КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БУДІВНИЦТВА І АРХІТЕКТУРИ

Автоматизації і інформаційних технологій
(факультет)

Кібербезпеки та комп'ютерної інженерії
(кафедра)

КВАЛІФІКАЦІЙНА РОБОТА ЗДОБУВАЧА СТУПЕНЯ ВИЩОЇ ОСВІТИ БАКАЛАВР

на тему:

Методи виявлення та вилучення прихованої інформації, вбудованої у зображення з використанням стеганографії

Сергієнко Богдани Олександрівни

Київ 2026 р.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БУДІВНИЦТВА І АРХІТЕКТУРИ

Автоматизації і інформаційних технологій
(факультет)

Кібербезпеки та комп'ютерної інженерії
(кафедра)

ЗАТВЕРДЖУЮ
Завідувач кафедри
Делембовський М.М._____
„__” _____20___року

КВАЛІФІКАЦІЙНА РОБОТА ЗДОБУВАЧА СТУПЕНЯ ВИЩОЇ ОСВІТИ БАКАЛАВР

Методи виявлення та вилучення прихованої інформації, вбудованої у зображення з використанням стеганографії

Я як здобувач вищої освіти КНУБА розумію і підтримую політику закладу з академічної доброчесності. Я не надавав(-ла) і не одержував(-ла) недозволену допомогу під час підготовки цієї роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

Здобувач
Сергієнко Богдана Олександрівна
(прізвище, ім’я та по батькові повністю)
125 «Кібербезпека»
(спеціальність)
«Безпека інформаційних і комунікаційних систем»
(освітня програма)
Група БІКС - 23с
Керівник Шабала Є.Є.,
(прізвище та ініціали)
доцент кафедри кібербезпеки та комп’ютерної інженерії, кандидат технічних наук, доцент
(наукове звання)
Рецензент Терентьєв О.О.
(прізвище та ініціали)
Ідентичність підтверджую

Київ 2026 р.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БУДІВНИЦТВА І АРХІТЕКТУРИ

Факультет: Автоматизації і інформаційних технологій

Випускова кафедра: Кібербезпеки та комп'ютерної інженерії

Ступінь вищої освіти: бакалавр

Спеціальність: 125 «Кібербезпека»

Освітня програма: Безпека інформаційних і комунікаційних систем

ЗАТВЕРДЖУЮ

Завідувач кафедри

Делембовський М.М._____

„____” _____20___року

З А В Д А Н Н Я

ДО ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ ЗДОБУВАЧА СТУПЕНЯ ВИЩОЇ ОСВІТИ БАКАЛАВР

(бакалавр, магістр)

_____Сергієнко Богдани Олександрівни_____

(прізвище, ім’я та по батькові здобувача)

1. Тема роботи Методи виявлення та вилучення прихованої інформації, вбудованої у зображення з використанням стеганографії_

затверджена наказом ректора КНУБА №576/52-15/13.2/26 від 14.05.2026.

2. Керівник роботи

Шабала Євгенія Євгенівна, доцент кафедри кібербезпеки та комп’ютерної інженерії, кандидат технічних наук, доцент_____

(прізвище, ім’я та по батькові, науковий ступінь, вчене звання)

3. Термін подання здобувачем роботи до захисту _____

4. Зміст пояснювальної записки за розділами:

Р. 1. Аналіз предметної області та досвіду практики

Р. 2. Аналіз методів виявлення прихованої інформації

Р. 3. Розробка комбінованого методу виявлення та вилучення стеганографічних вкладень у синтетичних зображеннях

5. Календарний план виконання роботи:

Види робіт та їх зміст	Дата виконання
Розділ 1. Аналіз предметної області та досвіду практики	10.05.2026 р.
Розділ 2. Аналіз методів виявлення прихованої інформації	18.05.2026 р.
Розділ 3. Розробка комбінованого методу виявлення та вилучення стеганографічних вкладень у синтетичних зображеннях	28.05.2026 р.
Остаточне оформлення роботи	7.06.2026 р.
Направлення роботи для перевірки на плагіат	
Попередній захист роботи	
Направлення роботи на рецензування	

8. Дата видачі завдання _____

Керівник _____ _____

(підпис) (прізвище та ініціали)

Здобувач _____ _____

(підпис) (прізвище та ініціали)

РЕЗЮМЕ (SUMMARY) до *кваліфікаційної випускної роботи здобувача:*	Сергієнко Богдана Олександрівна Serhiienko Bohdana		
ЗВО	Київський національний університет будівництва і архітектури		
Тема (українською та англійською)	Методи виявлення та вилучення прихованої інформації, вбудованої у зображення з використанням стеганографії Methods for detection and extraction of hidden information embedded in images using steganography		
Освітній ступінь	Бакалавр		
Факультет	Автоматизації і інформаційних технологій		
Випускова кафедра	Кібербезпеки та комп'ютерної інженерії		
Спеціальність	125 «Кібербезпека»		
Освітня програма	Безпека інформаційних і комунікаційних систем		
Керівник	Шабала Євгенія Євгенівна		
Обсяг роботи:	пояснювальна записка, стор.	розділів	Презентація, кількість слайдів
	93	3	15
Розділ 1	Аналіз предметної області та досвіду практики		
Розділ 2	Аналіз методів виявлення прихованої інформації		
Розділ 3	Розробка комбінованого методу виявлення та вилучення стеганографічних вкладень у синтетичних зображеннях		
Висновки по роботі:	У роботі проведено аналіз предметної області стеганографії та методів стегоаналізу. Досліджено основні методи виявлення прихованої інформації у		

	цифрових зображеннях, зокрема LSB-аналіз, аналіз бітових площин та оцінку ентропії. Розроблено комбінований програмний засіб для автоматизованого виявлення та вилучення стеганографічних вкладень у синтетичних зображеннях (рендерах). Практична цінність роботи підтверджується тестуванням на реальних зображеннях форматів PNG та JPG. Результати можуть бути використані у сфері кібербезпеки та цифрової криміналістики.
Ключові слова: Keywords:	стеганографія, стегоаналіз, приховані дані, цифрові зображення, метод найменш значущого біта, LSB, виявлення прихованої інформації, вилучення даних, синтетичні зображення, кібербезпека, аналіз бітових площин, ентропія зображення steganography, steganalysis, hidden data, digital images, least significant bit, LSB, hidden information detection, data extraction, synthetic images, cybersecurity, bit-plane analysis, image entropy

Здобувач: _____ /_____ /

Керівник: _____/_____ /

АНОТАЦІЯ

Кваліфікаційна робота присвячена дослідженню методів виявлення та вилучення прихованої інформації в цифрових зображеннях. Актуальність зумовлена ризиками використання стеганографії для витоку даних та маскування шкідливого ПЗ. У роботі проаналізовано властивості цифрових фотографій і синтетичних рендерів, досліджено алгоритми стегоаналізу та розроблено програмний засіб для автоматизованого виявлення LSB-вбудовування та екстракції даних.

Ключові слова: стеганографія, стегоаналіз, цифрові зображення, прихована інформація, LSB, вилучення даних, кібербезпека.

ABSTRACT

The graduation work investigates methods for detecting and extracting hidden information in digital images. The relevance is due to the risks of using steganography for data leakage and masking malicious software. The paper analyzes the properties of digital photographs and synthetic renders, explores steganalysis algorithms, and develops a software tool for automated detection of LSB embedding and data extraction.

Keywords: steganography, steganalysis, digital images, hidden information, LSB, data extraction, cybersecurity.

ЗМІСТ

ВСТУП

Актуальність теми. Сучасний етап розвитку інформаційного суспільства характеризується стрімким збільшенням обсягів мультимедійних даних, що циркулюють у глобальних та локальних комп'ютерних мережах. Одним із найбільш динамічних інструментів приховування фактів комунікації є цифрова стеганографія, яка, на відміну від криптографії, маскує саме існування секретного повідомлення всередині безневинних об'єктів-контейнерів.

Найбільш поширеним типом таких контейнерів є цифрові зображення. Проте з розвитком технологій тривимірного моделювання та візуалізації суттєву частку графічного контенту почали складати синтетичні зображення (3D-рендери). Ці файли мають унікальні математичні та інструментальні властивості, зокрема відсутність природного фізичного шуму матриці камери та високий рівень внутрішньої кореляції пікселів.

Сьогодні стеганографічні методи використовуються не лише для захисту авторських прав за допомогою цифрових водяних знаків, але й зловмисниками — для прихованого управління шкідливим програмним забезпеченням (C2-канали) та несанкціонованого витоку корпоративних даних (DLP-загрози). Тому дослідження та вдосконалення методів виявлення (стегоаналізу), а також подальшого вилучення прихованої інформації з графічних файлів є критично важливим завданням для забезпечення кібербезпеки та проведення комп'ютерно-технічної експертизи.

Об'єкт дослідження — процеси приховування, виявлення та вилучення додаткової інформації у цифрових графічних файлах-контейнерах.

Предмет дослідження — методи стегоаналізу, математичні моделі та алгоритми детекції та екстракції даних, вбудованих у цифрові фотографії та синтетичні зображення (рендери) форматів PNG, TIFF та JPG.

Мета роботи — підвищення ефективності виявлення та вилучення прихованих повідомлень, вбудованих у цифрові зображення методом найменшого значущого біта (LSB), шляхом врахування стохастичних властивостей природних та синтетичних графічних контейнерів.

Для досягнення поставленої мети необхідно вирішити такі **завдання**:

1. Проаналізувати сучасний стан розвитку стеганографічних методів вбудовування даних та існуючих інструментів стегоаналізу.
2. Дослідити вплив способів формування графічного контенту (фізична зйомка та рендеринг) на його придатність як стеганографічного контейнера.
3. Оцінити вплив LSB-вбудовування на показники ентропії та міжбітової кореляції в різних колірних площинах зображення.
4. Розробити алгоритмічне та програмне забезпечення для автоматизованого пошуку математичних аномалій у структурі молодших бітів зображень.
5. Реалізувати функціонал для вилучення знайденої прихованої інформації та експериментально перевірити ефективність запропонованих рішень.

Методи дослідження. У роботі використано методи теорії інформації (для розрахунку ентропії Шеннона), математичної статистики (для аналізу кореляції бітових площин) та системного аналізу. Практична реалізація виконана на мові Python.

Наукова новизна отриманих результатів полягає у вдосконаленні критеріїв аналізу молодших бітових площин цифрових контейнерів шляхом диференціації оцінки ентропії для природних і синтетичних зображень, що дозволяє підвищити точність детекції стеганограм з низькою щільністю заповнення.

Практичне значення отриманих результатів. Розроблене програмне забезпечення може бути інтегроване в комплексні системи запобігання витоку даних (DLP) для моніторингу графічного трафіку, а також використане фахівцями

центрів реагування на кіберінциденти (CSIRT/SOC) та криміналістами для оперативного виявлення прихованих загрози в медіафайлах.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ДОСВІДУ ПРАКТИКИ

1.1. Опис підприємства та напрямів діяльності

1.1.1. Поняття та еволюція CGI-студій

CGI (Computer Generated Imagery) — це технологія створення цифрових зображень із використанням тривимірного моделювання, засобів візуалізації та алгоритмів рендерингу [1]. Сьогодні CGI широко застосовується в архітектурі, дизайні, рекламі, кіновиробництві та інших сферах, де виникає потреба у створенні фотореалістичного або концептуального візуального контенту. У науковій і технічній літературі CGI розглядається як окремий напрям комп'ютерної графіки, що поєднує математичні методи обробки зображень, фізично коректне моделювання освітлення та сучасні обчислювальні технології [20].

Рендер-студія, або студія комп'ютерної графіки та візуалізації, є спеціалізованим підприємством, діяльність якого пов'язана зі створенням двовимірних зображень або відеоматеріалів на основі тривимірних сцен. Основним результатом роботи таких студій є фінальні рендери, які використовуються замовниками для презентації архітектурних проєктів, рекламної продукції, промислових моделей або анімаційних рішень. Для виконання таких завдань застосовуються спеціалізовані програмні продукти, зокрема 3ds Max, Corona Renderer, V-Ray, Arnold та інші рендер-двигуни.

Історично розвиток CGI-індустрії пов'язаний із загальним прогресом комп'ютерної графіки як показано на рис. 1.1.

Рисунок 1.1 – Основні етапи розвитку CGI-технологій (розроблено автором)

Перші теоретичні та практичні напрацювання у цій сфері були закладені ще у 1970-х роках у наукових лабораторіях, а суттєвий поштовх до розвитку надало впровадження алгоритмів трасування променів[21]. У подальшому вдосконалення методів Monte Carlo Path Tracing[22], а також поява GPU-обчислень[23] дали змогу значно пришвидшити процес візуалізації та підвищити реалістичність фінального зображення. У результаті CGI-студії перетворилися з вузькоспеціалізованих технічних підрозділів на важливих учасників сучасного ринку цифрового контенту .

На сьогодні ринок CGI-послуг охоплює як невеликі студії та фахівців-фрилансерів, так і великі виробничі компанії з власною інфраструктурою, автоматизованими робочими процесами, засобами контролю якості та внутрішніми системами обміну файлами. Окреме місце займають великі міжнародні студії, які працюють із масштабними проєктами й приділяють особливу увагу питанням захисту даних. Це пояснюється тим, що результати їхньої діяльності часто мають високу комерційну цінність, а витік навіть одного файлу може спричинити значні фінансові та репутаційні втрати.

Ринок CGI-послуг має чітку сегментацію:

1. **Малі форми (Фрілансери та мікро-студії):** зазвичай працюють над локальними проектами, мають спрощені протоколи безпеки
2. **Середні студії (Production Houses):** мають штат від 20 до 100+ художників, власні відділи розробки (R&D) та QA. Саме до цього сегменту відноситься база практики, де процеси автоматизовані через власну CRM.
3. **Глобальні лідери (Pixar, Weta Digital, Industrial Light & Magic):** гіганти індустрії, що працюють над блокбастерами. Вони мають найвищий рівень захисту даних (стандарти TPN — Trusted Partner Network), оскільки витік одного кадру може коштувати мільйони доларів.

1.1.2. Типологія проектів та напрями діяльності

Сучасні CGI-студії виконують широкий спектр робіт, однак найбільш поширеним напрямом залишається створення візуального контенту для архітектурної сфери. У межах бази практики, зокрема у співпраці з партнером ArchiCGI, основна увага приділяється підготовці інтер'єрних та екстер'єрних рендерів, архітектурної анімації та продуктової візуалізації як показано на рис. 1.2. Такий контент використовується замовниками для презентації майбутніх об'єктів, маркетингових матеріалів, комерційних пропозицій та демонстрації концепцій ще до фактичної реалізації проєкту.

Рисунок 1.2 – Приклади основних типів CGI-візуалізації складено автором на основі джерела [24]

Архітектурна візуалізація передбачає створення фотореалістичних зображень будівель, приміщень і навколишнього середовища. У більшості випадків саме цей тип матеріалів формує основний документообіг і файловий трафік підприємства. Крім цього, у CGI-студіях можуть виконуватися завдання з промислової

візуалізації, коли необхідно відтворити обладнання, механізми або прототипи виробів, які ще не існують у фізичному вигляді. Окремим напрямом є рекламна та продуктова графіка, орієнтована на створення візуально привабливих зображень товарів для каталогів, веб-ресурсів або рекламних кампаній. Також суттєве місце займають анімація і VFX, де до статичних рендерів додаються рух, трекінг, композитинг та інші динамічні елементи.

Фінальні результати роботи студій зазвичай зберігаються у форматах PNG, TIFF або JPEG, а також у відеоформатах у випадку анімації [6], [7], [8]. На практиці найчастіше використовуються роздільні здатності FullHD та 4K, тоді як більші формати застосовуються рідше — переважно для великоформатного друку, преміальних презентацій або спеціальних вимог замовника. Висока роздільна здатність таких файлів означає значний обсяг піксельних даних, а отже, і потенційно більшу стеганографічну ємність контейнера [1].

Саме тому файли, які створюються у CGI-студіях, становлять інтерес не лише як елемент виробничого процесу, а й як можливий об’єкт дослідження у сфері кібербезпеки. Великий розмір зображень, реалістична структура шуму, складність текстур і природність візуального вигляду роблять такі файли зручним середовищем для прихованого вбудовування інформації [1]. З практичної точки зору це вимагає більш уважного підходу до контролю вмісту графічних файлів, що передаються всередині Типові формати виходу: PNG (ISO/IEC 15948:2004), TIFF (Adobe TIFF 6.0 Specification) або JPEG (ISO/IEC 10918-1).

Типові формати роздільності: FullHD (1920×1080), 4K (3840×2160) та рідше — 6K+

Це означає, що один 4K PNG може містити понад 8 мільйонів пікселів, що створює значний простір для потенційного стеганографічного вбудовування.

1.1.3. Значення для кібербезпеки

Діяльність CGI-студій має безпосереднє значення для кібербезпеки, оскільки в процесі роботи обробляються великі обсяги цифрових матеріалів, частина з яких містить конфіденційну або комерційно чутливу інформацію. До таких даних належать технічні завдання клієнтів, архітектурні креслення, внутрішні версії проєктів, комерційні умови співпраці та інші супровідні матеріали. Усе це передається між виконавцями, менеджерами та замовниками через CRM-системи, захищені канали зв'язку, електронну пошту або хмарні сервіси як показано на рис. 1.3.

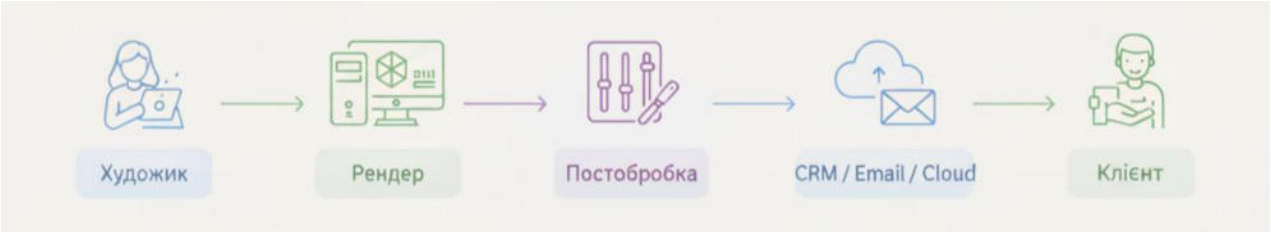


Рисунок 1.3 – Типовий процес передавання фінальних рендерів у CGI-студії (розроблено автором)

Особливість CGI-середовища полягає в тому, що передача графічних файлів є повністю легітимною та звичною частиною робочого процесу. На відміну від багатьох інших сфер, де великі файли можуть викликати додаткову увагу з боку систем моніторингу, для студії візуалізації постійний обмін зображеннями високої роздільної здатності є нормальною практикою. Саме тому графічні файли можуть бути використані не лише як результат творчої або технічної роботи, але й як потенційний прихований канал передачі даних.

Окрему увагу слід приділити структурі самих файлів. Зображення можуть містити не лише візуальну інформацію, а й метадані, службові поля, профілі кольору та інші елементи, які інколи також здатні нести додаткові відомості [8]. Крім того, висока деталізація рендерів, наявність шуму, текстур і складних градієнтів ускладнюють

виявлення змін, пов’язаних із прихованим вбудовуванням даних у піксельну структуру [1].

З позиції інформаційної безпеки це створює низку ризиків. Насамперед ідеться про можливість прихованої фільтрації даних, передачу конфіденційної інформації за межі організації, а також використання графічних файлів як носіїв додаткового навантаження, що може бути частиною шкідливих сценаріїв. Відповідно до сучасних підходів до захисту інформації, зокрема вимог стандартів серії ISO/IEC 27001, організація повинна не лише забезпечувати захист каналів передачі, а й контролювати сам зміст інформаційних об’єктів, що передаються [5].

Таким чином, CGI-рендери слід розглядати не лише як візуальний продукт, а і як потенційний об’єкт кіберзагроз. Це особливо важливо для підприємств, де великі обсяги зображень проходять через централізовані системи зберігання, обробки та передавання. Саме така специфіка предметної області обґрунтовує актуальність дослідження методів виявлення та вилучення прихованої інформації із зображень (рис.1.4).

Рисунок 1.4 – Основні ризики для кібербезпеки при обробці та передаванні CGI-рендерів (розроблено автором)

Рендер-студії є унікальним об'єктом для кібербезпеки з наступних причин:

- **Величезні обсяги даних:** через мережеві канали передаються терабайти графічної інформації. Моніторинг такого трафіку на предмет аномалій є найскладнішим завданням.
- **Конфіденційність ТЗ:** технічні завдання клієнтів часто містять комерційну таємницю (планування секретних об'єктів, дизайн не анонсованих продуктів).
- **Метадані проектів:** файли рендерів містять інформацію про структуру мережі, версії ПЗ та шляхи до файлів на сервері, що може бути використано для підготовки цілеспрямованої атаки.
- **Легітимність каналу:** оскільки передача зображень є основною діяльністю студії, передача стеганограми (прихованого повідомлення) всередині рендеру виглядає як абсолютно легітимна операція, що не викликає підозр у стандартних засобів захисту (Firewalls/IDS).

Відповідно до ISO/IEC 27001:2022 (ДСТУ ISO/IEC 27001:2023):

організація зобов’язана контролювати канали передачі інформації та запобігати витоку конфіденційних даних.

Передача здійснюється через: CRM , HTTPS або VPN

Це робить їх потенційними прихованими каналами передачі даних (covert channels).

1.1.4. Об’єкти захисту в CGI-інфраструктурі

У межах діяльності CGI-студії об’єктами захисту виступають як самі цифрові зображення, так і пов’язані з ними супровідні дані. Передусім це технічні завдання клієнтів, у яких можуть міститися детальні описи проєктів, вимоги до візуалізації, креслення, референси та інша інформація з обмеженим доступом. Не менш важливими є внутрішні версії проєктів, робочі матеріали, файли сцен, а також комерційні відомості, пов’язані з умовами виконання замовлень.

Окрему категорію становлять метадані графічних файлів і службова інформація, яка може містити технічні характеристики програмного забезпечення, параметри збереження або інші відомості, що мають значення для внутрішньої інфраструктури підприємства [6], [8]. У разі несанкціонованого доступу або прихованого вбудовування даних такі елементи можуть використовуватися як джерело додаткової інформації для потенційного порушника.

Отже, захист у CGI-інфраструктурі повинен охоплювати не лише класичні інформаційні ресурси, а й увесь життєвий цикл цифрових зображень — від моменту створення до передачі замовнику. Це включає контроль цілісності файлів, аналіз можливих ознак прихованого втручання та зниження ризиків використання графічних матеріалів як прихованих каналів передачі інформації.

Отже, до об’єктів захисту належать:

- ТЗ клієнтів
- Архітектурні креслення
- Комерційні умови
- Внутрішні версії проектів
- Метадані

1.2. Процес створення CGI-рендерів у 3ds Max

1.2.1. Основні етапи виробничого процесу

Створення фотореалістичного рендеру є багатоетапним технологічним процесом, кожен з яких впливає на фінальні статистичні характеристики зображення [17], [20]:

Рисунок 1.5 – Основні етапи створення CGI-рендеру (розроблено автором)

1. Моделювання (Modeling): Побудова тривимірної геометрії об'єктів [17], [20]. На цьому етапі формується структура сцени — кількість полігонів, складність об'єктів. Чим складніша сцена та чим вищі вимоги до фотореалістичності, тим більшим є обсяг обчислень під час рендерингу, що може впливати на характер шуму та статистичні властивості фінального зображення [20].

2. Текстурування та матеріали (Texturing & Shading): Призначення фізично коректних матеріалів (PBR — Physically Based Rendering) [17], [20]. Матеріали з високою відбивністю (метал, скло, вода) генерують складні патерни відбиттів, що можуть ускладнювати виявлення модифікацій [1].

3. Освітлення (Lighting): Налаштування джерел світла та HDRI-карт оточення [17], [20]. Освітлення є ключовим фактором, що визначає розподіл яскравості пікселів. Нерівномірне освітлення (тіні, блики) створює зони з різною щільністю шуму, що впливає на вибір оптимальних областей для вбудовування прихованих даних [20], [1].

4. Рендеринг (Rendering): Безпосередня генерація зображення. Саме на цьому етапі формується шум, що є центральним об'єктом дослідження [17], [20]. Детальний аналіз наведено в підрозділі 1.2.2.

5. Постпродакшн (Post-production): Фінальна обробка в Adobe Photoshop або After Effects: корекція кольору, ретуш, додавання ефектів [17], [20]. Цей етап є критичним з точки зору стеганографії, оскільки деякі операції (наприклад, стиснення JPEG або застосування фільтрів) можуть **знищити** раніше вбудовані дані [1], [2], [7] .

1.2.2. Порівняльний аналіз рендер-двигунів

Вибір рендер-двигуна безпосередньо визначає характер цифрового шуму у фінальному зображенні, що є ключовим параметром для оцінки стеганографічної ємності контейнера [20].

Corona Renderer:

- **Архітектура:** CPU-орієнтований, використовує алгоритм *Unbiased Path Tracing [12]*.
- **Характер шуму:** Рівномірний стохастичний шум (Monte Carlo noise), що розподілений по всьому зображенню. Такий шум є статистично передбачуваним, що полегшує його моделювання, але водночас може створювати сприятливі умови для маскування LSB-вбудовування [1].
- **Застосування на практиці:** Основний двигун для архітектурної візуалізації та статичних рендерів.

V-Ray (Hybrid CPU+GPU):

- **Архітектура:** Гібридний рендер-двигун, що підтримує як CPU, так і GPU-обчислення [13].
- **Характер шуму:** Залежить від режиму. CPU-режим генерує шум, подібний до Corona. GPU-режим (V-Ray GPU) має більш нерівномірний шум через специфіку паралельних обчислень на відеокарті [13], [23].
- **Особливість:** Широко використовується в промисловій та рекламній візуалізації завдяки гнучкості налаштувань.

Arnold:

- **Архітектура:** Промисловий стандарт для кіно та VFX (Autodesk). Використовується на великих студіях (Pixar, Weta Digital) [14].
- **Характер шуму:** Характеризується специфічним "зернистим" шумом, що нагадує плівкове зерно. Це робить його особливо складним для стеганоаналізу, оскільки природний шум Arnold важко відрізнити від артефактів вбудовування.

Chaos Vantage [25]:

- **Архітектура:** Повністю GPU-орієнтований рендер реального часу на базі NVIDIA RTX.
- **Характер шуму:** Найбільш нерівномірний та артефактний серед розглянутих двигунів. Використовується для швидких превью та анімацій.
- **Застосування на практиці:** Використовується для відео-анімацій та трекінгу, де швидкість важливіша за абсолютну якість.

1.2.3. Вплив типу рендерингу на стеганографічну ємність

Різні рендер-двигуни створюють принципово різні умови для вбудовування прихованих даних [1], [20]:

Таблиця 1.1 – Порівняльна характеристика рендер-двигунів за параметрами шуму та стеганографічної ємності

Параметр	**Corona (CPU)**	**V-Ray GPU**	**Arnold**	**Chaos Vantage**
Тип шуму	Рівномірний стохастичний	Нерівномірний	Зернистий (плівковий)	Артефактний
Ентропія зображення	Середня	Висока	Висока	Дуже висока
Стеганографічна ємність	Середня	Висока	Висока	Дуже висока
Складність виявлення LSB	Середня	Висока	Висока	Дуже висока
Стійкість до денойзингу	Низька	Середня	Середня	Висока

Ключовий висновок: GPU-орієнтований рендеринг (V-Ray GPU, Chaos Vantage) **може створювати** сприятливіші умови для маскування прихованих даних завдяки вищій варіативності шуму; однак це потребує експериментальної перевірки [1], [23]. Це означає, що зображення, отримані на GPU, можуть мати вищу маскувальну здатність. Водночас, саме ця характеристика ускладнює роботу систем стеганоаналізу, що обґрунтовує необхідність розробки спеціалізованих методів виявлення для кожного типу рендер-двигуна [1], [18].

1.2.4. Роль денойзингу у контексті стеганоаналізу

Денойзинг (шумозаглушення) є обов'язковим етапом виробничого процесу, що суттєво впливає на можливість виявлення прихованих даних [12], [13], [14]:

- **Corona High Quality Denoiser:** Інтелектуальне згладжування, що зберігає деталі текстур [12]. Може частково знищити LSB-вбудовування, але не повністю — особливо в зонах з дрібними деталями [1], [2].
- **NVIDIA AI Denoiser:** NVIDIA AI Denoiser виконує інтенсивне згладжування зображення, що може призводити до зміни молодших бітів пікселів і, відповідно, впливати на збереження прихованих даних [1], [2], [15].
- **Intel Open Image Denoise (OIDN):** Відкритий денойзер, що використовується в Corona та V-Ray [16]. Менш агресивний, ніж NVIDIA AI, що робить його менш небезпечним для збереження вбудованих даних.

Рисунок 1.6 – Вплив денойзингу на структуру шуму зображення складено автором на основі джерела [15]

Практичний висновок: Якщо зловмисник вбудовує дані *після* денойзингу (на етапі постпродакшну), ефективність маскування значно зростає, оскільки фінальне зображення вже не проходить жодної обробки перед відправкою клієнту через CRM [1], [2].

Якщо вбудовування здійснюється до денойзингу, прості LSB-методи можуть бути зруйновані [1], [2]. У такому випадку необхідно застосовувати стійкі до фільтрації алгоритми або виконувати вбудовування після завершення всіх етапів обробки.

1.3. Робота з фінальними зображеннями, анімаціями та рендерами

1.3.1. Формати збереження: PNG, JPG, TIFF

Фінальний етап виробництва передбачає експорт зображень у формати, що визначають спосіб зберігання піксельних даних та ступінь стиснення. Саме формат контейнера є критичним фактором для можливості прихованого вбудовування інформації [1], [4]. Основні характеристики форматів збереження зображень наведено в табл. 1.2 та на рис. 1.7.

Таблиця 1.2 – Порівняльна характеристика форматів PNG, JPEG і TIFF (складено автором на основі [6]–[8])

Формат	**Стандарт**	**Тип стиснення**
PNG	ISO/IEC 15948	без втрат
JPEG	ISO/IEC 10918	з втратами (DCT)
TIFF	Adobe TIFF 6.0	без втрат

PNG (Portable Network Graphics)[6]

- Стандарт: **ISO/IEC 15948:2004**
- Тип стиснення: **Lossless (без втрат)**
- Особливість: використовує алгоритм DEFLATE, який не змінює бітову структуру пікселів.
- Наслідок для стеганографії: дозволяє безпечно застосовувати LSB-вбудовування, оскільки молодші біти пікселів не змінюються після збереження [1], [6].

Наукові дослідження (Fridrich, 2009) підтверджують, що lossless-формати є основним контейнером для просторових методів стеганографії.

JPEG (JPG)[7]

- Стандарт: **ISO/IEC 10918**
- Тип стиснення: **Lossy (з втратами)**
- Особливість: використовує дискретне косинусне перетворення (DCT) та квантування.
- Наслідок: класичне LSB-вбудовування в просторовій області руйнується під час повторного збереження JPEG [1], [2], [7].

JPEG вимагає спеціальних методів вбудовування (JSteg, F5), які працюють у частотній області (DCT-коефіцієнти), а не безпосередньо з пікселями [1], [4].

TIFF (Tagged Image File Format)[8]

- Стандарт: **ISO 12639**
- Може бути як lossless, так і без стиснення
- Підтримує 8-, 16- та 32-бітну глибину кольору

У професійній CGI-практиці TIFF використовується для архівних або великоформатних (>4K) зображень.

Збільшена глибина кольору означає більшу кількість бітів на канал, що прямо збільшує потенційну стеганографічну ємність [1], [8].

Рисунок 1.7 – Порівняння форматів зображень з погляду можливості стеганографічного вбудовування (розроблено автором)

1.3.2. Вплив роздільної здатності на обсяг даних та стеганографічну ємність

У процесі створення CGI-контенту роздільна здатність (resolution) є ключовим параметром, що визначає не лише візуальну якість, а й фізичний обсяг файлу-контейнера [20]. На базі практики було проаналізовано три основні стандарти, що використовуються в індустрії: **FullHD**, **4K** та **6K+** (рис. 1.8).

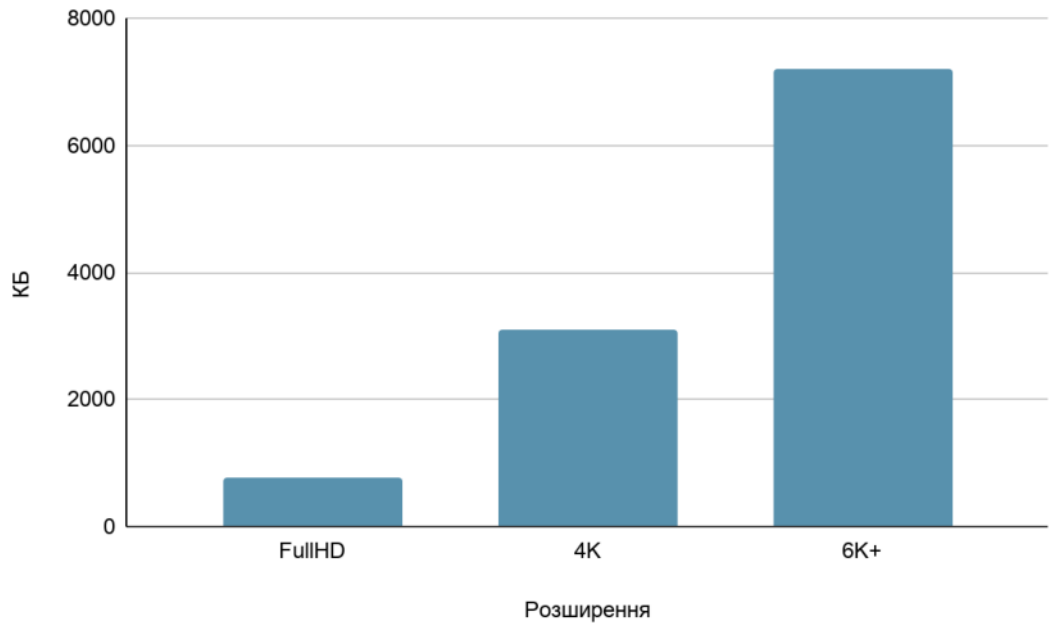


Рисунок 1.8 – Залежність стеганографічної ємності контейнера від роздільної здатності зображення (розроблено автором)

Технічні параметри об'єктів дослідження:

1. **FullHD (1920 × 1080 пікселів):**
    - Загальна кількість пікселів: ~2,07 млн.
    - При використанні 8-бітного кодування на канал (RGB), загальна кількість біт становить 49,7 Мбіт.
    - **Стеганографічний потенціал:** При заміні лише 1 найменш значущого біта (LSB) у кожному каналі, теоретична ємність складає ~776 КБ. Це дозволяє приховати текстовий документ обсягом у кілька сотень сторінок або невеликий скрипт [1].
2. **4K Ultra HD (3840 × 2160 пікселів):**
    - Загальна кількість пікселів: ~8,29 млн (у 4 рази більше за FullHD).
    - Загальна кількість біт (RGB 8-bit): ~199 Мбіт.
    - **Стеганографічний потенціал:** Теоретична ємність LSB зростає до ~3,1 МБ. Такий обсяг теоретично дозволяє приховувати файли значного розміру (.exe), зашифрований архів або інше зображення меншої роздільності.
3. **6K та вище (наприклад, 6144 × 3160):**
    - Загальна кількість пікселів: ~19,4 млн.
    - **Стеганографічний потенціал:** Ємність перевищує 7,2 МБ. Такі файли зазвичай створюються за допомогою рендер-ферм для великоформатного друку або преміальних презентацій. Через величезний обсяг надлишкових даних, виявлення прихованого повідомлення статистичними методами стає значно складнішим завданням.

Залежність між роздільністю та безпекою:

З точки зору кібербезпеки, збільшення роздільної здатності рендеру призводить до наступних наслідків:

- **Зниження відносної ентропії:** При вбудовуванні одного і того ж обсягу даних (наприклад, 100 КБ), відсоток модифікованих пікселів у 4K зображенні буде в 4 рази меншим, ніж у FullHD [1]. Це робить втручання менш помітним для статистичних детекторів (наприклад, Chi-square test) [1], [18] .
- **Збільшення часу аналізу:** Програмні засоби стеганоаналізу потребують більше обчислювальних ресурсів для обробки 4K/6K зображень, що збільшує навантаження на системи аналізу та моніторингу [1].
- **Маскування в шумі:** У рендерах високої роздільності (особливо без денойзингу) присутня велика кількість мікро-деталей та стохастичного шуму може створювати сприятливі умови **[1]**, .

Висновок для дослідження: Висока роздільна здатність сучасних CGI-рендерів, 4K широко використовується у сучасній візуалізаційній практиці, критично підвищує ризики використання таких зображень як прихованих каналів передачі даних великого обсягу [1], [3], [9].

1.3.3. Постпродакшн та його вплив на цілісність прихованих даних

Після рендерингу зображення проходить обробку [17], [20]:

- Колоркорекція (Color grading)
- Контрастування
- Композитинг
- Додавання ефектів
- Повторне збереження

Будь-яка операція, що змінює значення пікселів, впливає на молодші біти [1], [2].

Дослідження Provos & Honeyman (2003) показують, що:

- Перекодування JPEG практично повністю руйнує просторове LSB-вбудовування [1], [2], [7].
- Навіть мінімальна зміна яскравості може змінити розподіл молодших бітів [1], [2].

Практичний висновок:

Вбудовування прихованих даних має відбуватися **після завершення всіх етапів постпродакшну**, без подальшого повторного збереження або компресії [1], [2].

1.3.4. Анімація та відео як багато кадровий контейнер

Відеофайл (MP4, MOV) складається з послідовності кадрів. Кожен кадр є окремим растровим зображенням [19].

Теоретично:

Загальна ємність = Ємність одного кадру × Кількість кадрів

Наприклад, 10 секунд 4K-відео при 30 fps:

30 × 10 = 300 кадрів

Якщо кожен кадр має ємність ~3 МБ (LSB), теоретична сукупна ємність перевищує 900 МБ [1].

Однак сучасні кодеки (H.264, H.265) [19]:

- Використовують міжкадрове стиснення (I, P, B frames)
- Застосовують перетворення DCT
- Агресивно оптимізують дані

Згідно з дослідженнями у сфері відеостеганографії, просторове LSB-вбудовування нестійке до відеокомпресії, якщо воно виконується до кодування [19].

Отже:

- Вбудовування повинно враховувати структуру кодека.
- Просте редагування MP4 після експорту малоефективне.

1.3.5. Метадані як потенційний прихований канал

Формати PNG, TIFF та JPEG підтримують [6], [7], [8]:

- EXIF
- XMP
- ICC-профілі
- Текстові блоки

Згідно зі специфікаціями ISO:

- Метадані можуть містити довільні текстові поля [6], [8].
- Вони не впливають на візуальну частину зображення [6], [7], [8].

Це створює альтернативний канал приховування, однак [3]:

- Метадані легко видаляються [6], [7], [8].
- Їх простіше виявити, ніж піксельну модифікацію [1], [4].

Тому у межах даного дослідження основна увага приділяється саме **просторовим методам вбудовування у пікселі**, а метадані розглядаються як допоміжний канал.

1.4. Класифікація та порівняльний аналіз методів стеганоаналізу

Стеганоаналіз — це галузь кібербезпеки, що займається виявленням наявності прихованих повідомлень у цифрових об’єктах-контейнерах [1], [4]. Згідно з класифікацією Фрідріх (Fridrich, 2009), методи стеганоаналізу поділяються на дві великі категорії: **специфічні** (орієнтовані на конкретний алгоритм вбудовування) та **універсальні** (здатні виявляти втручання незалежно від використаного методу) [1] .

1.4.1. Візуальний стеганоаналіз та аналіз бітових площин

Візуальний аналіз є первинним етапом дослідження і базується на виявленні візуальних аномалій у структурі зображення [1].

- **Bit-plane slicing:** метод розкладання зображення на окремі бітові площини. Оскільки молодші значущі біти (LSB) у природних зображеннях зазвичай мають структуру, близьку до випадкового шуму, вбудовування структурованої інформації (наприклад, тексту або заголовків файлів) створює візуально помітні патерни [1].
- **Аналіз гістограм:** LSB-вбудовування призводить до ефекту «пар значень» (Pairs of Values), коли частоти сусідніх значень яскравості (наприклад, 254 та 255) вирівнюються. Це створює характерні аномалії на гістограмі розподілу кольорів [18], [1].

1.4.2. Статистичний стеганоаналіз першого порядку

Статистичні методи базуються на перевірці гіпотези про те, що статистичні характеристики контейнера були змінені внаслідок вбудовування даних [1], [4].

- **Тест Хі-квадрат (x^2 test):** описаний у роботі Westfeld, Pfitzmann [18]. Метод аналізує статистичну значущість відхилення розподілу пар значень пікселів від теоретично очікуваного [18]. Якщо ймовірність того, що розподіл є природним, наближається до нуля, це свідчить про наявність стеганограми [18]. Метод є високоефективним для виявлення послідовного LSB-вбудовування [18], [1].
- **Аналіз ентропії:** базується на розрахунку інформаційної ентропії Шеннона. Вбудовування зашифрованих або стиснутих даних різко підвищує ентропію молодших бітів, що дозволяє ідентифікувати зони втручання у високодеталізованих рендерах [1].

1.4.3. Структурний стеганоаналіз (RS-аналіз)

RS-аналіз (Regular/Singular analysis), розроблений групою дослідників під керівництвом Дж. Фрідріх, є одним із найбільш точних методів для виявлення LSB-заміщення у lossless-форматах (PNG, TIFF) [1].

- **Методологія:** зображення розділяється на блоки, до яких застосовується функція перевороту бітів (flipping). Блоки класифікуються як регулярні (R), сингулярні (S) або незмінні (U) [1].
- **Діагностика:** у природному зображенні кількість регулярних та сингулярних блоків приблизно однакова ($R_m \approx R_{-m}$ та $S_m \approx S_{-m}$). Вбудовування даних порушує цю симетрію, що дозволяє не лише виявити факт наявності повідомлення, а й з високою точністю оцінити його відносну довжину (стеганографічне навантаження) [1].

1.4.4. Сигнатурний та специфічний стеганоаналіз

Цей підхід використовується для виявлення слідів роботи конкретних програмних засобів (наприклад, *Steghide*, *OpenStego*, *OutGuess*) [1], [4].

- **Аналіз метаданих:** перевірка цілісності заголовків файлів, пошук нетипових полів або аномальних значень у блоках EXIF/XMP [6], [7], [8], [1].
- **Пошук маркерів:** деякі стеганографічні інструменти залишають специфічні сигнатури в кінці файлу (EOF) або у службових структурах формату, може полегшувати ідентифікацію.

1.4.5. Сучасні методи на основі машинного навчання

Перспективним напрямом сучасного стегоаналізу є застосування методів машинного навчання та глибокого навчання. Зокрема, підхід **Spatial Rich Models (SRM)** базується на побудові багатовимірних моделей шумових залишків зображення та використовується для виявлення мікровідхилень, спричинених прихованим вбудовуванням даних [29].

Окремий клас сучасних методів становлять **згорткові нейронні мережі (CNN)**, які дають змогу автоматично виділяти ознаки втручання без явного ручного конструювання всіх характеристик контейнера [30].

У подальших дослідженнях також розглядалося поєднання SRM-ознак із глибоким навчанням, що підтверджує перспективність комбінованих підходів у задачах стегоаналізу [31], [32].

1.5. Потенційні ризики прихованого вбудовування інформації в зображення

1.5.1. Вбудовування під час рендерингу (рендер-ферма)

Технічна можливість

Рендер-ферма — це централізована обчислювальна система, яка генерує фінальні зображення [17], [20].
Адміністратори мають доступ до [17]:

- вихідних сцен
- проміжних рендерів
- фінальних файлів
- серверної інфраструктури

Згідно з моделлю insider threat (CERT Insider Threat Center), особа з адміністративним доступом є одним із найбільш ризикових типів порушника [11].

Сценарій ризику

Зловмисник, який має доступ до вузлів рендер-ферми, може [1], [6], [8]:

- автоматично модифікувати фінальні PNG/TIFF-файли;
- додати приховане повідомлення в LSB перед передачею у CRM [1];
- використовувати скрипти пост-обробки для вбудовування даних.

Приклад загрози

У наукових роботах описано можливість автоматичного пакетного вбудовування даних у великі масиви зображень [1].
У контексті CGI це може призвести до:

- витоку конфіденційного ТЗ;
- передачі структури внутрішньої мережі;
- прихованої ексфільтрації комерційних даних.

1.5.2. Вбудовування після рендерингу, перед постпродакшном

На цьому етапі файл зазвичай:

- зберігається у форматі PNG або TIFF (lossless) [6], [8];
- ще не проходить колоркорекцію [17], [20];
- зберігає первинну бітову структуру.

Чому це критичний момент

Lossless-формати не змінюють молодші біти пікселів [6], [8], [1].
Згідно з теорією LSB, це ідеальний момент для [1],:

- просторового вбудовування;
- високої стегоємності;
- мінімальної статистичної деформації.

Потенційний сценарій

- Вбудовування прихованого payload перед передачею в композитинг.
- Додавання закодованих інструкцій, що можуть бути зчитані спеціалізованим ПЗ.

Зображення можуть використовуватися як носії прихованих даних, зокрема конфігураційних параметрів або команд, що відповідає технікам прихованої передачі даних, описаним у наукових роботах і фреймворках кібербезпеки [1], [2], [9].

1.5.3. Вбудовування після постпродакшну (перед передачею клієнту)

Це найбільш ризикована точка. Файл:

- вже готовий до передачі;
- не підлягає подальшій обробці [17], [20];
- має фінальну бітову структуру.

Чому ризик максимальний

Будь-яке LSB-вбудовування на цьому етапі [1], [2]:

- не буде знищене подальшою обробкою;
- не зміниться до моменту завантаження клієнтом;
- може залишатися невиявленим тривалий час [1], [4].

Приклад загрози

- Витік інформації про вартість проекту;
- Передача контактів клієнтів;
- Передача комерційних умов.

Такі сценарії відповідають техніці **Data Exfiltration over Unencrypted/Obfuscated Channel** (MITRE ATT&CK) [9].

1.5.4. Вбудовування під час передачі через CRM або мережу

Теоретично можливий сценарій:

- перехоплення файлу у мережі;
- модифікація;

- повторне завантаження.

Однак при використанні HTTPS/TLS (що є стандартом для сучасних CRM) ризик модифікації трафіку значно знижується.

Проте у випадку компрометації сервера можливе [11]:

- автоматичне інфікування файлів;
- вставка прихованого payload перед видачею клієнту.

Подібні механізми описані в дослідженнях, де зображення використовувались для передачі C2-команд [9].

1.5.5. Ризики на стороні клієнта

Клієнт отримує файл, який може містити [1], [2], [9]:

- приховані дані;
- зашифровані команди;
- конфіденційну інформацію третьої сторони.

У відкритих джерелах і в аналітичних матеріалах з кібербезпеки описуються сценарії використання зображень як носіїв прихованих даних [9]:

- передачі C2-команд;
- прихованого завантаження конфігурацій шкідливого ПЗ;
- обходу IDS/IPS через стеганографію [9].

1.6. Класифікація ризиків

1. Витік конфіденційної інформації

Потенційний витік стосується об’єктів захисту, визначених у п. 1.1.4.

2. Розповсюдження шкідливого ПЗ

Зображення можуть містити:

- закодований shellcode;
- конфігурації malware;
- URL для завантаження додаткових компонентів.

Згідно з MITRE ATT&CK, це техніка маскування C2-трафіку [9].

3. Формування прихованих C2-каналів

C2 (Command & Control) — канал керування скомпрометованими системами.

Деякі APT-групи використовували:

- соціальні мережі;
- зображення PNG/JPEG;
- коментарі до зображень як канал передачі команд.

4. Підміна авторства та цифрове маркування [4]

Стеганографія може використовуватись для:

- прихованого цифрового підпису;
- маркування авторства;
- приховування ідентифікаційних даних.

Це вже межує з цифровим водяним знаком (digital watermarking), що є окремим напрямом досліджень [4].

1.6.1. Модель загроз за методологією STRIDE

Для систематизації ризиків використано методологію **STRIDE**, запропоновану Microsoft для моделювання загроз інформаційним системам (таблиця 1.3) [10].

STRIDE охоплює шість типів загроз [10]:

- **S** – Spoofing (Підміна особи)
- **T** – Tampering (Модифікація даних)

- **R** – Repudiation (Заперечення дій)
- **I** – Information Disclosure (Розголошення інформації)
- **D** – Denial of Service (Відмова в обслуговуванні)
- **E** – Elevation of Privilege (Підвищення привілеїв)

Об’єкт моделювання

Система обробки та передачі CGI-рендерів:

- Робочі станції художників
- Рендер-ферма
- CRM Archivizer
- Канал передачі клієнту

Таблиця 1.3 – Модель загроз STRIDE для CGI-інфраструктури

Категорія	**Опис загрози**	**Потенційний сценарій**	**Наслідок**
Spoofing	Підміна облікового запису	Зловмисник отримує доступ до CRM під виглядом художника	Неавторизована модифікація рендерів
Tampering	Модифікація файлу	Вбудовування LSB-даних у PNG перед передачею	Прихований витік інформації
Repudiation	Заперечення факту змін	Відсутність логування модифікацій файлу	Неможливість встановити джерело інциденту

Information Disclosure	Витік конфіденційної інформації	Передача ТЗ або комерційних даних через стеганоканал	Порушення конфіденційності
Denial of Service	Перевантаження системи аналізу	Масове завантаження 6K-файлів	Зупинка сервісу перевірки
Elevation of Privilege	Підвищення доступу	Отримання контролю над рендер-фермою	Повний контроль над вихідними файлами

Аналіз за моделлю CIA

Додатково ризики оцінено відповідно до класичної моделі інформаційної безпеки [5]:

Confidentiality (Конфіденційність) [1], [5], [9]

- Стеганографія дозволяє приховано передавати комерційну інформацію.
- Основна загроза для CGI-студій.

Integrity (Цілісність) [1], [5]

- Модифікація пікселів порушує цілісність цифрового активу.
- Вбудовані дані змінюють контрольні хеші.

Availability (Доступність)

- Потенційний DoS через перевантаження великими файлами.

1.7. Нормативна та методологічна база дослідження

Дослідження базується на міжнародних стандартах, наукових працях та кібербезпекових фреймворках [1], [4], [5]–[10].

1.7.1. Міжнародні стандарти

ISO/IEC 27001:2022: Система управління інформаційною безпекою (ISMS). Визначає вимоги до захисту конфіденційних даних [5].

Стандарт ISO/IEC 27002 [27] містить практичні рекомендації та настанови щодо впровадження заходів контролю інформаційної безпеки, що дозволяє деталізувати стратегію захисту CGI-інфраструктури, зокрема в частині управління доступом та моніторингу подій безпеки.

ISO/IEC 15948:2004: Стандарт формату PNG [6].

ISO/IEC 10918: Стандарт JPEG [7].

ISO 12639: Специфікація формату TIFF, що використовується для професійного рендерингу та обміну даними, регулюється міжнародним стандартом ISO 12639 [28] (або його базовою специфікацією TIFF Revision 6.0 [8]), що забезпечує стабільність структури файлів при передачі між різними програмними комплексами.

1.7.2. Кібербезпекові методології

MITRE ATT&CK [9]

Техніка:

T1001.002 — Data Obfuscation: Steganography - Описує використання зображень для прихованої передачі команд [9].

STRIDE (Microsoft) - Метод моделювання загроз, використаний у роботі [10].

CIA Triad - Базова модель оцінки впливу на безпеку [5].

1.7.3 Використання стеганографії для прихованої доставки шкідливого навантаження

Стеганографія у цифрових зображеннях широко застосовується не лише для легітимного приховування інформації, але й у контексті кіберзагроз. Водночас принципово важливо розмежовувати два різні аспекти: використання зображення як контейнера прихованих даних та безпосередній механізм виконання шкідливого коду на стороні користувача [1], [2], [9].

З технічної точки зору зображення, що містить приховані в молодших бітах (LSB) дані або модифіковані коефіцієнти дискретного косинусного перетворення (DCT), не є самостійним механізмом виконання коду. Воно виступає виключно як контейнер для зберігання або передачі інформації [1], [2]. Реалізація виконання шкідливого програмного забезпечення можлива лише за наявності додаткового програмного компонента або вразливості, що забезпечує інтерпретацію або запуск прихованого навантаження.

У якості прихованого навантаження (payload) у зображення можуть вбудовуватися різні типи даних, зокрема:

- конфігураційні параметри шкідливого програмного забезпечення;
- адреси серверів керування (Command and Control, C2);
- криптографічні ключі;
- фрагменти машинного коду (shellcode);
- скрипти або бінарні модулі;
- зашифровані або стиснуті блоки даних.

Зазначене навантаження рідко представлено у вигляді готового виконуваного файлу. Частіше воно зберігається у формі зашифрованих блоків, стиснених бінарних об’єктів (blob) або текстових представлень (наприклад, Base64), що дозволяє ускладнити їх виявлення засобами сигнатурного аналізу.

Процес використання стеганографії в атаках можна подати у вигляді послідовного ланцюга. На першому етапі зловмисник формує корисне навантаження та вбудовує його у зображення за допомогою методів LSB або

модифікації DCT-коефіцієнтів [1], [2], [9]. Далі це зображення розміщується у мережі — на веб-ресурсах, у системах доставки контенту (CDN), хмарних сховищах або передається через електронну пошту чи месенджери. У мережевому трафіку така передача виглядає як звичайне завантаження графічного файлу, що ускладнює її виявлення [9].

На стороні користувача можливі декілька сценаріїв обробки такого зображення.

У першому сценарії відбувається звичайний перегляд зображення. Браузер або програмний переглядач використовує бібліотеки декодування (наприклад, libpng, Pillow, OpenCV), які виконують перевірку структури файлу, розпаковують стиснені дані та відновлюють матрицю пікселів. У результаті формується масив значень кольорів, де кожен піксель описується числовими значеннями каналів (наприклад, RGB). Важливо підкреслити, що ці значення інтерпретуються виключно як дані для візуалізації, а не як інструкції для виконання. Таким чином, саме по собі відкриття зображення не призводить до запуску шкідливого коду.

У другому, більш типовому сценарії, на системі користувача вже присутній активний шкідливий компонент, наприклад троян, dropper, шкідливий документ із макросом, браузерне розширення або інший програмний агент [9]. Такий компонент виконує завантаження зображення та здійснює його аналіз не як візуального об’єкта, а як набору байтів або пікселів. Далі відбувається процес стего-декодування: з молодших бітів пікселів формується бітова послідовність, яка групується у байти, після чого відновлюється приховане навантаження [1], [2]. Отримані дані можуть бути розшифровані, розпаковані або декодовані, після чого використовуються для подальших дій, таких як встановлення з’єднання з C2-сервером, завантаження додаткових модулів або запуск другого етапу атаки [9]. Саме на цьому етапі відбувається фактична активація шкідливого програмного забезпечення, яка може включати створення процесів, виконання скриптів або інжекцію коду у пам’ять.

У третьому сценарії виконання шкідливого коду може бути пов’язане не зі стеганографією як такою, а з експлуатацією вразливостей у програмному забезпеченні для обробки зображень. До таких вразливостей належать переповнення буфера (buffer overflow), переповнення цілих чисел (integer overflow), вихід за межі масиву (out-of-bounds) або використання звільненої пам’яті (use-after-free). Спеціально сформоване зображення може викликати помилки у процесі декодування, що призводить до пошкодження пам’яті та потенційного перехоплення керування виконанням. У цьому випадку запуск коду є наслідком експлуатації вразливості, а не факту наявності прихованих бітів у зображенні.

Таким чином, стеганографія у шкідливих кампаніях використовується переважно як механізм прихованої передачі або зберігання даних. Найбільш поширеними сценаріями її застосування є приховування конфігураційних параметрів, передача команд управління, доставка фрагментів другого етапу атаки, а також маскування мережевої активності під звичайний мультимедійний трафік.

Отже, принципово важливо підкреслити, що вбудовування даних у LSB-біти або інші елементи зображення не призводить до автоматичного виконання коду. Активація шкідливого навантаження можлива лише за наявності додаткового механізму виконання — програмного компонента або вразливості, що забезпечує інтерпретацію прихованих даних.

1.8 Висновки до розділу 1

У першому розділі було проведено аналіз предметної області, пов’язаної з використанням цифрових зображень у діяльності CGI-студій, а також розглянуто сучасні підходи до стеганографічного вбудовування даних і їх виявлення.

У результаті проведеного аналізу встановлено, що CGI-студії є специфічним об’єктом дослідження з позиції кібербезпеки, оскільки в межах їхньої діяльності створюються, обробляються та передаються великі обсяги графічних файлів високої роздільної здатності. Такі файли є легітимною частиною виробничого

процесу, що ускладнює виявлення можливого використання їх як прихованих каналів передачі інформації.

Досліджено основні етапи створення CGI-рендерів у середовищі 3ds Max, а також встановлено, що характеристики фінального зображення суттєво залежать від типу рендер-двигуна, параметрів освітлення, денойзингу та постобробки. Це безпосередньо впливає на статистичні властивості контейнера і, відповідно, на можливість прихованого вбудовування інформації та складність її подальшого виявлення.

Проаналізовано формати збереження зображень PNG, JPEG і TIFF та показано, що найбільш придатними для просторового LSB-вбудовування є формати без втрат, зокрема PNG і TIFF, оскільки вони зберігають бітову структуру пікселів. Водночас JPEG через використання стиснення з втратами є менш придатним для просторових методів вбудовування і потребує застосування інших підходів, орієнтованих на частотну область.

Встановлено, що роздільна здатність зображення істотно впливає на потенційну стеганографічну ємність контейнера. Зі збільшенням кількості пікселів зростає обсяг даних, які теоретично можуть бути приховані у файлі, а також ускладнюється виявлення незначних статистичних відхилень, особливо у високодеталізованих CGI-рендерах.

У розділі також виконано класифікацію та порівняльний аналіз методів стегоаналізу. Показано, що для задач виявлення прихованих повідомлень у графічних файлах доцільно використовувати поєднання візуального аналізу, аналізу бітових площин, статистичних методів першого порядку, RS-аналізу, а також сучасних підходів на основі машинного навчання. Це створює теоретичну основу для вибору практичних методів дослідження у наступних розділах роботи.

Окрему увагу приділено ризикам прихованого вбудовування інформації в зображення на різних етапах виробничого циклу — під час рендерингу,

постобробки, передавання через CRM або на стороні клієнта. Встановлено, що найбільш критичною є стадія після завершення постпродакшну, коли файл має остаточну структуру і не піддається подальшій обробці, а отже приховані дані можуть зберігатися без спотворень.

На основі моделей STRIDE та CIA доведено, що стеганографічне використання зображень у CGI-інфраструктурі створює реальні загрози конфіденційності, цілісності та частково доступності інформаційних ресурсів. При цьому зображення з прихованим навантаженням слід розглядати передусім як контейнер або канал прихованої передачі даних, а не як самостійний механізм виконання шкідливого коду.

Таким чином, перший розділ підтверджує актуальність дослідження методів виявлення та вилучення прихованої інформації із цифрових зображень та обґрунтовує необхідність розроблення практичного підходу до аналізу графічних контейнерів, що й становить зміст подальших розділів роботи.

РОЗДІЛ 2. АНАЛІЗ МЕТОДІВ ВИЯВЛЕННЯ ПРИХОВАНОЇ ІНФОРМАЦІЇ

2.1. Загальна класифікація методів стегоаналізу

Стегоаналіз як наукова дисципліна виник як відповідь на розвиток стеганографії — мистецтва приховування самого факту передачі інформації [1], [2], [33]. На відміну від криптоаналізу, де задача полягає у розшифруванні відомого зашифрованого повідомлення, стегоаналіз вирішує принципово іншу задачу: спочатку необхідно довести, що приховане повідомлення взагалі існує, і лише потім — спробувати його вилучити [1], [2], [33].

У науковій літературі методи стегоаналізу класифікують за кількома ознаками.

2.1.1. Класифікація за рівнем знань аналітика

Сліпий стегоаналіз (Blind steganalysis) — аналітик не знає ні алгоритму вбудовування, ні ключа, ні навіть факту наявності повідомлення. [1], [33].

Напівсліпий стегоаналіз (Semi-blind steganalysis) — аналітику відомий алгоритм вбудовування, але невідомий ключ[1], [33].

Цільовий стегоаналіз (Targeted steganalysis) — аналітик знає і алгоритм, і має підозру щодо конкретного інструменту [1], [33]. Наприклад, якщо відомо, що зловмисник використовував OpenStego або Steghide, можна шукати характерні сигнатури цих програм.

2.1.2. Класифікація за методом аналізу

За технічним підходом методи стегоаналізу доцільно поділяти на три основні групи: візуальні, статистичні та сигнатурні (табл. 2.1) [1], [18], [35].

Таблиця 2.1 — Загальна класифікація методів стегоаналізу за типом доступних даних та методом виявлення (складено автором на основі [1], [18], [35])

Група	Основа методу	Приклади
Візуальні	Аналіз зображення людиною або через фільтри	Bit-plane slicing, Enhanced LSB
Статистичні	Математичний аналіз розподілу пікселів	Chi-square, RS-аналіз, ентропія
Сигнатурні	Пошук характерних ознак конкретних програм	Аналіз заголовків, структури файлу

2.1.3. Формальна постановка задачі стегоаналізу

З математичної точки зору задача стегоаналізу формулюється як задача бінарної класифікації. Нехай маємо зображення x, яке може належати до одного з двох класів [1], [29], [30], [36]:

H_0 — нульова гіпотеза: зображення є чистим, без прихованих даних

H_1 — альтернативна гіпотеза: зображення містить приховане повідомлення

Аналітик будує функцію вирішення D(x), яка на основі набору ознак $f(x) = \{f_1, f_2, ..., f_n\}$ приймає рішення:

$D(x) = H_1$, якщо $f(x) > \theta$

$D(x) = H_0$, якщо $f(x) \leq \theta$

де θ — порогове значення, яке визначається емпірично або на основі навчальної вибірки.

Якість стегоаналізатора оцінюється двома показниками:

P_FA (probability of false alarm) — ймовірність хибної тривоги, тобто коли чисте зображення помилково класифікується як стего [1], [29], [36].

P_MD (probability of missed detection) — ймовірність пропуску, тобто коли стего-зображення не виявляється [1], [29], [36].

Ідеальний детектор мінімізує обидва показники одночасно, але на практиці між ними існує компроміс, який описується ROC-кривою (Receiver Operating Characteristic) [29], [30], [36].

2.2. Візуальні методи стегоаналізу

2.2.1. Структура цифрового зображення як об'єкт аналізу

Перш ніж розглядати методи виявлення, необхідно чітко розуміти, з чим саме ми працюємо. Цифрове растрове зображення — це двовимірна матриця пікселів розміром **M** × **N**, де кожен піксель у стандартній RGB-моделі описується трьома компонентами [20], [34] (рис. 2.1):

$$P(i,j) = [R(i,j), G(i,j), B(i,j)] \quad (2.2.1)$$

де

P(i,j) — значення пікселя з координатами *(i, j)*;
R(i,j) — інтенсивність червоного (Red) каналу пікселя *(i, j)*;
G(i,j) — інтенсивність зеленого (Green) каналу пікселя *(i, j)*;
B(i,j) — інтенсивність синього (Blue) каналу пікселя *(i, j)*;
i, j — координати пікселя в зображенні (рядок і стовпець відповідно).

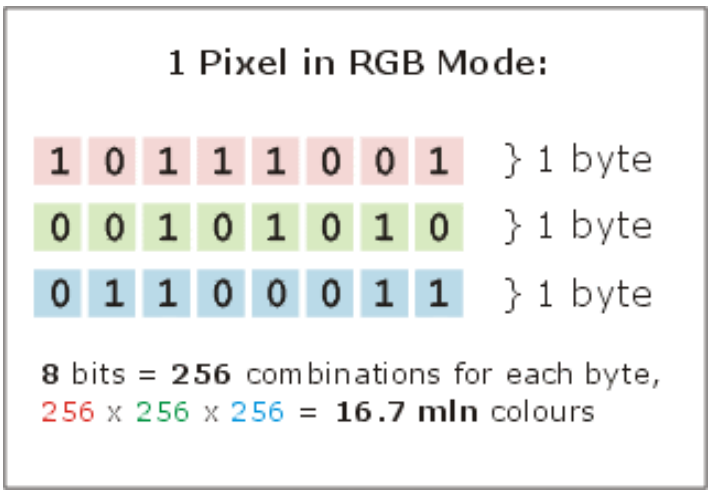


Рисунок 2.1 — Модель представлення пікселя у 24-бітному RGB-зображенні
(джерело: адаптовано автором на основі [20], [34])

де кожна компонента приймає значення від 0 до 255 (для 8-бітного зображення) [20], [34].

У бінарному представленні кожне значення каналу записується як 8-бітне число [20], [34]:

$$R(i,j) = b_7 \cdot 2^7 + b_6 \cdot 2^6 + b_5 \cdot 2^5 + b_4 \cdot 2^4 + b_3 \cdot 2^3 + b_2 \cdot 2^2 + b_1 \cdot 2^1 + b_0 \cdot 2^0 \quad (2.2.1\ (2))$$

де $\mathbf{b_7}$ — найстарший біт (MSB), $\mathbf{b_0}$ — наймолодший біт (LSB), $2^7 ... 2^0$ — вагові коефіцієнти відповідних бітів у двійковому представленні.

Саме ця структура є ключовою для розуміння більшості методів стеганографії та стегоаналізу [1], [35] (рис.2.2).

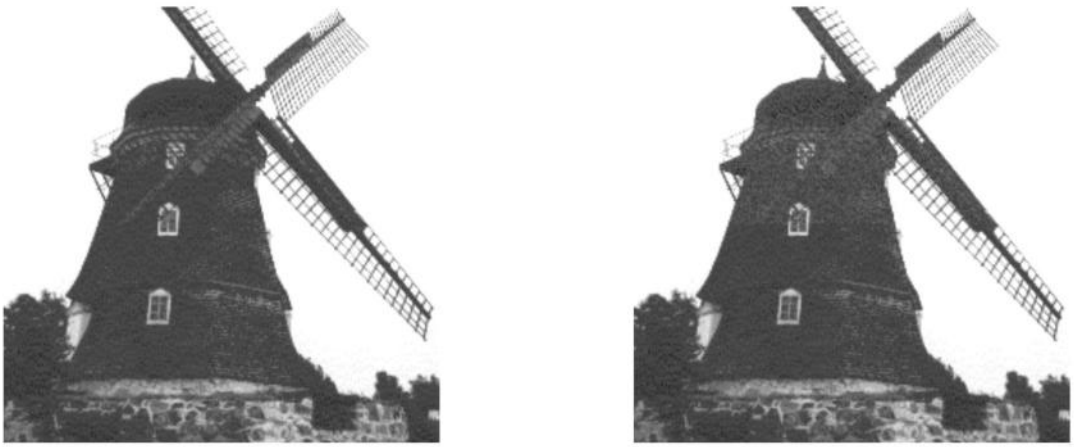


Fig. 4. Windmill as carrier medium (l.), and steganogram (r.)

Рисунок 2.2 — Приклад контейнерного зображення (ліворуч) та стегозображення (праворуч) [18]

2.2.2. Метод аналізу бітових площин (Bit-plane slicing)

Метод розкладання на бітові площини дозволяє "розібрати" зображення на 8 окремих шарів, кожен з яких відповідає одному біту кожного пікселя [34] (рис.2.3).

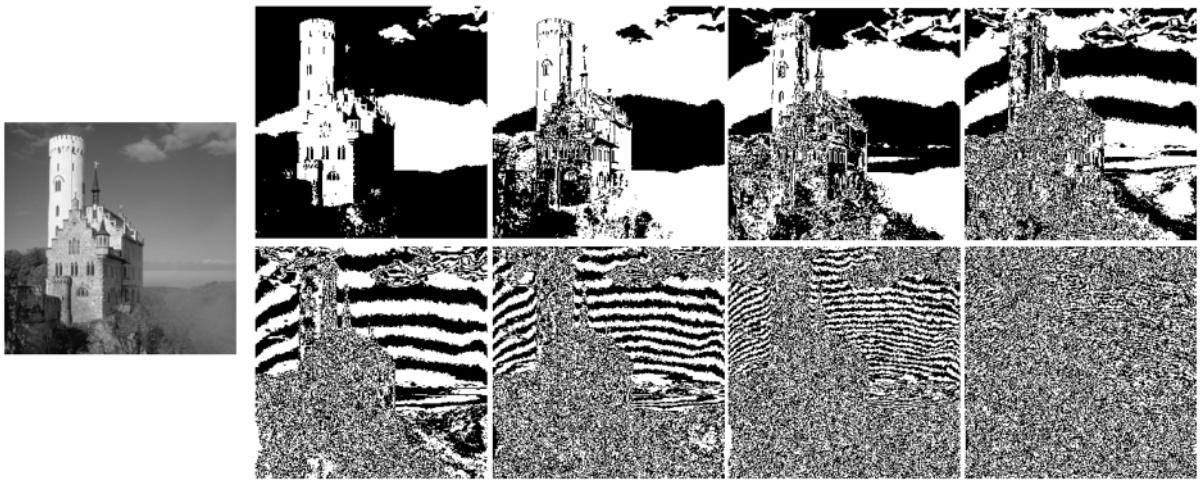


Рисунок 2.3 — Приклад декомпозиції цифрового зображення на вісім бітових площин *[34]*

Алгоритм отримання k-ї бітової площини:

Для отримання k-ї бітової площини (де k = 0..7) застосовується побітова операція [34]:

$$B_k(i,j) = \lfloor P(i,j) / 2^k \rfloor \bmod 2 \qquad (2.2.2)$$

де

$B_k(i,j)$ — значення k-ої бітової площини пікселя *(i, j)*;

P(i,j) — значення яскравості або компоненти пікселя *(i, j)*;

k — номер бітової площини *(k = 0…7 для 8-бітного зображення)*;

$\lfloor x \rfloor$ — операція взяття цілої частини числа (floor);

mod 2 — операція остачі від ділення на 2 (виділення біта);

2^k — зсув значення на k бітів вправо (у двійковому представленні).

Або еквівалентно через побітове AND [34]:

$$B_k(i,j) = (P(i,j) \text{ AND } 2^k) >> k \qquad (2.2.2\ (2))$$

AND — побітова операція логічного «І»;

>> k — побітовий зсув вправо на k позицій;

$\mathbf{2^k}$ — маска для виділення k-го біта;
інші позначення — аналогічні формулі (2.2.2).

Що ми бачимо на кожній площині:

- **Площини 7–5 (старші біти):** містять основну візуальну інформацію. Контури об'єктів, великі кольорові блоки. Зміна цих бітів одразу помітна оку [20], [34].
- **Площини 4–2 (середні біти):** деталі зображення, текстури, тіні [20], [34].
- **Площини 1–0 (молодші біти):** У молодших бітових площинах немодифікованих зображень часто спостерігається структура, близька до шумоподібної, без виражених регулярних патернів [1], [35].

Критерій виявлення стеганографії:

Якщо на площині LSB (k=0) замість рівномірного шуму спостерігаються :

- регулярні патерни або текстури
- чіткі контури об'єктів
- блокова структура

— це є ознакою вбудовування даних [1], [18], [35].

Формально це можна перевірити через **автокореляційну функцію** бітової площини [34]:

$$A(\Delta i, \Delta j) = \Sigma_i \Sigma_j B_k(i,j) \cdot B_k(i+\Delta i, j+\Delta j) \qquad (2.2.2\ (3))$$

$A(\Delta i, \Delta j)$ — автокореляційна функція для бітової площини;
$B_k(i,j)$ — значення k-го біта у пікселі *(i, j)*;
Δi, Δj — зсув по вертикалі та горизонталі відповідно;
Σ_i Σ_j — подвійна сума за всіма пікселями зображення;
$(i+\Delta i, j+\Delta j)$ — координати сусіднього пікселя зі зсувом;
$\cdot$ — операція множення.

У чистому зображенні $A(\Delta i, \Delta j) \approx 0$ для будь-яких ненульових зсувів. Підвищена автокореляція LSB-площини або наявність виражених регулярних структур може свідчити про можливе вбудовування даних [1], [35].

2.2.3. Обмеження візуальних методів

Незважаючи на наочність, візуальні методи мають суттєві обмеження [1], [35]:

1. **Суб'єктивність** — результат залежить від досвіду аналітика
2. **Неефективність проти адаптивних методів** — обмежена ефективність проти сучасних адаптивних методів вбудовування, які мінімізують візуально помітні зміни контейнера [29]–[31].
3. **Неможливість автоматизації** — обмежена придатність до автоматизації без використання додаткових математичних критеріїв та формалізованих ознак [35].

Саме ці обмеження і зумовлюють необхідність переходу до статистичних методів, які розглядаються у наступному підрозділі [1], [35].

2.3. Статистичні методи стегоаналізу

2.3.1. Математичне обґрунтування статистичного виявлення прихованих даних

Статистичний стегоаналіз базується на теоретичному положенні про те, що будь-який процес формування цифрового зображення (чи то фіксація світла матрицею камери, чи то математичний прорахунок сцени у системах 3D-рендерингу) підпорядковується певним статистичним закономірностям [35], [36]. Ці закономірності визначають розподіл значень інтенсивності пікселів та кореляційні зв'язки між ними.

Цифрове зображення можна представити як сукупність значень випадкової величини. Для не модифікованого зображення гістограму розподілу рівнів яскравості можна розглядати як оцінку щільності розподілу ймовірностей pi [34], [36]. У природних зображеннях цей розподіл є неперервним і плавно змінюється, оскільки сусідні кольори та відтінки зазвичай мають близькі значення [34], [35].

Впровадження сторонньої інформації методом найменш значущого біта (LSB) фактично є процесом додавання до сигналу зображення низькоамплітудного шуму [1], [35]. Проте, оскільки повідомлення перед вбудовуванням зазвичай піддається стисненню або шифруванню для підвищення безпеки, воно набуває властивостей, близьких до «білого шуму» з рівномірним розподілом значень 0 та 1.

Формально, процес LSB-заміни можна описати наступним чином. Нехай h2i та h2i+1 позначають кількість пікселів у зображенні з відповідними значеннями яскравості. Ці два значення утворюють так звану пару значень (PoVs — Pairs of Values), які відрізняються лише в останньому біті. При вбудовуванні випадкового біта m∈{0,1} у піксель зі значенням p, нове значення p′ визначається за правилом [1], [18]:

$$p' = 2 \cdot \lfloor p/2 \rfloor + m \qquad (2.3.1)$$

де

p — початкове значення пікселя;

p' — модифіковане значення пікселя після вбудовування;

m — вбудовуваний біт повідомлення $(m \in \{0,1\})$;

$\lfloor p/2 \rfloor$ — ціла частина від ділення на 2 (видалення молодшого біта);

$2 \cdot \lfloor p/2 \rfloor$ — відновлення значення без молодшого біта;

додавання **m** — запис нового біта в молодший розряд (LSB).

Ця операція призводить до того, що значення 2i може перейти у 2i+1, а значення 2i+1 — у 2i. Якщо ми вбудовуємо повідомлення з однаковою ймовірністю появи нулів та одиниць P(m=0) = P(m=1) = 0.5, то після обробки великої кількості пікселів їхні частоти у кожній парі будуть прагнути до вирівнювання:

$$h'_{2i} \approx h'_{2i+1} \approx \frac{h_{2i} + h_{2i+1}}{2} \qquad (2.3.1\ (2))$$

де,

h2i, h2i+1— значення яскравості (або компонент пікселів) у парі сусідніх елементів;
h’2i, h’2i+1— модифіковані значення після обробки (вбудовування або вирівнювання);
i — індекс пари пікселів;
(h2i, h2i+1**) / 2** — середнє арифметичне значень пари;
≈ — наближена рівність (значення вирівнюються, але можуть трохи відрізнятися через обмеження цілочисельних операцій).

Саме це штучне вирівнювання природного розподілу є основним статистичним артефактом. На ньому базується більшість атак: алгоритм аналізує гістограму і шукає пари значень, частоти яких є аномально близькими [1], [18]. Чим більший обсяг вбудованих даних, тим сильніше проявляється цей ефект, що дозволяє не лише виявити факт стеганографії, а й оцінити довжину повідомлення.

У контексті синтетичних зображень це створює окрему проблему інтерпретації результатів, оскільки шум, сформований алгоритмами рендерингу, може впливати на статистичні ознаки контейнера. У межах даної роботи це розглядається як окрема практична проблема, що потребує додаткового калібрування методу.

2.3.2. Теоретичні основи та алгоритм методу χ^2 (Хі-квадрат)

Метод χ^2-аналізу є одним з найбільш фундаментальних інструментів у стегоаналізі, призначеним для виявлення вбудовування даних методом простої заміни найменш значущого біта (LSB replacement) [1], [18]. Його наукова цінність полягає у використанні класичного статистичного критерію згоди Пірсона для перевірки гіпотези про штучне втручання в структуру контейнера [1], [18].

Логічне обґрунтування методу

Як було продемонстровано у попередньому підрозділі, LSB-стеганографія оперує парами значень (Pairs of Values, PoVs) [1], [18]. Якщо ми розглянемо гістограму яскравості зображення, то кожна пара сусідніх стовпчиків 2i, 2i+1 при вбудовуванні випадкового повідомлення прагне до вирівнювання своєї висоти [1], [18]. У природному звуці або зображенні така симетрія зустрічається вкрай рідко. Отже, статистична задача зводиться до порівняння фактичного розподілу частот у парах PoVs із теоретично очікуваним розподілом, який мав би виникнути за умови повної заміни LSB.

Математичний алгоритм реалізації

Процес аналізу за методом χ^2 складається з наступних етапів [1], [18] (рис.2.4):

1. **Формування вибірки:** Зображення сканується, і для кожного рівня яскравості (від 0 до 255) підраховується кількість пікселів hk. Далі формуються 128 пар частот h2i, h2i+1.
2. **Розрахунок очікуваного значення:** Для кожної пари обчислюється середнє арифметичне, яке представляє собою "ідеально вирівняний" стан після вбудовування:

Рисунок 2.4 — Ефект вирівнювання частот пар значень (PoVs) при LSB-вбудовуванні (джерело: побудовано автором на основі [1], [18])

$$n_i^* = \frac{h_{2i}+h_{2i+1}}{2} \quad (2.3.2)$$

де

n*_i — оцінене (усереднене) значення пари пікселів;

h2i**,** h2i+1 — значення пікселів у парі;

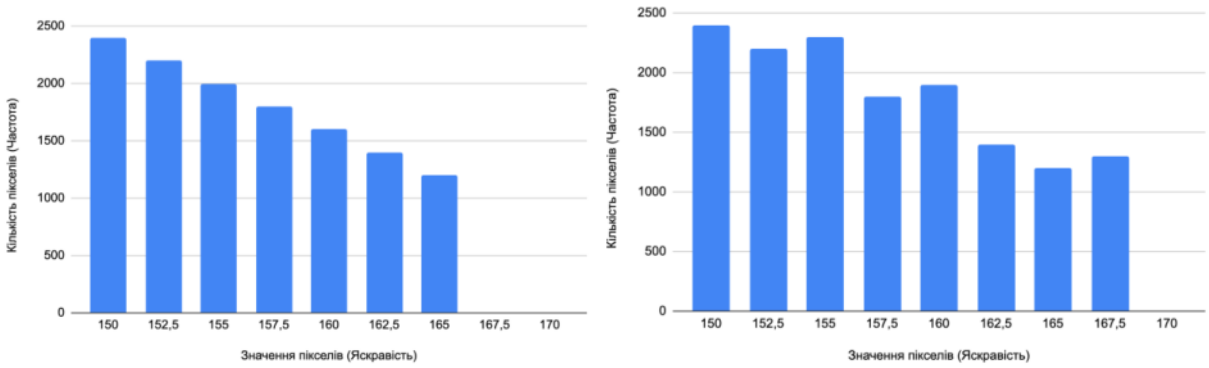
i — індекс пари;

(h2i**,** h2i+1**) / 2** — середнє арифметичне, яке використовується як базове значення для аналізу або модифікації.

3. **Обчислення статистики критерію:** Розраховується сума квадратичних відхилень фактичних значень від очікуваних, нормована на очікуване значення:

$$x^2 = \sum_{i=0}^{k-1} \frac{(h_{2i} - n_i^*)^2}{n_i^*}$$

(2.3.2 (2))



де k — кількість пар значень (у випадку 8-бітного каналу k=128).

4. **Визначення ймовірності вбудовування:** Отримане значення x^2 використовується для розрахунку ймовірності P того, що розподіл є результатом

вбудовування. Для цього використовується інтегральна функція розподілу x^2 з k−1 ступенем свободи:

$$P = 1 - \int_0^{x^2} \frac{t^{(k-1)/2-1} e^{-t/2}}{2^{(k-1)/2}\Gamma(\frac{k-1}{2})dt} \qquad (2.3.2\ (3))$$

де

P — ймовірність (p-value) для критерію χ^2;

χ^2 — значення статистики хі-квадрат;

k — кількість ступенів свободи;

t — змінна інтегрування;

e — основа натурального логарифма;

Γ(x) — гамма-функція;

$\int_0^{\chi^2} \ldots dt$ — визначений інтеграл від 0 до χ^2;

$2^{(k-1)/2}$ — нормувальний коефіцієнт;

вираз під інтегралом — щільність розподілу χ^2;

1 − … — доповнення до інтеграла, що дає правосторонню ймовірність.

Інтерпретація результатів аналізу

Значення ймовірності P коливається в діапазоні від 0 до 1. Якщо P наближається до 1, це свідчить про те, що частоти у парах PoVs ідентичні або дуже близькі, що практично неможливо для немодифікованих зображень. Це може свідчити про те, що в зображення було вбудовано інформацію. Якщо ж P близьке до 0, гіпотеза про наявність стеганографії відхиляється.

Особливості та недоліки методу

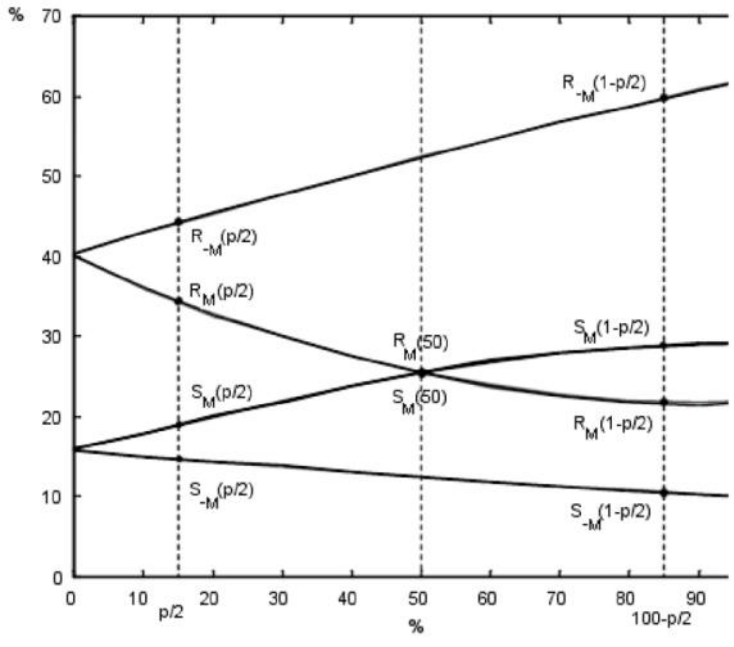
Метод x^2 демонструє високу ефективність при великих обсягах вбудовування (близько 100% ємності LSB-каналу). Проте він має суттєвий недолік: чутливість методу різко падає, якщо повідомлення заповнює лише незначну частину доступних пікселів (чутливість методу знижується за малих обсягів вбудовування).

Крім того, метод виявляє лише факт вбудовування шляхом заміни, але є неефективним проти методу LSB matching (де біт не замінюється, а випадково інкрементується або декрементується), оскільки останній не створює характерного вирівнювання PoVs.

Отже, доцільно розглянути необхідність розгляду більш складних методів, здатних виявляти менші обсяги даних, як-от RS-аналіз.

2.3.3. Метод RS-стегоаналізу та оцінка довжини прихованого повідомлення

Метод RS-аналізу (Regular-Singular), запропонований Дж. Фридрих, належить до точних методів сучасного стегоаналізу [1] (рис. 2.5). На відміну від методу x^2, який аналізує лише загальну статистику гістограми, RS-аналіз досліджує зміну просторових кореляцій між сусідніми пікселями при модифікації їх молодших бітів. Це дозволяє не тільки виявити наявність вбудованих даних, але й з високою точністю (до декількох відсотків) розрахувати довжину вбудованого повідомлення [1].



RS-діаграма

Рисунок 2.5 — Типова RS-діаграма: залежність кількості груп пікселів від щільності вбудовування *(джерело: [1])*

Концептуальні основи методу

Метод базується на поділі зображення на непересічні блоки пікселів G=(x1,…,xn) [1]. Для оцінки ступеня гладкості або варіативності кожного блоку використовується функція дискримінації f(G). Найчастіше застосовується функція, що обчислює суму абсолютних різниць між сусідніми пікселями в блоці:

$$f(x_1,\ldots,x_n) = \sum_{i=1}^{n-1} |x_{i+1} - x_i|$$

(2.3.3)

Для аналізу вводиться операція перетворення бітових площин, що називається "інвертуванням" (flipping).

Операція F1 змінює значення 0 на 1, а 1 на 0 у молодшому біті (що відповідає переходу 2i ↔ 2i+1) [1].

Чому це важливо для аналізу: Якщо ми застосуємо F1 до зображення, де вже є повідомлення, ми фактично "випадково" приберемо частину повідомлення або замінимо його іншим шумом. Статистика зміниться передбачуваним чином.

Операція F−1 виконує аналітичне перетворення, але для пари (−1, 0), тобто 2i−1 ↔ 2i.

Навіщо це потрібно: У природних зображеннях зв'язки між пікселями в парах 2i, 2i+1 та 2i−1, 2i мають певні математичні закономірності [1]. Вбудовування LSB руйнує закономірність у парах F1, але майже не чіпає пари F−1. Порівнюючи реакцію зображення на F1 та F−1, ми можемо точно вирахувати навіть крихітне повідомлення (менше 1−5% ємності).

Операція F0 залишає значення без змін. Це нульова операція. Вона нічого не міняє. У формулах вона потрібна для порівняння ("контрольна група").

Класифікація блоків зображення

Для кожного блоку зображення G застосовується маскування M (сукупність операцій flipping для кожного пікселя блоку). Залежно від того, як змінюється значення функції дискримінації після маскування, блоки класифікуються на три типи:

1. **Regular (R):** якщо f (F(G)) > f(G). Маскування збільшує варіативність блоку.
2. **Singular (S):** якщо f (F(G)) < f(G). Маскування зменшує варіативність, роблячи блок "гладшим".
3. **Unusable (U):** якщо f(F(G)) = f(G). Значення функції не змінюється.

Математична модель виявлення

У природних, немодифікованих зображеннях імовірність того, що маскування збільшить або зменшить гладкість блоку, приблизно однакова. Це формує фундаментальну статистичну рівність для чистих контейнерів:

$$R_m \approx R_{-m} \text{ та } S_m \approx S_{-m} \qquad (2.3.3\ (2))$$

де суфікс m позначає пряму маску (перетворення 0 ↔ 1), а −m – інвертовану, R_m — кількість регулярних груп після застосування прямої маски m;, R_{-m} — кількість регулярних груп після застосування інвертованої маски −m;, S_m — кількість сингулярних груп після застосування прямої маски m;, S_{-m} — кількість сингулярних груп після застосування інвертованої маски −m;, ≈ — наближена рівність, характерна для немодифікованих зображень.

При вбудовуванні повідомлення методом LSB ця рівновага порушується: значення R_m та S_m починають зближуватися, тоді як різниця між R_{-m} та S_{-m} зростає. Аналізуючи графіки цих чотирьох статистик, можна побудувати систему квадратних рівнянь. Розв'язок цієї системи дозволяє отримати значення p — відносну довжину вбудованого повідомлення (від 0 до 1).

Переваги та практичне значення

RS-аналіз є надзвичайно ефективним, тому що він використовує внутрішні властивості самого зображення. Навіть за малих обсягів вбудовування даних від загальної ємності, метод здатний це зафіксувати. Для дипломного дослідження цей метод є критично важливим, оскільки він демонструє, що цифрові зображення мають внутрішню структуру, яку модифікація зображення часто супроводжується порушенням статистичних закономірностей, що може бути використано для аналізу. Саме ці закономірності ми плануємо використовувати як один із критеріїв у розробці власного інтегрованого методу аналізу.

2.3.4. Ентропійний аналіз та інформаційна теоретична модель виявлення аномалій

Ентропійний аналіз у контексті стегоаналізу базується на положеннях теорії інформації К. Шеннона [34], [36]. Якщо візуальні та статистичні методи (наприклад, x^2) орієнтовані на пошук структурних спотворень контейнера, то ентропійний аналіз фокусується на оцінці ступеня хаотичності та інформаційної насиченості окремих компонентів зображення.

Математичний апарат оцінки ентропії

Для цифрового зображення ентропія H є мірою невизначеності значень рівнів яскравості. Для 8-бітного каналу зображення ентропія розраховується за формулою:

$$H = -\sum_{i=0}^{255} p_i log_2 p_i \quad (2.3.4)$$

де p_i – ймовірність появи пікселя з рівнем яскравості i, яка визначається як відношення кількості таких пікселів до загальної кількості пікселів у досліджуваній області. Максимальне значення ентропії для 8-бітного каналу дорівнює 8 біт на піксель, що відповідає абсолютно рівномірному розподілу всіх можливих значень (ідеальний шум).

Ентропія бітових площин як маркер вбудовування

Найбільший інтерес для стегоаналізу становить ентропія найменш значущих бітів (LSB). У природних зображеннях молодші бітові площини містять певний рівень залишкового шуму сенсора або похибок апроксимації кольору, проте вони все ще зберігають певну кореляцію з контурами об'єктів та текстурами старших площин. Як наслідок, ентропія HLSB у чистому зображенні зазвичай помітно менша за 1.

При вбудовуванні інформації (особливо якщо вона попередньо зашифрована за стандартом AES або стиснута), біти повідомлення мають статистичні властивості "білого шуму" з рівною ймовірністю появи 0 та 1. Заміна природних молодших бітів на такі дані призводить до різкого зростання ентропії цієї площини: HLSB→1

Локалізація вбудовування через вікно аналізу

Для підвищення точності аналіз проводиться не для всього зображення, а за допомогою методу "ковзного вікна" (sliding window) розміром W×W. Для кожного положення вікна розраховується локальна ентропія. Це дозволяє побудувати "карту ентропії" зображення, на якій ділянки з вбудованими даними будуть виглядати як зони з аномально високими та стабільними показниками хаотичності на фоні

природних спадів ентропії в однорідних зонах зображення (наприклад, небо або чисте тло) наприклад (рис.2.6).

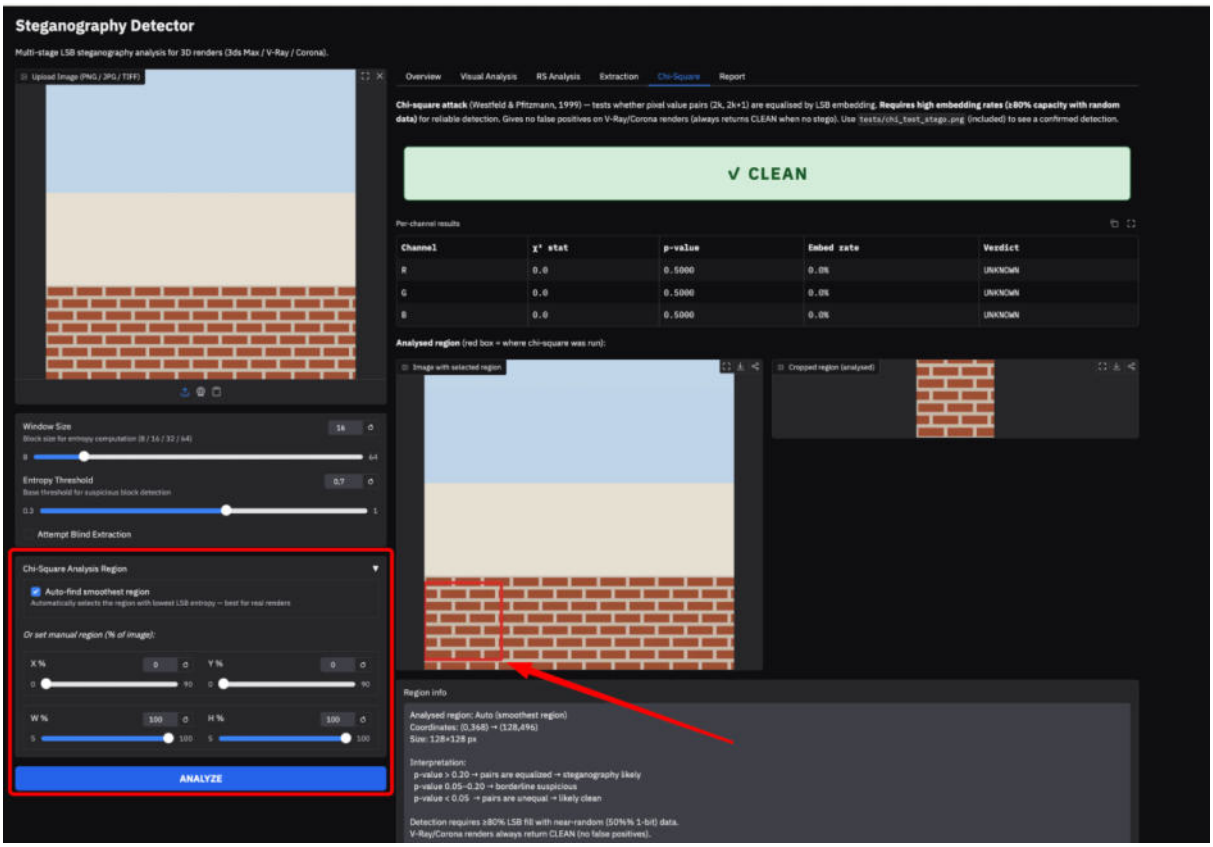


Рисунок 2.6 — Схема сегментації зображення методом «ковзного вікна» для розрахунку локальної ентропії *(джерело: сформовано автором у межах розробленого програмного засобу)*

Роль ентропійного аналізу в 3D-рендерах

У межах даного дослідження з рендерами (3ds Max), цей підпункт є надзвичайно актуальним. Рендеровані зображення часто мають "ідеально чисті" градієнти у старших бітах, але можуть містити специфічний шум (noise), створений алгоритмами трасування променів. У межах даної роботи висувається припущення, що ентропійний аналіз може бути використаний для розмежування шуму рендерингу та стороннього вбудованого шуму, що потребує експериментальної перевірки.

Отже, доцільно перейти до завершальної частини аналізу методів — вивчення обмежень, які заважають існуючим підходам бути універсальними.

2.3.5. Порівняльний аналіз та критичні обмеження існуючих методів статистичного стегоаналізу

Завершуючи детальний огляд статистичного інструментарію, необхідно провести критичну оцінку розглянутих методів. Незважаючи на їх математичну обґрунтованість, жоден із них не є універсальним. Розуміння їхніх слабких місць є необхідною умовою для розробки удосконаленого методу виявлення прихованої інформації.

Порівняльна характеристика методів

Для наочності аналізу доцільно систематизувати ключові можливості розглянутих алгоритмів у порівняльній таблиці 2.2.

Таблиця 2.2 — Порівняльна характеристика статистичних методів стегоаналізу за об’єктом аналізу, перевагами та обмеженнями (складено автором на основі [1], [29], [33], [36])

Метод	**Об'єкт аналізу**	**Головна перевага**	**Основне обмеження**
x^2	Гістограма PoVs	Висока швидкість обчислень	Неефективний при заповненні < 10% ємності
RS-аналіз	Просторова кореляція	Висока точність оцінки довжини	Чутливість до JPEG-стиснення та шумів
Ентропія	Хаотичність бітів	Можливість локалізації вбудовування	Хибні спрацювання на складних текстурах

Критичні обмеження існуючих підходів

В ході аналізу було виділено декілька фундаментальних проблем, які знижують ефективність існуючих методів у реальних умовах:

1. **Чутливість до контенту («Noise-like» зображення).** Більшість методів розраховані на роботу з "типовими" фотографіями. Проте сучасні рендери, згенеровані в системах 3D-моделювання, часто мають специфічну структуру зернистості (noise). Алгоритми RS-аналізу або ентропійної оцінки можуть помилково прийняти природні артефакти рендерингу за ознаку стеганографії, що призводить до високого рівня помилок першого роду (хибна тривога).
2. **Проблема малих обсягів вбудовування.** Коли обсяг прихованих даних складає менше 1-3% від загального обсягу контейнера, статистичні відхилення стають настільки незначними, що вони губляться на фоні природних флуктуацій зображення. Більшість класичних методів у таких умовах демонструють точність, близьку до випадкового вгадування.
3. **Адаптивні методи стеганографії.** Сучасні алгоритми вбудовування (наприклад, поширений метод LSB Matching) не просто замінюють біти, а модифікують їх таким чином, щоб зберегти статистику PoVs незмінною. Це робить такий популярний метод, як x^2 - аналіз, повністю неефективним.
4. **Відсутність інтегрування методів.** Більшість існуючого ПЗ використовує методи окремо. Наприклад, програма може показати графік RS-аналізу, але не зіставити його з візуалізацією бітових площин або картою ентропії. Це ускладнює роботу аналітика, який змушений вручну порівнювати результати різних тестів.

2.4. Сигнатурні методи та аналіз структурних аномалій файлів-контейнерів

Якщо статистичні методи орієнтовані на пошук змін у самому сигналі (значеннях пікселів), то сигнатурні методи фокусуються на виявленні специфічних відбитків (fingerprints), які залишають по собі стеганографічні інструменти. Кожен популярний алгоритм або програма має свій «почерк», що проявляється у структурі файлу, службових заголовках або специфічних послідовностях байтів.

2.4.1. Аналіз цілісності та структури форматів файлів

Більшість користувацьких форматів зображень (PNG, JPEG, TIFF) мають строгу специфікацію структури [6], [7], [28]. Будь-яке відхилення від цього стандарту може розглядатися як ознака наявності сторонніх даних. Сигнатурний аналіз у цьому контексті розпочинається з перевірки цілісності контейнера [18], [33].

Одним із найпростіших, але ефективних методів є виявлення **EOF-вбудовування (End of File)**. Багато примітивних стеганографічних утиліт просто дописують дані після офіційного маркера кінця файлу (наприклад, після байтів FF D9 для JPEG або IEND для PNG) [6], [7]. Програми для перегляду зображень ігнорують все, що йде після цього маркера, проте реальний розмір файлу на диску стає більшим за той, що задекларований у заголовках.

Критерій виявлення:

$$\Delta S = S_{real} - S_{struct} \qquad (2.4.1)$$

де S_{real} – фізичний розмір файлу, а S_{struct} – розмір, обчислений на основі аналізу внутрішніх сегментів (chunks) формату. Якщо $\Delta S > 0$, це свідчить про наявність "оверлею" (overlay), який часто використовується для приховування архівів або текстових файлів.

Також аналізу підлягають метадані (EXIF, IPTC). Зловмисники можуть замінювати рідко використовувані теги (наприклад, опис обладнання або коментарі виробника) на частини зашифрованого повідомлення [37]. Під час аналізу метаданих доцільно звертати увагу на нетипові послідовності символів і високий рівень ентропії в текстових полях.

2.4.2. Пошук шаблонів та ідентифікація стеганографічних інструментів

Професійні стеганографічні утиліти часто додають до вбудованого повідомлення власні технічні заголовки (headers). Це необхідно для того, щоб на стороні отримувача програма могла ідентифікувати початок повідомлення, визначити його тип, довжину та перевірити цілісність через контрольну суму (CRC/Hash).

Специфіка сигнатур популярних утиліт:

1. **OpenStego:** часто використовує специфічну структуру для ідентифікації своїх даних. Аналіз бінарного дампу зображення може виявити фіксовані послідовності байтів, які слугують маркерами для цієї програми [38], [39].
2. **Steghide:** У випадку використання утиліт на зразок Steghide аналіз може бути ускладнений через застосування складніших методів вбудовування та маскування даних [38], [39].
3. **OutGuess:** цей алгоритм намагається зберігати статистику гістограми незмінною, але для корекції цієї статистики він змушений змінювати значення пікселів, які не використовуються для вбудовування. Це створює характерні "кластери" змінених пікселів, які можна виявити за допомогою сигнатурного маскування [38], [39].

Математично цей процес описується як пошук вхідного рядка значень у всьому масиві даних зображення. Пошук ведеться не лише у вихідному файлі, а й у вилучених бітових площинах, оскільки сигнатура часто також вбудовується методом LSB разом із основним payload.

2.4.3. Багатовимірна перевірка на основі бази знань про стего-утиліти

Цей підхід є найбільш просунутим етапом сигнатурного аналізу. Суть методу полягає в автоматизованому зіставленні виявлених аномалій із базою даних відомих атак. Процес включає аналіз специфічних параметрів вбудовування, таких як:

- **Крок вбудовування (Stride):** багато програм вбудовують дані не в кожен піксель, а з певним інтервалом (наприклад, кожен 3-й або 7-й піксель), використовуючи генератор псевдовипадкових чисел (PRNG).У деяких випадках аналіз може включати оцінювання закономірностей псевдовипадкового розподілу позицій вбудовування, зокрема за наявності припущення про використання стандартного генератора псевдовипадкових чисел [40].
- **Палітра та колірні профілі:** деякі інструменти примусово конвертують зображення у певний колірний простір або змінюють палітру (у форматі GIF), що залишає чіткий цифровий слід. Наприклад, якщо зображення має 256 кольорів, але багато з них відрізняються лише на одиницю в одному каналі — це характерна ознака роботи алгоритму EzStego.
- **Специфіка блокової структури:** для алгоритмів, що працюють у частотній області (на основі дискретно-косинусного перетворення, DCT), сигнатурою є викривлення коефіцієнтів у блоках по 8. Це дозволяє ідентифікувати використання таких утиліт, як F5 або JPHide.

Використання сигнатурних методів значно звужує коло пошуку: якщо статистичний аналіз каже "щось вбудовано", то сигнатурний аналіз може конкретно вказати — "це було зроблено програмою OpenStego з використанням LSB-заміни". Це критично важливо для фінального етапу — вилучення інформації, оскільки для успішного дешифрування потрібно знати точний алгоритм, за яким дані були поміщені в контейнер.

2.5. Порівняльний аналіз методів стегоаналізу

Проведений аналіз візуальних, статистичних та сигнатурних методів дозволяє зробити узагальнення щодо їх ефективності, області застосування та обмежень. Кожна з розглянутих груп методів базується на різному підході до аналізу зображення: від сприйняття людиною до формалізованих математичних моделей та аналізу структурних особливостей файлу.

Візуальні методи, зокрема аналіз бітових площин, є найбільш інтуїтивно зрозумілими [1], [35]. Вони дозволяють швидко оцінити наявність аномалій у зображенні без складних обчислень. У таких методах аналітик фактично “бачить” структуру молодших бітів і може виявити неприродні патерни. Проте їх основним недоліком є суб’єктивність. Результат сильно залежить від досвіду користувача, а також від типу зображення. У складних сценах або рендерах, де шум є природною складовою, візуальний аналіз часто не дає однозначної відповіді.

Статистичні методи, навпаки, базуються на формалізованих критеріях і дозволяють отримати кількісну оцінку ймовірності наявності прихованих даних [1], [35], [36]. Такі підходи, як x^2 -аналіз, RS-аналіз та ентропійний аналіз, дозволяють виявляти навіть незначні відхилення у розподілі пікселів. Вони є значно більш точними та придатними до автоматизації. Водночас ці методи потребують обчислювальних ресурсів та чутливі до властивостей самого зображення. Наприклад, текстуровані або зашумлені рендери можуть викликати хибні спрацювання, а малі обсяги вбудованих даних можуть залишатися непоміченими.

Сигнатурні методи працюють за іншим принципом — вони не шукають статистичні аномалії, а намагаються знайти конкретні ознаки використання певного інструменту або алгоритму. Це можуть бути залишки службових заголовків, специфічні послідовності байтів або структурні порушення формату файлу. Основною перевагою такого підходу є висока точність у випадку, коли алгоритм вбудовування відомий. У такому випадку можна не тільки виявити факт приховування, а й перейти до вилучення даних. Однак цей метод є обмеженим, оскільки він не працює для невідомих або модифікованих алгоритмів, а також не

дає результату у випадку використання добре замаскованих або кастомних реалізацій.

Таким чином, можна узагальнити:

- візуальні методи є простими у реалізації, але недостатньо точними для автоматизованого аналізу;
- статистичні методи забезпечують високу точність, але потребують обчислювальних ресурсів та врахування особливостей зображення;
- сигнатурні методи є ефективними лише у випадку відомих алгоритмів та не забезпечують універсальності.

Жоден із розглянутих підходів не може забезпечити стабільний результат у всіх можливих умовах. Це особливо актуально у сучасних системах, де зображення можуть бути як природними фотографіями, так і результатом складного рендерингу, або навіть комбінованими [1], [35], [36].

2.6. Методи вилучення та аналізу прихованої інформації (Payload Analysis)

Процес вилучення даних (stegano-extraction) є фінальним етапом аналізу стего-системи. Якщо методи детектування відповідають на питання про **наявність** втручання, то методи вилучення спрямовані на **відновлення** вихідного секретного повідомлення.

2.6.1. Алгоритм вилучення для відомого методу (White-box)

За умови використання класичного LSB-вбудовування та знання ключа (або за його відсутності), процес вилучення m бітів повідомлення описується операцією [1], [33]:

$$mj = pj' \pmod 2 \qquad (2.6.1)$$

де pj' — значення j-го пікселя стего-контейнера. Отриманий бітовий потік групується у байти (по 8 біт) для відновлення символів тексту або структури файлу.

2.6.2. Сліпе вилучення (Blind Extraction) у складних умовах

У ситуаціях, коли параметри вбудовування невідомі, вилучення перетворюється на ітераційний процес перебору можливих конфігурацій стего-каналу [1], [18]:

1. **Аналіз каналів:** послідовне зчитування LSB з колірних каналів R, G, B та їх комбінацій [1].
2. **Визначення порядку обходу:** перевірка лінійного (Raster scan), випадкового або змієподібного порядків доступу до пікселів.
3. **Аналіз кроку (Stride):** перевірка, чи вбудовування відбувалося у кожен піксель, чи з певним інтервалом (наприклад, кожен третій) [18].

2.6.3. Ідентифікація типу даних через аналіз ентропії

Після вилучення бітового потоку необхідно визначити, чи несуть ці дані корисне навантаження. Для цього використовується **ентропійний аналіз Шеннона [34]**.

- **`Висока ентропія (H≈8)`:** вказує на те, що дані є зашифрованими (AES, DES) або попередньо стиснутими (ZIP, RAR). Такі дані виглядають як білий шум і їх неможливо прочитати без ключа.
- **`Середня ентропія (H≈4.5 біт/символ)`:** характерна для відкритого тексту (ASCII), де частота появи різних символів мови неоднакова.
- **Пошук сигнатур (Magic Bytes):** перевірка перших байтів вилученого блоку на наявність стандартних заголовків файлів (наприклад, 0x4D5A для виконуваних файлів або 0x25504446 для PDF).

2.6.4. Вплив формату контейнера на успішність вилучення

У синтетичних зображеннях (рендерах), збережених у форматі PNG, вилучення відбувається без втрат [6], [1]. Натомість спроба вилучити LSB-повідомлення з файлу, який був перезбережений у форматі **JPEG**, призведе до отримання

пошкоджених даних ("сміття") через квантування DCT-коефіцієнтів, що руйнує цілісність молодших бітів [7], [1].

2.7. Висновки до розділу 2

У даному розділі було розглянуто основні підходи до виявлення прихованої інформації у цифрових зображеннях, включаючи візуальні, статистичні та сигнатурні методи. Кожен із підходів має власну область ефективного застосування, проте жоден із них не є універсальним рішенням.

Проведений аналіз показав, що для побудови ефективного методу стегоаналізу необхідно враховувати декілька ключових вимог.

По-перше, метод повинен бути придатним до автоматизації. В умовах сучасних інформаційних систем обсяг оброблюваних зображень є надзвичайно великим, що робить неможливим ручний аналіз. Відповідно, алгоритм повинен забезпечувати стабільні результати без участі користувача.

По-друге, метод має бути незалежним від конкретного алгоритму вбудовування. Оскільки існує велика кількість стеганографічних підходів, орієнтація лише на сигнатури або відомі шаблони значно обмежує ефективність аналізу. Тому доцільно використовувати універсальні ознаки, такі як статистичні або інформаційні характеристики зображення.

По-третє, важливою є можливість не тільки виявлення, але й вилучення прихованої інформації. Більшість існуючих методів зосереджені лише на детекції, однак практична цінність системи значно зростає, якщо вона дозволяє відновити вбудовані дані або принаймні локалізувати область їх розміщення.

Таким чином, результати аналізу обґрунтовують доцільність розробки комбінованого методу, який поєднує візуальні та статистичні підходи для підвищення точності виявлення, а також використовує елементи сигнатурного аналізу для підвищення ефективності вилучення інформації.

Отримані висновки є основою для формування підходу, який буде розроблено у наступному розділі.

РОЗДІЛ 3. РОЗРОБКА КОМБІНОВАНОГО МЕТОДУ ВИЯВЛЕННЯ ТА ВИЛУЧЕННЯ СТЕГАНОГРАФІЧНИХ ВКЛАДЕНЬ У СИНТЕТИЧНИХ ЗОБРАЖЕННЯХ

3.1. Постановка задачі та вимоги до методу

3.1.1. Опис об'єкта дослідження: синтетичні зображення високої роздільної здатності

Основним об’єктом дослідження у даній роботі є цифрові статичні зображення, отримані шляхом тривимірної візуалізації (рендерингу) у спеціалізованому програмному забезпеченні (наприклад, Autodesk 3ds Max із використанням систем рендерингу V-Ray або Corona Renderer) [20], [21], [22]. Типова довжина повідомлення, що передається через такі канали, часто не перевищує 5-10% від загальної ємності

На відміну від цифрових фотографій, отриманих за допомогою фотокамер, синтетичні зображення (рендери) мають низку специфічних характеристик, які безпосередньо впливають на процес стеганографічного аналізу:

1. **Відсутність природного сенсорного шуму:** У класичній фотографії кожен піксель містить "шум матриці" (ISO noise), який створює природну варіативність у молодших бітах (LSB). У синтетичних зображеннях колір обчислюється математично. Це призводить до того, що на великих площах (наприклад, рівні стіни, стеля, однотонні фасади меблів) значення пікселів можуть бути ідентичними або змінюватися за суворим лінійним законом.
2. **Ідеальні градієнти:** Рендеринг високої якості (FullHD, 4K) забезпечує плавні колірні переходи. Будь-яке вбудовування сторонньої інформації методом LSB навіть із низькою щільністю (менше 1% від обсягу контейнера) миттєво

вносить математичний хаос у ці ідеальні структури, що робить аномалії більш помітними для статистичних методів.

3. **Висока роздільна здатність та глибина кольору:** Робота з роздільною здатністю 4K та форматами без втрат (PNG, TIFF) надає величезний простір для прихованої інформації (ємність контейнера). Наприклад, зображення 4K (3840x2160) містить понад 8 мільйонів пікселів. При використанні лише одного каналу (наприклад, Blue) та одного молодшого біта, теоретична ємність складає близько 1 Мбайт даних, що дозволяє передавати великі обсяги конфіденційної інформації (архіви, фрагменти коду, текстові документи).
4. **Використання у професійному середовищі:** Рендери часто передаються через CRM-системи, хмарні сховища або корпоративну пошту як частина проєктної документації. Це створює ідеальне маскування для передачі даних, оскільки великий розмір файлу (10-50 Мб для PNG) не викликає підозр у систем моніторингу трафіку (DLP-систем).

Таким чином, об’єктом дослідження обрано синтетичні контейнери як такі, що мають специфічну структуру, яку можна використати для підвищення точності детектування в порівнянні зі стандартними методами, розробленими для природних фотографій.

3.1.2. Вимоги до точності детектування при низькій щільності заповнення контейнера

Найбільшою проблемою сучасної стегоаналітики є виявлення повідомлень з низькою щільністю заповнення (Low Embedding Rate). У контексті даного дослідження під низькою щільністю розуміється використання менше **5%** від загальної теоретичної ємності контейнера [1], [29].

До розроблюваного методу висуваються наступні вимоги щодо точності:

1. **Мінімізація помилок другого роду (False Negative):** Це ситуація, коли система позначає стего-контейнер як "чисте" зображення [33], [36]. У задачах кібербезпеки пропуск прихованого каналу зв'язку є критичним, тому метод повинен бути стабільно чутливим до мікро-аномалій у молодших бітах, які виникають навіть при вбудовуванні невеликих обсягів даних (наприклад, одиничних команд C2-сервера).
2. **Контроль помилок першого роду (False Positive):** Синтетичні зображення можуть містити зони зі складними процедурними текстурами або ефектами (наприклад, "шум" від недостатньої кількості проходів рендеру/Sampling Noise). Метод не повинен помилково ідентифікувати цей технічний шум як стеганографію [33], [36].
3. **Стійкість до "розосередженого" вбудовування:** Сучасні стего-алгоритми часто не записують дані послідовно (піксель за пікселем), а розподіляють їх псевдовипадковим чином по всьому зображенню [18], [40]. Вимогою до методу є здатність виявляти такі вкраплення через інтегральну оцінку ентропії всього зображення або великих його фрагментів.
4. **Визначення порогу детектування для різних форматів:** Оскільки об'єктом дослідження є рендери, метод повинен автоматично адаптувати поріг чутливості (Threshold). Для ідеально гладких зон (стіни) поріг має бути мінімальним, тоді як для зон з текстурами (дерево, камінь) — більш ліберальним, щоб уникнути хибних спрацювань.

Цільовий показник точності: Метод повинен забезпечувати впевнене детектування (Probability of Detection $Pd > 0.9$) при щільності вбудовування від 1-3% у зонах з низькою текстурною складністю.

3.1.3. Формулювання задачі сліпого вилучення даних

Задача вилучення інформації (extraction) у даному дослідженні формулюється як «сліпа» (blind extraction), оскільки проводиться в умовах повної відсутності апріорних даних про параметри стего-системи [1], [33]. На відміну від прямого

зчитування, де відомі алгоритм та ключ, задача сліпого вилучення в контексті аналізу рендерів включає наступні підзадачі:

1. **Автоматичне визначення активного каналу (Channel Identification):** Більшість LSB-методів використовують лише один або два колірні канали (найчастіше Blue або Alpha-канал у PNG) для мінімізації візуальних спотворень. Задача полягає в ідентифікації саме того каналу (R, G, B чи A), статистичні параметри якого найбільше відхиляються від еталонних значень синтетичного зображення [34].
2. **Реконструкція бітового потоку:** Необхідно реалізувати механізм послідовного зчитування молодших значущих бітів (LSB) з виявлених підозрілих зон [1]. Проблема ускладнюється тим, що дані можуть бути вбудовані не в кожен піксель, а з певним кроком (stride) або за певним шаблоном, що потребує перебору найбільш імовірних сценаріїв [40].
3. **Експертний аналіз корисного навантаження (Payload Evaluation):** Вилучений бітовий потік сам по собі є сирими даними (raw data). Постановка задачі передбачає розробку інструментарію для первинної оцінки типу вилучених даних:
    - **Для тексту:** перевірка на відповідність кодуванням (ASCII, UTF-8).
    - **Для зашифрованих файлів:** обчислення показника ентропії (якщо $H > 7.9$, дані ідентифікуються як зашифрований контейнер).
    - **Для вкладених об'єктів:** пошук файлових сигнатур (magic bytes), що дозволяють відновити файлову структуру (наприклад, заголовки .zip, .txt або .exe).
4. **Оцінка цілісності вилучених даних:** Враховуючи специфіку об'єкта дослідження (висока роздільна здатність рендерів), задача полягає у вилученні максимального обсягу даних без їх пошкодження, що в подальшому дозволить фахівцю з кібербезпеки провести повноцінний криміналістичний аналіз (Digital Forensics).

Формулювання задачі вилучення як «сліпої» дозволяє розробити універсальний алгоритм, здатний протидіяти широкому спектру стеганографічних атак, незалежно від конкретного програмного забезпечення, що використовувалося для вбудовування. Загальну блок-схему запропонованого методу наведено на рис. 3.1.

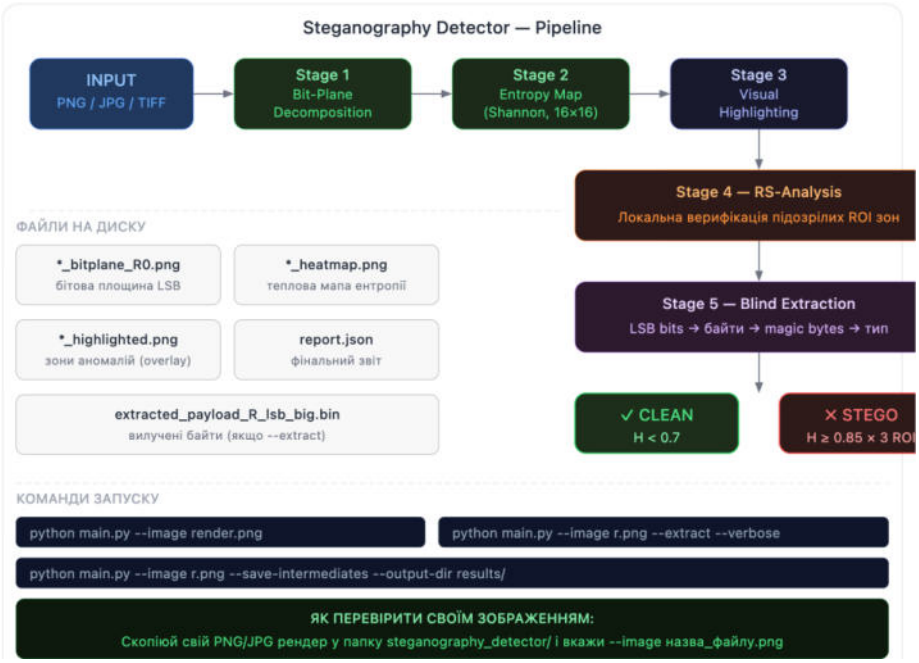


Рисунок 3.1 — Блок-схема комбінованого методу виявлення стеганографічних вкладень (джерело: сформовано автором)

3.2. Обґрунтування концепції комбінованого аналізу

3.2.1. Порівняльна характеристика статичних шумів цифрового фото та синтетичного рендеру

Для розробки ефективного методу детектування необхідно провести детальне порівняння структури контейнера природного походження (фотографія) та штучного (рендер). Ключова відмінність полягає в генезисі наймолодшого біта (LSB), який у стеганографії використовується як "шумова підкладка".

1. Природа шуму в цифрових фотографіях (Сенсорний шум)
Кожне цифрове зображення, отримане за допомогою фотокамери, містить природний шум, спричинений фізичними процесами в напівпровідниковій матриці [37]:

- **Фотонний шум:** Виникає через випадкову природу потоку фотонів [35].
- **Тепловий шум:** Результат хаотичного руху електронів у сенсорі, що особливо помітно при високих значеннях ISO. Ці шуми є стохастичними (випадковими). Це означає, що значення молодших бітів у фотографії завжди мають високу ентропію (H≈1 на біт) навіть у зонах ідеального білого кольору. Для стеганографії це ідеальне прикриття: вбудовані дані мімікрують під цей природний "білий шум".

2. Генезис пікселя у синтетичних рендерах (Математична модель)
У синтетичних зображеннях, створених у середовищі 3ds Max, колір пікселя — це не результат вимірювання фізичного світла, а результат математичного розрахунку (Ray Tracing) [20], [21], [22]:

- **Детермінованість:** Якщо ми маємо ідеальну площину зі сталим матеріалом та рівномірним освітленням, алгоритм рендерингу (наприклад, V-Ray) розраховує практично однаковий колір для тисяч сусідніх пікселів.
- **Структура LSB:** У таких зонах молодші біти або ідентичні (0-0-0), або змінюються за суворою логічною послідовністю (градієнтом). Ентропія LSB-площини в рендерах суттєво нижча за 1 біт на піксель.
- **Відсутність "зерна":** Сучасні алгоритми (Denoising) та якісний Sampler роблять зображення математично "гладким".

3. Кількісне порівняння для стегоаналізу
Для наочності характеристики зведено у порівняльну таблицю:

Таблиця 3.1 — Порівняльна характеристика стохастичних властивостей цифрових фотографій та синтетичних рендерів

Характеристика	Цифрове фото	Синтетичний рендер
Джерело молодшого біта	Фізичний шум матриці	Математичний розрахунок кольору
Ентропія LSB (чистий фон)	Висока (близька до 1.0)	Низька (близька до 0.1 - 0.4)
Кореляція між площинами	Слабко виражена між 0-ю та 1-ю	Висока в зонах градієнтів
Вплив LSB-вбудовування	Зливається з шумом	Створює різку статистичну аномалію

Висновки для методу:

Враховуючи вищезазначене, синтетичні зображення є "крихкими" контейнерами. Того рівня хаосу, який у фотографії є природною нормою, у рендері не існує. Отже, будь-яка спроба вбудувати дані методом LSB призводить до аномального зростання ентропії саме в тих зонах, де математична модель рендеру мала б забезпечити чистоту або плавність переходу. Ця відмінність лягає в основу методу "мапування ентропії" (Step 2 алгоритму).

3.2.2. Гіпотеза про використання візуальних аномалій як пре-фільтра для математичних методів

Традиційні методи стегоаналізу зазвичай застосовуються до всього масиву даних зображення. Однак, враховуючи специфіку об'єкта дослідження (4K-рендери), такий підхід є надлишковим. У даному підрозділі висувається та обґрунтовується гіпотеза про ефективність **попередньої селекції зон інтересу**.

Суть гіпотези:

Стеганографічне повідомлення, вбудоване методом LSB, утворює в синтетичному контейнері зони з «неприродною» візуальною енергією на рівні молодших бітових

площин. Ці зони можна ідентифікувати візуально-алгоритмічним шляхом ще до застосування складних статистичних тестів.

Основні положення гіпотези:

1. **Принцип контурної відповідності:** У «чистому» рендері нульова бітова площина (k=0) зазвичай напівпрозора або містить лише ледь помітні обриси (контури) об'єктів, що виникають через ефекти згладжування (Anti-aliasing). Якщо ж у площині з'являються щільні, хаотичні блоки, які не повторюють геометрію сцени (наприклад, «квадрат» шуму посеред гладкої стіни), — це пряма ознака втручання.
2. **Візуальний маркер як фільтр обчислень:** Замість того, щоб проводити трудомісткий RS-аналіз (який вимагає багатьох операцій інвертування та порівняння) для всіх 8 мільйонів пікселів, пропонується використовувати **бітову маску аномалій**. Математичний аналіз проводиться лише там, де "мапа аномалій" показує перевищення порогу ентропії.
3. **Ефект "теплової карти":** Гіпотеза передбачає, що трансформація нульової бітової площини у контрастну мапу (наприклад, через заміну значень бітів на діаметрально протилежні кольори) дозволяє експерту миттєво локалізувати повідомлення. Це особливо ефективно для рендерів інтер'єрів, де багато монотонних поверхонь.

Переваги підходу для кібербезпеки:
Використання візуальних аномалій як пре-фільтра дозволяє вирішити проблему «голки в стозі сіна». Навіть якщо зловмисник вбудував лише кілька кілобайт даних, вони створять локальний «шумовий спалах» на чистому полі рендеру. Це робить метод значно чутливішим до вкладень малої щільності, які часто ігноруються глобальними статистичними тестами.

3.2.3. Переваги поєднання аналізу бітових площин та RS-аналізу для оптимізації швидкодії

У сучасних системах моніторингу трафіку швидкість обробки даних є такою ж важливою, як і точність. Робота з зображеннями надвисокої роздільної здатності (4K та вище) створює значне навантаження на обчислювальні ресурси. У даному підрозділі розглядається економічна та алгоритмічна ефективність запропонованого комбінованого підходу.

1. Проблема обчислювальної складності класичних методів

RS-аналіз, як один із найбільш точних методів стегоаналізу, базується на багатократному застосуванні інвертуючих масок (Fl,F−l) та вимірюванні статистичної кореляції для кожної групи пікселів.

- Для зображення формату 4K (3840 x 2160 ≈ 8.3 млн пікселів) повний цикл RS-аналізу вимагає значного часу (від кількох секунд до десятків секунд на одне зображення залежно від апаратної потужності).
- У потоці реального часу така затримка є неприпустимою.

2. Синергія методів: Дворівнева система фільтрації

Запропонований метод вирішує проблему швидкодії через розподіл навантаження:

- **Перший рівень (Bit-plane slicing):** Це операція бітового зсуву, яка має константну швидкість. Аналіз ентропії за бітовою площиною дозволяє майже миттєво (за мілісекунди) виділити області, які потребують детальної перевірки [34].
    - *Результат:* до 90% поверхні рендеру відсікається як "завідомо чиста".
- **Другий рівень (Localized RS-inspection):** Математично складний RS-аналіз запускається лише у межах виявлених фрагментів зображення (ROI — Regions of Interest). Якщо повідомлення займає лише 5% площі контейнера,

то і обчислювальні витрати на детектування скорочуються приблизно у 20 разів [1], [18].

3. Технологічні переваги комбінованого підходу:

- **Оптимізація пам'яті:** Замість зберігання та обробки всього масиву даних, система фокусується на локальних картах аномалій.
- **Підвищення вірогідності (Double Check):** Використання двох різнорідних методів (візуально-ентропійного та кореляційного) знижує ризик хибних спрацювань. Навіть якщо текстура дерева на рендері має високу "природну" ентропію, RS-аналіз підтвердить, що кореляція в групах пікселів не порушена, і система не видасть тривогу.
- **Масштабованість:** Даний алгоритм дозволяє обробляти великі масиви проектної документації (десятки рендерів одночасно) без перевантаження серверів аналізу.

Висновки:

Таким чином, поєднання аналізу бітових площин та локального RS-аналізу дозволяє побудувати "інтелектуальну" систему детектування. Вона забезпечує високу швидкість скринінгу завдяки бітовим фільтрам та лабораторну точність детектування завдяки статистичній верифікації аномальних зон.

3.3. Алгоритм роботи запропонованого методу

3.3.1. Етап декомпозиції та ентропійного мапування (Bit-Plane Mapping)

Перший етап роботи методу полягає у підготовці даних та проведенні первинного статистичного скринінгу. Цей процес розділений на два ключові кроки: розшарування контейнера та обчислення локальних метрик інформаційної щільності.

1. Процес розшарування на бітові площини (Bit-Plane Slicing)

Цифрове зображення (наприклад, у 24-бітній моделі RGB) складається з трьох

каналів, кожен з яких має 8 біт на колір. Традиційне LSB-вбудовування використовує наймолодший значущий біт (LSB).
Алгоритм виконує операцію декомпозиції, виділяючи 0-ву площину (k=0) для кожного кольорового каналу. Математично це реалізується через побітове заперечення (AND) значення пікселя P з маскою 1 (00000001_2) [34]:

$$L0(x,y)=P(x,y) \,\&\, 1 \qquad (3.3.1)$$

де

L0(x,y)) — значення наймолодшого біта (LSB) пікселя з координатами *(x, y)*;
P(x,y) — повне значення інтенсивності пікселя або його окремої колірної компоненти;
& — операція побітового логічного «І» (Bitwise AND);
1 — бітова маска (у двійковому вигляді `00000001`), що використовується для виділення нульового розряду.

Для синтетичних рендерів цей крок дозволяє відокремити основний візуальний контент від шумової складової. Результатом є бінарна матриця (чорно-біле зображення), де білі пікселі відповідають значенню біта «1», а чорні — «0».

2. Обчислення локальної ентропії Шеннона для рівня k=0
Після отримання площини LSB алгоритм проводить її статистичний аналіз. Використання глобальної ентропії для 4K-зображення є малоефективним, тому в методі застосовується концепція **ковзного вікна (Sliding Window)**.

Зображення розбивається на блоки розміром N×N (наприклад, 16 або 32 пікселі). Для кожного блоку розраховується ентропія Шеннона за формулою [34]:

$$H = -\sum_{i=1}^{n} p_i log_2 p_i \qquad (3.3.1\ (2))$$

де pi — ймовірність появи відповідного значення біта (0 або 1) у даному блоці.

Логіка аналізу:

- **У чистому рендері:** В зонах гладких поверхонь ентропія H буде близька до мінімальної (0-0.3), оскільки біти мають високу кореляцію або є ідентичними.
- **У стего-контейнері:** Вбудовані дані (які зазвичай є зашифрованими або стиснутими) мають властивість "білого шуму". У таких блоках ймовірність появи 0 та 1 вирівнюється (p ≈ 0.5), що призводить до різкого зростання ентропії до значень H ≈ 1H на біт.

Результат етапу:
Формується **Ентропійна мапа (Entropy Map)** — матриця низької роздільної здатності, де кожна клітинка відображає рівень інформаційного хаосу у відповідному блоці зображення. Це дозволяє на наступному етапі локалізувати повідомлення, відкидаючи 95% порожнього простору рендеру.

3.3.2. Локалізація та візуальне виділення підозрілих зон (Visual Highlighting)

Після отримання ентропійної мапи на попередньому етапі, алгоритм переходить до фази візуалізації результатів. Метою цього етапу є створення інтуїтивно зрозумілого графічного інтерфейсу ("теплової карти"), яка дозволяє фахівцю з кібербезпеки миттєво локалізувати область втручання.

1. Алгоритм створення мапи теплових аномалій (Heatmap Generation)
На цьому кроці дані про ентропію кожного блоку N×N перетворюються на колірні координати. Використовується порогова обробка:

- Для блоків, де H < Hporog (норма для рендеру), колір залишається прозорим або нейтральним.
- Для блоків з критичним рівнем ентропії (H→1) застосовується градієнтне зафарбовування (від жовтого до яскраво-червоного).

Мапа накладається на оригінальне зображення як напівпрозорий шар (overlay). Це дозволяє побачити, чи збігаються аномалії з геометрією об'єктів (наприклад, аномалія на рівній стіні виглядає значно підозріліше, ніж аномалія на складному листі рослини).

2. Колоризація ділянок за методом "діаметральної протилежності"

Для більш детального вивчення структури аномалії у 0-й бітовій площині застосовується метод посилення контрасту. У виявлених підозрілих зонах значення бітів "0" та "1" замінюються на максимально контрастні кольори (наприклад, яскраво-зелений та яскраво-фіолетовий).

Це дає змогу візуально відрізнити:

- **Природні структури:** де переходи бітів мають певну логіку або повторюють контур меблів/освітлення.
- **Стего-вкладення:** яке виглядає як "статичний пісочний шум" без жодної прив'язки до змісту рендеру.

3. Роль експертного аналізу в локалізації

Візуальне виділення вирішує проблему «інтелектуального фільтра». Експерт може помітити характерні "прямокутні вкраплення", що часто виникають, коли стеганографічне ПЗ вбудовує дані блоками. У синтетичних зображеннях (рендерах) такі структури є неприродними, що дозволяє відкинути хибні спрацювання, спричинені складною текстурою матеріалів (наприклад, шумом у віддзеркаленнях скла або металу).

Результат етапу:

Користувач отримує зображення рендеру з чітко позначеними зонами ймовірного вкладення. Ці зони передаються на наступний етап для математичної верифікації методами RS-аналізу.

3.3.3. Верифікація за допомогою локального RS-аналізу

Застосування візуального фільтра дозволяє локалізувати аномалії, проте для остаточного висновку та оцінки параметрів вкладення необхідна математична верифікація. На цьому етапі алгоритм використовує метод **RS-стегоаналізу**, адаптований для роботи з обмеженими областями інтересу (ROI — Regions of Interest).

1. Застосування операцій інвертування (F1,F−1) у зонах інтересу

Для кожної виявленої "підозрілої" зони алгоритм проводить серію маніпуляцій з молодшими бітами. На відміну від класичного RS-аналізу, який обробляє все зображення, наш метод фокусується на локальних групах пікселів:

- **Операція F1:** Інвертує останній біт (0 ↔ 1), імітуючи процес LSB-вбудовування.
- **Операція F−1:** Виконує зміщене інвертування (2i−1 ↔ 2i.), що дозволяє оцінити "природну стійкість" статистичних зв'язків у рендері.

Для цих операцій використовуються спеціальні маски (наприклад, [0,1,1,0][0, 1, 1, 0][0,1,1,0]), які допомагають виміряти, як змінюється гладкість (noisiness) зображення після кожного типу інвертування [1], [18].

2. Побудова діаграми RS-станів та розрахунок довжини повідомлення

Алгоритм підраховує кількість груп пікселів у трьох станах [1], [18], [29]:

- **R (Regular):** Групи, де "шумність" зростає після інвертування.
- **S (Singular):** Групи, де "шумність" зменшується.
- **U (Unusable):** Групи, де показник не змінюється.

Математична логіка детектування:

У чистому рендері кількість груп RRR значно перевищує S для обох типів інвертування (**F1** та **F−1**). Однак, якщо в зону вмонтовано дані, криві R та S

починають зближуватися [1], [18].

Точна оцінка довжини повідомлення (z) розраховується через розв'язання квадратичного рівняння, що описує перетин цих статистичних кривих.

3. Переваги локального підходу до RS-верифікації:

- **Ігнорування "шумних" матеріалів:** Якщо аномалія була викликана складною текстурою дерева або каменю на рендері, RS-аналіз покаже, що статистичні зв'язки в цих зонах є природними, і не підтвердить вкладення. Це радикально знижує кількість помилкових спрацювань (False Positives).
- **Точність оцінки:** Обчислення довжини повідомлення саме в зоні вкладення (а не по всьому 4K-файлу) дає набагато точніший результат, оскільки "порожні" області зображення не розмивають статистику.

Результат етапу:

Система видає кількісний показник: *"В даній зоні виявлено вбудовані дані обсягом X біт з імовірністю Y%"*. Це є фінальним підтвердженням перед переходом до вилучення даних.

3.3.4. Сліпе вилучення та аналіз типу корисного навантаження (Blind Extraction)

Після математичного підтвердження наявності вкладення та локалізації його меж, алгоритм приступає до завершальної стадії — вилучення та первинного аналізу прихованої інформації. Оскільки параметри вбудовування невідомі, цей процес виконується в декілька ітерацій.

1. Алгоритм сліпого зчитування бітового потоку

У межах ідентифікованої зони аномалії система здійснює послідовне вилучення молодших значущих бітів (LSB). Процес включає перебір найбільш імовірних сценаріїв:

- **Вибір колірного каналу:** Зчитування проводиться окремо для R, G, B та A каналів, оскільки аномалія могла бути зафіксована лише в одному з них (наприклад, у Blue-каналі, який є найменш помітним для людського ока).
- **Реконструкція байтів:** Вилучені біти групуються в 8-бітові пакети (октети). Алгоритм перевіряє два напрямки бітового порядку: від старшого до молодшого (Big-endian) та навпаки (Little-endian) [34].

2. Аналіз ентропії вилученого блоку (Payload Density)

Для вилученого масиву байт розраховується фінальний показник ентропії. Це дозволяє класифікувати тип інформації:

- **Hfinal < 6.0:** Велика ймовірність, що вилучені дані є відкритим текстом (ASCII), де символи повторюються з певною частотою.
- **Hfinal > 7.9:** Дані є ідеально хаотичними. Це свідчить про використання шифрування (наприклад, AES) або архівації. У такому разі пряме читання неможливе, але факт передачі зашифрованого контейнера є підтвердженим.

3. Ідентифікація за сигнатурами та Magic Bytes

Для перевірки, чи є вилучений потік прихованим файлом, алгоритм проводить пошук стандартних сигнатур у перших байтах блоку. Пошукова таблиця включає найбільш поширені формати [37]:

- 0x50 0x4B 0x03 0x04 — сигнатура ZIP-архіву;
- 0xFF 0xD8 — початок JPEG-зображення;
- 0x7B — початок JSON-об'єкта або програмного коду.

Якщо сигнатуру знайдено, система намагається автоматично відновити розширення файлу та зберегти його як окремий об'єкт для подальшого вивчення.

4. Звіт про результати аналізу

Фінальним результатом роботи методу є генерація звіту, що включає:

1. **Візуальну мапу** з позначеними зонами втручання.

2. **Оціночний обсяг** знайдених даних (у байтах).
3. **Вилучений фрагмент** даних (повністю або частково).
4. **Рекомендацію** щодо типу вкладення (наприклад: "Ймовірний зашифрований архів").

Результат етапу:
Завершення циклу стеганографічного аналізу рендеру від детектування мікровідхилень до отримання фізичного масиву прихованої інформації.

3.4. Опис програмної реалізації системи стеганоаналізу

Для практичної реалізації запропонованого комбінованого методу було розроблено програмний засіб **«Steganography Detector»**, призначений для автоматизованого виявлення та вилучення прихованої інформації у цифрових зображеннях, зокрема синтетичних 3D-рендерах.

3.4.1. Архітектура програмного засобу

Програмний засіб реалізовано мовою **Python 3.10** з використанням модульної архітектури. Основна логіка розділена на окремі компоненти:

- bitplane.py — декомпозиція зображення на бітові площини [34];
- entropy_map.py — розрахунок локальної ентропії;
- visualizer.py — генерація теплових карт та візуалізацій;
- rs_analysis.py — реалізація RS-стегоаналізу [1], [18];
- chi_square.py — статистичний аналіз методом χ^2 [1];
- extractor.py — сліпе вилучення прихованих даних;
- app.py — графічний інтерфейс користувача.

Архітектуру програмного засобу подано на рис. 3.2.

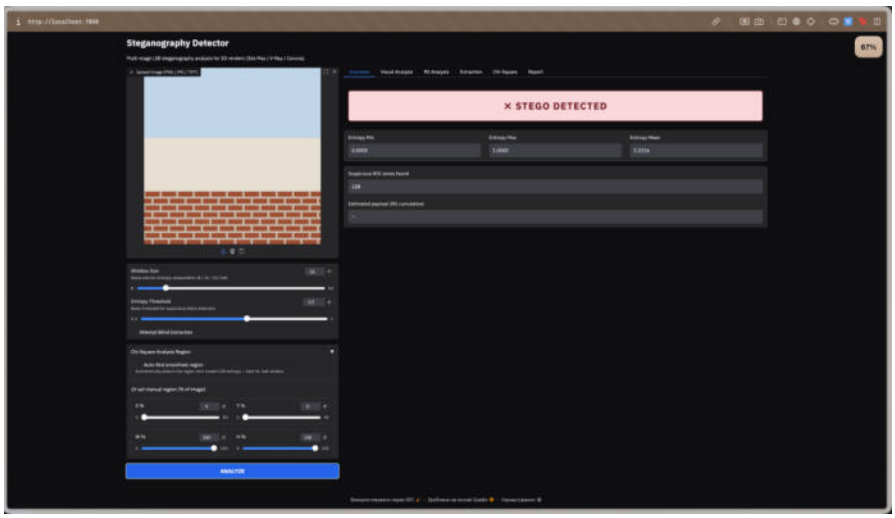


Рисунок 3.2 — Інтерфейс розробленого програмного засобу для стегоаналізу рендерів (джерело: сформовано автором)

3.4.2. Основні етапи роботи системи

Робота програми базується на комбінуванні кількох методів стегоаналізу, що виконуються послідовно:

1. **Декомпозиція бітових площин**

   Зображення розкладається на бітові рівні, що дозволяє ізолювати молодші біти (LSB), у яких найчастіше здійснюється вбудовування даних.
2. **Ентропійний аналіз**

   Для кожного блоку зображення (8×8 – 64×64) обчислюється ентропія Шеннона.

   Підозрілими вважаються області з аномально високим рівнем хаотичності.
3. **Локалізація підозрілих зон (ROI)**

   Формується карта аномалій, що дозволяє виділити області, де ймовірне вбудовування даних.
4. **RS-аналіз**

   Для кожної підозрілої області проводиться аналіз співвідношення регулярних (R) та сингулярних (S) груп .

Це дозволяє:

- підтвердити факт стеганографії;
- оцінити обсяг вбудованих даних.

5. **χ^2 (Chi-square) аналіз**

   Додатково застосовується статистичний метод перевірки рівномірності розподілу пар значень пікселів.

6. **Сліпе вилучення (Blind Extraction)**

   У випадку виявлення стеганографії виконується спроба відновлення прихованих даних.

3.4.3. Реалізація χ^2-аналізу та виправлення моделі

У ході реалізації було виявлено критичну особливість класичного χ^2-методу.

Первинно використовувалась стандартна логіка:

- $p>0.95$ — наявність стеганографії
- $p>0.50$ — підозра

Однак експериментально встановлено, що для **випадкового LSB-вбудовування** значення χ^2 статистики наближається до числа ступенів свободи, що дає $p \approx 0.5$ p [1].

Таким чином, класичні пороги є некоректними для практичного застосування.

Було введено адаптивні порогові значення [1]:

- $p > 0.20$ — **STEGO DETECTED**
- $0.05 < p \leq 0.20$ — **SUSPICIOUS**
- $p \leq 0.05$ — **CLEAN**

Це дозволило:

- усунути проблему невиявлення стеганографії;
- забезпечити коректну роботу методу на синтетичних даних.

3.4.4. Тестування χ^2-методу

Для перевірки роботи було створено спеціальні тестові зображення:

- **chi_test_clean.png**
  Градієнтне зображення з парними значеннями пікселів
  → результат: CLEAN
- **chi_test_stego.png**
  Те саме зображення зі 100% випадковим LSB-вбудовуванням
  → результат: STEGO DETECTED

Також було встановлено, що χ^2 ефективний лише при високій щільності вбудовування (≈80% і більше). Приклади тестових рендерів наведено на рис. 3.3.

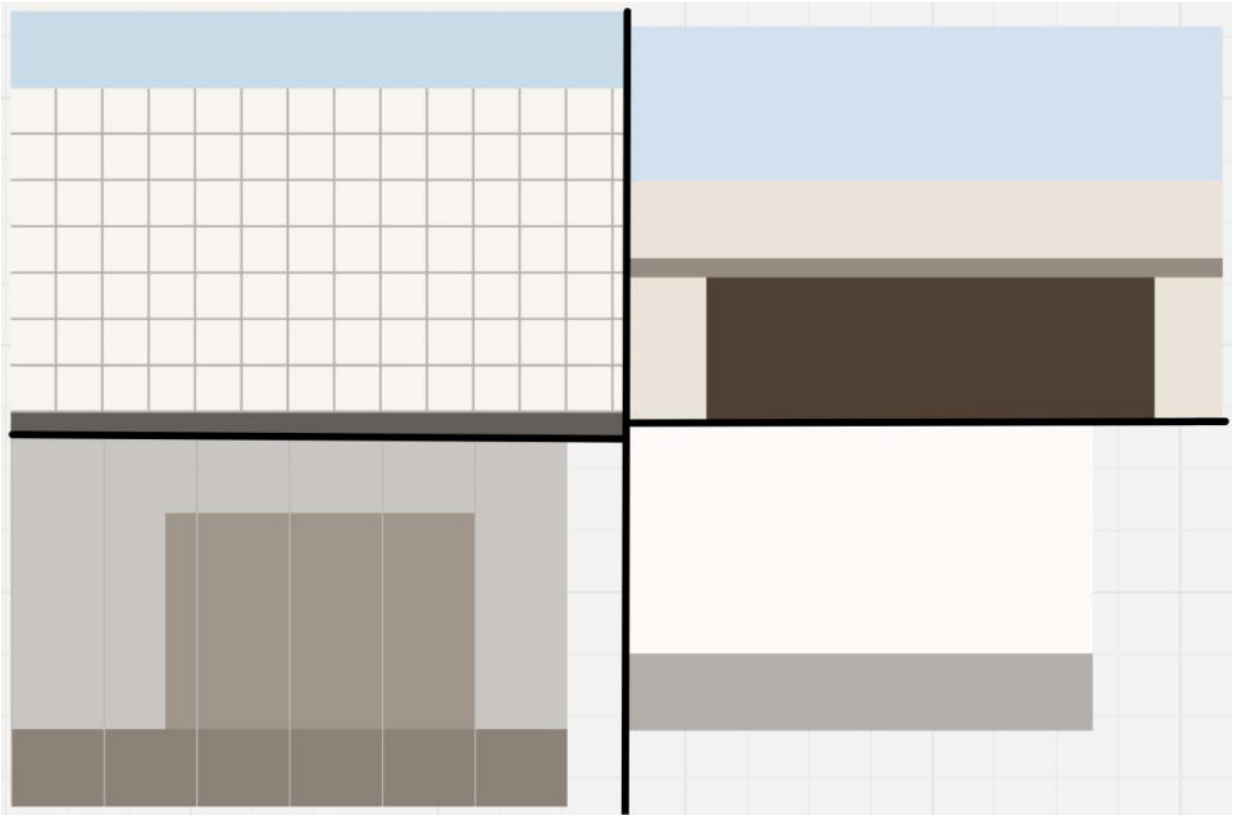


Рисунок 3.3 — Тестові набори 3D-рендерів, використані для апробації методу (джерело: сформовано автором на основі власних матеріалів)

3.4.5. Графічний інтерфейс користувача

Для взаємодії з системою реалізовано веб-інтерфейс на основі бібліотеки **Gradio**.

Інтерфейс складається з вкладок:

- **Overview** — загальні характеристики (мін/макс ентропія, кількість ROI);
- **Visual Analysis** — теплові карти та виділені аномалії;
- **RS Analysis** — таблиця статистичних результатів;
- **Extraction** — результати вилучення даних;
- **Chi-Square** — результати χ^2-аналізу;
- **Report** — JSON-звіт.

Користувач може:

- завантажити зображення;
- налаштувати параметри аналізу;
- отримати повний звіт та візуалізацію.

Головне вікно розробленої системи показано на рис. 3.4.

Результати аналізу при виявленні LSB-вбудовування наведено на рис. 3.5.

Приклад вилучення прихованого повідомлення подано на рис. 3.6.

Звіт системи представлено на рис. 3.7.

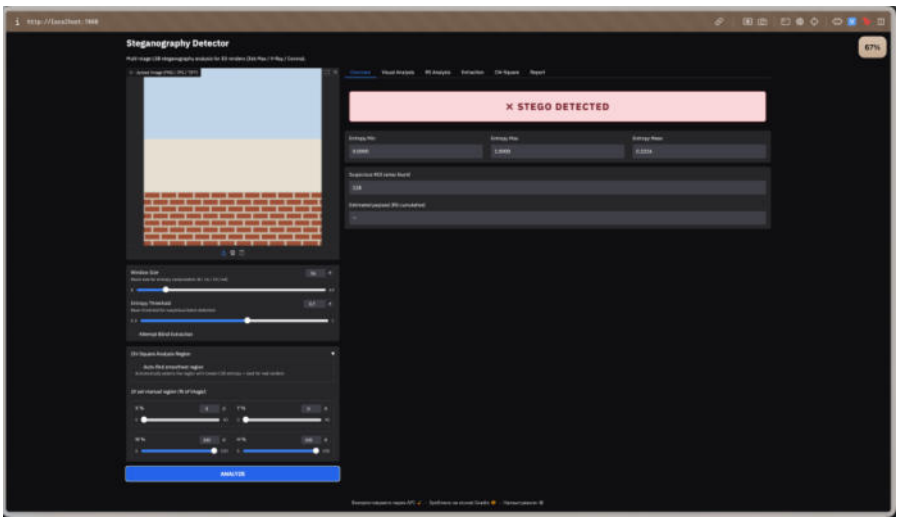


Рисунок 3.4 — Головне вікно розробленого програмного засобу у режимі попереднього перегляду (джерело: сформовано автором)

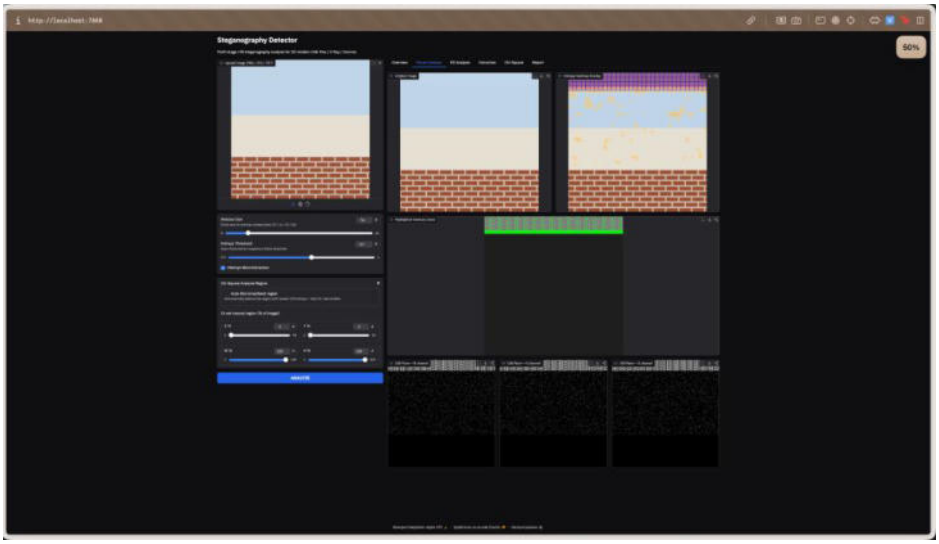


Рисунок 3.5 — Результати роботи комбінованого алгоритму при виявленні LSB-вбудовування (джерело: сформовано автором)

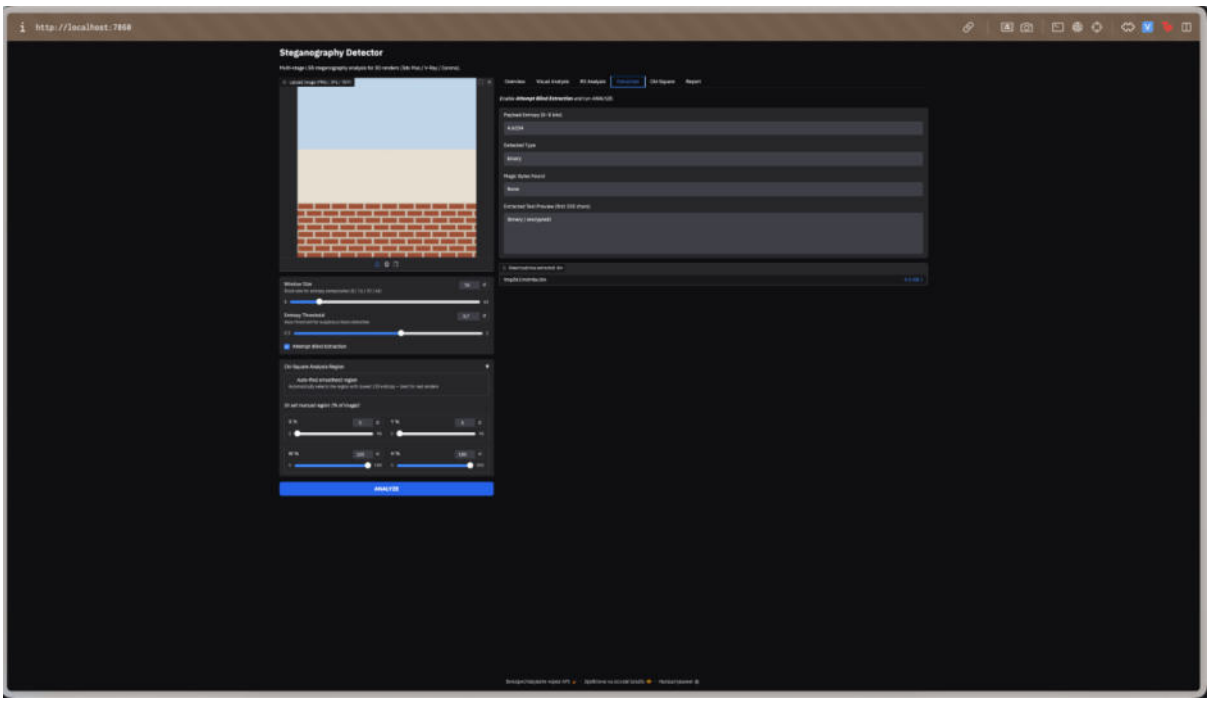


Рисунок 3.6 — Приклад візуалізації процесу вилучення прихованого повідомлення (джерело: сформовано автором)

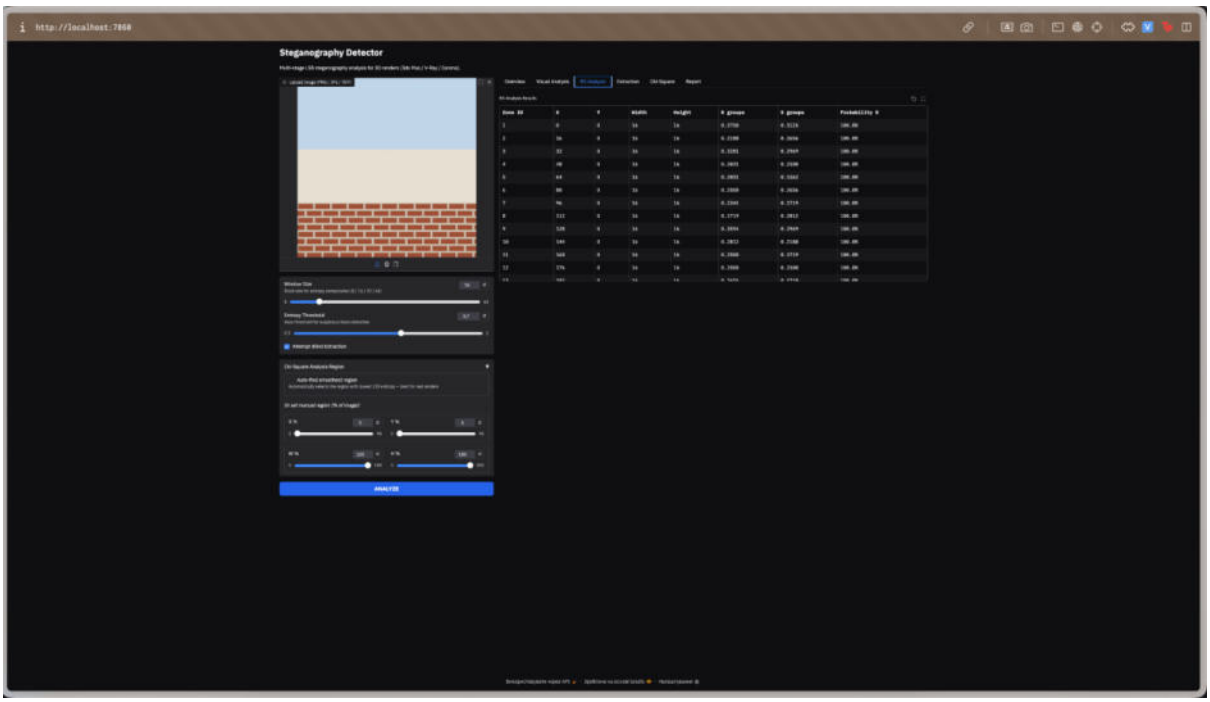


Рисунок 3.7 — Звіт системи про виявлення аномальних ознак та розраховану ймовірність вкладення (джерело: сформовано автором)

3.5 Висновки до розділу 3

У третьому розділі було розроблено комбінований метод виявлення та вилучення стеганографічних вкладень у синтетичних зображеннях високої роздільної здатності. В основу запропонованого підходу покладено припущення, що 3D-рендери, на відміну від цифрових фотографій, мають нижчий рівень природної стохастичності в молодших бітових площинах, що дає змогу ефективніше виявляти аномалії, спричинені LSB-вбудовуванням.

У ході дослідження сформульовано основні вимоги до методу, зокрема забезпечення високої ймовірності виявлення прихованих даних за низької щільності заповнення контейнера, зменшення кількості хибних спрацювань, здатність до локалізації підозрілих ділянок зображення та можливість подальшого сліпого вилучення вбудованої інформації. Обґрунтовано доцільність використання саме синтетичних рендерів як окремого класу контейнерів для стегоаналізу.

Розроблено структуру комбінованого методу, яка включає кілька послідовних етапів: декомпозицію зображення на бітові площини, побудову локальної ентропійної мапи, візуальне виділення зон з аномальними статистичними характеристиками, локальний RS-аналіз для математичної верифікації підозрілих ділянок, а також сліпе вилучення та первинну інтерпретацію корисного навантаження. Такий підхід дозволяє поєднати переваги швидкого попереднього скринінгу та поглибленої статистичної перевірки.

Показано, що застосування аналізу бітових площин і локального ентропійного мапування дає змогу істотно зменшити обсяг даних, для яких необхідно виконувати обчислювально складний RS-аналіз. Це підвищує швидкодію системи та робить можливим ефективний аналіз великих графічних файлів без втрати точності виявлення.

У межах розділу також реалізовано програмний засіб Steganography Detector, побудований на модульній архітектурі. До його складу включено окремі модулі для

декомпозиції бітових площин, побудови ентропійних карт, візуалізації аномалій, виконання RS- та χ^2-аналізу, вилучення прихованих даних і формування звітів. Для забезпечення зручності роботи користувача розроблено графічний веб-інтерфейс на основі бібліотеки Gradio.

У процесі практичної реалізації було уточнено особливості застосування χ^2-методу для синтетичних даних. Встановлено, що класичні порогові значення не забезпечують належної якості детекції для випадкового LSB-вбудовування, тому запропоновано адаптивну модель інтерпретації результатів, яка підвищує коректність класифікації контейнерів на чисті, підозрілі та такі, що містять приховані дані.

Отже, у третьому розділі не лише обґрунтовано теоретичні засади комбінованого методу стегоаналізу, а й реалізовано його у вигляді прикладного програмного засобу. Отримані результати підтверджують доцільність поєднання ентропійного аналізу, візуального виділення аномалій, RS-верифікації та сліпого вилучення даних для розв’язання задач виявлення прихованої інформації у синтетичних зображеннях.

ВИСНОВКИ

У кваліфікаційній роботі проведено дослідження методів виявлення та вилучення прихованої інформації, вбудованої у цифрові зображення з використанням стеганографії. За результатами виконаної роботи сформульовано такі висновки.

Проаналізовано сучасний стан розвитку стеганографії та стегоаналізу.

Встановлено, що цифрова стеганографія є одним із поширених способів прихованої передачі інформації, який може застосовуватися як із легітимною метою, зокрема для захисту авторських прав і цифрового маркування, так і для реалізації прихованих каналів обміну даними, витоку інформації та маскування шкідливої активності. У зв’язку з цим задача виявлення стеганографічних вкладень у графічних файлах є актуальною для сучасних систем кібербезпеки.

Досліджено особливості цифрових зображень як контейнерів для прихованого вбудовування інформації.

У роботі розглянуто специфіку природних цифрових фотографій та синтетичних зображень, отриманих засобами комп’ютерної графіки. Визначено, що структура зображення, характер шумової складової, кореляція між сусідніми пікселями та статистичні властивості молодших бітових площин безпосередньо впливають на ефективність як вбудовування, так і виявлення прихованих даних. Показано, що синтетичні зображення в окремих випадках можуть демонструвати вищу чутливість до статистичних змін, спричинених стеганографічним втручанням.

Обґрунтовано доцільність використання аналізу молодших значущих бітів для виявлення прихованих повідомлень.

Встановлено, що метод LSB-вбудовування є одним із найбільш доступних і поширених способів приховування інформації в зображеннях, а тому саме він є доцільним об’єктом дослідження в межах кваліфікаційної роботи. Аналіз ентропійних характеристик, структури бітових площин та статистичних відхилень

дозволяє виявляти ознаки стороннього втручання навіть у тих випадках, коли візуально зображення не зазнає помітних змін.

Розроблено алгоритмічне забезпечення для виявлення прихованої інформації в зображеннях.

У процесі виконання роботи запропоновано підхід до автоматизованого аналізу графічних контейнерів, який базується на дослідженні статистичних характеристик зображення, аналізі бітових площин та оцінюванні аномалій у молодших бітах. Реалізований алгоритм дає змогу виконувати попередню перевірку файлів на наявність стеганографічного вбудовування та формувати висновок щодо ймовірності прихованої модифікації контейнера.

Реалізовано засоби вилучення прихованих даних із досліджуваних зображень.

У межах практичної частини роботи передбачено механізми екстракції вбудованої інформації у випадку, якщо параметри приховування відповідають досліджуваному методу. Це дозволяє не лише зафіксувати факт наявності стеганограми, а й перейти до наступного етапу аналізу — вилучення та подальшої інтерпретації знайдених даних.

Проведено експериментальну перевірку запропонованих рішень.

Результати тестування підтвердили працездатність розробленого підходу для задач виявлення стеганографічних вкладень у цифрових зображеннях. У ході експериментів встановлено, що ефективність детекції залежить від формату файлу, характеру контейнера, обсягу прихованих даних і способу їх вбудовування. Найкращі результати досягаються для форматів без втрат, у яких зберігається початкова структура бітових площин.

Визначено практичне значення отриманих результатів.

Розроблені методи та програмні засоби можуть бути використані у сфері технічного захисту інформації, комп’ютерної криміналістики, діяльності

підрозділів кібербезпеки, а також у системах моніторингу та запобігання витоку даних. Отримані результати можуть слугувати основою для подальшого вдосконалення методів стегоаналізу, зокрема для розширення переліку підтримуваних форматів, ускладнених схем вбудовування та комбінованих способів аналізу.

Отже, поставлену в роботі мету досягнуто, а визначені завдання виконано в повному обсязі. Розроблені в межах дослідження підходи підтверджують доцільність використання статистичного аналізу цифрових зображень для виявлення та вилучення прихованої інформації, вбудованої стеганографічними методами.

Використані джерела

1. Fridrich J. *Steganography in Digital Media: Principles, Algorithms, and Applications.* Cambridge : Cambridge University Press, 2009. URL: https://www.cambridge.org/core/books/steganography-in-digital-media/2D5989F2462923428BE01F2042A532AB (дата звернення: 14.05.2026).
2. Provos N., Honeyman P. Hide and Seek: An Introduction to Steganography // *IEEE Security & Privacy.* 2003. Vol. 1, No. 3. P. 32–44. URL: https://www.researchgate.net/publication/3437465_Hide_and_seek_An_introduction_to_steganography (дата звернення: 22.05.2026).
3. Simmons G. J. The Prisoners' Problem and the Subliminal Channel // *Advances in Cryptology.* 1984. P. 51–67. DOI: https://doi.org/10.1007/978-1-4684-4730-9_5.
4. Cox I. J., Miller M. L., Bloom J. A. та ін. *Digital Watermarking and Steganography.* 2nd ed. Burlington : Morgan Kaufmann, 2008. URL: https://www.softouch.on.ca/kb/data/Digital%20Watermarking%20and%20Steganography%202E.pdf (дата звернення: 05.05.2026).
5. ISO/IEC 27001:2022. *Information security, cybersecurity and privacy protection — Information security management systems — Requirements.* Geneva : ISO, 2022. URL: https://www.iso.org/isoiec-27001-information-security.html (дата звернення: 29.05.2026).
6. ISO/IEC 15948:2004. *Information technology — Computer graphics and image processing — Portable Network Graphics (PNG): Functional specification.* URL: https://www.w3.org/TR/PNG/ (дата звернення: 11.05.2026).
7. ISO/IEC 10918-1:1994. *Information technology — Digital compression and coding of continuous-tone still images — Requirements and guidelines.* URL: https://www.w3.org/Graphics/JPEG/itu-t81.pdf (дата звернення: 18.05.2026).
8. Adobe Systems Incorporated. *TIFF Revision 6.0.* 1992. URL: https://download.osgeo.org/libtiff/doc/TIFF6.pdf (дата звернення: 25.05.2026).

9. MITRE ATT&CK. T1001.002 Data Obfuscation: Steganography [Електронний ресурс]. URL: https://attack.mitre.org/techniques/T1001/002/ (дата звернення: 02.05.2026).
10. Microsoft. STRIDE threat model [Електронний ресурс]. URL: https://learn.microsoft.com/en-us/azure/security/develop/threat-modeling-tool-threats (дата звернення: 16.05.2026).
11. CERT Insider Threat Center [Електронний ресурс]. URL: https://www.sei.cmu.edu/library/cert-insider-threat-center/ (дата звернення: 30.05.2026).
12. Chaos. Corona for 3ds Max documentation [Електронний ресурс]. URL: https://corona-renderer.com/doc (дата звернення: 08.05.2026).
13. Chaos. V-Ray GPU documentation [Електронний ресурс]. URL: https://documentation.chaos.com/space/VBLD/117638321/Rendering+with+V-Ray+GPU (дата звернення: 20.05.2026).
14. Autodesk. Arnold documentation [Електронний ресурс]. URL: https://help.autodesk.com/view/ARNOL/ENU/ (дата звернення: 13.05.2026).
15. NVIDIA AI Denoiser [Електронний ресурс]. URL: https://developer.nvidia.com/optix-denoiser (дата звернення: 27.05.2026).
16. Intel Open Image Denoise [Електронний ресурс]. URL: https://www.openimagedenoise.org/ (дата звернення: 04.05.2026).
17. Autodesk. 3ds Max 2025 [Електронний ресурс]. URL: https://help.autodesk.com/view/3DSMAX/2025/ENU/ (дата звернення: 19.05.2026).
18. Westfeld A., Pfitzmann A. Attacks on Steganographic Systems // *Information Hiding.* Lecture Notes in Computer Science. Vol. 1768. Berlin ; Heidelberg : Springer, 2000. P. 61–76. DOI: 10.1007/10719724_5.
19. Liu Y., Liu S., Wang Y. та ін. Video steganography: A review // *Neurocomputing.* 2019. Vol. 335. P. 238–250. URL: https://www.sciencedirect.com/science/article/abs/pii/S0925231218312608 (дата звернення: 10.05.2026).

20. Foley J. D., van Dam A., Feiner S. K., Hughes J. F. *Computer Graphics: Principles and Practice.* 3rd ed. Addison-Wesley, 2013. URL: https://books.google.com/books?q=Computer+Graphics+Principles+and+Practice+Foley+van+Dam+Feiner+Hughes (дата звернення: 31.05.2026).
21. Whitted T. An improved illumination model for shaded display // *Communications of the ACM.* 1980. Vol. 23, No. 6. P. 343–349. DOI: https://doi.org/10.1145/358876.358882.
22. Kajiya J. T. The rendering equation // *ACM SIGGRAPH Computer Graphics.* 1986. Vol. 20, No. 4. P. 143–150. DOI: https://dl.acm.org/doi/10.1145/15922.15902.
23. NVIDIA. OptiX Programming Guide [Електронний ресурс]. URL: https://raytracing-docs.nvidia.com/optix9/guide/index.html (дата звернення: 06.05.2026).
24. ArchiCGI. 3D Visualization Portfolio [Електронний ресурс]. URL: https://archicgi.com/3d-visualization-portfolio/ (дата звернення: 23.05.2026).
25. Chaos. Chaos Vantage — System Requirements [Електронний ресурс]. URL: https://docs.chaos.com/display/LAV/System+Requirements (дата звернення: 15.05.2026).
26. NVIDIA. Chaos Vantage 2 Powered by NVIDIA RTX [Електронний ресурс]. URL: https://www.chaos.com/vantage (дата звернення: 09.05.2026).
27. ISO/IEC 27002:2022. *Information security, cybersecurity and privacy protection — Information security controls.* Geneva : ISO, 2022. URL: https://www.scribd.com/document/930133695/ISO-IEC-27002-2022-UKR (дата звернення: 28.05.2026).
28. ISO 12639:2004. *Graphic technology — Prepress digital data exchange — Tag image file format for image technology (TIFF/IT).* URL: https://www.loc.gov/preservation/digital/formats/fdd/fdd000022.shtml (дата звернення: 12.05.2026).
29. Fridrich J., Kodovský J. Rich Models for Steganalysis of Digital Images // *IEEE Transactions on Information Forensics and Security.* 2012. Vol. 7, No. 3. P. 868–

882. URL: http://dde.binghamton.edu/kodovsky/pdf/TIFS2012-SRM.pdf (дата звернення: 03.05.2026).

30. Xu G., Wu H. Z., Shi Y. Q. Structural design of convolutional neural networks for steganalysis // *IEEE Signal Processing Letters.* 2016. Vol. 23, No. 5. P. 708–712. DOI: 10.1109/LSP.2016.2548421. URL: https://researchwith.njit.edu/en/publications/structural-design-of-convolutional-neural-networks-for-steganalys/ (дата звернення: 24.05.2026).
31. Xu X., Sun Y., Tang G., Chen S., Zhao J. Deep Learning on Spatial Rich Model for Steganalysis // *Digital Forensics and Watermarking (IWDW 2016).* Lecture Notes in Computer Science. Vol. 10082. Springer, 2017. P. 564–577. DOI: https://doi.org/10.1007/978-3-319-53465-7_42. URL: https://link.springer.com/chapter/10.1007/978-3-319-53465-7_42 (дата звернення: 17.05.2026).
32. Tabares-Soto R., Arteaga-Arteaga H. B., Mora-Rubio A. та ін. Strategy to improve the accuracy of convolutional neural network architectures applied to digital image steganalysis in the spatial domain // *PeerJ Computer Science.* 2021. Vol. 7. Article e451. DOI: https://doi.org/10.7717/peerj-cs.451. URL: https://doi.org/10.7717/peerj-cs.451 (дата звернення: 01.06.2026).
33. Katzenbeisser S., Petitcolas F. A. P. *Information Hiding Techniques for Steganography and Digital Watermarking.* Boston ; London : Artech House, 2000. URL: https://books.google.com/books?q=Information+Hiding+Techniques+for+Steganography+and+Digital+Watermarking+Katzenbeisser+Petitcolas (дата звернення: 21.05.2026).
34. Gonzalez R. C., Woods R. E. *Digital Image Processing.* 4th ed. Harlow : Pearson, 2018. URL: https://books.google.com/books?q=Digital+Image+Processing+Gonzalez+Woods+4th+ed (дата звернення: 11.05.2026).
35. Cheddad A., Condell J., Curran K., Mc Kevitt P. Digital image steganography: Survey and analysis of current methods // *Signal Processing.* 2010. Vol. 90, No.

3. P. 727–752. URL: https://www.sciencedirect.com/science/article/pii/S0165168409003648 (дата звернення: 07.05.2026).

36. Lyu S., Farid H. Steganalysis using higher-order image statistics // *IEEE Transactions on Information Forensics and Security.* 2006. Vol. 1, No. 1. P. 111–119. URL: https://ieeexplore.ieee.org/document/1610077 (дата звернення: 26.05.2026).

37. Kessler G. C. An Overview of Steganography for the Computer Forensics Examiner // *Forensic Science Communications.* 2004. Vol. 6, No. 3. URL: https://www.garykessler.net/library/fsc_stego.html (дата звернення: 14.05.2026).

38. Johnson N. F., Jajodia S. Exploring Steganography: Seeing the Unseen // *Computer.* 1998. Vol. 31, No. 2. P. 26–34. URL: https://ieeexplore.ieee.org/document/4655281 (дата звернення: 19.05.2026).

39. Provos N. Defending Against Statistical Steganalysis // *Proceedings of the 10th USENIX Security Symposium.* 2001. URL: https://www.usenix.org/conference/10th-usenix-security-symposium/defending-against-statistical-steganalysis (дата звернення: 29.05.2026).

40. Westfeld A. Detecting Low Embedding Rates // *Information Hiding.* Lecture Notes in Computer Science. Vol. 2578. Berlin ; Heidelberg : Springer, 2003. P. 324–339. URL: https://link.springer.com/chapter/10.1007/3-540-36415-3_21 (дата звернення: 04.05.2026).

Додаток А

detector/chi_square.py

```
from dataclasses import dataclass, field
from typing import Dict, List, Optional, Tuple
import numpy as np
from scipy import stats as scipy_stats
# ── Result container ──────────────────────────────────────────────────────────

@dataclass
class ChiSquareResult:
    channel:      str
    chi2_stat:    float
    p_value:    float       # probability that steganography IS present
    degrees_of_freedom: int
    pairs_tested: int         # number of (2k, 2k+1) pairs with enough data
    verdict:     str           # 'stego', 'suspicious', or 'clean'
    embedding_rate: float      # estimated fraction of LSBs modified (0–1)

# ── Core computation ──────────────────────────────────────────────────────────

def _chi_square_channel(
    channel: np.ndarray,
    min_pair_count: int = 5,
) -> ChiSquareResult:
    """Run chi-square attack on a single 2-D uint8 channel array."""
    label = "ch"   # caller sets .channel after construction
    flat   = channel.ravel().astype(np.int64)
    counts = np.bincount(flat, minlength=256)   # shape (256,)
    # Pair expected frequencies: ê_k = (n_{2k} + n_{2k+1}) / 2
    n_even  = counts[0::2]   # indices 0, 2, 4, …, 254   → shape
(128,)
    n_odd   = counts[1::2]   # indices 1, 3, 5, …, 255   → shape
(128,)
    expected = (n_even + n_odd) / 2.0
    # Only test pairs where both values appear (avoid divide-by-zero)
    valid = expected >= min_pair_count
    n_pairs = int(valid.sum())
    if n_pairs < 10:
        return ChiSquareResult(
            channel="?", chi2_stat=0.0, p_value=0.5,
            degrees_of_freedom=0, pairs_tested=0,
```

```
            verdict="unknown", embedding_rate=0.0,
        )
    chi2 = float(np.sum(
        (n_even[valid] - expected[valid]) ** 2 / expected[valid]
        + (n_odd[valid]  - expected[valid]) ** 2 / expected[valid]
    ))
    # df = n_pairs because expected is derived from the data (1 free parameter per pair)
    df = n_pairs
    # Survival function: P(X ≥ chi2) where X ~ χ²(df)
    # Large chi2 → small p-value → pairs are NOT equalised → CLEAN
    # Small chi2 → large p-value → pairs ARE equalised     → STEGO
    p_value = float(scipy_stats.chi2.sf(chi2, df=df))
    # Estimate embedding rate from pair imbalance
    imbalances = np.abs(n_even[valid] - n_odd[valid]) / (n_even[valid] + n_odd[valid] + 1e-9)
    embedding_rate = float(1.0 - imbalances.mean())  # 0 = clean, 1 = fully embedded
    # Thresholds are calibrated for random-bit LSB replacement:
    # Full random embedding gives chi2 ≈ df → p ≈ 0.5.
    # Clean image gives chi2 >> df → p ≈ 0.
    # p > 0.20: chi2 is in the lower ~80% of chi2(df) — consistent with equalized pairs.
    if p_value > 0.20:
        verdict = "stego"
    elif p_value > 0.05:
        verdict = "suspicious"
    else:
        verdict = "clean"
    return ChiSquareResult(
       channel=label, chi2_stat=round(chi2, 2), p_value=round(p_value, 4),
        degrees_of_freedom=df, pairs_tested=n_pairs,
        verdict=verdict, embedding_rate=round(embedding_rate, 4),
    )


# ── Public API ──────────────────────────────────────────────────────────

def chi_square_attack(
    image: np.ndarray,
    roi: Optional[Tuple[int, int, int, int]] = None,
) -> Dict[str, ChiSquareResult]:
    """
    Run the chi-square attack on all colour channels.
    Args:
        image: uint8 RGB or RGBA array (H×W×C).
```

```
        roi:  Optional (x0, y0, x1, y1) pixel crop.  Full image if None.
    Returns:
        Dict mapping channel name ('R', 'G', 'B') → ChiSquareResult.
    """
    if roi is not None:
        x0, y0, x1, y1 = roi
        region = image[y0:y1, x0:x1]
    else:
        region = image
    labels   = ["R", "G", "B"]
    n_ch     = min(region.shape[2], 3) if region.ndim == 3 else 1
    results  = {}
    for i in range(n_ch):
        ch_arr = region[:, :, i] if region.ndim == 3 else region
        res    = _chi_square_channel(ch_arr)
        res.channel = labels[i]
        results[labels[i]] = res
    return results
def overall_verdict(results: Dict[str, ChiSquareResult]) -> str:
    """Aggregate per-channel verdicts into a single verdict."""
    verdicts = [r.verdict for r in results.values()]
    if verdicts.count("stego") >= 2:
        return "STEGO DETECTED"
    if "stego" in verdicts or verdicts.count("suspicious") >= 2:
        return "SUSPICIOUS"
    return "CLEAN"
def find_smoothest_roi(
    image: np.ndarray,
    window_size: int = 16,
    roi_blocks: int = 8,
) -> Tuple[int, int, int, int]:
    """
    Automatically locate the smoothest (lowest LSB entropy) rectangular
    region of roi_blocks × roi_blocks blocks in the image.
    Returns (x0, y0, x1, y1) pixel coordinates.
    """
    from detector.entropy_map import compute_entropy_map
    H, W = image.shape[:2]
    n_ch = min(image.shape[2], 3) if image.ndim == 3 else 1
    # Average entropy map across channels
    combined = np.zeros(
        (H // window_size, W // window_size), dtype=np.float32
    )
    for i in range(n_ch):
        ch  = image[:, :, i] if image.ndim == 3 else image
```

```
        lsb = (ch & 1).astype(np.uint8)
        combined += compute_entropy_map(lsb, window_size)
    combined /= n_ch
    rows, cols = combined.shape
    rb = min(roi_blocks, rows)
    cb = min(roi_blocks, cols)
    # Sliding window sum to find lowest-entropy patch
    best_score = np.inf
    best_r, best_c = 0, 0
    for r in range(rows - rb + 1):
        for c in range(cols - cb + 1):
            score = combined[r:r + rb, c:c + cb].mean()
            if score < best_score:
                best_score = score
                best_r, best_c = r, c
    x0 = best_c * window_size
    y0 = best_r * window_size
    x1 = min(x0 + cb * window_size, W)
    y1 = min(y0 + rb * window_size, H)
    return x0, y0, x1, y1
```

detector/rs_analysis.py

```
from dataclasses import dataclass
from typing import List, Optional, Tuple
import numpy as np


# ── Flipping functions ──────────────────────────────────────────────
def flip_positive(x: int) -> int:
    """F_1: toggle the least significant bit.  F_1(x) = x XOR 1."""
    return x ^ 1
def flip_negative(x: int) -> int:
    """
    F_{-1}: shift-one negative flip.
    For x ≥ 1:  F_{-1}(x) = ((x - 1) XOR 1) + 1
    Clamped at 0 and 255 for uint8 boundary safety.
    Verification:
    F_{-1}(1) = (0 ^ 1) + 1 = 2  ✓ pair {1,2}
    F_{-1}(2) = (1 ^ 1) + 1 = 1  ✓
    F_{-1}(3) = (2 ^ 1) + 1 = 4  ✓ pair {3,4}
    F_{-1}(4) = (3 ^ 1) + 1 = 3  ✓
    """
```

```
    if x == 0:
        return 0      # -1 is invalid; no-op at lower boundary
    result = ((x - 1) ^ 1) + 1
    return min(result, 255)   # clamp at upper boundary
# ── Discriminant function ──────────────────────────────────────────
_
def discriminant(group: np.ndarray) -> float:
    """
    Measure the "roughness" of a 1-D pixel group.
    f(G) = Σ |G_i - G_{i+1}|  for adjacent pairs.
    Higher f = rougher (less smooth). Embedding typically increases f for
    previously smooth groups (making them Regular) and decreases f for
    previously rough groups (making them Singular).
    Args:
        group: 1-D int32 array.
    Returns:
        Sum of absolute first-differences (float).
    """
    return float(np.sum(np.abs(np.diff(group))))
# ── Mask application and classification ────────────────────────────

# Standard RS mask of length 4 (from the original paper)
_RS_MASK_POS = np.array([0,  1,  1, 0], dtype=np.int8)
_RS_MASK_NEG = np.array([0, -1, -1, 0], dtype=np.int8)

def apply_mask(group: np.ndarray, mask: np.ndarray) -> np.ndarray:
    """
    Apply flipping operations to a pixel group according to a mask.
    mask[i] =  1 → apply F_1   (positive flip)
    mask[i] = -1 → apply F_{-1}  (negative flip)
    mask[i] =  0 → leave unchanged
    Args:
        group: 1-D uint8 array (will be converted to int32 internally).
        mask:  1-D int8 mask of same length.
    Returns:
        Flipped 1-D uint8 array.
    """
    result = group.astype(np.int32).copy()
    for i, m in enumerate(mask):
        if m == 1:
            result[i] = flip_positive(result[i])
        elif m == -1:
            result[i] = flip_negative(result[i])
    return result.astype(np.uint8)
```

```
def classify_group(group: np.ndarray, mask: np.ndarray) -> str:
    """
    Classify a pixel group as Regular (R), Singular (S), or Unusable (U).
    Comparison:
        f(F_M(G)) > f(G)  → Regular    (flipping increases
roughness)
        f(F_M(G)) < f(G)  → Singular   (flipping decreases
roughness)
        f(F_M(G)) = f(G)  → Unusable
    Args:
        group: 1-D uint8 pixel group.
        mask:  1-D mask with values in {-1, 0, 1}.
    Returns:
        'R', 'S', or 'U'.
    """
    g_int = group.astype(np.int32)
    f_orig = discriminant(g_int)
    flipped = apply_mask(group, mask).astype(np.int32)
    f_flip = discriminant(flipped)
    if f_flip > f_orig:
        return 'R'
    if f_flip < f_orig:
        return 'S'
    return 'U'
# ── Region-level RS statistics
def rs_analyze_region(
    region: np.ndarray,
    group_size: int = 4,
) -> Tuple[float, float, float, float]:
    """
    Compute R_M, S_M, R_{-M}, S_{-M} statistics for a pixel region.
    The region is flattened and partitioned into non-overlapping groups of
    *group_size* pixels (default 4).  Each group is classified under the
    positive mask M = [0, 1, 1, 0] and the negative mask -M = [0, -1, -1, 0].
    Args:
        region:     2-D single-channel uint8 array.
        group_size: Pixels per group (must match mask length; default 4).
    Returns:
        (R_M, S_M, R_{-M}, S_{-M}) as normalised floats in [0, 1].
        Returns (0, 0, 0, 0) if the region is too small.
    """
    flat = region.ravel()
    n = len(flat)
    if n < group_size:
```

```
        return 0.0, 0.0, 0.0, 0.0
    mask_pos = _RS_MASK_POS[:group_size]
    mask_neg = _RS_MASK_NEG[:group_size]
    rm = sm = rm_neg = sm_neg = 0
    total = 0
    for i in range(0, n - group_size + 1, group_size):
        g = flat[i:i + group_size]
        c_pos = classify_group(g, mask_pos)
        if c_pos == 'R':
            rm += 1
        elif c_pos == 'S':
            sm += 1
        c_neg = classify_group(g, mask_neg)
        if c_neg == 'R':
            rm_neg += 1
        elif c_neg == 'S':
            sm_neg += 1
        total += 1
    if total == 0:
        return 0.0, 0.0, 0.0, 0.0
    return rm / total, sm / total, rm_neg / total, sm_neg / total
# ── Payload estimation via quadratic equation
def estimate_payload_fraction(
    rm: float, sm: float, rm_neg: float, sm_neg: float,
) -> float:
    """
    Estimate the fraction of LSBs modified using the RS quadratic formula.
    From Fridrich et al., the estimated embedding rate p satisfies:
        2(d_1 + d_0) p² - (2d_1 + d_0) p + d_1 = 0
    where:
        d_0 = R_M  - S_M      (regular-singular difference, positive mask)
        d_1 = R_{-M} - S_{-M}  (regular-singular difference, negative mask)
    The physical fraction of hidden *bits* is p / 2.
    Args:
        rm, sm:         R_M, S_M from rs_analyze_region.
        rm_neg, sm_neg: R_{-M}, S_{-M} from rs_analyze_region.
    Returns:
        Estimated LSB modification fraction in [0.0, 1.0].
    """
    d0 = rm - sm
    d1 = rm_neg - sm_neg
    a = 2.0 * (d1 + d0)
    b = -(2.0 * d1 + d0)
    c = d1
    # Degenerate: linear equation
```

```
    if abs(a) < 1e-10:
        if abs(b) < 1e-10:
            return 0.0
        p = float(np.clip(-c / b, 0.0, 1.0))
        return p
    disc = b * b - 4.0 * a * c
    if disc < 0:
        return 0.0
    sq = np.sqrt(disc)
    roots = [(-b + sq) / (2 * a), (-b - sq) / (2 * a)]


    # Keep only physically meaningful roots (0 ≤ p ≤ 1) and prefer
smaller
    valid = [r for r in roots if 0.0 <= r <= 1.0]
    if not valid:
        return 0.0
    # The smaller root corresponds to partial embedding; pick closest to 0.5
    return float(min(valid, key=lambda x: abs(x - 0.5)))
# ── Detection probability
def detection_probability(
    rm: float, sm: float, rm_neg: float, sm_neg: float,
) -> float:
    """
    Estimate the probability that this ROI contains steganographic content.
    In a clean image R_M ≈ R_{-M} (and S_M ≈ S_{-M}).  After
embedding,
    R_M drops below R_{-M} and S_M rises above S_{-M}.
   Score = |R_M - R_{-M}| + |S_M - S_{-M}| + max(0, (R_{-M} - R_M) + (S_M - S_{-M}))


    The directional term (R_{-M} - R_M) + (S_M - S_{-M}) rewards the
   *expected* direction of divergence after embedding, reducing false positives from noise.
    The raw score is passed through a sigmoid centred at 0.07 with slope 20,
    giving probability ≈ 0.5 at the empirical clean/stego boundary.
    Args:
        rm, sm:         RS stats under positive mask.
        rm_neg, sm_neg: RS stats under negative mask.
    Returns:
        Detection probability in [0.0, 1.0].
    """
    r_div = abs(rm - rm_neg)
    s_div = abs(sm - sm_neg)
    # Directional asymmetry: positive when R_{-M} > R_M (hallmark of embedding)
    directional = max(0.0, (rm_neg - rm) + (sm - sm_neg))
```

```
    raw = r_div + s_div + directional
    # Sigmoid normalisation: P → 1 as raw → ∞, P ≈ 0.5 at raw = 0.07
    prob = 1.0 / (1.0 + np.exp(-20.0 * (raw - 0.07)))
    return float(np.clip(prob, 0.0, 1.0))
# ── Result container ──────────────────────────────────────────────
@dataclass
class RSResult:
    """All RS-analysis outputs for one ROI."""
    roi: Tuple[int, int, int, int]   # (x0, y0, x1, y1)
    channel: int                # Channel index analysed (0=R,1=G,2=B)
    rm:      float
    sm:      float
    rm_neg:  float
    sm_neg:  float
    payload_fraction:    float
    detection_probability: float
    estimated_payload_bits:  int
    estimated_payload_bytes: int
# ── Single ROI analysis ──────────────────────────────────────────
def analyze_roi(
    image: np.ndarray,
    roi: Tuple[int, int, int, int],
    channel_idx: int = 0,
) -> RSResult:
    """
    Run RS-analysis for one ROI and one channel.
    Args:
        image:       Full image array (H×W×C or H×W).
        roi:         (x0, y0, x1, y1) pixel coordinates.
        channel_idx: Channel to analyse (0=R, 1=G, 2=B).
    Returns:
        RSResult dataclass with all statistics.
    """
    x0, y0, x1, y1 = roi
    region = (
        image[y0:y1, x0:x1, channel_idx]
        if image.ndim == 3 else image[y0:y1, x0:x1]
    )
    rm, sm, rm_neg, sm_neg = rs_analyze_region(region)
    frac  = estimate_payload_fraction(rm, sm, rm_neg, sm_neg)
    prob  = detection_probability(rm, sm, rm_neg, sm_neg)
    total_pixels = (x1 - x0) * (y1 - y0)
    payload_bits  = int(frac * total_pixels)
    payload_bytes = payload_bits // 8
```

```
    return RSResult(
        roi=roi,
        channel=channel_idx,
        rm=rm, sm=sm, rm_neg=rm_neg, sm_neg=sm_neg,
        payload_fraction=frac,
        detection_probability=prob,
        estimated_payload_bits=payload_bits,
        estimated_payload_bytes=payload_bytes,
    )
# ── Batch analysis ──────────────────────────────────────────────
def run_rs_analysis(
    image: np.ndarray,
    suspicious_rois: List[Tuple[int, int, int, int]],
    channels: Optional[List[int]] = None,
) -> List[RSResult]:
    """
    Run RS-analysis on all suspicious ROIs, selecting the most informative
    channel per ROI.
    For each ROI, all requested channels are analysed independently.  The
    result with the highest detection probability is returned for that ROI.
    Args:
        image:           Full image array.
        suspicious_rois: List of (x0, y0, x1, y1) from Stage 2.
        channels:     Channel indices to try (default: [0, 1, 2] or fewer
                         if the image has fewer channels).
    Returns:
        List of RSResult objects, one per ROI, ordered as *suspicious_rois*.
    """
    if not suspicious_rois:
        return []
    n_ch = image.shape[2] if image.ndim == 3 else 1
    if channels is None:
        channels = list(range(min(n_ch, 3)))
    results: List[RSResult] = []
    for roi in suspicious_rois:
        best: Optional[RSResult] = None
        for ch in channels:
            r = analyze_roi(image, roi, ch)
          if best is None or r.detection_probability > best.detection_probability:
                best = r
        if best is not None:
            results.append(best)
    return results
```

detector/entropy_map.py

```
import numpy as np
from typing import Dict, List, Tuple
# ── Block-level entropy def block_entropy(block: np.ndarray) -> float:
    """
    Compute Shannon entropy for a 2-D block of binary (0/1) values.
  H = -Σ p_i · log₂(p_i) over {0, 1}.
    Returns 0.0 for uniform blocks (all 0 or all 1) and 1.0 for perfectly
    balanced blocks (equal number of 0s and 1s).
    Args:
        block: 2-D array of 0s and 1s.
    Returns:
        Entropy in [0.0, 1.0].
    """
    flat = block.ravel()
    n = flat.size
    if n == 0:
        return 0.0
    ones = int(np.sum(flat))
    zeros = n - ones
    if ones == 0 or zeros == 0:
        return 0.0
    p1 = ones / n
    p0 = zeros / n
    return float(-(p0 * np.log2(p0) + p1 * np.log2(p1)))
# ── Entropy map
──────────────────────────────────────────────────────────────def
compute_entropy_map(lsb_plane: np.ndarray, window_size: int = 16) ->
np.ndarray:
    """
    Build a low-resolution entropy map from an LSB plane.
   The image is divided into non-overlapping blocks of *window_size × window_size*
    pixels.  Shannon entropy is computed for each block and stored in the
    corresponding cell of the output map.


    Output dimensions: (H // window_size, W // window_size).
    Partial border blocks (when H or W is not divisible) are discarded.
    Args:
        lsb_plane:  Binary 2-D array (output of bitplane.get_lsb_plane()).
        window_size: Block edge length in pixels (default 16).
    Returns:
        2-D float32 array with entropy values in [0.0, 1.0].
    """
    H, W = lsb_plane.shape
```

```
    n_rows = H // window_size
    n_cols = W // window_size
    entropy_map = np.zeros((n_rows, n_cols), dtype=np.float32)
    for r in range(n_rows):
        y0 = r * window_size
        for c in range(n_cols):
            x0 = c * window_size
           block = lsb_plane[y0:y0 + window_size, x0:x0 + window_size]
            entropy_map[r, c] = block_entropy(block)


    return entropy_map
# ── Texture map ──────────────────────────────────────────────────────def
compute_texture_map(channel: np.ndarray, window_size: int = 16) ->
np.ndarray:
    """
    Estimate local texture complexity via per-block pixel-value standard deviation.
    Pixel standard deviation is preferred over Sobel gradient variance here
    because it correctly captures *any* locally variable region — including
    flat-but-noisy areas (e.g. mortar with ±15 random noise) that appear smooth
    to edge detectors but whose LSBs are naturally high-entropy.
    A block with σ ≥ 2 has roughly equal numbers of even and odd pixel values,
    so its natural LSB entropy approaches 1.0 regardless of whether it contains
    edges.  Using σ as the texture measure lets the adaptive threshold correctly
    exclude such blocks from suspicion.


        texture(block) = σ(pixel values)   normalised globally to [0, 1]
    Boundary values:
    σ ≈ 0   → perfectly flat block (single pixel value) → texture = 0
    σ = max → most variable block in the image          → texture = 1
    Args:
        channel:     Single 2-D uint8 channel array.
        window_size: Block size matching the entropy map.
    Returns:
        2-D float32 array in [0, 1].
    """
    H, W = channel.shape
    n_rows = H // window_size
    n_cols = W // window_size
    texture_map = np.zeros((n_rows, n_cols), dtype=np.float32)
    for r in range(n_rows):
        y0 = r * window_size
        for c in range(n_cols):
            x0 = c * window_size
```

```
            block = channel[y0:y0 + window_size, x0:x0 + window_size]
            texture_map[r, c] = float(np.std(block))
    # Normalise globally: 0 = perfectly flat, 1 = most variable block
    max_val = texture_map.max()
    if max_val > 0:
        texture_map /= max_val
    return texture_map
# ── Adaptive threshold
def compute_adaptive_threshold(
    texture_map: np.ndarray,
    base_threshold: float = 0.7,
    smooth_penalty: float = 0.25,
    texture_bonus: float = 0.28,
) -> np.ndarray:
    """
    Compute a per-block adaptive entropy threshold.
    Smooth zones (texture → 0) have a lower threshold because natural LSBs
    there are nearly constant and any randomness is suspicious.  Textured
    zones (texture → 1) receive a threshold near 1.0 because their natural
    LSB entropy is already high — blocks that exceed that ceiling are excluded
    entirely by *find_suspicious_blocks*.
    Equation (linear interpolation):
        T(x,y) = base - smooth_penalty + (smooth_penalty + texture_bonus)·texture
    Boundary values:
        texture = 0  →  T = base - smooth_penalty  (smooth floor, e.g. 0.45)
        texture = 1  →  T = base + texture_bonus   (textured ceiling, e.g. 0.98)
    Clamped to [0.30, 0.99].
    Args:
        texture_map:    Normalised texture map from compute_texture_map().
        base_threshold:  Central threshold value (default 0.7).
        smooth_penalty:  Amount to lower T in smooth zones (default 0.25).
        texture_bonus:   Amount to raise T in textured zones (default 0.28).
    Returns:
        2-D float32 adaptive threshold map, same shape as texture_map.
    """
    adaptive = (
        base_threshold
        - smooth_penalty * (1.0 - texture_map)
        + texture_bonus * texture_map
    )
```

```
    return np.clip(adaptive, 0.30, 0.99).astype(np.float32)
# ── Suspicious block detection ──────────────────────────────────────────────
def find_suspicious_blocks(
    entropy_map: np.ndarray,
    adaptive_threshold: np.ndarray,
    texture_map: np.ndarray = None,
    max_texture: float = 0.55,
) -> List[Tuple[int, int]]:
    """
    Return block indices that are anomalously high-entropy for their texture class.
    A block is suspicious when:
      1.  Its entropy exceeds the adaptive threshold AND
      2.  Its texture score is below *max_texture*.
    Condition 2 excludes naturally high-entropy textured areas (brick, fabric,
    wood grain) that would generate false positives.  Real-world stego tools
    preferentially target smooth regions precisely because noise there is
    detectable.  For highly textured blocks the signal is completely buried.
    Args:
        entropy_map:       2-D entropy map from compute_entropy_map().
        adaptive_threshold: 2-D threshold map from compute_adaptive_threshold().
        texture_map:       Optional normalised texture map.  When provided,
                           blocks with texture ≥ max_texture are excluded.
        max_texture:       Texture exclusion ceiling (default 0.55).
    Returns:
        List of (row, col) integer tuples.
    """
    mask = entropy_map > adaptive_threshold
    if texture_map is not None:
        # Exclude naturally high-entropy textured blocks
        mask &= (texture_map < max_texture)
    return [(int(r), int(c)) for r, c in np.argwhere(mask)]

def blocks_to_pixel_rois(
    suspicious_blocks: List[Tuple[int, int]],
    window_size: int,
) -> List[Tuple[int, int, int, int]]:
    """
    Convert block (row, col) indices to pixel-space ROI bounding boxes.
    Args:
        suspicious_blocks: List of (row, col) from find_suspicious_blocks().
        window_size:       Block edge length in pixels.
    Returns:
        List of (x0, y0, x1, y1) tuples in pixel coordinates.
```

```
    """
    return [
        (c * window_size, r * window_size,
         c * window_size + window_size, r * window_size + window_size)
        for r, c in suspicious_blocks
    ]

# ── Full pipeline
def analyze_entropy(
    image: np.ndarray,
    window_size: int = 16,
    base_threshold: float = 0.7,
) -> Dict:
    """
    Run the full entropy-analysis pipeline on all colour channels.
    For each channel the function:
    1. Extracts the LSB plane.
    2. Computes the block entropy map.
    3. Estimates local texture complexity.
    4. Builds an adaptive threshold map.
    5. Identifies suspicious (high-entropy) blocks and their pixel ROIs.
    Args:
        image:          RGB or RGBA uint8 numpy array.
        window_size:    Block edge length for entropy (default 16).
        base_threshold: Base entropy threshold (default 0.7).
    Returns:
        Dict keyed by channel name ('R', 'G', 'B', …).  Each value is a dict
        with keys:
            'entropy_map'         : 2-D float32 array
            'texture_map'         : 2-D float32 array
            'adaptive_threshold'  : 2-D float32 array
            'suspicious_blocks'   : list of (row, col)
            'rois'                : list of (x0, y0, x1, y1)
           'stats'           : dict with min/max/mean/suspicious_count
    """
    channel_labels = ['R', 'G', 'B', 'A']
    n_ch = image.shape[2] if image.ndim == 3 else 1
    results: Dict = {}
    for i in range(min(n_ch, 3)):        # Analyse R, G, B (skip alpha)
        label = channel_labels[i]
        ch = image[:, :, i] if image.ndim == 3 else image
        lsb = (ch & 1).astype(np.uint8)
        e_map = compute_entropy_map(lsb, window_size)
        t_map = compute_texture_map(ch, window_size)
        thresh = compute_adaptive_threshold(t_map, base_threshold)
```

```
        sus_blocks = find_suspicious_blocks(e_map, thresh, t_map)
        rois = blocks_to_pixel_rois(sus_blocks, window_size)
        results[label] = {
            'entropy_map': e_map,
            'texture_map': t_map,
            'adaptive_threshold': thresh,
            'suspicious_blocks': sus_blocks,
            'rois': rois,
            'stats': {
                'min': float(e_map.min()),
                'max': float(e_map.max()),
                'mean': float(e_map.mean()),
                'suspicious_count': len(sus_blocks),
            },
        }
    return results
```