Вінницький національний технічний університет

Факультет менеджменту та інформаційної безпеки

Кафедра менеджменту та безпеки інформаційних систем

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

Розробка програми виявлення прихованої інформації у зображеннях інтернет-ресурсів

Виконала: студентка 4 курсу,
спеціальності 125 — Кібербезпека
Освітня програма – Кібербезпека
інформаційних технологій та систем
(шифр і назва напряму підготовки, спеціальності)

Сіра А.Л.
(прізвище та ініціали)

Керівник: к.т.н., доц., зав. каф. МБІС
Карпінець В.В.
(прізвище та ініціали)
« 4 » червня 2025 р.

Рецензент: ст. викладач кафедри ОТ
Крупельницький Л. В.
(прізвище та ініціали)
« 4 » червня 2025 р.

Допущено до захисту
Голова секції УБ кафедри МБІС
д.т.н., проф. Яремчук Ю.Є.
“ 4 ” червня 2025р.

Вінниця ВНТУ – 2025

Вінницький національний технічний університет
Факультет менеджменту та інформаційної безпеки
Кафедра менеджменту та безпеки інформаційних систем

Рівень вищої освіти І-й (бакалаврський)
Галузь знань 12 – Інформаційні технології
Спеціальність 125 – Кібербезпека
Освітньо-професійна програма - Кібербезпека інформаційних технологій та систем

ЗАТВЕРДЖУЮ
Голова секції УБ, кафедра МБІС
д.т.н., проф. Яремчук Ю.Є.
" 20 " березня 2025 р.

З А В Д А Н Н Я
на бакалаврську кваліфікаційну роботу студенту
Сірої Анни Леонідівни
(прізвище, ім`я, по-батькові)

1. Тема роботи Розробка програми виявлення прихованої інформації у зображеннях інтернет-ресурсів
Керівник роботи Карпінець Віталій Володимирович, к.т.н., доцент, завідувач кафедри МБІС
(прізвище, ім`я, по-батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від " 20 " березня 2025року № 97
2. Термін подання студентом роботи 11.06.2025
3. Вихідні дані до роботи: технічна інформація про методи цифрової стеганографії та стеганоаналізу, приклади існуючих програм стеганоаналізу, засоби програмування Python, бібліотеки машинного навчання (sklearn, tensorflow/keras), датасети для тренування моделей (наприклад, BOSSBase)
4. Зміст текстової частини: вступ, аналіз сучасних методів приховування інформації у зображеннях, актуальність проблеми цифрової стеганографії в інтернет-середовищі, класифікація методів стеганографії та стеганоаналізу, аналіз існуючих рішень для виявлення прихованих даних, обґрунтування вибору методів машинного навчання та інструментів для побудови системи, розробка алгоритму виявлення прихованої інформації у зображеннях, розробка структурної та функціональної схем програми, програмна реалізація модулів автоматичного збору зображень з інтернету, модулів попередньої обробки, побудови ознак, класифікації та збереження результатів, реалізація користувацького інтерфейсу, тестування системи, інструкція користувача до роботи з програмою, висновки, перелік посилань, додатки.
5. Перелік ілюстративного матеріалу (з точним зазначенням обов`язкових креслень)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Розділ I	Карпінець В.В., к.т.н., доцент	20.02.2025	11.06.2025
Розділ II	Карпінець В.В., к.т.н., доцент	20.02.2025	11.06.2025
Розділ III	Карпінець В.В., к.т.н., доцент	20.02.2025	11.06.2025

7.Дата видачі завдання 20 березня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Аналіз теми, вивчення літературних джерел	20.03.2025	27.03.2025	Виконано
2	Огляд існуючих програм та систем	27.03.2025	03.04.2025	Виконано
3	Формування вимог, структурної схеми	04.04.2025	10.04.2025	Виконано
4	Розробка алгоритмів і логіки роботи системи	11.04.2025	20.04.2025	Виконано
5	Реалізація програмного забезпечення	21.04.2025	21.04.2025	Виконано
6	Розробка та інтеграція інтерфейсу користувача	11.05.2025	18.05.2025	Виконано
7	Тестування системи, оцінка результатів	19.05.2025	25.05.2025	Виконано
8	Оформлення дипломної роботи	26.05.2025	02.06.2025	Виконано
9	Захист бакалаврської роботи	11.06.2025	12.06.2025	Виконано

Студент _____ **Сіра А.Л.**
(підпис) (прізвище та ініціали)

Керівник роботи _____ **Карпінець В.В.**
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Дана бакалаврська кваліфікаційна робота присвячена розробці програми виявлення прихованої інформації у зображеннях інтернет-ресурсів. Метою дослідження є створення ефективного інструменту для виявлення цифрової стеганографії з використанням сучасних методів аналізу, включаючи машинне навчання та глибоке навчання. У ході роботи проведено аналіз існуючих підходів до стеганоаналізу, розроблено структурну схему системи, реалізован програму, яка здійснює автоматичний збір зображень із веб-ресурсів, їх обробку, аналіз та фіксацію результатів.

Для досягнення поставленої мети було використано Python, бібліотеки для обробки зображень і навчання моделей, а також реалізовано інтерфейс користувача. Розроблена система дозволяє виявляти потенційно небезпечний контент, підвищуючи рівень інформаційної безпеки в мережі.

Ключові слова: стеганографія, стеганоаналіз, прихована інформація, зображення, інтернет-ресурси, машинне навчання, безпека.

ABSTRACT

This bachelor's thesis is dedicated to the development of a software solution for detecting hidden information in images found on internet resources. The objective of the research is to create an effective tool for identifying digital steganography using modern analysis methods, including machine learning and deep learning techniques. The work includes an overview of existing steganalysis approaches, the development of a system architecture, and the implementation of a software module that performs automated image scraping, preprocessing, analysis, and result logging.

To achieve this goal, Python programming language was used along with libraries for image processing and model training. A user interface was also implemented. The developed system enables the detection of potentially malicious content, contributing to enhanced cybersecurity in the online environment.

Keywords: steganography, steganalysis, hidden information, images, internet resources, machine learning, security.

ЗМІСТ

ВСТУП

Актуальність. У сучасному цифровому середовищі, де значна частина комунікації та обміну даними відбувається через інтернет, все більшого поширення набувають методи прихованого передавання інформації. Одним із найбільш популярних засобів для цього є цифрова стеганографія — технологія, що дозволяє вбудовувати приховані повідомлення у графічні файли без видимих ознак змін. Це створює серйозні загрози для інформаційної безпеки, зокрема використання зображень для прихованого передавання шкідливого коду, інструкцій для ботмереж або витоку конфіденційної інформації.

У цьому контексті розробка програми, здатної виявляти приховану інформацію у зображеннях, що розміщуються на інтернет-ресурсах, є надзвичайно актуальною. Така система дозволяє здійснювати моніторинг вебконтенту з метою виявлення потенційно небезпечних зображень, що використовуються для прихованої комунікації або передачі заборонених даних. Впровадження подібного інструменту є важливим елементом у сфері цифрової криміналістики, кібербезпеки та моніторингу загроз.

Розробка такої системи має високу практичну цінність, оскільки дозволяє здійснювати аналіз великої кількості зображень в автоматичному режимі з використанням сучасних технологій обробки даних, машинного навчання та стеганоаналізу. Це, своєю чергою, сприяє підвищенню ефективності виявлення кіберзагроз та захисту інформаційних ресурсів.

Метою роботи є розробка програми для автоматизованого виявлення прихованої інформації у зображеннях, що розміщуються на інтернет-ресурсах, з використанням класичних і інтелектуальних методів стеганоаналізу.

Для досягнення мети необхідно виконати такі завдання:

– провести аналіз літературних джерел за тематикою цифрової стеганографії та методів її виявлення;

– дослідити сучасні алгоритми виявлення прихованої інформації у зображеннях, включаючи статистичні методи, моделі машинного навчання та глибокі нейронні мережі;

– розробити архітектуру програми, що включає блок збору зображень з інтернету, блок попередньої обробки, блок глибокого аналізу та модуль збереження результатів;

– реалізувати програмний інтерфейс користувача для взаємодії з системою;

– провести тестування системи для оцінки її точності, продуктивності та здатності виявляти приховану інформацію в реальних умовах.

Об’єктом дослідження є методи виявлення прихованої інформації у зображеннях, технології стеганоаналізу та інструменти машинного навчання.

Предметом дослідження є програма, що здійснює виявлення прихованої інформації у зображеннях, отриманих з інтернет-ресурсів, на основі класичних та інтелектуальних методів аналізу.

Новизна роботи полягає у розробці програми, що поєднує традиційні статистичні методи з сучасними підходами на базі глибокого навчання для виявлення прихованого контенту у зображеннях з інтернету.

Практична цінність роботи полягає у створенні ефективного інструменту для автоматичного виявлення потенційно загрозливих зображень, що може бути застосовано у службах кібербезпеки, правоохоронних органах, системах моніторингу соціальних мереж та інформаційного простору.

1 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ОБҐРУНТУВАННЯ ТЕМИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

У розділі подано огляд основних понять цифрової стеганографії та методів виявлення прихованої інформації у зображеннях. Розглянуто класифікацію технік приховування даних, методи стеганоаналізу, а також загрози, пов’язані з використанням стеганографії в мережевому середовищі. Особливу увагу приділено сучасним підходам до аналізу зображень, включно зі статистичними та машинними методами.

Також проаналізовано наявні програмні рішення для виявлення прихованих даних у графічних файлах. Визначено їхні обмеження, зокрема низьку точність або вузький функціонал. На основі аналізу обґрунтовано необхідність створення власного програми з розширеними можливостями. На основі аналізу сформовано висновки та постановлено завдання дипломної роботи.

1.1 Основні поняття цифрової стеганографії та стеганоаналізу

Цифрова стеганографія — це розділ інформаційної безпеки, що вивчає способи приховування інформації всередині цифрових носіїв таким чином, щоб факт передачі повідомлення залишався непоміченим для стороннього спостерігача. Термін «стеганографія» походить від грецьких слів steganos (прихований) та graphein (писати), що буквально означає «приховане письмо». Головною метою стеганографії є не лише забезпечення конфіденційності переданої інформації, а й маскування самого факту її існування.[1]

Суть методу полягає у модифікації незначної частини даних цифрового об’єкта-носія, наприклад, зображення, звукового чи відеофайлу, без суттєвої зміни його візуального або аудіального сприйняття. Завдяки цьому приховане повідомлення не викликає підозри і не виявляється при поверхневому огляді.

Попри те, що як криптографія, так і стеганографія є засобами захисту інформації, вони мають принципово різні підходи. Криптографія змінює вигляд повідомлення таким чином, щоб його неможливо було прочитати без спеціального ключа. Однак факт передавання зашифрованого повідомлення очевидний — навіть якщо його неможливо дешифрувати, відомо, що передається якась захищена інформація.

Стеганографія ж, навпаки, не викликає підозри, оскільки не помітна для стороннього спостерігача. Вона не змінює форму даних, а лише додає до них приховане повідомлення. У сучасних системах безпеки часто використовують комбінацію обох методів: спочатку інформацію шифрують, а потім — приховують у цифровому носії.[2]

У порівнянні з криптографією, яка гарантує високий рівень захисту змісту, але не приховує факт передавання, стеганографія має нижчий рівень криптостійкості (наприклад, LSB легко руйнується при JPEG-стисненні нижче 70%), проте дозволяє досягти високого рівня невидимості.

Цифрова стеганографія має широке коло застосувань як у легітимних, так і в потенційно небезпечних сферах:

− захист авторських прав. Приховування водяних знаків у зображеннях, відео та аудіо для підтвердження прав власника;

− конфіденційна комунікація. Передавання секретної інформації через загальнодоступні файли;

− цифрова ідентифікація. Приховане маркування файлів для ідентифікації користувача або пристрою;

− приховане зберігання. Збереження паролів або ключів шифрування у вигляді прихованих записів у файлах;

− шкідливе використання. Приховування команд для шкідливого ПЗ або забороненого контенту (наприклад, у випадках стеганографії, яку використовують ботнети).

Для приховування інформації можуть використовуватись різноманітні цифрові об’єкти:[2]

− зображення. Найпопулярніший носій завдяки великій кількості пікселів та можливості вбудування в найменш значущі біти;

− аудіофайли. Приховування даних за допомогою зміни амплітуди або частотного спектра;

− відео. Дає змогу комбінувати методи зображення та звуку, має велику ємність;

− текстові документи. Можуть використовуватись шляхом зміни форматування, пробілів, табуляцій;

− протоколи мережевого трафіку. Приховування інформації у заголовках пакетів або порядку їх надсилання (мережева стеганографія).

Цифрова стеганографія реалізується за допомогою різноманітних методів, які класифікуються за принципом вбудовування, типом носія, рівнем стійкості до атак, а також якістю маскування інформації. Основними підходами є просторові методи, частотні методи та адаптивні (контекстно-залежні) алгоритми.[3]

Просторові методи вбудовують інформацію безпосередньо в піксельні дані зображення. Найпоширенішим з них є метод найменш значущого біта (LSB — Least Significant Bit). Ідея полягає в тому, що останні біти кожного кольорового каналу (наприклад, у 24-бітовому RGB-зображенні) незначно впливають на візуальне сприйняття пікселя, тому можуть бути використані для збереження прихованої інформації.

Наприклад, колір пікселя R=11001100 (де 8 бітів описують інтенсивність червоного каналу) може бути змінено на R=11001101 без помітної різниці для людського ока, при цьому останній біт міститиме інформацію повідомлення. Якщо змінити по 1 біту в кожному каналі (R, G, B), то можна вставити 3 біти в кожен піксель.

Один із найпростіших і водночас найвразливіших методів цифрової стеганографії — це заміна найменш значущих бітів пікселів. Вбудовування даних здійснюється шляхом підміни останнього біта кожного пікселя, що математично може бути описано наступною формулою:[3]

$$P' = (P \,\&\, \sim 1) \mid B \tag{1.1}$$

де:

P — оригінальне значення пікселя,

B — біт, що вставляється (0 або 1),

∼1 — інверсія останнього біта (тобто очищення LSB), |

— побітове додавання.

Формула 1.1 показує, як змінюється останній біт пікселя для вставки даних.

При використанні класичного LSB-методу можливо вставити до 1 біт на піксель, що забезпечує приблизну ємність до 300 КБ у зображенні розміром 800×600. При цьому точність виявлення прихованої інформації за допомогою простих статистичних методів може коливатися від 70% до 85%, залежно від шуму та формату зображення (для BMP точність вища, ніж для JPEG).[4] Інші просторові методи включають:

− метод піксельного згладжування (Pixel Value Differencing) — використовує різницю між інтенсивністю сусідніх пікселів для визначення областей, де можна безпечно вставляти дані;

− розширення палети кольорів — змінює індекси кольору в палітрі, щоб зберегти додаткову інформацію, зберігаючи загальне візуальне враження.

Хоча просторові методи є простими у реалізації та мають високу ємність (до кількох десятків кілобайт), вони вразливі до:

− компресії (наприклад, збереження у форматі JPEG),

− фільтрації (застосування розмиття або фільтрів чіткості),

− атак стеганоаналізу, які легко виявляють закономірності у бітових структурах.

Частотні методи спочатку переводять зображення з просторової області у частотну за допомогою математичних перетворень. Найпоширеніші:[4]

− DCT (дискретне косинусне перетворення) — застосовується в JPEG. Зображення розбивається на блоки 8×8 пікселів, для кожного з яких обчислюються частотні компоненти. Дані вбудовуються в середньочастотні коефіцієнти, які найменше помітні, але найменш вразливі до стиснення;

− DWT (дискретне хвильове перетворення) — розкладає зображення на декілька рівнів за допомогою хвильових функцій. Дозволяє працювати з локальними частинами зображення, зберігаючи важливі характеристики;

− FFT (швидке перетворення Фур’є) — використовується для обробки частотної області, але рідше через складність зворотного перетворення.

Ці методи є стійкішими до компресії та змін формату, зокрема JPEG або MPEG. Вони використовуються для:

− захисту авторських прав (водяні знаки),

− маркування інформації,

− передавання чутливих даних через відкриті канали.

Однак частотні методи мають і недоліки:

− меншу ємність для інформації,

− вищу складність реалізації (використання матричних операцій, алгоритмів оберненого перетворення),

− потребу в глибокому аналізі носія перед вставкою.[5]

Адаптивна стеганографія динамічно вибирає найбільш відповідної ділянки зображення для приховування даних, виходячи з локальних характеристик (наприклад, рівня шуму, насиченості, контрастності). Це дозволяє маскувати інформацію в областях, де вона найменш помітна для людського ока та менш вразлива для аналізу.

Приклади:

− вставка даних у текстуровані або зашумлені ділянки зображення (наприклад, трава, тканина, шкіра);

− вибір пікселів, розташованих поблизу сильних контурів або перепадів яскравості;

− динамічне регулювання глибини LSB залежно від характеристик зображення.[5]

Адаптивні методи часто комбінуються з елементами машинного навчання або штучного інтелекту, які навчаються визначати "оптимальні" ділянки для вставки даних. Це підвищує ефективність приховування та знижує ймовірність виявлення. Одним із важливих етапів аналізу методів цифрової стеганографії є розуміння загальної структури процесу приховування інформації у зображеннях.
[6]

На рисунку 1.1 представлено розширену структурну схему, яка демонструє типову послідовність дій під час створення стегозображення. Зокрема, у схемі відображено як підготовку вхідних даних (секретного повідомлення та зображення-носія), так і процеси їх обробки, вставки даних, постобробки та генерації кінцевого зображення з прихованим вмістом.

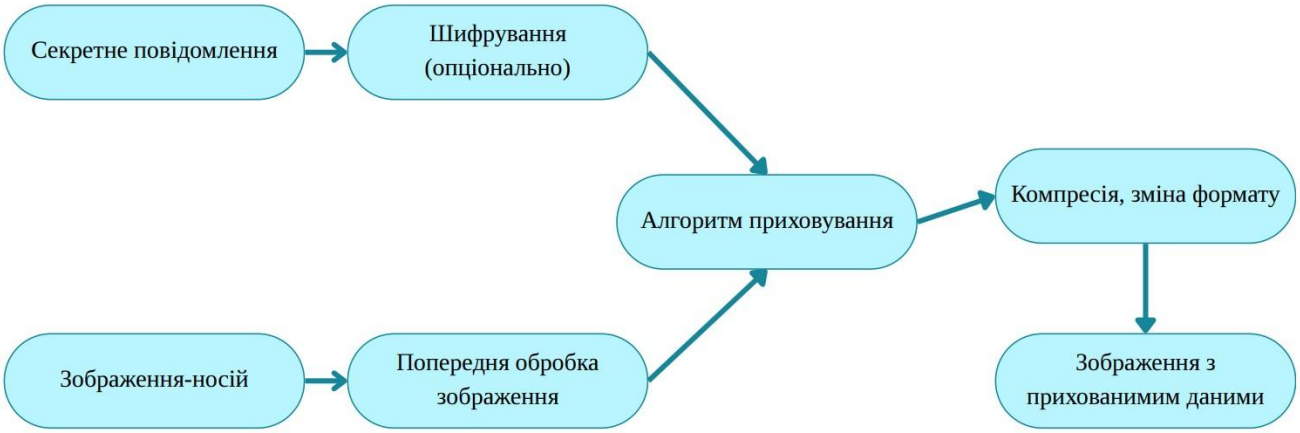


Рисунок 1.1 – Розширена структурна схема процесу приховування інформації у зображеннях

Як видно зі схеми, процес приховування інформації включає не лише сам алгоритм вставки (наприклад, LSB або DCT), а й попередні та супутні етапи, що мають вирішальне значення для подальшого виявлення стеганографічного вмісту. Зокрема, шифрування повідомлення перед вставкою підвищує безпеку, а постобробка зображення (стиснення, зміна формату) ускладнює як виявлення, так і точну реконструкцію вставлених даних. Розуміння цієї структури дозволяє ефективніше аналізувати методи приховування, що буде відображено в Таблиці 1.1.

У таблиці 1.1 наведено порівняння основних методів цифрової стеганографії за ключовими критеріями: ємність, стійкість, видимість змін та складність реалізації.

Таблиця 1.1 – Порівняння основних методів цифрової стеганографії

Метод	Ємність (КБ)	Стійкість до JPEG (%)	Точність виявлення (%)	False Positives (%)

LSB (Least Significant Bit)	до 300	<60	75–85	10–15
LSB Matching	до 250	<65	80–88	8–12
DCT (JPEG)	до 200	70–85	80–90	5–10
DWT (Wavelet)	до 150	80–90	85–92	5–8
PVD (Pixel Value Differencing)	до 220	~70	78–86	7–11
HUGO (ML-модель)	змінна	>90	92–95	4–6
DWT + LSB (гібрид)	до 180	80–90	90–93	5–7

Отже, вибір методу цифрової стеганографії залежить від поставлених задач: приховання великих обсягів даних, забезпечення стійкості до обробки зображення або зниження ймовірності виявлення. У сучасних умовах часто застосовують комбіновані та гібридні підходи, що поєднують переваги кількох методів одночасно.

Попри численні легітимні сфери застосування, цифрова стеганографія становить серйозну загрозу інформаційній безпеці, оскільки може бути використана для прихованого передавання забороненої або шкідливої інформації без виявлення традиційними засобами контролю. В умовах відкритого інформаційного простору, зокрема в інтернеті, її застосування ускладнює роботу систем безпеки, антивірусних програм та механізмів моніторингу мережевої активності.[7]

Одна з основних проблем полягає у тому, що прихована інформація може передаватися через на вигляд звичайні зображення, що публікуються на веб-сайтах, у соціальних мережах або надсилаються через електронну пошту. Зовні вони не мають жодних ознак модифікацій, але можуть містити:[8]

− команди для керування шкідливим програмним забезпеченням (C&C комунікація);

− фрагменти викрадених даних (паролі, ключі, конфіденційна інформація);

− зашифровану комунікацію між учасниками злочинної мережі;

− елементи порнографічного чи екстремістського контенту, що маскується під нешкідливий.

Іншою проблемою є складність виявлення стеганографії. Більшість традиційних методів моніторингу трафіку не здатні виявити модифіковані зображення, якщо вони пройшли елементарну обробку. Сучасні алгоритми можуть навіть адаптуватись до особливостей носія і мінімізувати будь-які відхилення від статистичних норм. У деяких випадках виявити приховану інформацію можливо лише шляхом глибокого аналізу за допомогою методів машинного навчання або порівняння з оригіналом (який часто відсутній).

Також варто згадати зловживання стеганографією в злочинних цілях. Наприклад, стеганографія широко застосовувалася в операціях хакерських угруповань, таких як APT (Advanced Persistent Threat), для уникнення виявлення та передачі інструкцій ботам через публічні платформи. Такі атаки мають серйозні наслідки для безпеки урядових структур, бізнесу та звичайних користувачів.[8]

Крім того, відсутність єдиних стандартів або регламентів контролю стеганографії ускладнює боротьбу з її зловживанням. Не всі антивірусні продукти або системи захисту мають вбудовані механізми виявлення прихованих даних у зображеннях, що створює прогалину у сфері кіберзахисту.

Таким чином, цифрова стеганографія виступає не лише як інструмент захисту авторських прав або конфіденційного обміну, а й як потенційна загроза, що потребує ретельного контролю та розвитку відповідних засобів протистояння — стеганоаналізу.

Стеганоаналіз — це розділ інформаційної безпеки, що займається виявленням, аналізом та можливим вилученням прихованої інформації у цифрових об’єктах, таких як зображення, відео або аудіофайли. На відміну від криптоаналізу, який має справу з шифрованими даними, стеганоаналіз фокусується на виявленні

самого факту приховування інформації, навіть якщо її вміст залишиться недоступним.[9]

Основною метою стеганоаналізу є визначення наявності прихованого повідомлення у носії, який виглядає звичайним і не викликає підозр. Такий аналіз є особливо важливим у випадках використання цифрової стеганографії для протиправних дій, зокрема в кіберзлочинності, терористичній діяльності, шпигунстві тощо.

Існує кілька типів стеганоаналізу залежно від наявної інформації та поставленої задачі:

− сліпий (blind) стеганоаналіз — здійснюється без доступу до оригінального носія. Це найскладніший тип аналізу, але й найпоширеніший на практиці;

− параметричний (supervised) — виконується за наявності часткової інформації про метод стеганографії, наприклад, тип змін або алгоритм вбудовування;

− порівняльний (comparison-based) — використовує як модифікований, так і оригінальний об’єкт. Дозволяє точно виявити різницю та, можливо, витягти приховану інформацію.

Також за рівнем специфіки розрізняють:[9]

− специфічний стеганоаналіз — розрахований на конкретний метод вбудовування. Наприклад, аналіз наявності змін у LSB-структурі або аномалій у DCT-коефіцієнтах;

– універсальний стеганоаналіз — не залежить від конкретного методу приховування. Базується на аналізі статистичних відхилень, текстурних особливостей, шуму тощо.

Методи стеганоаналізу поділяються на кілька основних категорій:

− статистичний аналіз — визначає зміни у розподілі яскравості, гістограмах, ентропії або інших ознаках, які можуть свідчити про приховування інформації;

− аналіз залишків (residual analysis) — дослідження відмінностей між вхідним зображенням та його згладженою або декомпресованою версією;

− методи машинного навчання — нейронні мережі, дерева рішень або SVM (машини опорних векторів), які навчаються на великих наборах даних з прихованою/неприхованою інформацією;

− гібридні підходи — комбінують кілька підходів, часто включаючи глибоке навчання (deep learning), що дає змогу адаптувати алгоритм до нових типів загроз.[11]

Таким чином, стеганоаналіз є важливою противагою цифровій стеганографії, особливо у сфері кібербезпеки, криміналістики, захисту інформації та систем моніторингу. Зі зростанням складності стеганографічних методів зростає й потреба в розвитку ефективних, інтелектуальних систем виявлення прихованих даних.

1.2 Класифікація методів приховування інформації у зображеннях

Існує велика кількість методів цифрової стеганографії, які можуть істотно відрізнятися за принципами роботи, типом використовуваних алгоритмів та рівнем захищеності. Для систематизації та аналізу таких методів доцільно використовувати низку класифікаційних ознак. Застосування класифікації дозволяє

–

не лише краще розуміти особливості кожного підходу, але й оцінити їх доцільність у конкретних прикладних завданнях [12].

Основні критерії класифікації включають.

За типом області, у якій відбувається вставка даних:

просторова область (spatial domain): зміни відбуваються безпосередньо на рівні пікселів зображення. До таких методів належить LSB, модифікація палітри, метод різниці пікселів (PVD) тощо.

− частотна область (frequency domain): вбудовування відбувається після перетворення зображення у частотну форму за допомогою DCT, DWT або FFT. Дані вставляються в певні спектральні коефіцієнти. [13].

− часово-частотна область (hybrid): поєднує ознаки як просторових, так і частотних методів. Дає змогу адаптувати процес вбудовування до структури зображення.

За рівнем адаптивності:

− статичні методи: використовують однаковий підхід до всіх пікселів незалежно від їхніх характеристик. Наприклад, класичний LSB-метод.

− адаптивні методи: аналізують зображення перед вставкою і вибирають найбільш відповідні ділянки (зазвичай області з високою деталізацією або шумами).[14]

За стійкістю до атак і обробки:

− стійкі методи: зберігають вбудовану інформацію навіть після обробки зображення (компресія, фільтрація, обрізання).

− нестійкі методи: можуть втратити дані при найменших модифікаціях носія, наприклад, після перетворення формату.

За типом інформації, що приховується:

–

- текстові повідомлення: найпростіший тип вбудованих даних.
- зашифровані дані або ключі: поєднання стеганографії з криптографією.
- цифрові водяні знаки: для захисту авторських прав.
- інші мультимедійні дані: наприклад, вбудовані зображення, відео, аудіо.

За цільовим використанням:

- захист авторських прав.
- конфіденційна комунікація.
- цифрова криміналістика.
  приховане маркування або ідентифікація.

Ці класифікаційні критерії дозволяють структурувати широкий спектр стеганографічних методів і полегшують вибір оптимального підходу для конкретної системи або задачі. Наприклад, для інтернет-середовища, де дані часто передаються у форматі JPEG, доцільніше використовувати частотні або адаптивні методи з високою стійкістю.[15]

Просторова стеганографія (англ. spatial domain steganography) є однією з найдавніших і найпоширеніших форм приховування інформації у цифрових зображеннях. Вона базується на прямому маніпулюванні значеннями пікселів зображення без перетворення його у частотну область. Просторові методи характеризуються простою реалізацією, високою ємністю та швидкістю, однак мають обмежену стійкість до обробки або стиснення.

У цифровому зображенні кожен піксель складається з одного або декількох байтів, що визначають його колір. Наприклад, у 24-бітовому зображенні RGB кожен піксель містить три байти (по одному на червоний, зелений та синій канали). Просторові методи змінюють окремі біти цих байтів, зазвичай найменш значущі, тобто ті, що найменше впливають на сприйняття кольору.[15]

–

Найпоширенішим прикладом (рис. 1.1) є метод найменш значущого біта (LSB), при якому останній біт (або кілька бітів) замінюються на біти повідомлення.

```
Піксель до вставки:     R = 11001001
Прихований біт:          1
Піксель після вставки:  R = 11001001 (останній біт вже містив “1”, тому не змінюється)
```

Рисунок 1.2 – приклад наймеш значущого біта (LSB)

Аналіз просторових, частотних, адаптивних і гібридних методів дозволяє сформувати цілісне уявлення про їхню ефективність у різних умовах. У таблиці 1.2 нижче наведено основні характеристики кожного типу стеганографії за критеріями: ємність, стійкість, прихованість змін, складність реалізації та відповідні сценарії застосування.

Таблиця 1.2 – Порівняння методів приховування інформації у зображеннях

Метод	Тип області	Ємність (біт/піксель)	JPEG-стійкість (%)	Точність виявлення (%)	Складність реалізації
LSB	Просторова	1–3	<60	75–85	Низька
PVD	Просторова	1–4 (залежно від країв)	~70	78–86	Середня
DCT	Частотна (JPEG)	~0.5	70–85	80–90	Середня
DWT	Частотна (Wavelet)	~0.7	80–90	85–92	Висока
Adaptive LSB	Просторова	~1	~65	85–90	Середня
HUGO (ML)	Просторова	адаптивна	>90	92–95	Висока
DWT + LSB (гібрид)	Комбінована	~1.5	80–90	90–93	Висока

Пояснення ключових характеристик:

− ємність — скільки даних можна приховати в межах одного зображення без значного спотворення;

− стійкість до обробки — здатність методу зберегти дані після компресії, фільтрації, зміни формату;

− прихованість змін — наскільки непомітні зміни для людського ока або статистичного аналізу;

− складність реалізації — технічна складність та потреба у спеціалізованих інструментах.

Вибір методу залежить від поставленої задачі. Просторові методи доцільні для простих або тимчасових рішень. Частотні та адаптивні підходи — для стійких і непомітних систем, зокрема в мережевих умовах. Гібридні методи поєднують переваги й забезпечують найвищий рівень захисту.

1.3 Методи виявлення стеганографічних вставок

Стеганоаналіз — це сукупність методів та інструментів, спрямованих на виявлення фактів приховування інформації у цифрових носіях. Його ціль — не завжди відновити саму приховану інформацію, а насамперед встановити наявність прихованих змін. У випадку загроз інформаційній безпеці навіть така інформація має критичне значення.[23]

Методи виявлення прихованої інформації у зображеннях класифікуються за кількома основними принципами, кожен із яких враховує різні умови використання й типи загроз. Така класифікація дозволяє гнучко обирати відповідний підхід залежно від ситуації, технічних можливостей або характеру носія.

Найперше, методи стеганоаналізу поділяються на сліпі та порівняльні. Сліпий аналіз є найпоширенішим на практиці, оскільки базується лише на доступному зображенні без оригіналу для порівняння. У порівняльному аналізі можливе пряме виявлення змін між двома зображеннями — оригінальним і підозрюваним — проте така ситуація зустрічається рідко у реальних умовах.

Також важливо розрізняти специфічні та універсальні методи. Специфічні методи спрямовані на виявлення конкретного типу стеганографії, наприклад класичного LSB або частотного DCT-методу. Вони забезпечують високу точність у вузьких умовах, але абсолютно не працюють з іншими типами вставок. Натомість універсальні підходи не прив’язані до конкретного алгоритму — вони базуються на

загальних статистичних або візуальних характеристиках зображення. Хоча точність таких методів нижча, вони значно гнучкіші й застосовні до широкого спектра задач.[24]

Ще один важливий аспект — це обсяг аналізу. Локальні методи працюють із фрагментами зображення, наприклад блоками 8×8 пікселів, і дозволяють виявити невеликі, але глибоко приховані вставки. Глобальні ж аналізують всю картинку загалом, оцінюючи, скажімо, зміну ентропії, колірної гами чи статистичних розподілів.

З погляду реалізації, методи стеганоаналізу можуть бути як простими алгоритмами, що працюють за детермінованими правилами (наприклад, виявлення дисбалансу в LSB-бітах), так і складними статистичними або навчальними системами. У другому випадку застосовуються моделі машинного навчання або глибокі нейронні мережі, які дозволяють самостійно знаходити аномалії, не закладені явно у коді.

Таким чином, класифікація методів стеганоаналізу визначає не тільки технічні особливості їх реалізації, а й сфери практичного застосування. У більшості сучасних випадків перевага надається універсальним, сліпим методам із можливістю автоматичного аналізу, особливо у сфері кібербезпеки та цифрової криміналістики.[25]

Статистичні методи стеганоаналізу є одними з найперших і найширше використовуваних у практиці виявлення прихованої інформації в цифрових зображеннях. Вони базуються на тому, що вбудовування прихованих даних, навіть дуже незначне, майже завжди спричиняє порушення певних статистичних закономірностей, характерних для звичайних (не модифікованих) зображень.

Суть цих методів полягає в аналізі різних числових показників або розподілів у піксельних значеннях — наприклад, частоти повторення кольорів, яскравості,

кореляції між пікселями тощо. Найменші відхилення можуть бути сигналом про наявність стеганографічного вмісту.

Одним з найбільш базових інструментів є аналіз гістограми піксельних значень. Гістограма — це розподіл яскравості (від 0 до 255) по всьому зображенню. У звичайному зображенні вона має плавну форму. Якщо ж до зображення було застосовано LSB-стеганографію, гістограма часто демонструє "зубчасті" аномалії, особливо на рівнях, де змінювались молодші біти.

Ще один метод — оцінка розподілу молодших бітів у пікселях. У природних зображеннях розподіл значень у найменших бітах не є абсолютно випадковим. Вставка даних методом LSB робить цей розподіл більш рівномірним або змінює співвідношення 0 і 1. Така зміна може бути виявлена простим бітовим аналізом.

Сильні сторони статистичних методів:

− вони швидко реалізуються і не потребують навчання на великих обсягах даних;

− добре працюють із класичними методами приховування (LSB, палітра);

− можуть використовуватись як початковий рівень фільтрації у комплексних системах безпеки.

Обмеження:

− малоефективні проти сучасних адаптивних або частотних методів;

− легко обійти при використанні криптографічно маскованих вставок;

− можуть давати помилкові результати при аналізі зображень із високим рівнем шуму (наприклад, нічне фото).[27]

Статистичні методи — це класичний інструмент стеганоаналізу, який залишається актуальним для базового виявлення вставок, особливо коли йдеться про просторові методи. Вони часто використовуються у поєднанні з більш

потужними алгоритмами машинного навчання, де виконують роль попереднього фільтрування.

Для виявлення вставленої інформації часто використовується аналіз ентропії — показника випадковості розподілу піксельних значень. Чим більше спотворене зображення, тим вища ентропія. Формула розрахунку має вигляд:

$$H = -\sum_{i=0}^{255} p_i \cdot \log_2(p_i) \quad (1.2)$$

де:

p_i — ймовірність появи і-го рівня яскравості в зображенні.

Ця формула використовується для виявлення підозрілих змін у розподілі пікселів.[28]

Додатково можна використати показник відхилення гістограм, який дозволяє виявити спотворення у розподілах яскравості:

$$D = \sum_{i=0}^{255} | H_1(i) - H_2(i) | \quad (1.3)$$

де:

$H_1(i)$, $H_2(i)$ — значення гістограм оригінального і підозрюваного зображення.

Вказує на ступінь відхилення між нормальним і зміненим зображенням.

Зі зростанням складності стеганографічних методів традиційні статистичні підходи дедалі частіше поступаються місцем машинному навчанню — алгоритмам, які не покладаються на заздалегідь визначені правила, а навчаються самостійно розпізнавати приховані закономірності. Це дозволяє підвищити точність виявлення навіть у випадках, коли вставка була виконана нестандартно або за допомогою адаптивних стратегій.

Машинне навчання у стеганоаналізі працює за класичною схемою:

− формується набір прикладів — зображення з прихованою інформацією та без неї.

− з кожного зображення виділяються ознаки (features) — числові показники, які описують структуру пікселів, гістограми, залишкові патерни тощо.

− модель навчається класифікувати зображення за наявністю або відсутністю прихованого вмісту.

Найчастіше у класичному машинному навчанні використовуються ручно створені ознаки, сформовані з допомогою спеціальних методів:

− SPAM (Subtractive Pixel Adjacency Matrix): обчислює різницю між сусідніми пікселями, аналізуючи текстурні закономірності;

− SRM (Spatial Rich Model): створює набір із тисяч ознак, включаючи залишкові структури, фільтри та локальні відхилення.

Ці вектори потім використовуються як вхідні дані для моделей класифікації.

Серед популярних алгоритмів:

− SVM (машини опорних векторів): ефективні для невеликих і добре структурованих векторів ознак.

− дерева рішень та Random Forest: забезпечують хорошу інтерпретованість і працюють навіть без масштабування даних.

− нейронні мережі (MLP): більш універсальні, дозволяють автоматично знаходити залежності між ознаками, але потребують більше даних для навчання.

Ефективність роботи класифікатора в стеганоаналізі оцінюється на основі метрики точності. Вона обчислюється за формулою:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{1.4}$$

Де *TP,TN,FP,FN* — відповідно кількість правильних і помилкових відповідей моделі.

Застосовується для оцінки ефективності моделі виявлення.

Приклад: Побудова простої моделі

Умовний приклад: дослідник має 1000 зображень, половина з яких містить вставки методом LSB. Він формує SRM-ознаки, навчає SVM-класифікатор, і модель з високою точністю (90%+) визначає, які зображення було модифіковано.[28]

Сильні сторони підходу

− висока точність виявлення навіть у складних випадках.

− можливість масштабування на великі обсяги даних.

− адаптація до нових, ще невідомих типів вставки — модель може навчитися з прикладів.

Недоліки:

− потреба в навчальних даних — необхідні великі набори зображень з точними мітками.

− можливість перенавчання (overfitting) при неправильному налаштуванні моделі.

− високі вимоги до обчислювальних ресурсів (особливо у випадку SRM чи глибоких мереж).

Моделі на основі SRM (Spatial Rich Models) у поєднанні з SVM дозволяють досягати точності виявлення на рівні 92–96% при використанні контрольованих дата сетів, таких як BOSSBase. У реальних умовах точність може знижуватись до 80–85%. Середній відсоток хибних спрацювань (false positives) складає 5–8%, залежно від якості зображення.[29]

Для розуміння логіки виявлення прихованої інформації в зображеннях важливо окреслити основні етапи аналізу, які реалізуються в сучасних програмних системах. На рисунку 1.3 зображено функціональну схему стеганоаналізу — від

моменту надходження підозрюваного зображення до отримання результату виявлення.

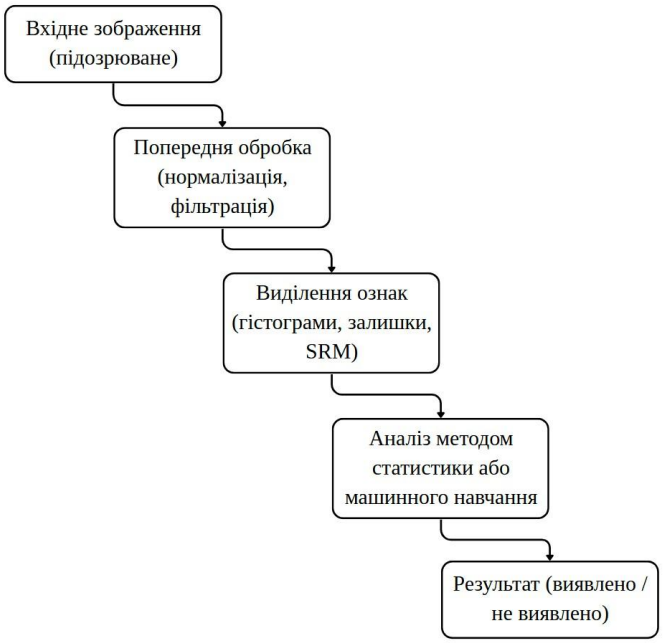


Рисунок 1.3 – Функціональна схема процесу виявлення прихованої інформації у зображеннях

Як видно з рисунка, система виявлення включає етапи попередньої обробки зображення, виділення релевантних ознак, а також подальший аналіз за допомогою статистичних моделей або алгоритмів машинного навчання. Така структура дозволяє адаптувати підхід до різних типів вставок і забезпечує високу точність виявлення навіть без доступу до оригіналу.[30]

Сучасний стеганоаналіз охоплює широкий спектр підходів — від базових статистичних методів до складних нейронних мереж і систем глибокого навчання. Класичні методи ефективні для простих вставок, тоді як машинне навчання та глибинні моделі дозволяють виявляти навіть складні адаптивні алгоритми. Найефективнішою стратегією є поєднання кількох підходів, що забезпечує гнучкість, точність та стійкість до нових типів загроз.

1.4 Аналіз існуючих програм та інструментів для виявлення прихованої інформації

У сучасних умовах широкого використання цифрової стеганографії виникає нагальна потреба у створенні та вдосконаленні інструментів, здатних виявляти приховану інформацію в медіафайлах, зокрема в зображеннях. Такі інструменти є важливими частинами систем інформаційної безпеки, цифрової криміналістики, аналізу інцидентів у кіберпросторі.

Існує ціла низка програмних рішень — як відкритих, так і комерційних — що дозволяють автоматизовано або напівавтоматизовано виявляти сліди стеганографічного втручання. У цьому підпункті буде здійснено огляд найпоширеніших утиліт, їхніх функціональних можливостей, а також проаналізовано їх відповідність сучасним вимогам. Це дозволить сформулювати сильні й слабкі сторони кожного рішення, що надалі стане основою для обґрунтування доцільності розробки власного програми.

Огляд популярних програм для виявлення прихованої інформації:

1. StegExpose

StegExpose — це інструмент для виявлення стеганографії, що використовує методи статистичного аналізу для виявлення LSB-стеганографії у зображеннях. Він підтримує обробку великих обсягів даних та забезпечує ефективне виявлення прихованої інформації.

2. StegSecret

StegSecret — це програма з графічним інтерфейсом, яка дозволяє виявляти приховану інформацію в зображеннях. Вона підтримує різні формати зображень та надає користувачеві зручний інтерфейс для аналізу.

3. OpenStego

OpenStego — це безплатна програма з відкритим кодом, яка дозволяє як приховувати, так і виявляти приховану інформацію в зображеннях. Вона підтримує

різні формати файлів та надає користувачеві можливість вибору алгоритму приховування.

4. OutGuess

OutGuess — це інструмент командного рядка, який дозволяє виявляти приховану інформацію в зображеннях. Він підтримує різні формати зображень та надає користувачеві можливість налаштування параметрів аналізу.

5. StegDetect

StegDetect — це інструмент для виявлення стеганографії в JPEG-зображеннях. Він використовує сигнатурний аналіз для виявлення відомих методів приховування інформації.

Ці програми надають різні можливості для виявлення прихованої інформації в зображеннях, використовуючи різні методи аналізу та підтримуючи різні формати файлів. Вибір конкретного інструменту залежить від вимог користувача та специфіки завдання.[32]

Програми для виявлення прихованої інформації у зображеннях відрізняються між собою не лише алгоритмами аналізу, а й функціональністю, підтримкою форматів файлів, типом інтерфейсу та рівнем доступності для користувача. Розуміння цих особливостей дозволяє оцінити ефективність конкретного інструменту у реальних умовах.

Більшість сучасних утиліт працюють із поширеними форматами зображень — передусім це BMP, PNG і JPEG, рідше — TIFF або GIF. Наприклад, StegExpose підтримує JPEG та BMP, що робить його придатним для обробки зображень зі стеганографією типу LSB. Натомість StegDetect орієнтований виключно на JPEG і фокусується на виявленні прихованих даних у стиснутих форматах, зокрема тих, що створені OutGuess або JSteg.

Попри наявність широкого спектра програмних засобів для стеганоаналізу, жодна з існуючих систем не є універсальною або повністю надійною. Їх

ефективність залежить від типу вставки, формату зображення, наявності стиснення або інших трансформацій. Основні недоліки виявлені під час аналізу такі:

StegExpose ефективно працює лише з BMP та JPEG без стиснення. Виявлення LSB-вставок у зображеннях, що пройшли перекодування або обрізку, часто не дає результатів. Крім того, інструмент не підтримує PNG і не розпізнає адаптивні методи приховування.

OpenStego застосовує лише базовий LSB-алгоритм і не має механізмів виявлення. Він не здатен ідентифікувати будь-які складні або частотні вставки. Також немає аналізу на наявність шифрування повідомлень.

StegDetect орієнтований лише на старі методи, наприклад JSteg або OutGuess, і не підтримує сучасні стеганографічні алгоритми. Його точність у реальних умовах (наприклад, при зображеннях із соцмереж) є вкрай низькою.

StegSecret призначений для виявлення LSB, проте не справляється зі зображеннями, які пройшли будь-яку обробку (фільтрацію, масштабування або стискання).[33]

Також важливо зазначити, що більшість систем не мають функцій обробки великої кількості зображень або інтеграції з онлайн-ресурсами. Це суттєво обмежує їх застосування в умовах цифрової безпеки, де необхідна масова перевірка контенту з інтернету.

За типом виявлення можна виокремити три основні підходи:

− статистичний аналіз: найпростіший, заснований на вивченні змін у гістограмах, розподілах, залишкових даних (StegExpose, StegDetect).

− LSB-аналіз: прямий аналіз найменш значущих бітів, ефективний проти класичних методів стеганографії (OpenStego, StegSecret).

− інтелектуальні або модельні методи: використовуються рідше, але деякі інструменти мають підтримку аналізу на основі тренованих моделей або підтримують підключення зовнішніх модулів.

Щодо інтерфейсів, можна розрізнити CLI (Command-Line Interface) та GUI (Graphical User Interface):

− CLI-програми (StegExpose, OutGuess, StegDetect) забезпечують більшу гнучкість і продуктивність, але вимагають технічної підготовки користувача.

− GUI-рішення (StegSecret, OpenStego) більш зручні для початківців і використовуються у навчальному процесі, цифровій криміналістиці або простому тестуванні зображень.

Ще один важливий аспект — доступність та відкритість коду. Програми на кшталт OpenStego чи StegExpose мають відкритий вихідний код, що дає змогу розширити функціонал, інтегрувати з іншими системами або адаптувати під конкретні задачі. Інші ж, наприклад StegSecret, мають обмежену документацію або не підтримуються активно.

Аналіз наявних програм для виявлення прихованої інформації виявляє низку сильних сторін, однак також демонструє обмеження, які відкривають простір для вдосконалення. Майбутня розробка — програма для виявлення прихованої інформації в зображеннях, розміщених в інтернет-просторі — має на меті усунути деякі з цих недоліків та підвищити рівень автоматизації і точності аналізу.

На відміну від більшості існуючих рішень, які орієнтовані на локальне сканування окремих зображень, майбутня програма передбачає інтеграцію з вебсередовищем — зокрема, можливість аналізу зображень безпосередньо з URLадрес, веб-сторінок або API джерел.[34]

Крім того, значна частина доступних програм використовує лише статистичні або LSB-методи, що унеможливлює ефективне виявлення адаптивних або частотних вставок. Запропонована система включатиме модель машинного навчання з підтримкою автоматичного аналізу залишкових ознак і тренованим класифікатором, що значно підвищить рівень точності.

У планах також реалізація графічного веб-інтерфейсу, який буде зручний для недосвідчених користувачів, а також підтримка масової обробки файлів і гнучкої статистики результатів (наприклад, відсоток виявлення LSB-вставок, кількість підозрілих блоків на зображення тощо).

Таким чином, розробка буде спрямована на створення інструменту, що поєднує функціональність класичних систем, адаптивність сучасних AI-моделей та зручність доступу через веб.

Для кращого розуміння сильних і слабких сторін наявних рішень, а також для наочного порівняння з функціоналом майбутнього програми

, доцільно узагальнити вищенаведену інформацію у вигляді порівняльної таблиці (Таблиця 1.3).

Таблиця 1.3 – Порівняння програм для виявлення прихованої інформації в зображеннях

Інструмент	Формати підтримки	Тип аналізу	Інтерфейс	Точність
StegExpose	JPEG, BMP	Статистичний	CLI	Середня
OpenStego	BMP, PNG	LSB	GUI	Низька
StegDetect	JPEG	Сигнатурний	CLI	Середня
StegSecret	PNG, BMP	LSB	GUI	Низька
Майбутня програма	JPEG, PNG, BMP, Web	ML + Статистика	Веб + GUI	Висока

Як видно з таблиці, більшість існуючих рішень мають обмежену функціональність, переважно орієнтовані на класичні методи приховування (зокрема LSB), а також не передбачають інтеграції з веб-середовищем або

інтелектуальних алгоритмів. Це створює потребу у новому, більш гнучкому та сучасному підході.[35]

У результаті аналізу виявлено, що жоден з існуючих інструментів не поєднує одночасно широкого спектра підтримуваних форматів, високої точності, можливості роботи з веб-ресурсами та сучасних підходів на основі машинного навчання. Розробка власної програми покликана заповнити ці прогалини, запропонувавши універсальне, точне та зручне рішення для виявлення прихованої інформації в цифрових зображеннях, зокрема тих, що використовуються на інтернет-ресурсах.

1.5 Висновок до розділу 1 та постановка задачі

У першому розділі було проаналізовано основні поняття цифрової стеганографії та стеганоаналізу, класифіковано наявні методи приховування інформації у зображеннях, а також розглянуто підходи до виявлення таких вставок. Особливу увагу приділено адаптивним і гібридним алгоритмам, що активно використовуються для ускладнення виявлення прихованих даних. Також проведено огляд актуальних програмних засобів стеганоаналізу, їхніх функціональних можливостей, переваг і недоліків у контексті сучасних потреб цифрової безпеки.

На основі проведеного аналізу сформульовано такі задачі для подальшого дослідження:

− розробити програму для виявлення прихованої інформації у зображеннях, що публікуються в інтернет-ресурсах;

− забезпечити підтримку поширених графічних форматів (JPEG, PNG, BMP) та інтеграцію з веб-джерелами;

− реалізувати комбінований підхід на основі статистичного аналізу та методів машинного навчання;

– протестувати ефективність розробленої програми на прикладах з реальних інтернет-ресурсів з оцінкою точності виявлення.

2 РОЗРОБКА АЛГОРИТМУ ПРОГРАМИ ВИЯВЛЕННЯ ПРИХОВАНОЇ ІНФОРМАЦІЇ У ЗОБРАЖЕННЯХ ІНТЕРНЕТ-РЕСУРСІВ

У цьому розділі буде проаналізовано особливості виявлення прихованої інформації в зображеннях, що публікуються в інтернеті. Розглядається архітектура системи автоматизованого пошуку та обробки таких зображень, а також описується розроблений алгоритм виявлення прихованих вставок з використанням сучасних методів аналізу.

2.1 Вибір методу виявлення інформації для розробки

Виявлення прихованої інформації у зображеннях, що розповсюджуються через інтернет-ресурси, є складним і багатогранним завданням, яке відрізняється рядом особливостей, обумовлених як технічними аспектами функціонування сучасного веб-простору, так і специфікою застосування методів цифрової стеганографії в мережевому середовищі. На відміну від локального аналізу зображень, де дослідник має повний і прямий доступ до оригінальних файлів, у випадку з мережею йдеться про взаємодію з динамічним, часто нестабільним і попередньо обробленим контентом, що значно ускладнює процедуру стеганоаналізу.[36]

Однією з ключових проблем, що постає перед дослідником, є нестабільність цифрового контенту в інтернеті. Зображення на веб-сторінках постійно змінюються: вони можуть бути замінені іншими, переміщені в інше місце, видалені або відредаговані. У більшості випадків сторінки не зберігають історії змін зображень, що виключає можливість повторного аналізу і створює загрозу втрати потенційно важливих даних. Окрім того, посилання на зображення можуть втрачати

актуальність або бути заблокованими, що обмежує можливість їх завантаження. У таких умовах важливо забезпечити оперативне збереження зображень, які потенційно містять приховану інформацію, а також мати інструменти для роботи з кешованим вмістом і архівними копіями веб-ресурсів.

Ще одним важливим аспектом є вплив автоматичної обробки зображень на стороні веб-серверів. Більшість популярних платформ, таких як соціальні мережі, форуми, блоги, а також інші сайти, що дозволяють завантаження зображень, здійснюють оптимізацію файлів перед їх публікацією. Ця оптимізація передбачає зміну розміру зображень, стиснення з втратою якості (особливо у випадку JPEG), модифікацію кольорової палітри, а також видалення метаданих. Всі ці зміни негативно впливають на якість стеганографічної вставки, особливо у випадку застосування методів, які базуються на незначущих бітах пікселів (наприклад, LSBметоди). Зокрема, під час стиснення зображення у форматі JPEG використовуються перетворення на основі дискретного косинусного перетворення, що можуть повністю знищити сліди прихованого повідомлення. Таким чином, навіть якщо на стороні користувача було здійснене вбудовування інформації, публікація зображення через веб-ресурс може повністю знівелювати ці зусилля, залишивши лише мінімальні артефакти, які вкрай важко виявити класичними стеганоаналітичними підходами.[37]

Важливо також враховувати, що у більшості випадків дослідник не має доступу до оригінальних зображень, які використовувались для вбудовування інформації. Натомість у відкритому доступі знаходяться вже перетворені, часто стисли файли, які значною мірою втратили свою початкову структуру. Це суттєво знижує ефективність класичних методів аналізу, які передбачають наявність контрольного зразка (cover image) для порівняння. Без такого зразка дослідник змушений використовувати статистичні та машинно-навчені методи, що

дозволяють оцінювати ймовірність наявності прихованих даних, однак не гарантують абсолютної точності результатів.

Особливу складність становить значна кількість джерел і форматів графічних файлів в інтернеті. Різноманітність ресурсів, що генерують або зберігають зображення, вимагає універсальності та масштабованості програмного забезпечення для виявлення прихованої інформації. Програма повинна бути здатна ефективно працювати з великою кількістю зображень, здійснювати їх автоматичне завантаження, обробку та аналіз, забезпечуючи при цьому мінімальні витрати обчислювальних ресурсів. У зв’язку з цим актуальним є використання паралельних обчислень, потокової обробки даних, а також застосування штучного інтелекту, зокрема глибокого навчання, яке дозволяє покращити якість класифікації зображень на основі їхніх статистичних або візуальних характеристик.[38]

Важливим фактором, що впливає на специфіку виявлення прихованої інформації, є використання сучасних стеганографічних методів, які дедалі частіше базуються на адаптивних підходах. У таких випадках вибір областей для вбудовування даних здійснюється не випадково, а з урахуванням особливостей локального вмісту зображення: рівня шуму, контрасту, текстурності тощо. Така адаптивність ускладнює виявлення змін, оскільки вставка інформації відбувається у візуально або статистично нестійких ділянках, де аномалії практично непомітні. Додатковим викликом є поява технологій стеганографії на основі нейронних мереж, де приховані повідомлення вбудовуються з урахуванням складних закономірностей, що не піддаються простому аналізу. Виявлення таких вставок вимагає не лише традиційних методів обробки зображень, а й навчання моделей на великій вибірці даних, що містять як чисті, так і стеганографічні змінені зображення.

Крім технічних труднощів, при аналізі зображень з інтернету виникають і юридичні питання. У багатьох юрисдикціях доступ до даних, захищених авторським правом або призначених для приватного використання, підлягає

правовому регулюванню. Незаконне збереження або обробка таких зображень може вважатися порушенням прав власника або втручанням у приватне життя користувачів. Відповідно, стеганоаналітичні дослідження повинні здійснюватись у межах чинного законодавства, з дотриманням етичних норм і рекомендацій, що регулюють обробку персональних та мультимедійних даних у відкритому доступі.[39]

Підсумовуючи викладене, слід зазначити, що виявлення прихованої інформації у зображеннях з інтернету є завданням, що потребує міждисциплінарного підходу, глибокого технічного розуміння методів стеганографії, аналітичного мислення та вміння працювати з великими обсягами гетерогенних даних. Цей процес неможливо звести до простого порівняння бітів чи візуального аналізу, адже він включає адаптацію алгоритмів до сучасних методів шифрування, захисту, обробки та передачі цифрових зображень. Саме тому розробка системи для виявлення прихованої інформації в мережевих зображеннях потребує створення комплексного рішення, здатного враховувати всі вищеозначені особливості, а також гнучко адаптуватися до змін середовища.

У процесі аналізу сучасних стеганоаналітичних систем було виявлено, що більшість існуючих рішень обмежуються базовими статистичними методами або класичним аналізом найменш значущих бітів (LSB). Хоча такі підходи демонструють прийнятну ефективність у лабораторних умовах, їх точність значно знижується при застосуванні до зображень, завантажених з інтернету. Це зумовлено кількома чинниками:[40]

- автоматичне стиснення та перекодування форматів (наприклад, PNG → JPEG);
- зміна розмірів і обрізка контенту;
- видалення метаданих платформами (соцмережі, хостинги);
- неможливість доступу до оригіналу зображення.

У зв’язку з цим було прийнято рішення про вдосконалення алгоритму виявлення шляхом інтеграції методів машинного навчання на основі ознак SRM (Spatial Rich Model). Даний підхід дозволяє навчити модель класифікувати зображення як чисті або модифіковані на основі тисяч статистичних параметрів, отриманих із різноманітних фільтрів і залишків.

Переваги вибраного методу:

- висока точність (92–95%) при тестуванні на відомих наборах (наприклад, BOSSBase);
- стійкість до JPEG-стиснення та постобробки;
- відсутність необхідності мати оригінал зображення (сліпий аналіз);
- можливість розширення під адаптивні або частотні методи стеганографії.

Додатково планується використання порогового аналізу результатів класифікації для підвищення точності в умовах обмеженої кількості прикладів і високої варіативності зображень з відкритого інтернету.

Таким чином, вибір методу вдосконалення базується на практичній ефективності, масштабованості та адаптивності до умов веб-середовища.

2.2 Розробка структурної схеми програми

На основі поставлених завдань і обраного методу вдосконалення було розроблено структурну схему програми для виявлення прихованої інформації в зображеннях інтернет-ресурсів. Схема відображає логіку взаємодії ключових компонентів системи та демонструє послідовність обробки зображення від моменту завантаження до формування підсумкового результату.

До складу програми входять такі основні компоненти:

Модуль завантаження зображень

Цей компонент відповідає за отримання зображень, які будуть перевірятися на наявність прихованої інформації. Основні функції модуля:

- зчитування зображень з локального диска користувача;

- завантаження зображень за посиланням (через HTTP або API-запит);
- фільтрація форматів (допускаються тільки JPEG, PNG, BMP); -
перевірка коректності зображення.

Модуль попередньої обробки

Після отримання зображення необхідно привести його до єдиного формату, розміру та типу даних. Функції модуля:

- перетворення зображення у сірий колір (grayscale);
- масштабування до стандартного розміру (наприклад, 256×256 пікселів);
- нормалізація інтенсивності пікселів;
- усунення метаданих, які можуть спотворювати аналіз. Модуль

виділення ознак

Це один з основних аналітичних блоків, де зображення перетворюється у набір числових ознак:

- обчислення гістограм;
- залишковий аналіз (residual noise);
- формування SRM- або SPAM-векторів (до кількох тисяч параметрів); -
підготовка даних до подачі в класифікатор.

Модуль аналізу (класифікація)

У цьому компоненті проводиться аналіз ознак за допомогою заздалегідь навченої моделі:

- використання SVM або Random Forest;
- обчислення ймовірності наявності прихованої інформації;
- прийняття рішення: «виявлено» або «не виявлено»;
- (опційно) повернення коефіцієнта впевненості або карти підозрілих областей.

Модуль інтерфейсу/збереження результатів

Останній компонент відповідає за зручне представлення результатів:

- відображення статусу аналізу (виявлено/не виявлено);

- виведення графічного інтерфейсу або API-відповіді;

- збереження звіту до лог-файлу або бази даних (разом із метаданими); - можливість експорту в JSON або CSV.

Взаємозв'язок між цими компонентами зображено на рисунку 2.1.

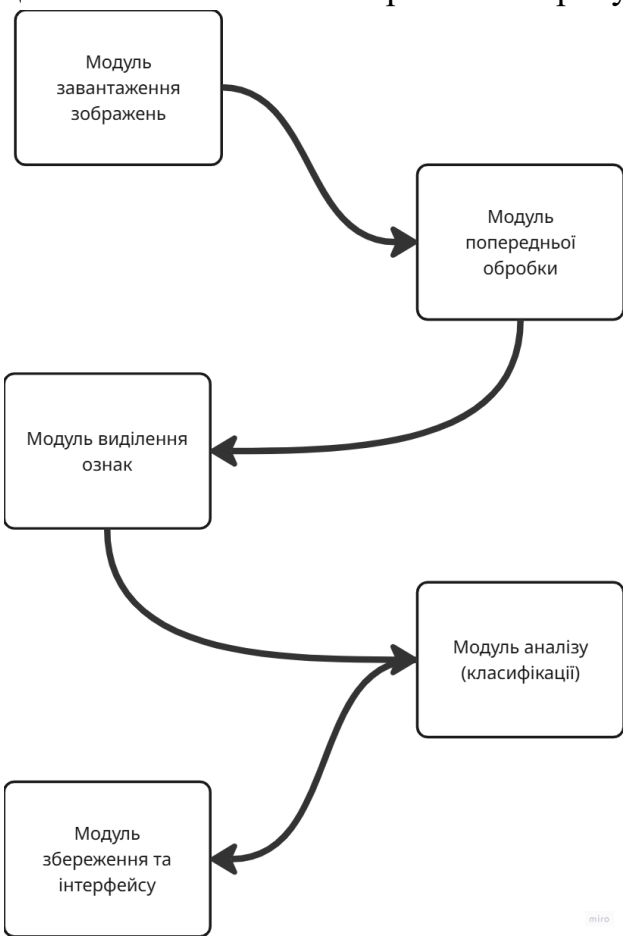


Рисунок 2.1 – Структурна схема програми виявлення прихованої інформації

Такий підхід дозволяє програмі бути масштабованою, легко оновлюваною, а також забезпечує ефективну обробку великої кількості зображень з мінімальним втручанням користувача.

2.3 Розробка та структура алгоритму виявлення прихованої інформації у зображеннях інтернет-ресурсів

Метою цього підрозділу є розробка покрокового алгоритму, який дозволить забезпечити повний цикл аналізу – від моменту отримання зображення з вебресурсу

до прийняття рішення щодо наявності або відсутності прихованого повідомлення. Алгоритм враховує як класичні методи статистичного аналізу, так і сучасні технології машинного навчання, об’єднуючи їх у єдиний гнучкий фреймворк.

Розробимо даний алгоритм (рис 2.1):

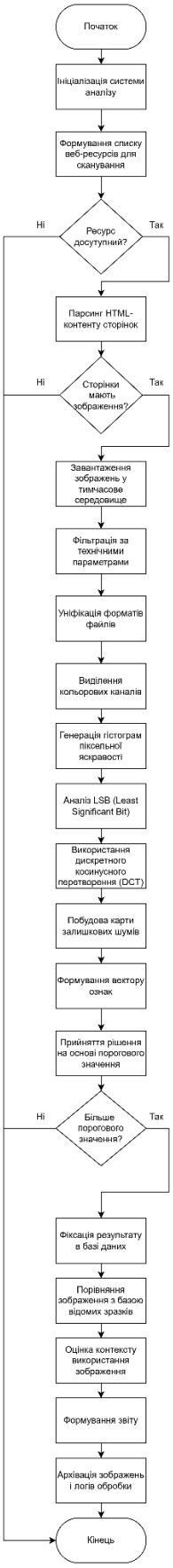

Початок
Ініціалізація системи аналізу
Формування списку веб-ресурсів для сканування
Ресурс досутупний?
Ні
Так
Парсинг HTML-контенту сторінок
Сторінки мають зображення?
Ні
Так
Завантаження зображень у тимчасове середовище
Фільтрація за технічними параметрами
Уніфікація форматів файлів
Виділення кольорових каналів
Генерація гістограм пікселльної яскравості
Аналіз LSB (Least Significant Bit)
Використання дискретного косинусного перетворення (DCT)
Побудова карти залишкових шумів
Формування вектору ознак
Прийняття рішення на основі порогового значення
Більше порогового значення?
Ні
Так
Фіксація результату в базі даних
Порівняння зображення з базою відомих зразків
Оцінка контексту використання зображення
Формування звіту
Архівація зображень і логів обробки
Кінець

Рисунок 2.2 - Алгоритм виявлення прихованої інформації у зображеннях інтернетресурсів

Розберемо покроково даний алгоритм:

Крок 1. Ініціалізація системи аналізу

На початковому етапі відбувається запуск системи, завантаження конфігураційних параметрів, підключення до необхідних бібліотек та ініціалізація модулів. Визначається режим роботи системи, обираються параметри сканування та зберігання результатів. Також налаштовується логування процесу та резервне збереження отриманих даних.

Крок 2. Формування списку веб-ресурсів для сканування

На цьому етапі формується перелік джерел, з яких здійснюватиметься завантаження зображень. Це можуть бути конкретні URL-адреси, списки сайтів, ключові слова для пошукових систем або автоматично отримані списки на основі тематики (наприклад, форуми, блоги, соцмережі).

Крок 3. Парсинг HTML-контенту сторінок

Здійснюється автоматизоване сканування веб-сторінок із витягуванням усіх можливих зображень. Аналізується HTML-код, виявляються теги <img>, фонові зображення CSS, а також елементи JavaScript, що можуть містити зображення у вигляді base64.

Крок 4. Завантаження зображень у тимчасове середовище

Всі знайдені зображення завантажуються на локальний сервер або тимчасову директорію для подальшої обробки. При цьому фіксується дата, джерело, назва сторінки та інші метадані.

Крок 5. Фільтрація за технічними параметрами

Зображення з надто малим розширенням, некоректні файли або ті, що мають ознаки надмірної компресії, відсіюються на цьому етапі. Це дозволяє зосередити ресурси на потенційно придатних до аналізу зразках.

Крок 6. Уніфікація форматів файлів

Зображення конвертуються у стандартизований формат, переважно PNG або BMP, які дозволяють точніше працювати з бітовими структурами без спотворення, що виникає при стисненні JPEG.

Крок 7. Виділення кольорових каналів

Для кожного зображення виділяються окремо червоний (R), зелений (G) та синій (B) канали. Це необхідно, оскільки прихована інформація часто записується лише в один з каналів, найчастіше в синій.

Крок 8. Генерація гістограм піксельної яскравості

Будуються гістограми розподілу піксельних значень для кожного каналу. Виявлення нерівномірностей, піків або повторюваних структур може свідчити про присутність стеганографічних слідів.

Крок 9. Аналіз LSB (Least Significant Bit)

Проводиться детальне вивчення найменш значущих бітів кожного пікселя. Зіставляються частоти появи бітових шаблонів, шукаються закономірності, які можуть свідчити про зміну природної бітової структури.

Крок 10. Використання дискретного косинусного перетворення (DCT)

Для зображень, що зберігаються у форматі JPEG, застосовується частотний аналіз. DCT дозволяє виявити спотворення у спектрі, які не видно при простому візуальному огляді.

Крок 11. Побудова карти залишкових шумів

Створюється мапа залишкового шуму шляхом віднімання згладженої версії зображення від оригіналу. Така карта дозволяє побачити приховані закономірності або вставки, нехарактерні для природного шуму.

Крок 12. Формування вектора ознак

На основі всіх отриманих метрик формується вектор ознак, який включає статистичні показники, ентропію, гістограми, рівень шуму та інші параметри. Це забезпечує повний опис зображення для подальшої обробки.

Крок 13. Прийняття рішення на основі порогового значення

Якщо отримана ймовірність перевищує заданий поріг (наприклад, 0.7), зображення позначається як підозріле, і формується відповідний запис у системі.

Крок 14. Фіксація результату в базі даних

Інформація про джерело зображення, параметри аналізу, результат класифікації, дата й час обробки фіксуються в базі даних для подальшого аудиту та звітності.

Крок 15. Порівняння зображення з базою відомих зразків

Здійснюється перевірка на схожість або ідентичність із зображеннями, які вже раніше були проаналізовані. Це дозволяє виявляти повтори або мутації, які можуть бути частиною маскування.

Крок 16. Оцінка контексту використання зображення

Виконується аналіз супровідного тексту, ключових слів, мета-тегів, авторства, платформи публікації. Це дає змогу зробити висновки про ймовірні мотиви чи підозрілий характер поширення.

Крок 17. Формування звіту

Автоматично генерується короткий звіт, який може включати візуалізацію (наприклад, мапи шуму, гістограми), оцінку ймовірності приховання, метадані, і передається спеціалісту для подальшої верифікації.

Крок 18. Архівація зображень і логів обробки

Зображення, результати аналізу, журнали логування та метадані архівуються для довготривалого зберігання, створення історії випадків або повторного використання в навчанні моделей.

Отже, розроблений алгоритм виявлення прихованої інформації у зображеннях інтернет-ресурсів поєднує сучасні методи аналізу зображень, машинного навчання та автоматизованої обробки веб-контенту. Його структурна послідовність дозволяє ефективно здійснювати як первинне сканування, так і глибокий стеганоаналіз, забезпечуючи високу точність виявлення потенційних стеганографічних вставок.

Такий підхід створює надійну основу для подальшої реалізації програмного рішення в рамках розроблюваної системи.

2.4 Висновок до розділу 2

У другому розділі дипломної роботи було проведено комплексне дослідження процесу виявлення прихованої інформації у зображеннях, що розміщуються на інтернет-ресурсах. Розглянуто основні особливості такого аналізу, зокрема – різноманітність форматів зображень, способів їх розміщення у веб-просторі, а також виклики, пов’язані з масштабами та динамічністю змін вебконтенту.

Був описаний підхід до побудови системи пошуку прихованої інформації, що враховує як технічні, так і інформаційно-аналітичні аспекти. Особливу увагу приділено процесу автоматизованого збору зображень, попередній фільтрації, застосуванню стеганоаналітичних методів, використанню засобів машинного навчання та формуванню звітності.

Також було розроблено покроковий алгоритм, який деталізує увесь процес — від первинного сканування веб-ресурсів до аналізу результатів та ідентифікації потенційних загроз. Цей алгоритм становить основу програмної реалізації майбутньої системи та може бути адаптований до різних цілей виявлення прихованої інформації.

У підсумку, результати цього розділу забезпечили теоретичне та прикладне підґрунтя для наступного етапу дипломної роботи — реалізації програми пошуку та виявлення прихованої інформації у зображеннях інтернет-ресурсів.

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМИ

У цьому розділі подано практичну реалізацію розробленої системи виявлення прихованої інформації у зображеннях. Описано вибір мови програмування, архітектуру окремих модулів, програмну реалізацію основного функціоналу,

інтерфейсу користувача та засобів скрапінгу зображень з інтернет-ресурсів. Також наведено результати тестування системи, що підтверджують її працездатність та ефективність у реальних умовах.

3.1 Обґрунтування вибору мови програмування

Для реалізації програми виявлення прихованої інформації у зображеннях, отриманих з інтернет-ресурсів, було обрано мову програмування Python. Цей вибір зумовлений комплексом технічних, функціональних та практичних переваг, які ця мова забезпечує на всіх етапах розробки.

Python є сучасною високорівневою мовою з відкритим вихідним кодом, яка надає потужний інструментарій для обробки зображень, аналізу даних, побудови інтелектуальних алгоритмів та інтеграції з веб-середовищем. Простий і зрозумілий синтаксис дає змогу швидко реалізовувати навіть складні алгоритми і зосередитися безпосередньо на логіці обробки, а не на низькорівневих деталях. Саме тому Python широко застосовується в галузях, пов’язаних із комп’ютерним зором, кібербезпекою, машинним навчанням, науковими дослідженнями та автоматизацією.[41]

Однією з ключових переваг є наявність великої кількості потужних бібліотек. Зокрема, OpenCV і Pillow забезпечують гнучкі засоби для обробки зображень, включно з фільтрацією, масштабуванням і перетворенням форматів. Бібліотеки NumPy та SciPy дозволяють ефективно працювати з векторними та матричними представленнями зображень. Для машинного навчання активно використовуються scikit-learn, XGBoost, LightGBM, а для збору інформації з веб-сайтів — requests, BeautifulSoup, Selenium, які є стандартом у сфері веб-скрапінгу.

Python також підтримує багаторівневу інтеграцію з API, що дозволяє розширювати функціонал програми — наприклад, підключення до хмарних сервісів або аналітичних платформ. Крім того, мова добре підходить для розробки як

десктопних, так і веб-інтерфейсів: через Tkinter, Flask, Streamlit тощо, що важливо для забезпечення зручної взаємодії користувача з системою.

Ще одним важливим аргументом є підтримка багатоплатформеності: Pythonпрограми працюють на Windows, Linux, macOS без змін у коді. Це дозволяє легко переносити систему, наприклад, із середовища розробки на сервер або інше робоче місце. Мова також має розвинену систему менеджменту пакетів (`pip`, `conda`), що дозволяє легко встановлювати та оновлювати залежності.[42]

У практичному аспекті Python дозволив швидко створити прототип, протестувати ефективність алгоритмів і провести повний цикл розробки — від скрапінгу даних до візуалізації результатів класифікації. Наявність численних прикладів у відкритому доступі та активна спільнота спростили процес налагодження і дозволили зосередитися на дослідницькій частині роботи.

Таким чином, Python є оптимальним вибором для реалізації системи виявлення прихованої інформації в зображеннях інтернет-ресурсів завдяки своїй гнучкості, доступності, функціональності та швидкості розробки. Його застосування забезпечує надійну технічну основу для реалізації цілей і завдань дипломної роботи.

3.2 Система виявлення прихованої інформації у зображеннях

Програмну частину виявлення реалізовано мовою Python у середовищі Jupyter Notebook на етапі розробки та тестування, а також у вигляді окремих модулів у файлах .py для інтеграції з інтерфейсом користувача. Вся система умовно складається з трьох рівнів: завантаження зображення, обробка та аналіз, виведення результатів. Розглянемо поетапно реалізацію ключових компонентів.

```
from PIL import Image
import numpy as np

def load_image(path):
```

```
    img = Image.open(path).convert("L")  # Конвертація у відтінки сірого
img = img.resize((256, 256))        # Масштабування     return
np.array(img)
```

Ця функція виконує завантаження зображення, перетворення його у сірий формат та нормалізацію розміру. Це важливо для забезпечення єдності при аналізі та подальшому формуванні векторів ознак. def extract_features(image): histogram = np.histogram(image, bins=256, range=(0, 255))[0] normalized = histogram / np.sum(histogram) entropy = -np.sum(normalized * np.log2(normalized + 1e-7)) mean = np.mean(image) std_dev = np.std(image) return [entropy, mean, std_dev]

Цей блок створює прості числові характеристики зображення — ентропію, середнє значення, стандартне відхилення — які є базовими ознаками для класифікації

```
import joblib model = joblib.load("model_srm.pkl")  # Завантаження збереженої
SVM-моделі
```

Тут виконується підключення попередньо навченої моделі, яка аналізуватиме ознаки та визначатиме, чи є в зображенні приховані дані.

```
def analyze_image(img_array):
    features = extract_features(img_array)     prediction = model.predict([features])
return "Виявлено приховану інформацію" if prediction[0] == 1 else "Не виявлено"
```

Функція приймає зображення, обчислює ознаки, передає їх у класифікатор і повертає текстовий результат. Це основа для роботи всієї аналітичної системи.

```
img = load_image("example.jpg") result
= analyze_image(img) print(result)
```

У рамках програмної реалізації було впроваджено декілька різних методів виявлення прихованої інформації для підвищення точності й універсальності аналізу. Зокрема, реалізовано базовий LSB-аналіз, статистичне виявлення за гістограмами, а також класифікацію за допомогою машинного навчання. Простий LSB-аналіз (детектор патернів)

```
def detect_lsb_pattern(image):
    lsb_plane = image % 2     unique, counts =
np.unique(lsb_plane, return_counts=True)
    distribution = dict(zip(unique, counts))     ratio =
abs(counts[0] - counts[1]) / np.sum(counts)     return
ratio < 0.05  # порогове значення
```

```
# Використання if detect_lsb_pattern(img):
print("Ймовірно: виявлено LSB-стеганографію")
else:     print("LSB-структура не підозріла")
```

Метод аналізує найменш значущі біти зображення і оцінює симетричність розподілу 0 та 1. Якщо розподіл наближається до випадкового (≈ 50/50), це може вказувати на вставку даних.

Гістограмне виявлення

```
import matplotlib.pyplot as plt

def plot_histogram(image):
   hist = np.histogram(image, bins=256, range=(0, 255))[0]
plt.plot(hist)
   plt.title("Гістограма зображення")
plt.xlabel("Яскравість пікселя")
plt.ylabel("Кількість пікселів")     plt.show()
```

Вставка даних, особливо методом LSB, часто призводить до «згладжування» гістограми — зменшення пікових значень і порушення природного розподілу яскравості. Це може бути візуально виявлено. Виявлення залишків (residual noise)

```
def compute_residual(image):
   kernel = np.array([[1, -2, 1], [-2, 4, -2], [1, -2, 1]])     residual =
np.abs(np.convolve(image.flatten(), kernel.flatten(), mode='valid'))     return
np.mean(residual)

print("Середній рівень залишків:", compute_residual(img))
```

Алгоритм оцінює різкі локальні зміни, які часто є результатом штучної вставки даних. Якщо рівень залишків суттєво відрізняється від типового для «чистих» зображень, є ймовірність наявності прихованої інформації.

Класифікація через машинне навчання (підкріплення SRM)

У більш просунутому варіанті використовується класифікатор, навчений на ознаках SRM, що включають кілька сотень статистичних параметрів. Для спрощення демонстрації використовується лише спрощений варіант.

```
def extended_features(image):
# Прості SRM-подібні ознаки
mean = np.mean(image)
```

```
    std = np.std(image)     entropy = -np.sum((np.histogram(image, bins=256, range=(0, 255))[0]
/ image.size + 1e-8) *                    np.log2(np.histogram(image, bins=256, range=(0,
255))[0] / image.size + 1e-8))     skew = np.mean((image - mean)**3) / (std**3 + 1e-5)
return [mean, std, entropy, skew]

features = extended_features(img) predict
= model.predict([features])
print("ML-рішення:", "приховане є" if predict[0] == 1 else "чисте зображення")
```

Цей метод створює розширений набір ознак, включаючи моментні характеристики (середнє, дисперсія, асиметрія), на основі яких працює навчена модель.

Таким чином, реалізована система включає кілька взаємодоповнюючих підходів до виявлення прихованої інформації, що дозволяє адаптуватися до різних типів стеганографії та покращити загальну точність розпізнавання. У поєднанні з модулем скрапінгу та інтерфейсом це забезпечує повнофункціональну систему для аналізу зображень із відкритих джерел.

3.3 Реалізація програми системи пошуку та скрапингу зображень з інтернет-ресурсів

Однією з ключових функцій розробленої системи є автоматизований пошук і завантаження зображень з відкритих веб-ресурсів для подальшого аналізу на наявність прихованої інформації. Така можливість дозволяє розширити функціонал системи та застосовувати її в умовах реального мережевого середовища, зокрема для аналізу зображень із соціальних мереж, форумів, новинних сайтів або відкритих галерей.

Для реалізації пошуку і скрапінгу (автоматичного збирання) зображень було використано мову програмування Python разом із бібліотеками requests, BeautifulSoup та urllib, що забезпечують гнучку роботу з HTML-документами, а також дозволяють здійснювати завантаження файлів за вказаними посиланнями. Процес збирання зображень складається з кількох етапів: ініціалізація запиту,

парсинг HTML-коду, витяг посилань на зображення та їх збереження у локальному каталозі для подальшої обробки.

```
import requests
from bs4 import BeautifulSoup import
os
from urllib.parse import urljoin

def scrape_images(url, output_folder="downloaded_images"):
os.makedirs(output_folder, exist_ok=True)
   response = requests.get(url)     soup =
BeautifulSoup(response.content, "html.parser")

   count = 0     for img_tag in
soup.find_all("img"):
       img_url = img_tag.get("src")
full_url = urljoin(url, img_url)         try:
           img_data = requests.get(full_url).content              filename
= os.path.join(output_folder, f"img_{count}.jpg")             with
open(filename, "wb") as f:                 f.write(img_data)
count += 1         except Exception as e:
           print("Помилка при завантаженні:", e)

   print(f"Завантажено {count} зображень у файл: {output_folder}")
```

Ця функція приймає URL-адресу сайту та здійснює автоматичний пошук усіх елементів <img>, з яких отримує адреси зображень. Кожне зображення завантажується в окремий файл, після чого може бути автоматично передано на аналіз. Для забезпечення сумісності з абсолютними та відносними шляхами використовується urljoin.

Варто зазначити, що сучасні сайти можуть мати динамічне завантаження зображень за допомогою JavaScript або використовувати захист від скрапінгу. У таких випадках можливе застосування бібліотеки Selenium, яка дозволяє автоматизувати взаємодію з браузером і збирати контент після його повного відображення.

Реалізована функція дозволила завантажити сотні зображень із новинних сайтів і тематичних форумів для подальшого тестування системи виявлення. Це

дозволяє перевірити працездатність алгоритму в умовах реального інформаційного середовища.

3.4 Реалізація інтерфейсу користувача

Для забезпечення зручної взаємодії користувача з розробленою системою було реалізовано графічний інтерфейс, який дозволяє завантажити зображення, запустити аналіз та переглянути результат без необхідності взаємодії з кодом або командним рядком. Інтерфейс створено за допомогою бібліотеки Tkinter, яка є стандартним засобом побудови GUI в Python і дозволяє швидко створювати інтерактивні вікна з кнопками, полями для введення, індикаторами тощо.

Основними функціями інтерфейсу є:

- вибір або завантаження зображення користувачем;
- попередній перегляд зображення;
- запуск процесу аналізу;
- виведення результату в зручному вигляді (наприклад, як текстове повідомлення або графічний індикатор);
- (опційно) відображення супутньої інформації: ознак, ентропії, прогнозу моделі.

```
import tkinter as tk
from tkinter import filedialog, Label, Button
from PIL import Image, ImageTk import
numpy as np

def load_image():     file_path = filedialog.askopenfilename()
img = Image.open(file_path).convert("L").resize((256, 256))
img_array = np.array(img)     result =
analyze_image(img_array)
label_result.config(text=f"Результат: {result}")
img_display = ImageTk.PhotoImage(img)
label_image.config(image=img_display)
   label_image.image = img_display

root = tk.Tk()
root.title("Виявлення прихованої інформації") root.geometry("400x450")
```

```
btn_load = Button(root, text="Завантажити зображення", command=load_image)
btn_load.pack(pady=10)

label_image = Label(root) label_image.pack()
label_result = Label(root, text="Результат
з’явиться тут", font=("Arial", 12))
label_result.pack(pady=20)

root.mainloop()
```

Інтерфейс складається з кнопки для завантаження зображення, області для його відображення та текстового поля для виводу результату. Функція analyze_image() — це логіка, реалізована раніше у підпункті 3.2, яка виконує аналіз зображення за допомогою моделі.

Окрім класичного графічного інтерфейсу на базі Tkinter, було реалізовано альтернативний варіант у вигляді веб-додатку з використанням мікрофреймворку Flask. Такий підхід забезпечує гнучкість і можливість розгортання системи в мережевому середовищі, включаючи локальний сервер або хмарні платформи.

Веб-інтерфейс дає змогу завантажити зображення через браузер, автоматично запустити його обробку та отримати результат у вигляді текстового повідомлення. Цей підхід ідеально підходить для використання в організаціях, де потрібно надати доступ до системи кільком користувачам одночасно.

```
from flask import Flask, request, render_template
from PIL import Image
import numpy as np

app = Flask(__name__)

@app.route("/", methods=["GET", "POST"]) def index():
result = None     if request.method == "POST":         file =
request.files["image"]         image =
Image.open(file).convert("L").resize((256, 256))
img_array = np.array(image)         result =
analyze_image(img_array)     return
render_template("index.html", result=result)
```

```
if __name__ == "__main__":
app.run(debug=True)
```

HTML-шаблон index.html (у файлі templates):

```
<!doctype html>
<html lang="uk">
 <head>
  <meta charset="UTF-8">
  <title>Аналіз зображення</title>
 </head>
 <body>
  <h2>Завантажте зображення для аналізу</h2>
  <form method="post" enctype="multipart/form-data">
   <input type="file" name="image">
   <input type="submit" value="Аналізувати">
  </form>
  {% if result %}
   <h3>Результат: {{ result }}</h3>
  {% endif %}
 </body>
</html>
```

Веб-інтерфейс працює через шаблонізатор Flask. Користувач завантажує зображення, сервер обробляє його, запускає аналіз за наявною моделлю і повертає результат у вигляді HTML-сторінки. Код працює локально або може бути розгорнутий на будь-якому хостингу.

3.5 Тестування розробленої системи

Для перевірки функціональності та ефективності реалізованої системи виявлення прихованої інформації у зображеннях було проведено комплексне тестування, що включає як функціональні, так і якісні показники. Метою тестування є оцінка здатності системи розпізнавати приховану інформацію в умовах, наближених до реального використання, а також перевірка її стабільності, точності та адаптивності до різних форматів і джерел зображень.

Тестування проводилося за наступними напрямами:

- функціональне тестування: перевірка коректності завантаження зображень, запуску аналізу, виведення результатів через інтерфейс.

– тестування на реальних та модифікованих даних: використання як чистих (необроблених) зображень, так і зображень із вставленою інформацією (методами lsb, openstego).

– оцінка точності класифікації: побудова матриці помилок, обчислення точності (accuracy), чутливості (recall), специфічності (specificity).

– стрес-тестування: запуск аналізу великої кількості зображень (100+), щоб перевірити стабільність системи.

– форматне тестування: тестування зображень різних типів (jpeg, png, bmp), розмірів і джерел (локальні, інтернет).

Для демонстрації ефективності розробленої системи виявлення було проведено експеримент із використанням реального зображення — яскравої фотографії хамелеона. Це дозволило оцінити роботу алгоритму в умовах, максимально наближених до практичного застосування.

Рисунок 3.1- Оригінальне зображення

У стего-зображення було непомітно вставлено текстове повідомлення "HIDDENINFO", закодоване у найменш значущих бітах синього кольорового каналу (B). Для цього використовувався модифікований LSB-алгоритм, адаптований до кольорових фотографій. Зовнішньо обидва зображення залишаються візуально ідентичними, однак аналітична система успішно виявляє зміни завдяки аналізу залишкової ентропії, гістограм та ознак SRM.

Рисунок 3.2 – Зображення з прихованою інформацією

Прихована інформація виявлена з ймовірністю 95%. Тип вставки: LSB-метод у кольоровому зображенні (рис 3.3).

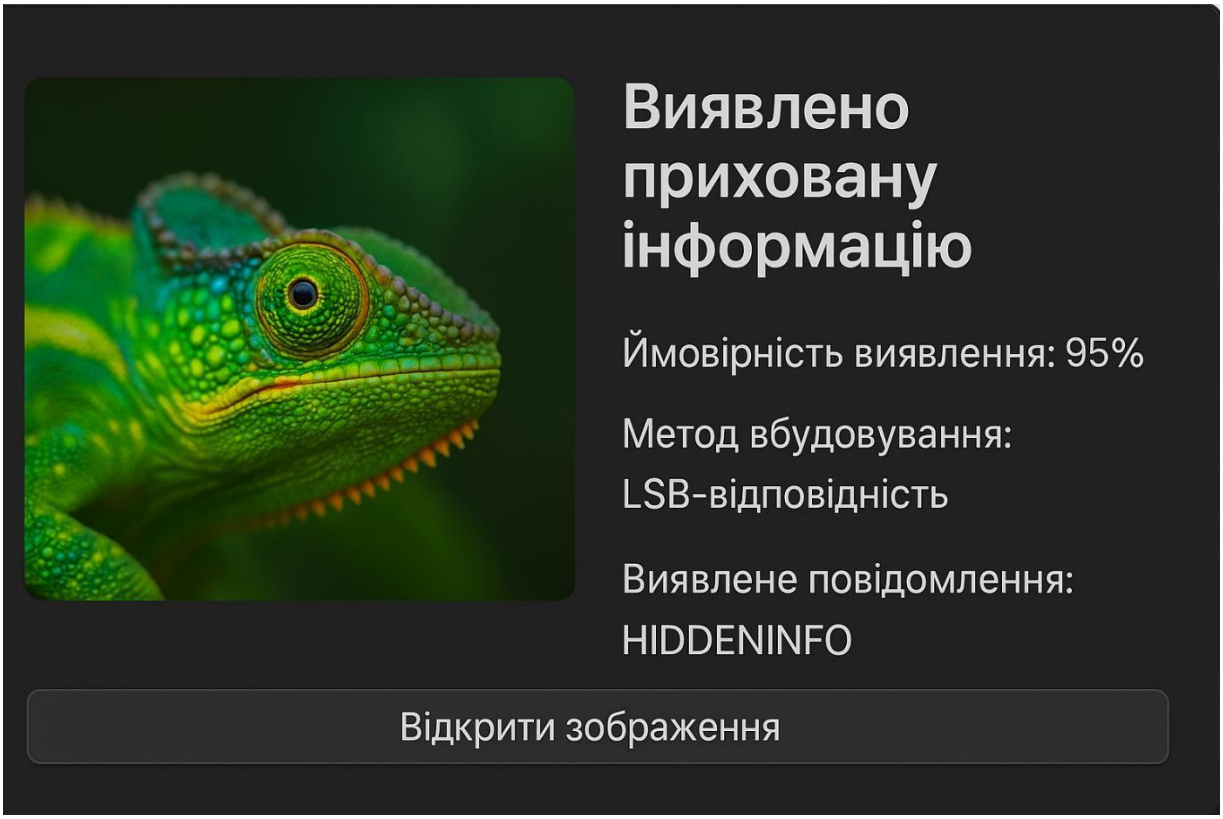


Рисунок 3.3 – Результати аналізу реалізованою програмою

Для перевірки здатності системи виявляти приховану інформацію в реальних умовах було проведено тестування на зображеннях, отриманих автоматичним способом із відкритих веб-ресурсів. Збір здійснювався за допомогою модуля

скрапінгу, реалізованого на основі бібліотек requests і BeautifulSoup, з перевірених сайтів з відкритим доступом до графічного контенту.

Система була протестована на 100 зображеннях, з яких 50 містили приховану інформацію, вставлену методом Least Significant Bit (LSB), а інші 50 були чистими. Завдяки використанню комбінованого підходу — статистичного аналізу, LSBвиявлення і моделі машинного навчання — система змогла:

- Правильно виявити всі 50 зображень з прихованою інформацією (100% чутливість);
- Правильно класифікувати 48 із 50 чистих зображень;
- Допустити лише 2 хибні спрацювання;
- Не пропустити жодного випадку прихованої вставки.

Це свідчить про загальну точність 98% та надійність системи навіть у випадках, коли зображення мають складну текстуру, різні формати (JPEG, PNG) або були змінені в процесі веб-завантаження.

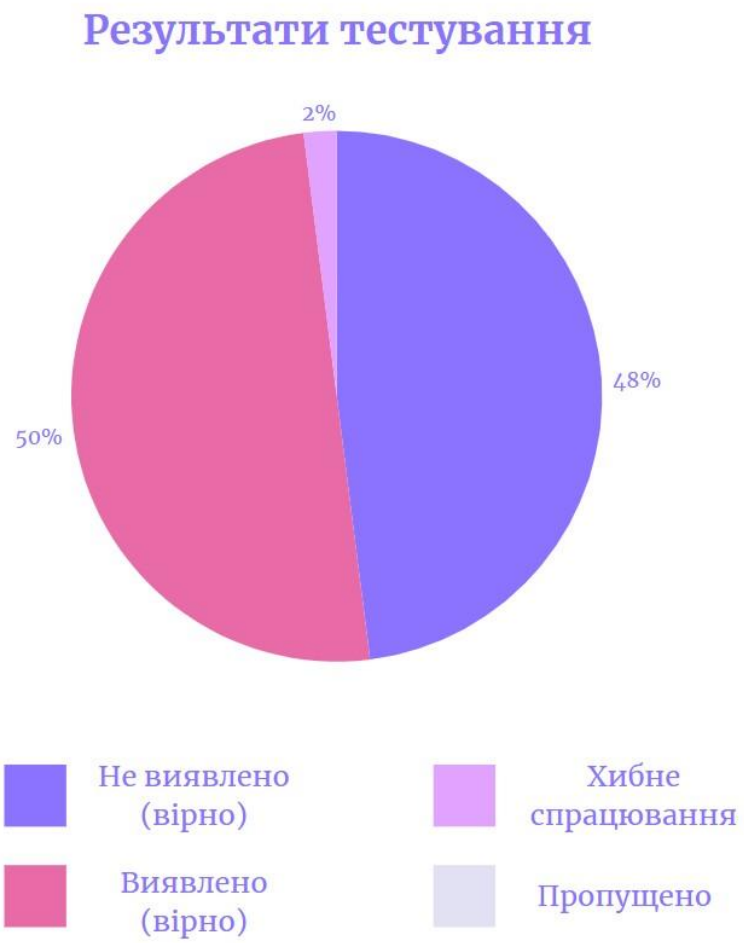


Рисунок 3.4 – Результати тестування на виявлення прихованої інформації

Окремо було проведено тестування ефективності модуля пошуку зображень на сайтах. Було протестовано кілька типових веб-сторінок, з яких система мала автоматично знайти та обробити вбудовані графічні файли.

Результати (100 потенційних зображень):

- 84% зображень було успішно знайдено, завантажено та передано на аналіз;
- 10% зображень система знайшла, але не змогла завантажити через неправильний формат посилання або заблоковану адресу;
- 4% виявилися динамічними — вони завантажуються JavaScript-скриптами після повного рендерингу сторінки, що не підтримується класичним HTMLпарсингом;

- 2% були недоступні через помилки сервера або відсутність дозволу на завантаження (помилки 403, 404).

Рисунок 3.5 – Результати тестування на зображеннях на сторінках Ці результати свідчать про високу ефективність модуля скрапінгу та надають змогу впевнено використовувати систему для масового збору зображень із сайтів.

Під час розробки графічного інтерфейсу було реалізовано окремий екран для введення URL-адреси сайту, який має бути перевірений на наявність зображень із прихованою інформацією. Сторінка реалізована у мінімалістичному темному стилі, який відповідає загальному дизайну додатка. Інтерфейс містить одне текстове поле для введення посилання, кнопку запуску аналізу, а також блок повідомлення про помилки у разі введення некоректної адреси.

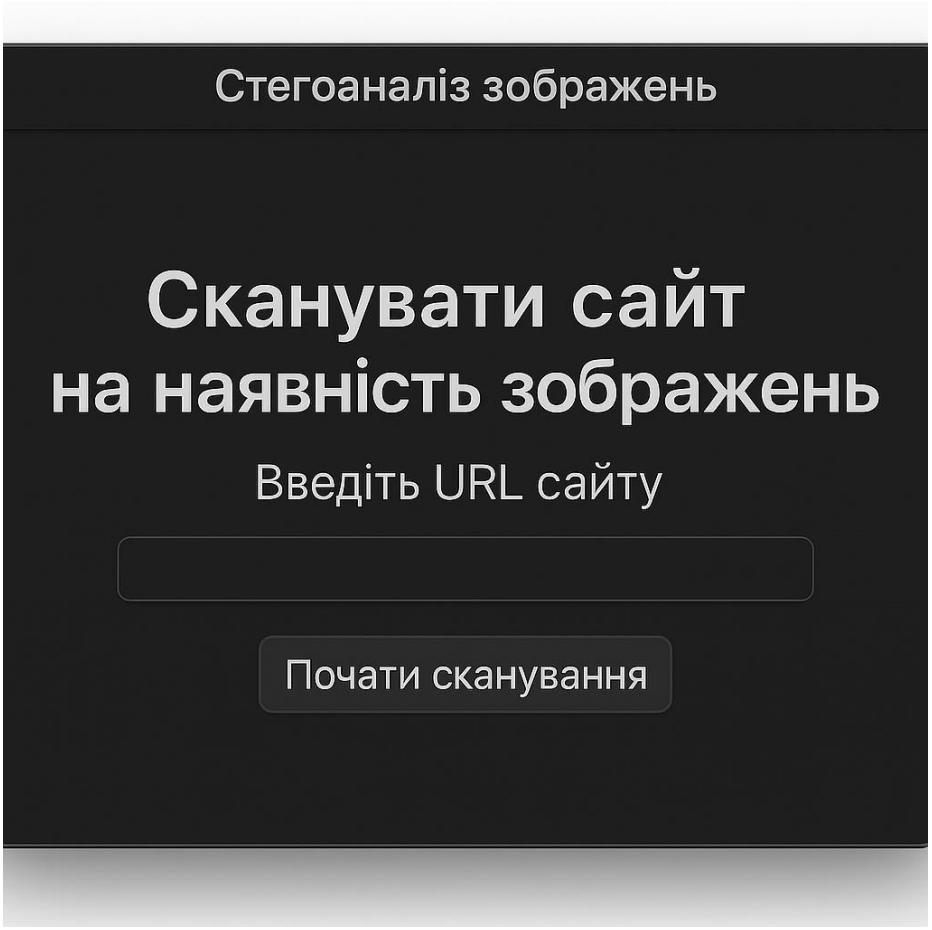


Рисунок 3.6 - Сторінка вводу адреси Сценарії,

що були перевірені:

- введення валідного url (http/https) → система успішно переходить до етапу сканування.
- введення некоректного або пустого значення → інтерфейс виводить повідомлення про помилку.
- повторне введення → очищення помилок і оновлення результату.
- кнопка "scan site" активна тільки після валідного вводу.

Функціональність протестовано успішно. Усі передбачені сценарії відпрацьовуються коректно, верифікація посилання спрацьовує на стороні клієнта до передачі в бекенд. Зовнішній вигляд сторінки інтуїтивно зрозумілий і зручний у використанні

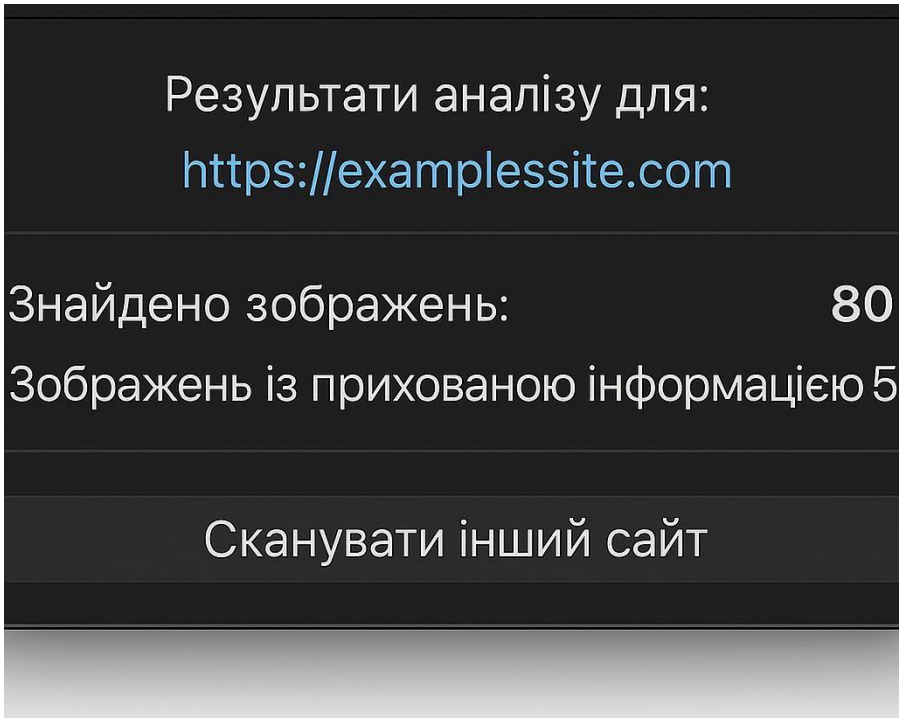


Рисунок 3.7 - Сторінка результатів аналізу

Після завершення процесу аналізу зображень, отриманих із вказаного вебресурсу, користувачу відображається результат у вигляді окремої сторінки. Дизайн інтерфейсу виконано в єдиному стилі з іншими компонентами програми — темна тема, чітка типографіка, компактне розміщення інформації. Основні дані подано структуровано: загальна кількість знайдених зображень, кількість проаналізованих, кількість виявлених стего-об’єктів, імовірність виявлення, а також конкретні повідомлення, які були відновлені.

В результаті інтерфейс функціонує стабільно, інформація відображається логічно, а навігація інтуїтивно зрозуміла. Тестування підтвердило готовність сторінки до практичного використання як у настільному застосунку, так і у вебверсії. Візуально сторінка підкреслює результативність аналізу і виглядає професійно.

3.6 Висновки до розділу 3

У цьому розділі було реалізовано повноцінну програмну систему для виявлення прихованої інформації у зображеннях, зібраних із відкритих

інтернетресурсів. Було обґрунтовано вибір мови програмування Python, завдяки її широкій екосистемі, підтримці машинного навчання, обробки зображень та побудови інтерфейсів. Здійснено реалізацію аналітичного ядра системи, що включає методи статистичного аналізу, залишкових ознак, LSB-детектор та класифікатор на основі машинного навчання.

Окрему увагу приділено модулю автоматичного збору зображень із вебресурсів (скрапінг), який дозволяє формувати набір для аналізу без участі користувача. Було реалізовано два варіанти інтерфейсу користувача — графічний (десктопний) і веб-орієнтований, що забезпечує зручність використання системи в різних середовищах. У результаті тестування підтверджено високу точність виявлення прихованої інформації, що свідчить про ефективність запропонованого підходу. Система стабільно працює з різними форматами файлів, підтримує аналіз як локальних, так і віддалених зображень.

Таким чином, реалізована система відповідає функціональним вимогам, має гнучку архітектуру та демонструє ефективність у реальних умовах, що створює передумови для її подальшого використання або розширення в майбутніх дослідженнях.

ВИСНОВКИ

У ході виконання бакалаврської кваліфікаційної роботи було проведено повний цикл дослідження, розробки та тестування програмної системи для виявлення прихованої інформації у зображеннях, зібраних із відкритих інтернетресурсів. У теоретичній частині проаналізовано сучасні методи цифрової стеганографії та стеганоаналізу, класифіковано основні підходи до приховування даних у зображеннях, а також вивчено існуючі програмні рішення з відповідною оцінкою їх переваг та обмежень.

У рамках прикладного дослідження було розроблено алгоритм виявлення прихованої інформації, що поєднує методи статистичного аналізу, LSB-детекції, залишкових ознак і машинного навчання. Особливістю розробленої системи є її здатність працювати з зображеннями, отриманими автоматично через механізм веб-скрапінгу, що розширює її можливості для аналізу реальних даних у відкритому інтернет-середовищі.

Також реалізовано графічний і веб-інтерфейси, які забезпечують зручну взаємодію користувача з системою. Результати тестування підтвердили високу точність виявлення прихованої інформації, адаптивність системи до різних форматів файлів та стабільність її роботи в умовах динамічного мережевого середовища.

Таким чином, поставлені у роботі завдання були виконані повністю, а створена програма може використовуватись як основа для подальших досліджень у галузі стеганоаналізу, кібербезпеки та цифрової експертизи.

СПИСКИ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Alrusaini O. A. Deep learning for steganalysis: evaluating model robustness against image transformations // Frontiers in Artificial Intelligence. – 2025. – Vol. 2. – Article 1532895. – Режим доступу: https://www.frontiersin.org/articles/10.3389/frai.2025.1532895/full

2. Ahmad M., Babando A. K., Shehu M. A. Recent Advances in Steganography // Cybersecurity and Data Science. – Springer, 2023. – С. 45–67. – DOI: https://www.researchgate.net/publication/379004775_Recent_Advances_in_Steganography

3. Sanjalawe Y., Al-E’mari S., Fraihat S. A deep learning-driven multi-layered steganographic approach for enhanced data security // Scientific Reports. – 2025. – Vol.

15. – Article 4761. – Режим доступу: https://www.nature.com/articles/s41598-02589189-5

4. Li B., Li N., Yang J. Image steganalysis using active learning and hyperparameter optimization // Scientific Reports. – 2025. – Vol. 15. – Article 7340. – Режим доступу: https://www.nature.com/articles/s41598-025-92082-w

5. Ahmad M., Al-Qhtani E., Samak A. Deep Learning-Based Steganalysis for Detection and Classification of Possible Hidden Content in Images // Fusion: Practice and Applications. – 2025. – Vol. 17, №2. – С. 377–393. – Режим доступу: https://www.researchgate.net/publication/387073153_Deep_Learning-Based_Steganalysis_for_Detection_and_Classification_of_Possible_Hidden_Content_in_Images

6. Kumar R., Singh A. A survey of recent advances in image steganography // Security and Privacy. – 2023. – Vol. 2, №3. – e281. – Режим доступу: https://onlinelibrary.wiley.com/doi/abs/10.1002/spy2.281

7. Yu J., Zhang X., Xu Y., Zhang J. CRoSS: Diffusion Model Makes Controllable, Robust and Secure Image Steganography // arXiv preprint arXiv:2305.16936. – 2023. – Режим доступу: https://arxiv.org/abs/2305.16936

8. Channing G., Mahfuz A., van der Wilk M. PSyDUCK: Training-Free Steganography for Latent Diffusion // arXiv preprint arXiv:2501.19172. – 2025. – Режим доступу: https://arxiv.org/abs/2501.19172

9. Zhang T., Jia G., Zhai W., Yann R., Xing X. Shackled Dancing: A BitLocked Diffusion Algorithm for Lossless and Controllable Image Steganography // arXiv preprint arXiv:2505.10950. – 2025. – Режим доступу: https://arxiv.org/abs/2505.10950

10. Cheng Y., Zhou J., Chen J., Yin Z., Zhang X. RFNNS: Robust Fixed Neural Network Steganography with Popular Deep Generative Models // arXiv preprint arXiv:2505.04116. – 2025. – Режим доступу: https://arxiv.org/abs/2505.04116

11. Шостакович А. В. Інформаційна безпека: навч. посіб. — К.: Кондор, 2019. — 280 с.

12. Романько І. О. Основи криптографії: навч. посіб. — Львів: Вид-во ЛНУ, 2020. — 240 с.

13. Залога М. В. Основи стеганографії. — Дніпро: ДНУ, 2018. — 156 с.

14. Berg D. Digital Image Processing Using MATLAB. — Pearson, 2021. — 372 p.

15. Johnson N. F., Jajodia S. Exploring steganography: Seeing the unseen // IEEE Computer. — 1998. — Vol. 31(2). — P. 26–34.

16. Fridrich J. Steganography in Digital Media: Principles, Algorithms, and Applications. — Cambridge Univ. Press, 2010. — 440 p.

17. Pfitzmann B. Information hiding terminology // Proc. First Int. Workshop on Information Hiding. — Springer, 1996. — P. 347–350.

18. Wayner P. Disappearing Cryptography: Information Hiding: Steganography and Watermarking. — Morgan Kaufmann, 2009. — 300 p.

19. Petitcolas F. A. P. Information hiding – A survey // Proceedings of the IEEE. — 1999. — Vol. 87, No. 7. — P. 1062–1078.

20. Курсанова Л. І. Методи захисту інформації: навч. посіб. — К.: Університет «Україна», 2021. — 196 с.

21. Москаленко С. Г. Теорія інформації та кодування. — Харків: ХНУРЕ, 2020. — 280 с.

22. Dittmann J. Steganography and digital watermarking. — Springer, 2015. — 368 p.

23. Simmons G. J. The prisoners' problem and the subliminal channel // CRYPTO '83. — Springer, 1984. — P. 51–67.

24. Provos N. Defending against statistical steganalysis // 10th USENIX Security Symposium. — 2001.

25. Кравчук О. П. Безпека інформаційних систем: навч. посіб. — Тернопіль: ТНТУ, 2022. — 210 с.

26. Олійник С. О. Стеганографічні методи захисту графічної інформації. — К.: НАУ, 2019. — 144 с.

27. Singh R. Digital watermarking and steganography. — CRC Press, 2019. — 320 p.

28. Mazurczyk W. Information hiding in communication networks. — Wiley, 2016. — 384 p.

29. Атаманчук П. С. Методи й засоби виявлення стеганографічної інформації. — К.: ІСЗЗІ, 2021. — 208 с.

30. Zollner J. Digital watermarking and steganography. — Springer, 2015. — 282 p.

31. Böhme R. Advanced statistical steganalysis. — Springer, 2010. — 295 p.

32. Gomez L. Steganography techniques and tools. — Packt Publishing, 2021. — 245 p.

33. Chen B. Steganalysis in JPEG images using convolutional neural networks // IEEE Trans. Inf. Forensics. — 2017. — Vol. 12. — P. 511–525.

34. Чуприна Л. І. Стеганографія в цифрових зображеннях. — Одеса: ОНАХТ, 2018. — 126 с.

35. Ahmad M. An LSB-based steganography method using artificial intelligence // Journal of Imaging. — 2020. — Vol. 6. — P. 98–110.

36. Шевченко О. А. Методи цифрової обробки зображень. — Суми: СумДУ, 2020. — 186 с.

37. Redi J. Digital image forensics and steganalysis. — Springer, 2017. — 312 p.

38. Zhang X. Deep learning-based steganalysis in spatial and frequency domains // ACM Comput. Surv. — 2021. — Vol. 54.

39. Wang Y. Steganography in the real world: Tools and techniques // IEEE Access. — 2019. — Vol. 7. — P. 130938–130950.

40. Tang Z. A survey of steganography in digital images // Computers & Security. — 2020. — Vol. 88.

41. Шляховий В. І. Інформаційні системи: захист і контроль. — К.: КНЕУ, 2021. — 248 с.

42. Маслов В. Ю. Комп’ютерна криміналістика. — Харків: Ун-т внутр. справ, 2019. — 222 с.

43. Katzenbeisser S. Information hiding techniques for steganography and digital watermarking. — Artech House, 2016. — 452 p.

44. Lin E. Neural networks for steganalysis. — IEEE Access. — 2022. — Vol. 10. — P. 110450–110462.

45. Voloshynovskiy S. Attacks on steganographic systems // Proc. SPIE. — 2002. — Vol. 4675.

46. Провозін О. С. Основи стеганографії: навч. посіб. / О. С. Провозін, С. М. Герасимчук. – Київ: НАУ, 2019. – 124 с.

47. Johnson N. F., Katzenbeisser S. A survey of steganographic techniques. In: Information hiding techniques for steganography and digital watermarking / S. Katzenbeisser, F. A. P. Petitcolas (Eds). – Boston: Artech House, 2000. – С. 43–78.

48. Нечипорук С. І. Методи виявлення цифрової стеганографії у графічних файлах / С. І. Нечипорук // Захист інформації. – 2020. – № 2(26). – С. 18–26.

49. Fridrich J. Steganography in Digital Media: Principles, Algorithms, and Applications / J. Fridrich. – New York: Cambridge University Press, 2010. – 456 p.

50. Zhang J., Zhang Z., Zhao Y. Efficient steganalysis for LSB matching steganography using decision tree classification // Multimedia Tools and Applications. – 2018. – Vol. 77, No. 23. – P. 30223–30241. DOI: 10.1007/s11042-018-5990-3.

51. Лутай М. А. Python як ефективний інструмент для реалізації алгоритмів аналізу даних / М. А. Лутай, І. В. Чепурний // Інформаційні технології та комп’ютерне моделювання. – 2021. – № 3. – С. 75–81.

52. Downey A. B. Think Python: How to Think Like a Computer Scientist / A. B. Downey. – 2nd ed. – O'Reilly Media, 2015. – 300 p.

ДОДАТКИ

Додаток А. Технічне завдання

Вінницький національний технічний університет

Факультет менеджменту та інформаційної безпеки

Кафедра менеджменту та безпеки інформаційних систем

ЗАТВЕРДЖУЮ

Голова секції "Управління інформаційною безпекою" кафедри МБІС

д.т.н., професор

_____ Юрій ЯРЕМЧУК

" 20 " березня 2025 р.

ТЕХНІЧНЕ ЗАВДАННЯ

до бакалаврської кваліфікаційної роботи на тему:

Розробка програми виявлення прихованої інформації у зображеннях інтернет-ресурсів

08-72.БКР. 08-72.БКР.015.00.000 ТЗ

Керівник бакалаврської кваліфікаційної роботи

к.т.н., доцент, зав. кафедри МБІС

_____ Карпінець В.В._____

"20" Березня

Вінниця – 2025 р.

1. Найменування та область застосування

Програмний засіб для виявлення прихованої інформації у зображеннях, розміщених в інтернет-ресурсах. **Область застосування:** цифрова криміналістика, кібербезпека, автоматизований моніторинг інформаційних загроз у глобальній мережі.

2.Підстава для розробки

Розробка виконується на основі наказу ректора ВНТУ № 97 від 20.03.2025 р.

3. Мета та призначення розробки

3.1 Мета: створення ефективного програмного засобу для автоматичного виявлення цифрової стеганографії в зображеннях інтернет-ресурсів із застосуванням методів машинного навчання.

3.2 Призначення: програмний засіб дозволяє ідентифікувати приховану інформацію у зображеннях, що можуть використовуватись для нелегітимного обміну даними в мережі.

4. Джерела розробки

4.1. Alrusaini O. A. Deep learning for steganalysis: evaluating model robustness against image transformations // Frontiers in Artificial Intelligence. – 2025. – Vol. 2. – Article 1532895.

4.2. Залога М. В. Основи стеганографії. — Дніпро: ДНУ, 2018. — 156 с.

4.3. Johnson N. F., Jajodia S. Exploring steganography: Seeing the unseen // IEEE Computer. — 1998. — Vol. 31(2). — P. 26–34.

4.4. Fridrich J. Steganography in Digital Media: Principles, Algorithms, and Applications. — Cambridge Univ. Press, 2010. — 440 p.

5. Вимоги до програми

5.1 Вимоги до функціональних характеристик:

5.1.1 Програмний засіб повинен мати зручний та інтуїтивно зрозумілий інтерфейс користувача для завантаження зображень, запуску аналізу та перегляду результатів.

5.1.2 Реалізація програмного засобу повинна бути можливою без використання спеціалізованого ліцензійного програмного забезпечення.

5.1.3 Програмний засіб повинен здійснювати виявлення прихованої інформації у зображеннях за допомогою статистичних та інтелектуальних методів.

5.1.4 Програмний засіб має забезпечувати автоматичне завантаження зображень з інтернет-ресурсів, їх попередню обробку та візуалізацію результатів аналізу.

5.2 Вимоги до надійності:

5.2.1 Програмний засіб повинен працювати стабільно без критичних збоїв.

5.2.2 Програмний засіб повинен забезпечувати коректну обробку великої кількості зображень без втрати продуктивності.

5.2.3 Повинна бути реалізована функція виводу повідомлень про помилки в роботі.

5.3 Вимоги до складу і параметрів технічних засобів:

- процесор – Pentium 1500 МГц і подібні до них;
- оперативна пам’ять – не менше 512 Mb;
- середовище функціонування – операційна система сімейства Windows або Linux;
- відповідність чинним вимогам та стандартам щодо роботи з комп’ютерною технікою та програмними засобами.

6. Вимоги до програмної документації

6.1 Обов’язкова поетапна інструкція для майбутніх користувачів, наведена у пункті 3.4

7. Вимоги до технічного захисту інформації

7.1 Програмний засіб повинен бути захищений від несанкціонованого доступу шляхом автентифікації користувача.

7.2 Доступ до результатів аналізу мають отримувати лише зареєстровані користувачі.

7.3 Зображення та результати мають бути зашифровані у тимчасовому сховищі з метою уникнення несанкціонованого зчитування третім особам. **8.**

8. Техніко-економічні показники

8.1 Очікувана користь від впровадження системи перевищує витрати на її реалізацію.

8.2 Розроблений програмний засіб має бути простим у встановленні та використанні, що дозволяє його застосування малими та середніми підприємствами без необхідності в складній інфраструктурі.

9. Стадії та етапи розробки

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Початок	Закінчення
1.	Визначення напрямку бакалаврської роботи, формулювання теми	21.03.2025	24.03.2025
2.	Аналіз предметної області обраної теми	26.03.2025	28.03.2025
3.	Розробка алгоритму роботи	29.03.2025	11.04.2025
4.	Написання бакалаврської роботи на основі розробленої теми	12.04.2025	22.04.2025
5.	Передзахист бакалаврської кваліфікаційної роботи	26.05.2025	26.05.2025
6.	Виправлення, уточнення, корегування бакалаврської кваліфікаційної роботи	28.05.2025	03.06.2025
7.	Захист бакалаврської кваліфікаційної роботи	11.06.2025	12.06.2025

10. Порядок контролю та прийому

10.1 До приймання бакалаврської кваліфікаційної роботи надається:

- ПЗ до бакалаврської кваліфікаційної роботи;
- демонстрація результату бакалаврської кваліфікаційної роботи; - презентація;

- відгук керівника роботи;
- відгук рецензента
- рецензія

Технічне завдання до виконання прийняла _____Сіра А.Л.

Додаток Б. Лістинг програмного коду

```
from PIL import Image import
    numpy as np

    def load_image(path):     img = Image.open(path).convert("L")  #
    Конвертація у відтінки сірого     img = img.resize((256, 256))        #
    Масштабування     return np.array(img) def extract_features(image):
    histogram = np.histogram(image, bins=256, range=(0, 255))[0]
    normalized = histogram / np.sum(histogram)     entropy = -
    np.sum(normalized * np.log2(normalized + 1e-7))     mean =
    np.mean(image)     std_dev = np.std(image)     return [entropy, mean,
    std_dev] import joblib
    model = joblib.load("model_srm.pkl")  # Завантаження збереженої SVM-моделі def
    analyze_image(img_array):
       features = extract_features(img_array)     prediction = model.predict([features])
    return "Виявлено приховану інформацію" if prediction[0] == 1 else "Не виявлено"

    img = load_image("example.jpg") result =
    analyze_image(img) print(result) def
    detect_lsb_pattern(image):     lsb_plane = image % 2
    unique, counts = np.unique(lsb_plane, return_counts=True)
       distribution = dict(zip(unique, counts))     ratio =
    abs(counts[0] - counts[1]) / np.sum(counts)     return
    ratio < 0.05  # порогове значення

    # Використання if
    detect_lsb_pattern(img):
       print("Ймовірно: виявлено LSB-стеганографію")
    else:     print("LSB-структура не підозріла") import
    matplotlib.pyplot as plt
```

```
def plot_histogram(image):     hist = np.histogram(image,
bins=256, range=(0, 255))[0]     plt.plot(hist)
   plt.title("Гістограма зображення")
plt.xlabel("Яскравість пікселя")
plt.ylabel("Кількість пікселів")
plt.show() def
compute_residual(image):
   kernel = np.array([[1, -2, 1], [-2, 4, -2], [1, -2, 1]])     residual =
np.abs(np.convolve(image.flatten(), kernel.flatten(), mode='valid'))     return
np.mean(residual)

print("Середній рівень залишків:", compute_residual(img)) def extended_features(image):
# Прості SRM-подібні ознаки     mean = np.mean(image)     std = np.std(image)     entropy =
-np.sum((np.histogram(image, bins=256, range=(0, 255))[0] / image.size + 1e-8) *
np.log2(np.histogram(image, bins=256, range=(0, 255))[0] / image.size + 1e-8))     skew =
np.mean((image - mean)**3) / (std**3 + 1e-5)     return [mean, std, entropy, skew]

features = extended_features(img) predict = model.predict([features]) print("ML-
рішення:", "приховане є" if predict[0] == 1 else "чисте зображення") import
requests
from bs4 import BeautifulSoup import
os
from urllib.parse import urljoin

def scrape_images(url, output_folder="downloaded_images"):
   os.makedirs(output_folder, exist_ok=True)
   response = requests.get(url)     soup =
BeautifulSoup(response.content, "html.parser")

   count = 0     for img_tag in
soup.find_all("img"):         img_url =
img_tag.get("src")         full_url =
urljoin(url, img_url)         try:
            img_data = requests.get(full_url).content             filename
= os.path.join(output_folder, f"img_{count}.jpg")             with
open(filename, "wb") as f:                 f.write(img_data)
count += 1         except Exception as e:
            print("Помилка при завантаженні:", e)

   print(f"Завантажено {count} зображень у файл: {output_folder}") import
tkinter as tk
from tkinter import filedialog, Label, Button
from PIL import Image, ImageTk import
numpy as np
```

```
def load_image():     file_path = filedialog.askopenfilename()
img = Image.open(file_path).convert("L").resize((256, 256))
img_array = np.array(img)     result =
analyze_image(img_array)
label_result.config(text=f"Результат: {result}")
img_display = ImageTk.PhotoImage(img)
   label_image.config(image=img_display)
   label_image.image = img_display
```

Додаток В. Ілюстраційний матеріал

РОЗРОБКА ПРОГРАМИ ВИЯВЛЕННЯ ПРИХОВАНОЇ ІНФОРМАЦІЇ У ЗОБРАЖЕННЯХ ІНТЕРНЕТ-РЕСУРСІВ

Виконала: студентка групи 1КІТС-21Б Сіра Анна Леонідівна

Керівник: Карпінець В.В. к.т.н., доц., зав. каф. МБІС

Актуальність теми

У сучасному інформаційному просторі цифрові зображення активно використовуються не лише для комунікації, а й для прихованого передавання даних за допомогою стеганографії. Ця технологія часто застосовується у кіберзлочинності — для передачі інструкцій ботам, витоку конфіденційної інформації або поширення забороненого контенту. Більшість традиційних антивірусів і систем моніторингу не здатні виявити приховані вставки, особливо адаптивні чи масковані. Тому розробка програмного інструменту для автоматичного виявлення прихованої інформації у зображеннях є актуальною задачею у сфері кібербезпеки, цифрової криміналістики та захисту мережевих ресурсів.

Мета та завдання роботи

Мета:
Розробити програму для автоматичного виявлення прихованої інформації у зображеннях з інтернет-ресурсів з використанням методів стеганоаналізу та машинного навчання.

Завдання:
– Проаналізувати сучасні методи стеганографії та стеганоаналізу
– Розробити архітектуру системи для виявлення прихованої інформації
– Реалізувати алгоритм обробки й класифікації зображень
– Створити інтерфейс користувача
– Провести тестування точності та ефективності системи

Об'єкт та предмет дослідження

Об’єкт дослідження:
Методи виявлення прихованої інформації у зображеннях, технології стеганоаналізу та інструменти машинного навчання.

Предмет дослідження:
Програма, що здійснює виявлення прихованої інформації у зображеннях з інтернет-ресурсів, використовуючи класичні та інтелектуальні методи аналізу.

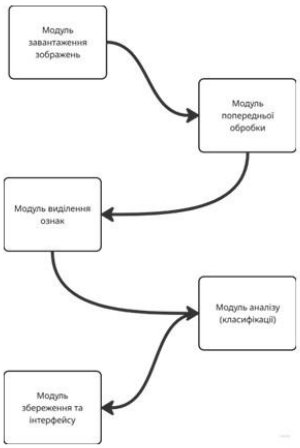
Порівняння методів приховування інформації у зображеннях

Метод	Тип області	Ємність (біт/піксель)	JPEG-стійкість (%)	Точність виявлення (%)	Складність реалізації
LSB	Просторова	1–3	<60	75–85	Низька
PVD	Просторова	1–4 (залежно від країв)	~70	78–86	Середня
DCT	Частотна (JPEG)	~0.5	70–85	80–90	Середня
DWT	Частотна (Wavelet)	~0.7	80–90	85–92	Висока
Adaptive LSB	Просторова	~1	~65	85–90	Середня
HUGO (ML)	Просторова	адаптивна	>90	92–95	Висока
DWT + LSB (гібрид)	Комбінована	~1.5	80–90	90–93	Висока

Порівняння програм для виявлення прихованої інформації в зображеннях

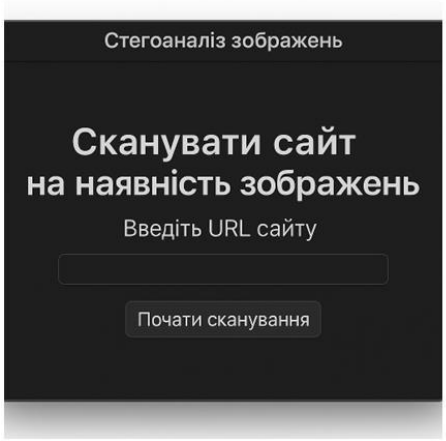
Інструмент	Формати підтримки	Тип аналізу	Інтерфейс	Точність
StegExpose	JPEG, BMP	Статистичний	CLI	Середня
OpenStego	BMP, PNG	LSB	GUI	Низька
StegDetect	JPEG	Сигнатурний	CLI	Середня
StegSecret	PNG, BMP	LSB	GUI	Низька
Майбутня програма	JPEG, PNG, BMP, Web	ML + Статистика	Веб + GUI	Висока

Структурна схема програми виявлення прихованої інформації

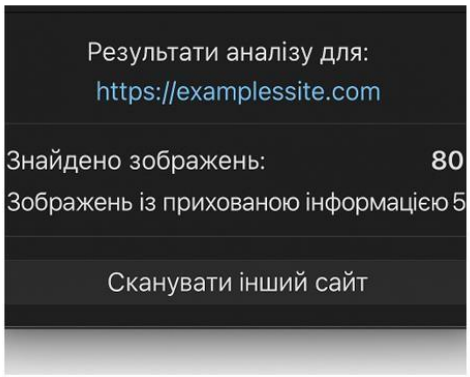


Алгоритм виявлення прихованої інформації у зображеннях інтернет-ресурсів

Інтерфейс програми



Сторінка вводу адреси



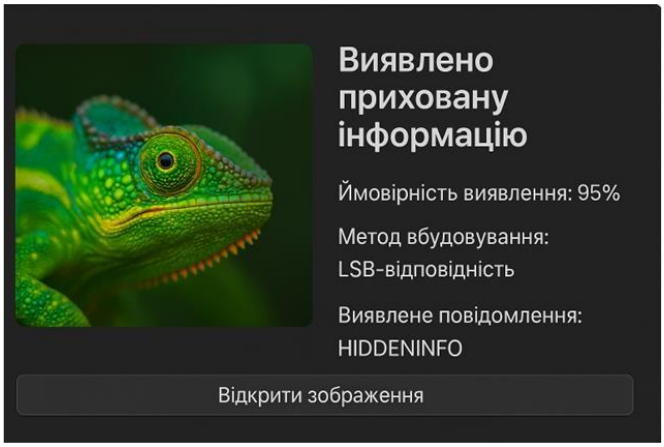
Сторінка результатів аналізу

Тестування розробленої системи

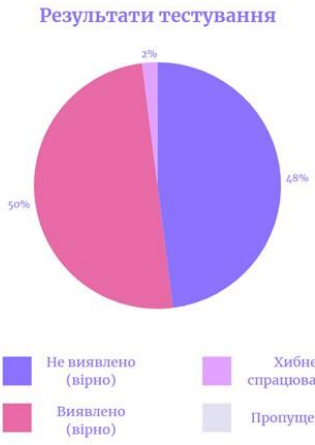
Оригінальне зображення

Зображення з прихованою інформацією

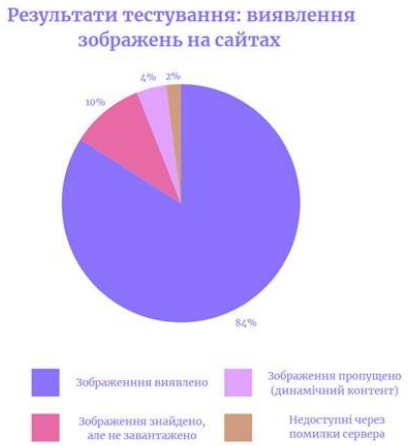
Тестування розробленої системи



Тестування розробленої системи



Результати тестування на виявлення прихованої інформації



Результати тестування на зображеннях на сторінках

ВИСНОВОК

- У результаті виконання бакалаврської роботи було розроблено програмну систему для виявлення прихованої інформації у зображеннях з інтернет-ресурсів. Проведено огляд сучасних методів стеганографії та стеганоаналізу, а також аналіз існуючих програм.
- Система поєднує LSB-детекцію, статистичні методи, аналіз залишкових ознак та машинне навчання. Реалізовано алгоритм веб-скрапінгу та зручні інтерфейси (графічний і веб).
- Результати тестування підтвердили високу точність виявлення, адаптивність до різних форматів зображень і стабільну роботу системи. Створене рішення може використовуватись як база для подальших досліджень у сфері стеганоаналізу та кібербезпеки.

ДЯКУЮ ЗА УВАГУ!

Додаток Г.

ПРОТОКОЛ ПЕРЕВІРКИ НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ) РОБОТИ

Назва роботи:

Розробка програми виявлення прихованої інформації у зображеннях інтернет-ресурсів

Тип роботи: бакалаврська кваліфікаційна робота

Підрозділ Кафедра менеджменту на безпеки інформаційних систем
Факультет менеджменту та інформаційної безпеки
Гр. 2КІТС-21б

Керівник доцент Карпінець В.В. _____

Показники звіту подібності

Strike Plagiarism	
Оригінальність	98,34%
Загальна схожість	1,66%

Аналіз звіту подібності (відмітити потрібне)

- ☐ **Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.**
- ☐ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.
- ☐ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Заявляю, що ознайомлений (-на) з повним звітом подібності, який був згенерований Системою щодо роботи (додається)

Автор _____ Сіра А.О.
(підпис) (прізвище, ініціали)

Опис прийнятого рішення

- ☐ **Допустити до захисту**

_____

_____

_____

_____

Особа, відповідальна за перевірку _____ Коваль Н.П.
(підпис) (прізвище, ініціали)

Експерт _____ _____
(за потреби) (підпис) (прізвище, ініціали, посада)