МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВОЛИНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ЛЕСІ УКРАЇНКИ
Кафедра комп’ютерних наук та кібербезпеки

БУТКЕВИЧ БОГДАН ОЛЕКСАНДРОВИЧ

ПРОГРАМНА РЕАЛІЗАЦІЯ МОДЕЛЕЙ СТЕГАНОГРАФІЇ ДЛЯ ПРИХОВУВАННЯ КОНФІДЕНЦІЙНОЇ ІНФОРМАЦІЇ У ГРАФІЧНИХ ФАЙЛАХ

Спеціальність: 122 Комп’ютерні науки
Освітньо-професійна програма: Комп’ютерні науки та інформаційні технології
Кваліфікаційна робота на здобуття освітнього ступеня «бакалавр»

Науковий керівник:
Гришанович Тетяна Олександрівна,
доцент кафедри комп’ютерних наук
та кібербезпеки

РЕКОМЕНДОВАНО ДО ЗАХИСТУ
Протокол No_____
засідання кафедри комп’ютерних наук
та кібербезпеки
від _____ 2024 р.
Завідувач кафедри
(_____) Гришанович Т. О.

Луцьк – 2024

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ

DCT – дискретне косинусне перетворення (англ. Discrete Cosine Transform)

LSB – найменш значущий біт (англ. Least Significant Bit)

PVD – різниця піксельних значень (англ. Pixel-Value Differencing)

ВСТУП

Інформація є одним з найцінніших сутностей сучасного життя. Доступ до неї є предметом та метою діяльності інформаційних спільнот. Досить часто при зберіганні та передачі даних, виникає потреба приховування від небажаних втручання, зміни та викрадення інформації.

Для захисту передачі даних використовують два різних підходи:

Приховування інформації шляхом її шифрування;

Приховування самого факту існування конфіденційної інформації у повідомленні.

Не існує абсолютно надійного способу зашифрувати інформацію. Кращим способом захистити інформацію – це приховати сам факт її існування. Приховування наявності закодованої інформації є предметом методів стеганографії.

Метою кваліфікаційної роботи є програмна реалізація деяких методів стеганографії для приховування конфіденційної інформації у графічних файлах, проведення аналізу, оцінка якості та ефективності реалізованих методів.

Сучасна стеганографічна система – це сукупність засобів та методів із застосуванням комп’ютерних технологій, які використовуються для формування таємних каналів зв’язку та ефективної підтримки інформаційної безпеки. Інформаційна безпека – пріоритет сучасної комунікації. Отже, тема кваліфікаційній роботи є **актуальною**.

Об’єктом дослідження в даній кваліфікаційній роботі є цифрові методи сучасної стеганографії.

Предметом дослідження є програмна реалізація окремих з них для приховування конфіденційної інформації у графічних файлах.

Результати роботи були представлені на Міжнародній науково-практичній конференції “Проблеми комп’ютерних наук, програмного моделювання та безпеки цифрових систем” 13-16 червня 2024 року.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ОСНОВИ ВИКОРИСТАННЯ СТЕГАНОГРАФІЇ

1.1. Історичні відомості про становлення стеганографії

Стеганографія може бути визначена як спосіб приховування інформації шляхом вставки повідомлень в інші, начебто звичайні тексти, графічні, звукові, загалом медіафайли.

Перша стеганографічна техніка приписується Стародавній Греції [4] (близько 440 року до н.е.). Грецькі правителі використовували ранню версію стеганографії, яка, наприклад, полягала в наступному: гоління голови раба, татуювання повідомлення на шкірі голови, очікування зростання волосся і відправлення раба для доставки таємного повідомлення. Одержувач повинен поголити голову раба, щоб прочитати повідомлення і може відповісти у тій самій, або іншій формі стеганографії.

Стеганографія продовжує з часом розвиватися. Особливо активно стеганографія використовувалась під час військових конфліктів. Наприклад, під час американських війн британські та американські сили використовували різні види невидимих чорнил. В склад невидимих чорнил зазвичай входило молоко, оцет, фруктовий сік тощо. Щоб розшифрувати приховані повідомлення, написані невидимим чорнилом, потрібно було застосувати світло та/або тепло.

1.1. Визначення стеганографії

Основна мета стеганографії – приховати факт передачі конфіденційної інформації, на відміну від криптографії, яка займається забезпеченням конфіденційності та цілісності даних, але не приховує сам факт існування цієї інформації [3].

Можна сказати, що стеганографія – це один із шляхів підтримки інформаційної безпеки. Вона являє собою метод організації зв'язку, який приховує сам факт наявності таємних повідомлень. Стеганографічні методи активно використовуються для захисту інформації від сторонніх користувачів та для маскування програмного забезпечення.

Згідно з прийнятою термінологією стеганографічна система або стегосистема – це сукупність засобів та методів, які використовуються для формулювання таємного каналу зв'язку. Будь-яка інформація, в якій приховані таємні дані, називається контейнером. За контейнер може слугувати будь-який файл чи потік даних. Контейнер, який не містить таємного повідомлення, називають порожнім, а той, що містить – заповненим або стегоконтейнером [3, 4]. Канал передачі сгегоконтейнера має назву стеганографічного каналу або стегоканалу. Таємний ключ, який необхідний для приховування інформації в контейнер, називається стегоключем або просто ключем. Залежно від кількості рівнів захисту в стегосистемі може використовуватись як один, так і декілька ключів.

Схема типової стеганографічної системи зображена на рисунку 1.1.

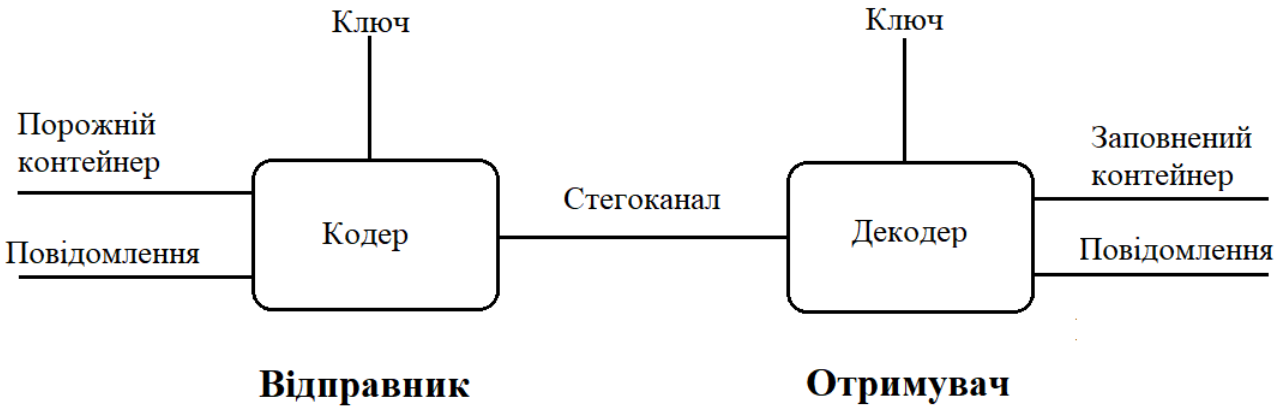

Рис. 1.1 - Схема типової стегосистеми

Повідомлення – це термін, що використовується для загальної назви прихованої інформації, що передається, будь то лист або цифровий файл.

Контейнер – так називається будь-яка інформація, яка використовується для приховування повідомлення.

Порожній контейнер – контейнер, який не містить прихованого повідомлення.

Заповнений контейнер (стегоконтейнер) – контейнер, що містить приховане повідомлення.

Стеганографічний канал (стегоканал) – канал передачі стегоконтейнера.

Ключ (стегоключ) – секретний ключ, необхідний для приховування стегоконтейнера. Ключі в стегосистемах бувають двох типів: секретні та відкриті. Якщо стегосистема використовує секретний ключ, то він має бути створений або до початку обміну повідомленнями, або переданий захищеним каналом. Стегосистема, що використовує відкритий ключ, повинна бути влаштована таким чином, щоб неможливо було отримати з нього закритий ключ. У цьому випадку відкритий ключ ми можемо передавати незахищеним каналом.

Кодер – пристрій, призначене для здійснення вкладення прихованого повідомлення у контейнер з урахуванням їхньої моделі.

Декодер – пристрій, який відновлює приховане повідомлення.

Комп’ютерна стеганографія розвивається у кількох напрямках, кожен з яких має як багато спільних рис, так і деякі унікальні відмінності, зумовлені специфікою практичного використання. Серед стеганографічних систем можна виділити системи прихованої передачі даних, цифрових водяних знаків, ідентифікаційних номерів (“відбитків пальців”) та заголовків. Основне завдання будь-якої стеганографічної системи – приховати повідомлення в контейнері так, щоб стороння особа не змогла помітити різниці між зміненим контейнером та оригіналом. Зазвичай, стеганографічна система розробляється з урахуванням

необхідного балансу її основних характеристик, таких як непомітність, стійкість, безпека, пропускна здатність та обчислювальна складність.

Системи прихованої передачі даних використовуються для забезпечення таємного зв’язку. Вони відрізняються від інших тим, що оригінальний вміст контейнера не має значення ні для відправника, ні для одержувача, яких цікавить лише успішна передача прихованого повідомлення. При цьому важливо, щоб факт відправлення стегоконтейнера не викликав підозр, і не було помітних відхилень контейнера від норми. Головна мета таких систем – приховати наявність стеганографічного каналу, зробити неможливим розрізнення порожніх і заповнених контейнерів без знання ключа. Для цих систем зазвичай передбачається, що контейнер не буде спотворений під час передачі через канал зв’язку оскільки таємний зв’язок здійснюється через відкритий цифровий канал мережі, наприклад, Інтернет, що гарантує відсутність спотворень інформації під час передачі.

1.2. Класифікація методів стеганографії

В стеганографії виділяють декілька напрямів: класична стеганографія, комп’ютерна, цифрова [3, 12]. Кожен із напрямів є доповненням класичної стеганографії та бере початки з давніх часів.

Кожен різновид стеганографії має свої унікальні характеристики та можливості використання в різних сферах, включаючи кібербезпеку, ділову та військову розвідку, інформаційну безпеку та інші.

Класична (традиційна) стеганографія – спосіб приховування даних, що здійснюється за допомогою технічних засобів захисту інформації. Сучасна класична стеганографія (рис. 1.2) включає в себе хімічні та фізичні методи.

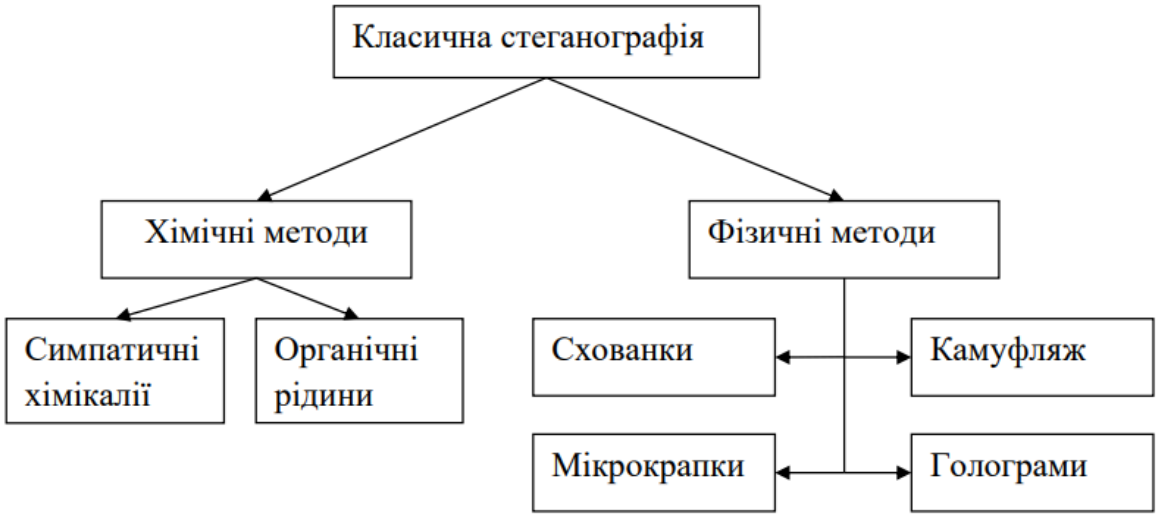

Рис. 1.2 - Методи класичної стеганограії

Загалом, хімічні методи стеганографії зводяться до застосування невидимого чорнила. До цих методів відносяться симпатичні хімікалії і органічні рідини. Симпатичні хімікалії є одним з найбільш поширених методів класичної стеганографії. Зазвичай, процес запису здійснюється наступним чином: перший шар – наноситься важливий запис невидимим чорнилом, другий шар – нічого не значущий запис видимим чорнилом. Текст записаний таким чином, що проявляється тільки при певних умовах (нагрівання, освітлення, хімічний проявник тощо). Органічні рідини мають схожі властивості з симпатичними хімікаліями: при нагріванні вони змінюють забарвлення (в них міститься велика кількість вуглецю).

До фізичних методів можна віднести різного виду схованки, методи камуфляжу та мікрокрапки.

Використання мікрокрапок є однією з традиційних технік стеганографії, що використовується для приховування інформації в тексті або документах. Мікрокрапки - це дрібні частки або точки, які невидимі за звичайних обставин і вимірюються міліметрами чи навіть мікрометрами. Вони можуть бути друковані

на документах або приховані в тексті так, що навіть при дуже докладному ретельному огляді їх важко помітити [3].

Метод голограми полягає в тому, що в зображення-контейнер вбудовується не самі конфіденційні дані, а їх голографічне зображення. Цей підхід забезпечує найвищий рівень стійкості до злому. Використання голографічного методу дозволяє вбудовувати конфіденційні дані в звичайні фотографії на папері або пластику. Основний недолік цього методу полягає в обмеженому обсязі даних, які можна вбудувати. Голографічний підхід найкраще підходить для приховування невеликих зображень, відновлення яких може допустити незначну втрату якості, наприклад, підписів чи відбитків пальців.

Метод камуфляжу полягає в маскуванні конфіденційного повідомлення таким чином, щоб воно зливалося із забарвленням предмета, який використовується як контейнер.

До переваг класичної стеганографії належить доступність засобів реалізації, а основними недоліками є складність практичної реалізації та ризик випадкового виявлення прихованого повідомлення.

Комп’ютерна стеганографія – це галузь стеганографії, яка реалізується на основі комп’ютерної техніки і відповідного програмного забезпечення [4, 12]. Одним із методів комп’ютерної стеганографії є використання зарезервованих полів комп’ютерних форматів файлів. Суть методу полягає в тому, що частина поля розширень, не заповнена інформацією про розширення, за замовчуванням заповнюється нулями. Відповідно ми можемо використовувати цю “нульову” частину для запису своїх даних. Недоліком цього методу є низький ступінь прихованості і обмеження на обсяг переданої інформації.

Існують також інші методи комп’ютерної стеганографії, зокрема, метод приховування інформації в невикористовуваних місцях гнучких дисків, метод використання особливих властивостей полів форматів, які не відображаються на екрані тощо.

Цифрова стеганографія. У контексті цифрових технологій стеганографія використовується для приховування або вбудовування додаткової інформації в цифрові об’єкти, такі як зображення, аудіофайли чи відео. Як правило, дані об’єкти є мультимедійними і внесення спотворень, які знаходяться нижче порога чутливості людини, не призводить до їх помітних змін [1, 4]. Методи цифрової стеганографії наведені на рисунку 1.3.

Рис. 1.3 - Методи цифрової стеганографії

Приховування даних у нерухомих зображеннях може здійснюватися кількома способами. Один з них – приховування в просторовій області, де дані вбудовуються безпосередньо в пікселі зображення шляхом зміни значень найменш значущих бітів. Інший метод – приховування в частотній області, який передбачає перетворення зображення у частотну область за допомогою, наприклад, дискретного косинусного перетворення (DCT), і вбудовування даних у коефіцієнти частотних компонентів. Ще один підхід – розширення спектру, де дані

розподіляються по всьому зображенню, роблячи їх менш помітними і стійкими до різних атак.

У випадку приховування даних в аудіосигналах також використовуються кілька методів. Наприклад, кодування найменш значущих бітів (LSB) вбудовує дані у найменш значущі біти аудіосигналу. Метод фазового кодування змінює фазу аудіосигналу для приховування даних. Використовується також розширення спектру, що дозволяє робити дані менш помітними. Інший цікавий метод – приховування з використанням ехо-сигналу, де до основного аудіосигналу додаються слабкі ехо-сигнали, що містять приховані дані.

Для приховування даних у тексті застосовуються синтаксичні та семантичні методи, де дані приховуються шляхом зміни синтаксису або семантики тексту. Це може включати перестановку слів, заміну синонімів або зміну розмітки. Крім того, використовуються методи довільного інтервалу, які включають використання різних інтервалів між словами або літерами для приховування даних [4].

Таким чином, цифрова стеганографія пропонує різні підходи для приховування даних в зображеннях, аудіосигналах і тексті, забезпечуючи різний рівень непомітності, стійкості та складності реалізації.

1.3. Критерії оцінки якості методів стеганографії

Будь-яка стеганографічна система характеризується трьома якісними параметрами, а саме ємність, непомітність і стійскість [1, 2].

Ємність показує, скільки даних може бути приховано в носії. Ця характеристика залежить від стеганографічного алгоритму та властивостей контейнера. Існують різні метрики для вимірювання місткості, для прикладу кількість бітів секретного повідомлення, які можна вбудувати в один піксель

зображення або відношення розміру секретного повідомлення до максимального розміру повідомлення, що може бути вбудоване в даний контейнер [8].

Непомітність є найважливішим аспектом оцінки якості методу приховування інформації. Непомітність оцінюється як різниця між стегоконтейнером і оригінальним контейнером. Існує низка показників для оцінки відмінностей стегоконтейнера від оригінального контейнера. До таких показників відносять сенсорні і статистичні.

Стійкість означає невразливість до модифікацій стегоконтейнера, наприклад, стиснення, перетворення в інший формат або часткового пошкодження тощо. Це здатність зберігати приховану інформацію у первісному вигляді навіть після різних видів обробки, таких як фільтрація, додавання шуму, зміни розміру, повороти, або інші геометричні та нелінійні перетворення. Стійкість також включає здатність протистояти спробам виявлення та витягнення прихованих даних за допомогою статистичних або криптографічних методів.

Усі ці параметри можуть бути конфліктними, їх взаємовплив та взаємозв’язок залежить від конкретних застосувань. Якщо потрібна велика ємність то більша частина носія використовується для секретного зберігання даних. Це дозволяє передавати більше даних, але водночас призводить до більших спотворень файлу-носія. Таким чином, оптимальний вибір параметрів залежить від конкретних вимог і пріоритетів стеганографічного застосування (рис. 1.4).

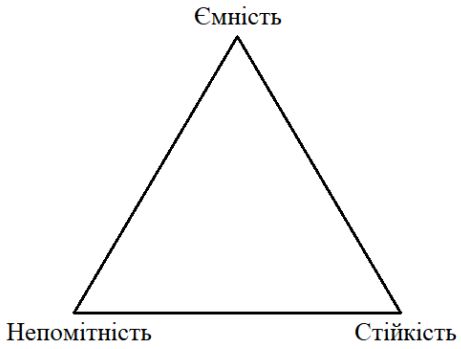

Рис. 1.4 - Взаємозв’язок між ємністю, непомітністю та стійкістю

Існують кількісні критерії, які обчислюються на основі значень пікселів зображення, що дозволяє здійснювати оцінку ефективності стеганографічних методів. Наведемо декілька з них, які використаємо для виконання порівняльного аналізу [10].

PSNR (Peak Signal-to-Noise Ratio). Це виміряне у децибелах відношення максимальної потужності сигналу до потужності шуму. Використовується для оцінки якості візуальної стійкості стеганографічних методів.

Високе значення $PSNR$ свідчить про те, що якість зображення після приховування інформації залишається високою, тобто зміни є непомітними для людського ока.

$PSNR$ визначається за формулою: $PSNR = 10 \times \frac{255^2}{MSE}\,(dB)$, де MSE – середня квадратична помилка між оригінальним зображенням та стегозображенням. Це метрика, яка визначає наскільки сильно змінені пікселі у стеганографічному зображенні порівняно з оригінальним. Мала величина MSE свідчить про те, що зміни в зображенні є незначними, що вказує на високу якість стеганографічного зображення.

Для оригінального зображення, ширина та висота якого дорівнює w та h, MSE визначається: $MSE = \frac{1}{w \times h}\sum_{i=1}^{w}\sum_{j}^{h}(Stego(i,\ j) - Original\,(i,\ j))^2$,

де i – значення пікселів стего та оригінального(носія) зображення відповідно.

SSIM (Structural Similarity Index Measure) вимірює структурну схожість між стегоконтейнером і оригінальним контейнером. Він дає значення від -1 до 1, де 1 вказує на ідеальну схожість [10].

Індекс структурної схожості (SSIM) виділяє із зображення три ключові характеристики, на основі яких виконується порівняння яскравості, контрасту та структури. Система вимірювання структурної схожості наведена на рисунку 1.5 , де Signal x і Signal y – оригінальне та модифіковане зображення.

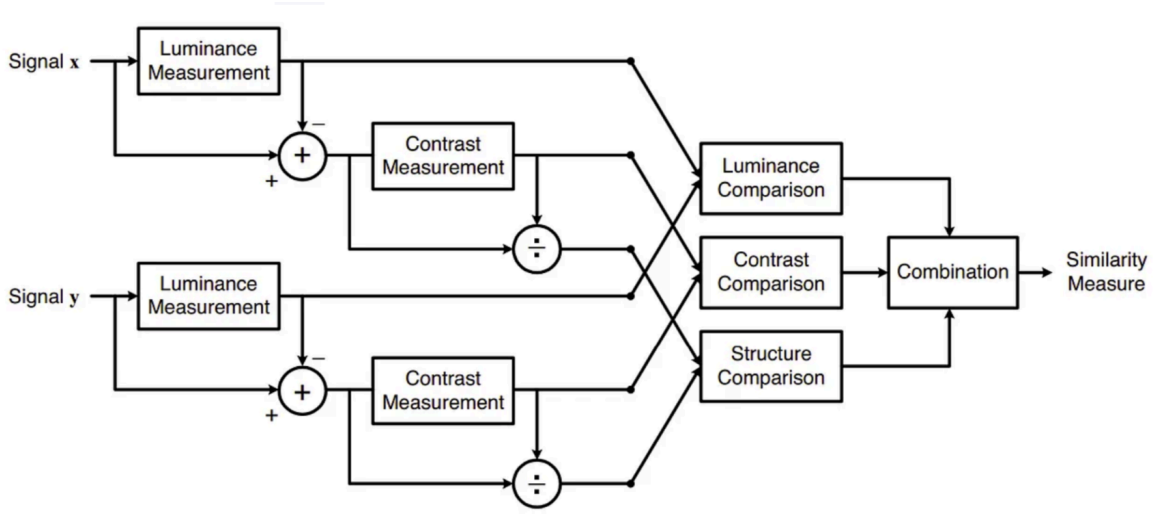

Рис. 1.5 - Система вимірювання структурної схожості

Яскравість (Luminance) вимірюється шляхом усереднення усіх значень пікселів області. Її позначають μ, обчислюється як $\mu_x = \frac{1}{N}\sum_{i=1}^{N} x_i$, де x_i – значення пікселя i зображення x. N- загальна кількість пікселів області.

Контраст (Contrast) вимірюється квадратичним відхиленням (корінь квадратний із дисперсії для вибірки усіх значень пікселів). Позначається σ і обчислюється за формулою $\sigma_x = \left(\frac{1}{N-1}\sum_{i=1}^{N}\left(x_i - \mu_x\right)^2\right)^{\frac{1}{2}}$, де x – область зображення, μ – усереднене значення усіх значень пікселів.

Функція порівняння яскравості (Luminance Comparison) $l(x,\ y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}$, де x, y – зображення, які порівнюються, C_1 = 6.5025 [10].

Функція порівняння контрасту (Contrast Comparison) $c(x,y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}$, x і y зображення, які порівнюються, C_2 = 58.5225.

Функція порівняння структури (Contrast Comparison) $s(x,\ y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3}$, де $\sigma_{xy} = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_x)(y_i - \mu_y)$, $C_3 = \frac{C_2}{2}$.

Індекс структурної схожості обчислюється за формулою: $SSIM(x,y) = [l(x,y)]^{\alpha}[c(x,y)]^{\beta}[s(x,y)]^{\gamma}$, де $\alpha > 0$, $\beta > 0$, $\gamma > 0$ позначають відносну важливість кожного з показників. Якщо усі показники мають одну вагу, то $\alpha = \beta = \gamma = 1$ [10].

NCC (Normalized Cross-Correlation) – нормована взаємна кореляція. Описує зв'язок між незаповненим та заповненим контейнером. Чим ближче значення цього критерію до 1, тим подібніші вихідне та модифіковане зображення.

Нормована взаємна кореляція $NCC(I,J) = \frac{\sum_{xy}(I(x,y) - \grave{I})(J(x,y) - \grave{J})}{\sqrt{\sum_{x,y}(I(x,y) - \grave{I})^2 \sum_{x,y}(J(x,y) - \grave{J})^2}}$, де $I(x,y)$ і $J(x,y)$ – значення пікселів зображень I, J у позиціях (x,y). $\grave{I}$, $\grave{J}$ - середні значення пікселів зображень I та J відповідно.

Існують також інші кількісні критерії оцінки якості та ефективності стеганографічних методів [7, 10].

Кількісна оцінка стійкості стеганографічної системи до зовнішніх втручань вимагає застосування методів системного аналізу, математичного моделювання або експериментального дослідження. Надійна стеганосистема вирішує дві основні

задачі: перша полягає у приховуванні самого факту існування повідомлення (перший рівень захисту), а друга - у запобіганні несанкціонованому доступу до інформації шляхом вибору відповідного методу приховування інформації (другий рівень захисту). Також можливий третій рівень захисту – застосування криптографічного захисту повідомлення (шифрування).

Модель аналізу загроз та оцінки стійкості стеганосистем складається з декількох ключових елементів. У центрі моделі знаходиться стеганосистема, яка включає в себе стеганографічні методи і технології, спеціалізоване програмне забезпечення та стеганографічні засоби. Важливим компонентом стеганосистеми є повідомлення для приховування, яке підлягає захисту [3, 5].

Модель аналізу загроз та оцінки стійкості стеганосистем наведена на рисунку 1.6.

Основними загрозами для стеганосистеми є виявлення, заміна, витяг, виділення, блокування передачі та інші можливі загрози. Крім того, стеганосистема використовує стегано-ключ і контейнер для захисту інформації.

Види атак на стеганосистему дуже різноманітні. Серед них атаки, пов’язані з компресією із втратою даних, спрямовані на знищення прихованих даних (фільтрація), а також геометричні перетворення, такі як поворот, масштабування та фрагментація. Інші атаки базуються на відомих характеристиках контейнерів або повідомлень: відомий заповнений контейнер, відоме вбудоване повідомлення, зворотне приховане повідомлення, зворотний заповнений контейнер, відомий порожній контейнер, зворотний порожній контейнер, а також на основі відомої математичної моделі контейнера або його частини.

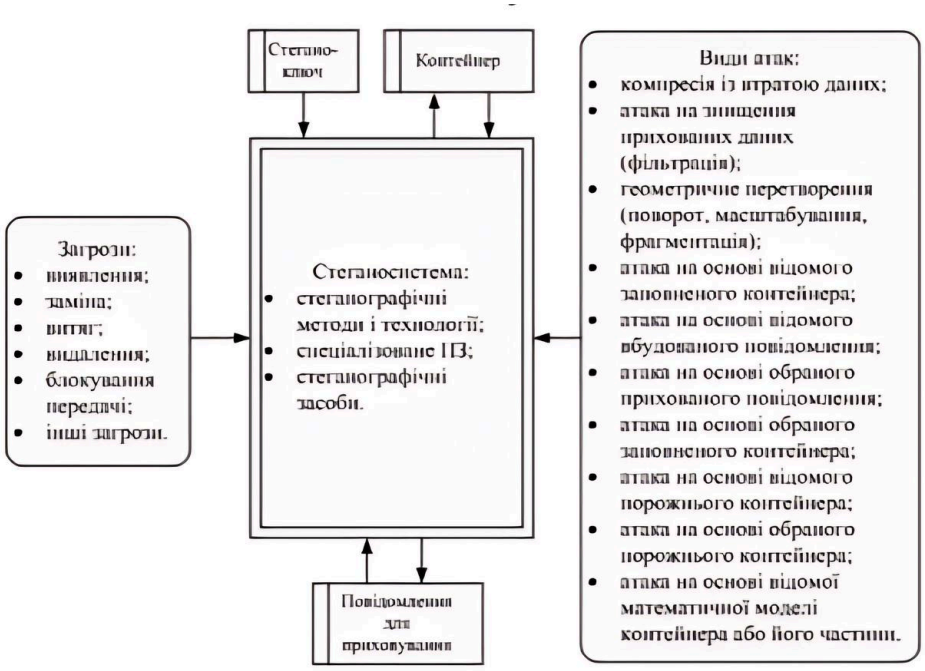

Рис. 1.6 - Модель аналізу загроз та оцінки стійкості стеганосистеми

РОЗДІЛ 2
ПРОЄКТУВАННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ ПРИХОВУВАННЯ КОНФІДЕНЦІЙНОЇ ІНФОРМАЦІЇ У ГРАФІЧНИХ ФАЙЛАХ

1.1. Постановка задачі

Метою даної роботи є застосування методів приховування текстової інформації у графічних файлах та їх програмна реалізація, яка дозволить приховувати секретну інформацію з високою надійністю та мінімальним впливом на якість зображення.

В основу роботи були покладені наступні етапи:

- дослідження та аналіз існуючих методів стеганографії, зокрема, алгоритмів LSB, DCT, PVD;
- опис алгоритмів для кожного з обраних методів приховування інформації;
- реалізація стеганографічних алгоритмів у вигляді програмних рішень засобами мови програмування Python;
- проведення тестування та аналіз якості і ефективності впроваджених рішень.

Таким чином, програмна реалізація (програмний застосунок) має забезпечити ефективне приховування конфіденційних даних в зображенні з використанням методів LSB, DCT, PVD.

Вимоги до програмного застосунку:

1. Застосунок повинен забезпечувати ефективне приховування даних, забезпечуючи достатню (велику) ємність для прихованої інформації, зберігаючи при цьому високу якість зображення.

2. Зміни, внесені програмним застосунком повинні бути непомітними для людського сприйняття. Візуальні артефакти або зміни в якості зображення повинні бути мінімізовані.
3. Застосунок повинен бути сумісним із різними форматами зображень, такими як JPEG, PNG, BMP та іншими популярними форматами.
4. Застосунок повинен забезпечити певний рівень стійкості до випадкових небажаних модифікацій стегоконтейнера.

1.2. Опис алгоритмів приховування інформації в графічних файлах

Метод LSB (Least Significant Bit). Основна ідея методу полягає в тому, що значення молодших бітів пікселів зображення можуть бути змінені з мінімальним впливом на якість зображення, оскільки вони вносять найменший вклад у сприйняття людиною [9, 11].

Процес приховування відбувається шляхом заміни найменших значущих бітів пікселів зображення відповідними бітами конфіденційної інформації. Наприклад, якщо ми хочемо приховати повідомлення представлене послідовністю “10101101” в контейнер зображення, ми можемо замінити найменш значущі біти кожного пікселя зображення на відповідні біти з цієї послідовності (рис. 2.1).

Алгоритм приховування даних методом LSB опишемо наступним чином.

Крок 1. Зчитуємо зображення.

Крок 2. Секретне повідомлення яке маємо приховати у зображення, перетворюємо у двійковий код.

Крок 3. Визначаємо молодші біти пікселів зображення для кожного каналу кольорів.

Крок 4. Замінюємо послідовно найменш значущі біти на відповідні біти повідомлення, що має бути приховане в даному контейнері.

Для вилучення повідомлення, спочатку зчитуємо зображення. Визначаємо молодші біти пікселів зображення для кожного каналу кольорів. Записуємо отримані значення і перетворюємо їх двійковий код у текстове повідомлення.

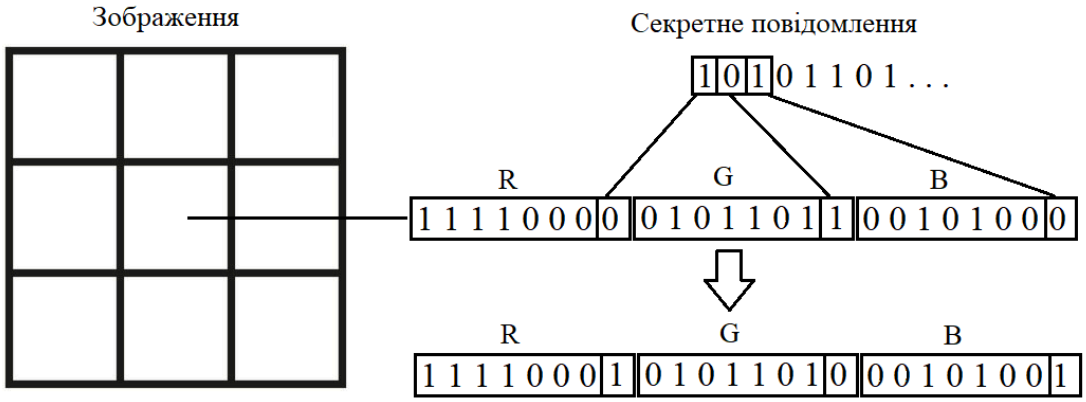

Рис. 2.1 - Алгоритм приховування даних методом LSB

Метод DCT(Discrete Cosine Transform). Даний метод використовує косинусне перетворення для розкладання сигналу на набір коефіцієнтів, які відображають його частотний склад.

Для двовимірного зображення $f(x, y)$, де $x = 0, 1, 2, ..., N - 1$ і $y = 0, 1, 2, ..., M - 1$, дискретні коефіцієнти обчислюється для кожної двомірної просторової області – блоку розміром $N \times M$ пікселів. Формула обчислення дискретних коефіцієнтів $F(u, v)$ для двовимірного зображення виглядає наступним чином:

$$F(u,v) = \sum_{x=0}^{N-1} \sum_{Y=0}^{M-1} f(x,y) \cos cos \left(\frac{\pi}{N}\left(x + \frac{1}{2}\right)u\right) \cos cos \left(\frac{\pi}{M}\left(y + \frac{1}{2}\right)v\right),$$

де $F(u, v)$ – значення DCT-коефіцієнта для частот u та v.

Компоненти зображення можуть бути високої, середньої та низької частоти У низькочастотному піддіапазоні велика частина сигналу лежить у низькій частоті, яка містить найважливіші візуальні частини зображення, тоді як у

високочастотному піддіапазоні високочастотні компоненти зображення зазвичай видаляються за допомогою стиснення та шумових атак. Таким чином, секретне повідомлення вбудовується шляхом зміни коефіцієнтів піддіапазону середніх частот (рис. 2.2) [14].

Після завершення вбудовування застосовується зворотній алгоритм DCT для перетворення сигнальних коефіцієнтів назад у просторову область.

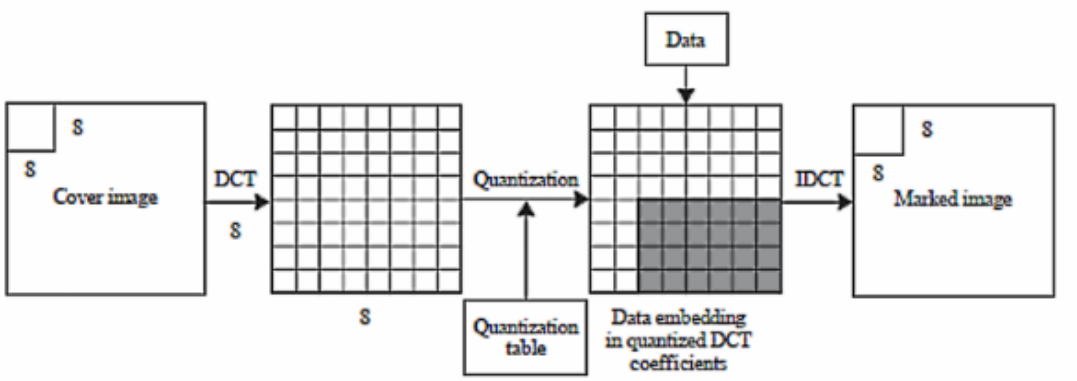

Рис. 2.2 - Алгоритм приховування даних методом DCT

Алгоритм приховування даних методом DCT опишемо наступним чином:

Крок 1. Зчитуємо зображення.

Крок 2. Перетворюємо секретне повідомлення у двійковий код.

Крок 3. Розбиваємо зображення на блоки розміром 8x8 пікселів. Від кожного блоку віднімаємо 128 (Віднімання 128 зміщує діапазон значень до -128..127. Центрування даних навколо нуля дозволяє перетворити сигнал так, щоб середнє значення було близьким до нуля. Це покращує ефективність DCT, оскільки косинусні функції в DCT краще працюють з даними, розподіленими симетрично навколо нуля).

Крок 4. Для кожного блоку застосовується DCT.

Крок 5. До кожного блоку виконуємо стиснення за допомогою таблиці квантування.

Крок 6. Замінюємо найменш значущі біти DC коефіцієнтів на біти секретного повідомлення.

Крок 7. Застосовуємо зворотній алгоритм DCT для перетворення сигнальних коефіцієнтів назад у просторову область.

Для вилучення секретного повідомлення із стегозображення, використовуємо ті самі кроки 1-4, що й для алгоритму приховування. Вираховуємо найменш значущі біти DC коефіцієнтів та записуємо їх у рядок. Далі перетворюємо бінарний код у текстовий формат.

Метод PVD (Pixel Value Differencing) забезпечує високу непомітність стегозображення шляхом вибору двох послідовних пікселів і розробки таблиці діапазону квантування для визначення наповненості за значенням різниці між послідовними пікселями. Метод пропонує перевагу передачі великої кількості даних, одночасно зберігаючи візуальні та статистичні характеристик зображення, після вбудовування даних.

Зображення розділене на неперекриваючі блоки з послідовних пікселів. З кожного блоку ми отримуємо значення різниці $d = \left|p_1 - p_2\right|$; тоді d коливається від 0 до 255. Якщо d маленьке значення, то блок розташований у гладкій області та зможе взяти на себе менше секретних даних. В іншому випадку, коли d велике значення, блок розташовується на крайовій ділянці, і в ньому може бути прихована більша кількість секретних даних. Таблиця діапазонів квантування розроблена з безперервними діапазонами, а таблиця діапазонів становить від 0 до 255. Кількість секретних бітів, прихованих у двох послідовних пікселях, залежить від таблиці діапазону квантування [13, 15].

Алгоритм приховування даних опишемо наступним чином:

Крок 1. Зчитуємо та розділяємо зображення на окремі канали (червоний, зелений, синій). У кожному каналі зображення утворюємо неперекриваючі блоки з

двох послідовних пікселів за допомогою зигзагоподібного сканування (рис. 2.3). Для кожного блоку обчислюємо значення різниці пікселів $d_i = |p_i - p_{i+1}|$.

Крок 2. За допомогою таблиці діапазону квантування для d визначаємо, скільки секретних бітів можна приховати у даному блоці. Отримуємо діапазон R_i, в якому $R_i = [l_i, u_i]$, де l_i та u_i – нижня та верхня межі R_i,

$m = \lfloor (u_i - l_i) \rfloor$ – кількість бітів вбудовування.

Крок 3. Зчитуємо біти секретного повідомлення m і перетворюємо в десяткове значення b.

Крок 4. Визначаємо різницю $d_i^{`} = l_i + b_i$. $d_i$ та $d_i^{`}$ повинні бути в діапазоні R_i.

Крок 5. Нові значення пікселів $p_i^{`}$ та $p_{i+1}^{`}$ обчислюємо за наступною формулою:

$$\left(p_i^{`}, p_{i+1}^{`}\right) = \{\left(p_i + \lceil \frac{\left|d_i^{`} - d_i\right|}{2} \rceil, p_{i+1} - \lfloor \frac{\left|d_i^{`} - d_i\right|}{2} \rfloor\right), if\ p_i \geq p_{i+1},\ d_i^{`} > d_i,\ \left(p_i - \lceil \frac{\left|d_i^{`} - d_i\right|}{2} \rceil, p_{i+1} +$$

Повторюємо кроки 1-5, доки бінарне секретне повідомлення повністю не буде приховане у зображення. Алгоритм приховування секретного повідомлення методом PVD зображено на рисунку 2.4.

Для вилучення секретного повідомлення із стегозображення, використовуємо кроки 1 та 2 що й для алгоритму приховування. Для кожного блоку обчислюємо значення різниці пікселів $d_i^{`} = \left|p_i^{`} - p_{i+1}^{`}\right|$ у стегозображенні, далі виконуємо пошук за таблицею діапазону квантування, щоб знайти $l_i$. Рахуємо $b = d_i^{`} - l_i$, переводимо у бінарний формат. Повторюємо, доки всі дані не будуть вилучені.

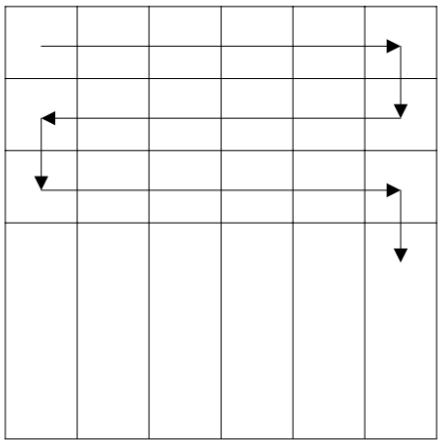

Рис. 2.3 - Зигзагоподібне сканування

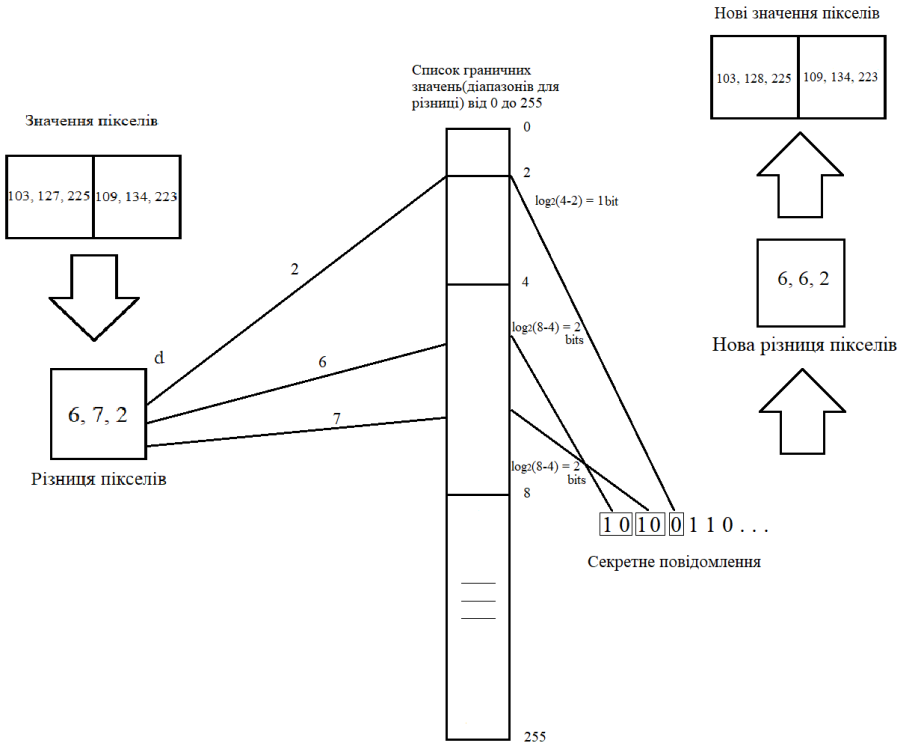

Рис. 2.4 - Алгоритм приховування повідомлення методом PVD

1.3. Обґрунтування вибору інструментальних засобів розробки

Visual Studio Code (VS Code) – це потужне, безкоштовне і кросплатформне середовище розробки, саме його було обрано для реалізації даного проєкту. Дане середовище підтримує широкий спектр мов програмування, включаючи Python.

Інтерфейс VS Code є інтуїтивно зрозумілим і налаштовується під індивідуальні потреби розробника. Це дозволяє швидко і ефективно працювати з проєктами будь-якої складності. Вбудований термінал, дебаггер, підсвічування синтаксису і автодоповнення роблять процес написання коду більш комфортним і менш схильним до помилок.

Visual Studio Code підтримує роботу з різноманітними форматами файлів та робочими процесами, що є необхідним для повноцінної розробки, тестування та документування проєктів. Це включає підтримку файлів конфігурацій, файлів проєктів, скриптів автоматизації та інше.

VS Code має вбудовану підтримку для Git та інших систем контролю версій. Це дозволяє легко відслідковувати зміни в коді, керувати гілками та об’єднувати зміни, що є важливим для командної роботи і підтримки якості коду.

В якості середовища розробки програмного забезпечення нами обраний Python. Python має простий і зрозумілий синтаксис, що робить його простим у вивченні та використанні. Це дозволяє розробникам писати та тестувати код швидко, що особливо важливо для досліджень і створення прототипів. Python сприяє зниженню кількості помилок і спрощує підтримку та модифікацію коду завдяки своїй структурованості та читабельності [20].

Такий вибір був обумовлений також великою кількістю бібліотек, які дозволяють працювати з зображеннями. Використання бібліотек Pillow, OpenCV, NumPy та CTkinter забезпечує ефективну обробку та аналіз зображень, створення зручного графічного інтерфейсу та швидке тестування розроблених методів. Це

робить Python ідеальним вибором для реалізації проєкту з приховування інформації у графічних файлах.

Бібліотека Pillow [16] є потужним ресурсом для роботи з графічними файлами в Python. Вона забезпечує широкий спектр функцій для обробки зображень, включаючи:

- Обрізку та масштабування: Pillow дозволяє легко змінювати розміри зображень, що важливо для підготовки даних перед приховуванням інформації.
- Фільтрацію та корекцію кольорів: Бібліотека підтримує різні методи фільтрації та корекції, що дозволяє покращити якість зображення перед стеганографічними маніпуляціями.
- Збереження та завантаження зображень у різних форматах: Pillow підтримує більшість популярних форматів зображень (JPEG, PNG, BMP тощо), що робить його сумісним і гнучким для роботи з різними типами медіафайлів.

Бібліотека OpenCV [15] (Open Source Computer Vision Library) в мові Python є однією з найбільш популярних бібліотек для обробки зображень. Її використання в проєкті надає такі переваги:

- Розширені можливості обробки зображень: OpenCV підтримує складні методи обробки, такі як дискретне косинусне перетворення (DCT), яке використовується для приховування інформації.
- Аналіз об’єктів та розпізнавання образів: Бібліотека дозволяє здійснювати розпізнавання об’єктів, що може бути корисним для перевірки та валідації стеганографічних методів.
- Висока продуктивність: OpenCV забезпечує високу швидкість обробки зображень завдяки оптимізованим алгоритмам.

Бібліотека NumPy [14] є основною бібліотекою для роботи з числовими даними в Python. Вона забезпечує ефективні засоби для обробки багатовимірних

масивів та матриць, що є важливим для стеганографічних алгоритмів. Основні можливості NumPy включають:

- Швидкі та ефективні операції над масивами: NumPy дозволяє виконувати масові обчислення значно швидше, ніж при використанні стандартних циклів Python.
- Підтримка складних математичних операцій: Бібліотека включає функції для лінійної алгебри, статистики, випадкових чисел та інших наукових обчислень.
- Інтеграція з іншими науковими бібліотеками: NumPy є основою для багатьох інших наукових бібліотек, таких як SciPy та Matplotlib, що забезпечує інтегроване середовище для наукових досліджень.

Бібліотека CTkinter [9] є основною бібліотекою Python для створення графічних інтерфейсів користувача (GUI). Її використання дозволяє створити зручний та інтуїтивно зрозумілий інтерфейс для взаємодії з програмою. Основні переваги Ctkinter включають:

- Простота використання: CTkinter дозволяє швидко створювати прості та функціональні інтерфейси без необхідності вивчення складних графічних фреймворків.
- Крос-платформеність: Програми написані із використанням CTkinter можуть працювати на різних операційних системах без значних змін у коді.
- Гнучкість та розширюваність: Бібліотека підтримує створення різноманітних елементів управління, таких як кнопки, поля вводу, меню тощо, що дозволяє створювати інтерфейси будь-якої складності.

1.4. Особливості програмної реалізації додатку StegoProg

На початковому етапі розробки програмного застосунку, потрібно підключити усі необхідні нам бібліотеки (рис. 2.5).

```
from PIL import Image
import cv2
import numpy as np
import matplotlib.pyplot as plt
from math import log10, sqrt
import bisect
import xlwt
from customtkinter import *
```

Рис. 2.5 - Підключення бібліотек

Наступним етапом буде проєктування та створення графічного інтерфейсу користувача, який дозволить користувачеві легко вибирати файли, вводити текст та взаємодіяти з програмою. Для цього використаємо бібліотеку CTkinter.

Код головного вікна програми StegoProg наведено на рисунку 2.6.

```
main_window = CTk()
main_window.title("StegoProg") # Встановлення заголовку вікна
main_window.geometry("740x740") # Встановлення розмірів вікна
set_appearance_mode("dark") # Встановлення темного режиму вигляду


tabView = CTkTabview(master=main_window,
                     width=700,
                     height=640)
tabView.pack(padx=10, pady=10)  # Розміщення вкладок у вікні
# Додавання вкладок
hide_tab = tabView.add("Hide Text")
reveal_tab = tabView.add("Reveal Text")
preview_tab = tabView.add("Image Preview")
comparison_tab = tabView.add("Comparison")
manual_tab = tabView.add("Manual")
```

Рис. 2.6 - Створення головного вікна додатку

Після створення інтерфейсу головного вікна програми, додаємо відповідні поля, меню та кнопки для кожної із вкладок. На рисунку 2.7 наведено лістинг коду інтерфейсу вкладки “Hide Text”.

```
# Створення мітки для вибору файлу
hide_text_window_file_label = CTkLabel(hide_tab, text="Select File:")
hide_text_window_file_label.pack(pady=5)

# Створення текстового поля для введення шляху до файлу
hide_text_window_file_entry = CTkEntry(hide_tab, height=20, width=350)
hide_text_window_file_entry.pack(pady=5)

# Створення кнопки для відкриття діалогового вікна вибору файлу
hide_text_window_browse_button = CTkButton(hide_tab, text="Browse", cursor="hand2")
hide_text_window_browse_button.pack(pady=5)

# Створення текстової області для введення тексту, який буде приховано
hide_text_window_text_area = CTkTextbox(hide_tab, height=200, width=400)
hide_text_window_text_area.pack(pady=5)

# Створення мітки для відображення результатів операції приховування тексту
hide_text_window_result_label = CTkLabel(hide_tab, text="")
hide_text_window_result_label.pack(pady=5)

# Створення мітки для відображення властивостей вибраного файлу
hide_text_window_props_label = CTkLabel(hide_tab, text="")
hide_text_window_props_label.pack(pady=5)

# Створення меню для вибору методу приховування тексту
hide_text_window_option_method = CTkOptionMenu(hide_tab, values=["Least Significant Bit", "Discrete Cosine Transform, Pixel Value Differencing"])
hide_text_window_option_method.pack(pady=20)

# Створення кнопки для запуску процесу приховування тексту у зображенні
hide_text_window_hide_button = CTkButton(hide_tab, text="Hide", cursor="hand2")
hide_text_window_hide_button.pack(pady=5)
```

Рис. 2.7 - Реалізація інтерфейсу вкладки “Hide Text”

На рисунку 2.8 наведено лістинг коду інтерфейсу вкладки “Reveal Text”.

```
# Мітка для вибору файлу
reveal_text_window_file_label = CTkLabel(reveal_tab, text="Select File:")
reveal_text_window_file_label.pack(pady=5)

# Текстове поле для введення шляху до файлу
reveal_text_window_file_entry = CTkEntry(reveal_tab, height=20, width=350)
reveal_text_window_file_entry.pack(pady=5)

# Кнопка для відкриття діалогового вікна вибору файлу
reveal_text_window_browse_button = CTkButton(reveal_tab, text="Browse", cursor="hand2")
reveal_text_window_browse_button.pack(pady=5)

# Текстова область для відображення розкритого тексту (спочатку вимкнена)
reveal_text_window_result_text = CTkTextbox(reveal_tab, height=200, width=400)
reveal_text_window_result_text.pack(pady=5)
reveal_text_window_result_text.configure(state='disabled')

# Мітка для відображення результатів операції розкриття тексту
reveal_text_window_result_label = CTkLabel(reveal_tab, text="")
reveal_text_window_result_label.pack(pady=5)

# Мітка для відображення властивостей вибраного файлу
reveal_text_window_props_label = CTkLabel(reveal_tab, text="")
reveal_text_window_props_label.pack(pady=5)

# Меню для вибору методу розкриття тексту
reveal_text_window_option_method = CTkOptionMenu(reveal_tab, values=["Least Significant Bit", "Discrete Cosine Transform, Pixel Value Differencing"])
reveal_text_window_option_method.pack(pady=20)

# Кнопка для запуску процесу розкриття тексту
reveal_text_window_reveal_button = CTkButton(reveal_tab, text="Reveal Text", cursor="hand2")
reveal_text_window_reveal_button.pack(pady=5)
```

Рис. 2.8 - Реалізація інтерфейсу вкладки “Reveal Text”

На рисунку 2.9 наведено лістинг коду інтерфейсу вкладки “Image Preview”

```
# Створення мітки для перегляду оригінального зображення
img_label_before = CTkLabel(preview_tab, text="Before")
img_label_before.pack(pady=10)

# Створення мітки для перегляду зображення після вбудовування тексту
img_label_after = CTkLabel(preview_tab, text="After")
img_label_after.pack(pady=10)
```

Рис. 2.9 - Реалізація інтерфейсу вкладки “Image Preview”

На рисунку 2.10 наведено лістинг коду інтерфейсу вкладки “Comparison”

```python
# Мітка для вибору файлу
comparison_tab_file_label_before = CTkLabel(comparison_tab, text="Select File Before Hiding Text:")
comparison_tab_file_label_before.pack(pady=5)

# Текстове поле для введення шляху до файлу
comparison_tab_file_entry_before = CTkEntry(comparison_tab, height=20, width=350)
comparison_tab_file_entry_before.pack(pady=5)

# Кнопка для відкриття діалогового вікна вибору файлу
comparison_tab_browse_button_before = CTkButton(comparison_tab, text="Browse", cursor="hand2")
comparison_tab_browse_button_before.pack(pady=5)

# Мітка для вибору файлу
comparison_tab_file_label_after = CTkLabel(comparison_tab, text="Select File After Hiding Text:")
comparison_tab_file_label_after.pack(pady=5)

# Текстове поле для введення шляху до файлу із вбудованим повідомленням
comparison_tab_file_entry_after = CTkEntry(comparison_tab, height=20, width=350)
comparison_tab_file_entry_after.pack(pady=5)

# Кнопка для відкриття діалогового вікна вибору файлу із вбудованим повідомленням
comparison_tab_browse_button_after = CTkButton(comparison_tab, text="Browse", cursor="hand2")
comparison_tab_browse_button_after.pack(pady=(0, 20))

# Кнопка для порівняння червоного каналу
comparison_tab_button_red = CTkButton(comparison_tab, text="Compare Red", cursor="hand2")
comparison_tab_button_red.pack(pady=5)

# Кнопка для порівняння зеленого каналу
comparison_tab_button_green = CTkButton(comparison_tab, text="Compare Green", cursor="hand2")
comparison_tab_button_green.pack(pady=5)

# Кнопка для порівняння синього каналу
comparison_tab_button_blue = CTkButton(comparison_tab, text="Compare Blue", cursor="hand2")
comparison_tab_button_blue.pack(pady=5)
```

Рис. 2.10 - Реалізація інтерфейсу вкладки “Comparison”

Таким чином, створивши потрібний нам користувацький інтерфейс, перейдемо до розробки програмного коду головних функцій стеганографії.

Лістинги кодів функцій приховування конфіденційних повідомлень, на основі методів LSB, DCT, PVD, наведені в додатках А, Б, В (див. Додатки).

1.5. Демонстрація роботи додатку StegoProg

Для запуску програмного застосунку StegoProg потрібно перейти до його директорії та запустити виконуючий файл додатку StegoProg.exe. Процес запуску показано на рисунку 2.11.

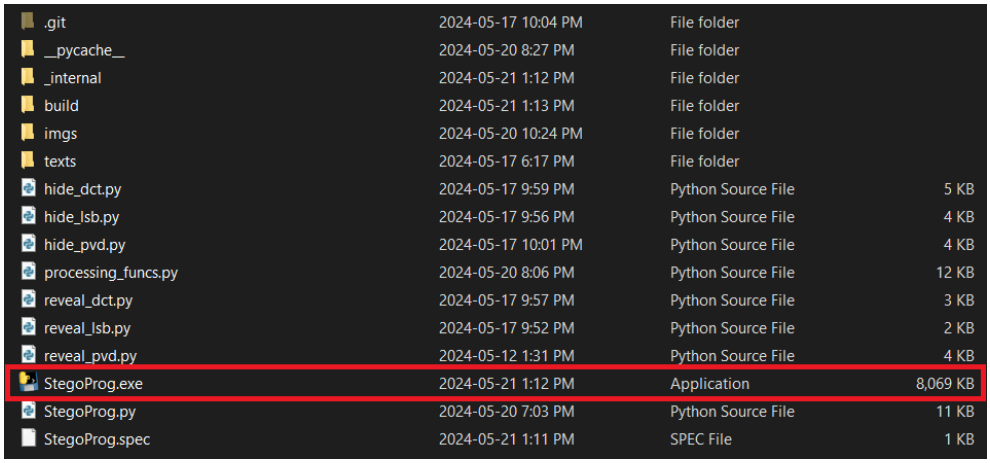

Рис. 2.11 - Демонстрація запуску застосунку StegoProg.exe

Після виконання запуску, відкриється вікно програми. Вкладка “Hide Text” обрана за замовчуванням (рис. 2.12). На даній вкладці користувач повинен вказати розміщення оригінального зображення, ввести повідомлення (текст) та обрати метод, за яким буде приховано цей текст у зображення.

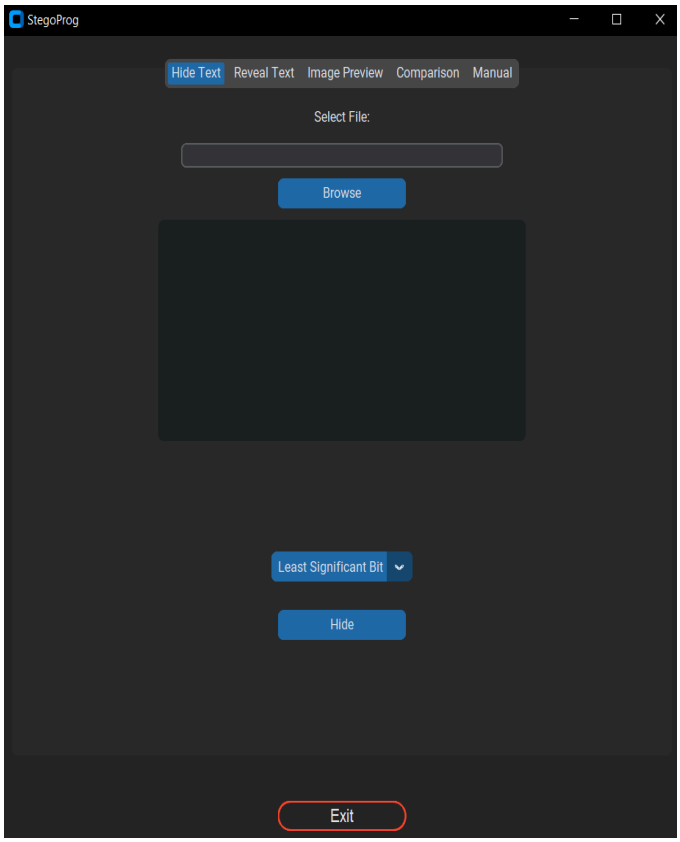

Рис. 2.12 - Вкладка “Hide Text” застосунку StegoProg.exe

За натисканням кнопки “Hide”, текст буде приховано у графічний файл та збережено у папці “encoded”, що знаходиться у головній директорії застосунку: stegoProg\imgs\encoded. Крім цього, на цій вкладці буде виведено метрики оцінки якості стегозображення, зокрема: час виконання приховування (Embedding time), PSNR, MSE (див. Розділ 1.4). Процес приховування тексту “Hello World!”, методом DCT наведено на рисунку 2.13.

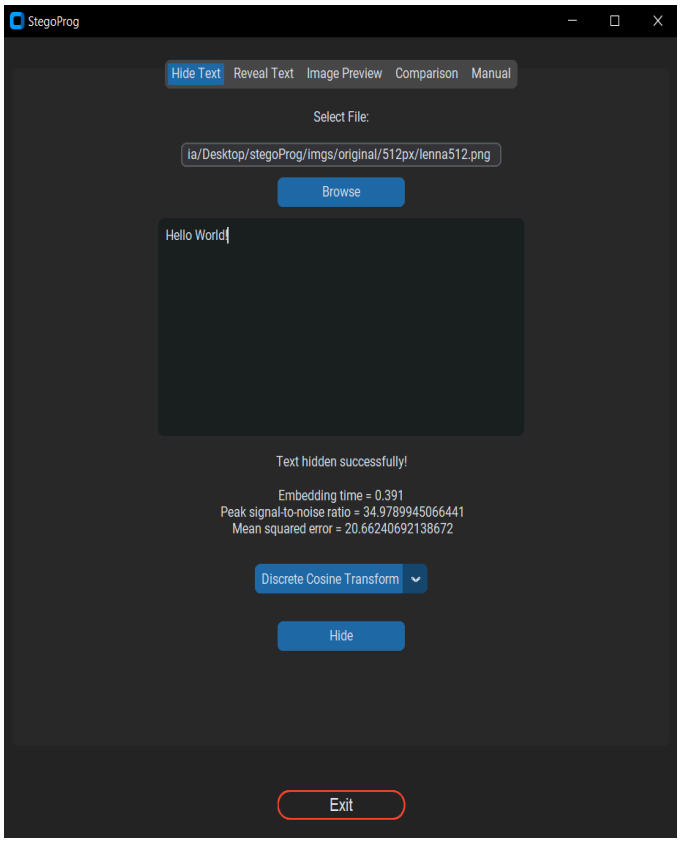

Рис. 2.13 - Процес приховування

Приховавши текст у графічний файл, користувач може візуально переглянути та порівняти зображення із порожнім контейнером та заповненим стегоконтейнером у вкладці “Image Preview” (рис. 2.14).

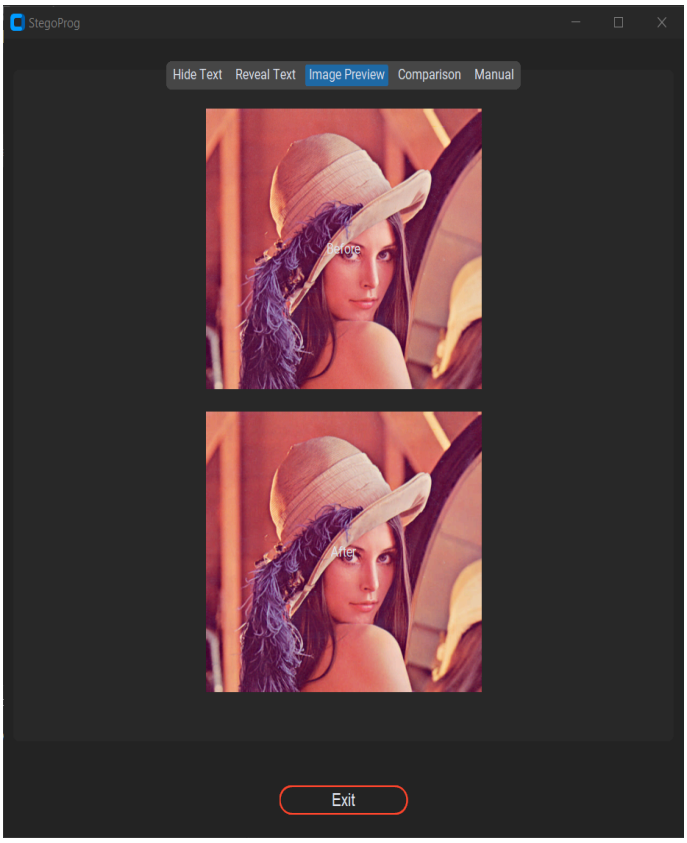

Рис. 2.14 - Зображення із порожнім контейнером та заповненим стегоконтейнером за вкладкою “Image Preview”

Для порівняння показників інтенсивності кольорів та частоту відповідного рівня яскравості на зображеннях, користувач може перейти до вкладки “Comparison”. На цій вкладці необхідно вказати розміщення оригінального зображення та стегозображення, після чого можна переглянути графіки, натиснувши кнопки “Compare Red”, “Compare Green”, “Compare Blue” (рис. 2.15).

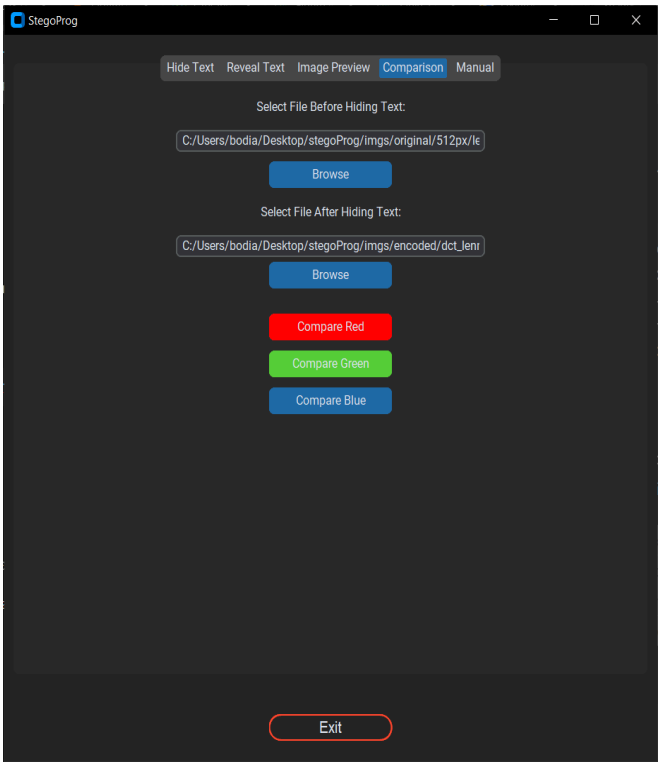

Рис. 2.15 - Порівняння рівня яскравості кольорів зображень за вкладкою “Comparison”

На рисунках 2.16, 2.17 і 2.18 наведено графіки трьох каналів, що відображають кількість пікселів у зображенні, що мають відповідний рівень інтенсивності до та після приховування інформації.

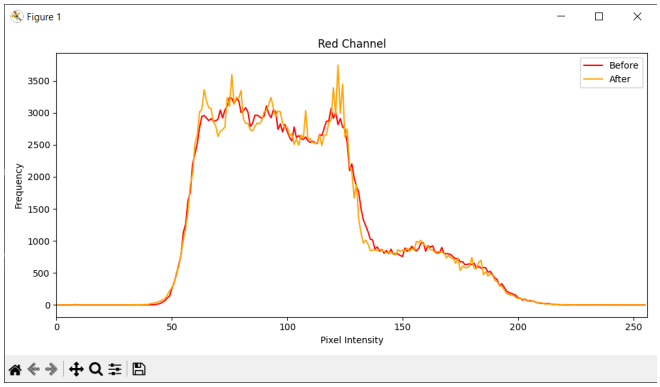

Рис. 2.16 - Графік інтенсивності червоного кольору

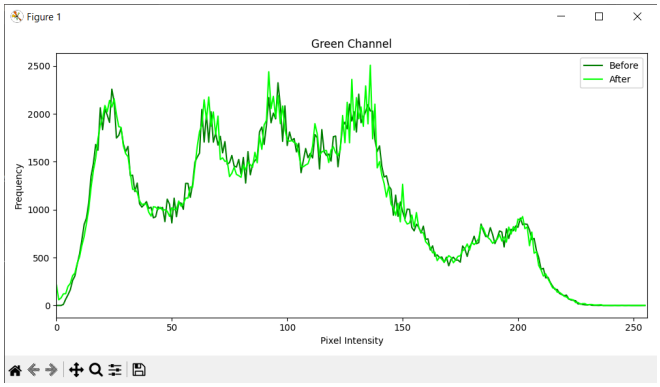

Рис. 2.17 - Графік інтенсивності зеленого кольору

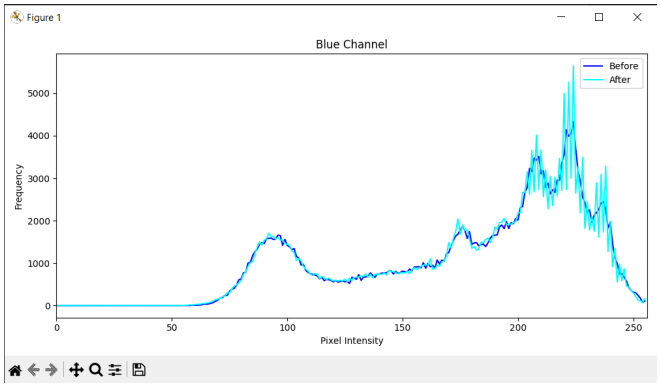

Рис. 2.18 - Графік інтенсивності синього кольору

Для вилучення повідомлення із стегоконтейнера потрібно перейти до вкладки “Reveal Text”. Далі необхідно вказати шлях до зображення із прихованим текстом, обрати метод, за яким приховано інформацію, та натиснути кнопку “Reveal”. У відповідному полі буде виведено відновлене повідомлення та час декодування прихованого повідомлення .

Результат роботи, вилучення інформації із стегозображення наведено на рисунку 2.19.

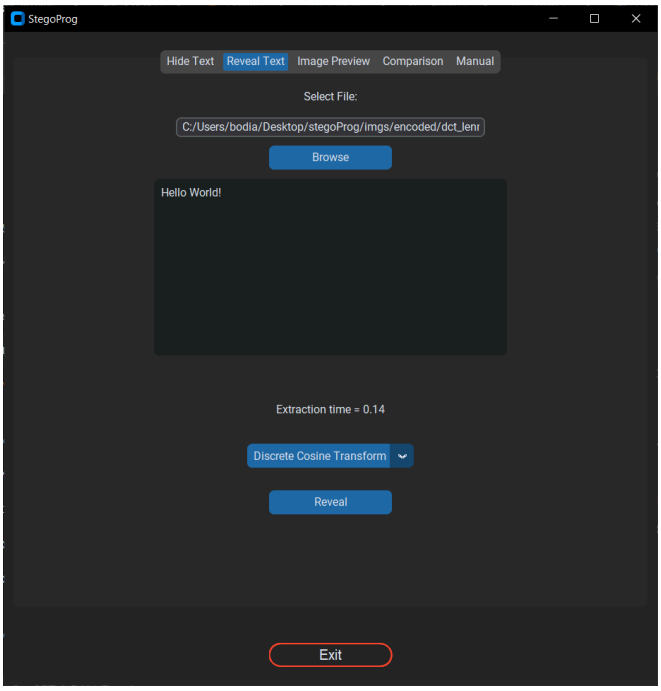

Рис. 2.19 - Декодування повідомлення

Для кращого ознайомлення з функціоналом додатку StegoProg, користувач має можливість звернутися до інструкції, яка розташована у вкладці ‘Manual’ (рис. 2.20).

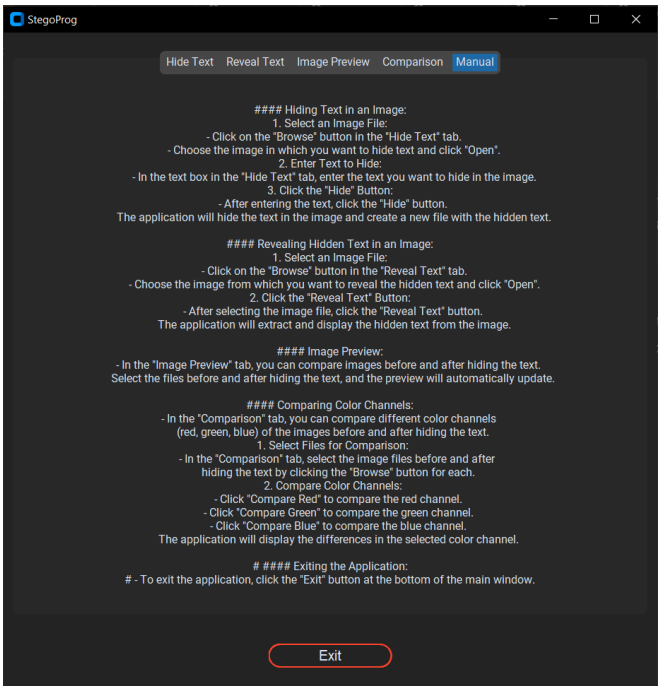

Рис. 2.20 - Інструкція по використанню програмного засобу

1.6. Організація тестування та аналіз отриманих результатів

Тестування програм та систем є важливим етапом у розробці програмного забезпечення. Тестування забезпечує впевненість у тому, що програма працює коректно, відповідає вимогам та очікуванням користувачів. Також, при цьому, виявляються та усуваються потенційні помилки до впровадження системи у виробниче середовище. Тестування допомагає перевірити функціональність, продуктивність, безпеку та сумісність програмного забезпечення. Зокрема, під час розробки методів стеганографії, тестування дозволяє оцінити ефективність алгоритмів приховування інформації та їх вплив на якість зображень. Використання метрик, таких як MSE та $PSNR$, допомагає кількісно оцінити, наскільки зміни у зображенні є непомітними для людського ока, що є критично важливим для стеганографічних застосувань. Завдяки тестуванню можна виявити та виправити недоліки, підвищити надійність та якість кінцевого продукту, забезпечуючи тим самим його успішне функціонування в реальних умовах експлуатації.

У процесі тестування додатку StegoProg встановлено, що інтерфейс програми досить зручний та інтуїтивно зрозумілий для взаємодії користувача із системою, що дозволяє достатньо легко виконувати усі потрібні дії.

Для проведення тестування нами було підібрано набір зображень розміром 512x512 пікселів та 1024x1024 пікселів (рис. 2.21, 2.22).

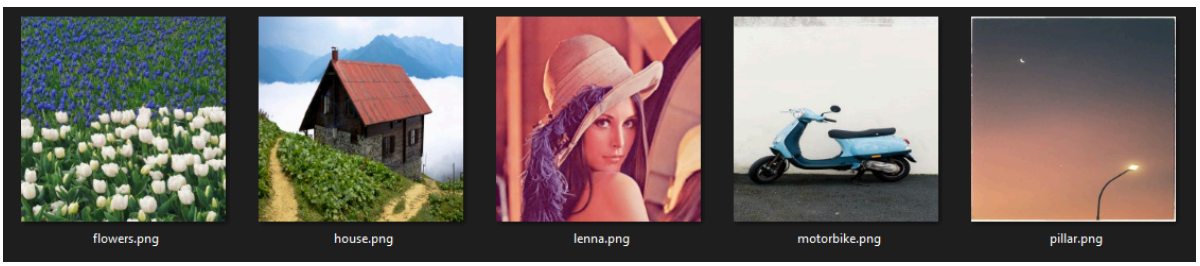

Рис. 2.21 - Набір зображень розміром 512x512 пікселів

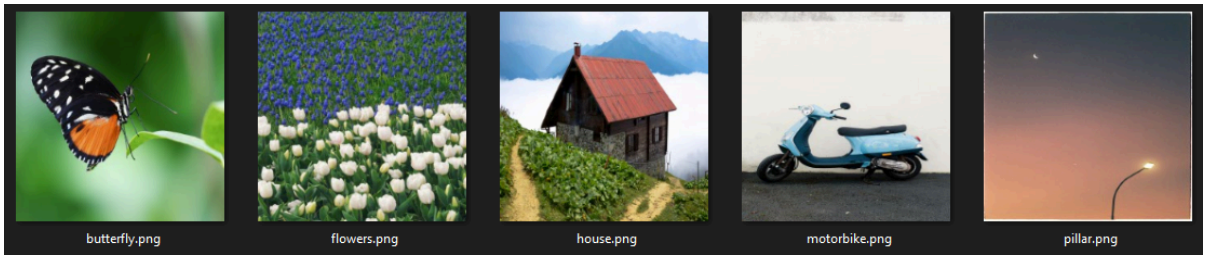

Рис. 2.22 - Набір зображень розміром 1024x1024 пікселів

У якості повідомлень нами використано згенерований текст сервісом Lorem Ipsum, розміром 100 байт та 10000 байт. Властивості згенерованих повідомлень зображені на рисунку 2.23.

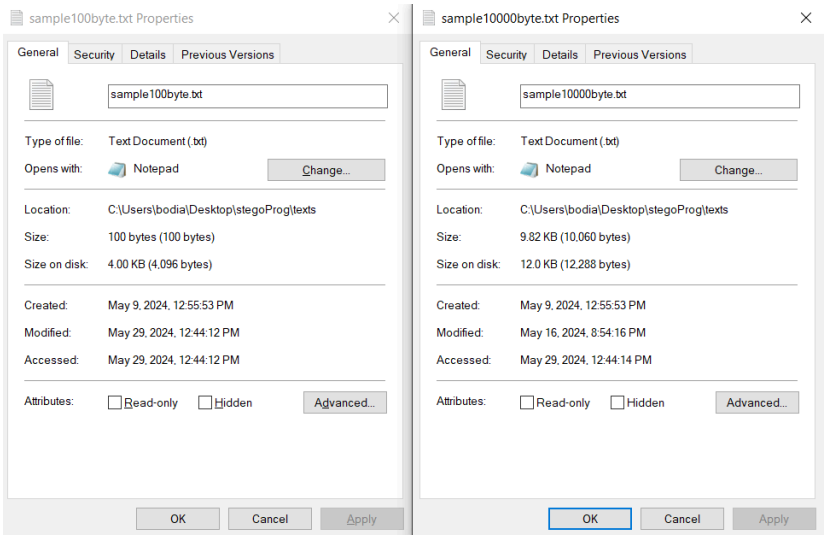

Рис. 2.23 - Властивості згенерованих повідомлень

Після етапу підготовки та завантаження наборів зображень і текстів, тестування було проведено наступним чином: у кожному зображенні різного розміру нами було приховано тексти різної довжини та змісту, використовуючи різні методи стеганографії. Загалом нами проведено чотири тестування. При кожному тестуванні нами була проведена наступна робота:

Тестування 1 – взято набір зображень розміром 512x512 пікселів і текст розміром 100 байт. Для кожного зображення виконано приховування секретного повідомлення зазначеним розміром, використовуючи реалізовані методи LSB, DCT, PVD.

Тестування 2 – взято набір зображень розміром 1024x1024 пікселів і текст розміром 100 байт. Для кожного зображення виконано приховування секретного повідомлення зазначеним розміром, використовуючи реалізовані методи LSB, DCT, PVD.

Тестування 3 – взято набір зображень розміром 512x512 пікселів і текст розміром 10000 байт. Для кожного зображення виконано приховування секретного повідомлення зазначеним розміром, використовуючи реалізовані методи LSB, DCT, PVD.

Тестування 4 – взято набір зображень розміром 1024x1024 пікселів і текст розміром 10000 байт. Для кожного зображення виконано приховування секретного повідомлення зазначеним розміром, використовуючи реалізовані методи LSB, DCT, PVD.

Для оцінки впливу приховування інформації на якість зображення використано дві основні метрики MSE та $PSNR$. Додатково нами було проведено оцінку якості між оригінальним та модифікованим зображеннями за такими параметрами: SSIM (Structural Similarity Index Measure) та NCC (Normalized Cross-Correlation). Програмна підтримка параметрів оцінки наведена у додатку Г.

Для зручності запису отриманих результатів тестування до таблиці Excel нами була розроблена функція наведена на рисунку 2.24, яка автоматизує цей процес. Ця функція використовує бібліотеку xlwt, що дозволяє легко працювати з файлами Excel у Python.

```python
def record_results(data, filename = "recorded_reslts.xls", sheet = "Sheet1"):
    book = xlwt.Workbook()
    sh = book.add_sheet(sheet)

    headers = [
        'Method', 'File Name', 'Text Size', 'Image Resolution',
        'MSE (Mean Squared Error)', 'PSNR (Peak Signal-to-Noise Ratio)',
        'Embedding Time', 'Extraction Time'
    ]

    for col, header in enumerate(headers):
        sh.write(0, col, header)

    for row, row_data in enumerate(data, start=1):
        for col, item in enumerate(row_data):
            sh.write(row, col, item)

    book.save(filename)
```

Рис. 2.24 - Функція запису отриманих результатів до таблиці Excel

Отримані результати тестування 1, наведені в таблиці 2.1, відображають ефективність трьох методів стеганографії: LSB, DCT та PVD при приховуванні текстової інформації розміром 100 байт у наборі зображень із роздільною здатністю 512x512 пікселів.

Таблиця 2.1 - Результати тестування 1

Method	File Name	Text Size	Image Resolution	MSE	PSNR	Embedding Time	Extraction Time	SSIM	NCC
LSB	flowers.png	100byte	512px	0.000565847	80.60381527	0.363	0.568	0.999941984	0.999999956
LSB	house.png	100byte	512px	0.000574748	80.53603103	0.348	0.616	0.999996011	0.999999921
LSB	lenna.png	100byte	512px	0.000536601	80.83429087	0.34	0.505	0.999996011	0.999999921
LSB	motorbike.png	100byte	512px	0.00055186	80.72253641	0.409	0.587	0.999998917	0.999999959
LSB	pillar.png	100byte	512px	0.000550588	80.72253641	0.512	0.649	0.999999995	0.999999859
DCT	flowers.png	100byte	512px	39.36485545	32.179717	0.347	0.173	0.911309368	0.992959424
DCT	house.png	100byte	512px	28.77057648	33.54131797	0.323	0.198	0.69014166	0.996491594
DCT	lenna.png	100byte	512px	20.4274203	35.02866836	0.304	0.203	0.665944524	0.996365964
DCT	motorbike.png	100byte	512px	17.24373118	35.76449117	0.33	0.295	0.397755611	0.998088456
DCT	pillar.png	100byte	512px	12.24219894	37.25220928	0.284	0.68	0.321593651	0.996330874
PVD	flowers.png	100byte	512px	0.003037771	73.30525388	0.06	0.068	0.999924442	0.99999996
PVD	house.png	100byte	512px	0.000507355	81.07768642	0.077	0.101	0.999995733	0.999999893
PVD	lenna.png	100byte	512px	0.000740051	79.43818553	0.032	0.073	0.999998503	0.999999917
PVD	motorbike.png	100byte	512px	0.001140594	77.55949094	0.021	0.061	0.999999995	0.999999879
PVD	pillar.png	100byte	512px	0.00048701	81.25542764	0.037	0.054	0.999989339	0.999999983

На рисунку 2.25 подано зображення із порожнім та заповненим контейнерами.

Рис. 2.25 - Зображення розміром 512x512 пікселів до та після приховування повідомлення розміром 100 байт методом DCT

Результати тестування 2, наведені у таблиці 2.2, при приховуванні текстової інформації розміром 100 байт у різних зображеннях роздільною здатністю 1024x1024 пікселів.

Таблиця 2.2 - Результати тестування 2

Method	File Name	Text Size	Image Resolution	MSE	PSNR	Embedding Time	Extraction Time	SSIM	NCC
LSB	butterfly.png	100byte	1024px	0.000146866	86.46159553	1.283	2.458	0.999991979	0.999999982
LSB	flowers.png	100byte	1024px	0.000137011	86.76324259	1.626	2.66	0.999999953	0.999999982
LSB	house.png	100byte	1024px	0.00014623	86.48043697	1.378	2.1	0.99998878	0.999999989
LSB	motorbike.png	100byte	1024px	0.000137011	86.76324259	1.314	1.94	0.999999873	0.999999989
LSB	pillar.png	100byte	1024px	0.000129382	87.0120712	1.24	2.55	0.999962271	0.999999968
DCT	butterfly.png	100byte	1024px	25.24402396	34.10921777	1.181	0.735	0.505038639	0.998444288
DCT	flowers.png	100byte	1024px	7.537981351	39.35825302	1.15	0.7	0.90044616	0.996441004
DCT	house.png	100byte	1024px	23.9254233	34.34220731	1.161	0.763	0.667432267	0.997358346
DCT	motorbike.png	100byte	1024px	18.630874	35.42847132	1.161	0.68	0.410741555	0.998106587
DCT	pillar.png	100byte	1024px	20.63357417	34.98505897	1.21	0.695	0.522605866	0.994722043
PVD	butterfly.png	100byte	1024px	0.000139236	86.69327418	0.049	0.044	0.999999926	0.999999952
PVD	flowers.png	100byte	1024px	0.000130335	86.98017672	0.046	0.039	0.99998218	0.99999999
PVD	house.png	100byte	1024px	0.000372887	82.41503517	0.073	0.054	0.999999835	0.999999982
PVD	motorbike.png	100byte	1024px	0.0002683	83.84459082	0.062	0.055	0.999978088	0.999999969
PVD	pillar.png	100byte	1024px	0.000122388	87.25340799	0.049	0.096	0.999804706	0.999969695

На рисунку 2.26 наведено зображення розміром 1024x1024 пікселів до та після вбудовування тексту розміром 100 байт.

Рис. 2.26 - Зображення розміром 1024x1024 пікселів до та після приховування повідомлення розміром 100 байт методом LSB

Результати тестування 3, наведені в таблиці 2.3, при приховуванні текстової інформації розміром 10000 байт у різних зображеннях роздільною здатністю 512x512 пікселів.

Таблиця 2.3 - Результати тестування 3

Method	File Name	Text Size	Image Resolution	MSE	PSNR	Embedding Time	Extraction Time	SSIM	NCC
LSB	flowers.png	10000byte	512px	0.051183065	61.03954074	0.438	0.564	0.999937812	0.999993579
LSB	house.png	10000byte	512px	0.05132548	61.0274734	0.515	0.653	0.977641046	0.99999612
LSB	lenna.png	10000byte	512px	0.050991058	61.05586335	0.414	0.748	0.998105111	0.999992675
LSB	motorbike.png	10000byte	512px	0.050912221	61.06258315	0.522	0.539	0.992334326	0.999996197
LSB	pillar.png	10000byte	512px	0.050977071	61.05705481	0.438	0.548	0.997191366	0.999987251
DCT	flowers.png	10000byte	512px	39.97412872	32.11301354	0.431	0.207	0.853069194	0.98907782
DCT	house.png	10000byte	512px	29.50245285	33.43222236	0.382	0.295	0.564422945	0.99424037
DCT	lenna.png	10000byte	512px	21.35538101	34.83573037	0.36	0.22	0.531130748	0.991884579
DCT	motorbike.png	10000byte	512px	18.12255478	35.5486094	0.357	0.223	0.314177419	0.995782575
DCT	pillar.png	10000byte	512px	13.23407491	36.91386772	0.372	0.301	0.185147635	0.988600307
PVD	flowers.png	10000byte	512px	0.24180603	54.29613234	0.879	0.787	0.999804706	0.999969695
PVD	house.png	10000byte	512px	0.14181623	58.43222236	0.382	0.456	0.990902732	0.999995628
PVD	lenna.png	10000byte	512px	0.166381042	61.23523304	0.67	0.588	0.999495628	0.999495622
PVD	motorbike.png	10000byte	512px	0.061421712	60.24758442	1.681	0.9	0.990902732	0.999995628
PVD	pillar.png	10000byte	512px	0.122554779	56.51860339	0.357	0.312	0.990902732	0.999995628

Оригінальне зображення та стегозображення розміром 512x512 пікселів, з прихованим повідомленням розміром 10000 байт, зображені на рисунку 2.27.

Рис. 2.27 - Зображення розміром 512x512 пікселів до та після приховування повідомлення розміром 10000 байт методом PVD

Результати тестування 4, наведені у таблиці 2.4, при приховуванні текстової інформації розміром 10000 байт у різних зображеннях роздільною здатністю 1024x1024 пікселів.

Таблиця 2.4. Результати тестування 4

Method	File Name	Text Size	Image Resolution	MSE	PSNR	Embedding Time	Extraction Time	SSIM	NCC
LSB	butterfly.png	10000byte	1024px	0.012884458	67.0301421	1.514	2.251	0.996605896	0.999998402
LSB	flowers.png	10000byte	1024px	0.012705167	67.09099978	1.526	2.339	0.99997782	0.999998409
LSB	house.png	10000byte	1024px	0.012810071	67.05528814	1.474	2.209	0.992775122	0.999999036
LSB	motorbike.png	10000byte	1024px	0.012752851	67.0747308	1.441	1.999	0.999730996	0.999999053
LSB	pillar.png	10000byte	1024px	0.012678464	67.1001371	1.375	2.173	0.999624259	0.999996872
DCT	butterfly.png	10000byte	1024px	8.543396632	40.12234106	1.524	0.747	0.255644128	0.994340158
DCT	flowers.png	10000byte	1024px	34.18298492	38.81449792	1.351	0.737	0.785060176	0.992358984
DCT	house.png	10000byte	1024px	24.81886228	34.18298492	1.391	0.746	0.528318162	0.995052606
DCT	motorbike.png	10000byte	1024px	33.97496353	35.21948806	1.586	0.695	0.318771425	0.99568295
DCT	pillar.png	10000byte	1024px	21.65505377	34.77521094	1.312	0.773	0.350719888	0.986645683
PVD	butterfly.png	10000byte	1024px	0.012839953	63.92496451	1.972	1.778	0.996697009	0.999998403
PVD	flowers.png	10000byte	1024px	0.038779895	67.04516923	1.079	0.931	0.999953221	0.999995188
PVD	house.png	10000byte	1024px	0.032581212	62.24473737	1.011	1.01	0.999739292	0.999998541
PVD	motorbike.png	10000byte	1024px	0.019740105	65.1773091	1.495	1.335	0.999739292	0.999998541
PVD	pillar.png	10000byte	1024px	0.051053725	64.67511014	1.122	0.897	0.999953221	0.999995188

Оригінальне зображення та стегозображення розміром 512x512 пікселів з прихованим повідомленням обсягом 10000 байт зображено на рисунку 2.28.

Рис. 2.28 - Зображення розміром 1024x1024 пікселів до та після приховування повідомлення розміром 10000 байт методом DCT

Проведене тестування трьох методів стеганографії (LSB, DCT, PVD) дозволяє зробити кілька важливих зауважень щодо якості та ефективності приховування текстової інформації в зображеннях.

Метод LSB (Least Significant Bit) демонструє найкращі результати у збереженні якості зображень після вбудовування інформації. Значення MSE залишаються помітно низькими, значення PSNR – достатньо високими, що свідчить про мінімальні візуальні зміни в зображеннях. Параметри SSIM та NCC також свідчать про незначні зміни яскравості, контрасту та структури незаповненого та заповненого контейнерів. Це робить метод LSB найкращим вибором для сценаріїв, де важливо зберегти високу якість зображень. Процес вбудовування та вилучення інформації методом LSB займає більше часу порівняно з іншими методами, особливо для зображень з високою роздільною здатністю.

Метод DCT (Discrete Cosine Transform) показує значні спотворення зображень, що виражається у високих значеннях MSE і низьких значеннях PSNR. Нижчі значення SSIM порівняно з іншими методами, що свідчить про меншу візуальну схожість. Це вказує на помітну втрату якості зображення після вбудовування інформації. Однак, метод DCT може бути корисним у випадках, де

допустима деяка втрата якості зображення. Метод DCT характеризується середнім часом вбудовування та вилучення інформації. Він швидший за метод LSB, але повільніший за метод PVD.

Метод PVD (Pixel Value Differencing) демонструє задовільні результати щодо збереження якості зображень, особливо для зображень з високою роздільною здатністю. Значення PSNR є достатньо високими, що свідчить про те, що метод PVD добре підходить для збереження якості зображень при приховуванні інформації. Майже досконалі значення SSIM та NCC. Цей метод є найшвидшим серед трьох методів. Час вбудовування та вилучення інформації значно менший, що робить цей метод привабливим для сценаріїв, де важлива швидкість обробки.

Кожен з розглянутих методів стеганографії має свої переваги та недоліки, які роблять їх більш або менш ефективними для конкретних застосувань. Метод LSB забезпечує найвищу якість зображень, але потребує більше часу для виконання операцій. Метод DCT може бути використаний у випадках, де допустима втрата якості зображення. Метод PVD є найшвидшим і забезпечує прийнятну якість зображень, що робить його ефективним для часокритичних завдань. Використання методів стеганографії залежить від конкретних вимог до якості зображень та швидкості обробки.

1.7. Рекомендації до впровадження програмного додатку StegoProg

Розроблений додаток StegoProg є інструментом стеганографії, який призначений для приховування текстової інформації у графічних файлах. Важливою характеристикою додатку є те, що, окрім базової функціональності стеганографічного вбудовування і вилучення інформації у графічних файлах, додаток надає користувачеві можливість оцінювати якість стеганографічного вбудовування даних за допомогою метрик MSE та $PSNR$. Він також забезпечує

вимірювання часу виконання функцій приховування тексту у вихідне зображення та час вилучення.

Завдяки інтегрованій функціональності аналізу показників інтенсивності кольорів та частоту зустрічі певного рівня яскравості на зображеннях, користувач може враховувати зміну рівня яскравості червоного, зеленого та синього кольорів зображення в ході стеганографічних операцій. Це дозволяє ліпше зрозуміти вплив на зображення вбудовування тексту і допомагає оцінити ці зміни рівня інтенсивності кольорів через графічну підтримку. Такий підхід дозволяє користувачеві аналізувати та порівнювати різноманітні аспекти стеганографічних операцій для покращення їх ефективності та якості результату.

Для оптимального використання програмного додатку StegoProg, рекомендується встановлення його на локальний комп’ютер. Для підвищення якості користувацького досвіду, рекомендується ознайомитися з інструкціями щодо використання програмного продукту які знаходяться у вкладці “Manual” (див. Рис. 2.6). Це дозволить користувачам краще зрозуміти функціональні можливості та оптимально використовувати їх для досягнення своїх цілей у контексті стеганографії.

ВИСНОВКИ

Метою кваліфікаційної роботи є програмна реалізація деяких методів стеганографії для приховування конфіденційної інформації у графічних файлах, проведення аналізу, оцінка якості та ефективності реалізованих алгоритмів.

У роботі на основі аналізу методів стеганографії розглянуті алгоритми для проєктування та програмної реалізації у форматі окремого застосунку. Проєктування та програмна реалізація методів стеганографії здійснено з використанням графічних файлів. Проведено тестування та аналіз ефективності реалізованих в програмному застосунку стеганографічних методів.

Для виконання поставленої мети нами було виконано наступні завдання:

1. Розглянуто теоретичні основи, класифікація та порівняльний аналіз існуючих методів стеганографії для приховування інформації, зокрема, алгоритмів LSB, DCT, PVD;
2. Здійснено опис та алгоритмізацію кожного з обраних стеганографічних методів приховування інформації в графічних файлах;
3. Реалізовано стеганографічні алгоритми LSB, DCT, PVD у вигляді програмних рішень засобами мови програмування Python;
4. Проведено експериментальне тестування та аналіз якості впроваджених рішень.

Розроблений нами програмний застосунок StegoProg забезпечує наступні визначальні для стеганографії параметри:

- Ефективність: застосунок забезпечує ефективне приховування даних, для великого обсягу даних, зберігаючи при цьому високу якість стегозображення.
- Непомітність: зміни, внесені програмним застосунком є непомітними. Візуальні зміни в якості зображення є мінімізованими.
- Сумісність: застосунок підтримує різні форматами зображень, зокрема, JPEG, PNG, BMP та інші популярні формати.

StegoProg надає користувачеві можливість за допомогою метрик MSE та PSNR оцінювати якість стеганографічного вбудовування даних у графічні файли. Додаток також забезпечує вимірювання часу виконання функцій приховування тексту у вихідне зображення та час вилучення.

Завдяки інтегрованій функціональності аналізу показників інтенсивності кольорів та частоту появи певного рівня яскравості зображення, користувач додатку StegoProg може враховувати зміну рівня яскравості червоного, зеленого та синього кольорів зображення в ході стеганографічних операцій. Для цього в додатку StegoProg передбачена відповідна графічна підтримка. Такий підхід дозволяє користувачеві аналізувати та порівнювати різноманітні аспекти стеганографічних операцій для покращення їх ефективності та якості результату.

Таким чином, програмний застосунок StgoProg забезпечує ефективне приховування конфіденційних даних в зображеннях з використанням методів LSB, DCT, PVD і може мати практичне застосування у забезпеченні відповідної комунікаційної конфіденційності.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Горпенюк А., Стороженко А. Дослідження та порівняльний аналіз стеганографічних методів для впровадження даних у цифрові файли. *Вісник Національного технічного університету України "ЛЬВІВСЬКА ПОЛІТЕХНІКА"*. 2012. № 741. С. 2–4. URL: https://science.lpnu.ua/sites/default/files/journal-paper/2017/jun/3733/horpeniukayastorozhenkoao.pdf.
2. Денисюк В. Стеганографічний алгоритм захисту даних з використанням файлів зображень. *Ефективна економіка*. 2017. № 5. URL: http://www.economy.nayka.com.ua/?op=1&z=5584.
3. Конахович Г., Прогонов Д., Пузиренко О. Комп’ютерна стеганографічна обробка й аналіз мультимедійних даних : підручник. Київ : Центр учб. літ., 2018. 558 с. URL: https://pdf.lib.vntu.edu.ua/books/2019/Konahovich_2018_558.pdf.
4. Кузнецов О., Євсеєв С., Король Х. Стеганографія : навч. посіб. Харків : Харків. нац. екон. ун-т, 2011. 232 с. URL: http://repository.hneu.edu.ua/bitstream/123456789/2289/1/Стеганография.pdf .
5. Методика оцінки стеганографічних методів приховування інформації в зображеннях / О. Туровський та ін. *Інфокомунікаційні та комп’ютерні технології*. 2021. № 2. С. 306–314. URL: https://www.researchgate.net/publication/366119479_METODIKA_OCINKI_STEGANOGRAFICNIH_METODIV_PRIHOVUVANNA_INFORMACII_V_ZOBRAZENNAH.
6. Планування досліджень методів стеганографії та стегоаналізу / В. Корольов та ін. *Вісник Хмельницького національного університету*. 2011. № 4. URL: http://journals.khnu.km.ua/vestnik/pdf/tech/2011_4/53kor.pdf.

7. Пузиренко О. Протокол оцінки ефективності алгоритмів комп’ютерної стеганографії. *Ukrainian information security research journal*. 2006. № 2. 29. URL: https://www.academia.edu/88420550/Протокол_оцінки_ефективності_алгоритмів_компютерної_стеганографії.
8. Швідченко І. Методи виявлення стеганографічного приховання інформації в зображеннях. *Вісник Національного технічного університету України "ЛЬВІВСЬКА ПОЛІТЕХНІКА"*. 2012. № 741. 36. URL: https://science.lpnu.ua/sites/default/files/journal-paper/2017/jun/3737/shvidchenkoiv.pdf.
9. Graphical user interfaces with ctk. *Python documentation*. URL: https://docs.python.org/3/library/tkinter.html (date of access: 08.05.2024).
10. Hiding data in images using steganography techniques with compression algorithms / O. F. A. Wahab et al. *TELKOMNIKA (telecommunication computing electronics and control)*. 2019. Vol. 17, no. 3. P. 1168. URL: https://doi.org/10.12928/telkomnika.v17i3.12230 (date of access: 07.06.2024).
11. Image quality assessment: from error visibility to structural similarity / Z. Wang et al. *IEEE transactions on image processing*. 2004. Vol. 13, no. 4. P. 600–612. URL: https://doi.org/10.1109/tip.2003.819861 (date of access: 07.06.2024).
12. Juneja M., Sandhu P. S. An improved LSB based steganography technique for RGB color images. *International journal of computer and communication engineering*. 2013. P. 513–517. URL: https://doi.org/10.7763/ijcce.2013.v2.238 (date of access: 07.06.2024).
13. Mara D. S. S., R K., L A. Computer forensic in image steganography. *International journal for research in applied science and engineering technology*. 2022. Vol. 10, no. 5. P. 3831–3842. URL: https://doi.org/10.22214/ijraset.2022.43237 (date of access: 07.06.2024).
14. Numpy. *PyPI*. URL: https://pypi.org/project/numpy/ (date of access: 15.05.2024).

15. Opencv-python. *PyPI*. URL: https://pypi.org/project/opencv-python/ (date of access: 12.05.2024).
16. Pillow. *PyPI*. URL: https://pypi.org/project/pillow/ (date of access: 05.05.2024).
17. Prasad S., Pal A. K. An RGB colour image steganography scheme using overlapping block-based pixel-value differencing. *Royal society open science*. 2017. Vol. 4, no. 4. P. 161066. URL: https://doi.org/10.1098/rsos.161066 (date of access: 07.06.2024).
18. Testing of image steganography with use of LSB and DCT techniques. *Volume-8 issue-10, august 2019, regular issue*. 2019. Vol. 8, no. 10. P. 3694–3697. URL: https://doi.org/10.35940/ijitee.j9659.0881019 (date of access: 07.06.2024).
19. Tseng H.-W., Leng H.-S. A steganographic method based on pixel-value differencing and the perfect square number. *Journal of applied mathematics*. 2013. Vol. 2013. P. 1–8. URL: https://doi.org/10.1155/2013/189706 (date of access: 07.06.2024).
20. Welcome to python.org. *Python.org*. URL: https://www.python.org/ (date of access: 02.05.2024).

ДОДАТКИ

Додаток А

Реалізація методу LSB(Least Significant Bit) для приховування текстової інформації у графічний файлах

```
def hideText_LSB(file_path_entry, text_to_hide_entry, result_label, props_label, img_label_after):
    # Отримання шляху до файлу
    file_path = file_path_entry.get()
    if os.path.isfile(file_path):
        try:
            start = timer()  # Початок вимірювання часу
            end_marker = "###END###"  # Маркер кінця тексту
            original_image_file = os.path.basename(file_path)
            message_to_hide = text_to_hide_entry.get(1.0, "end-1c")

            if message_to_hide.strip() != "":
                message_to_hide = text_to_hide_entry.get(1.0, "end-1c") + end_marker

                image = PIL.Image.open(file_path)
                width, height = image.size
                img_arr = np.array(list(image.getdata()))

                # Перевірка, чи не є зображення у палітровому режимі
                if image.mode == "P":
                    result_label.configure(text="Not supported!")
                    return False

                # Визначення кількості каналів (3 для RGB, 4 для RGBA)
                channels = 4 if image.mode == "RGBA" else 3
                pixels = img_arr.size // channels

                # Перетворення повідомлення у бінарний формат
                byte_message = ''.join(f"{ord(c):08b}" for c in message_to_hide)
                bits = len(byte_message)

                # Перевірка, чи повідомлення не занадто довге для приховування
                if bits > pixels:
                    result_label.configure(text="The message is too long to encode!")
                else:
                    index = 0
                    for i in range(pixels):
                        for j in range(0, 3):
                            if index < bits:
                                img_arr[i][j] = int(bin(img_arr[i][j])[2:-1] + byte_message[index], 2)
                                index += 1

                    # Перетворення масиву назад у зображення
                    img_arr = img_arr.reshape((height, width, channels))
                    result = PIL.Image.fromarray(img_arr.astype('uint8'), image.mode)
                    dct_encoded_image_file_path = "./imgs/encoded/lsb_" + original_image_file
```

```
                    # Збереження зображення з прихованим текстом
                    result.save(dct_encoded_image_file_path)
                    end = timer()  # Кінець вимірювання часу

                     # Відображення попереднього перегляду зображення після приховування тексту
                                display_image_preview_after(dct_encoded_image_file_path, img_label_after)

                    processing_time = round(end - start, 3)
                    # Обчислення PSNR та MSE для оцінки якості прихованого зображення
                         psnr_mse(file_path, dct_encoded_image_file_path, props_label, processing_time)

                    result_label.configure(text="Text hidden successfully!")
                    return True
            else:
                result_label.configure(text="Type a text you want to hide!")
                return False
        except Exception as e:
            print(f"Error: {str(e)}")
            result_label.configure(text="An error occurred during the process!")
            return False
    else:
        result_label.configure(text="No such file or directory!")
        return False
```

Додаток Б

Реалізація методу DCT(Discrete Cosine Transform) для приховування текстової інформації у графічний файлах

```
def hideText_DCT(file_path_entry, text_to_hide_entry, result_label, props_label, img_label_after):
    # Отримання шляху до файлу
    file_path = file_path_entry.get()
    if os.path.isfile(file_path):
        try:
            start = timer()  # Початок вимірювання часу
                original_image_file = os.path.basename(file_path)  # Отримання імені файлу
                secret_message = text_to_hide_entry.get(1.0, "end-1c")  # Отримання тексту для приховування

            if secret_message.strip() != "":
                    secret = f"{len(secret_message)}*{secret_message}"  # Додавання довжини до повідомлення
                bit_message = textToBinary(secret)  # Перетворення тексту у двійковий рядок

                # Зчитування кольорового зображення
                img = Image.open(file_path)
                img = img.convert('RGB')
                img_arr = np.array(img)
                height, width, _ = img_arr.shape

                # Перевірка, чи вміститься повідомлення у зображення
                if (width // 8) * (height // 8) < len(bit_message):
                    result_label.configure(text="The message is too long to encode!")
                    return False

                # Додавання padding, якщо необхідно
                  padded_height = height + (8 - height % 8) if height % 8 != 0 else height
                padded_width = width + (8 - width % 8) if width % 8 != 0 else width
                if padded_height != height or padded_width != width:
                    img = add_padding(img, padded_height, padded_width)
                    img_arr = np.array(img)

                height, width, _ = img_arr.shape
                N = 8  # Розмір блоку для DCT
                img_arr = np.float32(img_arr) - 128

                index = 0
                # DCT, квантування і вставка секретного повідомлення для кожного каналу (R, G, B)
                for channel in range(3):
                    for i in range(0, height, N):
                        for j in range(0, width, N):
                            block = img_arr[i:i + N, j:j + N, channel]
                            dct_block = dct(block)
                                        quantized_block = np.round(dct_block /
```

```
QUANTIZATION_MATRIX)
                            if index < len(bit_message):
                                 quantized_block[4, 4] += -(int(quantized_block[4, 4])
% 2) + int(bit_message[index]) # зміни коефіцієнтів піддіапазону середніх частот
                                 index += 1
                                img_arr[i:i + N, j:j + N, channel] = quantized_block *
QUANTIZATION_MATRIX
                            img_arr[i:i + N, j:j + N, channel] = idct(img_arr[i:i + N,
j:j + N, channel])

                img_arr += 128
                img_arr = np.clip(img_arr, 0, 255)
                encoded_img = np.uint8(img_arr)
                encoded_img = Image.fromarray(encoded_img, 'RGB')

                            dct_encoded_image_file_path = "./imgs/encoded/dct_" +
original_image_file
                encoded_img.save(dct_encoded_image_file_path)

                end = timer()  # Кінець вимірювання часу

                processing_time = round(end - start, 3)
                       psnr_mse(file_path, dct_encoded_image_file_path, props_label,
processing_time)
                result_label.configure(text="Text hidden successfully!")

                              display_image_preview_after(dct_encoded_image_file_path,
img_label_after)
            else:
                result_label.configure(text="Type a text you want to hide!")
                props_label.configure(text="")
                return False
        except Exception as e:
            print(f"Error: {str(e)}")
            props_label.configure(text="")
            return False
    else:
        result_label.configure(text="No such file or directory!")
        props_label.configure(text="")
        return False
```

Додаток В

Реалізація методу PVD(Pixel Value Differencing) для приховування текстової інформації у графічний медіафайл

```
def hideText_PVD(file_path_entry, text_to_hide_entry, result_label, props_label, img_label_after):
    file_path = file_path_entry.get()
    if os.path.isfile(file_path):
        try:
            start = timer()
            message_to_hide = text_to_hide_entry.get(1.0, “end-1c”)
            original_image_file = os.path.basename(file_path)
            if message_to_hide.strip() != ““:
                img = cv2.imread(file_path)
                height, width = img.shape[0], img.shape[1]
                width -= width % 2

                    data = bin(int.from_bytes(message_to_hide.encode()))[2:]        # Повідомлення конвертується у двійковий формат. Потім до нього додаються нулі на початку, щоб довжина була кратною 8.
                  data = ‘0’ * (8 - len(data) % 8) + data      #Додається довжина повідомлення (у 32-бітовому двійковому форматі) на початок, щоб можна було витягти його при розшифровці.


                data_len = bin(len(data))[2:].zfill(32)
                data = data_len + data

                i = capacity = 0
                while i < height:
                    for j in range(0, width, 2):
                        for k in range(3):
                            print(“img max = “, max(img[i, j + 1, k], img[i, j, k]))
                            print(“img min = “, min(img[i, j + 1, k], img[i, j, k]))
                            print(“img = “, img[i, j])
                            print(“img + 1 = “, img[i, j+1])
                            dif = max(img[i, j + 1, k], img[i, j, k]) - min(img[i, j + 1, k], img[i, j, k])
                            print(“dif = “, dif )
                            emb, n, maxr = embed_number(dif)
                            print(“emb, n, maxr  - “, emb, “ “, n,” “, maxr)
                              #перевіряється, чи можна змінити різницю між пікселями на максимальну допустиму.
                             res, _, _ = change_diff(maxr - dif, min(img[i, j + 1, k], img[i, j, k]), max(img[i, j + 1, k], img[i, j, k]))
                            print(“change_dif  = “, res)
                            if not res:
                               continue

                              #Зчитуються біти з повідомлення і додаються до різниці, після чого знову використовується change_diff для корекції значень пікселів.
                            bits = data[capacity:capacity + n]
                            capacity += len(bits)
                              #Зчитуємо біти секретного повідомлення і перетворюємо в
```

```
десяткове значення
                            new_dif = emb + int(bits, 2)
                            #Рахуємо нову різницю
                             _, img[i, j, k], img[i, j + 1, k] = change_diff(new_dif -
dif, img[i, j, k], img[i, j + 1, k])
                             print("new pixels values - ", img[i, j, k], " ", img[i, j
+ 1, k])
                            print("-------------------------------------")
                                if capacity == len(data): #Якщо всі біти повідомлення
вбудовані, цикл завершується.
                                end = timer()

                                             result_path = "./imgs/encoded/pvd_" +
original_image_file
                                cv2.imwrite(result_path, img)
                                            display_image_preview_after(result_path,
img_label_after)

                                # print(file_path, "        ", result_path)
                                processing_time = round(end - start, 3)
                                        psnr_mse(file_path, result_path, props_label,
processing_time)
                                # print(str(start) + "      "+ str(end)+"          "
+ str(end-start))
                                           result_label.configure(text="Text  hidden
successfully!")
                                return True

                    i += 1
                result_label.configure(text="The message is too long to encode!")
                return False
            else:
                result_label.configure(text="Type a text you want to hide!")
                props_label.configure(text="")
                return False
        except Exception as e:
            print(f"Error: {str(e)}")
            props_label.configure(text="")
            return False
    else:
        result_label.configure(text="No such file or directory!")
        props_label.configure(text="")
        return False
```

Додаток Г

Реалізація функцій оцінки якості зображень

```
def psnr_mse(image1_path, image2_path):
        original = cv2.imread(image1_path)
        encoded = cv2.imread(image2_path)
        mse = np.mean((original - encoded) ** 2)
        if(mse == 0):  # MSE is zero means no noise is present in the signal .
                    # Therefore PSNR have no importance.
            psnr = 100
        max_pixel = 255.0
```

```
        psnr = 20 * log10(max_pixel / sqrt(mse))
       return mse, psnr

def calculate_ssim(imageA, imageB):
    # Розбиваємо зображення на три канали: R, G, B
    ssim_vals = []
    for i in range(3):
        ssim_val, _ = ssim(imageA[:, :, i], imageB[:, :, i], data_range=1.0,
full=True)
        ssim_vals.append(ssim_val)
    # Середнє значення SSIM для всіх каналів
    return np.mean(ssim_vals)

def normalized_cross_correlation(imageA, imageB):
    # Перетворення зображень у формат з плаваючою комою
    imageA = img_as_float(imageA)
    imageB = img_as_float(imageB)

    # Обчислення середніх значень
    meanA = np.mean(imageA)
    meanB = np.mean(imageB)

    # Відхилення від середніх значень
    deviationA = imageA - meanA
    deviationB = imageB - meanB

   # Обчислення чисельника і знаменника NCC
    numerator = np.sum(deviationA * deviationB)
    denominator = np.sqrt(np.sum(deviationA ** 2) * np.sum(deviationB ** 2))

    # Обчислення NCC
    ncc = numerator / denominator

    return ncc
```

Додаток Д

Технічне завдання для застосунку StegoProg

ПІДСТАВИ ДЛЯ РОЗРОБКИ

На ринку стегоносистем існує значна кількість різноманітних рішень, що відрізняються за функціональною повнотою. Багато стегоносистем є безкоштовними або з відкритим кодом (наприклад, OpenStego, Steghide), але їхні функціональні можливості часто обмежені. Комерційні системи, такі як S-Tools або Prophecy, пропонують розширені функції та кращу підтримку, але їхня вартість може бути значною. Апаратні вимоги більшості стегоносистем є помірними, однак деякі просунуті рішення можуть вимагати потужніше обладнання. Безкоштовні рішення часто мають обмежену підтримку і не завжди отримують регулярні оновлення, тоді як комерційні системи забезпечують професійну підтримку та регулярні оновлення, що може включати додаткові витрати.

З огляду на ці проблеми, існує запит на нові розробки стеганографічних систем, які б були менш вимогливими до апаратного забезпечення, більш доступними та дешевими, але при цьому ефективними та надійними. Розробка таких рішень, як StegoProg, що забезпечують високу якість стеганографічного вбудовування, підтримку різних алгоритмів, зручність використання та графічну підтримку для аналізу результатів, може задовольнити ці потреби і підвищити доступність стеганографії для широкого кола користувачів.

ПРИЗНАЧЕННЯ РОЗРОБКИ

Програмний застосунок StgoProg забезпечує ефективне приховування конфіденційних даних в зображеннях з використанням методів LSB, DCT, PVD і може мати практичне застосування у забезпеченні відповідної комунікаційної конфіденційності.

StegoProg надає користувачеві можливість за допомогою метрик MSE та PSNR оцінювати якість стеганографічного вбудовування даних у графічні файли.

Додаток також забезпечує вимірювання часу виконання функцій приховування тексту у вихідне зображення та час вилучення.

Завдяки інтегрованій функціональності аналізу показників інтенсивності кольорів та частоту появи певного рівня яскравості зображення, користувач додатку StegoProg може враховувати зміну рівня яскравості червоного, зеленого та синього кольорів зображення в ході стеганографічних операцій. Такий підхід дозволяє користувачеві аналізувати та порівнювати різноманітні аспекти стеганографічних операцій для покращення їх ефективності та якості результату.

СТАДІЇ І ЕТАПИ РОЗРОБКИ

1. Дослідження та аналіз існуючих методів стеганографії, зокрема, алгоритмів LSB, DCT, PVD;
2. Опис алгоритмів для кожного з обраних методів приховування інформації;
3. Реалізація стеганографічних алгоритмів у вигляді програмних рішень засобами мови програмування Python;
4. Проведення тестування та аналіз якості і ефективності впроваджених рішень.

ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

1. Застосунок повинен забезпечувати ефективне приховування даних, забезпечуючи достатню (велику) ємність для прихованої інформації, зберігаючи при цьому високу якість зображення.
2. Зміни, внесені програмним застосунком повинні бути непомітними для людського сприйняття. Візуальні артефакти або зміни в якості зображення повинні бути мінімізовані.
3. Застосунок повинен бути сумісним із різними форматами зображень, такими як JPEG, PNG, BMP та іншими популярними форматами.
4. Застосунок повинен забезпечити певний рівень стійкості до випадкових небажаних модифікацій стегоконтейнера.

АНОТАЦІЯ

Буткевич Б. О. – **Програмна реалізація деяких моделей стеганографії для приховування конфіденційної інформації у графічних файлах** – Рукопис.

Кваліфікаційна робота за спеціальністю 122 Комп’ютерні науки. – Волинський національний університет імені Лесі Українки, Луцьк. – 2024р.

Ця робота присвячена розробці програмного додатку для підтримки технології стеганографічного захисту інформації. Розроблений додаток StegoProg забезпечує достатньо надійний захист інформації на основі методів LSB (Least Significant Bit), DCT (Discrete Cosine Transform), PVD (Pixel Value Differencing) для приховування конфіденційних даних в графічних файлах. Додаток розроблений засобами мови програмування Python.

Ключові слова: цифрова стеганографія, методи стеганографії, приховування конфіденційної інформації у графічних файлах.