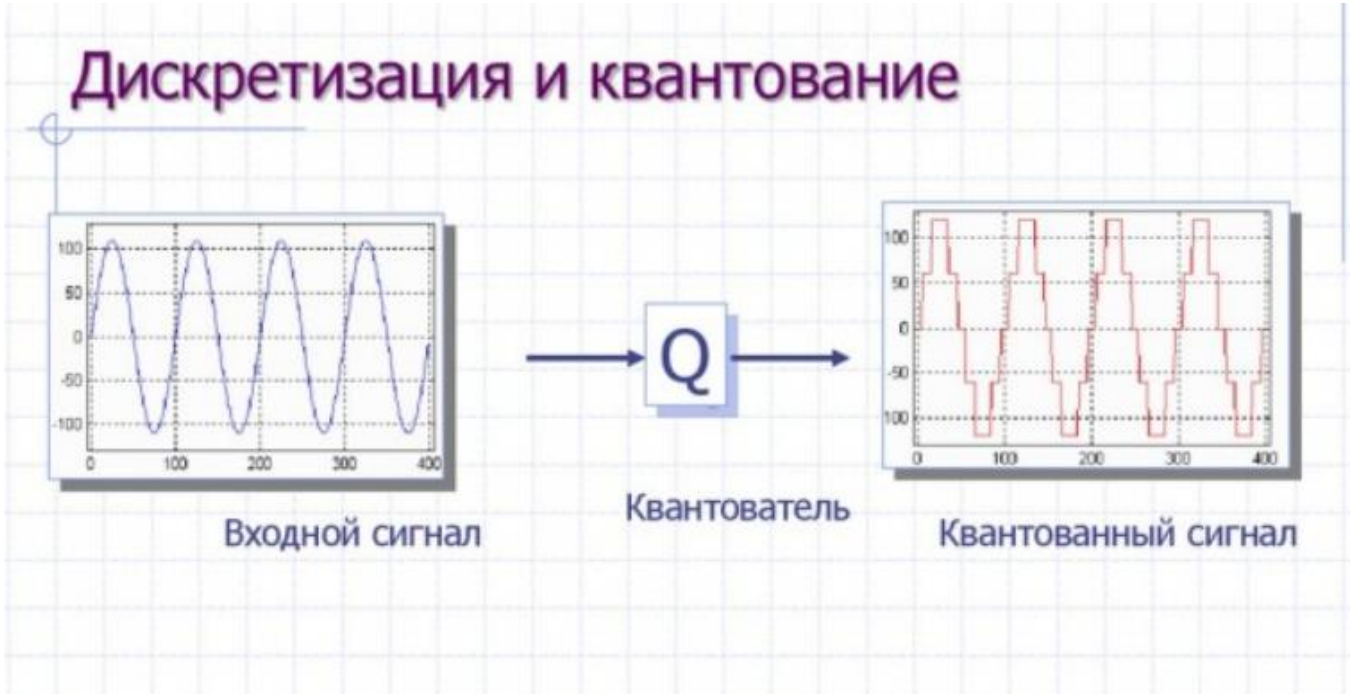

Рисунок 2.2 – Графік y = sin(x) до і після квантування

Далі необхідно відкинути непотрібні нулі після квантування, тому проведемо зигзагоподібний запис цієї матриці як показано на рисунку 2.3:

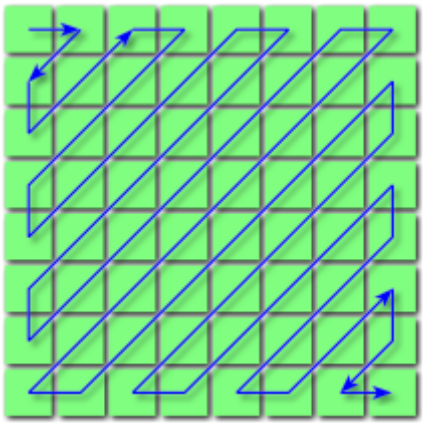

Рисунок 2.3 – Зигзагоподібний запис матриці

Отримуємо одновимірний масив з числами. Ми бачимо, що в ньому багато нулів, їх можна видалити. Для цього замість послідовності з множини нулів ми вписуємо 1 нуль і після нього число, що позначає їх кількість у послідовності. Таким чином, можна скинути до 1/3 розміру всього масиву.

4. Кодування:

Кодування квантованих коефіцієнтів: Квантовані коефіцієнти кодуються за допомогою методу кодування, такого як кодування Хаффмана. Коефіцієнти з меншою частотою появи отримують більш короткі коди, що сприяє подальшій стисненню. Цей алгоритм базується на ідеї, що символи, які зустрічаються рідше, кодуються більш довгими двійковими кодами, тоді як частіше зустрічаються символи мають коротші коди. Кодування може проводитися за допомогою фіксованої таблиці, де кожен символ має свій двійковий код з певною довжиною, або за допомогою таблиці, створеної спеціально для конкретного набору символів, що потребує кодування.

5. Збереження:

Збереження стиснутого зображення: Кодовані біти записуються в файл зображення, разом з необхідною метаінформацією.

Інший алгоритм стиснення, який може бути використаний в стеганографії, - це алгоритм PNG (Portable Network Graphics). PNG використовує безстислий метод стиснення, що дозволяє зберегти всю інформацію про зображення без втрати якості.

Вбудовування повідомлення в алгоритмі PNG може виконуватись шляхом зміни значень пікселів або додавання додаткових метаданих в файл зображення. Основні кроки алгоритму стиснення PNG наступні:

1. Фільтрація: Початкове зображення перетворюється за допомогою п'яти різних фільтрів, які відстежують піксельні значення та намагаються зменшити рівномірність даних. Це допомагає зменшити величину даних та поліпшити стискання.

2. Стиснення безлосісним методом: Використовується алгоритм стиснення Deflate, який заснований на комбінації алгоритмів Huffman та LZ77. Він використовує повторювані послідовності даних та створює таблиці для ефективного представлення даних.

3. Безвтратне стиснення: Після безлосісного стиснення, дані проходять через процес безвтратного стиснення, що використовує алгоритм Deflate з виключенням стадії стискання. Це дозволяє досягти додаткового стиснення та зменшити розмір даних.

4. Збереження інформації: Зображення PNG зберігається у форматі, що містить метадані, такі як інформація про палітру кольорів, прозорість, гаму тощо. Ці дані допомагають відтворити оригінальне зображення з високою якістю після розпакування.

Алгоритм стиснення PNG забезпечує безвтратне збереження зображень, знижує їх розмір та підтримує прозорість та інші додаткові функції.

2.4 Алгоритми втілення методів стеганографії

2.4.1 Алгоритм LSB

Цей метод полягає в заміні останніх (найменш значущих) бітів пікселів зображення на біти впроваджуваного повідомлення. Оскільки LSB зберігає найменш значущу інформацію про колір пікселя, заміна цих бітів найчастіше не спричиняє помітних змін у вигляді зображення.[19] Різниця між порожнім (оригінальним) зображенням і зображенням, в яке впроваджено повідомлення, повинна бути

непомітною для органів сприйняття людини. Принцип приховування інформації показано на рисунку 2.4.

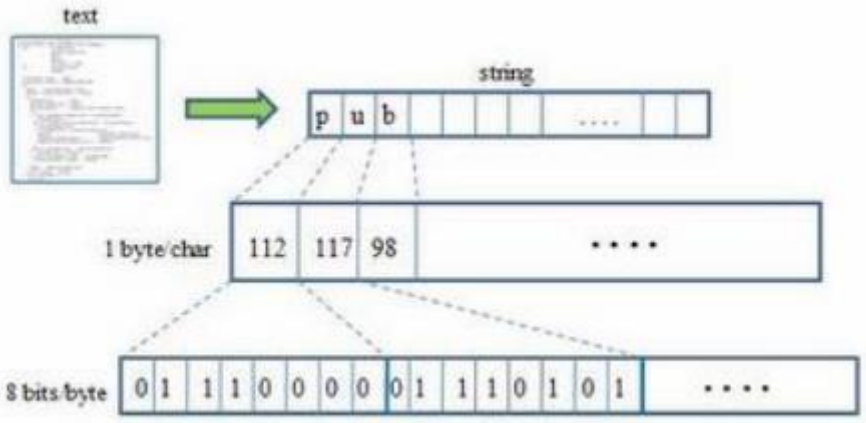

Рисунок 2.4 – Принцип приховування інформації методу LSB

Перетворення тексту в байтову послідовність.

Завдяки слабкій чутливості зору до незначних змін у значеннях кольору, заміна LSB в кожному з трьох каналів (червоний, зелений, синій) може бути виконана з відносно невеликою помітністю для спостерігача. Зміна одного LSB може вплинути лише на 1/256 або менше (менше ніж 1%) відтінку кольору даної точки зображення.

Це дає можливість використовувати метод LSB для стеганографії, де інформація може бути впроваджена в останні біти кольорових компонентів зображення, зберігаючи при цьому вигляд зображення майже без помітних змін.[20-21].

Усі BMP-контейнери потрібно розділити на два класи: «чисті» і «зашумлені». У «чистих» картинках простежується зв'язок між молодшим бітом, який ми змінюємо, і рештою 7-ма бітами елементів кольору, а також простежується істотна залежність самих молодших бітів між собою. Впровадження повідомлення в «чисту» картинку руйнує існуючі залежності, що дуже легко виявляється пасивним спостерігачем. Якщо ж картинка «зашумлена» (наприклад, отримана зі сканера або фотокамери), то визначити вкладення стає на порядок складніше. Таким чином, в якості файлів-контейнерів для LSB-методу рекомендується використовувати файли які не були створені на комп'ютері спочатку. LSB-метод з використанням BMP-контейнера

полягає в наступному: замінюються молодші біти в байтах, що відповідають за кодування кольору.

Припустимо:

00011011 – це черговий байт нашого секретного повідомлення;

...11101111-01001110-01111101-0101100100… – це байти в зображенні.

Тоді кодування буде виглядати так:

1) розбити байт секретного повідомлення на, наприклад, 4 двобітові частини 00-01-10-11;

2) замінити отриманими фрагментами двійку молодших бітів зображення ...11101100– 01001101-01111110-01011011.

Така заміна в загальному випадку не помітна людському оку. Більш того, багато старих пристроїв виведення, навіть не зможуть відобразити такі незначні зміни. Зрозуміло, що можна змінювати не тільки 2 молодших біта, але і будь-яку їх кількість. Тут є наступна закономірність – чим більша кількість біт замінюється, тим більший обсяг інформації можна заховати, але й тим більші похибки у вихідному зображенні це викличе.

Методи LSB (заміна найменш значущого біта) є вразливими до різних видів атак і можуть бути використані тільки у випадку, коли канал передачі даних не містить шуму або перешкод. Виявлення контейнера, що використовує LSB, здійснюється шляхом аналізу аномальних характеристик розподілу значень найменш значущих бітів в цифровому сигналі. Аномалії можуть виявлятися в нерівномірності розподілу значень, які можуть бути викликані внесенням змін у молодші біти пікселів.

2.4.2 Алгоритм JSteg

Алгоритм JSteg є одним із відомих методів стеганографії, який використовуєся для приховування інформації в цифрових зображеннях. Його назва походить від поєднання слів "JPEG" (Joint Photographic Experts Group) і "Steganography" (стеганографія). Алгоритм надає значну потужність у вбудовуванні стеганографічних

повідомлень, оскільки він дозволяє приховати до 12.8% від загального обсягу зображення (контейнера).

Основна ідея JSteg полягає у використанні менш значущих бітів коефіцієнтів Дискретного Косинусного Перетворення (ДКП), які виникають під час стиснення зображення у форматі JPEG. Коефіцієнти ДКП представляють частотні компоненти зображення, і найменш значущі біти зазвичай несуть менше візуальної інформації.

Процес вбудовування інформації виконується шляхом заміни найменш значущих бітів коефіцієнтів ДКП вибраних блоків зображення на біти прихованого повідомлення. Це дозволяє впровадити інформацію без помітних змін у зовнішньому вигляді зображення.

Алгоритм JSteg є відносно простим у реалізації, але має обмеження, пов'язані зі специфікою формату JPEG і можливістю втрати даних при стисненні зображення.

Процес вбудовування в алгоритмі JSteg включає в себе наступні кроки:

- Підготовка повідомлення: повідомлення перетворюється на послідовність бітів для вбудовування.
- Вибір контейнера: вибирається зображення, в яке буде вбудовуватися повідомлення.
- Вбудовування: найменш значущі біти кожного пікселя контейнера замінюються на біти повідомлення.
- Збереження контейнера: зображення з вбудованим повідомленням зберігається.

JSteg має важливу особливість - він є стійким до візуальних атак, оскільки зміни, внесені алгоритмом, зазвичай непомітні для людського сприйняття. Однак, він може бути вразливим до статистичних атак, які здатні виявити наявність прихованого повідомлення. Це пов'язано з тим, що заміна бітів впливає на частоту появи певних значень в менш значущих бітах.

JSteg також впливає на пари частот появи коефіцієнтів JPEG. Як показує рисунок 2.5 Змінений алгоритм, що використовує розподіл частот, передбачає, що суміжні частоти мають однакові значення. Для оцінки очікуваного розподілу,

обчислюється середнє арифметичне значень частот і порівнюється з емпіричним розподілом для виявлення можливих змін.

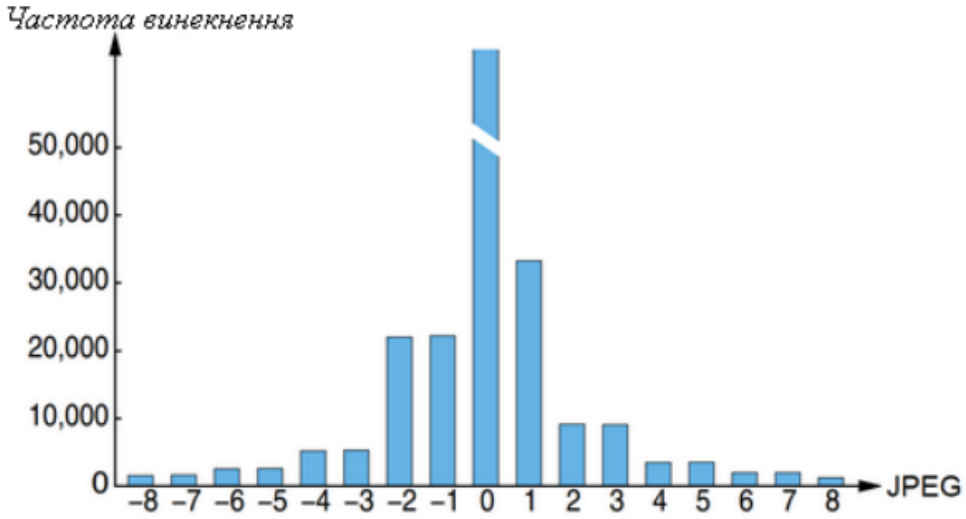

Рисунок 2.5 – Залежність частоти винекнення в НЗБ

Висновки за розділом 2

В даном розділі було розглянуто моделі та методи стеганографічного запису інформації в графічні об’єкти включаючи основні формати зображень які широко використовуються. Зокрема, метод LSB (найменш значущий біт) був обраний для дослідження, а формат зображень PNG був використаний як контейнер.

Суть методу LSB (Least Significant Bit) полягає в тому, що в прихованому повідомленні замінюються найменш значущі біти (LSB) в контейнері. Оскільки найменш значущі біти мають менший вплив на сприйняття людьми, їх заміна на біти повідомлення не суттєво змінює зовнішній вигляд або якість контейнера.

Наприклад, у випадку зображення, RGB-значення кожного пікселя складаються з 8 бітів на канал (Red, Green, Blue). Метод LSB використовує останній (найменш значущий) біт в кожному каналі пікселя для зберігання біта повідомлення. Це означає, що значення кожного каналу замінюється на менш значущий біт повідомлення, не впливаючи при цьому на помітність зображення для спостерігача.

Таким чином, метод LSB надає можливість приховати повідомлення в контейнері, зберігаючи при цьому зовнішній вигляд і якість контейнера майже без помітних змін.

РОЗДІЛ 3
ПРОГРАМНА РЕАЛІЗАЦІЯ ПРИХОВУВАННЯ ІНФОРМАЦІЇ В ГРАФІЧНИХ ОБ'ЄКТАХ З ВИКОРИСТАННЯМ СТЕГАНОГРАФІЧНИХ І КИПТОГРАФІЧНИХ ТЕХНОЛОГІЙ

3.1 Опис майбутнього застосунку

Актуальність стеганографії та криптографії визначається необхідністю захисту конфіденційності та безпеки цифрової інформації в сучасному світі. Зростання кількості електронних комунікацій, збільшення обсягу цифрових даних і залучення нових технологій створюють нові виклики для збереження конфіденційності і запобігання несанкціонованому доступу до даних. А в період війни зосередження на даній темі стало особливо ретельним. Розглянувши актуальність даної теми, було прийнято рішення в даній роботі створити свій програму для шифрування і приховування в графічних файлах даних з використанням криптографічних і стеганографічних технологій. Проаналізувавши методи стеганографії, було вирішено, що програма повинна не тільки приховувати інформацію, а й шифрувати її до цього. Адже передача даних, в більшості випадків, відбувається по відкритих джерелах. І хоча метод передачі закритої інформації в прихованому форматі утруднює її виявлення, але не робить це неможливим. Мій вибір упав на алгоритм LSB для вбудовування вже шифрованого повідомлення в графічний об'єкт. Мову програмування було ж вибрано Pyton. Дана мова програмування поєднує простоту використання з повноцінними можливостями для розробки складних програм. Вона надає широкий набір інструментів для структурування і підтримки великих проектів. Python дозволяє розробникам легко виразити свої ідеї за допомогою зрозумілої синтаксичної конструкції і великої кількості вбудованих функцій. Водночас, вона дозволяє розширювати свої можливості за допомогою сторонніх бібліотек і модулів. Без чого в ми не обійдемся в даній роботі.

3.2 Розробка програмного забезпечення

Підготовка до розробки програми

Першим етапом стало встановлення всіх необхідних програм, які були потрібні для роботи, а саме:

- Pycharm community Edit – середовище розробки програми;
- QT Designer – програма для створення інтерфейсу.

Також було встановлено деякі бібліотеки Python:

- - plyer – використовується для виводу спливаючих повідомлень в трей.
- Cryptography - З неї буде використовуватися модуль Fernet для генерації ключа, за допомогою якого шифруватиметься і дешифруватиметься повідомлення, що приховується в картинці за допомогою алгоритму LSB.
- Pillow – потрібна для роботи із зображенням.

Створення інтерфейсу

Інтерфейс було створено за допомогою програми QT Designer, як показано на рисунку 4.1.

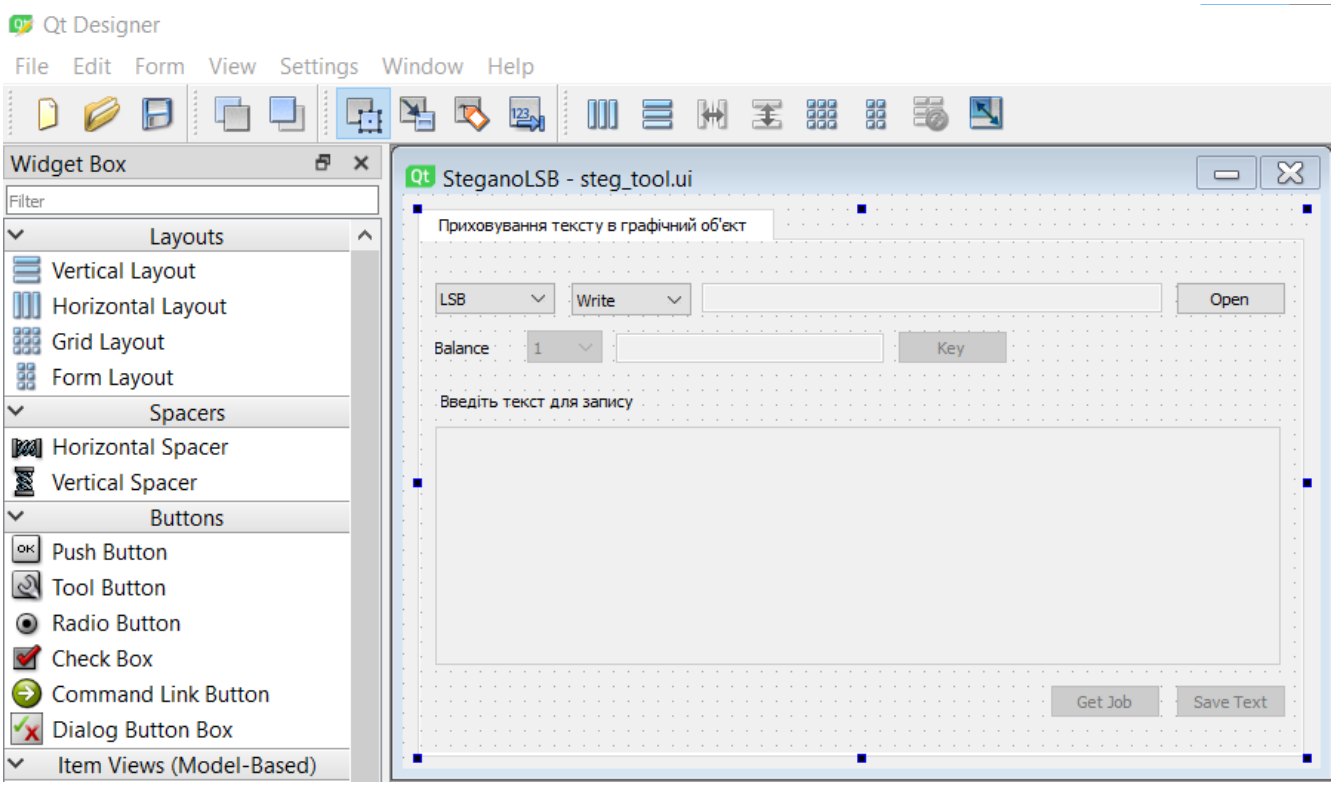

Рисунок 4.1 – Майбутній інтерфейс програми

В інтерфейсі є кілька списків, які надають вибір користувачу. Перший список дозволяє обрати алгоритм для приховування тексту, на разі можливий тільки один варіант LSB. Другий список визначає дію, яку потрібно виконати, а саме запис або читання. Крім того, є ще одне поле для параметра "баланс", який використовується в алгоритмі LSB. Даний параметр визначає кількість молодших бітів, що будуть використані при стеганографічній операції. Діапазон значень для цього параметра від 1 до 4. Чим більше значення балансу, тим менше пікселів буде задіяно в алгоритмі, що може бути помітним для людського сприйняття. Значення балансу впливає на помітність змін в кольорових каналах. Тому, коли баланс менший, зміни стають менш помітними для ока.

Додатково, я реалізував виведення тексту в QTextEdit, але без можливості редагування. Однак, користувач може зберегти цей текст. Також присутній вибір ключа, який стає активним лише при читанні тексту зі зображення, яке було приховано за допомогою алгоритму LSB.

Після створення інтерфейсу в QT Designer, зберігаємо файл і помічаємо, що його формат - .ui, як показано на рисунку 4.2. Для використання цього інтерфейсу в програмі, потрібно перетворити цей файл у формат .py.

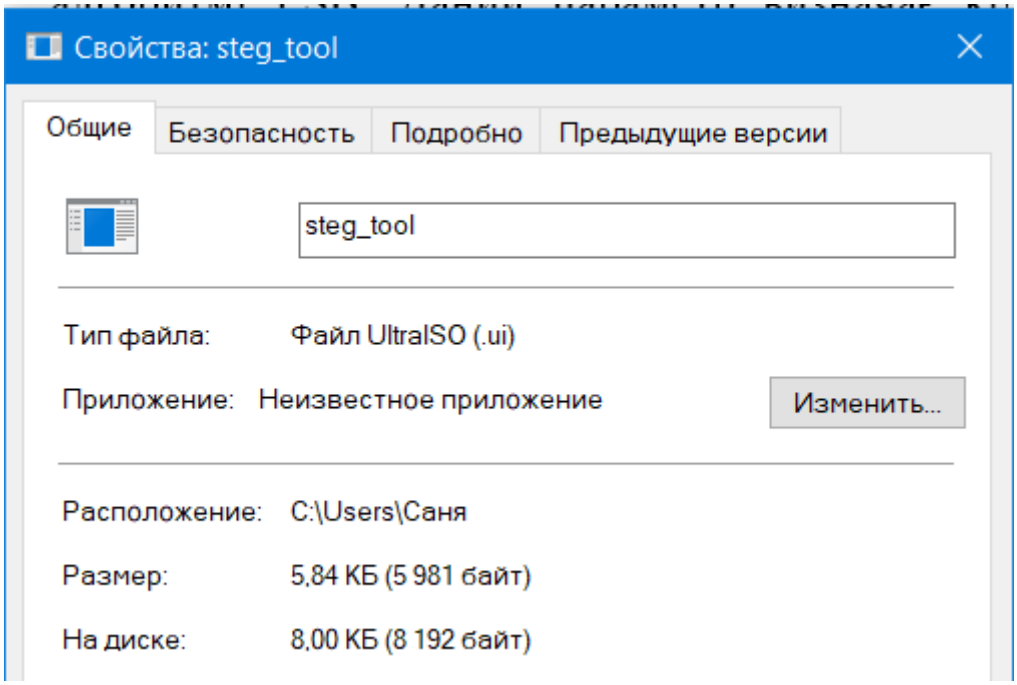

Рисунок 4.2 – Збережений інтерфейс в форматі .ui

Для цього потрібно зайти в термінал і встановити PyQT і PyQT tools за допомогою наступних команд: "pip install pyqt5" і "pip install pyqt5-tools". Після успішного встановлення, переходимо до директорії з необхідним файлом і конвертуємо його у формат .py за допомогою наступної команди: "pyuic5 -x steg-tool.ui -o steg-tool.py". Отриманий конвертований файл можна вставити в нашу програму, як показано на рисунку 4.3.

main.py | steg_tool.py | stegopy.py

```
from PyQt5 import QtCore, QtGui, QtWidgets


class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(620, 388)
        icon = QtGui.QIcon()
        icon.addPixmap(QtGui.QPixmap("Разное/Pictures/X2Q6dQuRLw4.jpg"), QtGui.QIcon.Normal, QtGui.QIcon.On)
        MainWindow.setWindowIcon(icon)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.tabWidget = QtWidgets.QTabWidget(self.centralwidget)
        self.tabWidget.setGeometry(QtCore.QRect(10, 10, 601, 371))
        self.tabWidget.setObjectName("tabWidget")
        self.tab = QtWidgets.QWidget()
        self.tab.setObjectName("tab")
        self.label = QtWidgets.QLabel(self.tab)
        self.label.setGeometry(QtCore.QRect(14, 99, 241, 16))
        self.label.setObjectName("label")
        self.pathImg = QtWidgets.QLineEdit(self.tab)
        self.pathImg.setEnabled(False)
        self.pathImg.setGeometry(QtCore.QRect(190, 28, 311, 20))
        self.pathImg.setObjectName("pathImg")
        self.txtSteg = QtWidgets.QTextEdit(self.tab)
        self.txtSteg.setEnabled(False)
        self.txtSteg.setGeometry(QtCore.QRect(10, 125, 571, 161))
        self.txtSteg.setObjectName("txtSteg")
        self.takeBtn = QtWidgets.QPushButton(self.tab)
```

Рисунок 4.3 – Інтерфейс програми в форматі .py

На даному етапі розробки інтерфейсу завершено, і залишається лише встановити належні налаштування для кнопок. Однак, цей процес буде виконано пізніше, після написання основного коду для шифрування та приховування інформації в фотографіях.

Перед написанням самого коду створюємо в папці програми файл main.py і добавляємо всі потрібні нам бібліотеки встановлені раніше, як представлено на рисунку 4.4.

```
1
2   from os import system
3   from platform import system as psystem
4   from sys import argv, exit
5
6   from cryptography.fernet import Fernet
7   from PyQt5.QtWidgets import QFileDialog
8   from plyer import notification
9
10
11  from stegopy import encrypt, decrypt
12  from steg_tool import *
13
14
```

Рисунок 4.4 – Добавлені бібліотеки в документі main

Спочатку ми реалізовуємо шифрування інформації, для цього було використано модуль fernet з бібліотеки cryptography. Fernet — це реалізація симетричної (також відомої як «секретний ключ») автентифікованої криптографії. Цей клас забезпечує як засоби шифрування, так і дешифрування.

Першим кроком ми створюємо ключ за допомогою команди generate _key. І шифруємо текст за допомогою функції des_encrypt, продемонстровано на рисунку 4.5. І не забуваємо вказати путь де буде збережено ключ як показано на рисунку 4.6. Тому, що без нього розшифрувати повідомлення буде не можливо. В нашому варіанті ключ буде зберігаться в файлі key.dat.

```
def des_encrypt(text, key):

    cipher = Fernet(key.encode())
    result = cipher.encrypt(text.encode())

    return result.decode()
```

Рисунок 4.5 – Функція шифрування тексту

```
file = open("key.dat", "w")
file.write(str(balance) + '$' + str(count) + '$' + key)
file.close()
```

Рисунок 4.6 – Збереження ключа в файлі key.dat

Далі ми реалізували метод декодування зашиврованого тексту за допомогою команди decrypt(text) для цього нам потрібно в функцію задать ключ як це зроблено на рисунку 4.7

```
def des_decrypt(text, key):
    """
    DES Decrypting
    :param text:
    :param key:
    :return:
    """
    try:
        cipher = Fernet(key.encode())
        result = cipher.decrypt(text.encode())
```

Рисунок 4.7 – Реалізація декодування тексту

Далі нам потрібно створити фунції def encrypt рисунок 4.8. і виймання тексту з фото def decrypt рисунок 4.9.Функція encrypt() використовується для шифрування інформації. Вона відкриває зображення, зчитує дані, які необхідно зашифрувати, та генерує ключ шифрування. Зображення модифікується шляхом зміни найменш значущих бітів кольорових компонент пікселів відповідно до бінарного представлення шифруваної інформації. Результат зберігається у новому зображенні, а ключ шифрування зберігається у файлі key.dat.

Функція decrypt() використовується для розшифрування інформації. Вона відкриває зашифроване зображення та використовує ключ для відновлення прихованої інформації шляхом зчитування найменш значущих бітів кольорових компонент пікселів. Розшифрована інформація повертається як результат.

Але спочатку створемо функцію для визначення балансу. Баланс відповідає за кількість молодших бітів пікселя, які використовуються для кодування інформації. Значення балансу може варіюватися від 1 до 4. Зі збільшенням криптографічної міцності кількість пікселів, задіяних у процесі кодування, зменшується. Тому, я рекомендую використовувати значення 1 для найбільшої стійкості до виявлення. Однак, якщо текст не поміщається в обране зображення, можна використовувати значення 2 або 3. Особливість системи балансу полягає в тому, що для певного тексту баланс 3 може бути більш стійким до виявлення, ніж баланс 1.

```
def encrypt(path_to_image, text, key, balance):

    img = dict()
    size = dict()
    coord = dict()

    img["image"] = Image.open(path_to_image)
    img["draw"] = ImageDraw.Draw(img["image"])
    img["pix"] = img["image"].load()

    size["width"] = img["image"].size[0]
    size["height"] = img["image"].size[1]

    text = des_encrypt(text, key)
    binary_text = text_to_binary(text)
    list_two = split_count(''.join(binary_text), balance)

    coord["x"] = 0
    coord["y"] = 0
    count = 0

    for i in list_two:
        red, green, blue = img["pix"][coord["x"], coord["y"]]

        (red, green, blue) = balance_channel([red, green, blue], i)

        img["draw"].point((coord["x"], coord["y"]), (red, green, blue))
```

des_encrypt()

Рисунок 4.8 – Функція приховування даних в фото

```
def decrypt(path_to_image, key):

    balance = int(key.split('$')[0])
    count = int(key.split('$')[1])
    end_key = key.split('$')[2]

    img = dict()
    coord = dict()

    img["image"] = Image.open(path_to_image)
    img["width"] = img["image"].size[0]
    img["height"] = img["image"].size[1]
    img["pix"] = img["image"].load()

    coord["x"] = 0
    coord["y"] = 0
    code = ''

    i = 0
    while i < count:
        pixels = img["pix"][coord["x"], coord["y"]]

        pixel = str(bin(max(pixels)))

        if balance == 4:
            code += pixel[-4] + pixel[-3] + pixel[-2] + pixel[-1]
```

decrypt()

Рисунок 4.9 – Функція виймання даних з фото

Кінцевим етапом створення програми є налаштування кнопок в інтерфейсі. Функція кодування тексту викликається при натисканні на кнопку Get job продемонстровано на рисунку 4.10.

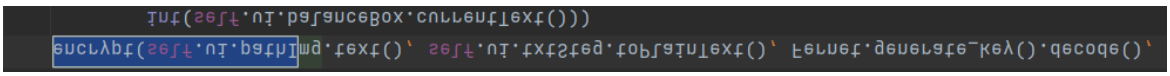

Рисунок 4.10 – Налаштування кнопки Get job для кодування

Ця функція отримує шлях до зображення, яке потрібно закодувати, а також згенерований ключ за допомогою Fernet і значення балансу. Декодування також викликається при натисканні на кнопку Get job. Функція отримує шлях до зображення, яке містить зашифровану інформацію. Продемонстровано на рисунку 4.11. Також була налаштована кнопка вибору файлу ключа. Завдяки цьому ключ не повинен обов’язково лежати в папці з програмою.

```
elif self.ui.hexexCombo.currentText() == "LSB" and self.ui.wreadCmb.currentText() == "Read":
    if self.ui.pathKey.text():
        with open(self.ui.pathKey.text(), 'r') as k:
            key = k.read()
        dec = decrypt(self.ui.pathImg.text(), key)
```

Рисунок 4.11 – Налаштування кнопки Get job для декодування

4.3 Перевірка і тестування працездатності програми

Запускаємо програму і перевіряємо її функціональність. У головному вікні обираємо опцію "Записати" і вибираємо фотографію, в яку хочемо внести інформацію. Потім вибираємо значення балансу, вводимо текст, який бажаємо вбудувати, і натискаємо "Get Job", як показано на рисунку 4.12.

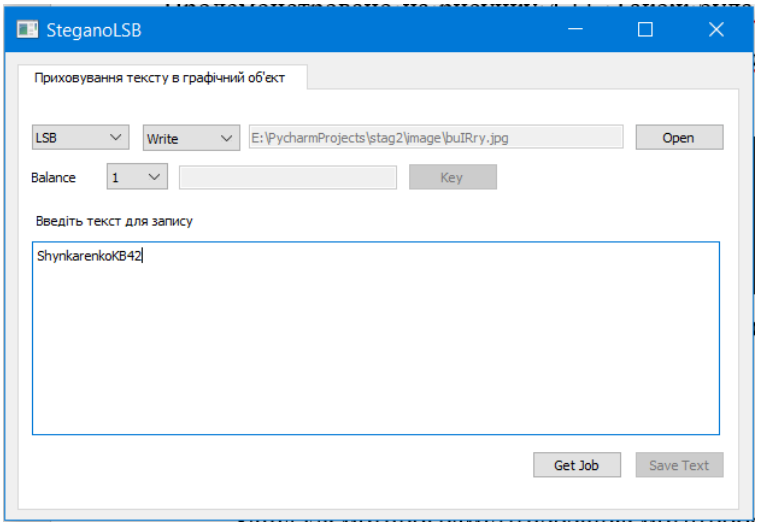

Рисунок 4.12 – Кодування текст в фото

Для перевірки правильності роботи програми, ми переходимо до папки, де знаходиться програма, і перевіряємо наявність фотографії та ключа для декодування, як показано на рисунку 4.13. Також ми перевіряємо, на скільки збільшилось зайняте місце на диску після вбудовування тексту в фотографію, як показано на рисунку 4.14. Лівий знімок зображає порожню фотографію, а правий знімок показує фотографію з вбудованим текстом.

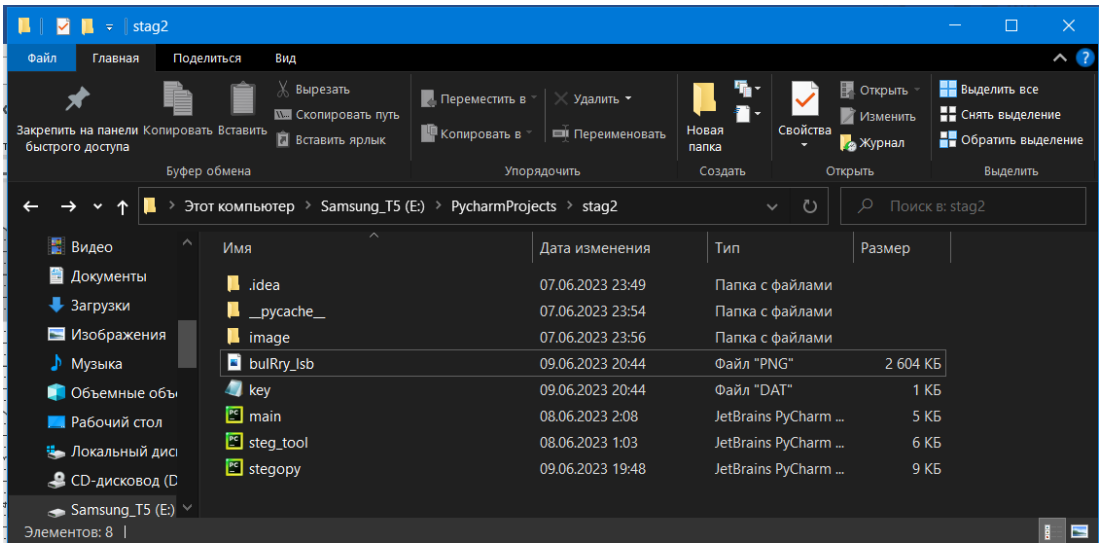

Рисунок 4.13 – Наявність ключа і фото з прихованим текстом

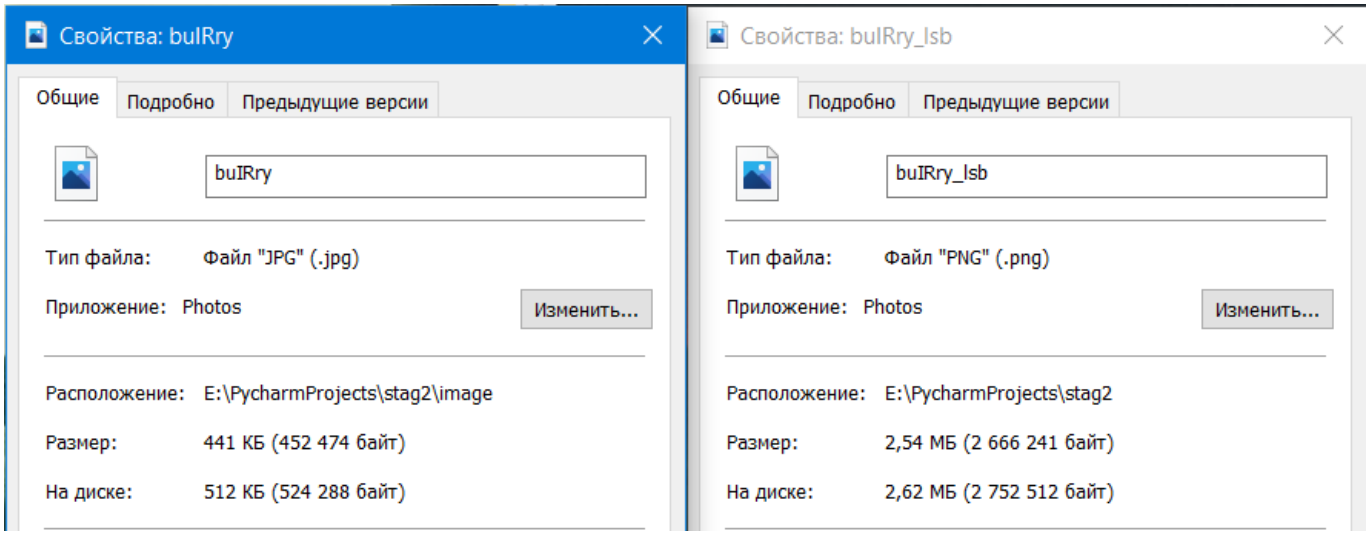

Рисунок 4.14 – Розмір фото з текстом і без

Як видно, в папці дійсно з'явилися ключ і фотографія. Крім того, розмір файлу з текстом збільшився на 2 МБ, що свідчить про успішне вбудовування тексту. Наступним кроком перевірки програми буде спроба дістати зашифрований текст з нашого фото. Для цього змінюємо значення на "read", вибираємо фото з текстом і

вказуємо файл з ключом, а потім натискаємо кнопку "Get Job", як показано на рисунку 4.15.

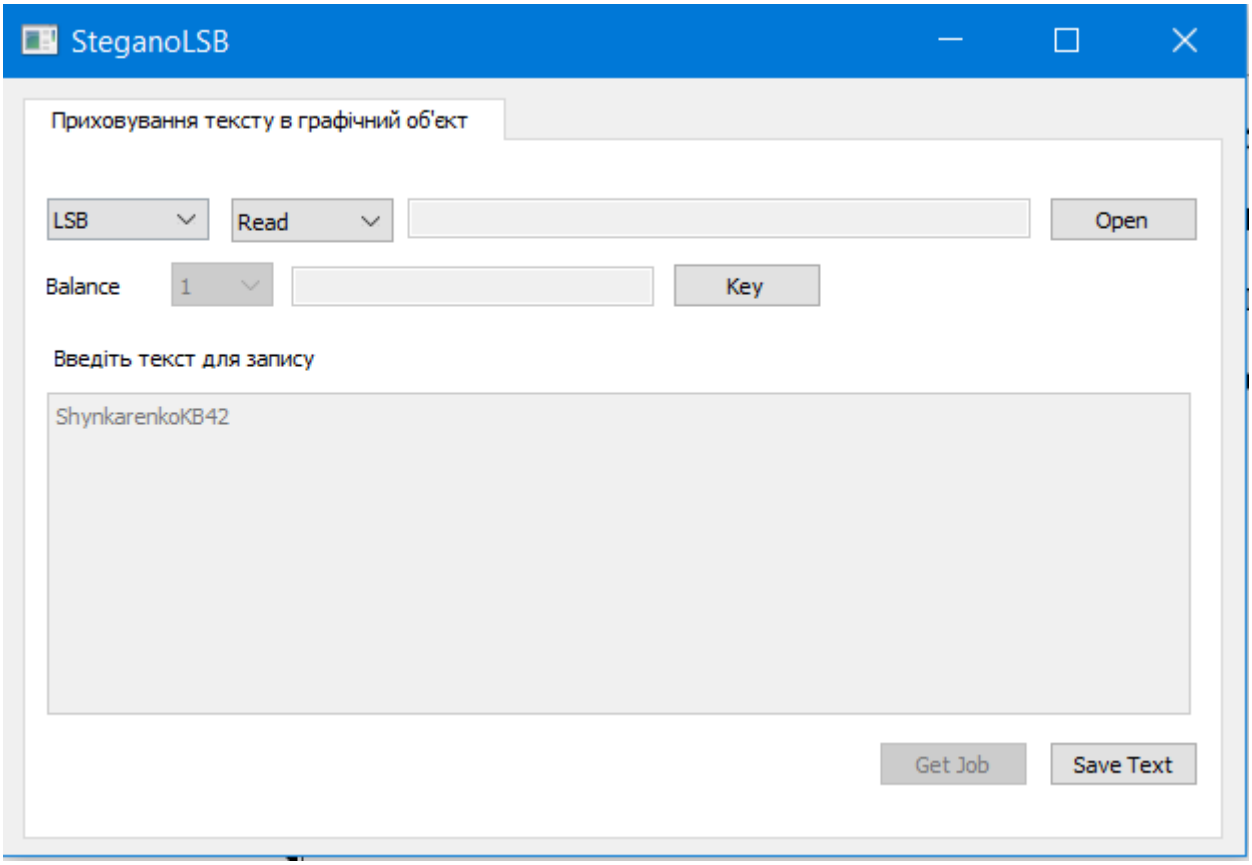

Рисунок 4.15 – Декодування тексту з фото

Як видно з рисунку, все працює належним чином. Також, ми перевіримо можливість зберегти текст з фотографії до файлу. Для цього натискаємо кнопку "Save Text", обираємо назву файлу та місце збереження. Результат демонстровано на рисунку 4.16.

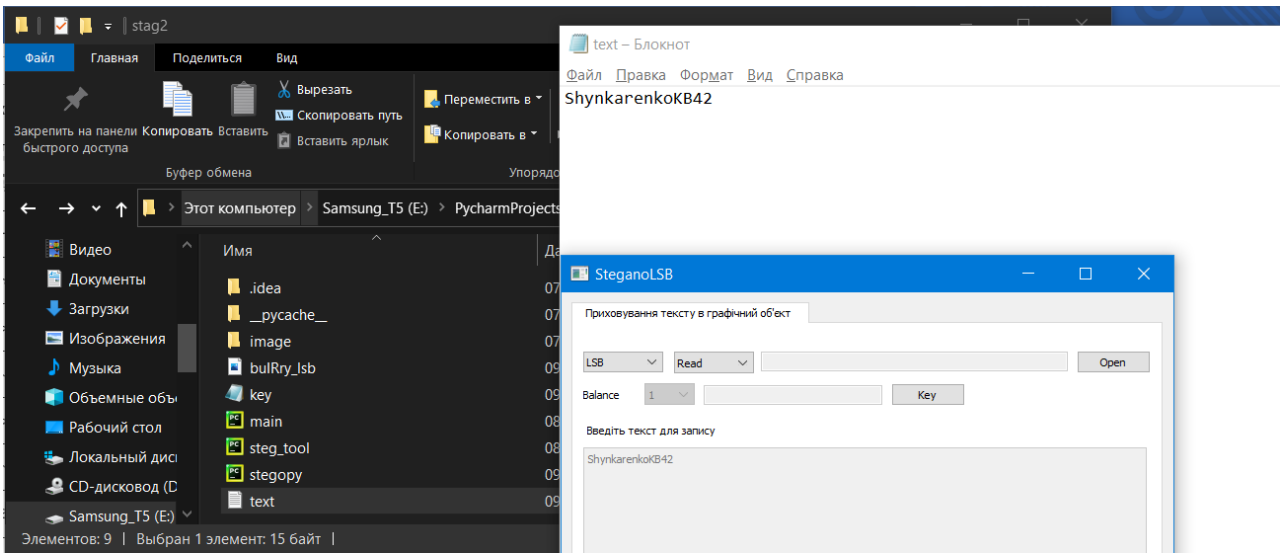

Рисунок 4.16 – Результат зберігання фото в текстовий файл

Також, вважаю важливим перевірити, чи розкодує програма файл з шифротекстом, використовуючи не новий ключ, а старий. Для цього заново проводимо дії, які були виконані раніше, беремо нове фото з прихованим шифротекстом і намагаємося його розкодувати. Проте, на етапі вибору ключа, вибираємо не тільки новостворений ключ, а й той, що залишився з попереднього разу. Результат продемонстровано на малюнку 4.17.

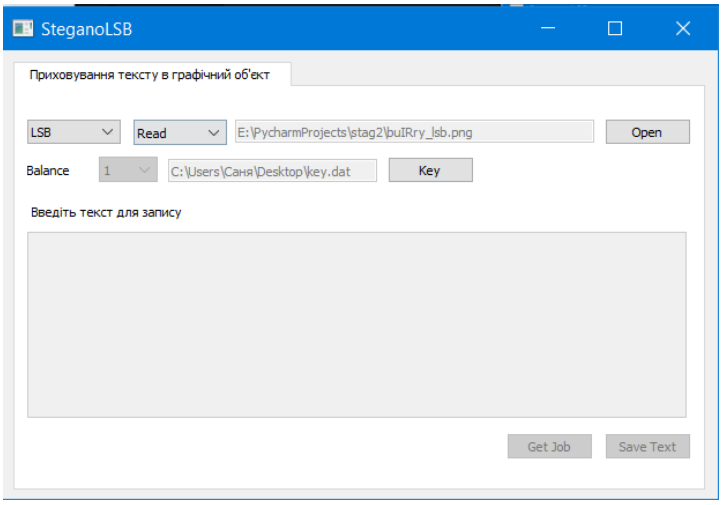

Рисунок 4.17 – Спроба розкодувати файл раніше згенерованим ключем

Як видно з результату, нічого не вийшло, програма не виводить ніякого тексту, але, взявши потрібний ключ, текст все ж є. Продемонстровано на рисунку 4.18.

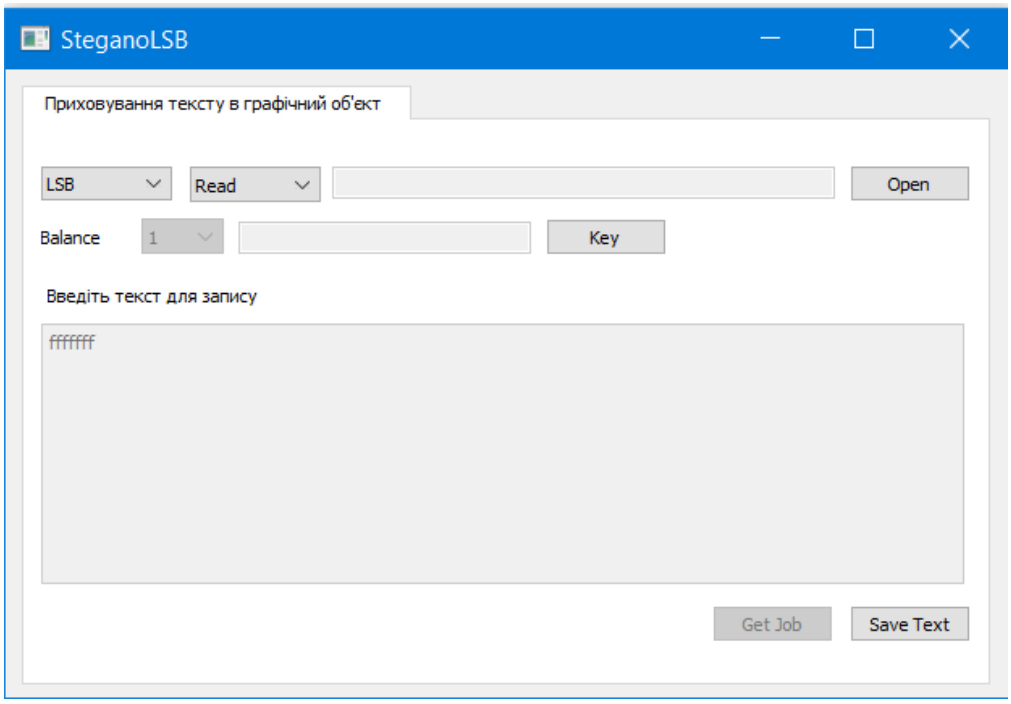

Рисунок 4.18 – Декодування фото з правильним ключем

Також вважаю, не зайвим перевірити розміри закодованих фотографій з однаковим шифротекстом, але з використанням різного балансу. Результат наведено на рисунку 4.19, зліва розмір файлу з балансом 1, і по зростанню - направо.

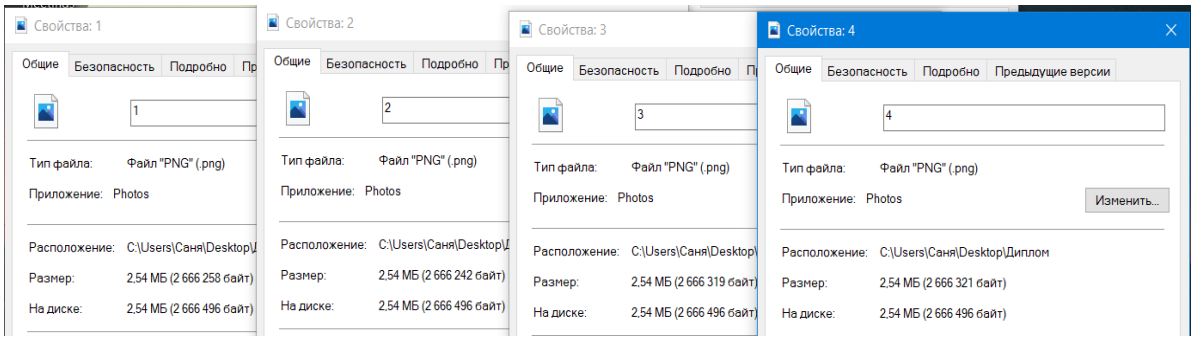

Рисунок 4.19 – Розмір закодованих фото з різним балансом

Як можна побачити на рисунку, різниця хоч і не велика, але все ж є. Що є очікуваним результатом, адже при використанні різного балансу використовується різна кількість молодших бітів пікселя, які використовуються для кодування інформації.

Наступним важливим кроком є перевірка роботи програми за допомогою доступних сервісів стегоаналізу. Я використав два сервіси: Aperi'Solve і Steganography Online. За допомогою першого сервісу було перевірено, як за допомогою системи стегоаналізу видно прихований текст у фото, а за допомогою другого сервісу було спробувано декодувати інформацію з фото з прихованим текстом. Для отримання більш наглядного результату, було взято фото зі значно гіршою якістю. Результати даних випробувать представлено на рисунку 4.20 і рисунку 4.21.

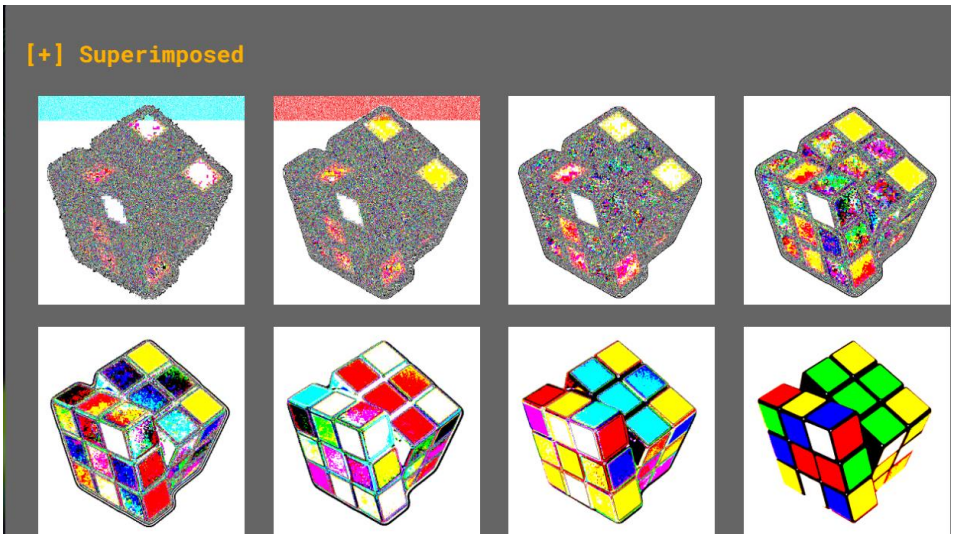

Рисунок 4.20 – Результати стегоаналізу онлайн сервісом Aperi’Steganography

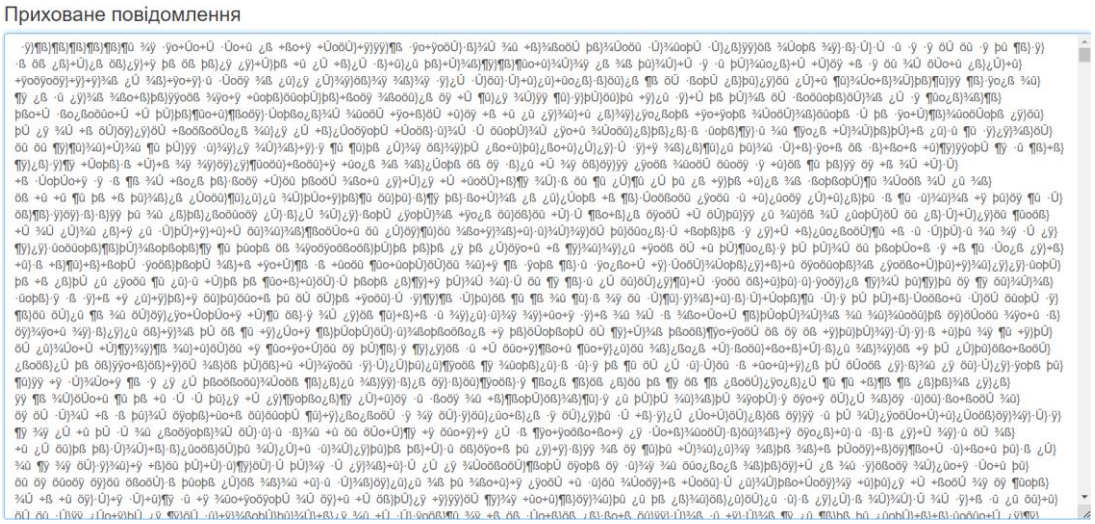

Рисунок 4.21 – Результат спроби декодування тексту з фото сервісом Steganography Online

З результатів можна зробити висновки, що хоча для людського ока не видно різниці між фото з шифротекстом і без нього, але використовуючи систему стегоаналізу, можна легко помітити, що фото має яке-небудь наповнення. Проте, за допомогою сервісу Steganography Online, не вдалося декодувати текст з фото. Це означає, що розроблена програма не є ідеальною, але все ж надає певний рівень захищеності інформації.

Висновки за розділом 3

У даному розділі було розроблено програму для приховування інформації в графічних об'єктах, використовуючи криптографічні і стеганографічні технології. Описано основні модулі програми, а також проведено тестування застосунку. Під час перевірки його працездатності помилок не виявлено. Однак, при тестуванні розробленого програмного продукту за допомогою онлайн-сервісів стегоаналізу було виявлено інформацію в графічному контейнері, але декодувати її не вдалося. Це свідчить про достатній рівень ефективності розробленого застосунку.

Вибраний алгоритм стеганографії є не поганим варіантом для використання в повсякденному житті, оскільки, кодуючи навіть велику кількість інформації в фото з доброю якістю, не спричиняються помітні зміни у вигляді зображення для органів сприйняття людини.

ВИСНОВКИ

У даній кваліфікаційній роботі було розроблено програму для кодування і декодування інформації в графічний об'єкт на основі алгоритму LSB. Мовою реалізації є Python. На першому етапі було розглянуто основні поняття стеганографії, її особливості і сутність. Також було описано види контейнерів, їх особливості і нюанси при їх використанні. На другому етапі було розглянуто і проаналізовано основні моделі і методи комп'ютерної стеганографії.

На основі розглянутої інформації було прийнято рішення використати алгоритм заміни останніх найменш значущих бітів. Оскільки зміна одного з каналів (червоного, зеленого, синього) може вплинути лише на 1/256 або менше (менше ніж 1%) відтінку кольору даної точки зображення, це не помітно для органів сприйняття людини.

Це дає можливість використовувати метод LSB для стеганографії, де інформація може бути впроваджена в останні біти кольорових компонентів зображення, зберігаючи при цьому вигляд зображення майже без помітних змін. А для уникнення витоку інформації, випадку її виявлення. Було вирішено, реалізувати шифрування тексту перед його вкрапленням у фото. Задля цього було використано модуль fernet із бібліотеки cryptography python. Fernet — це реалізація симетричної (також відомої як «секретний ключ») автентифікованої криптографії, це гарантує, що зашифрованим повідомленням не можливо маніпулювати або прочитати без ключа.

Для запобігання втрати інформації у випадку її виявлення, було вирішено застосувати шифрування тексту перед його внесенням у фото.. Для цього було використано модуль Fernet з бібліотеки cryptography в Python. Він забезпечує симетричне шифрування (відоме також як "секретний ключ") з автентифікацією, що гарантує, що зашифроване повідомлення не може бути маніпульоване або прочитане без ключа.

На кінцевому етапі даної роботи було проведено опис функціоналу розробленого застосунку і також тестування розробленої програми на виявлення помилок, і перевірку його захищеності різними системами стегоаналізу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. С.В.Мельник, С.В.Кондакова // Актуальні проблеми управління інформаційною безпекою держави : зб. матер. наук.-практ. конф. –К. : Наук.-вид. відділ НА СБ України, 2010. –С. 134-138.

2. Конахович Г. Ф. Комп'ютерна стеганографія. Теорія та практика / Г. Ф. Конахович, А. Ю. Пузиренко., 2006. – 288 с. – (К.: МК-Пресс).

3. Грибунін В. Г. Цифрова стеганографія / В. Г. Грибунін, О. Н. Ігор, Т. В. Ігор., 2002. – 272 с. – (Солон-Пресс). – (Аспекти захисту).

4. Биков С. Ф С. Ф. Основи стегоаналізу // Захист інформації. Конфідент. / С. Ф. Биков С. Ф, О. В. Мотуз. – 2000. – №3. – С. 38–41.

5. Грибунін В. Г. Цифрова стеганографія / В. Г. Грибунін, І. М. Оков, І. В. Туринців. – 2018. – 262 с. – (СОЛОН-Пресс).

6. Кузнецов О. О. СТЕГАНОГРАФІЯ / О. О. Кузнецов, С. П. Євсеєв, О. Г. Король. – Харків, 2011. – 207 с. – (ХНЕУ)..

7. Кувшинов С. С. Методи та алгоритми приховування великих обсягів даних на основі стеганографії : дис. канд. техн. наук / Кувшинов С. С., 2010. – 116 с.

8. Бернет С., Пейн С.: Криптографія. Офіційний посібник RSA Security – М. «Біном», 2012. – 325 с.

9. Венбо Мао Сучасна криптографія: теорія та практика = Modern Cryptography: Theory and Practice. – М.: «Вільямс», 2005. – 768 с.

10. Воробйов В.І., Грибунін В.Г. Теорія та практика вейвлет-перетворення. – СПб: ВУС, 2009. – 325 с.

11. Зима В. Безпека глобальних мережевих технологій / В. Зима, А. Молдов'ян, Н. Молдов'ян., 2011. – 344 с.

12. Нільс Фергюсон, Брюс Шнайєр Практична криптографія: Practical Cryptography: Designing and Implementing Secure Cryptographic Systems. – М.: «Диалектика», 2012. – С. 432.

13. Петров А. А. Комп'ютерна безпека. Криптографічні методи захисту / А. А. Петров. // М.: ДМК. – 2000. – С. 448.

14. Барич С. Г. Основи сучасної криптографії / С. Г. Барич, В. В. Гончаров, Р. Є. Сєров., 2001. – 120 с. – (Гаряча лінія – Телеком).

15. Столлінгс В. Криптографія та захист мереж: теорія та практика. М: Вільямс. 2001. Пер. с англ. – 235 с.

16. Чмора А.Л. Сучасна прикладна криптографія. 2-е вид., Стер. – М.: Геліос АРВ, 2012. – 256 с.

17. Шнайер, Брюс. Прикладна криптографія. Протоколи, алгоритми, вихідні тексти мовою Сі – М.: Видавництво ТРІУМФ, 2002 – 816 с.

18. Shapiro J. Embedded Image Coding Using Zerotrees Of Wavelet Coefficients / / IEEE Transactions on Signal Processing, 1993. – Vol. 41, No. 12.

19. Задирака В.К. Статистичний аналіз систем із цифровими водяними знаками / В.К. Задирака, Н.В. Кошкіна, Л.Л. Нікітенко // Штучний інтелект.– 2008.– № 3. – С. 315–324.

20. Suresh A. Image Texture Classification using Gray Level Co-Occurrence Matrix Based Statistical Features / A. Suresh, K.L. Shunmuganathan // European Journal of Scientific Research. – 2012. – Vol.75, № 4. – P. 591–597.

21. Voloshynovskiy S.V. Visual communications with side information via distributed printing channels: extended multimedia and security perspectives / S.V. Voloshynovskiy, O. Koval, F. Deguillaume, T. Pun // Proc. of SPIE: Security.

ДОДАТОК А

Лістинг програмного файлу STEGOPY:

```
import sys
import os
from platform import system as psystem

from PIL import Image, ImageDraw
from cryptography.fernet import Fernet
from cryptography.fernet import InvalidToken
from colorama import Fore, Style
from plyer import notification

DATA = dict()
name = sys.argv[0].split('/')[-1]
version = "v0.0.4release"



def main():
    try:
        DATA["action"] = sys.argv[1]
        DATA["image"] = sys.argv[2]

        if DATA["image"][0] != '/':
            DATA["image"] = os.getcwd() + '/' + DATA["image"]

        if not os.path.exists(DATA["image"]):
            raise FileExistsError

        if DATA["action"] == '-e':
            DATA["data"] = sys.argv[3]

            # Check directory to Data file
            if DATA["data"][0] != '/':
                DATA["data"] = os.getcwd() + '/' + DATA["data"]
```

```
        if not os.path.exists(DATA["data"]):
            raise FileExistsError
        # Check directory to Data file - END

        # Validate balance
        if isset(sys.argv, 4):
            value = int(sys.argv[4])

            if value >= 1 and value <= 4:
                DATA["balance"] = value
            else:
                raise ValueError
        # Validate balance - END

except IndexError:
    using("Encrypt: {} -e [path_to_image*] [path_to_data*] [balance]".format(name))
    using("Decrypt: {} -d [path_to_image*]".format(name))
    using("Program information: {} -i".format(name), True)

except FileExistsError:
    error("Image or data file not found.", True)

if DATA["action"] == '-e':
    if not isset(DATA, "balance"):
        try:
            DATA["balance"] = int(input(Style.BRIGHT + Fore.RED + "     Balance (1 to 4) > "))
        except ValueError:
            error("Set to 2.")
            DATA["balance"] = 2
    file = open(DATA["data"], 'r')
    text = file.read()
    file.close()

    encrypt(DATA["image"], text.strip(), Fernet.generate_key().decode(), DATA["balance"])
```

```
    elif DATA["action"] == '-d':
        key = input(Style.BRIGHT + Fore.RED + "     Key: ")
        try:
            decrypt(DATA["image"], key)
        except IndexError:
            error("Invalid key.")
        except ValueError:
            error("Invalid key.")
    print('')


def find_max_index(array):
    max_num = array[0]
    index = 0
    for i, val in enumerate(array):
        if val > max_num:
            max_num = val
            index = i
    return index


def balance_channel(colors, pix):
    max_color = find_max_index(colors)
    colors[max_color] = int(last_replace(bin(colors[max_color]), pix), 2)
    while True:
        max_sec = find_max_index(colors)
        if max_sec != max_color:
            colors[max_sec] = colors[max_color] - 1
        else:
            break
    return colors


def encrypt(path_to_image, text, key, balance):
```

```
img = dict()
size = dict()
coord = dict()

img["image"] = Image.open(path_to_image)
img["draw"] = ImageDraw.Draw(img["image"])
img["pix"] = img["image"].load()

size["width"] = img["image"].size[0]
size["height"] = img["image"].size[1]

text = des_encrypt(text, key)
binary_text = text_to_binary(text)
list_two = split_count(''.join(binary_text), balance)

coord["x"] = 0
coord["y"] = 0
count = 0

for i in list_two:
    red, green, blue = img["pix"][coord["x"], coord["y"]]
    (red, green, blue) = balance_channel([red, green, blue], i)
    img["draw"].point((coord["x"], coord["y"]), (red, green, blue))
    if coord["x"] < (size["width"] - 1):
        coord["x"] += 1
    elif coord["y"] < (size["height"] - 1):
        coord["y"] += 1
        coord["x"] = 0
    else:
        error("Message too long for this image.", True)

    count += 1

path_new = os.path.join(os.getcwd(), f'{os.path.split(path_to_image)[1].split(".")[0]}_lsb.png')
img["image"].save(path_new, "PNG")
```

```
    file = open("key.dat", "w")
    file.write(str(balance) + '$' + str(count) + '$' + key)
    file.close()

def decrypt(path_to_image, key):

    balance = int(key.split('$')[0])
    count = int(key.split('$')[1])
    end_key = key.split('$')[2]

    img = dict()
    coord = dict()

    img["image"] = Image.open(path_to_image)
    img["width"] = img["image"].size[0]
    img["height"] = img["image"].size[1]
    img["pix"] = img["image"].load()

    coord["x"] = 0
    coord["y"] = 0
    code = ''

    i = 0
    while i < count:
        pixels = img["pix"][coord["x"], coord["y"]]

        pixel = str(bin(max(pixels)))

        if balance == 4:
            code += pixel[-4] + pixel[-3] + pixel[-2] + pixel[-1]

        elif balance == 3:
            code += pixel[-3] + pixel[-2] + pixel[-1]
```

```
            elif balance == 2:
                code += pixel[-2] + pixel[-1]

            else:
                code += pixel[-1]

            if coord["x"] < (img["width"] - 1):
                coord["x"] += 1
            else:
                coord["y"] += 1
                coord["x"] = 0

            i += 1

        outed = binary_to_text(split_count(code, 8))

        try:
            des_decrypt(''.join(outed), end_key)
        except TypeError:
            return "В файле нет текста или неверный ключ!"
        return des_decrypt(''.join(outed), end_key)


    def des_encrypt(text, key):

        cipher = Fernet(key.encode())
        result = cipher.encrypt(text.encode())

        return result.decode()


    def des_decrypt(text, key):
        try:
            cipher = Fernet(key.encode())
            result = cipher.decrypt(text.encode())
```

```
    except InvalidToken:
        return
    return result.decode()

def split_count(text, count):
    result = list()
    txt = ''
    var = 0
    for i in text:
        if var == count:
            result.append(txt)
            txt = ''
            var = 0
        txt += i
        var += 1
    result.append(txt)
    return result

def last_replace(main_string, last_symbols):
    return str(main_string)[:-len(last_symbols)] + last_symbols

def text_to_binary(event):
    return ['0' * (8 - len(format(ord(elem), 'b'))) + format(ord(elem), 'b') for elem in event]

def binary_to_text(event):
    return [chr(int(str(elem), 2)) for elem in event]

def isset(array, key):

    try:
        if type(array) is list:
            array[key]
        elif type(array) is dict:
            return key in array.keys()
        return True
```

```
    except:
        return False


def error(text, quit=False):

    print(Style.BRIGHT + Fore.YELLOW + "     " + text + Style.RESET_ALL)
    if quit:
        sys.exit()


def using(text, quit=False):

    print(Style.BRIGHT + Fore.WHITE + "     " + text + Style.RESET_ALL)
    if quit:
        sys.exit()


def success(text):

    print(Style.BRIGHT + Fore.GREEN + "     " + text + Style.RESET_ALL)


if __name__ == "__main__":
    main()
```

ДОДАТОК В

ЛІСТИНГ ПРОГРАМНОГО ФАЙЛУ MAIN:

```
from os import system
from platform import system as psystem
from sys import argv, exit

from cryptography.fernet import Fernet
from PyQt5.QtWidgets import QFileDialog
from plyer import notification

from stegopy import encrypt, decrypt
from steg_tool import *

class MyWin(QtWidgets.QMainWindow):
    def __init__(self, parent=None):
        QtWidgets.QWidget.__init__(self, parent)
        self.ui = Ui_MainWindow()
        self.ui.setupUi(self)
        self.ui.imgOpenBtn.clicked.connect(self.img_open)
        self.ui.takeBtn.clicked.connect(self.get_job)
        self.ui.savehsText.clicked.connect(self.save_text)
        self.ui.keyBtn.clicked.connect(self.key_path)
        self.ui.hexexCombo.currentTextChanged.connect(self.hexex_change)
        self.ui.wreadCmb.currentTextChanged.connect(self.hexex_change)

    def hexex_change(self):
        if self.ui.hexexCombo.currentText() == "LSB" and self.ui.wreadCmb.currentText() ==
"Write":
            self.ui.balanceBox.setEnabled(True)
            self.ui.keyBtn.setEnabled(False)
        elif self.ui.hexexCombo.currentText() == "LSB" and self.ui.wreadCmb.currentText() ==
"Read":
            self.ui.balanceBox.setEnabled(False)
            self.ui.keyBtn.setEnabled(True)
```

```
    def key_path(self):
        self.ui.pathKey.setText('')
        if psystem() == "Windows":
            file = QtWidgets.QFileDialog.getOpenFileName(self, "Выбор ключа .dat", None,
                                                         "Key file (*.dat)")[0].replace('/', '\\')
        if file == '':
            return

        self.ui.pathKey.setText(file)

    def img_open(self):
        self.ui.savehsText.setEnabled(False)
        self.ui.txtSteg.setText('')
        self.ui.pathImg.setText('')
        extension = '*.png *.jpg'
        if psystem() == "Windows":
            file = QtWidgets.QFileDialog.getOpenFileName(self, "Выбор файла", None,
                                                         f"Image file ({extension})")[0].replace('/', '\\')
        if file == '':
            return
        if self.ui.wreadCmb.currentText() == "Write":
            self.ui.txtSteg.setEnabled(True)
        self.ui.pathImg.setText(file)
        self.ui.takeBtn.setEnabled(True)

    def get_job(self):
        if self.ui.hexexCombo.currentText() == "LSB" and self.ui.wreadCmb.currentText() ==
"Write":
            if self.ui.txtSteg.toPlainText():
                encrypt(self.ui.pathImg.text(), self.ui.txtSteg.toPlainText(),
Fernet.generate_key().decode(),
                        int(self.ui.balanceBox.currentText()))
                self.ui.balanceBox.setEnabled(False)
```

```
                self.ui.pathImg.setText('')
                self.ui.takeBtn.setEnabled(False)
                self.ui.txtSteg.setText('')
                self.ui.txtSteg.setEnabled(False)
            elif self.ui.hexexCombo.currentText() == "LSB" and self.ui.wreadCmb.currentText() ==
"Read":
                if self.ui.pathKey.text():
                    with open(self.ui.pathKey.text(), 'r') as k:
                        key = k.read()
                    dec = decrypt(self.ui.pathImg.text(), key)
                    if dec is None:

                        self.ui.savehsText.setEnabled(False)
                        self.ui.takeBtn.setEnabled(False)
                    else:
                        self.ui.txtSteg.setText(dec)
                        self.ui.pathImg.setText('')
                        self.ui.takeBtn.setEnabled(False)
                        self.ui.pathKey.setText('')
                        self.ui.savehsText.setEnabled(True)

        def save_text(self):
            ok = QFileDialog.getSaveFileName(self, "Сохранить файл", ".", "(*.txt)")
            path_s = ok[0].replace("/", "\\") + ok[1]
            if path_s:
                with open(path_s, 'w', encoding='utf-8') as text:
                    text.write(self.ui.txtSteg.toPlainText())
                self.ui.txtSteg.setText('')
                self.ui.savehsText.setEnabled(False)
    if __name__ == '__main__':
        app = QtWidgets.QApplication(argv)
        myapp = MyWin()
        myapp.show()
        exit(app.exec_())
```