**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**Чорноморський національний університет**
**імені Петра Могили**
**Факультет комп'ютерних наук**
**Кафедра інтелектуальних інформаційних систем**

**ДОПУЩЕНО ДО ЗАХИСТУ**
Завідувач кафедри інтелектуальних
інформаційних систем, д-р. техн. наук, проф.
_____Ю. П. Кондратенко
«____»_____2023 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

# ЗАСТОСУВАННЯ АЛГОРИТМІВ СТЕГАНОГРАФІЇ ПРИ ПЕРЕДАЧІ ТА ЗБЕРІГАННІ ДАНИХ

Спеціальність 122 «Комп'ютерні наук»

**122 – БКР – 401.21910122**

***Виконала студентка 4-го курсу, групи 401***
***_____ А. І. Слободенюк***
«  » червня 2023 р.

***Керівник: канд. техн. наук, доцент***
***_____Є. В. Сіденко***
«  » червня 2023 р.

**Миколаїв – 2023**

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**Чорноморський національний університет ім. Петра Могили**
**Факультет комп'ютерних наук**
**Кафедра інтелектуальних інформаційних систем**

Рівень вищої освіти **бакалавр**
Спеціальність **122 «Комп'ютерні науки»**
*(шифр і назва)*
Галузь знань **12 «Інформаційні технології»**
*(шифр і назва)*

**ЗАТВЕРДЖУЮ**

Завідувач кафедри інтелектуальних інформаційних систем, д-р техн. наук, проф. _____ Ю. П. Кондратенко
«____» _____20____р.

**ЗАВДАННЯ**
**на виконання кваліфікаційної роботи**

Видано студентці групи 401 факультету комп'ютерних наук Слободенюк Антоніні Ігорівні.

1. Тема кваліфікаційної роботи «Застосування алгоритмів стеганографії при передачі та зберіганні даних».

Керівник роботи Сіденко Євген Вікторович, канд. техн. наук, доцент.

Затв. наказом Ректора ЧНУ ім. Петра Могили від «___» ____ 20__ р. № _____

2. Строк представлення кваліфікаційної роботи студентом «__» _____ 20__ р.

3. Вхідні (початкові) дані до роботи: огляд існуючих методів стеганографії, технологічні вимоги, вимоги до безпеки, вибір інструментів та технологій для реалізації програмного забезпечення.

Очікуваний результат роботи: реалізація обраного алгоритму стеганографії у вигляді програмного забезпечення, яке дозволяє приховувати та витягувати інформацію з контейнерів.

4. Перелік питань, що підлягають розробці (зміст пояснювальної записки):

- огляд методів стеганографії;
- опис основних алгоритмів стеганографії;
- принципи вбудовування і вилучення прихованої інформації;
- огляд сучасних стеганографічних систем;
- реалізація алгоритму вбудовування і вилучення прихованої інформації;
- представлення результатів експериментів і тестування системи;
- висновки про ефективність та застосування розробленої системи.

5. Перелік графічних матеріалів презентація.

6. Завдання до спеціальної частини:

- здійснити аналіз умов праці в робочому приміщенні;
- встановити необхідний рівень показників для робочого приміщення, де проводяться роботи з розробки системи;
- встановити основні принципи техніки безпеки.

7. Консультанти:

| Розділ | Прізвище, ініціали та посада консультанта | Підпис |
|---|---|---|
| Спеціальна частина з охорони праці | Алексєєва А.О., канд. техн. наук, доцент | |
| | | |
| | | |

Керівник роботи _канд. техн. наук, доцент, Сіденко Є.В.
*(наук. ступінь, вчене звання, прізвище та ініціали)*

_____
*(підпис)*

Завдання прийнято до виконання Слободенюк А.І.
*(прізвище та ініціали)*

_____
*(підпис)*

Дата видачі завдання «____» _____ 202__ р.

**КАЛЕНДАРНИЙ ПЛАН**
**виконання кваліфікаційної роботи**

Тема: Застосування алгоритмів стеганографії при передачі та зберіганні даних.

| № | Найменування роботи | Початок | Закінчення | Примітки |
|---|---|---|---|---|
| **1** | Подання заяви на затвердження теми та керівників БКР | 27.10.2022 | 27.10.2022 | Виконано |
| **2** | Отримання завдання на виконання БКР | 21.11.2022 | 21.11.2022 | Виконано |
| **3** | Складання календарного плану роботи на весь період виконання БКР | 05.12.2022 | 05.12.2022 | Виконано |
| **4** | Отримання завдання на переддипломну практику | 25.04.2023 | 25.04.2023 | Виконано |
| **5** | Проходження переддипломної практики, збір та аналіз матеріалів до БКР | 01.05.2023 | 14.05.2023 | Виконано |
| **6** | Оформлення звіту з переддипломної практики | 14.05.2023 | 14.05.2023 | Виконано |
| **7** | Виконання БКР: аналіз алгоритмів, огляд існуючих рішень, формування задачі, розробка інформаційної системи | 10.02.2023 | 15.06.2023 | Виконано |
| **8** | Оформлення звіту БКР | 26.04.2023 | 16.06.2023 | Виконано |
| **9** | Попередній захист БКР на засіданні комісії кафедри | 29.05.2023 | 29.05.2023 | Виконано |
| **10** | Доробка та виправлення помилок в звіті | 30.05.2023 | 16.06.2023 | Виконано |
| **11** | Подання БКР рецензенту | 17.06.2023 | 17.06.2023 | Виконано |
| **12** | Подання БКР, її електронної копії та інших документів (відгуку, рецензії) до захисту | 20.06.2023 | 20.06.2023 | Виконано |
| **13** | Захист БКР перед екзаменаційною комісією (ЕК) | 27.06.2023 | 27.06.2023 | Виконано |

Розробив студент _____Слободенюк А.І._____ _____
*(прізвище та ініціали)* (*підпис*)

Керівник роботи __канд. техн. наук, доцент, Сіденко Є.В. _____
*(наук. ступінь, вчене звання, прізвище та ініціали)* *(підпис)*

«____» _____ 202__ р.

# АНОТАЦІЯ

**бакалаврської кваліфікаційної роботи**

**студентки групи 401 ЧНУ ім. Петра Могили**

**Слободенюк Антоніни Ігорівни**

**Тема: «Застосування алгоритмів стеганографії при передачі та зберіганні даних»**

Актуальність полягає у тому, що цифрові формати мультимедіа використовуються у всіх сферах діяльності, існують проблеми управління цифровими ресурсами, а також потреба у передачі та зберіганні даних, дослідження в області стеганографії стають все більш актуальними. Розвиток глобальних комп'ютерних мереж дозволяє швидко та ефективно передавати електронні документи, але при цьому виникає ризик незаконного копіювання та розповсюдження матеріалів. Тому виникає потреба у розробці методів та програмних засобів, які дозволяють приховувати конфіденційну інформацію в різних типах файлів, включаючи текстові, графічні, аудіо та відео файли.

Об'єкт роботи – процеси стенографії при передачі та зберіганні даних.

Предмет роботи – алгоритми стеганографії для захисту інформації у зображеннях при передачі та зберіганні даних.

Мета роботи – захист інформації у зображеннях з використанням алгоритмів стеганографії.

Робота складається із вступу, чотирьох розділів, висновків та додатків.

У першому розділі проведено аналіз предметної області, основні поняття, розглянуто аналоги розроблюваної системи, опрацьовано останні публікації на цю тему.

У другому розділі проаналізовано алгоритми реалізації стеганографічних методів, їх застосування, а також використання при безпеці спілкування та передачі даних.

У третьому розділі описано моделювання системи, метод LSB, шифр Цезаря та шифрування DES, які застосовувались для проєктування програмного застосунку. Також розглянуто методи стеганоаналізу .

У четвертому розділі описано процес проєктування системи, описано покроковий алгоритм роботи з програмним інтерфейсом, проведено тестування та здійснено атаки на результат, задля виявлення недоліків системи.

Бакалаврська кваліфікаційна робота містить 61 сторінку, 37 рисунків, 1 таблиця, 35 використаних джерел та 7 додатків.

Ключові слова: *стеганографія, алгоритм LSB, приховування даних, зображення, метод, стегоаналіз, безпека, спотворення*.

# ABSTRACT

**Bachelor's qualification work**

**of the student of 401 group of Petro Mohyla Black Sea National University**

**Slobodeniuk Antonina Igorivna**

**Title: «Application of steganography algorithms in data transmission and storage»**

The relevance lies in the fact that digital multimedia formats are used in all spheres of activity, there are problems of digital resource management, as well as the need for data transmission and storage, research in the field of steganography is becoming more and more relevant. The development of global computer networks allows fast and efficient transfer of electronic documents, but at the same time there is a risk of illegal copying and distribution of materials. Therefore, there is a need to develop methods and software tools that allow hiding confidential information in various types of files, including text, graphics, audio and video files.

The object of the work is shorthand processes during data transmission and storage.

The subject of the work is steganography algorithms for protecting information in images during data transmission and storage.

The purpose of the work is to protect information in images using steganography algorithms.

The work consists of an introduction, four chapters, conclusions and appendices.

In the first section, an analysis of the subject area, basic concepts, analogues of the developed system were considered, and the latest publications on this topic were elaborated.

In the second section, the algorithms for the implementation of steganographic methods, their application, as well as their use in the security of communication and data transmission are analyzed.

The third chapter describes the system simulation, LSB method, Caesar cipher and DES encryption, which were used to design the software application. Methods of steganoanalysis are also considered.

The fourth chapter describes the system design process, describes the step-by-step algorithm for working with the software interface, conducted testing and carried out attacks on the result in order to identify system flaws.

The bachelor thesis contains 61 pages, 37 figures, 1 table, 35 used sources and 7 appendices.

Key words: *steganography, LSB algorithm, data hiding, image, method, stegoanalysis, security, distortion.*

# ЗМІСТ

# ПЕРЕЛІК СКОРОЧЕНЬ

| | | |
|---|---|---|
| ДКП | – | дискретне косинусне перетворення |
| ДСП | – | дискретне синусне перетворення |
| СМ | – | стеганографічні методи |
| AAC | – | Advanced Audio Coding |
| BMP | – | bitmap-формат |
| DES | – | Data Encryption Standard |
| GIF | – | graphics interchange format |
| HMAC | – | hash-based message authentication code |
| JPEG | – | Joint Photographic Experts Group |
| LINQ | – | Language Integrated Query |
| LSB | – | Least Significant Bit |
| LZW | – | Lempel-Ziv-Welch |
| MDCT | – | Modified discrete cosine transform |
| RGB | – | red, green, blue |
| RLE | – | run-length encoding |
| RS | – | Regular-Singular |
| SAR | – | Specific absorption rate |
| TIFF | – | Tagged Image File Format |
| TDOM | – | Trading Day of Month |

# ВСТУП

За останні роки стеганографія, як метод приховування інформації, набула значного інтересу завдяки широкому використанню комп'ютерної технології у всіх сферах життя. Це сприяє активному впровадженню і розповсюдженню стеганографічних методів захисту. Застосування стеганографії дозволяє не лише приховано передавати дані, але й вирішувати завдання забезпечення безпеки інформації від несанкціонованого доступу, незаконного копіювання, відстеження поширення інформації та пошуку даних у мультимедійних базах.

Незважаючи на те, що багато вчених працюють над розробкою та вдосконаленням методів приховування даних, існує проблема вибору найбільш оптимального алгоритму для конкретних випадків застосування. Ця проблема є актуальною через наявність критеріїв оцінки якості стеганографічних систем і вимог до їх функціонування, але відсутність чітко визначених найкращих алгоритмів для широкого спектру стеганографічних застосувань. Оскільки зображення є найпоширенішим засобом обміну нетекстовими повідомленнями, в багатьох випадках найбільш природнім видається приховування інформації в картинках.

Об'єкт роботи – процеси стенографії при передачі та зберіганні даних.

Предмет роботи – алгоритми стеганографії для захисту інформації у зображеннях при передачі та зберіганні даних.

Мета роботи – захист інформації у зображеннях з використанням алгоритмів стеганографії.

Основні завдання для досягнення поставленої мети:

– проведення аналізу сучасних методів стеганографії;

− розробка програмного забезпечення для вбудовування та вилучення інформації на основі вивчених методів;

− проведення тестування розробленої системи;

− оцінка ефективність розробленого програмного забезпечення.

# 1 ОГЛЯД СУЧАСНОЇ КОМП'ЮТЕРНОЇ СТЕГАНОГРАФІЇ ТА ЇЇ ЗАСТОСУВАННЯ. ПОСТАНОВКА ЗАДАЧІ

## 1.1 Становлення стеганографії як науки

У сучасному світі, коли зберігання конфіденційної інформації є одним з основних завдань, стеганографія стає все більш популярною. Стеганографія - це наука про приховування інформації, що полягає в використанні спеціальних технік для забезпечення конфіденційності даних.

Історія стеганографії сягає давніх часів, коли люди використовували приховану інформацію для передачі важливих повідомлень, зберігаючи їх від проникнення ворога.

Стеганографія походить від грецького steganos (прикритий або таємний) і -graphy (письмо або малювання). Стеганографію можна визначити як приховування інформації шляхом вбудовування повідомлень серед інших текстів, зображень, графіків чи звуків. Перша стеганографічна техніка була розроблена в Стародавній Греції приблизно в 440 році до нашої ери. Вона передбачала таке приховання інформації: рабу голять голову, роблять татуювання повідомлення на шкірі голови, чекають, поки виросте волосся, щоб закрити таємне послання, і посилають раба в дорогу для доставки інформації. Одержувач голить голову раба і читає надіслане йому повідомлення. Він може відповісти у тій самій формі стеганографії.

У той же самий період часу була використана інша рання форма стеганографії. Цей спосіб задіяв Демерст, який написав послання до спартанців, попереджаючи про значне вторгнення Ксеркса. Це повідомлення було вирізане на дереві воскової таблички, а потім покрите свіжим шаром воску. І здавалося, це були порожні дошки, які без проблем проходили обшуки у сотників [1].

Вперше цей термін був використаним у 1499 році Йоганнесом Тритеміусом у своїй «Стеганографії», трактаті про криптографію та стеганографію, замаскованому під книгу про магію. Спочатку автор вирішив не друкувати його і навіть знищив його значні частини, вважаючи, що вони ніколи не повинні були

побачити світ, але текст продовжував циркулювати у вигляді попередньої чернетки і був опублікований посмертно в 1606 році [2].

Відтоді багато людей використовували цю техніку для безпечної доставки повідомлень. Наприклад, відомо, що під час обох світових воєн жінки-шпигуни використовували в’язання для надсилання повідомлень, роблячи неправильний стібок або залишаючи навмисно дірку в тканині.

У 1985 році персональні комп'ютери почали використовувати для класичної стеганографії. Тепер повідомлення можна вбудовувати в різні типи цифрових даних, таких як мова, аудіозаписи, зображення та відео. Навіть генна послідовність ДНК не залишається без уваги, і були розроблені алгоритми, які можуть приховувати повідомлення в цій послідовності. Наприклад, в 2000 році Вівіана Риска на конкурсі молодих вчених продемонструвала технологію впровадження комп'ютерних повідомлень в генну послідовність молекули [3].

Таким чином, сьогодні стеганографія представляє себе як ідеальний інструмент для створення таємних каналів зв’язку, які можна використовувати в складних сценаріях шпигунства, комп’ютерних злочинів і порушення приватності як публічних, так і приватних суб’єктів.

## 1.2 Стеганографія: поняття, особливості, сутність

Стегарнографія займає своє місце у безпеці даних. Вона не замінює криптографію, а навпаки, доповнює її. Методи стеганографія ніяк не втручаються в саме повідомлення, вони лише приховують сам факт передачі інформації. А якщо повідомлення зашифроване до того, то вони створюють ще один рівень захисту.

Набором методів та інструментів для створення прихованого каналу передачі інформації називають стеганографічну систему, її представлено на рис. 1.1.

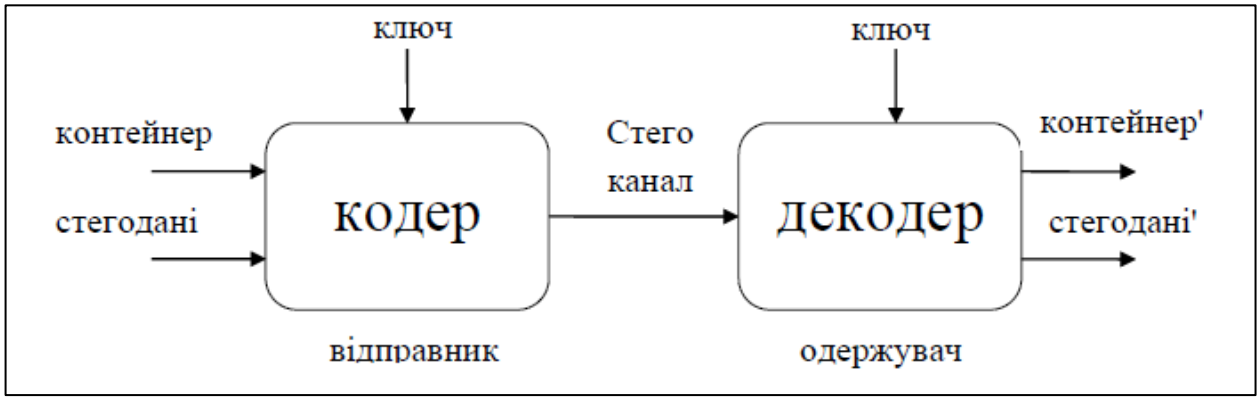



Рисунок 1.1 – Модель стегосистеми

За типом стеганоключа вона може бути [4] :

- з секретним ключем;
- відкритим ключем.

У першому випадку використовується один ключ, що передається по захищеному каналу. В другому - різні ключі, що можуть передаватися вільно по незахищеному каналу. Така система працює ефективно навіть при взаємній недовірі між відправником та одержувачем.

Контейнер - це об'єкт, який містить секретну інформацію. Він може бути порожнім або заповненим. Порожній контейнер не містить вбудованого повідомлення, тоді як заповнений контейнер, який називається стегоконтейнером, містить вбудовану інформацію.

Контейнери поділяють на два основні типи: потокові та фіксовані. Потоковий контейнер представляє собою неперервну послідовність бітів, у який можуть бути приховані декілька повідомлень. Інтервали між вбудованими бітами визначаються генератором псевдовипадкових послідовностей з рівномірним розподілом інтервалів між відліками. Головна складність полягає в синхронізації, визначенні початку і кінця послідовності. На відміну від потокового, у фіксованому контейнері розмір і характеристики повідомлення відомі наперед. Контейнер може бути вибраний, випадковий або нав'язаний, в залежності від вбудовуваного повідомлення. Однак, у потоковому контейнері повідомлення вміщується в нього

в реальному масштабі часу, тому в кодері невідомо наперед, чи вистачить розміру контейнера для передачі всього повідомлення.

Можливі такі варіанти контейнерів:

− стегосистема сама створює контейнер, який використовується для приховування повідомлення. Прикладом такої системи може бути програма MandelSteg, яка генерує фрактал Мандельброта в якості контейнера для приховування інформації. Цей підхід можна назвати конструктивною стеганографією;

− вибіркова стеганографія передбачає вибір контейнера зі значної кількості альтернативних варіантів, щоб знайти найбільш підходящий для вбудовування повідомлення. Важливим критерієм при цьому є природність контейнера. Проте навіть найоптимальніший контейнер може містити невелику кількість прихованих даних порівняно з його величиною;

− контейнер надходить ззовні. В цьому випадку потрібний контейнер може бути не обраним, тому що приховування повідомлення здійснюється в першому доступному контейнері, незалежно від того, наскільки він підходить для цієї мети. Цей підхід можна назвати безальтернативною стеганографією.

Створення прихованого повідомлення в контейнері може потребувати використання одного або кількох ключів. Стеганографічний ключ (або просто "ключ") є секретним інструментом, необхідним для вбудовування та вилучення прихованої інформації. Стеганосистема формує канал, через який передається зміст заповненого контейнера. Цей канал називається стеганографічним каналом (або "стегоканалом"), і він використовується для передачі стеганографічних даних.

### 1.3 Огляд останніх публікацій та існуючих аналогів

Останнім часом стеганографія набуває все більшої популярності в зв'язку з ростом зацікавленості у збереженні конфіденційності та безпеки даних. Дослідження в цій галузі постійно продовжуються і відбувається активний пошук нових ідей та методів.

Так як криптографія і стеганографія виконують одну головну роль – захищають дані, але різними способами, використання їх разом створює гібридну двошарову систему безпеки, яка захищає дані з точки зору секретності та непомітності. У статті [5] автори пропонують використання подвійної сліпої оборотної схеми безпеки, щоб забезпечити захист переданої інформації. У цій схемі використовується шифрування Blowfish як криптографічний рівень, за яким слідує стеганографічний рівень вбудовування найменш значущого біта зображення, що дозволяє розповсюджувати зашифровані секретні дані по пікселях носія покриття зображення. Автори також використовують розподіл даних по кольорових каналах пікселів, вибраних для вбудовування, згідно з послідовністю сірого коду. Для оцінки ефективності запропонованої схеми були обчислені значення пікового відношення сигнал/шум, середньоквадратичної помилки та індексу структурної подібності та порівняні з іншими схемами вбудовування найменш значущих бітів у літературі. Результати показали, що запропонована схема є більш потужною, непомітною та має меншу обчислювальну складність порівняно з іншими схемами.

Стеганографія зазвичай використовується, щоб приховати важливу інформацію у видимих носіях, переважно зображення чи відео. У дослідженні [6] пропонується нова схема, яка служить подвійній меті приховування та стиснення зображень радара із синтезованою апертурою (SAR) за допомогою техніки стеганографії для аерокосмічного та супутникового застосування. Зображення SAR є важливим джерелом інформації для багатьох програм, особливо для картографування географічних зон. Стиснення зображень SAR допомагає знизити вартість їх зберігання та передачі даних у повільних каналах. Проте зі зростанням обсягу даних, здатність їх зберігати та передавати не зростає настільки швидко. Тому потрібні алгоритми стиснення та декодування даних, які можуть забезпечити вищу ступінь стиснення та швидку передачу даних, зберігаючи якість зображення на прийнятному рівні. Радар з синтетичною апертурою (SAR) використовується для створення радіолокаційних зображень, які можуть бути приховані. Схема стегання зображень SAR забезпечує задовільне стиснення як видимих, так і

прихованих зображень, за допомогою алгоритму Лемпеля-Зіва-Велча (LZW). Реконструйовані зображення візуально не відрізняються від вихідного зображення, тому ця проста схема може бути легко адаптована для різних програм обробки зображень.

А от для відео було порівняно аналіз дискретного косинусного перетворення (ДКП) і дискретного синусного перетворення (ДСП) на основі відеостеганографії в стисненій області [7]. Цей метод полягає в приховуванні секретної інформації в випадково вибраних секретних кадрах, які витягуються з послідовності обкладинки RGB. Для цього він виділяє нединамічну область з секретного кадру і перетворює її піксельні значення в частотну область за допомогою ДКП або ДСП. Потім використовується випадковий найменший значущий біт цілочисельної складової ДКП або ДСП, як об'єкт-носій для приховування секретної інформації, що забезпечує якість відео та секретну ємність для передачі даних. Захищене стиснене стеговідео може бути реконструйовано за допомогою технології стиснення відео H.264, що полегшує передачу по мережевому каналу між відправником і одержувачем. Метод був експериментально перевірений на різних наборах відеоданих з різною роздільною здатністю зображень RGB для приховування секретного повідомлення.

В медицині наприклад важливо зберігати в безпеці медичні зображення, які можуть містити конфіденційну інформацію. Для цього вже є варіанти вирішення, але найефективнішим та найпопулярнішими із них є, ймовірно, шифрування за допомогою традиційних алгоритмів шифрування, таких як Advanced Encryption Standard [8]. Ця пропозиція намагається дослідити можливість використання спеціальних алгоритмів шифрування зображень, які мають застосування в галузях стеганографії та біометричних систем для шифрування медичних зображень. Основна перевага цієї пропозиції полягає в додаванні блоку підстановки, що має підвищити безпеку всього алгоритму.

Щоб підвищити безпеку стеганографії, у дослідженні [9] пропонується генеративна стеганографія покриття, яка створює генератор, кодер-декодер,

мережу оптимізації стеганалізу та змагальну мережу стеганалізу. Завдяки цьому забезпечуються менші спотворення кольору та більшу стійкість до стеганографії при вбудовуванні секретних зображень в обкладинки.

У статті [10] для забезпечення надійності, приховування та безпеку описано стеганографічний алгоритм, який поєднує хаотичне шифрування та функцію мінімального спотворення. Цей алгоритм генерує випадкові ключі та шифрує секретну інформацію за допомогою відображення Logistic і ChebyShev, після чого вбудовує ключ та зашифровану інформацію в зображення носія за допомогою стеганографічного алгоритму Хілла. Для підвищення безпеки, алгоритм зберігає вбудований ключ та ключ дешифрування окремо, так що зловмисник не може отримати правильну секретну інформацію, навіть якщо він отримає вбудований ключ. Експерименти показали, що швидкість зміни пікселів алгоритму становить 6,76%, пікове співвідношення сигнал/шум складає 59,51 дБ, і він має дуже хорошу здатність до захисту від антистеганографічного аналізу.

Для стеганографії аудіо був розроблений алгоритм AAC, який використовує генетичний алгоритм та коригування коефіцієнта MDCT [11]. Він вибирає область з низькими значеннями коефіцієнта MDCT як місце для вбудовування бітів, а коефіцієнти з кодової книги 1/2 використовуються для зміни. Генетичний алгоритм використовується для оптимізації зміни коефіцієнта, що допомагає підвищити стійкість до стеганалізу. Результати експериментів свідчать про високу ефективність алгоритму при вбудовуванні та майже непомітність.

Ще стеганографія використовується для приховування інформації за допомогою джойстика та сенсорних датчиків [12]. Прості у використанні пристрої дозволяють приховано вводити дані та вбудовувати кілька повідомлень в один контейнер за допомогою двох методів - videostego та metastego. Ці методи були вибрані через їх низьку складність для роботи в середовищах з обмеженими ресурсами. Пропоновані датчики можна замінити на інші з аналогічними ункціями.

У дослідженні [13] розглянуто переваги застосування блокчейну в стеганографії, а саме можливість вставки прихованих даних без зміни вихідних та

готовність платформи блокчейну до їх зберігання та передачі. Також запропоновано два алгоритми стеганографії в блокчейні, високої та середньої ємності. Досліджено попередні схеми стеганографії в блокчейні та їхні недоліки, а також розглянуто проблеми з точки зору стеганографів та стеганалізаторів.

Авторами публікації [14] запропоновано використання поворотів та переворотів як нового підходу до стеганографії для уникнення виявлення прихованих повідомлень. Для збільшення швидкості кодування та зменшення обсягу зберігання, мультимедійні файли стискаються. Для забезпечення автентифікації та цілісності повідомлень обрано алгоритм HMAC. Ефективність запропонованих методів стеганографії було оцінено за допомогою порівняльного аналізу гістограм та підтверджено за допомогою моделювання.

Техніка стеганографії вебсторінок використовується для прихованого спілкування, коли секретне повідомлення вставляється в звичайний файл вебсторінки за допомогою методів стеганографії. Зазвичай оригінальна вебсторінка і стего-сторінка виглядають однаково, тому їх важко відрізнити. У статичному підході вебсторінка стего залишається незмінною, що обмежує її приховувальну здатність. У статті [15] запропоновано модель динамічної роботи на основі часу (TDOM), яка покращує продуктивність існуючих методів стеганографії вебсторінок, дозволяючи динамічно заміняти вебсторінку стего іншими вебсторінками стего або оригінальною вебсторінкою. Це покращує її приховувальну здатність та неможливість виявлення. Крім того, розроблено два алгоритми динамічної роботи на основі часу (TDOA-C і TDOA-U), які покращують здатність приховувати стего-вміст існуючими методами, а TDOA-U покращує невиявленість існуючих методів.

## 1.4 Огляд існуючих технологій, методів, підходів для вирішення поставленої задачі

Загальна характеристика стеганографічних методів (СМ) полягає у можливості приховування повідомлення в нешкідливих об'єктах, які не

привертають уваги, з метою передачі інформації непомітно для зловмисників. Відмінність від криптографії полягає в тому, що в наявності зашифрованої інформації зловмисник може звернути на неї увагу, тоді як використання СМ забезпечує захист інформації на трьох рівнях: невідомість самого факту передачі прихованої інформації, невідомість алгоритму впровадження прихованої інформації в контейнері та невідомість способу кодування інформації.

Методи приховування в просторовій області зображені на рис. 1.1.

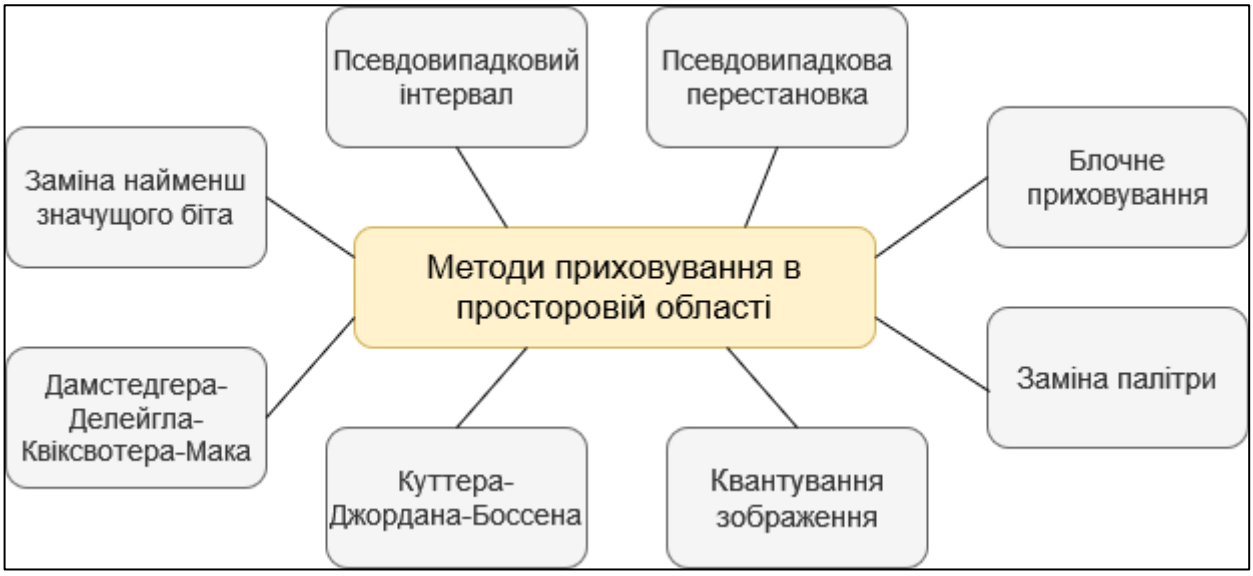



Рисунок 1.1 – Методи приховування в просторовій області

**Метод заміни найменш значущого біта (LSB)** [16] - це один з найпоширеніших методів стеганографії, при якому інформація приховується шляхом заміни найменш значущого біта (біта з найменшою вагою) пікселів в зображенні. Завдяки тому, що зміна найменш значущого біта майже не помітна для ока людини, цей метод є ефективним у приховуванні додаткової інформації у зображеннях. Однак, цей метод може бути вразливим до стеганалізу, тому він може бути покращений за допомогою додаткових технік та алгоритмів.

**Метод псевдовипадкового інтервалу** [17] використовується для вбудовування бітів секретного повідомлення в контейнер, де відстань між вбудовуваними бітами визначається псевдовипадково. Цей метод є ефективним, коли бітова довжина повідомлення значно менша за кількість пікселів у зображенні. Однак, недоліком методу є те, що біти повідомлення в контейнері

розміщені в тій же самій послідовності, що й в самому повідомленні, а інтервал між ними визначається випадково.

**У методі псевдовипадкової перестановки** [18] біти секретного повідомлення перемішуються у контейнері за допомогою псевдовипадкової перестановки. У цьому методі біти повідомлення не розміщуються в контейнері послідовно, а замість цього їх перемішують випадковим чином за допомогою алгоритму перестановки. Це робить його більш ефективним за рахунок того, що зменшується кореляція між бітами повідомлення і місцем, де вони розташовані в контейнері, що забезпечує більшу захищеність. Недоліком методу є те, що після використання алгоритму перестановки біти повідомлення можуть все ще знаходитися поруч один з одним, що може створити певний ризик детекції.

**Метод блочного приховування** [19] використовується для приховування повідомлень у цифрових зображеннях. В цьому методі повідомлення розбивається на блоки, які потім вбудовуються у пікселі зображення. При вбудовуванні блоків в зображення, кожен блок замінює певну кількість пікселів. Це забезпечує високий рівень стійкості, так як зловмисники, які намагаються розкрити приховане повідомлення, не зможуть знайти точний розмір і положення блоків. Приховане повідомлення можна виявити за допомогою ключа, який використовується для відновлення блоків у правильному порядку. Ключ може бути введений вручну або згенерований за допомогою певного алгоритму. Однією з переваг методу блочного приховування є його висока стійкість до атак перебору. Недоліком є необхідність використовувати зображення з достатньою кількістю пікселів для приховування повідомлення.

**Метод заміни палітри** [20] зображення полягає в заміні значень інтенсивності кольорів пікселів зображення на інші значення, які візуально мало відрізняються від початкових. Це дозволяє приховати додаткову інформацію в зображенні без зниження якості зображення або помітної зміни в його вигляді. Для захисту прихованої інформації метод заміни палітри може використовувати різні алгоритми шифрування, які забезпечують конфіденційність і цілісність інформації.

Однак, як і у більшості методів стеганографії, метод заміни палітри не є абсолютно безпечним і може бути розкритий при використанні спеціальних атак.

**Метод квантування зображення** - це процес [21] редукції кількості кольорів (точніше, їх інтенсивностей) у зображенні. Цей метод заснований на принципі кластеризації, коли кольори зображення групуються в кластери за допомогою алгоритму кластеризації. Кожному кластеру присвоюється певний значення, яке стає новим кольором зображення. Після квантування кількість кольорів зменшується, що дозволяє зменшити розмір зображення і зменшити об'єм пам'яті, необхідний для зберігання зображення. Однак, квантування зображення може призводити до втрати деталей і якості зображення, особливо якщо кількість кольорів зменшується значно.

**Метод Куттера-Джордана-Боссена** [22] (Cut-Jordan-Boss) полягає в заміні пікселів зображення на інші, що мають малу відмінність від оригінальних пікселів. Цей метод використовується для вбудовування секретного повідомлення в пікселі зображення. Він використовує два ключі - ключ приховування та ключ витягування. Секретне повідомлення вбудовується в пікселі зображення з використанням ключа приховування. Для вилучення секретного повідомлення зображення проходить через алгоритм витягування з використанням ключа витягування. Цей метод має декілька переваг, зокрема, його важко виявити та атакувати. Однак, він також має деякі недоліки, такі як втрата якості зображення та складність вбудовування та витягування повідомлення.

**Метод Дамстедгера-Делейгла-Квіксвотера-Мака** використовує перетворення Хаара для розбиття зображення на рівні розклади. Інформація приховується шляхом модифікації коефіцієнтів розкладу на певних рівнях. Він має кілька переваг, серед яких є відсутність статистичних відмінностей між оригінальним і модифікованим зображеннями, що забезпечує його роботу прихованого каналу передачі інформації. Також метод забезпечує високу міцність стеганографії та стійкість до атак на перетворення. Недоліком методу є те, що він

потребує більшої обчислювальної потужності порівняно з іншими методами стеганографії, такими як LSB або метод псевдовипадкового інтервалу.

Взагалі СМ приховування у просторовій області є нестійкими до більшості спотворень. Наприклад, якщо до них застосувати операцію стиснення із втратами скоріш за все вбудована у контейнер інформація буде повністю знищена. Більш стійкими до спотворень є методи приховування інформації у частотній області.

Методи приховування в частотній області можна побачити на рис. 1.2.

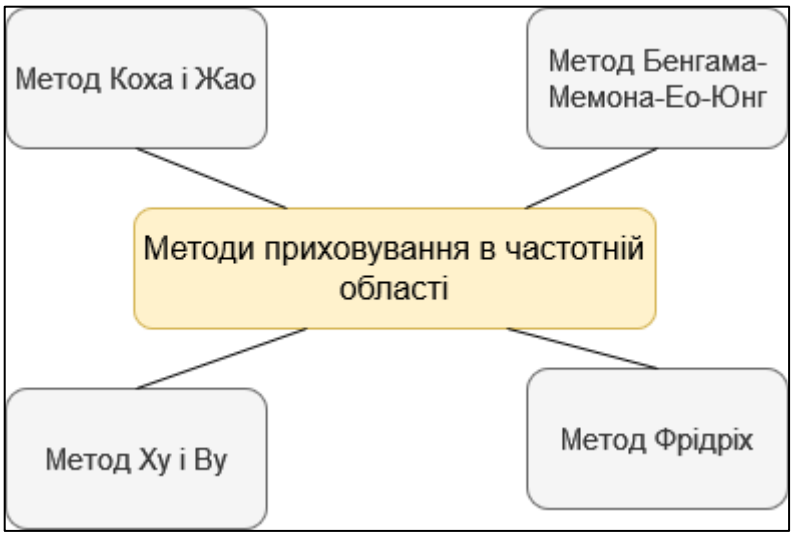



Рисунок 1.2 – Методи приховування в частотній області

Основною ідеєю **методу Коха і Чжао** [23] є використання «білого шуму», тобто додавання до зображення випадкових значень, що дозволяє зменшити візуальну видимість змін вихідного зображення. Крім того, метод Коха і Чжао має можливість контролювати розмір додаткової інформації та її вбудовування в зображення. Він використовує блокове вбудовування, тобто розбиває зображення на блоки та використовує їх для вбудовування додаткової інформації. Щоб забезпечити захист від атак на основі статистичного аналізу, метод використовує техніку блокової перестановки. Метод ефективний і надійний, дозволяє приховувати інформацію в зображеннях з високим ступенем непомітності. Однак він має свої обмеження, такі як висока обчислювальна складність і погіршення якості зображення при вбудовуванні великих обсягів інформації.

**Метод Бенгама-Мемона-Ео-Юнг** - це метод приховування інформації в зображенні шляхом вбудовування бітів повідомлення в пікселі зображення. Використовує два рівні: рівень простих замін та рівень зміни місць. На першому рівні, біти повідомлення вбудовуються в найменш значущі біти кожного пікселя. На другому рівні, зберігається інформація про те, які пікселі містять приховану інформацію, і змінюється порядок цих пікселів у зображенні, що дозволяє ускладнити процес виявлення та вилучення прихованої інформації. Цей метод має декілька переваг, таких як висока стійкість до атак методом перевантаження та зниження якості зображення, а також можливість вбудовувати більшу кількість бітів в порівнянні з іншими методами. Однак, він може бути вразливий до стеганалізу, тобто метода виявлення прихованої інформації, що базується на відмінностях між оригінальним зображенням і зображенням з прихованою інформацією.

**Метод Ху і Ву** використовується для отримання контурів зображення, особливо для зображень зі складною геометрією. Працює на основі обчислення моментів зображення і отримує вихідне зображення в чорно-білому форматі.

**Метод Фрідріха** [24] полягає у вбудовуванні секретної інформації в низькочастотні коефіцієнти дискретного косинусного перетворення (ДКП) сигналу-контейнера. Перед початком вбудовування зображення-контейнер перетворюється за допомогою ДКП, а потім низькочастотні коефіцієнти, що потрапляють в певний діапазон значень, замінюються на коефіцієнти секретного повідомлення. Це забезпечує зниження помітності вбудованого повідомлення, оскільки зміна значень низькочастотних коефіцієнтів мало впливає на зовнішній вигляд зображення. Крім того, метод Фрідріха має високу стійкість до атак, пов'язаних зі зміною значень пікселів контейнера. Однак, вбудовування повідомлень занадто великого обсягу може призвести до помітної деградації якості зображення.

## 1.5 Проблематика та постановка завдання

Зараз, у цифрову епоху, коли мультимедійні формати широко використовуються, з'являються нові виклики управління цифровими ресурсами та зберіганням даних. Це створює актуальність досліджень в області стеганографії, яка займається приховуванням інформації в різних типах файлів з метою забезпечення конфіденційності та захисту від несанкціонованого доступу.

Одним з основних факторів, що зробили стеганографію насущною, є широка доступність та поширеність цифрових технологій. Глобальні комп'ютерні мережі, такі як Інтернет, забезпечують швидку та ефективну передачу електронних документів у різних форматах. Однак, це також відкриває можливості для незаконного копіювання, розповсюдження або зловживання інформацією. І саме тут стеганографія набуває значення, надаючи засоби для приховування конфіденційної інформації в цифрових ресурсах.

Дослідження в області стеганографії спрямовані на розробку методів та програмних засобів, які дозволяють ефективно приховувати дані у різних типах файлів. Це може бути текстовий документ, графічне зображення, аудіо або відеофайл. Ідея полягає в тому, щоб такі зміни в файлі були непомітними для звичайного спостерігача, але при цьому інформація може бути легко витягнута з файлу за допомогою спеціальних алгоритмів та ключів.

Одним з головних завдань стеганографії є забезпечення конфіденційності та захисту інформації, а також запобігання незаконному доступу до неї. Це особливо важливо для передачі чутливої інформації, наприклад, у випадку комерційних або державних документів, особистих даних, банківської інформації тощо. Застосування стеганографії дозволяє зберігати та передавати ці дані, зберігаючи їх конфіденційність та виключаючи можливість їх небажаного розголошення.

Таким чином, розробка методів та програмних засобів стеганографії є актуальним завданням, яке відповідає сучасним вимогам безпеки та захисту інформації у цифровому середовищі. Використання стеганографії дозволяє зберігати та передавати дані у вигляді звичайних файлів, забезпечуючи їх

конфіденційність та захищеність. При цьому, вирішуються проблеми управління цифровими ресурсами, а також запобігається незаконне копіювання та розповсюдження матеріалів, зберігаючи приватність та безпеку інформації.Об'єкт роботи – процеси стенографії при передачі та зберіганні даних.

Предмет роботи – алгоритми стеганографії для захисту інформації у зображеннях при передачі та зберіганні даних.

Мета роботи – захист інформації у зображеннях з використанням алгоритмів стеганографії.

Завдання дослідження:

- проаналізувати методи та алгоритми стеганографії;
- вивчити стандарти та протоколи стеганографії;
- розробити програмне забезпечення для вбудовування та вилучення інформації на основі вивчених методів;
- провести експериментальне дослідження розробленого програмного забезпечення;
- оцінити ефективність розробленого програмного забезпечення;
- проаналізувати можливості застосування розробленого програмного забезпечення в різних сферах діяльності.

**Висновки до розділу 1**

У цьому розділі розглянуто походження стеганографії, її основні особливості, принцип роботи, застосування алгоритмів при передачі та зберіганні даних. Проведено дослідження та порівняльний аналіз різних методів стеганографії, що дозволило визначити їх переваги та недоліки.

Також опрацьовано останні публікації на цю тему, проаналізовано використання стеганографії у різних галузях, зокрема у медичній сфері, для супутникових знімках з технологією SAR, а також різні способи та використання технологій для приховання інформації у зображеннях, аудіо та відео.

# 2 МОДЕЛІ ТА МЕТОДИ СТЕГАНОГРАФІЇ ДЛЯ ЗАБЕЗПЕЧЕННЯ БЕЗПЕКИ СПІЛКУВАННЯ

## 2.1 Алгоритми реалізації стеганографічних методів

### 2.1.1 Алгоритми вбудовування інформації

**Алгоритм JSteg** [25], розроблений Дереком Уфамом в 2004 році, є одним з алгоритмів стеганографії, який призначений для вбудовування даних в зображення JPEG. Цей алгоритм відзначається високою місткістю стеганографічних повідомлень, оскільки можливо приховати повідомлення, що займає до 12.8% від загального обсягу зображення (контейнера).

Процедура роботи алгоритму JSteg складається з послідовної заміни найменш значущих бітів (LSB) коефіцієнтів ДКП на біти прихованої інформації, не вносячи при цьому жодних інших змін у зображення. Щоб пояснити, що таке ДКП, потрібно пам’ятати, що для кожного кольорового компонента зображення формат JPEG використовує математичну функцію, яка називається дискретним косинусним перетворенням або просто ДКП, щоб перетворити блоки 8×8 пікселів (так звані також Canonical Bases) зображення у 64 відповідних коефіцієнтах ДКП.

Ця формула представляє перетворення канонічної основи , що належить просторовій області, у відповідний блок 8×8, що належить частотній області. Блок 8×8 пікселів більш спрощено перетворюється в частотний спектр, який утворюється лише пікселями чорно-білого кольору. Ділянки, представлені більшою щільністю білих пікселів, є областями з вищими частотами, які не сприймаються людським оком, а тому є витратними. Саме з видаленням цих областей алгоритм стиснення настільки ефективний.

JSteg використовує LSB коефіцієнтів ДКП як надлишкові біти для вставки прихованого вмісту в обкладинку. Треба зазначити, що модифікація одного коефіцієнта ДКП поширюється на всі 64 пікселі в блоці. Оскільки зміни відбуваються в частотній області, а не в просторовій області, JSteg не сприйнятливий до візуальних атак.

Алгоритм не використовує стеганоключ, тому кожен, хто його знає про приховане повідомлення, може відновити його. Хоча цей метод забезпечує ефективний захист від візуальних атак, він досить вразливий до статистичних атак. Заміна бітів в JSteg впливає на частоту виникнення конкретних значень LSB. Дослідження, показані на рис. 2.1, демонструють вплив JSteg на пари частот появи коефіцієнтів JPEG. При модифікації зображення припускається, що суміжні частоти є однаковими. Для визначення очікуваного розподілу проводиться обчислення середнього арифметичного, яке порівнюється з емпіричним розподілом.

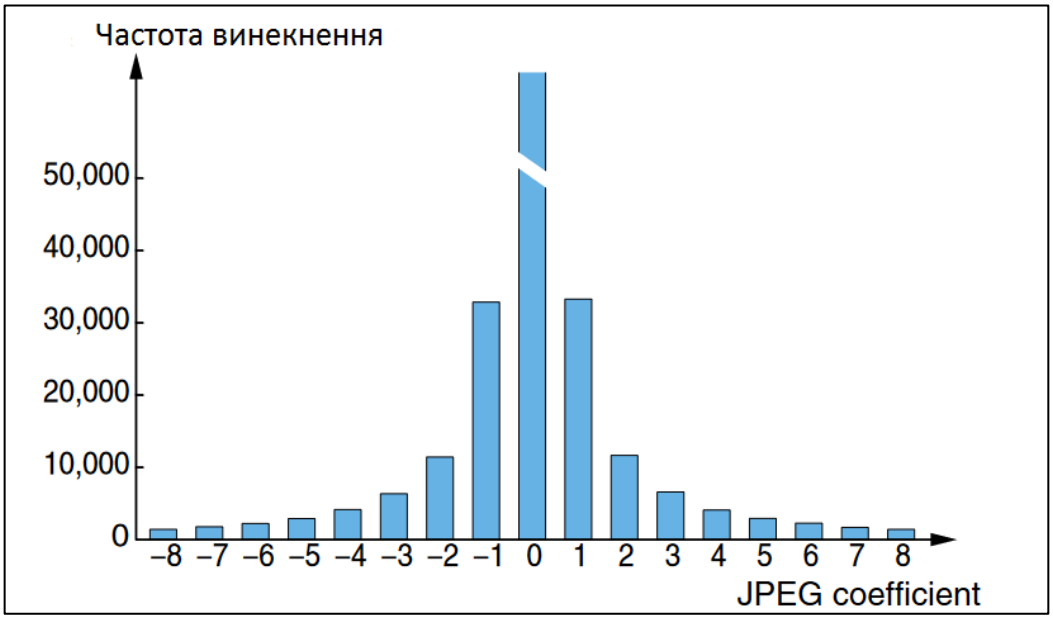



Рисунок 2.1 – JSteg зрівнює пари коефіцієнтів

На діаграмі 2.2 представлена статистична атака на стеганографічний алгоритм JSteg, виконана з використанням контейнера, заповненого на 50% (тобто 7680 байт). Діаграма ілюструє ймовірність успішного вбудовування прихованого повідомлення.


Рисунок 2.2 – Імовірність впровадження в Jsteg стеганограму (50% заповнення)

**Алгоритм F5** [26] є набагато складнішим, ніж JSteg. Він базується на двох окремих фазах:

– послідовність сортування коефіцієнтів ДКП, які потрібно змінити за допомогою псевдовипадкових чисел на основі секретного ключа (який використовується для розподілу в «розсіяному» порядку змін у зображенні);

– використання так званого матричного кодування (корисно для підвищення ефективності вбудовування, значно зменшуючи частку змін, необхідних для вбудовування прихованих даних у зображення).

Реалізацію цього алгоритму зображено на рис. 2.3, в якому ДКП - дискретне косинусне перетворення, К – квантування, П – перестановка, ФВ - функція вбудовування, Х - кодер Хаффмана, ГВЧ - генератор псевдовипадкових чисел.

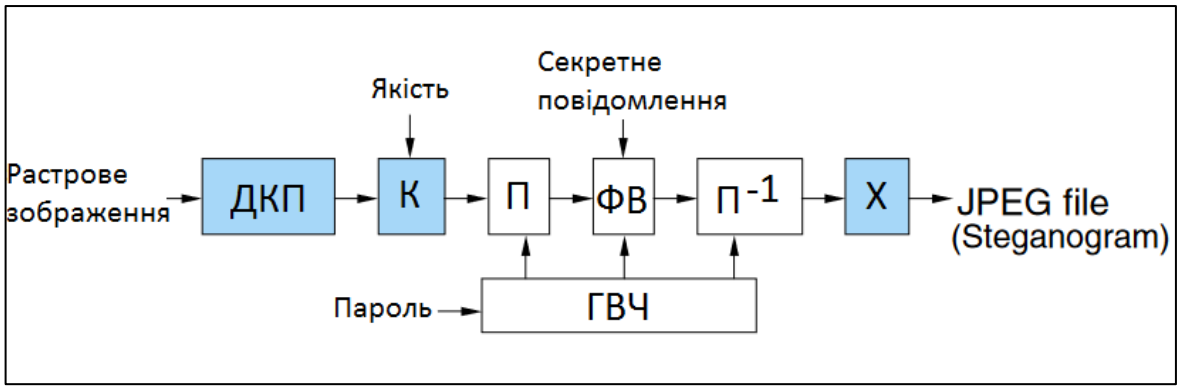



Рисунок 2.3 – Реалізація F5

Цей алгоритм значно мінімізує кількість коефіцієнтів ДКП, необхідних для зберігання секретної інформації, відповідно зменшуючи зміни, які відбуваються в зображенні обкладинки. Завдяки цим характеристикам F5 має гарну стійкість до статистичних атак.

**Outguess** можна вважати серединою між JSteg і F5, був запропонований Нілсом Профосом [27] для протидії статистичній атаці хі-квадрат. У першому проході, аналогічно до JSteg, OutGuess вбудовує біти повідомлення за допомогою випадкової послідовності в найменш значущі біти коефіцієнтів, пропускаючи 0 та 1. Після вбудовування зображення обробляється знову за допомогою другого проходу. На цей раз вносяться корекції до коефіцієнтів, щоб гістограма стегозображення відповідала гістограмі оригінального зображення. Оскільки атака з використанням квадрату хі-квадрат базується на аналізі першого порядку статистики стегозображення, вона не здатна виявити повідомлення, вбудовані за допомогою OutGuess. Профос також повідомляє, що корекції виконуються таким чином, щоб уникнути виявлення за допомогою його узагальненої атаки з використанням квадрату хі-квадрат [28].

### 2.1.2 Алгоритми стиснення зображень

Стиснення зображення – це процес, застосований до графічного файлу для мінімізації його розміру в байтах без погіршення якості зображення нижче прийнятного порогу. Зменшивши розмір файлу, можна зберігати більше зображень на заданому об’ємі диска чи пам’яті. Зображення також вимагає меншої пропускної здатності під час передачі через Інтернет або завантаження з вебсторінки, що зменшує перевантаження мережі та пришвидшує доставку вмісту.

Методи, які використовуються для стиснення файлів зображень, зазвичай належать до однієї з двох категорій: із втратами та без втрат.

Стиснення із втратами зменшує розмір файлу зображення, остаточно видаляючи менш важливу інформацію, зокрема зайві дані. Воно може значно зменшити розмір файлу, але також може знизити якість зображення до рівня спотворення. Однак якість можна зберегти, якщо стиснення застосовано ретельно.

Однією з проблем стиснення з втратами є його незворотність. Після того, як його було застосовано, зображення ніколи не можна відновити до початкового стану. Якщо стиснення з втратами багаторазово застосовується до того самого зображення, воно дедалі більше спотворюється. Тим не менш, стиснення з втратами даних виявилося цінною стратегією для Інтернету, де часто допускається помірне погіршення якості зображення.

Стиснення зображення називається без втрат застосовує стиснення без видалення критичних даних або зниження якості зображення, що призводить до стисненого зображення, яке можна відновити до початкового стану без погіршення чи спотворення. Однак стиснення без втрат не зменшує розмір файлу майже так само, як стиснення з втратами, пропонуючи невелику перевагу з точки зору місця для зберігання, пропускної здатності мережі або швидкості завантаження. Стиснення без втрат зазвичай використовується в ситуаціях, коли якість зображення важливіша за простір на диску чи продуктивність мережі, наприклад для зображень продуктів або для демонстрації творів мистецтва.

**Алгоритм RLE** – це простий алгоритм стиснення даних без втрат [29]. Він стискає дані, зменшуючи кількість повторюваних і послідовних даних, які називаються циклами. Для цього зберігається кількість цих прогонів, а потім дані. Оскільки при цьому типі стиснення даних немає втрат, вихідні дані можна повністю відновити після декомпресії. Гнучкість алгоритму як у процесах кодування (стиснення), так і декодування (декомпресії) є однією з його найпривабливіших якостей.

Алгоритм в першу чергу призначений для зображень, які мають великі області з повторюваними кольорами, такі як ділова графіка, схеми, малюнки і т.д. Однак, недоліком такого підходу є те, що в деяких випадках він може призводити до збільшення розміру файлу замість його зменшення, особливо при обробці кольорових фотографій.

Роботу цього алгоритму можна продемонструвати наступним чином :

Вхідний потік даних:

AAAAAABBBBBCCCCCCCCDDEEEEEEFFF

Потік даних після кодування:

6A5B8C2D5E3F

У цьому прикладі ми маємо 29 вхідних символів, на виході вони стискаються до 12 символів.

Формати: BMP, TIFF, GIF і PCX.

Чим менше значення коефіцієнта стиснення, тим більш ефективним є метод стиснення. Його можна розрахувати за наступною формулою (2.1):

$$\text{коефіцієнт стиснення} = Vx / Vn, \quad (2.1)$$

де Vx - об'єм пам'яті, необхідний для збереження вихідної послідовності даних,

Vn - об'єм вхідної послідовності даних.

Отже, чим більша довжина послідовних повторюваних даних, тим ефективнішим буде алгоритм RLE.

У 1952 році, під час написання своєї курсової роботи, аспірант Массачусетського технологічного інституту Девід Хаффман розробив алгоритм, який отримав його ім'я - **алгоритм Хаффмана** [30]. Це техніка стиснення даних для зменшення їх розміру без втрати даних.

Припустімо, що наведений нижче рядок потрібно надіслати через мережу (див. рис. 2.4).


Рисунок 2.4 – Початковий рядок

Кожен символ займає 8 біт. Загалом у наведеному вище рядку 15 символів. Таким чином, для надсилання цього рядка потрібна загальна кількість: 8 * 15 = 120 бітів. Використовуючи техніку кодування Хаффмана, можна стиснути рядок до меншого розміру.

Кодування Хаффмана спочатку створює дерево, використовуючи повторювані символу, а потім генерує код для кожного символу. Декодування також здійснюється за допомогою того ж дерева.

Алгоритм Хаффмана запобігає будь-якій неоднозначності в процесі декодування за допомогою концепції префіксного коду , тобто код, пов'язаний із символом, не повинен бути присутнім у префіксі будь-якого іншого коду.

Кодування Хаффмана виконується за допомогою наступних кроків.

1. Потрібно обчислити частоту кожного символу в рядку (див. рис. 2.5).


Рисунок 2.5 – Повторюваність символів

2. Відсортувати символи в порядку збільшення частоти. Вони зберігаються в пріоритетній черзі *Q* (див. рис. 2.6).

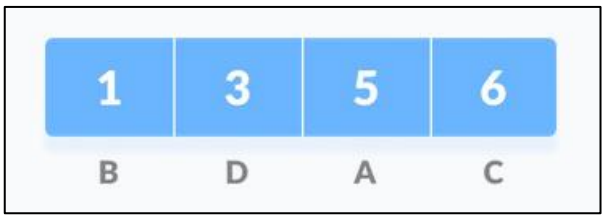



Рисунок 2.6 – Відсортовані символи

3. Зробити кожен унікальний символ листковим вузлом.

4. Сворити порожній вузол z. Призначити мінімальну частоту лівому нащадку z і призначити другу мінімальну частоту правому нащадку z. Встановити значення z як суму двох зазначених вище мінімальних частот.

5. Видалити ці дві мінімальні частоти з Q і додати суму до списку частот (* позначено внутрішні вузли на рис. 2.7).

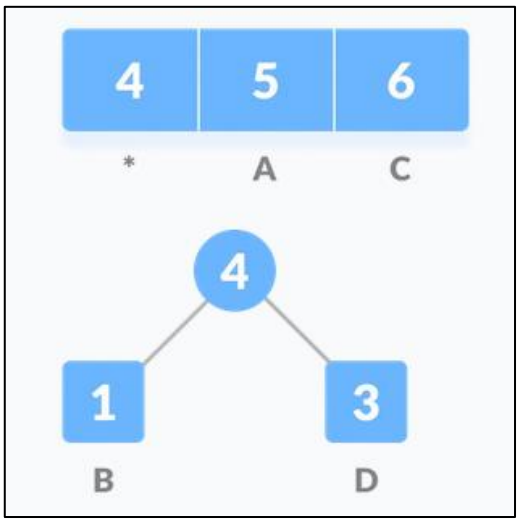



Рисунок 2.7 – Початковий рядок

6. Вставити вузол z в дерево.
7. Повторити кроки 3-5 для всіх символів (див. рис. 2.8).

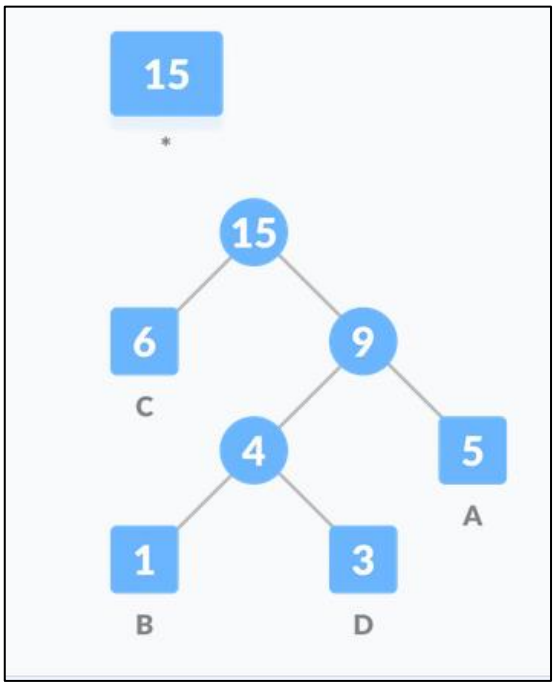



Рисунок 2.8 – Кроки 3-5 для всіх символів

8. Для кожного вузла призначити 0 лівому краю та 1 правому краю (див. рис. 2.8).

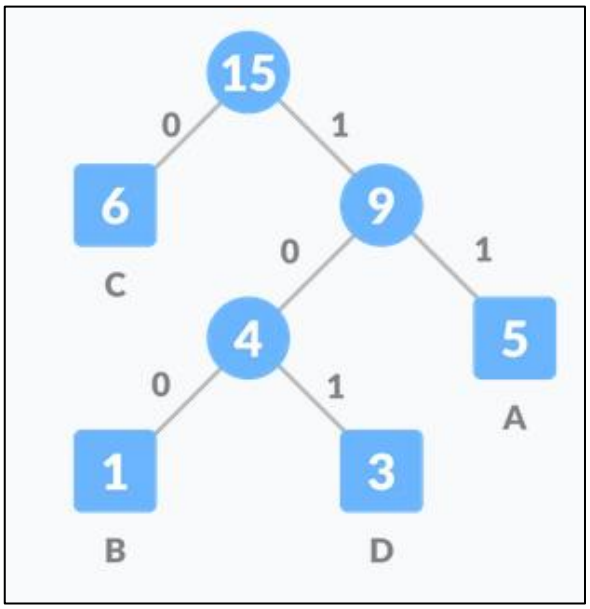



Рисунок 2.9 – Призначення 0 лівому краю та 1 правому краю

Щоб надіслати наведений вище рядок через мережу, потрібно спершу надіслати дерево, а також стиснутий код. Результуючий розмір повідомлення наведено в таблиці 2.1.

Таблиця 2.1 – Розмір повідомлення після стиснення

| **Символ** | **Повторюваність** | **Код** | **Розмір** |
|---|---|---|---|
| A | 5 | 11 | 5*2 = 10 |
| B | 1 | 100 | 1*3 = 3 |
| C | 6 | 0 | 6*1 = 6 |
| D | 3 | 101 | 3*3 = 9 |
| 4 * 8 = 32 біти | 15 біт | | 28 біт |

Без кодування загальний розмір рядка становив 120 біт. Після кодування розмір зменшується до 32 + 15 + 28 = 75.

Кодування Хаффмана використовується у форматах стиснення, таких як GZIP, BZIP2, PKZIP, TIFF, GIF, а також передачі тексту та факсу.

**Алгоритм стиснення LZW**, винайдений Абрахамом Лемпелем, Джейкобом Зівом і Террі Велчем у 1984 році, є одним із типів стиснення без втрат [31]. Це алгоритм на основі «словника», який сканує файл на наявність шаблонів даних, які з'являються більше одного разу. Потім ці шаблони зберігаються в словнику, а посилання розміщуються в стисненому файлі, де повторюються дані. У апаратних реалізаціях алгоритм простий і має потенціал для дуже високої пропускної здатності.

Алгоритм LZW ділиться на дві частини: алгоритм кодування, який перетворює рядки на цілі коди, і алгоритм декодування, який робить протилежне. Алгоритми як кодера, так і декодера мають таблицю або набір даних за замовчуванням, які служать початковою моделлю як для кодера, так і для декодера. Під час роботи алгоритму до цієї таблиці додаються нові цілочисельні коди для різних шаблонів рядків. Кількість записів у кодовій таблиці зазвичай дорівнює 4096. Коли починається кодування, кодова таблиця містить лише перші 256 записів, решта записів залишаються порожніми. LZW виявляє повторювані послідовності в даних і додає їх до таблиці кодів у міру просування кодування. Кожен код зі стисненого файлу декодується за допомогою кодової таблиці, щоб визначити, який символ або символи він представляє.

Вхідні дані:

123 145 201 4 119 89 243 245 59 11 206 145 201 4 243 245

Дані після кодування:

123 256 119 89 257 59 11 206 256 257

Формати: TIFF, GIF.

Метод LZW, подібно до RLE, проявляє кращу ефективність на зображеннях з однорідними ділянками кольорів. В порівнянні з RLE, він пропонує значно кращі результати при стисненні різноманітних графічних даних. Однак, процес кодування і декодування у методу LZW відбувається повільніше.

**Алгоритм JPEG** (Joint Photographic Experts Group) [32] - один з популярних растрових графічних форматів, що використовується для зберігання фотозображень та подібних до них зображень. Розроблено цей стандарт Об'єднаною групою експертів із фотографії ще 1991 року для ефективного стиснення зображень.

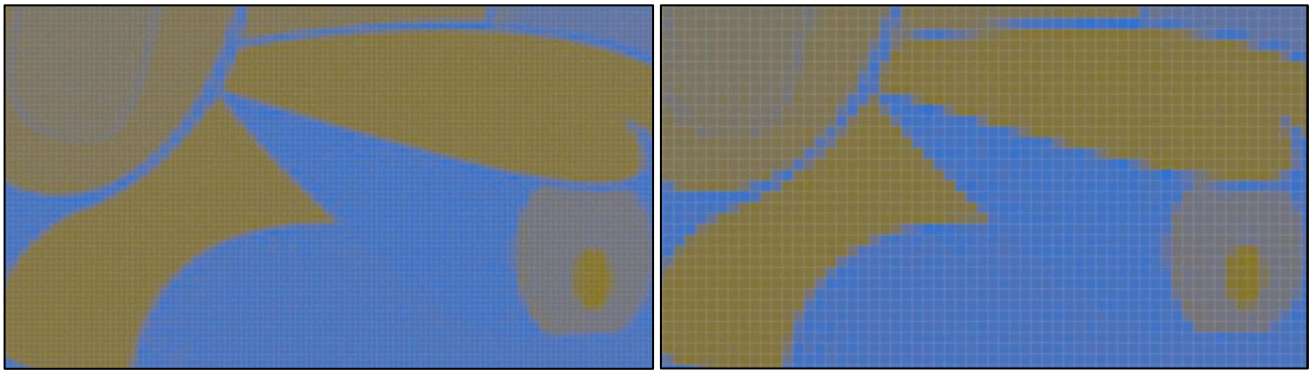
Дехто вважає, що JPEG-картинки - стислі методом Хаффмана сирі дані, але це не так. Перед контрольним стиском дані проходять довгий шлях.

Спочатку кольорову модель змінюють із RGB на YCbCr (для цього навіть є спеціальний алгоритм). Y не чіпають, тому що він відповідає за яскравість, і його зміна буде помітною.

*Проріджування*

Перше, що роблять із зображенням - це "проріджування" (subsampling). Береться 2x2 масив пікселів, далі беруться Cb і Cr — середні значення кожного компонента YCbCr цих 4 пікселів. Замість 4 Y, 4 Cb, 4 Cr ми отримали 4 Y і однакові для кожного з них Cb і Cr (4 + 4 + 4 = 12; 4 + 1 + 1 = 6; 12 - 6 = 6). Отже, ми виграли 6 байт. Це стосується всього зображення. І так скинули половину розміру. Такий прийом ми можемо використовувати завдяки нашому сприйняттю кольору. Людина з легкістю помітить різницю в яскравості, але не в кольорі, якщо вона усереднена в маленькому піскелевому блоці. Також проріджування може виконуватися в лінію, 4 пікселі по горизонталі та вертикалі. Перший варіант використовується найчастіше. Якщо важлива якість зображення, то проріджування не виконується взагалі.

Наочна ілюстрація проріджування зображена на рис. 2.10.



а) б)

Рисунок 2.10 – Початкове зображення (а), зображення після проріджения (б)

*ДКП*

Тепер найскладніша і найнеобхідніша частина. Вся картинка розбивається на блоки 8x8 (використовують заповнення у випадку, якщо роздільна здатність не кратна стороні блоку).

Тепер до кожного блоку застосовують ДКП. У цій частині з картинки виймають усе зайве. Використовуючи ДКП треба зрозуміти, чи описує даний блок (8x8) якусь монотонну частину зображення: неба, стіни; або він містить складну структуру (волосся, символи і т.д.). Не дивно, що 64 схожих за кольором пікселів можна описати всього одним, так як розмір блоку вже відомий. Тому стискання може бути 64 до 1.

*Квантування*

Тут вже використовується стиснення. Кожен з коефіцієнтів у кожній із матриць 8x8 ділиться на певне число. Якщо якість зображення після всіх його модифікацій ви більше не зменшуватимете, то дільник повинен бути одиницею. Якщо вам важливіше пам'ять, яку займає ця фотографія, то дільник буде більше 1, і частка округляється. Так виходить, що після округлення нерідко з’являється багато нулів.

Квантування роблять для створення ще більшого стиснення. Ось як це виглядає з прикладу квантування графіка y = sin(x):

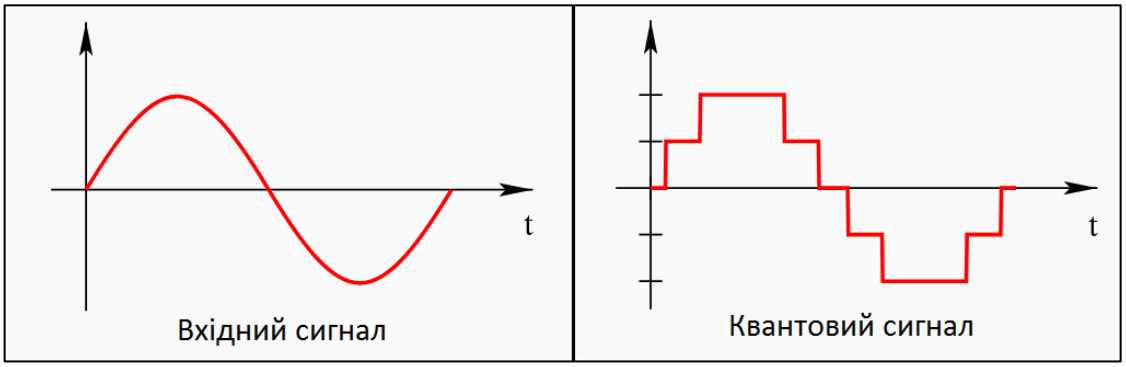



Рисунок 2.11 – Квантування графіка y = sin(x)

*Стиснення*

Спочатку проходимо по матриці зиг-загом:

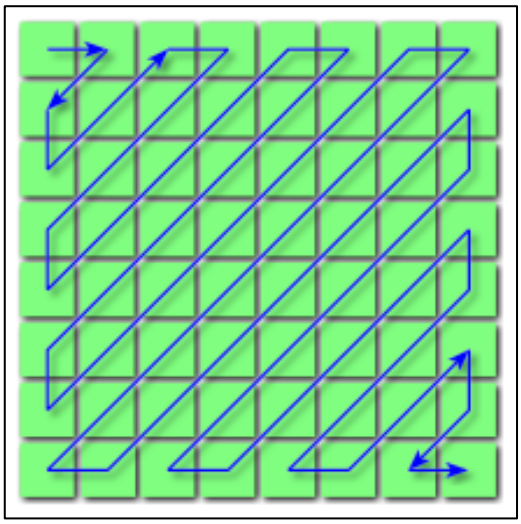


Рисунок 2.12 – Спектр «зиг-заг»

Отримуємо одновимірний масив з числами. Якщо в ньому багато нулів, їх можна забрати. Для цього замість послідовності з множини нулів ми вписуємо 1 нуль і після нього число, що позначає їх кількість у послідовності. Таким чином, можна скинути до 1/3 розміру всього масиву. А далі просто стискає цей масив методом Хаффмана та вписуємо вже у сам файл.

Формати: JPEG, PNG.

На даний момент знання про JPEG цінні лише з освітньою метою, адже він уже скрізь вбудований і оптимізований великими групами людей.

## 2.2 Використання стеганографічних методів для забезпечення безпеки спілкування.

Загалом, криптографія відповідає за безпеку спілкування. Однак, у випадках, коли потрібно не тільки забезпечити конфіденційність даних, але й приховати факт їх існування, краще скористатися стеганографічними методами. Розглянемо три наступні сценарії:

– самоспілкування (self-communication);
– один-на-один спілкування (one-to-one communication);
– один до багатьох спілкування (one-to-many communication).

### 2.1.3 Самоспілкування (self-communication)

Вимоги до безпеки самоспілкування включають наступні аспекти:

– непідозрілість спілкування - важливо, щоб комунікація не викликала підозри і була природньою;

– конфіденційність інформації - необхідно забезпечити захист від несанкціонованого доступу до передаваної інформації;

– дотримання законодавства - система повинна працювати в межах правових вимог і регуляцій.

– простота в користуванні - система має бути легкою у використанні для забезпечення зручного та ефективного спілкування.

Для безпечного отримання доступу до секретної інформації рекомендується зберігати інформацію локально. Це означає, що дані знаходяться на пристроях, які можуть бути переміщені з одного місця до іншого разом з користувачем. Однак, цей підхід також вносить ризик крадіжки пристрою та можливості доступу до секретної інформації третіми особами.

Один зі способів забезпечення портативності та конфіденційності є використання стеганографії для приховування інформації у зображеннях, а потім зберігання цих зображень на мобільних пристроях. Такий підхід дозволяє забезпечити переносимість інформації та збереження конфіденційності.

Щоб продемонструвати застосування стеганографії зображень для безпечного самоспілкування, можна взяти систему паролів. Паролі часто використовуються для захисту доступу до комп'ютерних систем, і віддалені користувачі зазвичай мають різні паролі для різних систем, які необхідно зберігати та використовувати за потребою. Однак паролі мають свої недоліки, такі як можливість отримання їх зловмисниками через сніффінг-атаку або інші методи [33]. Особливо поширеною є повторна атака, яка передбачає пасивне перехоплення паролів шляхом підслуховування. Якщо пароль не змінюється з моменту останнього використання, це дає зловмиснику можливість повторно використати його для аутентифікації.

Один зі способів захисту від повторних атак - використання одноразових паролів [34], де кожен пароль використовується лише один раз для аутентифікації. Щоб забезпечити безпечне зберігання та отримання одноразових паролів на мобільних пристроях, можна застосувати стеганографію для приховування цих паролів у зображеннях, які зберігаються на пристрої. Запропонована система забезпечує стійкість проти повторних атак за допомогою використання одноразових паролів і зберігає конфіденційність, приховуючи існування одноразових паролів. Портативність забезпечується через використання мобільних пристроїв для зберігання цих зображень і паролів.

У більшості випадків зображення на телефонах зберігаються у форматі JPEG, який використовує алгоритм стиснення з втратами. Це означає, що класичні методи просторової стеганографії не можуть бути застосовані. Алгоритм стиснення JPEG має етапи з втратами і без втрат. ДКП і фаза квантування є частиною етапу з втратами, тоді як кодування Хаффмана, що використовується для подальшого стиснення, не призводить до втрати інформації.

Оскільки стеганографія зазвичай приховує інформацію в надлишкових даних, а надлишкові дані видаляються під час етапу з втратами, просторова стеганографія не може бути використана під час ДКП та квантування. Тому краще застосовувати стеганографію між двома фазами для вбудовування повідомлення в

найменш значущі біти всіх ненульових коефіцієнтів ДКП перед застосуванням кодування Хаффмана. Іншими словами, замість вбудовування інформації в пікселі просторової області, інформація вбудовується в коефіцієнти ДКП у частотну область перетворення, що дозволяє зробити приховану інформацію невидимою.

### 2.1.4 Один-на-один зв’язок (one-to-one communication)

Забезпечення безпечної один-на-один комунікації передбачає встановлення зв'язку між одним відправником і одним отримувачем. Цей вид комунікації може мати дві потенційні вразливості: кінцеві точки (комп'ютери або пристрої) відправника і отримувача, а також передачу даних.

Для захисту кінцевих точок можна використовувати брандмауери і мережеву безпеку, щоб запобігти несанкціонованому доступу. Щодо передачі даних, шифрування зазвичай використовується згідно з безпечними протоколами. Однак деякі країни можуть обмежувати законність використання шифрування. В такому випадку користувачеві може знадобитись видалити програмне забезпечення для шифрування та дешифрування при в'їзді в ці країни. Після отримання надійного доступу до сервера або мережі, користувач може знову завантажити це програмне забезпечення в країнах, де його використання дозволено.

Замість постійного видалення та встановлення шифрувальних програм, можна приховати їх в зображеннях на мобільному телефоні. Якщо необхідно зберегти зашифроване повідомлення разом з програмним забезпеченням для його розшифрування, рекомендується використовувати різні параметри і алгоритми для приховування цих компонентів.

### 2.1.5 Один до багатьох зв’язок (one to many communication)

Під час спілкування "один до багатьох" відправник передає інформацію декільком отримувачам, які зазвичай є віддаленими користувачами. У такому випадку, коли інформація передається в різних напрямках, збільшується ризик підслуховування під час її транзиту. Важливо розрізнити "один до багатьох" спілкування від групового спілкування або захищеної багатоадресної передачі, які

є протоколами зв'язку, що дозволяють спілкуватися з декількома сторонами одночасно. Наприклад, розсилка секретної інформації агентством новин журналістам, які працюють у відповідній галузі, є прикладом "один до багатьох" спілкування.

У таких випадках важливо не тільки забезпечити конфіденційність інформації, але й захистити факт передачі цієї інформації до конкретних осіб. Якщо фігуранти справи підозрюють, що хтось намагається розкрити злочин, то журналісти, що працюють над цією справою, можуть опинитись у небезпеці. Тому методи стеганографії, які дозволяють приховати частину секретної інформації в зображеннях, можуть бути використані для вирішення цієї проблеми. Це дозволить не лише зберегти конфіденційність, але й зробити передачу інформації непомітною для неповинних осіб.

## Висновки до розділу 2

У другому розділі досліджено теоретичний аспект стеганографії, визначені основні формати зображень та представлено ряд методів для стеганографічного вбудовування. Для подальшого аналізу був обраний метод заміни найменш значущих бітів (LSB) та застосовано формат зображень BMP. Цей метод ґрунтується на заміні менш значущих бітів у на біти секретного повідомлення.

В розділі також наведено практичні приклади, де стеганографія застосовується для безпечного спілкування та передачі даних між різною кількістю учасників.

# 3 МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

## 3.1 Опис структури системи та вхідних даних

Інформаційна система буде будуватися на основі алгоритму LSB. У якості контейнера, для приховування повідомлення, буде використано зображення. Для додаткового захисту інформації будуть використовуватись методи шифрування, а саме шифр Цезаря або DES.

Вхідні дані до стеганосистеми включають:

– зображення - це контейнер, в якому буде приховано інформацію. Зображення може бути у різних форматах, наприклад, JPEG, PNG і т.д;

– повідомлення або дані - це інформація, яку потрібно приховати у зображенні. Це може бути текстове повідомлення, файл;

– ключ шифрування - це ключ, який використовується для симетричного шифрування або шифру Цезаря. Він використовується для зашифрування інформації перед приховуванням у зображенні, що забезпечує додатковий рівень захисту.

Процес стеганографії з використанням алгоритму LSB і шифрування може включати такі кроки:

– зчитування зображення-контейнера, у якому буде приховуватися інформація;

– шифрування повідомлення або даних з використанням вказаного методу шифрування і ключа шифрування;

– перетворення зашифрованого повідомлення на бітовий формат;

– введення бітів зашифрованого повідомлення у менш значущі біти пікселів зображення-контейнера. Це здійснюється шляхом заміни менш значущих бітів кожного пікселя зображення бітами повідомлення;

– збереження модифікованого зображення, яке містить приховану інформацію.

Процес розшифрування і отримання прихованої інформації включає обернені кроки:

– зчитування модифікованого зображення з прихованою інформацією;

– витягування менш значущих бітів пікселів зображення, які були використані для приховування бітів повідомлення;

– розшифрування отриманих бітів з використанням відповідного методу розшифрування і ключа;

– отримання розшифрованої інформації, яка була прихована у зображенні.

### 3.2 Метод LSB

Використання методу вставки найменш значущого біта є простим та поширеним підходом до приховування інформації в контейнері. Можна розглянути структуру такого контейнера на прикладі файлу зображення у форматі BMP. Файл BMP умовно можна поділити на 4 частини:

– заголовок файлу;

– заголовок зображення;

– палітра;

– зображення.

Перші 2 байти заголовка файлу - це сигнатура BM за якою комп'ютерна програма визначає, що це саме BMP файл зображення. Далі в чотирьох наступних байтах записано розмір файлу. Наступні чотири байти зарезервовані та повинні містити нулі. В наступній послідовності чотирьох байтів вказано зміщення від початку файлу до байтів, які представляють зображення. У файлі формату BMP з розмірністю 24 біти кожен піксель кодується трьома байтами RGB (Red, Green, Blue).У методі стеганографії LSB (від англ. Least Significant Bit - найменш значущий біт) біти коду секретного повідомлення поміщають замість молодших бітів (останніх, найменших, на схемі 2.14 виділено сірим) в байтах, які відповідають за кодування кольорів. Зміна відтінку кольору пікселя при заміні останнього біта палітри на «0» або «1» буде майже непомітна людському оку.

Подання байтів, які відповідають за кодування кольору до внесення стеганограми (див. рис. 3.1).

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | R = 195 | |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | G = 32 | |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | B = 102 | |

Рисунок 3.1 – Подання байтів до внесення стеганограми

Подання байтів, які відповідають за кодування кольору після внесення стеганограми в останні значущі біти (див. рис. 3.2).

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | R' = 194 | |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | G' = 33 | |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | B' = 103 | |

Рисунок 3.2 – Подання байтів після внесення стеганограми

Максимальна ємність 24-х бітного BMP- файлу, як стегоконтейнера, при використанні останніх бітів всіх байтів, відповідальних за кодування трьох кольорів, становить одну восьму від обсягу файлу зображення. Наприклад, в файл зображення розміром 1080 на 1080 пікселів (це максимальний розмір картинки для публікації в популярній соціальної мережі Instagram) можна помістити стеганограму дуже великого тексту.

Розрахунки такі: 1080 x 1080 x 3 x 1/8 = 437400.

Стільки байтів текстової інформації можна помістити в картинку в Instagram, а саме 437 400 символів.

Якщо для стеганограми використовувати тільки останні біти байтів, що відповідають за передачу палітри синього кольору (Blue), то це знизить ємність файлу зображення, як стегоконтейнера, втричі, але помітити людським оком наявність стеганограми в такому зображенні навіть зіставивши поряд оригінал без

стеганограми буде практично неможливо. Це все тому, що людське око найгірше розрізняє відтінки синього кольору.

## 3.3 Методи шифрування

### 3.3.1 Шифр Цезаря з ключем

Шифр Цезаря — це проста техніка шифрування, яку Юлій Цезар використовував для надсилання секретних повідомлень своїм союзникам. Він працює шляхом зміщення літер у текстовому повідомленні на певну кількість позицій, відомих як «shift» або «key».

Техніка шифру Цезаря є одним із найперших і найпростіших методів техніки шифрування. Це просто тип шифру підстановки, тобто кожна літера певного тексту замінюється літерою з фіксованою кількістю позицій у алфавіті. Наприклад, зі зсувом на 1 A буде замінено на B, B стане C і так далі.

Таким чином, щоб зашифрувати текст, потрібне ціле число, відоме як зсув, яке вказує на кількість позицій, на яку кожна літера тексту була переміщена вниз.

Шифрування можна представити за допомогою модульної арифметики, спочатку перетворивши літери в числа за схемою A = 0, B = 1,…, Z = 25. Наприклад, якщо зсув дорівнює 3, то буква A буде замінена літерою D, B стане E, C стане F і так далі (див. рис. 3.3). Алфавіт обертається так, що після Z починається знову з A.

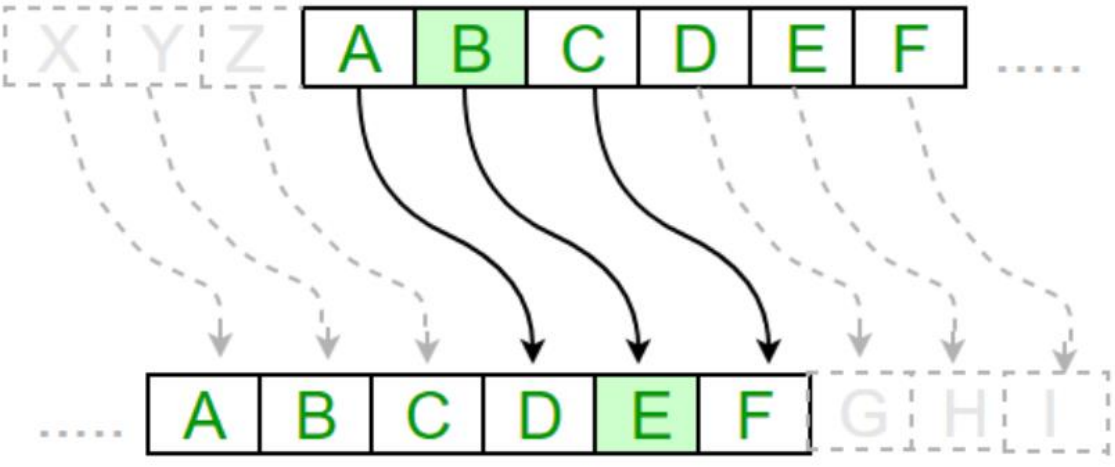



Рисунок 3.3 – Алфавіт, якщо зсув дорівнює 3

Ось приклад використання шифру Цезаря для шифрування повідомлення «HELLO» зі зсувом 3:

– записати повідомлення відкритим текстом: HELLO;

– вибрати значення зсуву (у цьому випадку використовується зсув 3);

– замінити кожну літеру в текстовому повідомленні літерою, яка займає три позиції праворуч в алфавіті (H стає K (зсув 3 від H), E стає H, L стає O, O стає R);

– зашифроване повідомлення тепер «KHOOR».

Щоб розшифрувати повідомлення, потрібно просто зрушити кожну літеру назад на таку ж кількість позицій. У цьому випадку ви б зрушили кожну літеру в «KHOOR» назад на 3 позиції, щоб отримати вихідне повідомлення «HELLO».

### 3.3.2 Симетричне шифрування DES з ключем

DES - це блоковий шифр, який шифрує дані в блоках розміром 64 біти. Це означає, що 64 біти звичайного тексту надходять як вхідні дані для DES, який створює 64 біти зашифрованого тексту. Для шифрування та дешифрування використовуються той самий алгоритм і ключ, з незначними відмінностями. Довжина ключа становить 56 біт.

Насправді початковий ключ складається з 64 бітів. Однак перед початком процесу DES кожен 8-й біт ключа відкидається для створення 56-бітного ключа. Тобто бітові позиції 8, 16, 24, 32, 40, 48, 56 і 64 відкидаються.

DES базується на двох фундаментальних атрибутах криптографії: заміні (також називається плутаниною) і транспозиції (також називається дифузією). DES складається з 16 кроків, кожен з яких називається раундом. Кожен раунд виконує кроки заміни та транспозиції. Тепер наглядно розглянемо ці кроки.

Генерація ключа - визначається 56-бітовий ключ, який використовується для шифрування й дешифрування даних. Ключ може бути згенерований випадковим чином або визначений користувачем.

Розширення ключа - початковий 56-бітовий ключ розширюється до 64 бітів шляхом додавання контрольних бітів. Це забезпечує більшу стійкість шифру.

Розбиття даних - перед шифруванням дані, які потрібно зашифрувати, розбиваються на блоки фіксованого розміру. Зазвичай цей розмір становить 64 біти.

Початкова перестановка - кожен блок даних проходить початкову перестановку, яка перегруповує біти в блоку.

Раунди шифрування - DES використовує 16 раундів шифрування для забезпечення надійності. Кожен раунд включає в себе наступні етапи:

– розбиття блоку даних на ліву і праву половини;

– розширення правої половини до 48 бітів за допомогою таблиці розширення;

– порівняння розширеної правої половини з раундовим ключем, що визначається згідно з ключем DES;

– застосування логічної операції XOR між результатом порівняння та лівою половиною блоку даних;

– передача лівої половини наступній ітерації раунда, а праву половину об'єднується з лівою половиною попереднього раунда;

– повторення кроків для кожного раунда шифрування.

Фінальний обмін - після останнього раунда шифрування ліва і права половини блоку даних обмінюються місцями.

Остання перестановка - зашифрований блок даних проходить останню перестановку, яка є інверсією початкової перестановки.

Отримання зашифрованих даних - отримані змінені блоки даних об'єднуються для отримання кінцевого зашифрованого тексту.

Ці кроки дозволяють зашифрувати дані за допомогою симетричного шифрування DES з використанням відповідного ключа. При дешифруванні дані проходять через аналогічні кроки, але в зворотному порядку.

## 3.4 Методи стеганоаналізу

### 3.4.1 «Хі-квадрат»

Метод хі-квадрат - це статистичний метод, який використовується для порівняння спостережуваних і очікуваних результатів. Мета цього методу - визначити, чи є розбіжність між фактичними та прогнозованими даними випадковістю чи зв’язком між змінними, що розглядаються.

У даному випадку використовується аналіз гістограми зображення, яка отримується на основі його елементів, а також оцінка розподілу пар значень у цій гістограмі. Для BMP файлів пари значень формуються на основі значень пікселів зображення, а для JPEG - квантованих коефіцієнтів ДКП, які відрізняються LSB. Молодші біти зображень не є випадковими, і частоти двох сусідніх елементів мають розташовуватися досить далеко від значення частоти середнього арифметичного цих елементів.

Атака $\chi^2$ базується на пошуку близьких значень частот та підрахунку ймовірності вкраплення інформації на основі розташування значень частот парних і непарних елементів контейнера. Метод $\chi^2$ є універсальним, оскільки може застосовуватися до аналізу зображень, в які інформація вкраплювалася за допомогою різних стеганографічних програм. Проте результати цього методу значно залежать від самого методу приховання даних. При послідовній заміні LSB елементів контейнера (див. рис.3.4) і вкрапленні повідомлення з заповненням (див. рис.3.5), метод $\chi^2$ виявляє наявність прихованих даних, але в разі псевдовипадкового вибору молодших бітів (розподілене вкраплення) метод не ефективний.

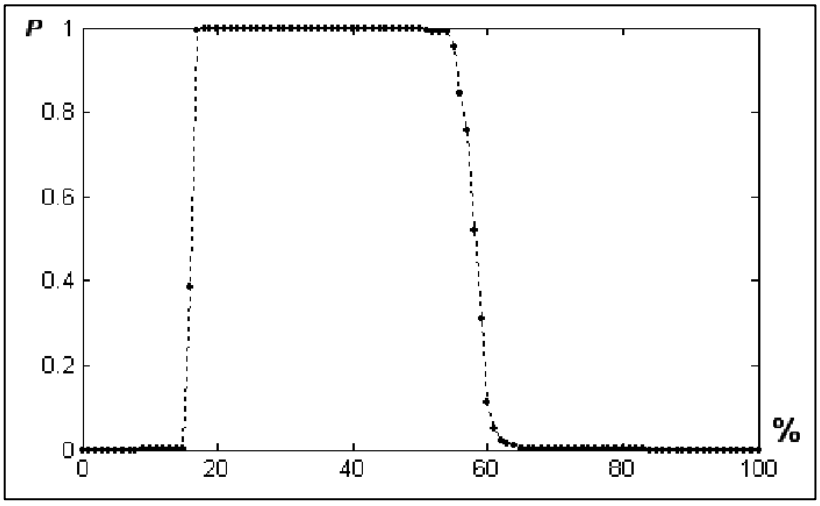



Рисунок 3.4 – Послідовне вкраплення за критерієм $\chi^2$

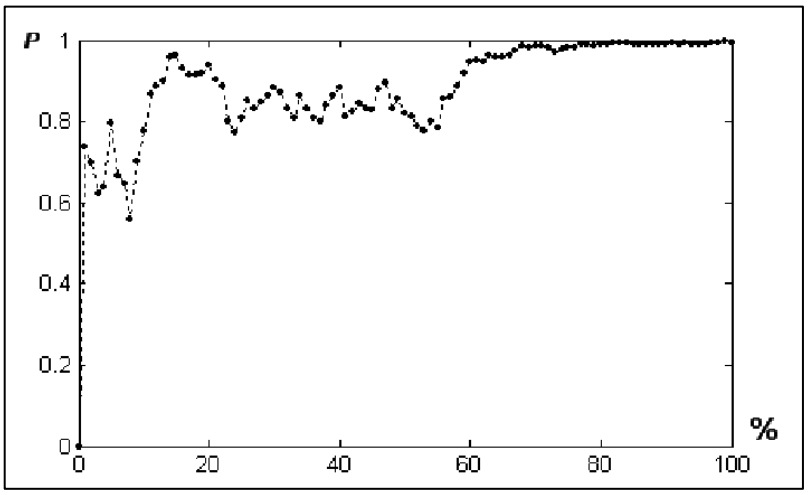



Рисунок 3.5 – Вкраплення із заповненням за критерієм $\chi^2$

### 3.4.1 RS-аналіз

Ученими під керівництвом Дж. Фрідріха в 2001 році був розроблений метод RS (Regular-Singular) - один з оригінальних методів статистичного стегоаналізу. Цей метод полягає в розбитті зображення на групи пікселів, де кількість пікселів у групі (n) є парним числом, наприклад, по 2 пікселя поруч по горизонталі. Для кожної групи пікселів визначається функція регулярності або "гладкості" f(G), яка може бути обчислена, наприклад, як дисперсія значень всередині групи або сума перепадів значень суміжних пікселів. Значення пікселя вважається цілим числом в діапазоні від 0 до 255.

Метод RS ґрунтується на статистичному припущенні, що природні зображення, які є незаповненими контейнерами, мають подібний розподіл значень пікселів до оригінального зображення, але зсунуті на одиницю. У звичайних зображеннях співвідношення між групами пікселів повинно залишатися майже незмінним. Значна розбіжність між цими значеннями свідчить про застосування LSB-стеганографії для вбудовування даних у молодші біти зображення.

При 100% перезаписі молодших біт зображення повідомленням, половина з цих бітів буде інвертована. На діаграмі для типового зображення (див. рис. 3.6) на осі абсцис відображається кількість інвертованих бітів x, а на осі ординат - відносні значення регулярних і сингулярних груп відносно загальної кількості груп у зображенні.

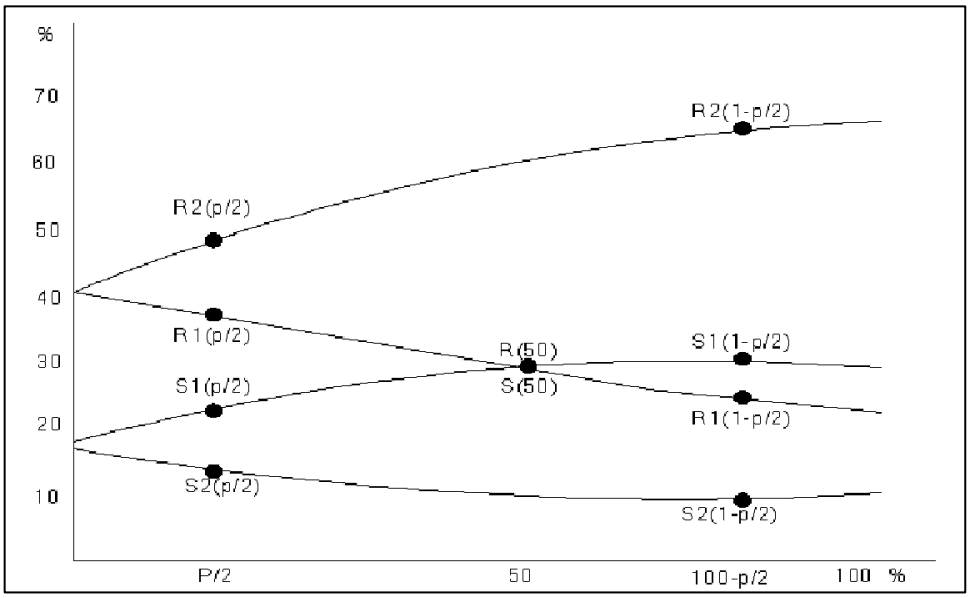



Рисунок 3.6 – Діаграмі для типового зображення

За умови, що в зображенні внесено повідомлення довжиною p біт, при цьому 50% з нижніх бітів, використаних для запису повідомлення, інвертувалися, значення статистики досягне точки p/2. Якщо всі нижні біти зображення інвертувати й перерахувати статистику, на діаграмі це буде відображено точками на кривих при значенні x = 100 - p/2. Повна рандомізація нижньої бітової площини відповідатиме точці 1/2.

У випадку дуже зашумлених і зображень з дрібною текстурою, різниця між кількістю регулярних і сингулярних груп у контейнері буде невеликою. Це означає, що лінії на RS-діаграмі будуть перетинатися під невеликим кутом, і точність методу RS-стегоаналізу зменшиться. Метод RS-стегоаналізу є більш точним для повідомлень, де біти стегоповідомлення розташовані випадково в площині стегоконтейнера, ніж для повідомлень, в яких вбудовання відбувається локально.

**Висновки до розділу 3**

У третьому розділі надано опис структури системи та вхідних даних, що включає основні компоненти та процеси, необхідні для функціонування системи. Далі розглянуто метод LSB, який є одним з найпоширеніших методів вбудовування прихованої інформації у цифрові зображення.

Для забезпечення додаткового рівня безпеки розглянуто методи шифрування– шифр Цезаря з ключем і симетричне шифрування DES з ключем.

Також розглянуто методи стеганоаналізу, а саме Хі-квадрат" і RS-аналіз, які використовуються для виявлення наявності та витягнення прихованої інформації зі стеганографічних зображень.

# 4 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ РОЗРОБЛЕНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

## 4.1 Засоби розробки

Для вирішення поставленої задачі було обрано мову програмування C#. Чому саме C#? Він зручний та популярний через свою «простоту». Простота грає велику роль у створенні програмного продукту, адже у невеликі терміни ми можемо створювати функціональні і продуктивні застосунки. Цьому сприяють нетипові конструкції мови і специфічний синтаксис, що допомагає максимально органічно реалізувати намічені цілі.

Мова програмування C# надає низку функцій, які сприяють створенню надійних і стійких додатків. C# має вбудовану систему збору мусора, яка автоматично вивільняє пам'ять, зайняту недоступними невикористаними об'єктами. Це дозволяє уникнути проблем з утриманням непотрібної пам'яті і сприяє ефективному використанню ресурсів. Дозволяє використовувати типи, які можуть мати значення null. Це допомагає захистити від змінних, які не посилаються на виділені об'єкти, тим самим запобігаючи помилкам виклику нульового посилання.

C# надає структурований і розширений підхід до обробки виключень. Це дозволяє виявляти помилки і відновлюватися після них, спрощуючи процес розробки стійких додатків. Також підтримує лямбда-вирази, що дозволяють використовувати прийоми функціонального програмування. Це забезпечує більш компактний код, зокрема при роботі з делегатами та обробці подій. Синтаксис LINQ створює загальний шаблон для роботи з даними з будь-якого джерела. Це спрощує роботу з колекціями, запитами до баз даних та іншими джерелами даних. Ще C# має підтримку мов для асинхронних операцій, що дозволяє створювати розподілені системи та ефективно працювати з асинхронним кодом.

В C# існує єдина система типів, де всі типи, включаючи типи-примітиви, належать від одного корінного типу об'єкта. Це спрощує роботу з типами,

забезпечує їх зручну обробку та зберігання. C# надає ітератори, які дозволяють розробникам класів колекцій визначати користувацькі варіанти поведінки для клієнтського коду [35].

Однією з переваг C# є можливість використовувати готові бібліотеки для розширення функціональності. Ці бібліотеки доступні через вбудований менеджер пакетів NuGet Package Manager у середовищі розробки Visual Studio 2019. Це дає змогу швидко і легко додавати нові функції до проєкту без необхідності писати все з нуля.

C# є крос-платформною, це означає, що програми, написані на C#, можуть запускатись на різних операційних системах, таких як Windows, UNIX і Macintosh. Це надає більшу гнучкість і доступність для розробників.

C# об'єднує найкращі ідеї сучасних мов програмування, таких як Java, C++, Visual Basic і багатьох інших. Вона не просто комбінує їх достоїнства, але є мовою нового покоління. Це означає, що C# пропонує вдосконалені засоби програмування, які враховують сучасні потреби розробників і забезпечують високу продуктивність і ефективність роботи.

Для реалізації користувацького інтерфейсу було обрано Windows Forms. Це структура для настільних додатків Windows, яка була представлена одночасно з самим .NET ще в 2002 році. З тих пір вона, безумовно, переглядалася багато разів. У 2006 році відбулось велике оновлення .NET 2.0. Хоч і Microsoft задумала замінити його на Windows Presentation Foundation (WPF), Windows Forms не втрачає своєї популярності, широко використовується.

Windows Forms пропонує просту модель програмування, що робить його легким у використанні, навіть для початківців. Розробка інтерфейсу здійснюється за допомогою методу "перетягни і випусти", де розробник може вибрати потрібні елементи управління з палітри і розмістити їх на формі. Після цього можна налаштувати властивості елементів та визначити їх поведінку за допомогою подій і методів.

Windows Forms також підтримує механізм візуального проєктування, що дозволяє розробникам переглядати та змінювати інтерфейс безпосередньо у середовищі розробки. Це спрощує процес розробки, дозволяючи швидко тестувати та вносити зміни до інтерфейсу.

Середовищем для розробки додатку було обрано Visual Studio 2019.

### 4.2 Вимоги до технічного забезпечення

Для належної роботи даного програмного продукту необхідно мати певну технічну і програмну конфігурацію. До складу технічних засобів повинні наступні параметри.

Комп'ютер з такими характеристиками:

– процесор з тактовою частотою не менше 1 ГГц;

– дискова підсистема об'ємом не менше 40 ГБ;

– достатній обсяг оперативної пам'яті - не менше 512 МБ;

– інші компоненти комп'ютера можуть мати будь-які параметри, оскільки вони не мають суттєвого впливу на роботу програми.

Додатково необхідне наступне програмне забезпечення:

– операційна система Windows XP/Vista/Windows 7/Windows 10 або інша сумісна;

– встановлений .NET Framework версії 3.5 або вище.

Комп'ютерна периферія, яка включає:

– монітор;

– мишку;

– клавіатуру.

Інтерфейсний рівень є ключовою складовою програмного продукту, оскільки він забезпечує взаємодію користувача з програмою. У даному випадку для відображення даних на стороні клієнта використовується графічний інтерфейс у форматі Windows Forms.

### 4.3 Розробка програмного застосунку

Для зображень різного формату використаний метод заміни найменш значущого біта для приховування інформації. Цей метод не є надто стійким до різних видів спотворень контейнера, а також до його стиснення, оскільки такі операції можуть призвести до втрати прихованої інформації. Крім того, використання методу заміни найменш значущого біта є вразливим до стеганоалітичних атак, які можуть виявити наявність і природу прихованої інформації.

Для забезпечення більшого рівня захисту прихованої інформації у зображенні, яке використовує метод заміни, можна використовувати криптографічні методи. Перед вкрапленням інформації у зображення, повідомлення можна зашифрувати з використанням криптографічних алгоритмів, такими як шифр Цезаря (див. додаток А) та DES (див. додаток Б). Також додатково стегозображення можна приховати у інше звичайне зображення, що додасть це один рівень захисту.

В даній роботі у користувача є можливість самостійно обирати, скільки саме бітів повідомлення замінювати на найменш значущі біти пікселя. Вибір від 1 до 4. При впровадженні в 1 LSB (див. додатки В,Г) зображення, значення одного біта повідомлення замінюється на один найменш значущий біт пікселя. Це означає, що для кожного пікселя використовується лише один біт повідомлення, тим самим стеганографічна ємність зменшується, але зображення залишається візуально незмінним. При впровадженні в 4 LSB зображення, значення всіх чотирьох бітів повідомлення замінюються на всі чотири найменш значущих біти пікселя. Це найбільша стеганографічна ємність, але це також може вплинути на візуальну якість зображення і стати помітним для людського спостереження.

Це також забезпечує додатковий рівень захисту інформації. Навіть якщо стане відомий сам факт приховання інформації, залишається невідомим скільки саме бітів замінювались на біти повідомлення.

Після завантаження зображення, спочатку виконується обчислення корисної ємності контейнера. Введений користувачем текст, спочатку шифрується а потім перетворюється на бінарний формат, використовуючи кодування UTF-8, де кожен символ представлений різною кількістю байтів. Наприклад, українські літери кодуються двома байтами (16 біт), деякі спеціальні символи, які використовуються українською мовою (наприклад, ґ, є, ї, й, ю, я), кодуються трьома байтами, а латинські літери, арабські цифри і розділові знаки - одним байтом.

Потім зображення розбивається на компоненти R, G і B та замінюються за вивченим користувачем порядком.

Для розрахунку ставлення пікового сигналу до шуму використовуються змінені елементи зображення. Для створення зображення з впровадженою інформацією створюється копія завантаженого зображення, до якої вносяться необхідні зміни. Початкове зображення залишається оригінальним, а зображення з впровадженою інформацією зберігається окремо.

Для витягу даних зі стеганоконтейнера виконуються ті ж самі етапи, що і при впровадженні даних, але у зворотному порядку (див. додатки Д,Е). Компоненти всіх пікселів зображення записуються у вектор в порядку R, G, B. Обчислюється довжина повідомлення, яка записана на початку масиву повідомлення і може бути витягнута. Після цього можна витягнути всі повідомлення.

Блок – схему роботи системи представлено у додатку Є.

### 4.4 Демонстрація та тестування програмного забезпечення

Для початку використання програмного запезпечення потрібно відкрити файл застосунку, який знаходиться у папці проекту. Після цього відкривається перше вікно, в якому відображені інструменти для подальшої роботи (див. рис. 4.1).

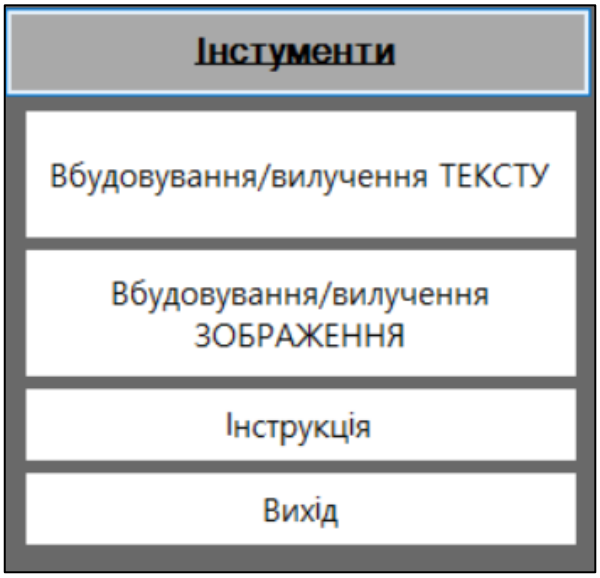



Рисунок 4.1 – Інструменти для роботи з системою

Це вікно включає в себе:

– «Вбудовування/вилучення ТЕКСТУ» - приховування/вилучення тестового повідомлення у/із зображення;

– «Вбудовування/вилучення ЗОБРАЖЕННЯ» - приховування або вилучення одного зображення у інше, або з іншого;

– «Інструкція» - покроковий опис роботи із стегостемою (див. рис. 4.2) ;

– «Вихід».

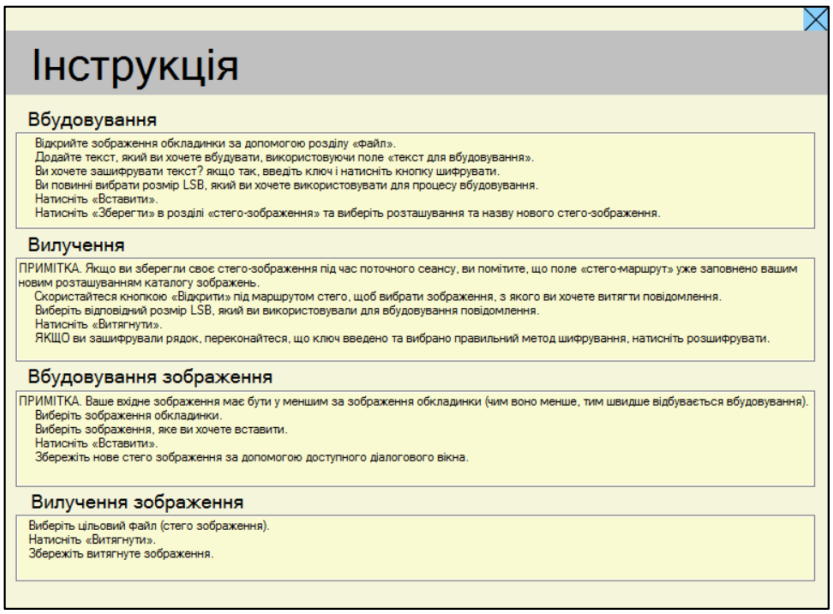



Рисунок 4.2 – Вікно з детальним описом користування системою

Обираємо вікно для приховування та вилучення тексту (див. рис. 4.2). У ньому ми маємо можливість завантажувати контейнер, після цього виводиться інформація про обране зображення. Далі можна обрати шифрування та ввести ключ. Обрати кількість бітів, у які буде прихована інформація. Також є можливість вилучення інформації, обравши стегоконтейнер та зробивши всі дії у зворотному порядку.

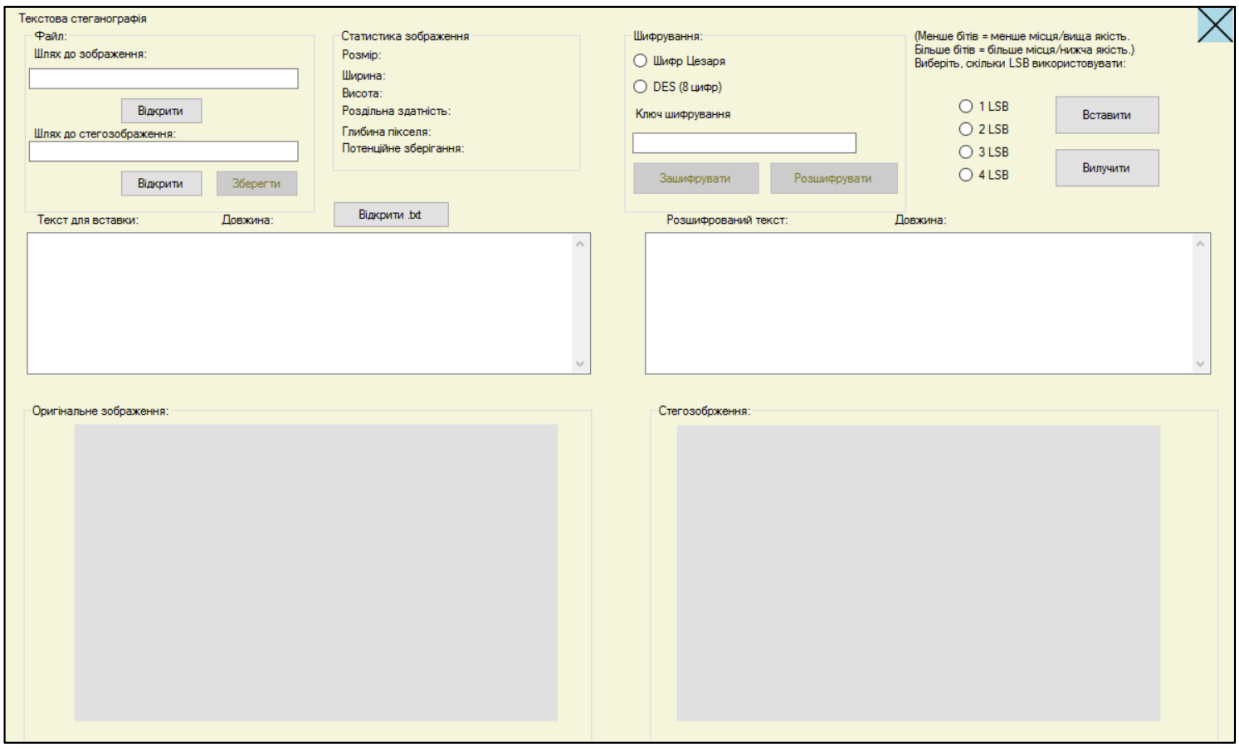



Рисунок 4.3 – Вікно для приховування та вилучення тексту

Спочатку необхідно завантажити файл-контейнер, натиснувши на кнопку «Відкрити». Після цього вписуємо у полі під написом «Текст для вставки» або завантажуємо, натиснувши кнопку «Відкрити .txt», необхідне повідомлення (див. рис. 4.4).

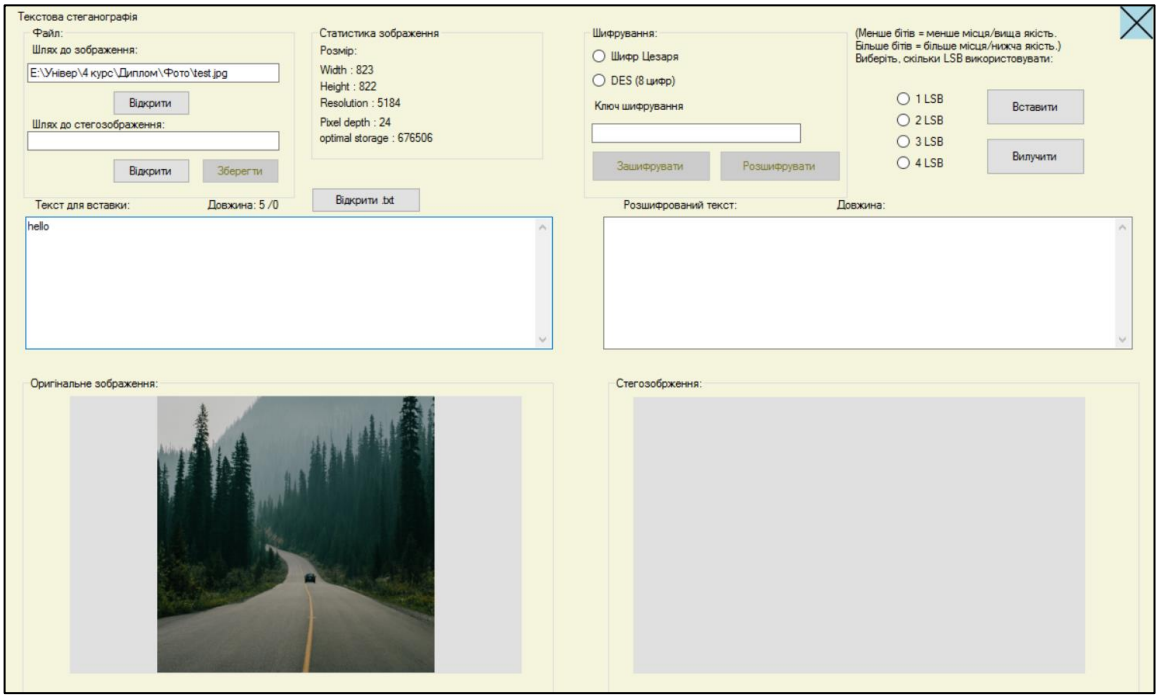



Рисунок 4.4 – Обрання контейнера та введення повідомлення

Обираємо шифрування, натискаємо «Зашифрувати». Текст тепер представено у зашифрованому вигляді. Далі обираємо кількість наймених бітів та кнопку «Вставити». З’являється стеганозображення разом із прихованим повімленням. Приклад кінечного результату зображено на рис. 4.5.

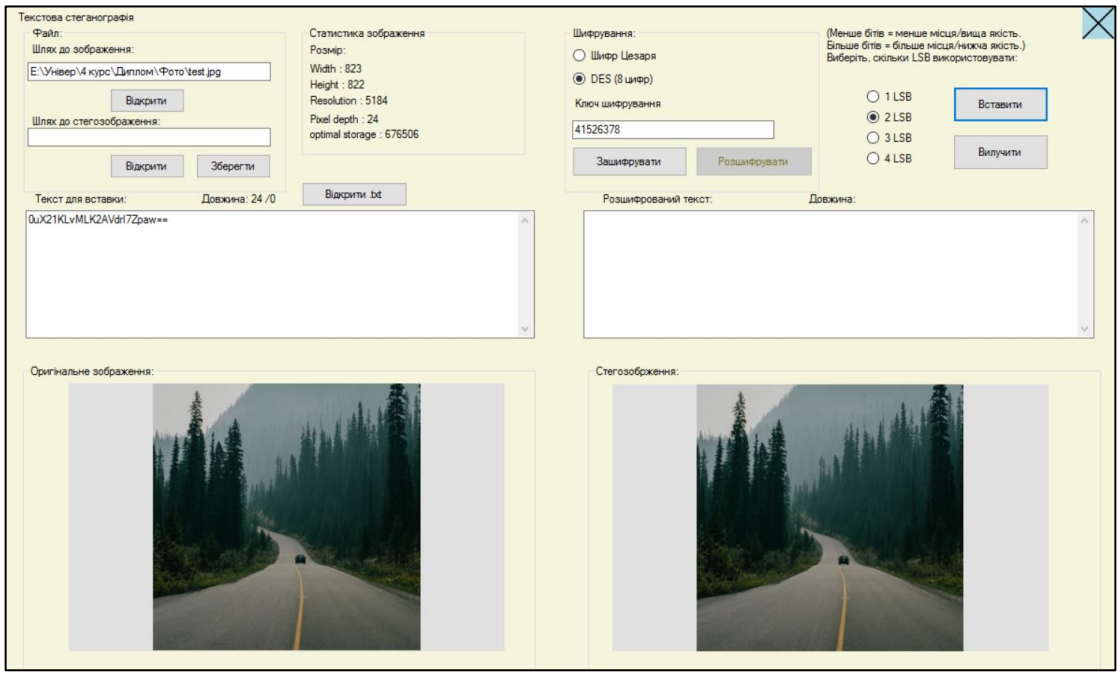



Рисунок 4.5 – Шифрування та вставка повідомлення

Для того, щоб зберегти це зображення, достатньо натиснути «Зберегти» та підписати новостворений файл. Це продемонстровано на рис. 4.6.

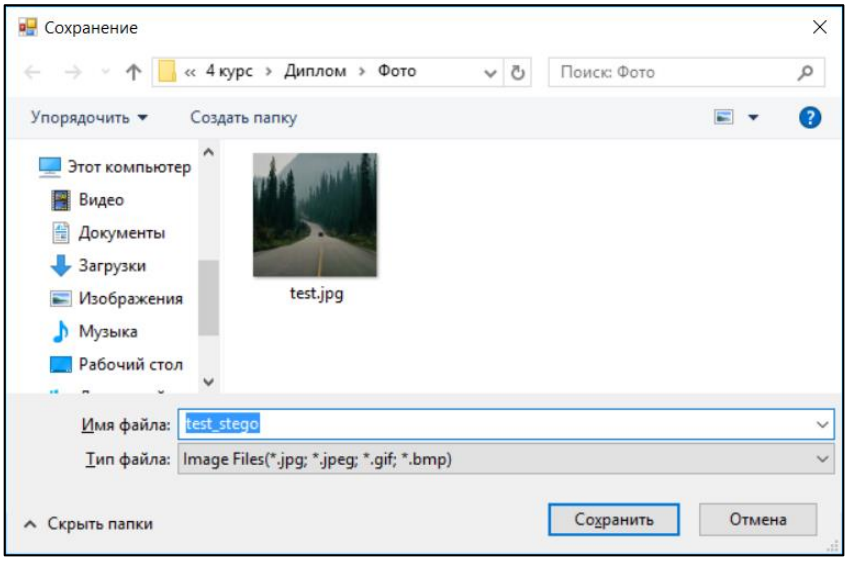



Рисунок 4.6 – Збереження файлу

Після збереження у вікні «Шлях до стегозображення» з’являється шлях до створеного стегозображення (див. рис. 4.7). Одразу можна робити вилучення даних. Якщо це потрібно зробити з іншого зображення, достаньо відкрити його, натиснувши «Відкрити».

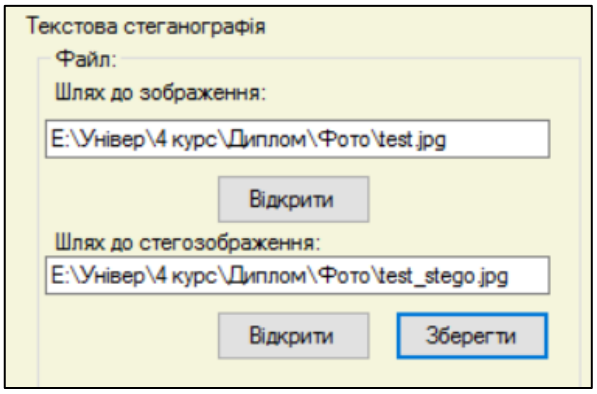



Рисунок 4.7 – Шлях до файлу

Для вилучення робимо все навпаки. Спочатку обираємо кількість бітів та натискаємо «Вилучити» як зображено на рис. 4.8. Далі обираємо шифрування та вводимо ключ і «Розшифрувати» (див. рис. 4.9). Вилучення пройшло успішно.

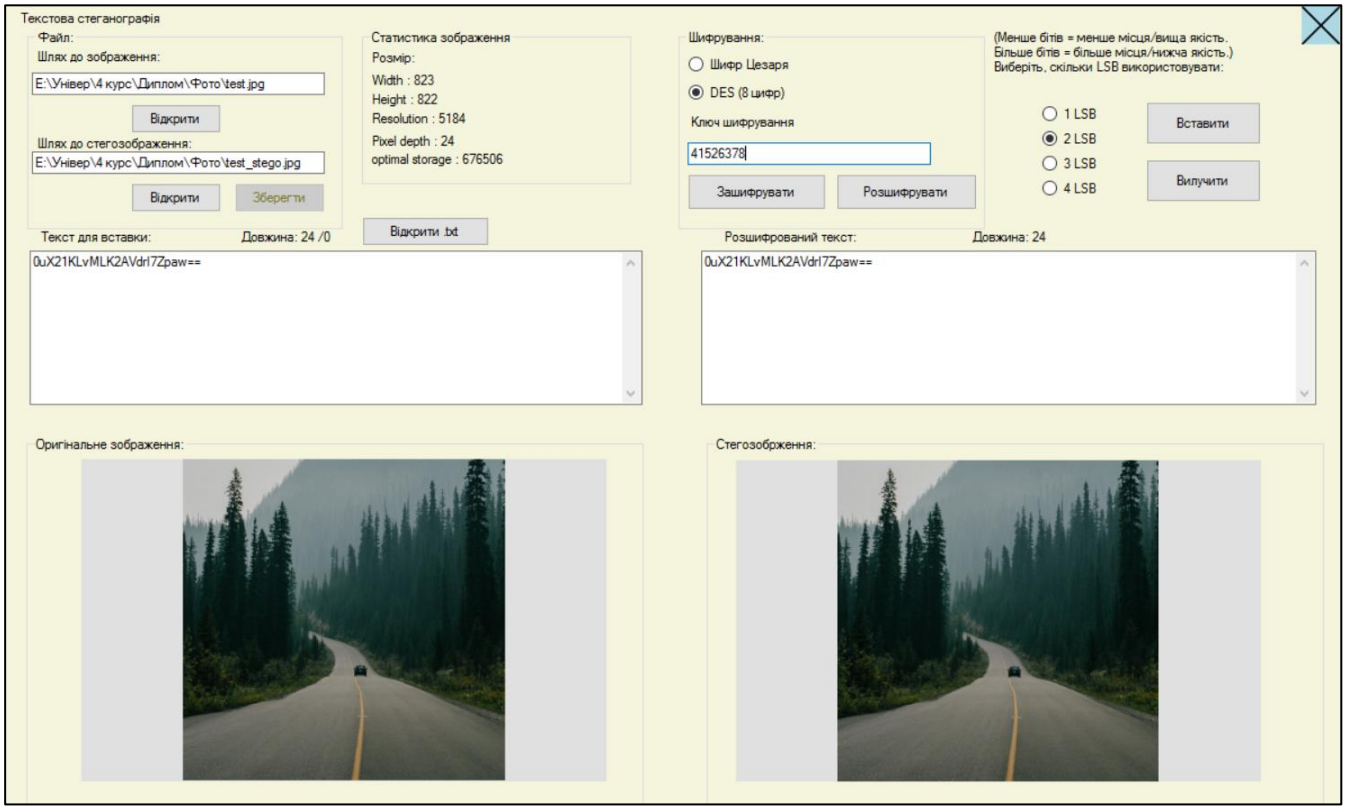



Рисунок 4.8 – Вилучення даних

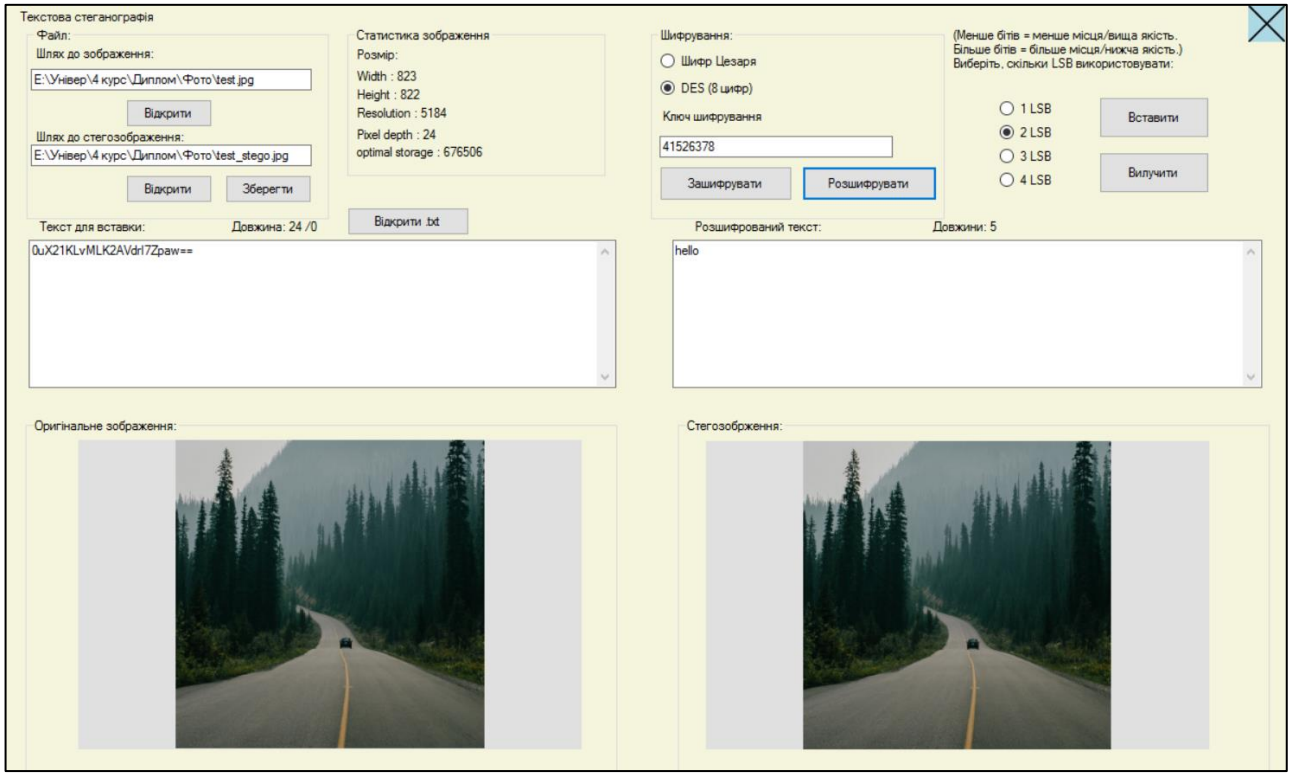



Рисунок 4.9 – Розшифрування даних

Також в системі реалізовано стеганографію зображення. Такий вид стеганографії можна використовувати коли повідомленням є не текст, а власне картинка. Вона може мати свої особливості та обмеження, які можуть зробити її менш ефективною порівняно зі стеганографією тексту. Зображення, зазвичай, мають більший об'єм даних порівняно з текстом. Це означає, що можна втрачати більше простору для вбудовування повідомлень без помітних змін у зображенні. Таким чином, ємність стеганоконтейнера може бути обмежена, і можливість вбудовувати багато інформації може бути скорочена.

Вбудовування додаткових даних в зображення може призвести до візуальних артефактів або змін, які можуть бути помітні людському оку. Це особливо важливо для використання стеганографії у вимогливих візуально середовищах, наприклад, в обробці зображень або відео, де навіть незначні зміни можуть бути помітними.

Зображення може бути вразливим до різних атак, спрямованих на виявлення або вилучення прихованої інформації.

Але якщо такий спосіб використати для зображення, в яке було вже вбудовано зашифроване повідомлення, створюється подвійна стеганографія. Цей підхід може підвищити рівень захисту, оскільки для розкриття прихованої інформації потрібно буде виявити обидва шари стеганографії.

Для стеганографії зображення  є окреме вікно, воно показане на рис. 4.10. Важливо дотримуватися правила, що картинка для вставки повинна бути меншою за обкладинку. Також, чим більші за розмірами зображення і чим більше бітів обрано, тим довше буде відбуватися процес стенографії і візуально буде сильніше спотворення.

Сама форма складається з обкладинки, зображення, яке буде вставлятися та результату. Користувач може самостійно обирати кількість бітів для приховування.

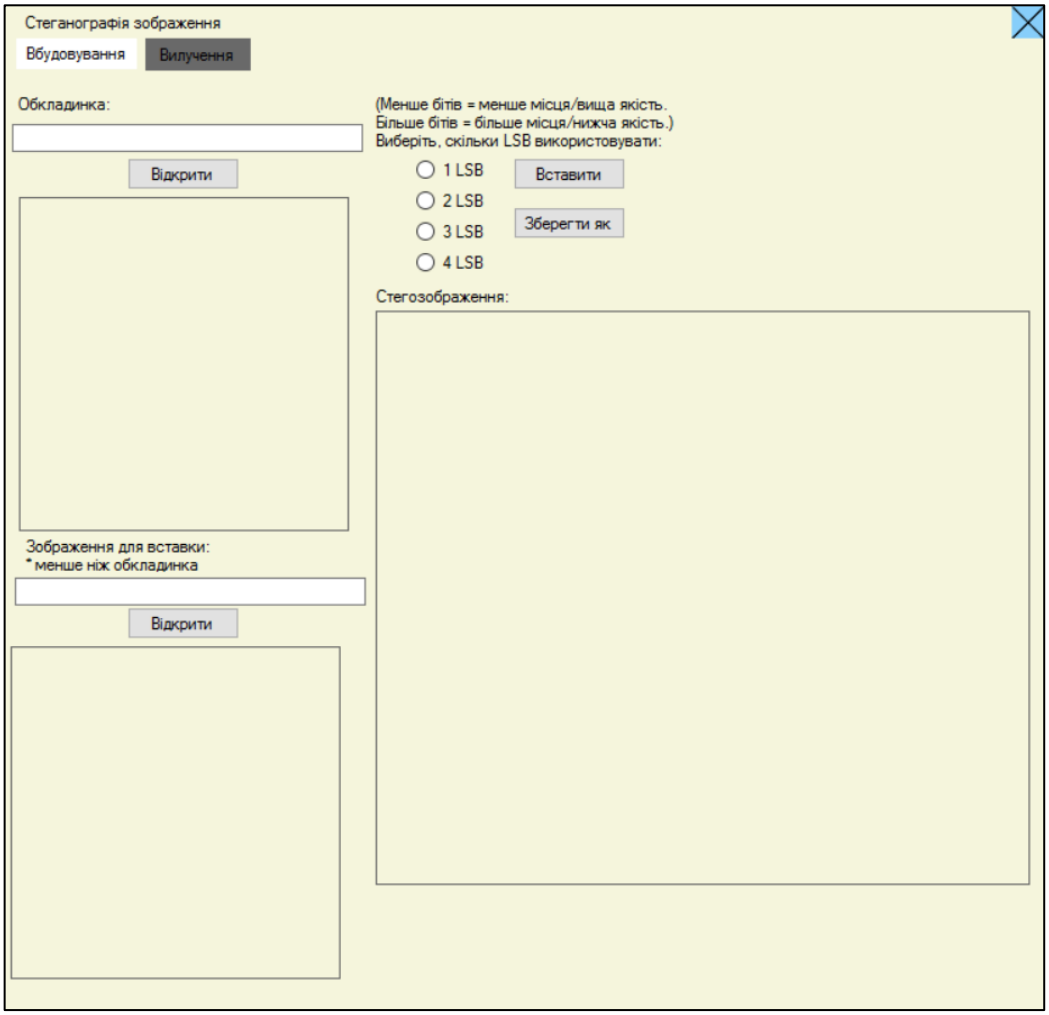



Рисунок 4.10 – Форма для стеганографії зображення

На рис. 4.11 показано вже заповнену форму та результат роботи. Його можна зберегти як окремий файл, таким же способом, як і у попередній формі.

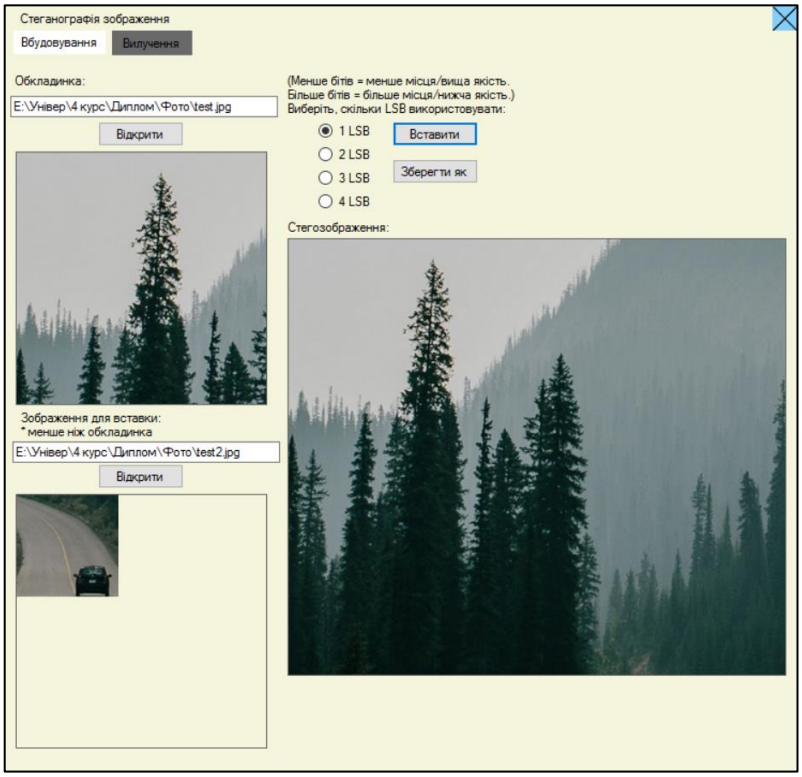



Рисунок 4.11 – Заповнена форма

Також окрім вбудовування є можливість вилучення картинки. Для цього є інша форма, вона зображена на рис. 4.12. Спочатку прописуємо шлях до картинки. Якщо приховування відбулося щойно, це поле заповнюється автоматично шляхом до картинки, яка була тільки що створена.

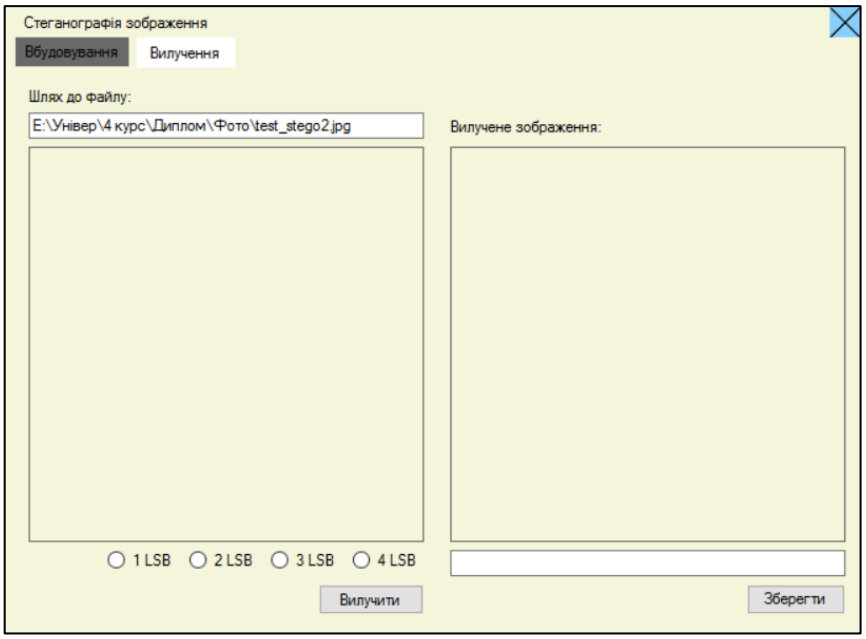



Рисунок 4.12 – Вилучення даних

Як видно з результату показаного на рис. 4.13, вилучення відбулось успішно.

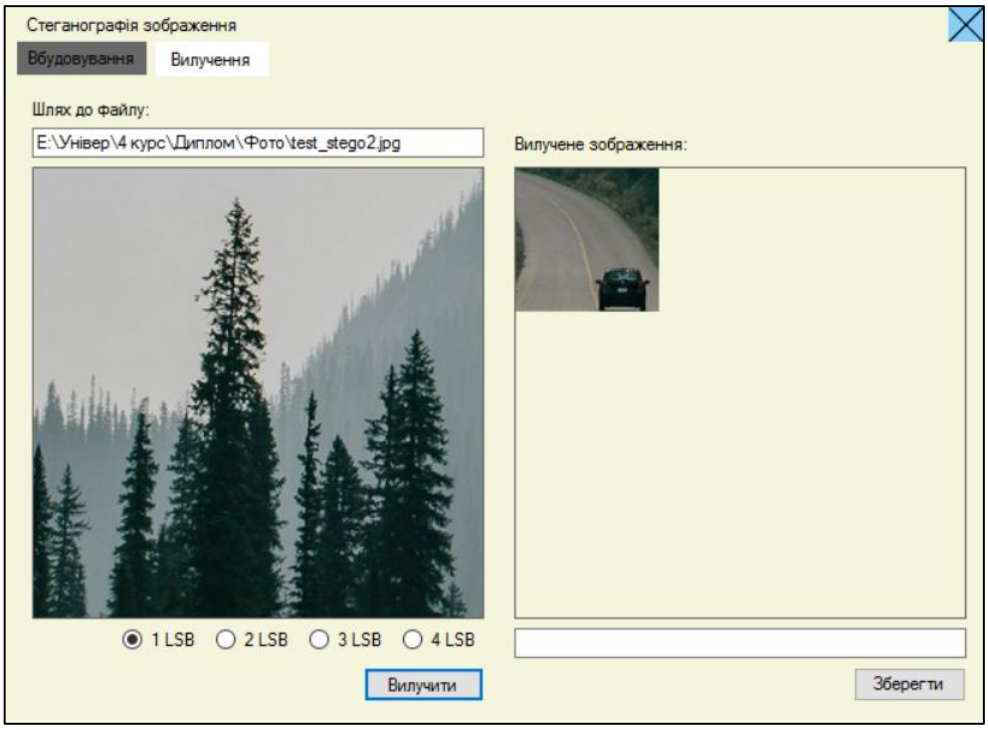



Рисунок 4.13 – Вилучення зображення

### 4.5 Результати роботи

Для оцінки результатів роботи програми будуть використані два методи стеганоаналізу - хі-квадрат атака та RS-атака. Ці методи є статистичними та використовуються для виявлення наявності прихованого повідомлення в областях пікселів. Зокрема, вони добре працюють при звичайному послідовному методі вкраплення на основі найменш значущого біта.

Для проведення експерименту використовується стеганоконтейнер, в якому було вкраплено зашифроване повідомлення : 0uX21KLvMLK2AVdrI7Zpaw==.

Розмір даного сегменту розподілу складає 24 байти, оскільки його довжина – 24 символів, а кодування в системі UTF-8, звідси і отримується розмір у 24 байт. Розмір контейнера складає 1 152 892 байтів. Тобто, повідомлення становить 0,002% від розміру контейнера.

Для виявлення обсягу прихованої інформації у контейнері, який піддається стеганоаналізу, було використано дві атаки - хі-квадрат атаку та RS-атаку. Результати цих атак були представлені на рис. 4.14 та 4.15.

Для здійснення цих атак було використано онлайн-ресурс під назвою "lsbtools", який є npm-модулем для розробки додатків на мові JavaScript, зокрема для розробки додатків на Node.js, React, Angular, Vue та VulcanJs. Для отримання доступу до проведення атак на контейнери з вкрапленим повідомленням, необхідно перейти за посиланням https://desudesutalk.github.io/lsbtools.

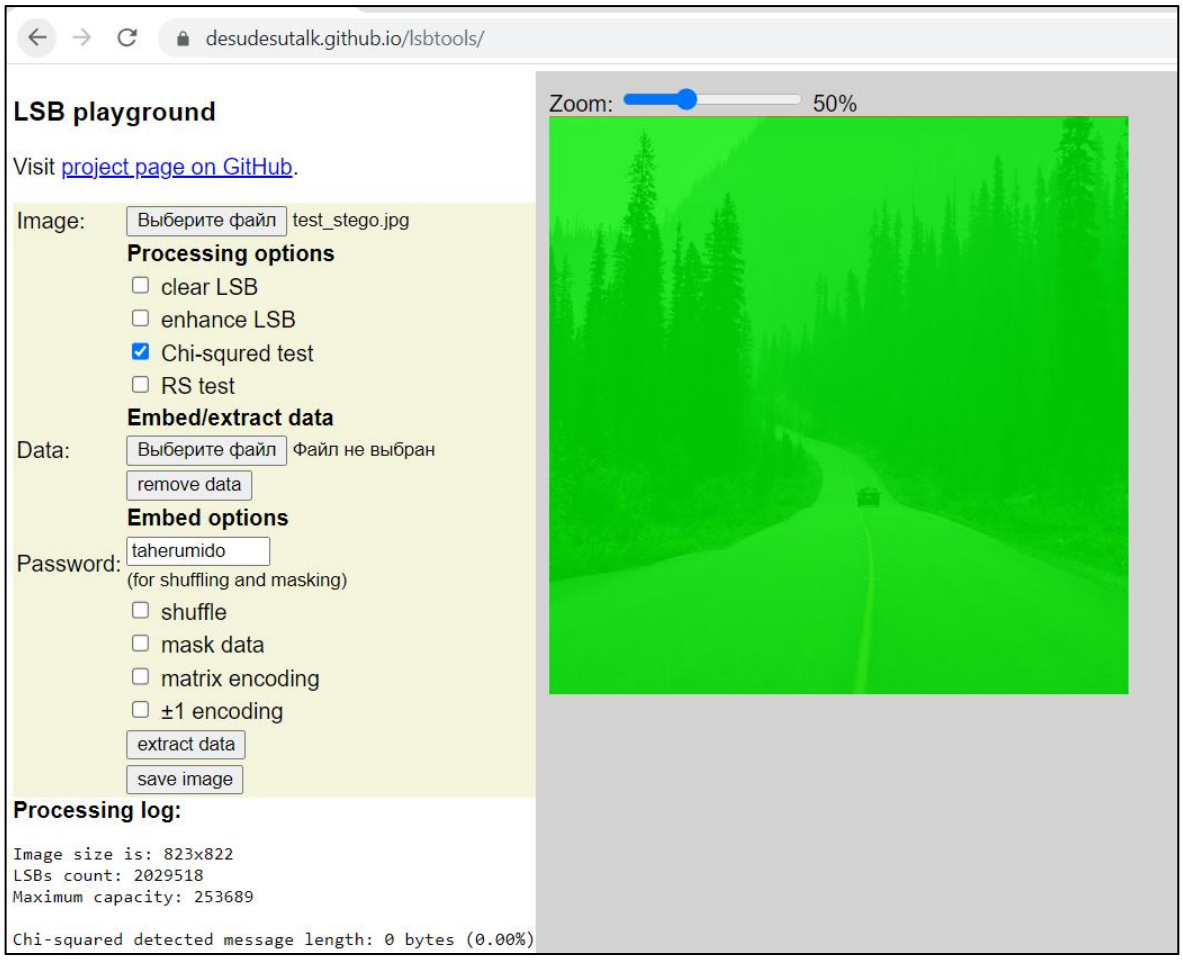



Рисунок 4.14 – Атака хі-квадрат

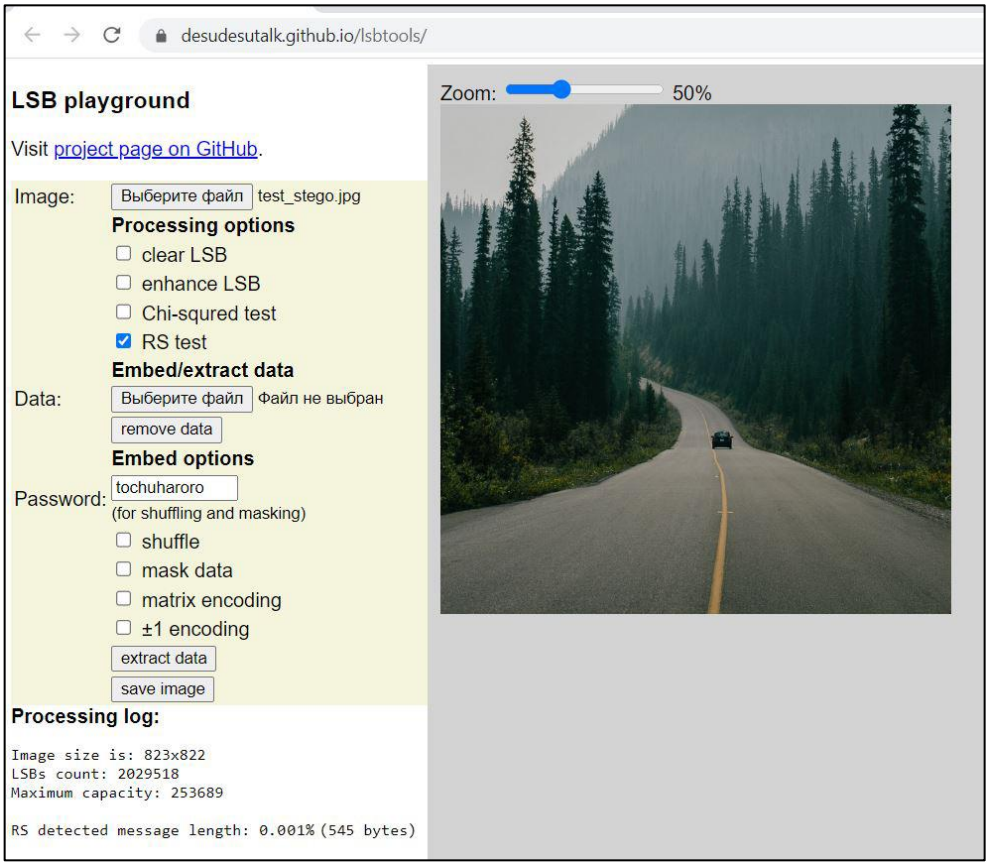



Рисунок 4.15 – RS-атака стеганоконтейнеру

За результатами проведених атак видно, що їхні результати є неточними, що свідчить про високу стійкість використаного методу приховування для даної довжини вкрапленого повідомлення. Атака методом хі-квадрат не змогла виявити приховану інформацію, а RS-атака змогла розпізнати лише 0,001% даних від загального обсягу контейнера, хоча вбудовано було 0,002% від розміру контейнера.

Тепер приховаємо зашифрований текст у зображення, а потім ще в одне зображення і перевіримо його. Розмір повідомлення (зображення) складає 52 384 байти, а розмір контейнера - 1 243 628 байтів. Повідомлення становить 4,2% від розміру контейнера. Результати атак представлено на рис. 4.16 та 4.17.

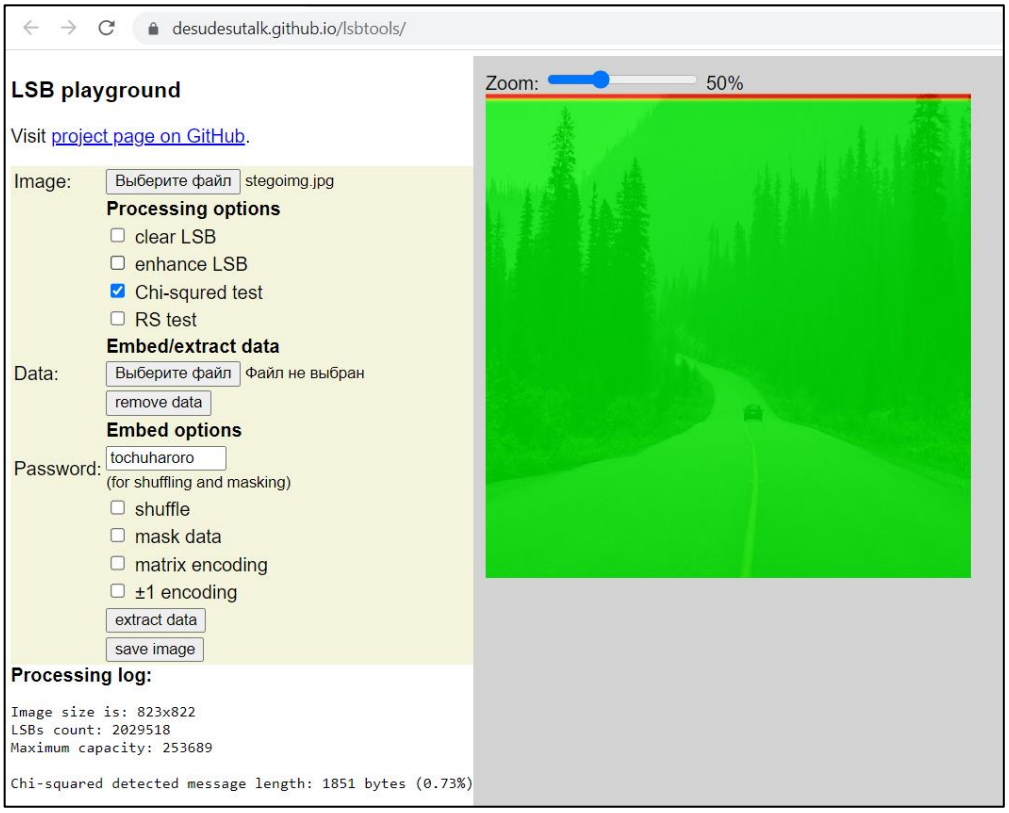



Рисунок 4.16 – Атака хі-квадрат

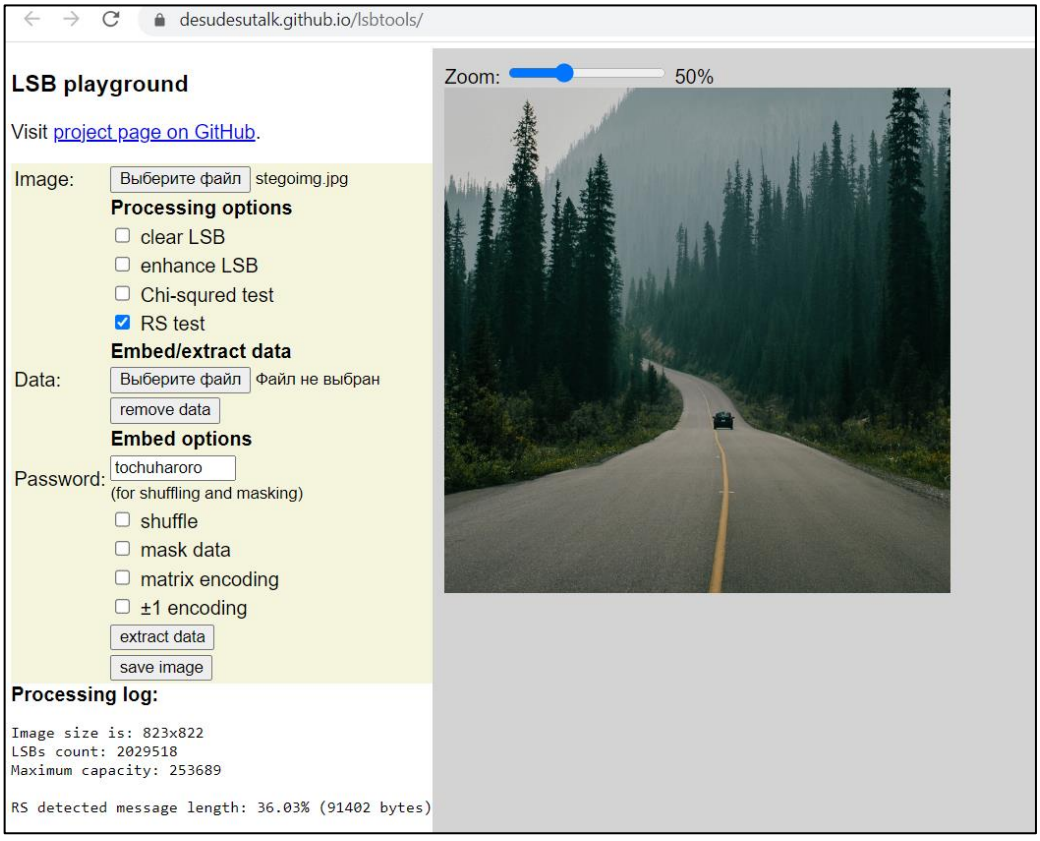



Рисунок 4.17 – RS-атака стеганоконтейнеру

За результатами атак видно, що атака методом хі-квадрат знову не змогла виявити приховану інформацію. Це свідчить про високу стійкість використаного методу приховування, оскільки хі-квадрат атака спрямована на виявлення статистичних відмінностей, але не виявила незвичайних патернів або вкраплених даних.

З іншого боку, RS-атака вже змогла розпізнати 36,03% даних від загального обсягу контейнера. Це означає, що RS-атака успішно виявила певні ознаки вкрапленої інформації, що відрізняються від нормального зображення. Однак, цей відсоток все ще недостатній для повного відновлення повідомлення, яке було вкраплено. Навіть якщо буде розкрито зображення, інформацію з нього буде важко дістати.

**Висновки до розділу 4**

У цьому розділі було розроблено програмний застосунок, використовуючи необхідні засоби розробки та враховуючи вимоги до технічного забезпечення. Була проведена демонстрація роботи програми, показавши її функціональні можливості. Результати тестування підтвердили правильну роботу системи. В цілому, розроблений програмний застосунок є надійним та відповідає поставленим вимогам.

# ВИСНОВКИ

У рамках бакалаврської кваліфікаційної роботи проведено дослідження процесів стеганографії при передачі та зберіганні даних з метою захисту інформації у зображеннях та на основі результатів розроблено інформаційну систему.

В ході дослідження було проведено огляд сучасної комп'ютерної стеганографії, вивчено її особливості та основні поняття. Був проведений аналіз останніх публікацій та існуючих аналогів у даній галузі, а також вивчено існуючі технології, методи та підходи для вирішення поставленої задачі.

Наступним етапом проведено моделювання та проектування інформаційної системи, що використовує методи стеганографії для захисту даних у зображеннях. Описано структуру системи, вхідні дані та використання методу LSB для стеганографічного приховування інформації.

Програмне забезпечення, розроблене в рамках даної роботи, продемонструвало добрі результати при застосуванні методу заміни найменш значущого біта для приховування інформації у зображеннях різного формату. Однак, було виявлено, що використання методу заміни найменш значущого біта є вразливим до стеганоалітичних атак, які можуть виявити наявність і природу прихованої інформації. Для забезпечення вищого рівня захисту прихованої інформації у зображенні, використовувалися криптографічні методи. Перед вкрапленням інформації у зображення, повідомлення шифрувалося з використанням криптографічних алгоритмів, таких як шифр Цезаря та DES. Також використовувалася можливість приховати стегозображення у іншому звичайному зображенні, що додавало додатковий рівень захисту.

В результаті роботи було досягнуто мети захисту інформації у зображеннях з використанням алгоритмів стеганографії. Отримані результати сприятимуть покращенню захисту конфіденційної інформації та забезпеченню безпеки при передачі та зберіганні даних у цифровому середовищі.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Alan Siper Roger Farley and Craig Lombardo. The Rise of Steganography. *Seidenberg School of CSIS Homepage | Pace University New York*. URL: https://csis.pace.edu/~ctappert/srd2005/d1.pdf (date of access: 10.04.2023).

2. The Art of Hiding Information – Johannes Trithemius' Steganography. URL: http://scihi.org/johannes-trithemius-steganography/ (date of access: 11.04.2023).

3. Hiding messages in DNA microdots. https://www.researchgate.net/publication/12921709_Hiding_messages_in_DNA_microdots (date of access: 15.04.2023).

4. Конахович Г.Ф. , Прогонов Д.О. , Пузиренко О.Ю. Комп'ютерна стеганографічна обробка й аналіз мультимедійних даних : навчальна література. Центр навч. літ., 2018. 560 с.

5. G. Mostafa and W. Alexan, "A High capacity Double-Layer Gray Code Based Security Scheme for Secure Data Embedding," *2019 International Symposium on Networks, Computers and Communications (ISNCC)*, Istanbul, Turkey, 2019, pp. 1-6, doi: 10.1109/ISNCC.2019.8909192.

6. M. Jayachandran and J. Manikandan, "SAR Image Compression Using Steganography," *2010 International Conference on Advances in Computer Engineering*, Bangalore, India, 2010, pp. 203-206, doi: 10.1109/ACE.2010.15.

7. Mstafa RJ, Elleithy KM (2017) Compressed and raw video steganography techniques: a comprehensive survey and analysis. Multimed Tools Appl 76(20):21749–21786

8. J. Oravec, Ľ. Ovseník, M. Lapčák, N. Zdravecký and S. Andrejčík, "An Image Encryption Algorithm Suitable for Medical Images," *2022 IEEE 16th International Scientific Conference on Informatics (Informatics)*, Poprad, Slovakia, 2022, pp. 245-250, doi: 10.1109/Informatics57926.2022.10083313

9. Ma, B., Han, Z., Li, J., Wang, C., Wang, Y., Cui, X. (2022). A High-Capacity and High-Security Generative Cover Steganography Algorithm. In: Sun, X., Zhang, X., Xia, Z., Bertino, E. (eds) Advances in Artificial Intelligence and Security. ICAIS 2022.

Communications in Computer and Information Science, vol 1588. Springer, Cham. https://doi.org/10.1007/978-3-031-06764-8_32 (date of access: 01.05.2023).

10. Jiao, G., Liu, J., Zhou, S., Luo, N. (2021). A Image Adaptive Steganography Algorithm Combining Chaotic Encryption and Minimum Distortion Function. In: Liu, Q., Liu, X., Shen, T., Qiu, X. (eds) The 10th International Conference on Computer Engineering and Networks. CENet 2020. Advances in Intelligent Systems and Computing, vol 1274. Springer, Singapore. https://doi.org/10.1007/978-981-15-8462-6_84 (date of access: 06.05.2023).

11. C. Li, X. Zhang, T. Luo and L. Tian, "Audio Steganography Algorithm Based on Genetic Algorithm for MDCT Coefficient Adjustment for AAC," *2020 IEEE International Symposium on Multimedia (ISM)*, Naples, Italy, 2020, pp. 111-112, doi: 10.1109/ISM.2020.00026.

12. Koptyra, K.; Ogiela, M.R. Steganography in IoT: Information Hiding with Joystick and Touch Sensors. *Sensors* 2023, *23*, 3288. https://doi.org/10.3390/s23063288 (date of access: 13.05.2023).

13. O. Torki, M. Ashouri-Talouki and M. Mahdavi, "Blockchain for steganography: advantages, new algorithms and open challenges," *2021 18th International ISC Conference on Information Security and Cryptology (ISCISC)*, Isfahan, Iran, Islamic Republic of, 2021, pp. 1-5, doi: 10.1109/ISCISC53448.2021.9720480.

14. A. Kotkar, S. Khadapkar, A. Gupta and S. Jangale, "Multiple layered Security using combination of Cryptography with Rotational, Flipping Steganography and Message Authentication," *2022 IEEE International Conference on Data Science and Information System (ICDSIS)*, Hassan, India, 2022, pp. 1-5, doi: 10.1109/ICDSIS55133.2022.9915922.

15. Yuk, S.; Cho, Y. A Time-Based Dynamic Operation Model for Webpage Steganography Methods. *Electronics* **2020**, *9*, 2113. https://doi.org/10.3390/electronics9122113 (date of access: 16.05.2023).

16. Least Significant Bit. Science Direct. URL: https://www.sciencedirect.com/topics/computer-science/least-significant-bit (date of access: 16.05.2023).

17. Interval-based Recording of Generated Pseudorandom Numbers - Tomasevic et al 2019a - Scipedia. *Scipedia - Communicating science : Scipedia*. URL: https://www.scipedia.com/public/Tomasevic_et_al_2019a (date of access: 17.05.2023).

18. On the Construction of Pseudo-Random Permutations: Luby-Rackoff Revisited. *ACM Digital Library*. URL: https://dl.acm.org/doi/pdf/10.1145/258533.258581 (date of access: 17.05.2023).

19. M. Fujiyoshi, S. Sato, H. L. Jin and H. Kiya, "A Location-Map Free Reversible Data Hiding Method using Block-Based Single Parameter," *2007 IEEE International Conference on Image Processing*, San Antonio, TX, USA, 2007, pp. III - 257-III - 260, doi: 10.1109/ICIP.2007.4379295.

20. An iterative method of palette-based image steganography. *Research Gate*. URL:https://www.researchgate.net/publication/222570001_An_iterative_method_of_palette-based_image_steganography (date of access: 18.05.2023).

21. Color Image Quantization: A Short Review and an Application with Artificial Bee Colony Algorithm - IOS Press. *Home - IOS Press*. URL: https://content.iospress.com/articles/informatica/inf25-3-08 (date of access: 18.05.2023).

22. L. Nikolay and L. Gleb, "Applying Kutter-Jordan-Bossen seganographic algorithm in video sequences," *2016 IEEE NW Russia Young Researchers in Electrical and Electronic Engineering Conference (EIConRusNW)*, St. Petersburg, Russia, 2016, pp. 271-272, doi: 10.1109/EIConRusNW.2016.7448172.

23. Koch E and Zhao J 1995 Towards robust and hidden image copyright labeling IEEE Workshop on Nonlinear Signal and Image Processing pp 452-455

24. The correct use of the Lax-Friedrichs method. Numdam. URL: http://www.numdam.org/item/M2AN_2004__38_3_519_0/ (date of access: 18.05.2023).

25. K. L. Chiew, L. Jane, F. Sarah, and S. Juan, "Steganography: DCT Coefficients Reparation Technique in JPEG Image," International

Journal of Digital Content Technology and its Applications, vol. 2, no. 2, 2008.

26. Westfeld, A. (2001). F5—A Steganographic Algorithm. In: Moskowitz, I.S. (eds) Information Hiding. IH 2001. Lecture Notes in Computer Science, vol 2137. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-45496-9_21 (date of access: 19.05.2023).

27. Niels Provos. Abstract Defending Against Statistical Steganalysis. URL: https://www.researchgate.net/publication/2538327_Abstract_Defending_Against_Statistical_Steganalysis (date of access: 19.05.2023).

28. Provos N., Honeyman P. Detecting Steganographic Content on the Internet.URL:https://www.researchgate.net/publication/2379632_Detecting_Steganographic_Content_on_the_Internet (date of access: 19.05.2023).

29. Arbell, O., Landau, G.M., Mitchell, J.S.: Edit distance of run-length encoded strings. Inf. Process. Lett. 83(6), 307–314 (2002).

30. Discovery of Huffman Codes | Mathematical Association of America. *Homepage | Mathematical Association of America*. URL: https://www.maa.org/press/periodicals/convergence/discovery-of-huffman-codes.

31. W. Kinsner and R. H. Greenfield, "The Lempel-Ziv-Welch (LZW) data compression algorithm for packet radio," *[Proceedings] WESCANEX '91*, Regina, SK, Canada, 1991, pp. 225-229, doi: 10.1109/WESCAN.1991.160551.

32. Furht, B., Smoliar, S.W., Zhang, H. (1995). JPEG Algorithm for Full-Color Still Image Compression. In: Video and Image Processing in Multimedia Systems. The Springer International Series in Engineering and Computer Science, vol 326. Springer, Boston, MA. URL: https://doi.org/10.1007/978-1-4615-2277-5_5(date of access: 20.05.2023).

33. One-Time Passwords in Everything (OPIE): Experiences with Building and Using Strong Authentication | USENIX. USENIX | The Advanced Computing Systems Association. URL: https://www.usenix.org/conference/5th-usenix-unix-security-

symposium/one-time-passwords-everything-opie-experiences (date of access: 20.05.2023).

34. Jeong J., Chun M. Y., Choo H. Integrated OTP-Based User Authentication Scheme Using Smart Cards in Home Networks. URL: https://www.researchgate.net/publication/221179773_Integrated_OTP-Based_User_Authentication_Scheme_Using_Smart_Cards_in_Home_Networks (date of access: 20.05.2023).

35. Албахарі Б. C# 7.0. Довідник. Повний опис мови / Б. Албахарі, А. Албахари., 2018.

# ДОДАТОК А

## Файл шифру Цезаря

```
class Caesar
    {
        char[] alpha = " ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz!£$%^&*()_+-
=<>,./?@';:#~[{]}`¬|".ToCharArray();

        public string encrypt(string input_)
        {
            char[] inArr = input_.ToCharArray();
            string encrypted = "";
            //for each letter in inut string
            for (int i = 0; i < inArr.Length; i++)
            {
                //compare against alhabet
                for (int j = 0; j < alpha.Length; j++)
                {
                    string c1 = inArr[i].ToString();
                    string c2 = alpha[j].ToString();
                    int c = string.Compare(c1, c2);
                    if (c == 0)
                    {
                        //check if out of bounds
                        if (j + 2 >= alpha.Length)
                        {
                            int overlap = (j + 2) - alpha.Length;
                            encrypted += alpha[overlap].ToString();
                        }
                        else
                        {
                            encrypted += alpha[j + 2].ToString();
                        }
                    }
                }
            }
            return encrypted;
        }
        public string decrypt(string input_)
        {
            char[] inArr = input_.ToCharArray();
            string decrypted = "";
            //for each letter in inut string
            for (int i = 0; i < inArr.Length; i++)
            {
                //compare each letter against alhabet
                for (int j = 0; j < alpha.Length; j++)
                {
                    string c1 = inArr[i].ToString();
                    string c2 = alpha[j].ToString();
                    int c = string.Compare(c1, c2);
                    int asc = (int)alpha[j];
                    if (c == 0)
                    {
                        //check if out of bounds
                        if (j - 2 < 0)
                        {
                            int overlap = (j - 2) + alpha.Length;
                            decrypted += alpha[overlap].ToString();
                        }
                        else
                        {
                            decrypted += alpha[j - 2].ToString();
                        } }   }return decrypted;} }
```

# ДОДАТОК Б

## Файл шифру DES

```
public class Cryptog
    {
        private byte[] key = new byte[8] { 3, 1, 4, 5, 6, 2, 4, 9 };
        private byte[] iv = new byte[8] { 1, 2, 3, 4, 5, 6, 7, 8 };

        public string Crypt(string text, string key_)
        {
            iv = Encoding.ASCII.GetBytes(key_);
            SymmetricAlgorithm algorithm = DES.Create();
            ICryptoTransform transform = algorithm.CreateEncryptor(key, iv);
            byte[] inputbuffer = Encoding.Unicode.GetBytes(text);
            byte[] outputBuffer = transform.TransformFinalBlock(inputbuffer, 0,
inputbuffer.Length);
            return Convert.ToBase64String(outputBuffer);
        }

        public string Decrypt(string text, string iv_)
        {
            iv = Encoding.ASCII.GetBytes(iv_);
            SymmetricAlgorithm algorithm = DES.Create();
            ICryptoTransform transform = algorithm.CreateDecryptor(key, iv);
            byte[] inputbuffer = Convert.FromBase64String(text);
            byte[] outputBuffer = transform.TransformFinalBlock(inputbuffer, 0,
inputbuffer.Length);
            return Encoding.Unicode.GetString(outputBuffer);
        }
                                                }
```

# ДОДАТОК В

## Метод 1 LSB для вбудовування тексту

```
public Bitmap Embed1lsb(string inRoute_, string inText_)
{
    bmp = new Bitmap(inRoute_);
    string bitString = "";
    Point pixelIndex = new Point(1, 0);
    //setLength MUST be after new bmp or it will be wiped.
    SetLength(inText_.Length);

    //EMBED LETTER
    inText_.Insert(0, "0");

    //for each letter in the message.
    for (int i = 0; i < inText_.Length; i++)
    {
        //get the binary value of current letter in message
        string newBit = ops.convLetterToBits(inText_.Substring(i, 1));
        //append bitstream with new byte
        bitString = bitString + newBit;
    }
    //calculate how many pixels are needed for length of message
    int noOfPixels = (inText_.Length * 8) / 3;
    pBarSetup(noOfPixels);
    //for each pixel in image.
    for (int i = 0; i < noOfPixels; i++)
    {
        //increase progress bar
        pBar1.PerformStep();
        //retrieve pixel at index
        Color pixelCol = bmp.GetPixel(pixelIndex.X, pixelIndex.Y);
        int finalR = 0, finalG = 0, finalB = 0;
        //loop through R/G/B of pixel
        for (int rgb = 0; rgb < 3; rgb++)
        {
            if (!String.IsNullOrEmpty(bitString))
            {
                switch (rgb)
                {
                    //R
                    case 0:
                        {
                            //get the binary values of R
                            string rBitString = ops.convNumberToBits(pixelCol.R);
                            //get first 7 digit of R
                            string rFirstFour = rBitString.Substring(0, 7);
                            //get first digit from bitStream
                            string lFirstFour = bitString.Substring(0, 1);
                            //remove first digit bitstring
                            bitString = bitString.Substring(1, bitString.Length - 1);
                            //merge and convert to back to int
                            int newR = Convert.ToInt32(rFirstFour + lFirstFour);

                            finalR = ops.binaryToDecimal(newR);
                            break;
                        }
```

```
                        //G
                        case 1:
                            {
                                //get the binary values of G
                                string gBitString = ops.convNumberToBits(pixelCol.G);
                                //get first 7 digit of G
                                string gFirstFour = gBitString.Substring(0, 7);
                                //get first digit from bitStream
                                string lLastFour = bitString.Substring(0, 1);
                                //remove first digit bitstring
                                bitString = bitString.Substring(1, bitString.Length - 1);
                                //merge
                                int newG = Convert.ToInt32(gFirstFour + lLastFour);

                                finalG = ops.binaryToDecimal(newG);
                                break;
                            }
                        //B
                        case 2:
                            {
                                //get the binary values of B
                                string bBitString = ops.convNumberToBits(pixelCol.B);
                                //get first 7 digit of B
                                string bFirstFour = bBitString.Substring(0, 7);
                                //get first digit from bitStream
                                string lLastFour = bitString.Substring(0, 1);
                                //remove first digit bitstring
                                bitString = bitString.Substring(1, bitString.Length - 1);
                                //merge
                                int newB = Convert.ToInt32(bFirstFour + lLastFour);

                                finalB = ops.binaryToDecimal(newB);
                                break;
                            }
                    }
                }
            }
            bmp.SetPixel(pixelIndex.X,  pixelIndex.Y,  Color.FromArgb(finalR,  finalG,
finalB));

            pixelIndex.X++;
            if (pixelIndex.X >= bmp.Width)
            {
                pixelIndex.X = 1;
                pixelIndex.Y++;
            }
        }
        pBar1.Visible = false;
        return bmp;
    }
```

# ДОДАТОК Г

## Метод 1 LSB для вбудовування зображення

```
public Bitmap Embed1lsb(string coverRoute_, string stegoRoute_)
      {
          bmpCover = new Bitmap(coverRoute_);
          bmpStego = new Bitmap(stegoRoute_);
          string bitString = "";
          Point pixelIndex = new Point(1, 0);
          //width * height * 8 * 3
          SetLength((bmpStego.Width * bmpStego.Height) * 24);
          //set width and height in second to last pixel
          SetWH(bmpStego.Width, bmpStego.Height);
          //convert image to bitstring.
          for (int y = 0; y < bmpStego.Height; y++)
          {
              for (int x = 0; x < bmpStego.Width; x++)
              {
                  Color pixelCol = bmpStego.GetPixel(x, y);

                  for (int rgb = 0; rgb < 3; rgb++)
                  {
                      switch (rgb)
                      {
                          //R
                          case 0:
                              {
                                  //get the binary value of current letter in message
                                  string newBit = ops.convNumberToBits(pixelCol.R);
                                  //append bitstream with new byte
                                  bitString = bitString + newBit;
                                  break;
                              }
                          //G
                          case 1:
                              {
                                  //get the binary value of current letter in message
                                  string newBit = ops.convNumberToBits(pixelCol.G);
                                  //append bitstream with new byte
                                  bitString = bitString + newBit;
                                  break;
                              }
                          //B
                          case 2:
                              {
                                  //get the binary value of current letter in message
                                  string newBit = ops.convNumberToBits(pixelCol.B);
                                  //append bitstream with new byte
                                  bitString = bitString + newBit;
                                  break;
                              }
                      }
                  }

              }
          }
          int noOfPixels = (int)bitString.Length / 3;
          for (int i = 0; i < noOfPixels; i++)
          {
              //retrieve pixel at index
              Color pixelCol = bmpCover.GetPixel(pixelIndex.Y, pixelIndex.X);

              int finalR = 0, finalG = 0, finalB = 0;
              //loop through R/G/B of pixel
```

```
                for (int rgb = 0; rgb < 3; rgb++)
                {
                    if (!String.IsNullOrEmpty(bitString))
                    {
                        switch (rgb)
                        {
                            //R
                            case 0:
                                {
                                    //get the binary values of R
                                    string rBitString = ops.convNumberToBits(pixelCol.R);
                                    //get first 7 digit of R
                                    string rFirstFour = rBitString.Substring(0, 7);
                                    //get first digit from bitStream
                                    string lFirstFour = bitString.Substring(0, 1);
                                    //remove first digit bitstring
                                    bitString = bitString.Substring(1, bitString.Length - 1);
                                    //merge and convert to back to int
                                    int newR = Convert.ToInt32(rFirstFour + lFirstFour);
                                    finalR = ops.binaryToDecimal(newR);
                                    break;
                                }
                            //G
                            case 1:
                                {
                                    //get the binary values of G
                                    string gBitString = ops.convNumberToBits(pixelCol.G);
                                    //get first 7 digit of G
                                    string gFirstFour = gBitString.Substring(0, 7);
                                    //get first digit from bitStream
                                    string lLastFour = bitString.Substring(0, 1);
                                    //remove first digit bitstring
                                    bitString = bitString.Substring(1, bitString.Length - 1);
                                    //merge
                                    int newG = Convert.ToInt32(gFirstFour + lLastFour);
                                    finalG = ops.binaryToDecimal(newG);
                                    break;
                                }
                            //B
                            case 2:
                                {
                                    //get the binary values of B
                                    string bBitString = ops.convNumberToBits(pixelCol.B);
                                    //get first 7 digit of B
                                    string bFirstFour = bBitString.Substring(0, 7);
                                    //get first digit from bitStream
                                    string lLastFour = bitString.Substring(0, 1);
                                    //remove first digit bitstring
                                    bitString = bitString.Substring(1, bitString.Length - 1);
                                    //merge
                                    int newB = Convert.ToInt32(bFirstFour + lLastFour);
                                    finalB = ops.binaryToDecimal(newB);
                                    break;
                                }
                        }
                    }
                }
                bmpCover.SetPixel(pixelIndex.Y, pixelIndex.X, Color.FromArgb(finalR, finalG,
finalB));

                pixelIndex.Y++;
                if (pixelIndex.Y > bmpCover.Height)
                {
                    pixelIndex.Y = 0;
                    pixelIndex.X++;
                }
            }  return bmpCover;
```

# ДОДАТОК Д

## Метод 1 LSB для вилучення тексту

```
  public string Retrieve1lsb(string inRoute_)
        {
            Point pixelIndex = new Point(1, 0);
            Bitmap bmp = new Bitmap(inRoute_);

            //get length from final pixel
            Color lastPixel = bmp.GetPixel(bmp.Width - 1, bmp.Height - 1);
            string first = ops.convNumberToBits(lastPixel.R);
            string second = ops.convNumberToBits(lastPixel.G);
            string third = ops.convNumberToBits(lastPixel.B);
            string concat = first + second + third;
            long finalLength = Convert.ToInt64(concat);

            //length in decimal
            long msgLength = ops.binaryToDecimalLong(finalLength);

            //DECODING
            string message = "";
            int counter = 0;
            //progress bar
            pBarSetup(msgLength * 8);

            while (counter < msgLength * 8)
            {
                //increase progress bar
                pBar1.PerformStep();
                Color pixelCol = bmp.GetPixel(pixelIndex.X, pixelIndex.Y);
                //loop through R/G/B
                for (int rgb = 0; rgb < 3; rgb++)
                {
                    if (counter < msgLength * 8)
                    {
                        switch (rgb)
                        {
                            //R
                            case 0:
                                {
                                    //get the binary values of R (notice the "2" param)
                                    string rBitString = ops.convNumberToBits(pixelCol.R);
                                    //get last 4 digits of letter (which is the first 4 digits
of our letter)
                                    string rLastFour = rBitString.Substring(7, 1);

                                    message += rLastFour;
                                    counter++;
                                    break;
                                }
                            //G
                            case 1:
                                {
                                    //get the binary values of G
                                    string gBitString = ops.convNumberToBits(pixelCol.G);
                                    //get first 4 digit of G
                                    string gLastFour = gBitString.Substring(7, 1);
                                    message += gLastFour;
                                    counter++;
                                    break;
                                }
```

```
                    //B
                    case 2:
                        {
                            //get the binary values of B
                            string bBitString = ops.convNumberToBits(pixelCol.B);
                            //get first 4 digit of B
                            string bLastFour = bBitString.Substring(7, 1);
                            message += bLastFour;
                            counter++;
                            break;
                        }

                }
            }

        }
        pixelIndex.X++;
        if (pixelIndex.X >= bmp.Width)
        {
            pixelIndex.X = 1;
            pixelIndex.Y++;
        }
    }

    pBar1.Visible = false;
    //update textbox with decoded
    return ops.BinaryToString(message);
    //update image info panel
    // updateImgInfo();
}

private void pBarSetup(long inLength_)
{
    //Convert long to int for maximum
    int length = Convert.ToInt32(inLength_);
    // Display the ProgressBar control.
    pBar1.Visible = true;
    // Set Minimum to 1 to represent the first file being copied.
    pBar1.Minimum = 1;
    // Set Maximum to the total number of files to copy.
    pBar1.Maximum = length;
    // Set the initial value of the ProgressBar.
    pBar1.Value = 1;
    // Set the Step property to a value of 1 to represent each file being copied.
    pBar1.Step = 1;
                                            }}
```

# ДОДАТОК Е

## Метод 1 LSB для вилучення зображення

```
public Bitmap Retrieve1lsb(string inRoute_)
     {
         Point pixelIndex = new Point(1, 0);
         Bitmap bmp = new Bitmap(inRoute_);
         //get length from final pixel
         Color lastPixel = bmp.GetPixel(bmp.Width - 1, bmp.Height - 1);
         string first = ops.convNumberToBits(lastPixel.R);
         string second = ops.convNumberToBits(lastPixel.G);
         string third = ops.convNumberToBits(lastPixel.B);
         string concat = first + second + third;
         long finalLength = Convert.ToInt64(concat);
         Color SLastPixel = bmp.GetPixel(bmp.Width - 2, bmp.Height - 1);
         int width = SLastPixel.G;
         int height = SLastPixel.B;
         //length in decimal
         long msgLength = ops.binaryToDecimalLong(finalLength);
         //DECODING
         string message = "";
         int counter = 0;
         while (counter < msgLength)
         {
             Color pixelCol = bmp.GetPixel(pixelIndex.Y, pixelIndex.X);
             //loop through R/G/B
             for (int rgb = 0; rgb < 3; rgb++)
             {
                 if (counter < msgLength * 8)
                 {
                     switch (rgb)
                     {
                         //R
                         case 0:
                             {
                                 //get the binary values of R (notice the "2" param)
                                 string rBitString = ops.convNumberToBits(pixelCol.R);
                     //get last 4 digits of letter (which is the first 4 digits of our letter)
                                 string rLastFour = rBitString.Substring(7, 1);
                                 message += rLastFour;
                                 counter++;
                                 break;
                             }
                         //G
                         case 1:
                             {
                                 //get the binary values of G
                                 string gBitString = ops.convNumberToBits(pixelCol.G);
                                 //get first 4 digit of G
                                 string gLastFour = gBitString.Substring(7, 1);
                                 message += gLastFour;
                                 counter++;
                                 break;
                             }
                         //B
                         case 2:
                             {
                                 //get the binary values of B
                                 string bBitString = ops.convNumberToBits(pixelCol.B);
                                 //get first 4 digit of B
                                 string bLastFour = bBitString.Substring(7, 1);
                                 message += bLastFour;
                                 counter++;
                                 break;
```

```
                    }

                }
            }

        }
        pixelIndex.Y++;
        if (pixelIndex.Y > bmp.Height)
        {
            pixelIndex.Y = 0;
            pixelIndex.X++;
        }
    }

    //convert bitstring(message) back to image
    finalIMG = new Bitmap(width, height);
    for (int y = 0; y < finalIMG.Height; y++)
    {
        for (int x = 0; x < finalIMG.Width; x++)
        {

            int r = 0;
            int g = 0;
            int b = 0;
            //rgb
            for (int i = 0; i < 3; i++)
            {
                switch (i)
                {
                    //R
                    case 0:
                        {
                            int newR = Convert.ToInt32(message.Substring(0, 8));
                            //remove first digit message(bitstring)
                            message = message.Substring(8, message.Length - 8);
                            r = ops.binaryToDecimal(newR);
                            break;
                        }
                    //G
                    case 1:
                        {
                            int newG = Convert.ToInt32(message.Substring(0, 8));
                            //remove first digit message(bitstring)
                            message = message.Substring(8, message.Length - 8);
                            g = ops.binaryToDecimal(newG);
                            break;
                        }
                    //B
                    case 2:
                        {
                            int newB = Convert.ToInt32(message.Substring(0, 8));
                            //remove first digit message(bitstring)
                            message = message.Substring(8, message.Length - 8);
                            b = ops.binaryToDecimal(newB);
                            break;
                        }

                }

            }

            Color colIn = Color.FromArgb(r, g, b);
            finalIMG.SetPixel(x, y, colIn);
        }
    }

    return finalIMG; }
```

# ДОДАТОК Є

## Блок – схема алгоритму роботи системи