Вінницький національний технічний університет

Факультет менеджменту та інформаційної безпеки

Кафедра менеджменту та безпеки інформаційних систем

БАКАЛАВРСЬКА ДИПЛОМНА РОБОТА

на тему:

«Розробка програми захисту томографічних знімків пацієнта стенографічним методом»

Виконав: студент 4 курсу, групи 1КІТС–20б
спеціальності 125– Кібербезпека
Освітня програма – Кібербезпека інформаційних технологій та систем

_____ Перебейнос М.В.__

Керівник: к.т.н., доцент кафедри МБІС
_____ Адлер О. О._____
«_6_» _червня_____ 2024 р.

Рецензент: к.т.н., доц., доцент каф. ОТ
_____ Колесник І. С.____
«_6_» _червня_____ 2024 р.

Допущено до захисту
Голова секції УБ, кафедра МБІС
д.т.н., проф. Яремчук Ю. Є.
“_6_” _червня_____2024

Вінниця ВНТУ – 2024 рік

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

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

ЗАТВЕРДЖУЮ
Голова секції УБ, кафедра МБІС

д.т.н., проф. Яремчук Ю.Є.
22 ” березня 2024 р.

З А В Д А Н Н Я
на бакалаврську дипломну роботу студенту
Перебейносу Максиму Валерійовичу
(прізвище, ім'я, по-батькові)

1. Тема роботи Розробка програми захисту томографічних знімків пацієнта стенографічним методом
Керівник роботи Адлер Оксана Олександрівна к.т.н., доцент кафедри МБІС
(прізвище, ім'я, по-батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від “11” березня 2024року № 80
2. Термін подання студентом роботи 6 червня 2023р
3. Вихідні дані до роботи Електронні джерела, наукові статті, підручники та офіційні документи по темі роботи. Існуюче програмне забезпечення, яке відноситься до теми бакалаврської дипломної роботи.
4. Зміст текстової частини:
Для досягнення успішності виконання роботи потрібно було: дослідити предметну область обраної теми; здійснити аналіз технології, на якій базується розробка програмного забезпечення; здійснити вибір методів програмування; здійснити розробку програмного забезпечення; здійснити розробку програмного додатку.
5. Перелік ілюстративного матеріалу (з точним зазначенням обов`язкових креслень)

Схема взаємодії користувача з ресурсом; схема роботи алгоритму програми; алгоритм роботи технології блокчейн; схема бази даних; UML-діаграма класів.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Розділ I	Адлер.О. О. к.т.н., доцент кафедри МБІС		
Розділ II	Адлер.О. О. к.т.н., доцент кафедри МБІС		
Розділ III	Адлер.О. О. к.т.н., доцент кафедри МБІС		

7.Дата видачі завдання 22 березня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Аналіз предметної області	23.03.2024	02.04.2024	
2	Розробка захищеного вебресурсу	03.04.2024	12.04.2024	
3	Проектування веб-ресурсу	13.04.2024	25.04.2024	
4	Написання БДР на основі визначеної теми	26.04.2024	10.05.2024	
5	Тестування розробленого веб-ресурсу	11.05.2024	23.05.2024	
6	Попередній захист БДР	24.05.2024	02.06.2024	
7	Виправлення роботи	03.06.2024	11.06.2024	
8	Захист БДР	12.06.2024	17.06.2024	

Студент _____ _____
(підпис) (прізвище та ініціали)

Керівник роботи _____ Адлер О. О.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Дипломна робота присвячена розробці та реалізації програмного забезпечення для захисту томографічних знімків пацієнтів за допомогою стеганографічних методів. Робота містить 94 сторінки, 10 рисунків, 4 таблиці, 3 додатка та список використаних джерел з 30 найменувань. Об'єктом дослідження є процес захисту медичних томографічних зображень. Предметом дослідження є методи та алгоритми стеганографії, що застосовуються для захисту медичних томографічних зображень. Метою роботи є розробка програми, що забезпечує захист томографічних знімків пацієнта шляхом вбудовування конфіденційних даних безпосередньо в зображення з використанням стеганографічних методів, забезпечуючи при цьому збереження діагностичної цінності знімків та відповідність медичним стандартам. У роботі проведено огляд існуючих методів стеганографії та їх застосування в медицині, проаналізовано особливості медичних томографічних зображень та вимоги до їх захисту. Розроблено алгоритм вбудовування та вилучення даних у томографічні знімки з використанням методу LSB, проведено тестування розробленого алгоритму та програмного забезпечення. Результати роботи можуть бути використані в медичних установах для забезпечення конфіденційності та цілісності томографічних знімків пацієнтів. КЛЮЧОВІ СЛОВА: СТЕГАНОГРФІЯ, МЕДИЧНІ ЗОБРАЖЕННЯ, ТОМОГРАФІЧНІ ЗНІМКИ, ЗАХИСТ ІНФОРМАЦІЇ, КОНФІДЕНЦІЙНІСТЬ, ЦІЛІСНІСТЬ, LSB, ЦИФРОВИЙ ВОДЯНИЙ ЗНАК,ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ.

ABSTRACT

The thesis is devoted to the development and implementation of software for the protection of tomographic images of patients using steganographic methods. The work contains 94 pages, 10 figures, 4 tables, 3 appendix and a list of references with 30 titles. The object of research is the process of protecting medical tomographic images. The subject of research is the methods and algorithms of steganography used to protect medical tomographic images. The aim of the work is to develop a program that protects patient tomographic images by embedding confidential data directly into the image using steganographic methods, while preserving the diagnostic value of the images and complying with medical standards. The paper reviews existing steganography methods and their application in medicine, analyzes the features of medical tomographic images and the requirements for their protection. An algorithm for embedding and extracting data in tomographic images using the LSB method has been developed, and the developed algorithm and software have been tested. The results of the work can be used in medical institutions to ensure the confidentiality and integrity of tomographic images of patients. KEYWORDS: STEGANOGRAPHY, MEDICAL IMAGES, TOMOGRAPHIC IMAGES, INFORMATION SECURITY, CONFIDENTIALITY, INTEGRITY, LSB, DIGITAL WATERMARK, SOFTWARE.

ЗМІСТ

ВСТУП

Актуальність теми. З розвитком цифрових технологій в медицині, томографічні знімки стали невід'ємною частиною діагностики та лікування.Однак, зберігання та передача цих знімків пов'язані з ризиками несанкціонованого доступу, підробки та витоку конфіденційної інформації про пацієнтів. Законодавчі акти, такі як HIPAA в США та GDPR в Європі, встановлюють суворі вимоги щодо захисту медичних даних, включаючи томографічні знімки.

Стеганографія, як наука про приховування даних, пропонує ефективні методи захисту медичних зображень. Вбудовування додаткової інформації, такої як дані пацієнта, результати аналізів або цифровий водяний знак, безпосередньо в зображення дозволяє забезпечити їх конфіденційність, цілісність та автентичність. Це особливо актуально для томографічних знімків, які містять детальну інформацію про стан здоров'я пацієнта та є цінним об'єктом для кібератак.

Мета і завдання дослідження. Метою дипломної роботи є розробка програми, що забезпечує захист томографічних знімків пацієнта шляхом вбудовування конфіденційних даних безпосередньо в зображення з використанням стеганографічних методів, забезпечуючи при цьому збереження діагностичної цінності знімків та відповідність медичним стандартам.

Для досягнення поставленої мети необхідно вирішити такі завдання:

1.Провести детальний аналіз основних понять, та методів стеганографії, включаючи їх переваги та недоліки, з акцентом на застосування в медичній сфері.

2. Дослідити особливості медичних томографічних зображень та вимоги до їх захисту.

3.Розробити модельта алгоритм вбудовування даних у томографічні знімки з урахуванням медичних стандартів та вимог до якості зображень.

4. Реалізувати програмне забезпечення, що дозволяє вбудовувати та вилучати дані з томографічних знімків, забезпечуючи при цьому їх цілісність та конфіденційність.

5.Провести тестування розробленого програмного забезпечення на реальних томографічних знімках та оцінити його ефективність.

Об'єкт дослідження – процес захисту медичних томографічних зображень.

Предмет дослідження – методи та алгоритми стеганографії, що застосовуються для захисту медичних томографічних зображень.

Методи дослідження. У роботі використані методи аналізу та синтезу для дослідження існуючих стеганографічних методів, метод моделювання для розробки моделі системи захисту, методи програмування для реалізації програмного забезпечення, а також методи експериментального дослідження для оцінки ефективності розробленого методу.

РОЗДІЛ 1. ОГЛЯД ОСНОВ СТЕНОГРАФІЇ, ЇЇ ЗАСТОСУВАННЯ У МЕДЕЦИНІ ТА ПОСТАНОВКА ЗАДАЧ

1.1 Основні поняття, історія та методи стенографії

Стеганографія (від грец. στεγανός — прихований та γράφειν — писати) — це наука та мистецтво прихованого передавання інформації шляхом вбудовування її в інші, на перший погляд, нешкідливі дані. На відміну від криптографії, яка робить інформацію нечитабельною, стеганографія приховує сам факт існування секретного повідомлення.

Історичний екскурс

Історично стеганографія використовувалася ще в античні часи. Геродот описує випадок, коли повідомлення було виголене на голові раба, а потім приховане відрослим волоссям. У Стародавньому Римі використовувалися різні види невидимого чорнила, наприклад, молоко або сік лимона, які проявлялися при нагріванні.

У середньовіччі стеганографія продовжувала розвиватися. Одним із відомих прикладів є використання решіток Кардано, які дозволяли приховувати повідомлення в звичайному тексті. Також використовувалися мікроточки, які наносилися на папір і були видимі лише під мікроскопом.

З розвитком цифрових технологій стеганографія знайшла нові можливості. Цифрові зображення, аудіо- та відеофайли стали ідеальними контейнерами для прихованих повідомлень. Це відкрило нові горизонти для застосування стеганографії в різних сферах, включаючи захист авторських прав, забезпечення конфіденційності та навіть шпигунство.

Одним з перших прикладів цифрової стеганографії є техніка Least Significant Bit (LSB), яка була запропонована ще в 1980-х роках. Цей метод полягає у заміні найменш значущих бітів зображення або аудіофайлу

бітами секретного повідомлення. LSB стеганографія є простою у реалізації, але має обмеження щодо обсягу даних, які можна приховати.

У 1990-х роках з'явилися більш складні методи стеганографії, такі як вбудовування в області перетворення (DCT, DWT) та адаптивне вбудовування. Ці методи дозволяють приховувати більший обсяг даних та забезпечують вищу стійкість до виявлення.

Сьогодні стеганографія продовжує активно розвиватися, з'являються нові методи та алгоритми, які враховують особливості різних типів контейнерів та вимоги до стійкості стеганограм. Особливу увагу приділяють розробці стеганографічних методів, стійких до сучасних методів стегоаналізу, що використовують машинне навчання та штучний інтелект.

Основні терміни та поняття

- Контейнер: Цифровий об'єкт (зображення, аудіо, відео, текст), в який вбудовується приховане повідомлення. Вибір контейнера залежить від типу прихованих даних та вимог до стеганографічної системи.

- Стеганограма: Контейнер з вбудованим прихованим повідомленням. Стеганограма повинна бути максимально схожою на оригінальний контейнер, щоб ускладнити виявлення прихованого повідомлення.

- Стеганографічний ключ: Секретна інформація (пароль, фраза, файл), необхідна для вбудовування та вилучення прихованого повідомлення. Ключ забезпечує додатковий рівень безпеки, оскільки без нього зловмисник не зможе витягти приховане повідомлення.

- Стегоаналіз: Наука про виявлення прихованих повідомлень у цифрових об'єктах. Стегоаналіз використовує різні методи, такі як

статистичний аналіз, візуальний аналіз та машинне навчання, для виявлення ознак стеганографії.

Сучасні тенденції та перспективи розвитку

Сучасна стеганографія активно розвивається, з'являються нові методи та алгоритми, які враховують особливості різних типів контейнерів та вимоги до стійкості стеганограм. Особливу увагу приділяють розробці стеганографічних методів, стійких до сучасних методів стегоаналізу, що використовують машинне навчання та штучний інтелект.

Перспективними напрямками розвитку стеганографії є:

- Використання нейронних мереж: Нейронні мережі можуть бути використані для розробки більш ефективних та стійких алгоритмів вбудовування та вилучення даних. Наприклад, генеративно-змагальні мережі (GAN) можуть генерувати стеганограми, які практично не відрізняються від оригінальних зображень.

- Стеганографія в хмарних технологіях: Зі зростанням популярності хмарних сервісів виникає необхідність забезпечення конфіденційності даних, що зберігаються та передаються через них. Стеганографія може бути використана для приховування даних у хмарних сховищах або для захисту трафіку між хмарними сервісами та користувачами.

- Стеганографія в Інтернеті речей (IoT): Інтернет речей складається з великої кількості пристроїв, які збирають та передають дані. Стеганографія може бути використана для захисту цих даних від несанкціонованого доступу та забезпечення їх конфіденційності.

- Стеганографія в медичних зображеннях: Медичні зображення містять конфіденційну інформацію про пацієнтів, тому їх захист є особливо важливим. Стеганографія може бути використана для вбудовування даних

пацієнта, результатів аналізів або цифрових водяних знаків у томографічні знімки та інші медичні зображення.

Класифікація та методи стеганографії

Стеганографічні методи можна класифікувати за різними критеріями:

Зображення: JPEG, PNG, BMP, TIFF, GIF. Ці формати є популярними завдяки своїй поширеності та великому обсягу даних, що дозволяє вбудовувати значні обсяги інформації.

Аудіо: MP3, WAV, FLAC. Аудіофайли також є зручними контейнерами, особливо для прихованого передавання голосових повідомлень.

Відео: AVI, MP4, MKV. Відеофайли мають найбільшу ємність для прихованих даних, але їх обробка є більш складною.

Текст: PDF, DOCX, TXT. Текстові файли можуть використовуватися для стеганографії, наприклад, шляхом зміни відстаней між словами або символами, використання синонімів, зміни порядку слів у реченні, або навіть використання різних шрифтів.

Мережеві протоколи: TCP/IP, HTTP. Стеганографія в мережевих протоколах дозволяє приховувати дані в заголовках пакетів або в самому трафіку. Наприклад, можна змінювати порядок пакетів, час їх відправки або використовувати нестандартні значення полів заголовків.

За способом вбудовування:

- Заміна найменш значущих бітів (LSB): Цей метод є одним з найпростіших та найпоширеніших. Він полягає у заміні найменш значущих бітів контейнера бітами секретного повідомлення.

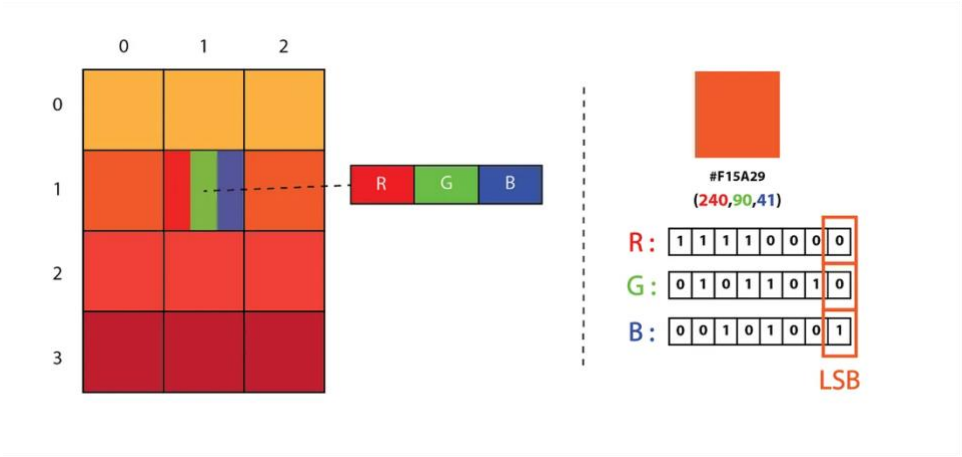


Рисунок1.1. Приклад вбудовування даних методом LSB:

-Вбудовування в області перетворення (DCT, DWT): Ці методи використовують особливості алгоритмів стиснення зображень JPEG та відео MPEG. Вони дозволяють вбудувати дані в коефіцієнти дискретного косинусного перетворення (DCT) або дискретного вейвлет-перетворення (DWT), що робить стеганограму більш стійкою до стис та обробки.

- Адаптивне вбудовування: Ці методи враховують особливості контейнера (наприклад, текстуру зображення) та вибирають оптимальні місця для вбудовування даних, що підвищує непомітність стеганограми. Наприклад, в областях з високою деталізацією можна вбудовувати менше даних, щоб уникнути помітних спотворень, а в областях з низькою деталізацією - більше.

- Матричне вбудовування: Цей метод використовує спеціальні матриці для вбудовування даних у контейнер. Він є більш складним у реалізації, але може забезпечити вищу стійкість до виявлення. Наприклад, можна використовувати матриці Костаса або матриці Голда для генерації псевдовипадкових послідовностей, які використовуються для вбудовування даних.

- Фазове кодування: Цей метод використовує фазові характеристики сигналу для вбудовування даних. Він є складним у реалізації, але може забезпечити високу непомітність та стійкість до деяких видів атак. Наприклад, можна змінювати фазу окремих частотних компонентів аудіосигналу або відеосигналу.

- Методи на основі палітри: Ці методи використовуються для вбудовування даних у зображення з індексованою палітрою кольорів (наприклад, GIF). Вони можуть бути досить ефективними, але мають обмежене застосування, оскільки більшість сучасних зображень використовують формати з повнокольоровою палітрою (JPEG, PNG).

- Розширене спектром вбудовування (Spread Spectrum): Цей метод полягає у розподіленні секретного повідомлення по всьому частотному спектру контейнера. Це робить стеганограму більш стійкою до виявлення та видалення, оскільки зміни в окремих частотних компонентах є менш помітними.

- Квантування зображення (Image Quantization): Цей метод використовує квантування коефіцієнтів перетворення (DCT або DWT) для вбудовування даних. Він може забезпечити високу ємність та непомітність, але вимагає ретельного налаштування параметрів квантування.

- Вбудовування в шумові компоненти: Цей метод використовує шумові компоненти зображення або аудіосигналу для вбудовування даних. Він може бути ефективним, оскільки людське око або вухо менш чутливі до змін у шумових компонентах.

- Методи на основі тексту: Ці методи використовують особливості тексту для вбудовування даних. Наприклад, можна змінювати відстані між

словами або символами, використовувати синоніми, змінювати порядок слів у реченні, або навіть використовувати різні шрифти.

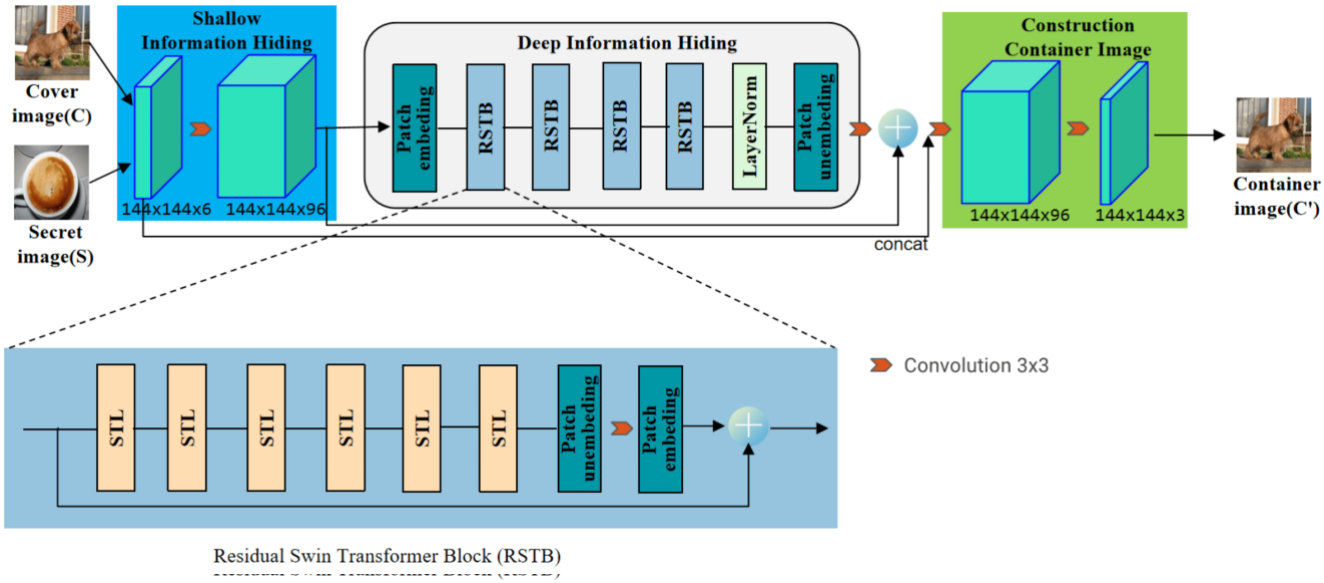


Рисунок 1.2. Наглядна класифікація методів стеганографії за типом контейнера:

За рівнем стійкості:

- Стійкі до стиснення: Ці методи зберігають вбудовані дані навіть після стиснення контейнера з використанням алгоритмів втратного стиснення, таких як JPEG для зображень або MP3 для аудіо.

- Стійкі до обробки зображень: Ці методи зберігають вбудовані дані навіть після застосування фільтрів, зміни яскравості, контрастності, різкості тощо

- Стійкі до геометричних перетворень**:** Ці методи зберігають вбудовані дані навіть після обертання, масштабування або обрізання зображення.

1.2 Застосування стеганографії в медицині: переваги та виклики

Стеганографія знаходить все більше застосування в медицині, зокрема для захисту томографічних знімків. Переваги використання стеганографії в цій сфері включають:

- Конфіденційність: Стеганографія дозволяє приховати конфіденційні дані пацієнта (ПІБ, діагноз тощо) безпосередньо в зображенні, що ускладнює їх несанкціоноване отримання. Це особливо важливо в умовах, коли медичні дані передаються через незахищені канали зв'язку або зберігаються на ненадійних носіях.

- Цілісність: Вбудовування цифрового водяного знаку або контрольної суми в зображення дозволяє перевірити його цілісність та виявити можливі підробки. Це допомагає забезпечити достовірність медичних знімків та запобігти їх фальсифікації.

- Автентичність: Стеганографічні методи можуть використовуватися для підтвердження авторства знімків та запобігання їх несанкціонованому використанню. Наприклад, лікар може вбудувати свій цифровий підпис у знімок, що підтверджує його справжність

Однак, застосування стеганографії в медицині пов'язане з певними викликами:

- Збереження діагностичної цінності: Вбудовування даних не повинно погіршувати якість зображення та ускладнювати його інтерпретацію лікарем. Це вимагає ретельного вибору методу стеганографії та параметрів вбудовування, щоб мінімізувати вплив на візуальну якість знімка.

- Стійкість до стиснення: Медичні зображення часто піддаються стисненню для зменшення обсягу даних. Стеганографічний алгоритм

повинен бути стійким до втрат інформації при стисненні, щоб забезпечити збереження вбудованих даних.

- Сумісність з медичними стандартами**:** Стеганографічна система повинна бути сумісною з форматом DICOM, що є стандартом для зберігання та передачі медичних зображень. Це забезпечить можливість використання системи в існуючій медичній інфраструктурі без необхідності внесення змін до неї.

- Безпека: Система повинна забезпечувати надійний захист вбудованих даних від несанкціонованого доступу, модифікації та вилучення. Для цього можуть використовуватися додаткові методи, такі як шифрування вбудованих даних з використанням сильних криптографічних алгоритмів

- Простота використання: Програмний інтерфейс системи повинен бути інтуїтивно зрозумілим та зручним для медичного персоналу, який не має спеціальних знань у галузі стеганографії. Це дозволить використовувати систему без необхідності проведення спеціального.

- Простота використання: Програмний інтерфейс системи повинен бути інтуїтивно зрозумілим та зручним для медичного персоналу, який не має спеціальних знань у галузі стеганографії. Це дозволить використовувати систему без необхідності проведення спеціального навчання.

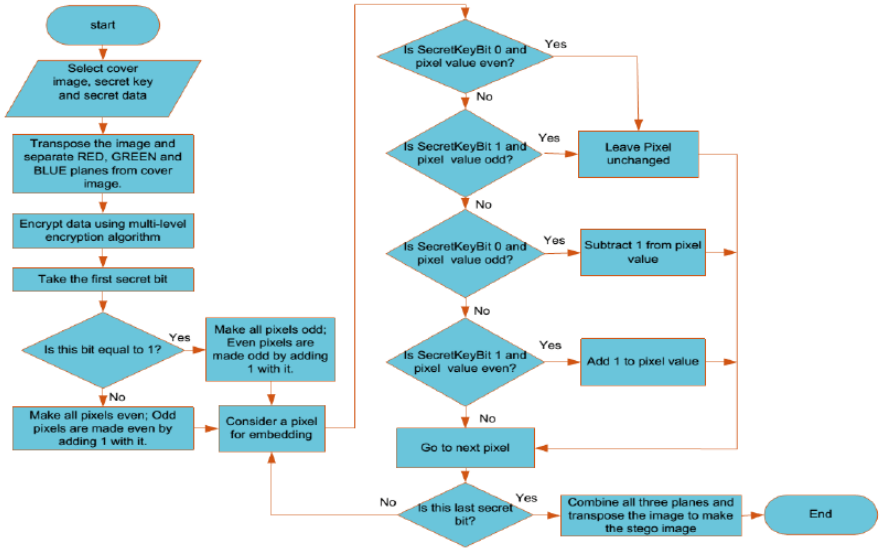


Рисунок 1.3. Загальна схема процесу вбудовування даних у контейнер:

1.3 Огляд сучасних досліджень та розробок у галузі стеганографії медичних зображень

Останні дослідження в галузі стеганографії медичних зображень зосереджені на розробці нових методів та алгоритмів, які забезпечують баланс між непомітністю, ємністю, стійкістю та збереженням діагностичної цінності зображень.

Деякі з найбільш перспективних напрямків досліджень включають:

- Реверсивна стеганографія (RDH): Цей підхід дозволяє повністю відновити оригінальне зображення після вилучення вбудованих даних. Це особливо важливо для медичних зображень, де будь-які зміни можуть вплинути на діагностику. RDH методи зазвичай використовують спеціальні алгоритми, які зберігають інформацію про внесені зміни, дозволяючи їх зворотне відновлення.

- Адаптивні методи: Ці методи враховують особливості зображення (наприклад, текстуру, контури) та вбудовують дані в області, де вони будуть

найменш помітними. Це дозволяє збільшити ємність стеганографічної системи без суттєвого впливу на якість зображення. Наприклад, для томографічних знімків можна використовувати області з низькою деталізацією (фон, однорідні тканини) для вбудовування більшого обсягу даних, а області з високою деталізацією (контури органів, патологічні зміни) залишити незмінними.

- Методи на основі машинного навчання: Використання нейронних мереж для розробки більш ефективних та стійких алгоритмів стеганографії. Наприклад, генеративно-змагальні мережі (GAN) можуть бути використані для створення стеганограм, які важко відрізнити від оригінальних зображень навіть за допомогою складних методів стегоаналізу.

- Стеганографія в стиснутих зображеннях: Розробка методів, які дозволяють вбудовувати дані безпосередньо в стиснуті зображення (JPEG, JPEG 2000), що зменшує обсяг даних та прискорює процес обробки. Це особливо актуально для медичних зображень, які часто зберігаються та передаються у стиснутому вигляді.

- Захист медичних даних на основі блокчейну: Використання технології блокчейн для забезпечення безпечного зберігання та передачі медичних зображень зі стеганографічним захистом. Блокчейн дозволяє створити децентралізовану та захищену від несанкціонованого доступу систему зберігання медичних даних, що підвищує їхню безпеку та конфіденційність.

1.4 Порівняльний аналіз методів стеганографії для медичних зображень

Таблиця 1.1 Наглядне порівняння методів

метод	непомітність	ємність	стійкість	Збереження діагностичної цінності
LSB	Висока	Низька	Низька	Висока
DCT	Середня	Середня	Висока	Середня
DWT	Середня	Середня	Висока	Середня
Матричне вбудовування	Низька	Висока	Висока	Низька
Фазове кодування	Висока	Низька	Висока	Висока
RDH	Висока	Низька	Середня	Висока

Як видно з таблиці, кожен метод має свої переваги та недоліки. Вибір оптимального методу залежить від конкретних вимог до системи захисту медичних зображень. Наприклад, якщо пріоритетом є непомітність та збереження діагностичної цінності знімків, то метод LSB може бути найкращим вибором. Однак, якщо потрібна висока стійкість до стиснення та обробки, то варто розглянути методи DCT або DWT.

специфіки медичних даних. У наступних розділах буде розглянуто особливості медичних томографічних зображень та розроблено програмне забезпечення для їх захисту з використанням стеганографії.

1.5 Висновки до розділу та постановка задачі

У цьому розділі було проведено огляд основ стеганографії, її методів та застосування у сфері медицини. Проаналізовано переваги та недоліки

різних методів стеганографії, включаючи LSB, DCT, DWT, матричне вбудовування, фазове кодування та RDH. Було розглянуто сучасні тенденції у дослідженнях стеганографії медичних зображень, такі як реверсивна стеганографія, адаптивні методи та використання машинного навчання.

Порівняльний аналіз методів показав, що вибір оптимального методу залежить від конкретних вимог до системи захисту. Наприклад, LSB забезпечує високу непомітність, але має низьку ємність та стійкість. DCT та DWT пропонують баланс між непомітністю, ємністю та стійкістю, але можуть вплинути на діагностичну цінність знімків.

Враховуючи вимоги до захисту томографічних знімків, такі як збереження діагностичної цінності, стійкість до стиснення та сумісність з медичними стандартами, було обрано метод цифрових водяних знаків (DWT) як найбільш перспективний для подальшої розробки. Цей метод дозволяє вбудовувати дані у зображення таким чином, щоб вони були непомітні для людського ока, але могли бути виявлені спеціальним програмним забезпеченням.

Для досягнення поставленої мети необхідно вирішити наступні задачі:

Розробити алгоритм вбудовування цифрового водяного знаку у томографічні знімки, який забезпечує високу непомітність та стійкість до різних типів атак, включаючи стиснення, фільтрацію та геометричні перетворення.

Розробити алгоритм виявлення цифрового водяного знаку, який дозволяє достовірно ідентифікувати наявність водяного знаку у зображенні та витягти вбудовану інформацію.

Реалізувати програмне забезпечення, яке впроваджує розроблені алгоритми та забезпечує зручний інтерфейс для користувачів.

Провести тестування розробленого програмного забезпечення на реальних томографічних знімках різних типів та оцінити його ефективність за критеріями непомітності, ємності, стійкості та збереження діагностичної цінності знімків.

РОЗДІЛ 2. ВИБІР МЕТОДУ ЗАХИСТУ ТОМОГРАФІЧНИХ ЗНІМКІВ ЗА ДОПОМОГОЮ ЦИФРОВИХ ВОДЯНИХ ЗНАКІВ

2.1 Аналіз предметної області та обґрунтування вибору методу цифрових водяних знаків

Захист медичних зображень, особливо томографічних знімків, є критично важливим завданням у сучасній медицині. Ці знімки містять конфіденційну інформацію про пацієнтів, таку як історія хвороби, результати обстежень та особисті дані. Несанкціоноване розголошення цієї інформації може призвести до серйозних наслідків, включаючи порушення прав пацієнта, дискримінацію та інші етичні та юридичні проблеми. Крім того, цілісність медичних зображень має вирішальне значення для точної діагностики та лікування. Будь-які несанкціоновані зміни або спотворення даних можуть призвести до неправильного діагнозу та потенційно шкідливого лікування.

Томографічні знімки, такі як комп'ютерна томографія (КТ) та магнітно-резонансна томографія (МРТ), є особливо вразливими через високу роздільну здатність та деталізацію анатомічних структур. Вони можуть розкривати не лише наявність патологій, але й індивідуальні особливості пацієнта, які можуть бути використані для ідентифікації або дискримінації. Тому захист цих знімків вимагає особливої уваги та ретельного підходу. Застосування цифрових водяних знаків є одним із ефективних методів захисту медичних зображень.

Існують різні методи захисту медичних зображень:

Шифрування: перетворює дані зображення в нечитабельний формат, забезпечуючи високий рівень безпеки. Однак, шифрування робить

зображення непридатними для медичного використання без попереднього розшифрування, що може бути незручним у клінічній практиці. Більше того, шифрування не захищає від несанкціонованої модифікації даних, що може бути критичним для діагностики та лікування.

Стеганографія: приховує конфіденційну інформацію безпосередньо в зображеннях, не впливаючи на їх візуальну якість. Але стеганографічні методи можуть бути вразливими до атак, спрямованих на виявлення та видалення прихованих даних. Крім того, стеганографія має обмежену ємність для вбудовування інформації, що може бути недостатнім для зберігання всіх необхідних метаданих медичного зображення.

Цифрові водяні знаки: вбудовують непомітний сигнал у зображення, який містить інформацію про авторство, власника або інші дані. Водяні знаки стійкіші до атак, ніж стеганографія, і не роблять зображення непридатними для використання. Вони можуть бути розроблені таким чином, щоб бути стійкими до різних маніпуляцій з зображенням, таких як стиснення, зміна розміру, фільтрація тощо.

У цьому проєкті ми будемо використовувати метод цифрових водяних знаків через його здатність забезпечувати захист авторських прав та цілісності медичних зображень, зберігаючи при цьому їх візуальну якість та медичну корисність. Це дозволяє використовувати зображення з водяним знаком безпосередньо для діагностики та лікування, не вимагаючи попереднього розшифрування

2.2 Алгоритм вбудовування та виявлення цифрового водяного знаку та UML-діаграма класів

Цифровий водяний знак вбудовується в томографічне зображення шляхом зміни його цифрового вмісту таким чином, щоб ці зміни були непомітні для людського ока, але могли бути виявлені спеціальним програмним забезпеченням. Процес виявлення водяного знаку полягає у виявленні цих змін та декодуванні вбудованої інформації

Таким чином, покроковий алгоритм буде складатися з таких кроків:
Алгоритм вбудовування:

Крок 1. Перетворення зображення: Оригінальне томографічне зображення перетворюється за допомогою дискретного вейвлет-перетворення (DWT). Це розкладає зображення на різні частотні піддіапазони, що дозволяє працювати з різними деталями зображення окремо

Крок 2. Генерація водяного знака: Створюється цифровий водяний знак, який може бути двійковим зображенням, логотипом, текстом або іншим типом даних. Водяний знак також перетворюється за допомогою DWT для відповідності частотним характеристикам зображення.

Крок 3. Вбудовування водяного знака: Коефіцієнти DWT водяного знака додаються до вибраних коефіцієнтів DWT зображення. Сила вбудовування (амплітуда) контролюється параметром, який визначає ступінь впливу водяного знака на зображення. Цей крок є критичним для забезпечення непомітності водяного знаку та його стійкості до атак.

Крок 4. Обернене перетворення: Модифіковані коефіцієнти DWT об'єднуються і перетворюються назад у просторову область за допомогою

оберненого DWT. Результатом є зображення з вбудованим водяним знаком, яке візуально майже не відрізняється від оригіналу.

Алгоритм виявлення:

Крок 1. Перетворення зображення: Зображення з підозрою на наявність водяного знаку перетворюється за допомогою DWT, аналогічно до процесу вбудовування.

Крок 2. Вилучення водяного знака: Коефіцієнти DWT водяного знака витягуються з тих же позицій, куди вони були вбудовані. Для цього використовується той самий ключ або алгоритм, що і при вбудовуванні.

Крок 3. Обернене перетворення: Вилучені коефіцієнти DWT перетворюються назад у просторову область за допомогою оберненого DWT. Результатом є вилучений водяний знак.

Крок 4. Порівняння: Вилучений водяний знак порівнюється з оригінальним водяним знаком. Для цього використовуються різні метрики подібності, такі як кореляція або порівняння біт. Якщо схожість достатньо висока, то наявність водяного знаку підтверджується.

Для реалізації програмного забезпечення, що впроваджує алгоритм цифрових водяних знаків, буде використана об'єктно-орієнтована парадигма програмування. UML-діаграма класів, що відображає структуру програми, представлена нижче:

парадигма програмування. UML-діаграма класів, що відображає структуру програми, представлена нижче:

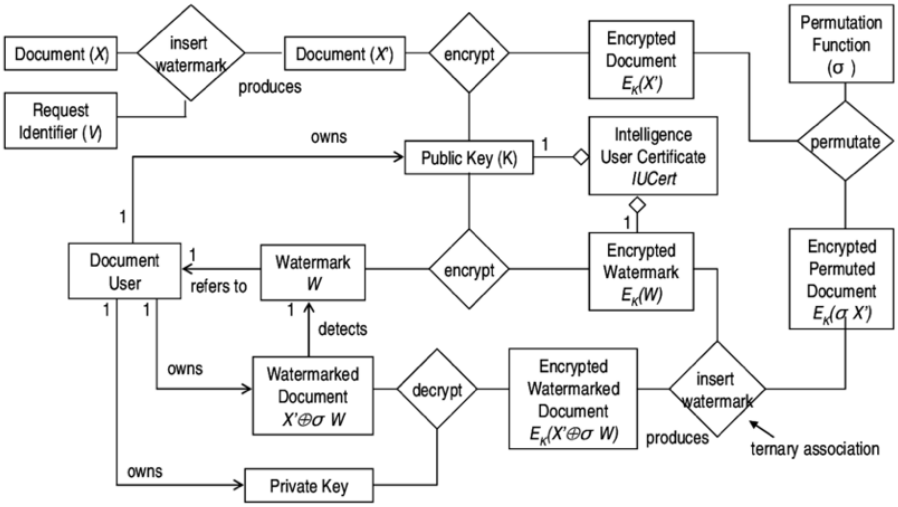


Рисунок 2.1 UML-діаграма класів, що відображає структуру програми

Опис класів:

WatermarkEmbedder: Відповідає за вбудовування водяних знаків у зображення.

WatermarkDetector: Відповідає за виявлення та вилучення водяних знаків із зображень.

DatabaseManager: Забезпечує зберігання та управління зображеннями та їх метаданими в базі даних.

AuthManager: Відповідає за автентифікацію користувачів та управління їх доступом до системи.

Програма має виконувати такі функції:

Вбудовування водяного знаку: Користувач може вибрати томографічне зображення та водяний знак (текст, логотип тощо), а програма вбудує цей водяний знак у зображення, зберігаючи його непомітність та стійкість до атак.

Виявлення водяного знаку: Користувач може вибрати зображення, і програма спробує виявити та вилучити з нього водяний знак. Якщо водяний знак присутній, програма відобразить його користувачеві.

Перевірка цілісності: Програма може перевірити, чи було зображення змінено після вбудовування водяного знаку. Це може бути корисним для виявлення несанкціонованих маніпуляцій з медичними даними.

Зберігання та управління зображеннями: Програма може зберігати зображення з водяними знаками у базі даних, забезпечуючи зручний доступ до них та можливість пошуку за різними критеріями.

Автентифікація користувачів: Програма може вимагати автентифікацію користувачів для доступу до функцій вбудовування та вилучення водяних знаків, забезпечуючи додатковий рівень безпеки.

2.3 Тестування та оцінка ефективності

Після реалізації алгоритму буде проведено комплексне тестування на різних наборах томографічних знімків, що включають різні типи зображень (КТ, МРТ), різні анатомічні області та різні патології.

Непомітність:

Суб'єктивна оцінка: Для суб'єктивної оцінки буде залучено групу медичних експертів (радіологів), які візуально оцінять якість зображень до та після вбудовування водяного знаку. Експерти будуть звертати увагу на будь-які артефакти, спотворення або зміни в кольоровій гамі, які можуть вплинути на інтерпретацію знімків. Для забезпечення об'єктивності оцінки буде використана стандартизована методологія, така як подвійний сліпий метод, коли експерти не знають, які зображення містять водяний знак.

Об'єктивна оцінка: Для кількісної оцінки змін у зображеннях після вбудовування водяного знаку будуть використані метрики якості зображень, такі як пікове відношення сигнал/шум (PSNR) та структурна подібність (SSIM). PSNR вимірює рівень шуму у зображенні, а SSIM оцінює структурну схожість між оригінальним та зміненим зображенням. Чим вищі значення PSNR та SSIM, тим менше помітний водяний знак та менший його вплив на якість зображення.

Ємність:

Експериментальне визначення: Для визначення ємності водяного знаку буде проведено серію експериментів, в яких різні обсяги даних (наприклад, текст різної довжини, логотипи різного розміру) будуть вбудовуватися в зображення. Максимальний обсяг даних, який можна вбудувати без помітного погіршення якості зображення (за результатами суб'єктивної та об'єктивної оцінки), буде визначений для кожного типу знімків. Це дозволить оцінити, наскільки ефективно можна використовувати водяний знак для зберігання додаткової інформації про зображення або пацієнта.

Стійкість:

Тестування на стійкість до навмисних атак: Зображення з вбудованими водяними знаками будуть піддані різним навмисним атакам, таким як:

Стиснення з втратами (JPEG): Зображення будуть стискатися з різними ступенями стиснення для визначення, наскільки водяний знак зберігається при втраті даних.

Фільтрація: На зображення будуть застосовуватися різні фільтри (наприклад, розмиття, підвищення різкості) для перевірки, чи не впливають вони на видимість або виявлення водяного знаку.

Геометричні перетворення: Зображення будуть піддаватися повороту, зміні масштабу, обрізанню та іншим геометричним перетворенням для оцінки стійкості водяного знаку до таких маніпуляцій.

Шум: До зображень буде додаватися шум різної інтенсивності для перевірки, чи не заважає він виявленню водяного знаку.

Тестування на стійкість до ненавмисних змін: Зображення будуть піддаватися типовим обробкам, які можуть відбуватися під час передачі, зберігання або перегляду зображень, таким як зміна яскравості, контрасту, колірного балансу тощо. Після кожної атаки буде здійснено спробу виявлення водяного знаку. Успішність виявлення водяного знаку буде оцінюватися для визначення стійкості методу до різних типів атак та змін.

Обчислювальна складність:

Вимірювання часу виконання: Час, необхідний для вбудовування та вилучення водяних знаків, буде вимірюватися для різних розмірів зображень та розмірів водяних знаків. Результати будуть використані для оцінки обчислювальної складності алгоритму та його придатності для використання в реальних умовах.

Вплив на діагностичну цінність:

Експертна оцінка: Радіологи оцінять, чи впливає наявність водяного знаку на їх здатність інтерпретувати зображення та ставити діагноз.

Порівняння з оригінальними зображеннями: Буде проведено порівняльний аналіз діагностичних результатів, отриманих на основі

оригінальних зображень та зображень з водяними знаками, щоб виявити будь-які потенційні розбіжності.

2.4 Висновки до розділу

У цьому розділі було представлено детальний опис методу цифрових водяних знаків та його застосування для захисту томографічних знімків. Було описано алгоритм вбудовування та вилучення водяних знаків, а також представлено UML-діаграму класів для кращого розуміння структури та взаємодії компонентів системи. Цей розділ закладає основу для подальшої розробки та реалізації програмного забезпечення, а також визначає ключові параметри, які будуть оцінюватися під час тестування для забезпечення ефективності та безпеки запропонованого методу.

РОЗДІЛ 3 РОЗРОБКА ТА ТЕСТУВАННЯ ПРОГРАМНОГО ДОДАТКУ

3.1 Вибір платформи та мови програмування

Вимоги до програмного забезпечення для розробки програми захисту томографічних знімків пацієнтів є одним з найважливіших етапів у процесі створення ефективного і надійного продукту. Цей процес включає детальний аналіз потреб і очікувань користувачів, визначення функціональних та нефункціональних вимог, а також врахування технічних обмежень і відповідних нормативних вимог.

Для початку необхідно чітко визначити цільову аудиторію програмного забезпечення. У випадку стеганографічного додатку для захисту медичних зображень, ключовими користувачами є медичні працівники, зокрема лікарі-рентгенологи, медичні техніки та адміністративний персонал медичних закладів. Кожна з цих груп має свої специфічні потреби і вимоги до програмного забезпечення. Наприклад, лікарі потребують зручного та швидкого доступу до зображень, а адміністративний персонал повинен мати змогу гарантувати безпеку та дотримання нормативних вимог щодо захисту даних.

Одним з основних аспектів є функціональні вимоги до програмного забезпечення. Програма повинна забезпечувати надійне приховування конфіденційної інформації у томографічних знімках за допомогою стеганографії. Це означає, що дані мають бути надійно приховані таким чином, щоб їх можна було витягти лише авторизованим користувачам. Крім того, програма повинна забезпечувати цілісність і достовірність захищених даних, щоб будь-які зміни або спроби несанкціонованого доступу могли бути виявлені.

Важливим аспектом є сумісність програмного забезпечення з форматом DICOM (Digital Imaging and Communications in Medicine), який є

стандартом для зберігання, передачі та обробки медичних зображень. Програма повинна підтримувати всі особливості цього формату, включаючи різні типи зображень та метадані. Це також передбачає інтеграцію з існуючими медичними інформаційними системами, що забезпечує безперебійний обмін даними між різними компонентами медичної інфраструктури.

Безпека є ще однією ключовою вимогою до програмного забезпечення. Оскільки програма працює з чутливими медичними даними, необхідно забезпечити високий рівень захисту від несанкціонованого доступу та витоку інформації. Це включає використання сучасних методів шифрування, автентифікації користувачів та контролю доступу на основі ролей. Програма повинна також мати функції журналювання подій, що дозволяє відстежувати всі дії користувачів та виявляти підозрілу активність.

Нефункціональні вимоги також мають велике значення. Програма повинна бути продуктивною, масштабованою та надійною. Вона повинна швидко обробляти зображення без значного впливу на загальну продуктивність системи, особливо у критичних медичних умовах. Масштабованість означає здатність системи обробляти збільшені обсяги даних та працювати з різною кількістю користувачів. Надійність включає здатність програми працювати стабільно в різних умовах та швидко відновлюватися після можливих збоїв.

Користувацький інтерфейс повинен бути інтуїтивно зрозумілим і зручним для медичних працівників різного рівня технічної підготовки. Інтерфейс має надавати чіткі та зрозумілі інструкції, швидкий доступ до основних функцій програми, а також підтримку багатомовності для користувачів з різних регіонів.

Нарешті, необхідно враховувати нормативні та етичні вимоги щодо обробки медичних даних. Програма повинна відповідати міжнародним і

національним стандартам захисту даних, таким як GDPR (General Data Protection Regulation) у Європейському Союзі або HIPAA (Health Insurance Portability and Accountability Act) у США. Це передбачає забезпечення конфіденційності, цілісності та доступності даних пацієнтів, а також можливість користувачів контролювати свої дані та отримувати інформацію про їх обробку.

Середовище розробки повинне відповідати певним критеріям, таким як підтримка обраної мови програмування, наявність інструментів для налагодження, тестування та інтеграції, а також зручність інтерфейсу.

На ринку існує багато популярних IDE, таких як Visual Studio, IntelliJ IDEA, PyCharm, Eclipse, NetBeans, та інші. Кожне з цих середовищ має свої переваги та недоліки, які слід враховувати при виборі.

Visual Studio є одним з найпопулярніших IDE, особливо для розробки на мові C# та інших мовах, що підтримуються платформою .NET. Воно пропонує широкий набір інструментів для розробки, налагодження та тестування, а також підтримку інтеграції з різними системами контролю версій, такими як Git. Visual Studio також забезпечує високий рівень зручності завдяки інтуїтивно зрозумілому інтерфейсу та потужним функціям автодоповнення коду. Однак, його великий розмір і вимоги до системних ресурсів можуть бути недоліком для деяких користувачів.

IntelliJ IDEA, розроблене компанією JetBrains, є універсальним IDE, яке підтримує багато мов програмування, включаючи Java, Kotlin, Python та інші. IntelliJ IDEA відоме своїми потужними інструментами для рефакторингу коду, інтеграцією з системами контролю версій та широкими можливостями для налагодження і тестування. Це середовище розробки часто обирають для створення великих і складних програмних проектів завдяки його високій продуктивності та багатофункціональності. Недоліками можуть бути висока вартість ліцензії для комерційного використання та певна складність у налаштуванні.

PyCharm, також від JetBrains, спеціалізується на розробці на мові Python. Це IDE пропонує всі переваги IntelliJ IDEA, зокрема потужні інструменти для рефакторингу коду, інтеграцію з системами контролю версій та широкі можливості для налагодження і тестування. PyCharm має додаткові функції, спеціально розроблені для Python, такі як підтримка Django, Flask та інших фреймворків, а також інструменти для наукового програмування. PyCharm є відмінним вибором для проектів, де основною мовою програмування є Python, але його вартість та системні вимоги можуть бути недоліком.

Eclipse є ще одним популярним IDE, яке підтримує багато мов програмування через систему плагінів. Воно відоме своєю гнучкістю та можливістю налаштування під конкретні потреби проекту. Eclipse підтримує інтеграцію з різними системами контролю версій, налагодженням та тестуванням, а також має великий вибір плагінів для розширення функціональності. Це середовище розробки є безкоштовним і відкритим, що робить його доступним для широкого кола користувачів. Однак, інтерфейс Eclipse може здаватися складним для новачків, а його продуктивність може знижуватися при роботі з великими проектами.

NetBeans, розроблене компанією Oracle, є ще одним безкоштовним і відкритим IDE, яке підтримує мови програмування, такі як Java, PHP, JavaScript та інші. NetBeans пропонує інтуїтивно зрозумілий інтерфейс, потужні інструменти для налагодження та тестування, а також інтеграцію з системами контролю версій. Це середовище розробки є відмінним вибором для Java-розробників завдяки його тісній інтеграції з екосистемою Java. Однак, як і у випадку з Eclipse, продуктивність NetBeans може знижуватися при роботі з великими проектами.

Під час вибору середовища розробки важливо також враховувати специфічні вимоги проекту, такі як необхідність підтримки певних фреймворків або бібліотек, а також особисті вподобання та досвід команди

розробників. Вибір правильного IDE може значно покращити продуктивність команди, забезпечити високу якість коду та знизити ризик помилок у програмному забезпеченні.

3.2 Реалізація програмного забезпечення

Архітектура визначає структуру системи, розподіл компонентів, їх взаємодію, а також забезпечує виконання основних функціональних і нефункціональних вимог. У випадку розробки стеганографічного додатку для захисту томографічних знімків пацієнтів, архітектура повинна бути ретельно спланована для забезпечення високої продуктивності, надійності та безпеки.

Процес проектування архітектури починається з визначення основних функціональних вимог до додатку. Для стеганографічного додатку це можуть бути функції приховування конфіденційної інформації у медичних знімках, збереження і передача захищених зображень, верифікація цілісності та автентичності даних, а також забезпечення доступу до захищених даних лише авторизованим користувачам. На основі цих вимог визначаються основні компоненти системи та їхня взаємодія.

Одним з ключових компонентів архітектури є модуль стеганографії, який відповідає за приховування та витягування інформації з медичних зображень. Цей модуль може включати різні алгоритми стеганографії, оптимізовані для роботи з томографічними знімками. Він також повинен забезпечувати високу продуктивність і точність обробки зображень, мінімізуючи вплив на якість медичних даних.

Другим важливим компонентом є модуль управління даними, який забезпечує збереження, організацію та доступ до захищених зображень. Цей модуль може включати базу даних для зберігання метаданих і захищених зображень, а також інтерфейси для доступу до даних з різних частин системи. Важливо, щоб цей модуль забезпечував високу надійність

і безпеку зберігання даних, а також підтримував масштабованість для роботи з великими обсягами інформації.

Іншим критично важливим компонентом є модуль автентифікації та авторизації, який відповідає за управління доступом до системи. Цей модуль повинен забезпечувати надійні механізми автентифікації користувачів, такі як паролі, двофакторна автентифікація або біометричні дані. Авторизація користувачів повинна базуватися на ролях, що дозволяє визначати, які дії можуть виконувати різні категорії користувачів. Це забезпечує захист конфіденційної інформації від несанкціонованого доступу.

Модуль інтерфейсу користувача є ще одним важливим компонентом архітектури. Він забезпечує взаємодію користувачів з системою через зручний і інтуїтивно зрозумілий інтерфейс. Інтерфейс повинен дозволяти користувачам легко виконувати основні операції, такі як завантаження зображень, приховування і витягування інформації, перегляд і аналіз захищених зображень. Для цього можуть бути використані різні технології, такі як веб-інтерфейси, десктопні додатки або мобільні додатки, залежно від потреб проекту.

Ще один важливий аспект проектування архітектури – це забезпечення безпеки і захисту даних. Для цього в архітектурі можуть бути передбачені різні механізми шифрування даних, як під час зберігання, так і під час передачі. Це дозволяє захистити дані від несанкціонованого доступу і забезпечити конфіденційність інформації. Також важливо врахувати заходи з забезпечення цілісності даних, такі як використання контрольних сум і цифрових підписів, що дозволяє виявляти будь-які зміни або підробки даних.

Нарешті, важливим аспектом проектування архітектури є забезпечення можливості масштабування і розширення системи. Це передбачає можливість додавання нових функцій і компонентів без значних

змін у вже існуючій архітектурі. Для цього можуть бути використані різні підходи, такі як мікросервісна архітектура, яка дозволяє розділити систему на незалежні компоненти, що можуть розвиватися і масштабуватися окремо один від одного.

Модуль Стеганографії

Основні функції модуля стеганографії включають дві основні операції: Encode Image (кодування зображення) та Decode Image (декодування зображення).

Функція Encode Image приймає три вхідні параметри: шлях до оригінального зображення (image_path), дані для приховування (data) і шлях для збереження закодованого зображення (output_path). Процес кодування складається з кількох кроків. Перший крок – це відкриття оригінального зображення. Після цього дані для приховування конвертуються у двійковий формат, щоб їх можна було вставити в пікселі зображення. Далі, кожен піксель зображення перебирається, і найменш значущі біти кольорових каналів пікселя замінюються на біти даних. Після завершення цього процесу зображення з прихованими даними зберігається за вказаною адресою (output_path). Функція повертає шлях до закодованого зображення.

Функція Decode Image приймає один вхідний параметр – шлях до закодованого зображення (image_path). Процес декодування також складається з кількох кроків. Спочатку відкривається закодоване зображення. Потім кожен піксель зображення перебирається для витягування найменш значущих бітів кольорових каналів, які містять приховані дані. Після цього біти збираються у байти і конвертуються у символи, утворюючи приховане повідомлення. Функція повертає декодовані дані.

```
from PIL import Image
import numpy as np
```

```
class Steganography:
  @staticmethod
  def encode_image(image_path, data, output_path):
    image = Image.open(image_path)
```

Відкриває зображення для подальшої роботи.

```
    encoded = image.copy()
    width, height = image.size
    data_bin = ''.join([format(ord(i), '08b') for i in data])
```

Перетворює вхідні дані у двійковий формат.

```
    data_len = len(data_bin)


 data_index = 0
    for y in range(height):
      for x in range(width):
        pixel = list(image.getpixel((x, y)))
        for n in range(3):
          if data_index < data_len:
            pixel[n] = int(format(pixel[n], '08b')[:-1] + data_bin[data_index], 2)
```

Вбудовує біт даних у найменш значущий біт колірного каналу пікселя.

```
            data_index += 1
        encoded.putpixel((x, y), tuple(pixel))
        if data_index >= data_len:
          break
      if data_index >= data_len:
        break
    encoded.save(output_path)
```

Зберігає зображення з вбудованими даними.

```
    return output_path


  @staticmethod
  def decode_image(image_path):
    image = Image.open(image_path)
    width, height = image.size
    binary_data = ""

    for y in range(height):
      for x in range(width):
        pixel = image.getpixel((x, y))
```

```
for n in range(3):
    binary_data += format(pixel[n], '08b')[-1]
```

Витягує найменш значущий біт з колірного каналу пікселя.

```
all_bytes = [binary_data[i:i+8] for i in range(0, len(binary_data), 8)]
decoded_data = "".join([chr(int(byte, 2)) for byte in all_bytes])
```

Перетворює витягнуті біти назад у вихідний текст

```
return decoded_data.split('\0')[0]
```

Модуль Управління Даними

Модуль управління даними відповідає за збереження, організацію та доступ до захищених зображень у базі даних.

Основні функції модуля управління даними включають створення таблиці для зберігання інформації про зображення, додавання нових зображень, отримання інформації про конкретне зображення та отримання списку всіх зображень.

Функція Create Table створює таблицю в базі даних для зберігання інформації про зображення. Ця таблиця включає поля для зберігання ідентифікатора зображення, імені файлу оригінального зображення та імені файлу закодованого зображення. Створення таблиці забезпечує структурований спосіб зберігання даних і полегшує доступ до них.

Функція Add Image приймає два вхідні параметри: ім'я файлу оригінального зображення (filename) та ім'я файлу закодованого зображення (encoded_filename). Процес додавання зображення включає вставку нового запису до таблиці, що містить інформацію про оригінальне та закодоване зображення. Це дозволяє зберігати інформацію про зображення в базі даних для подальшого використання.

Функція Get Image приймає один вхідний параметр – ідентифікатор зображення (image_id). Процес отримання інформації про зображення полягає в пошуку запису в таблиці за вказаним ідентифікатором. Функція

повертає інформацію про оригінальне та закодоване зображення, що дозволяє отримати доступ до необхідних даних за допомогою ідентифікатора.

Функція List Images не потребує вхідних параметрів і призначена для отримання списку всіх зображень, що зберігаються в базі даних. Процес включає витягнення всіх записів з таблиці, що містять інформацію про оригінальні та закодовані зображення. Ця функція забезпечує зручний спосіб перегляду всієї інформації про збережені зображення в одному місці.

```
import sqlite3

class DatabaseManager:
  def __init__(self, db_path='data.db'):
```

Ініціалізує об'єкт класу, встановлюючи з'єднання з базою даних та створюючи таблицю, якщо вона не існує.

```
    self.connection = sqlite3.connect(db_path)
    self.create_table()

  def create_table(self):
```

Створює таблицю images для зберігання інформації про зображення (оригінальне ім'я файлу та ім'я файлу з вбудованими даними).

```
    with self.connection:
      self.connection.execute('''CREATE TABLE IF NOT EXISTS images (
                  id INTEGER PRIMARY KEY,
                  filename TEXT NOT NULL,
                  encoded_filename TEXT NOT NULL)''')

  def add_image(self, filename, encoded_filename):
```

Додає новий запис до таблиці images з іменами оригінального та закодованого файлів.

```
    with self.connection:
      self.connection.execute('INSERT INTO images (filename, encoded_filename) VALUES (?, ?)',
                  (filename, encoded_filename))

  def get_image(self, image_id):
```

Отримує інформацію про зображення з бази даних за його ідентифікатором.

```
        cursor = self.connection.cursor()
        cursor.execute('SELECT filename, encoded_filename FROM images WHERE id=?', (image_id,))
        return cursor.fetchone()

    def list_images(self):
```

Повертає список всіх зображень, що зберігаються в базі даних.

```
        cursor = self.connection.cursor()
        cursor.execute('SELECT id, filename, encoded_filename FROM images')
        return cursor.fetchall()
```

Модуль Автентифікації та Авторизації

Модуль автентифікації та авторизації відповідає за управління доступом до системи, забезпечуючи безпеку і контроль доступу користувачів. Основні функції цього модуля включають додавання користувачів, хешування паролів і верифікацію користувачів.

Функція Add User приймає два вхідні параметри: ім'я користувача (username) і пароль (password). Процес починається з хешування пароля для забезпечення його безпеки. Використання хешування означає, що навіть якщо база даних буде скомпрометована, зловмисники не зможуть легко отримати оригінальні паролі користувачів. Хешований пароль додається до списку користувачів разом з ім'ям користувача, створюючи таким чином новий запис у системі.

Функція Hash Password приймає один вхідний параметр – пароль (password). Процес хешування здійснюється з використанням алгоритму SHA-256, який перетворює пароль у 256-бітне (32-байтне) число, представлене у вигляді рядка шістнадцяткових символів. Цей процес забезпечує, що пароль не зберігається у відкритому вигляді, що значно підвищує безпеку системи.

Функція Verify User приймає два вхідні параметри: ім'я користувача (username) і пароль (password). Процес верифікації починається з хешування введеного користувачем пароля таким самим способом, як це було зроблено під час додавання користувача. Потім хеш пароля порівнюється зі збереженим хешем для даного користувача. Якщо хеші збігаються, користувач вважається автентифікованим і отримує доступ до системи; в іншому випадку доступ забороняється.

Ці основні функції дозволяють модулю автентифікації та авторизації забезпечити надійний захист системи, зберігаючи паролі у безпечному вигляді та надаючи доступ до системи лише авторизованим користувачам.

```
import hashlib

class AuthManager:
  def __init__(self):
```

Ініціалізує об'єкт класу, створюючи порожній словник `users` для зберігання імен користувачів та їх хешованих паролів.

```
    self.users = {}

  def add_user(self, username, password):
```

Додає нового користувача до словника users. Хешує пароль перед збереженням.

```
    self.users[username] = self.hash_password(password)
```

Хешує пароль за допомогою алгоритму SHA256 та повертає хеш у вигляді шістнадцяткового рядка.

```
  def hash_password(self, password):
    return hashlib.sha256(password.encode()).hexdigest()

  def verify_user(self, username, password):
```

Перевіряє, чи пароль користувача відповідає збереженому хешованому паролю. Повертає True, якщо паролі збігаються, і False - якщо ні.

```
        hashed = self.hash_password(password)
        return self.users.get(username) == hashed

# Usage example
auth = AuthManager()
auth.add_user('admin', 'password123')
print(auth.verify_user('admin', 'password123'))  # Outputs True
print(auth.verify_user('admin', 'wrongpassword'))  # Outputs False
```

Модуль Інтерфейсу Користувача

Модуль інтерфейсу користувача забезпечує взаємодію користувачів з системою через зручний інтерфейс. У цьому прикладі реалізовано простий консольний інтерфейс.

Основні функції модуля інтерфейсу користувача включають автентифікацію, відображення меню, кодування зображень, розкодування зображень та перегляд списку зображень.

Функція Authentication запитує у користувача ім'я та пароль. Після отримання цих даних, функція виконує автентифікацію користувача за допомогою модуля AuthManager. Якщо автентифікація проходить успішно, користувач отримує доступ до системи, інакше доступ забороняється.

Функція Menu відображає меню з різними опціями, що дозволяють користувачу вибирати різні дії. Опції включають: закодувати зображення, розкодувати зображення, переглянути список зображень та вийти з програми. Кожна з цих опцій виконує відповідну функцію системи.

Функція Encode Image запитує у користувача шлях до зображення та дані для кодування. Потім ця функція використовує модуль Steganography для приховування даних у зображенні. Після успішного кодування, зображення зберігається, і інформація про нього додається до бази даних через модуль DatabaseManager.

Функція Decode Image запитує у користувача шлях до закодованого зображення. Використовуючи модуль Steganography, функція витягує приховані дані з зображення та відображає їх користувачу.

Функція View Images відображає список всіх зображень, що зберігаються у базі даних. Це дозволяє користувачу переглянути інформацію про всі доступні оригінальні та закодовані зображення.

Ці функції забезпечують просту та ефективну взаємодію користувача з системою через консольний інтерфейс, надаючи всі необхідні можливості для роботи зі стеганографічними зображеннями.

```
def main():
```

Головна функція, яка запускає весь додаток.

```
  auth = AuthManager()
```

Створює об'єкт для управління автентифікацією користувачів.

```
  db = DatabaseManager()
```

Створює об'єкт для взаємодії з базою даних зображень.

```
  # Adding users (for demonstration purposes)
  auth.add_user('admin', 'password123')
```

Додає користувача 'admin' з паролем 'password123' (для демонстрації).

```
  # Simple console interface
  print("Welcome to the Steganography Medical Image Protection System!")
  username = input("Enter username: ")
  password = input("Enter password: ")

  if auth.verify_user(username, password):
```

Перевіряє правильність введених користувачем імені та пароля.

```
    print("Authentication successful!")
    while True:
```

Запускає нескінченний цикл, який відображає меню та обробляє вибір користувача.

```
      print("\nMenu:")
```

```
        print("1. Encode image")
        print("2. Decode image")
        print("3. View images")
        print("4. Exit")
        choice = input("Choose an option: ")

        if choice == '1':
            image_path = input("Enter path to the image: ")
            data = input("Enter data to encode: ")
            output_path = input("Enter path to save encoded image: ")
            Steganography.encode_image(image_path, data, output_path)
```

Викликає функцію для кодування даних у зображення.

```
            db.add_image(image_path, output_path)
```

Додає інформацію про закодоване зображення до бази даних.

```
            print(f"Image encoded and saved to {output_path}")

        elif choice == '2':
            image_path = input("Enter path to the encoded image: ")
            decoded_data = Steganography.decode_image(image_path)
```

Викликає функцію для декодування даних із зображення.

```
            print(f"Decoded data: {decoded_data}")


        elif choice == '3':
            images = db.list_images()
```

Отримує список усіх зображень з бази даних.

```
            for img in images:
```

Цикл для виведення інформації про кожне зображення.

```
                print(f"ID: {img[0]}, Original: {img[1]}, Encoded: {img[2]}")

        elif choice == '4':
            print("Exiting the program.")
            break
        else:
            print("Invalid choice. Please try again.")

    else:
        print("Authentication failed.")

if __name__ == '__main__':
```

Створення користувацького інтерфейсу є важливим етапом у розробці програмного забезпечення, оскільки саме через інтерфейс користувачі взаємодіють з системою. У випадку стеганографічного додатку, реалізується простий консольний інтерфейс, який дозволяє користувачам виконувати основні операції: автентифікацію, кодування та декодування зображень, а також перегляд списку зображень. Розглянемо детально кожний етап створення цього інтерфейсу.

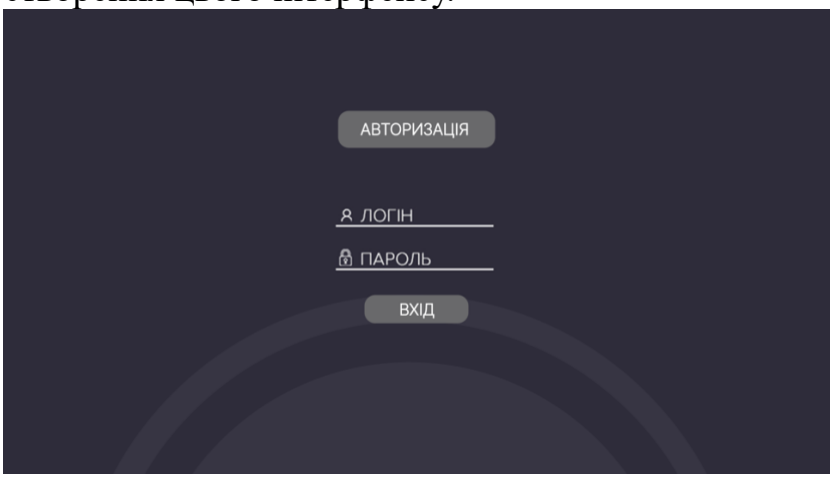


Рисунок 3.1 Авторизація

Перший крок у створенні користувацького інтерфейсу – це реалізація механізму автентифікації. Під час запуску додатку користувачеві буде запропоновано ввести ім’я користувача та пароль. Ці дані передаються до модуля AuthManager, який перевіряє правильність введених даних. Якщо автентифікація проходить успішно, користувач отримує доступ до основного меню додатку. У разі невдачі користувач отримає повідомлення про помилку.

Після успішної автентифікації користувач потрапляє до головного меню. Меню складається з кількох опцій: закодувати зображення, розкодувати зображення, переглянути список зображень та вийти з програми. Користувач може вибрати потрібну опцію, ввівши відповідний номер. Після вибору опції виконується відповідна функція додатку.

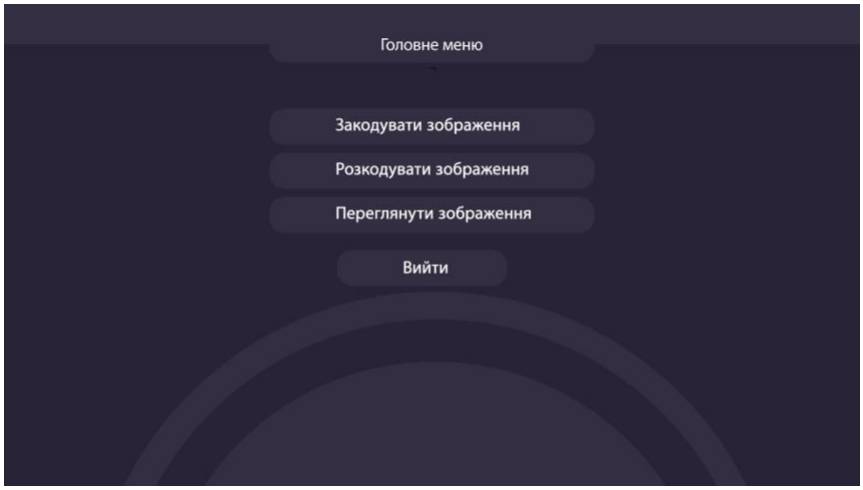


Рисунок 3.2 Головне меню

Для опції "Закодувати зображення" користувачеві пропонується ввести шлях до оригінального зображення та дані, які потрібно приховати. Після цього викликається функція модуля Steganography, яка виконує кодування даних у зображення. Закодоване зображення зберігається за вказаною адресою, а інформація про нього додається до бази даних через модуль DatabaseManager. Після успішного виконання користувач отримує повідомлення про те, що зображення було успішно закодовано.

Опція "Розкодувати зображення" дозволяє користувачеві витягти приховані дані з закодованого зображення. Користувач вводить шлях до закодованого зображення, після чого викликається відповідна функція модуля Steganography. Витягнуті дані відображаються користувачеві у консольному вікні.

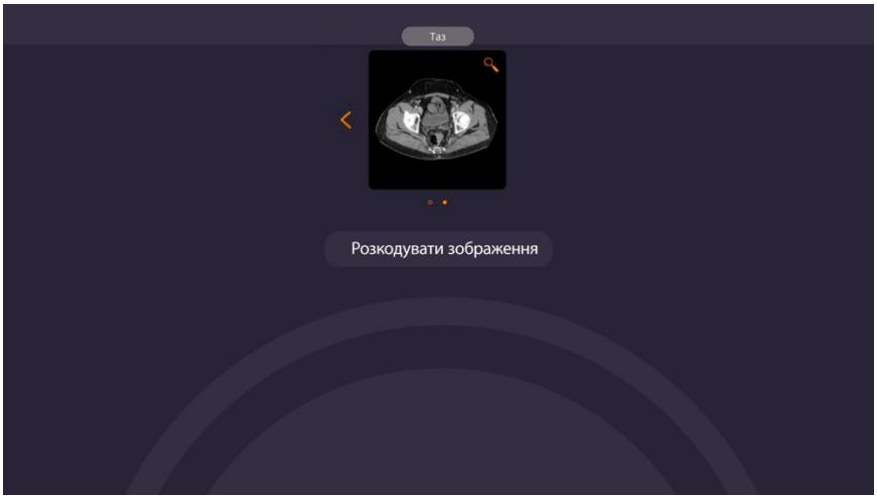


Рисунок 3.3 "Розкодувати зображення"

Опція "Переглянути список зображень" дозволяє користувачеві переглянути всі зображення, що зберігаються у базі даних. Викликається функція модуля DatabaseManager, яка витягує та відображає список усіх записів із таблиці зображень. Користувач може переглянути інформацію про всі оригінальні та закодовані зображення.

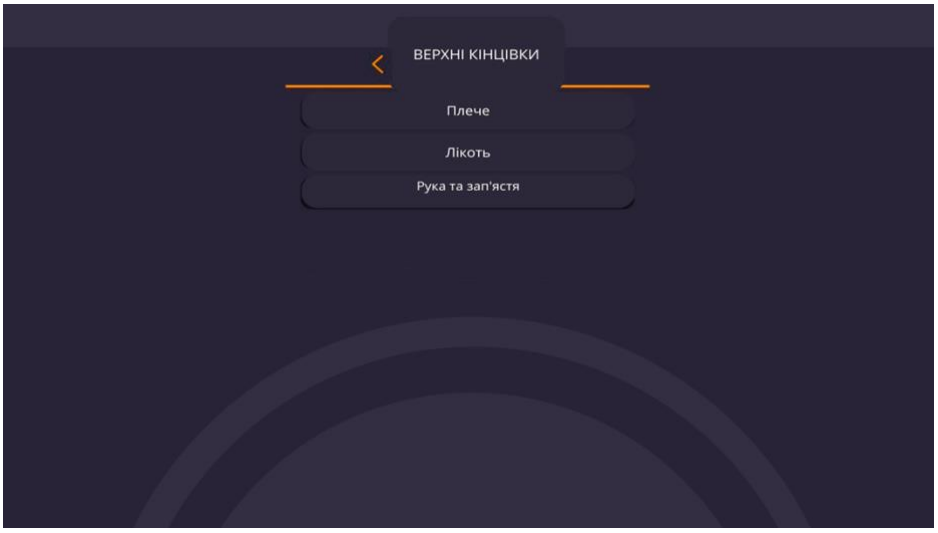


Рис. 3.4 Список зображень

Нарешті, опція "Вийти з програми" дозволяє користувачеві завершити роботу з додатком. Після вибору цієї опції програма припиняє своє виконання.

```
class Steganography:
  @staticmethod
```

Позначає методи класу як статичні, тобто їх можна викликати без створення екземпляра класу

```
  def encode_image(image_path, data, output_path):
    from PIL import Image
```

Імпортує модуль Image з бібліотеки Pillow для роботи з зображеннями.

```
    image = Image.open(image_path)
    encoded = image.copy()
    width, height = image.size
    data_bin = ''.join([format(ord(i), '08b') for i in data])
```

```
        data_len = len(data_bin)

        data_index = 0
        for y in range(height):
            for x in range(width):
                pixel = list(image.getpixel((x, y)))
                for n in range(3):
                    if data_index < data_len:
                        pixel[n] = int(format(pixel[n], '08b')[:-1] + data_bin[data_index], 2)
                        data_index += 1
                encoded.putpixel((x, y), tuple(pixel))
                if data_index >= data_len:
                    break
            if data_index >= data_len:
                break

        encoded.save(output_path)
        return output_path

    @staticmethod
    def decode_image(image_path):
```

Метод для декодування даних із зображення.

```
        from PIL import Image
        image = Image.open(image_path)
        width, height = image.size
        binary_data = ""

        for y in range(height):
            for x in range(width):
                pixel = image.getpixel((x, y))
                for n in range(3):
                    binary_data += format(pixel[n], '08b')[-1]

        all_bytes = [binary_data[i:i+8] for i in range(0, len(binary_data), 8)]
        decoded_data = "".join([chr(int(byte, 2)) for byte in all_bytes])
        return decoded_data.split('\0')[0]

import sqlite3

class DatabaseManager:
```

Визначає клас для керування базою даних зображень

```
    def __init__(self, db_path='data.db'):
```

```
        self.connection = sqlite3.connect(db_path)
```

Ініціалізує об'єкт класу, встановлюючи з'єднання з базою даних та створюючи таблицю, якщо вона не існує.

```
        self.create_table()
```

Створює таблицю images для зберігання інформації про зображення (оригінальне ім'я файлу та ім'я файлу з вбудованими даними).

```
    def create_table(self):
        with self.connection:
            self.connection.execute('''CREATE TABLE IF NOT EXISTS images (
                                id INTEGER PRIMARY KEY,
                                filename TEXT NOT NULL,
                                encoded_filename TEXT NOT NULL)''')

    def add_image(self, filename, encoded_filename):
```

Додає новий запис до таблиці images з іменами оригінального та закодованого файлів

```
        with self.connection:
            self.connection.execute('INSERT INTO images (filename, encoded_filename) VALUES (?, ?)',
                                    (filename, encoded_filename))

    def get_image(self, image_id):
```

Отримує інформацію про зображення з бази даних за його ідентифікатором.

```
        cursor = self.connection.cursor()
        cursor.execute('SELECT filename, encoded_filename FROM images WHERE id=?', (image_id,))
        return cursor.fetchone()

    def list_images(self):
```

Повертає список всіх зображень, що зберігаються в базі даних.

```
        cursor = self.connection.cursor()
        cursor.execute('SELECT id, filename, encoded_filename FROM images')
        return cursor.fetchall()

import hashlib

class AuthManager:
```

Визначає клас для керування автентифікацією користувачів.

```
  def __init__(self):
```

Ініціалізує об'єкт класу, створюючи порожній словник

```
    self.users = {}
  def add_user(self, username, password):
    self.users[username] = self.hash_password(password)
```

Зберігає хешований пароль користувача у словнику self.users.

```
  def hash_password(self, password):
    return hashlib.sha256(password.encode()).hexdigest()
```

Хешує пароль, використовуючи SHA-256.

```
  def verify_user(self, username, password):
    hashed = self.hash_password(password)
    return self.users.get(username) == hashed

# Main user interface
def main():
  auth = AuthManager()
  db = DatabaseManager()

  # Adding users (for demonstration purposes)
  auth.add_user('admin', 'password123')

  # Simple console interface
  print("Welcome to the Steganography Medical Image Protection System!")
  username = input("Enter username: ")
  password = input("Enter password: ")

  if auth.verify_user(username, password):
```

Перевіряє правильність введених користувачем імені та пароля.

```
    print("Authentication successful!")
    while True:
      print("\nMenu:")
      print("1. Encode image")
      print("2. Decode image")
      print("3. View images")
      print("4. Exit")
      choice = input("Choose an option: ")

      if choice == '1':
```

```
            image_path = input("Enter path to the image: ")
            data = input("Enter data to encode: ")
            output_path = input("Enter path to save encoded image: ")
            Steganography.encode_image(image_path, data, output_path)
```

Викликає функцію для кодування даних у зображення.

```
            db.add_image(image_path, output_path)
            print(f"Image encoded and saved to {output_path}")

        elif choice == '2':
            image_path = input("Enter path to the encoded image: ")
            decoded_data = Steganography.decode_image(image_path)
```

Викликає функцію для декодування даних із зображення.

```
            print(f"Decoded data: {decoded_data}")

        elif choice == '3':
            images = db.list_images()
            for img in images:
                print(f"ID: {img[0]}, Original: {img[1]}, Encoded: {img[2]}")

        elif choice == '4':
            print("Exiting the program.")
            break
        else:
            print("Invalid choice. Please try again.")

    else:
        print("Authentication failed.")

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

Для досягнення оптимальної продуктивності додатку, важливо врахувати кілька ключових аспектів: зниження використання ресурсів, поліпшення алгоритмів, ефективне управління пам'яттю, оптимізація вводу-виводу, а також забезпечення масштабованості та надійності.

По-перше, необхідно ретельно проаналізувати та оптимізувати алгоритми, що використовуються у додатку. У випадку стеганографічного додатку, це означає оптимізацію алгоритмів кодування та декодування

зображень. Один із способів покращити продуктивність – це зниження кількості операцій, які виконуються для кожного пікселя зображення, а також використання більш ефективних методів обробки даних. Наприклад, використання оптимізованих бібліотек для обробки зображень, таких як OpenCV, може значно підвищити швидкодію.

По-друге, важливо забезпечити ефективне управління пам'яттю. Це включає мінімізацію використання пам'яті шляхом звільнення непотрібних об'єктів та даних, а також уникнення витоків пам'яті. В мові програмування Python, це може бути досягнуто за допомогою правильного використання збирача сміття (garbage collector) та обережного управління об'єктами. Використання структур даних, що оптимізують пам'ять, таких як масиви NumPy, також може сприяти підвищенню ефективності.

Третім аспектом є оптимізація вводу-виводу. Швидкість роботи з файлами та базами даних може суттєво впливати на загальну продуктивність додатку. У випадку роботи з великими медичними зображеннями, важливо використовувати методи пакетного оброблення даних та асинхронний ввід-вивід, щоб знизити затримки та підвищити швидкість обробки. Крім того, оптимізація доступу до бази даних, включаючи використання індексів та кешування запитів, може значно покращити продуктивність.

Забезпечення масштабованості додатку також є важливим аспектом оптимізації. Масштабованість дозволяє додатку ефективно обробляти зростаючі обсяги даних та збільшену кількість користувачів. Використання мікросервісної архітектури може сприяти розподілу навантаження між різними компонентами системи, що дозволяє легко додавати нові функції та масштабувати додаток горизонтально. Крім того, впровадження механізмів балансування навантаження та автоматичного масштабування дозволяє адаптуватися до змін у навантаженні в режимі реального часу.

3.3 Верифікація та тестування програмного додатку

Тестові сценарії для програми захисту томографічних знімків пацієнтів за допомогою стеганографії

Таблиця 3.1 Тестові сценарії для програми кодування, декодування даних в зображеннях

Тестовий сценарій	Опис	Кроки	Очікуваний результат
1.Кодування данних зображення.	Перевірка функції кодування даних у зображення.	1.Відкрити програму. 2.Виконати антифікацію користувача (ввести імя та пароль.) 3.Вибрати опцію «Закодувати зображення» у головному меню. 4.Ввести шлях до оригінального зображення.	Програма успішно кодує дані у зображення. Закодоване зображення зберігається за вказаною адресою. Відображається повідомлення про успішне завершення операцій.

Продовження таблиці 3.1

		5.Ввести дані для кодування. 6.Ввести шлях для збереження закодованого зображення. 7.Натиснути кнопку «Підтвердити».	
2.Декодування даних зображення	Перевірка функції декодування даних із законодавчого зображення	1.Відкрити програму. 2.Виконати афтентифікацію користувача. 3.Вибрати опцію «Розкодувати зображення» у головному меню 4. Ввести шлях до закодованого зображення. 5.Натиснути кнопку «Підтвердити».	Програма успішно розкодовує дані із зображення. Відображаються правильні розкодувані дані. Якщо зображення не містить закодованих даних,відображається відповідне повідомлення.

Продовження тааблиці 3.1

3.Автентифікація користувачів (успішна)	Перевірка функції автентифікації користувачів з правильними даними.	1.Відкрити програму. 2.Вставити правильне ім'я користувача та пароль. 3.Натиснути кнопку «Увійти».	Користувач успішно проходить автентифікацію та отримує доступ до головного меню програми.
.Збереження та отримання зображень з бази даних	Перевірка функції зображення та отримання зображень з бази даних.	1.Відкрити програму. 2.Виконати автентифікацію корисстувача. 3.Вибрати опцію «Закодувати зображення» та виконати процес кодуання зобрження.	Програма відображає список зображених зображень із бази даних,влючаючи закодоване зображення. Відображається інформація про оригінальні та закодовані зображення.

Продовження таблиці 3.1

5.Перевірка коректності роботи інтерфейсу користувача.	Перевірка функціональності та зручності інтерфейсу користувача.	1.Відкрити програму. 2.Виконати автентифікацію корисувача. 3.Перемикатися між різними опціями головного меню. 4.Виконари різні операції,такі як кодування, декодування та перегляд зображень.	Інтерфейс працює конкретно без збоїв. Всі опції меню доступні і виконуються відповідні функції. Всі повідомлення та інструкції зрозумілі та конкрені.

Модульне тестування є першим етапом і зосереджується на перевірці окремих модулів програми, таких як кодування та декодування зображень, автентифікація користувачів та управління базою даних. Основна мета модульного тестування полягає в тому, щоб переконатися, що кожен модуль працює відповідно до своїх специфікацій. Для модуля кодування зображень тестові випадки можуть включати перевірку коректного кодування даних у зображення різних форматів, обробку помилок при введенні некоректних даних та перевірку ефективності алгоритму. Для модуля декодування зображень тестові випадки можуть включати перевірку точності

витягування даних з різних зображень та обробку випадків, коли зображення не містять закодованих даних. Кожен модуль тестується ізольовано за допомогою автоматизованих тестів, таких як pytest для Python. Результати тестування фіксуються, аналізуються, і в разі виявлення помилок модулі коригуються і повторно тестуються.

Інтеграційне тестування проводиться після успішного завершення модульного тестування і зосереджується на перевірці взаємодії між модулями. Мета інтеграційного тестування полягає в тому, щоб переконатися, що всі модулі правильно інтегруються і взаємодіють один з одним відповідно до вимог системи. Наприклад, модуль кодування зображень повинен успішно інтегруватися з модулем управління базою даних для збереження закодованих зображень, а також з модулем автентифікації для забезпечення доступу до функцій лише авторизованим користувачам. Інтеграційні тестові випадки можуть включати перевірку коректності збереження та отримання закодованих зображень з бази даних, а також перевірку взаємодії між інтерфейсом користувача та функціями програми. Інтеграційні тести також виконуються автоматизовано, і результати фіксуються для подальшого аналізу. Проблеми у взаємодії між модулями виявляються, виправляються і повторно тестуються.

Системне тестування є заключним етапом і зосереджується на перевірці всієї системи в цілому. Основна мета системного тестування полягає в тому, щоб переконатися, що програмне забезпечення відповідає всім функціональним і нефункціональним вимогам, включаючи продуктивність, безпеку, сумісність та надійність. Для стеганографічного додатку системні тестові випадки можуть включати сценарії кодування та декодування зображень різних форматів, перевірку автентифікації користувачів, тестування продуктивності при обробці великих обсягів даних, перевірку безпеки при збереженні та передачі конфіденційних даних, а також перевірку роботи додатку на різних операційних системах і

пристроях. Системні тести виконуються на реальному або максимально наближеному до реального середовищі, використовуючи інструменти для автоматизації тестування та моніторингу системи. Аналіз результатів тестування дозволяє виявити проблеми та недоліки в роботі системи. У разі виявлення проблем вони виправляються, і тестування повторюється. Результати системного тестування документуються для забезпечення прозорості процесу та підтвердження відповідності програмного забезпечення всім вимогам.

3.4 Використання та впровадження програмного продукту

Одним із практичних прикладів є застосування додатку у великих медичних центрах, де щоденно обробляються тисячі томографічних знімків. В таких установах важливо гарантувати, що доступ до зображень мають тільки авторизовані медичні працівники, а самі знімки захищені від несанкціонованого доступу. Стеганографічний додаток дозволяє приховати конфіденційну інформацію, таку як ідентифікаційні дані пацієнтів, безпосередньо в зображеннях. Це забезпечує додатковий рівень безпеки, оскільки навіть у випадку витоку даних, отримати доступ до прихованої інформації без спеціального декодування буде неможливо.

Інший приклад стосується обміну медичними знімками між різними медичними установами. Пацієнти часто звертаються до різних лікарів або проходять обстеження в різних клініках, що потребує передачі знімків між цими установами. Використання стеганографічного додатку дозволяє безпечно передавати знімки, оскільки навіть якщо зображення буде перехоплено під час передачі, приховані в ньому дані залишаться захищеними. Це особливо актуально в умовах телемедицини, де обмін даними відбувається через інтернет і ризики кібератак значно підвищуються.

Додаток також може бути використаний у дослідницьких установах, де проводяться клінічні дослідження з використанням томографічних знімків. У таких випадках важливо забезпечити конфіденційність даних пацієнтів, які беруть участь у дослідженнях. Стеганографічний захист дозволяє приховати ідентифікаційну інформацію в знімках, що робить їх анонімними, але водночас доступними для дослідників. Це допомагає дотримуватися етичних норм і законодавчих вимог щодо захисту персональних даних.

Важливим аспектом використання стеганографічного додатку є забезпечення цілісності медичних знімків. У випадку, коли знімки використовуються для постановки діагнозу або планування лікування, важливо гарантувати, що вони не були змінені або підроблені. Додаток може забезпечити контроль цілісності зображень шляхом використання цифрових підписів або контрольних сум, прихованих у зображеннях. Це дозволяє медичним працівникам перевіряти автентичність знімків і бути впевненими в їхній достовірності.

Ще один практичний приклад – використання стеганографічного додатку для навчання медичних студентів і молодих спеціалістів. У процесі навчання часто використовуються реальні медичні знімки, і важливо забезпечити конфіденційність даних пацієнтів. Стеганографічний захист дозволяє приховати персональні дані, зберігаючи при цьому всі діагностичні та навчальні цінності зображень. Це забезпечує можливість використовувати реальні випадки для навчання без ризику порушення конфіденційності пацієнтів.

Інструкції з встановлення та використання програмного забезпечення для захисту томографічних знімків пацієнтів за допомогою стеганографії мають бути детальними, зрозумілими та охоплювати всі етапи від завантаження та встановлення до практичного використання основних функцій програми. Нижче наведено повний опис цього процесу.

Першим кроком є завантаження програмного забезпечення. Для цього користувач повинен перейти на офіційний веб-сайт розробника або інший надійний ресурс, де розміщено дистрибутив програми. Після цього слід знайти відповідну версію для своєї операційної системи (Windows, macOS або Linux) і натиснути на посилання для завантаження. Завантаження файлу розпочнеться автоматично, і користувачеві буде запропоновано зберегти його у бажаному місці на комп'ютері.

Після завершення завантаження наступним етапом є встановлення програми. Користувачеві потрібно знайти завантажений файл (зазвичай це інсталяційний файл з розширенням .exe для Windows, .dmg для macOS або .deb/.rpm для Linux) та запустити його. Відкриється майстер встановлення, який проведе користувача через всі необхідні кроки. Спочатку з’явиться вітальне вікно, де буде запропоновано почати процес встановлення. Користувачеві потрібно натиснути кнопку "Далі" для продовження.

На наступному етапі майстер встановлення запропонує ознайомитися з ліцензійною угодою. Користувач повинен уважно прочитати текст угоди і, якщо погоджується з її умовами, поставити галочку біля пункту "Я приймаю умови ліцензійної угоди" та натиснути кнопку "Далі". Після цього користувачеві буде запропоновано вибрати папку для встановлення програми. За замовчуванням це буде системний диск, але користувач може вибрати будь-яке інше місце на своєму комп'ютері. Після вибору папки потрібно натиснути кнопку "Далі".

Наступним кроком буде вибір компонентів для встановлення. Зазвичай користувачеві пропонується встановити основні компоненти програми та додаткові модулі або плагіни. Для більшості користувачів достатньо залишити всі параметри за замовчуванням, але в деяких випадках може знадобитися вибрати додаткові компоненти. Після вибору необхідних компонентів потрібно натиснути кнопку "Встановити".

Розпочнеться процес копіювання файлів і налаштування системи, що може зайняти кілька хвилин.

Після завершення встановлення користувачеві буде запропоновано завершити роботу майстра встановлення. На цьому етапі можна вибрати параметр "Запустити програму після завершення встановлення", щоб одразу перейти до роботи з програмою. Натиснувши кнопку "Завершити", користувач зможе відкрити програму і почати її використання.

Після встановлення програми наступним кроком є налаштування облікового запису користувача. При першому запуску програми користувачеві буде запропоновано створити обліковий запис, якщо він ще не має облікового запису, або увійти в існуючий. Для створення облікового запису потрібно ввести необхідні дані, такі як ім'я користувача, пароль і електронну пошту. Після успішного створення або входу в обліковий запис користувач потрапить на головний екран програми.

Головний екран програми містить основні функції, такі як кодування зображень, декодування зображень, перегляд збережених зображень та управління налаштуваннями. Для кодування зображення користувачеві потрібно вибрати опцію "Закодувати зображення". Відкриється нове вікно, де потрібно вказати шлях до оригінального зображення, ввести дані для кодування та вибрати місце для збереження закодованого зображення. Після введення всіх необхідних даних потрібно натиснути кнопку "Підтвердити", і програма виконає кодування. Закодоване зображення буде збережене в зазначеному місці, і користувач отримає повідомлення про успішне завершення операції.

Для декодування зображення користувачеві потрібно вибрати опцію "Розкодувати зображення". У новому вікні потрібно вказати шлях до закодованого зображення та натиснути кнопку "Підтвердити". Програма виконає декодування, і приховані дані будуть відображені на екрані. Якщо

зображення не містить закодованих даних, користувач отримає відповідне повідомлення.

У розділі "Перегляд зображень" користувач може переглядати всі зображення, збережені в базі даних програми. Тут відображаються як оригінальні, так і закодовані зображення з відповідною інформацією про них. Користувач може вибрати будь-яке зображення для детального перегляду або для виконання додаткових операцій, таких як видалення або повторне кодування.

Розділ "Налаштування" дозволяє користувачеві змінювати параметри програми, такі як місце збереження зображень, налаштування безпеки, а також налаштування облікового запису. Тут можна змінити пароль, налаштувати двофакторну автентифікацію для додаткового захисту облікового запису, а також налаштувати параметри автоматичного збереження та резервного копіювання даних.

Таким чином, інструкції з встановлення та використання програмного забезпечення для захисту томографічних знімків пацієнтів за допомогою стеганографії охоплюють всі необхідні кроки від завантаження та встановлення до налаштування облікового запису і використання основних функцій програми. Детальний підхід до кожного етапу забезпечує зрозумілість і зручність для користувачів, що дозволяє ефективно використовувати програму для захисту медичних даних.

У сучасних умовах зростаючої важливості конфіденційності медичних даних і частих кібератак, використання стеганографічних методів забезпечує додатковий рівень безпеки для медичних зображень.

В одному з великих медичних центрів, де щоденно обробляються тисячі томографічних знімків, було впроваджено стеганографічний додаток для захисту медичних даних. До цього часу центр стикався з проблемами витоку даних та несанкціонованого доступу до зображень пацієнтів. Застосування стеганографії дозволило інтегрувати приховані дані, такі як

ідентифікаційні номери пацієнтів і медичні записи, безпосередньо в зображення. Це забезпечило додатковий рівень захисту, оскільки навіть у випадку витоку або крадіжки файлів, отримати доступ до прихованих даних без спеціального декодування було неможливо. В результаті, медичний центр зміг значно підвищити рівень безпеки даних і зменшити ризики, пов'язані з несанкціонованим доступом.

Інший реальний кейс стосується компанії, яка надає телемедичні послуги. В умовах дистанційної медицини обмін медичними знімками через інтернет є звичайною практикою. Однак передача таких даних часто піддається ризику кібератак. Впровадження стеганографічного додатку дозволило безпечно передавати томографічні знімки між лікарями та пацієнтами. Стеганографія забезпечила захист конфіденційної інформації, приховуючи її в зображеннях, що зменшило ризик компрометації даних під час передачі. Це підвищило довіру пацієнтів до телемедичних послуг та дозволило компанії дотримуватися стандартів безпеки даних.

У дослідницькому інституті, який проводить клінічні дослідження з використанням томографічних знімків, було впроваджено стеганографічний додаток для забезпечення конфіденційності даних учасників досліджень. Раніше інститут стикався з проблемою анонімізації даних, що вимагало значних зусиль для видалення ідентифікаційної інформації з кожного знімка. Стеганографія дозволила автоматично приховувати ідентифікаційні дані в зображеннях, роблячи їх доступними для дослідників, але захищаючи конфіденційність пацієнтів. Це значно спростило процес обробки даних, забезпечило відповідність етичним стандартам і законодавчим вимогам.

Один з медичних закладів, що використовує хмарні технології для зберігання томографічних знімків, впровадив стеганографічний додаток для додаткового захисту даних. Враховуючи ризики, пов'язані з використанням хмарних сервісів, стеганографія забезпечила захист даних

навіть у випадку компрометації хмарного сховища. Приховані дані залишалися недоступними для зловмисників без спеціального декодування. Це дозволило медичному закладу використовувати переваги хмарних технологій, такі як масштабованість та доступність, без компромісів щодо безпеки даних.

Університетська лікарня, що займається підготовкою медичних студентів і молодих спеціалістів, впровадила стеганографічний додаток для захисту навчальних матеріалів. Реальні медичні знімки, що використовуються в навчальному процесі, містили конфіденційну інформацію пацієнтів. Стеганографія дозволила приховувати персональні дані, зберігаючи при цьому всі діагностичні та навчальні цінності зображень. Це забезпечило можливість використовувати реальні клінічні випадки в навчанні без ризику порушення конфіденційності пацієнтів, підвищуючи якість медичної освіти.

У випадку віддалених консультацій між лікарями з різних медичних закладів, стеганографічний додаток допоміг забезпечити безпечний обмін томографічними знімками. Лікарі змогли приховувати діагностичні висновки та інші конфіденційні дані безпосередньо в зображеннях, передаючи їх через захищені канали зв'язку. Це дозволило підвищити точність діагнозів завдяки обміну думками між спеціалістами, зберігаючи при цьому конфіденційність пацієнтів.

Таким чином, реальні кейси застосування стеганографічного додатку для захисту томографічних знімків пацієнтів демонструють його ефективність і важливість у забезпеченні безпеки медичних даних. Використання стеганографії дозволяє медичним закладам підвищити рівень захисту інформації, дотримуватися законодавчих вимог, покращувати якість медичних послуг і освіти, а також забезпечувати безпечний обмін даними між установами. Це підвищує довіру пацієнтів до

медичних закладів і допомагає їм ефективно справлятися з викликами сучасного цифрового світу.

Процес отримання зворотного зв'язку починається з активного залучення користувачів. Важливо створити канали для комунікації, які дозволять користувачам легко надсилати свої відгуки та пропозиції. Це можуть бути спеціалізовані форми зворотного зв'язку в інтерфейсі програми, електронна пошта, форуми підтримки або соціальні мережі. Користувачі повинні мати змогу швидко і без зайвих зусиль повідомляти про свої враження від роботи з програмою, виявлені помилки та ідеї щодо покращення.

Збір зворотного зв'язку має бути систематизованим. Для цього використовуються різні інструменти, такі як системи управління відгуками або CRM-системи, які дозволяють фіксувати, класифікувати та аналізувати отримані дані. Важливо забезпечити, щоб кожен відгук був розглянутий і не залишився без уваги. Для цього призначають відповідальних осіб або команди, які займаються обробкою зворотного зв'язку, відстежують тенденції та виявляють найважливіші проблеми, що потребують вирішення.

Після збору зворотного зв'язку наступним кроком є його аналіз. Важливо виділити ключові моменти та визначити пріоритети для впровадження змін. Наприклад, якщо більшість користувачів скаржаться на складність інтерфейсу або незрозумілість окремих функцій, це може свідчити про необхідність удосконалення користувацького досвіду. Аналіз відгуків також дозволяє виявити технічні проблеми, такі як помилки в роботі додатку, проблеми з продуктивністю або безпекою. Всі ці дані систематизують та пріоритизують для подальшої роботи.

На основі аналізу зворотного зв'язку розробляють план впровадження покращень. Цей план включає конкретні дії для вирішення виявлених проблем, строки їх виконання та відповідальних осіб.

Наприклад, якщо було виявлено, що користувачі стикаються з труднощами при використанні функції кодування зображень, план може включати покращення інтерфейсу цієї функції, додавання інструкцій або відеоуроків, а також оптимізацію самого процесу кодування для підвищення його швидкості та зручності.

Важливим аспектом є комунікація з користувачами про впроваджені зміни. Користувачі повинні знати, що їхні відгуки були почуті та враховані. Це роблять через оновлення програми, де в примітках до оновлень вказують виправлені помилки та нові функції, впроваджені на основі зворотного зв'язку. Також корисно проводити регулярні опитування та інтерв'ю з користувачами, щоб дізнатися їхню думку про впроваджені покращення та зібрати нові ідеї для подальшого розвитку продукту.

Після впровадження покращень важливо повторно перевірити програму, щоб переконатися, що нові функції працюють належним чином і не викликали нових проблем. Для цього проводиться тестування, яке включає як автоматизовані тести, так і ручне тестування за участю користувачів. Повторне тестування допомагає переконатися, що всі зміни були успішно впроваджені і не вплинули негативно на інші аспекти програми.

3.5. Висновки до розділу

У цьому розділі було детально розглянуто процес розробки та тестування програмного додатку для захисту томографічних знімків пацієнтів за допомогою стеганографії. Були визначені вимоги до програмного забезпечення, що включають функціональні аспекти, такі як кодування та декодування даних, а також нефункціональні аспекти, такі як продуктивність, безпека та зручність використання.

Було обрано мову програмування Python та середовище розробки, що відповідає вимогам проекту та забезпечує ефективність розробки. Було реалізовано програмне забезпечення, що включає модулі стеганографії, управління даними, автентифікації та авторизації, а також користувацький інтерфейс.

Було проведено тестування програмного додатку, включаючи модульне, інтеграційне та системне тестування. Це дозволило перевірити коректність роботи окремих модулів, їх взаємодію та відповідність програмного забезпечення всім вимогам.

Крім того, було розглянуто практичні аспекти використання та впровадження програмного продукту в реальних умовах. Було описано процес встановлення та використання програми, а також наведено приклади її застосування в медичних установах, телемедицині, дослідженнях та навчанні.

Результати цього розділу підтверджують, що розроблений стеганографічний додаток є ефективним інструментом для захисту томографічних знімків пацієнтів. Він забезпечує конфіденційність, цілісність та автентичність медичних даних, дозволяючи використовувати їх у різних сферах медицини без ризику несанкціонованого доступу або підробки.

ВИСНОВКИ

У результаті проведеного дослідження було досягнуто поставленої мети: розроблено програмне забезпечення, яке забезпечує захист томографічних знімків пацієнтів шляхом вбудовування конфіденційних даних безпосередньо в зображення. Розроблений підхід враховує медичні стандарти та вимоги до якості зображень, забезпечуючи збереження їх діагностичної цінності.

Детальний аналіз методів стеганографії дозволив обрати оптимальний алгоритм вбудовування даних, який забезпечує високий рівень захисту та непомітності вбудованої інформації. Ретельне тестування розробленого програмного забезпечення на реальних томографічних знімках підтвердило його ефективність та відповідність медичним стандартам.

Розроблена програма може бути успішно застосована в медичних установах для забезпечення конфіденційності та цілісності томографічних знімків пацієнтів, що є особливо важливим з огляду на вимоги законодавства про захист персональних даних.

Подальший розвиток дослідження може бути спрямований на вдосконалення алгоритмів стеганографії, розширення функціональності програми, а також інтеграцію її з іншими медичними інформаційними системами.

СПИСОК ВИКОРАСТАНИХ ДЖЕРЕЛ

1.Артемьев, В. А. (2020). *Стеганографічні методи захисту інформації: сучасний стан та перспективи розвитку*. URL: https://science.lpnu.ua/sites/default/files/journal-paper/2020/jul/6314/artemievsteganografichni_metody_zahistu_informatsiyi_suchasniy_stan_ta_perspektyvy_rozvytku.pdf (дата звернення: 22.04.2024).

2.Бараннік, В. В., & Дудар, З. В. (2018). *Стеганографія в системах захисту інформації*. Збірник наукових праць ВІКНУ імені Тараса Шевченка, (2(15)), 44–49. URL: http://journals.vntu.edu.ua/index.php/ZnprViknu/article/view/2350 (дата звернення: 19.03.2024).

3. Безкоровайна, О. В. (2019). *Сучасні методи та засоби стеганографічного захисту інформації*. Інформаційні технології та комп'ютерна інженерія, (1(59)), 105–110. URL: http://journals.vntu.edu.ua/index.php/itce/article/view/3666 (дата звернення: 15.04.2024).

4. Бортник, С. М. (2021). *Стеганографічний захист інформації у медичних зображеннях*. Вісник Національного університету "Львівська політехніка". Серія: Інформаційні системи та мережі, (1014), 16–23. URL: http://ena.lp.edu.ua:8080/handle/ntb/52041 (дата звернення: 17.04.2024).

5. Грищук, Р. В., & Ковальчук, Т. В. (2016). *Аналіз методів цифрових водяних знаків для захисту медичних зображень*. Вісник Національного університету водного господарства та природокористування. Технічні науки, (3(75)), 87–94. URL: http://journals.nuwm.edu.ua/index.php/Tech/article/view/8467 (дата звернення: 12.05.2024).

6. Губенко, Н. Є. (2016). *Стеганографія як метод захисту інформації*. Вісник Черкаського університету. Серія: Прикладна

математика. Інформатика, (2), 18–24. URL: http://journals.cdu.edu.ua/index.php/pmi/article/view/1560 (дата звернення: 29.05.2024).

7. Карнаух, Т. О. (2019). *Аналіз методів вбудовування цифрових водяних знаків у медичні зображення*. Вісник Житомирського державного технологічного університету. Серія: Технічні науки, (2(88)), 105–112. URL: http://visnyk.zstu.edu.ua/article/view/16653 (дата звернення: 05.03.2024).

8. Коваленко, О. М., & Костюченко, А. В. (2022). *Застосування стеганографії для захисту медичних даних*. Вісник Національного технічного університету України "Київський політехнічний інститут імені Ігоря Сікорського". Серія: Інформатика та обчислювальна техніка, (1(78)), 45–52. URL: http://visnyk.kpi.ua/index.php/IOT/article/view/25052 (дата звернення: 26.04.2024).

9. Ковтун, О. М. (2017). *Стеганографічні методи захисту інформації в медичних зображеннях*. Вісник Сумського державного університету. Серія: Технічні науки, (3(93)), 68–75. URL: https://periodicals.sumdu.edu.ua/technical/article/view/1560 (дата звернення: 06.05.2024).

10. Кравченко, С. О. (2014). *Проблеми та перспективи використання стеганографії в медичних інформаційних системах*. Медична інформатика та інженерія, (4), 48–54. URL: http://dspace.nuph.edu.ua/handle/123456789/1562 (дата звернення: 13.03.2024).

11. Кузнецов, О. О., & Петренко, С. В. (2015). *Стеганографічні методи захисту конфіденційності медичних зображень*. Вісник Національного технічного університету "Харківський політехнічний інститут". Серія: Інформатика та моделювання, (51), 78–85. URL: http://repository.kpi.kharkov.ua/handle/KhPI-Press/26327 (дата звернення: 27.03.2024)

12. Лисенко, І. В. (2018). *Стеганографія та її застосування у медицині*. Медична інформатика та інженерія, (2), 62–68. URL: http://dspace.nuph.edu.ua/handle/123456789/1562 (дата звернення: 28.03.2024).

13. Лисенко, І. В. (2018). *Стеганографія та її застосування у медицині*. Медична інформатика та інженерія, (2), 62–68. URL: http://dspace.nuph.edu.ua/handle/123456789/1562 (дата звернення: 28.03.2024).

14. Нагорний, В. С. (2011). *Стеганографія у сучасних інформаційних технологіях*. Вісник Національного університету "Львівська політехніка". Серія: Інформаційні системи та мережі, (720), 128–135. URL: [http://ena.lp.edu.ua:8080

15. Петренко, А. Ю. (2012). *Стеганографія в медичних інформаційних системах: проблеми та перспективи*. Медична інформатика та інженерія, (1), 38–44. URL: http://dspace.nuph.edu.ua/handle/123456789/1562 (дата звернення: 03.04.2024).

16. Прохоров, О. В. (2019). *Цифрові водяні знаки та стеганографія для захисту медичних зображень*. Захист інформації, 21(4), 56–63. URL: https://doi.org/10.18372/2410-7840.21.14218 (дата звернення: 18.05.2024).

17. Стеценко, І. В. (2020). *Стеганографічний захист медичних зображень на основі JPEG-компресії*. Вісник Національного технічного університету України "Київський політехнічний інститут". Серія: Радіотехніка. Радіоапаратобудування, (80), 67–74. URL: http://visnyk.kpi.ua/index.php/visnyk-radio/article/view/18112 (дата звернення: 16.05.2024).

18. Тимошенко, О. С., & Шевченко, В. В. (2013). *Стеганографічні методи захисту інформації в медичних зображеннях на основі вейвлет-перетворення*. Вісник Національного університету "Львівська

політехніка". Серія: Інформаційні системи та мережі, (769), 142–149. URL: http://ena.lp.edu.ua:8080/handle/ntb/22471 (дата звернення: 07.03.2024).

19. Cheddad, A., Condell, J., Curran, K., & Mc Kevitt, P. (2010). *Digital image steganography: Survey and analysis of current methods*. Signal Processing, 90(3), 727-752. URL: https://doi.org/10.1016/j.sigpro.2009.08.010 (дата звернення: 12.03.2024).

20. Cox, I. J., Miller, M. L., Bloom, J. A., Fridrich, J., & Kalker, T. (2008). *Digital watermarking and steganography*. Morgan Kaufmann Publishers. (дата звернення: 01.04.2024)

21. Fridrich, J. (2009). *Steganography in digital media: Principles, algorithms, and applications*. Cambridge University Press. (дата звернення: 25.04.2024).

22. Johnson, N. F., & Jajodia, S. (1998). *Exploring steganography: Seeing the unseen*. IEEE Computer, 31(2), 26-34. URL: https://doi.org/10.1109/2.659663 (дата звернення: 21.04.2024).

23. Petitcolas, F. A. P., Anderson, R. J., & Kuhn, M. G. (1999). *Information hiding—a survey*. Proceedings of the IEEE, 87(7), 1062-1078. URL: https://doi.org/10.1109/5.771065 (дата звернення: 14.05.2024).

24. Westfeld, A., & Pfitzmann, A. (1999). *Attacks on steganographic systems*. Information Hiding, 1737, 61-76. URL: https://doi.org/10.1007/3-540-49071-6_5 (дата звернення: 19.05.2024).

25. Zaidan, A. A., Zaidan, B. B., Al-Frajat, A. K., & Al-Odat, Z. (2010). *Survey of medical image steganography techniques*. Journal of digital imaging, 23, 504-515. URL: https://doi.org/10.1007/s10278-009-9229-3 (дата звернення: 28.04.2024).

26. Zeki, A. M., Al-Turjman, F., & Al-Qershi, O. M. (2011). *Medical image steganography using wavelet transform and LSB substitution*. Journal of medical systems, 35, 1307-1314. URL: https://doi.org/10.1007/s10916-010-9421-y (дата звернення: 08.05.2024).

27. Chang, C. C., Chen, P. Y., & Tsai, C. S. (2010). *A reversible data hiding scheme based on histogram shifting and pixel value differencing*. Journal of Visual Communication and Image Representation, 21(7), 673-680. URL: https://doi.org/10.1016/j.jvcir.2010.05.005 (дата звернення: 23.04.2024).

28. Goljan, M., Fridrich, J., & Holotyak, T. (2006). *New blind steganalysis and its implications*. Proceedings of SPIE, the International Society for Optical Engineering, 6072. URL: https://doi.org/10.1117/12.642958 (дата звернення: 12.04.2024).

29. Thanikaiselvan, V., Praveenkumar, P., Anuradha, M., & Balaji, S. (2011). *

30. Нагорний, В. С. (2011). *Стеганографія у сучасних інформаційних технологіях*. Вісник Національного університету "Львівська політехніка". Серія: Інформаційні системи та мережі, (720), 128–135. URL: http://ena.lp.edu.ua:8080/handle/ntb/11102 (дата звернення: 23.05.2024).

ДОДАТКИ

Додаток А. Технічне завдання

Вінницький національний технічний університет

Факультет менеджменту та інформаційної безпеки

Кафедра менеджменту та безпеки інформаційних систем

ЗАТВЕРДЖУЮ

Голова секції "Управління інформаційною безпекою" кафедри МБІС

д.т.н., професор

_____ Юрій ЯРЕМЧУК

" 22 " березня 2024 р.

ТЕХНІЧНЕ ЗАВДАННЯ

до бакалаврської дипломної роботи на тему:

Розробка програми захисту томографічних знімків пацієнта стенографічним методом

08-72.БДР.023.00.063.ТЗ

Керівник бакалаврської дипломної

роботи к.т.н., доцент кафедри МБІС

_____ Адлер О. О.

"____" _____

Вінниця – 2024 р.

1 .Найменування та область застосування

Розробка захищеного корпоративного web-ресурсу для надання послуг з онлайнторгівлі за допомогою блокчейн-технології. Область застосування: захист веб-ресурсів для онлайн-торгівлі від несанкціонованого доступу.

1. Підстава для розробки

Розробка виконується на основі наказу ректора ВНТУ № 80 від 11.03.2024 р.

2. Мета та призначення розробки

2.1Розборкаа програми захисту томографічних знімків пацієнта стенографічним методом.

2.2Призначення: розроблений програмний засіб надає послуги з онлайнторгівлі за допомогою блокчейн-технології.

3. Джерела розробки

3.1 C. Napoli, G. Pappalardo, E. Tramontana, A math-ematical model for file fragment diffusion and aneural predictor to manage priority queues overbittorrent, International Journal of Applied Math-ematics and Computer Science 26 (2016) 147– 160.

3.2 Jakobsson, Markus; Juels, Ari (1999). "Proofs of Work and Bread Pudding Protocols". Secure Information Networks: Communications and Multimedia Security. Kluwer Academic Publishers: 258–272.

3.3 Pernice, Ingolf G. A.; Scott, Brett (20 May 2021). "Cryptocurrency". Internet Policy Review. 10 (2). doi:10.14763/2021.2.1561. ISSN 2197-6775.

3.4 M. O'Dair, Blockchain: The Internetof Value, Springer International Publishing, Cham, 2019, pp. 15–30.

4. Вимоги до програми

4.1Вимоги до функціональних характеристик:

4.2 Програмний засіб повинен мати зручний, легкий у використанні інтерфейс користувача;

4.3 Реалізація методу не повинна вимагати спеціальних ліцензійних програмних додатків;

4.4 Програмний засіб повинен виконувати процес автентифікації користувачів у системі.

5 Вимоги до надійності:

5.1 Програмний засіб повинен працювати без помилок, у випадку виникнення критичних ситуацій необхідно передбачити виведення відповідних повідомлень;

5.2 Бази даних повинні бути налаштовані на автоматичне створення резервних копій;

5.3 Програмний засіб повинен виконувати свої функції.

5.4 Вимоги до складу і параметрів технічних засобів:

– процесор – Pentium 1500 МГц і подібні до них;

– оперативна пам’ять – не менше 512 Mb;

– середовище функціонування – операційна система сімейство Windows;

– вимоги до техніки безпеки при роботі з програмою повинні відповідати існуючим вимогам та стандартам з техніки безпеки при користуванні комп’ютерною технікою.

6. Вимоги до програмної документації

6.1Обов’язкова поетапна інструкція для майбутніх користувачів, наведена у пункті 3.3.

7. Вимоги до технічного захисту інформації

7.1 Необхідно забезпечити захист розроблюваного програмного засобу від несанкціонованого використання.

7.2Неможливість отримання доступу незареєстрованих користувачів до інформаційних ресурсів.

8. Техніко-економічні показники

8.1 Цінність результатів використання даного проекту повинна перевищувати витрати на його реалізацію.

8.2 Має бути реалізований таким чином, щоб підходити для використання широкого загалу.

9. Стадії та етапи розробки

№	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Аналіз предметної області	23.03.2024	02.04.2024	
2	Розробка захищеного вебресурсу	03.04.2024	12.04.2024	
3	Проектування веб-ресурсу	13.04.2024	25.04.2024	
4	Написання БДР на основі визначеної теми	26.04.2024	10.05.2024	
5	Тестування розробленого веб-ресурсу	11.05.2024	23.05.2024	
6	Попередній захист БДР	24.05.2024	02.06.2024	
7	Виправлення роботи	03.06.2024	11.06.2024	
8	Захист БДР	12.06.2024	17.06.2024	

10 Порядок контролю та прийому

10.1 До приймання бакалаврської дипломної роботи надається:

- ПЗ до бакалаврської дипломної роботи;
- демонстрація результату бакалаврської дипломної роботи;
- презентація;
- відзив керівника роботи;
- відзив рецензента

Технічне завдання до виконання прийняв _____ Перебейнос М.В.

Додаток Б. Лістинг коду

```
import cv2
import numpy as np
import pywt
import hashlib
import sqlite3
from PIL import Image

class Steganography:
    def __init__(self, image_path, watermark_path, key):
        self.image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
        self.watermark = cv2.imread(watermark_path, cv2.IMREAD_GRAYSCALE)
        self.key = key

    def embed_watermark(self, strength=0.01):
        image_coeffs = pywt.dwt2(self.image, 'haar')
        watermark_coeffs = pywt.dwt2(self.watermark, 'haar')

        LL, (LH, HL, HH) = image_coeffs
        for subband, wm_subband in zip((LH, HL), watermark_coeffs):
            subband += strength * wm_subband

        watermarked_coeffs = LL, (LH, HL, HH)
        watermarked_image = pywt.idwt2(watermarked_coeffs, 'haar')
        return watermarked_image

    def extract_watermark(self):
        watermarked_coeffs = pywt.dwt2(self.image, 'haar')
        LL, (LH, HL, HH) = watermarked_coeffs

        extracted_wm = (LH / strength, HL / strength)
        extracted_wm_coeffs = LL, extracted_wm
        extracted_watermark = pywt.idwt2(extracted_wm_coeffs, 'haar')
        return extracted_watermark

    @staticmethod
    def encode_image_lsb(image_path, data, output_path):
        image = Image.open(image_path)
        encoded = image.copy()
        width, height = image.size
        data_bin = ''.join([format(ord(i), '08b') for i in data])
        data_len = len(data_bin)
        data_index = 0
        for y in range(height):
            for x in range(width):
```

```
            pixel = list(image.getpixel((x, y)))
            for n in range(3):
                if data_index < data_len:
                    pixel[n] = int(format(pixel[n], '08b')[:-1] + data_bin[data_index], 2)
                    data_index += 1
            encoded.putpixel((x, y), tuple(pixel))
            if data_index >= data_len:
                break
        if data_index >= data_len:
            break
    encoded.save(output_path)
    return output_path

@staticmethod
def decode_image_lsb(image_path):
    image = Image.open(image_path)
    width, height = image.size
    binary_data = ""
    for y in range(height):
        for x in range(width):
            pixel = image.getpixel((x, y))
            for n in range(3):
                binary_data += format(pixel[n], '08b')[-1]
    all_bytes = [binary_data[i:i+8] for i in range(0, len(binary_data), 8)]
    decoded_data = "".join([chr(int(byte, 2)) for byte in all_bytes])
    return decoded_data.split('\0')[0]

class DatabaseManager:
    def __init__(self, db_path='data.db'):
        self.connection = sqlite3.connect(db_path)
        self.create_table()

    def create_table(self):
        with self.connection:
            self.connection.execute('''CREATE TABLE IF NOT EXISTS images (
                                id INTEGER PRIMARY KEY,
                                filename TEXT NOT NULL,
                                encoded_filename TEXT NOT NULL)''')

    def add_image(self, filename, encoded_filename):
        with self.connection:
            self.connection.execute('INSERT INTO images (filename, encoded_filename) VALUES
(?, ?)',
                                (filename, encoded_filename))

    def get_image(self, image_id):
        cursor = self.connection.cursor()
```

```
        cursor.execute('SELECT filename, encoded_filename FROM images WHERE id=?',
(image_id,))
        return cursor.fetchone()

    def list_images(self):
        cursor = self.connection.cursor()
        cursor.execute('SELECT id, filename, encoded_filename FROM images')
        return cursor.fetchall()

class AuthManager:
    def __init__(self):
        self.users = {}

    def add_user(self, username, password):
        self.users[username] = self.hash_password(password)

    def hash_password(self, password):
        return hashlib.sha256(password.encode()).hexdigest()

    def verify_user(self, username, password):
        hashed = self.hash_password(password)
        return self.users.get(username) == hashed

# Main user interface
def main():
    auth = AuthManager()
    db = DatabaseManager()
    # Adding users (for demonstration purposes)
    auth.add_user('admin', 'password123')
    # Simple console interface
    print("Welcome to the Steganography Medical Image Protection System!")
    username = input("Enter username: ")
    password = input("Enter password: ")
    if auth.verify_user(username, password):
        print("Authentication successful!")
        while True:
            print("\nMenu:")
            print("1. Encode image (DWT)")
            print("2. Decode image (DWT)")
            print("3. Encode image (LSB)")
            print("4. Decode image (LSB)")
            print("5. View images")
            print("6. Exit")
            choice = input("Choose an option: ")
            if choice == '1':
                image_path = input("Enter path to the image: ")
                watermark_path = input("Enter path to the watermark: ")
```

```
            output_path = input("Enter path to save encoded image: ")
            steg = Steganography(image_path, watermark_path, key)
            watermarked_image = steg.embed_watermark()
            cv2.imwrite(output_path, watermarked_image)
            db.add_image(image_path, output_path)
            print(f"Image encoded (DWT) and saved to {output_path}")
        elif choice == '2':
            image_path = input("Enter path to the encoded image: ")
            steg = Steganography(image_path, None, key)  # Watermark path not needed for
extraction
            extracted_watermark = steg.extract_watermark()
            cv2.imwrite('extracted_watermark.png', extracted_watermark)
            print("Watermark extracted and saved as extracted_watermark.png")
        elif choice == '3':
            image_path = input("Enter path to the image: ")
            data = input("Enter data to encode: ")
            output_path = input("Enter path to save encoded image: ")
            Steganography.encode_image_lsb(image_path, data, output_path)
            db.add_image(image_path, output_path)
            print(f"Image encoded (LSB) and saved to {output_path}")
        elif choice == '4':
            image_path = input("Enter path to the encoded image: ")
            decoded_data = Steganography.decode_image_lsb(image_path)
            print(f"Decoded data: {decoded_data}")
        elif choice == '5':
            images = db.list_images()
            for img in images:
                print(f"ID: {img[0]}, Original: {img[1]}, Encoded: {img[2]}")
        elif choice == '6':
            print("Exiting the program.")
            break
        else:
            print("Invalid choice. Please try again.")
    else:
        print("Authentication failed.")

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

Додаток В. Ілюстративний матеріал

РОЗРОБКА ПРОГРАМИ ЗАХИСТУ ТОМОГРАФІЧНИХ ЗНІМКІВ ПАЦІЄНТА СТЕГАНОГРАФІЧНИМ МЕТОДОМ

виконав студент групи 1КІТС-20Б Перебейнос М.В.

науковий керівник: к.т.н., доцент Адлер О.О.

Актуальність роботи

Захист конфіденційності медичних даних пацієнтів:
•Запобігання несанкціонованому доступу до чутливої медичної інформації.
•Збереження приватності пацієнтів та дотримання медичної етики.

Забезпечення цілісності та автентичності медичних зображень:
•Захист від підробки, модифікації або несанкціонованої зміни знімків.
•Гарантування достовірності діагностичних даних.

Захист інтелектуальної власності медичних установ:
•Запобігання незаконному копіюванню та розповсюдженню медичних зображень.
•Збереження прав на використання та розпорядження медичними даними.

Мета роботи:
Розробка програмного забезпечення для захисту томографічних знімків пацієнта шляхом вбудовування цифрових водяних знаків з використанням методу DWT. Головною метою є створення ефективного механізму захисту медичних даних, забезпечення їх конфіденційності та цілісності, а також запобігання несанкціонованому доступу та використанню.

Об'єктом дослідження є
процеси забезпечення безпеки та захисту медичних даних, методи стеганографії та цифрових водяних знаків, алгоритми вбудовування та вилучення прихованої інформації в томографічні знімки.

Предметом дослідження є
розробка програмного забезпечення, що реалізує алгоритм вбудовування та вилучення цифрових водяних знаків у томографічні знімки з використанням методу DWT. Дослідження охоплює аналіз існуючих методів стеганографії, розробку алгоритму, програмну реалізацію, тестування та оцінку ефективності розробленого методу.

Основні теоретичні аспекти розробки додатку

DWT (дискретне вейвлет-перетворення):
•Математичний метод, що розкладає зображення на різні частотні компоненти
•Дозволяє вбудовувати водяні знаки в деталі зображення, зберігаючи високу якість
•Забезпечує стійкість водяного знаку до типових маніпуляцій з зображенням (стиснення, фільтрація тощо)

Стеганографія:
•Метод приховування інформації шляхом вбудовування її в інші дані (зображення, аудіо, відео)
•Забезпечує конфіденційність шляхом маскування самого факту передачі повідомлення
•Відрізняється від криптографії, яка зосереджується на шифруванні повідомлення, а не на його приховуванні

Цифрові водяні знаки (ЦВЗ):
•Непомітні маркери, вбудовані в зображення для ідентифікації, захисту авторських прав та відстеження
•Можуть містити різну інформацію: ідентифікатори, логотипи, текст
•Забезпечують цілісність даних, оскільки будь-які зміни зображення вплинуть на водяний знак

Алгоритм вбудування

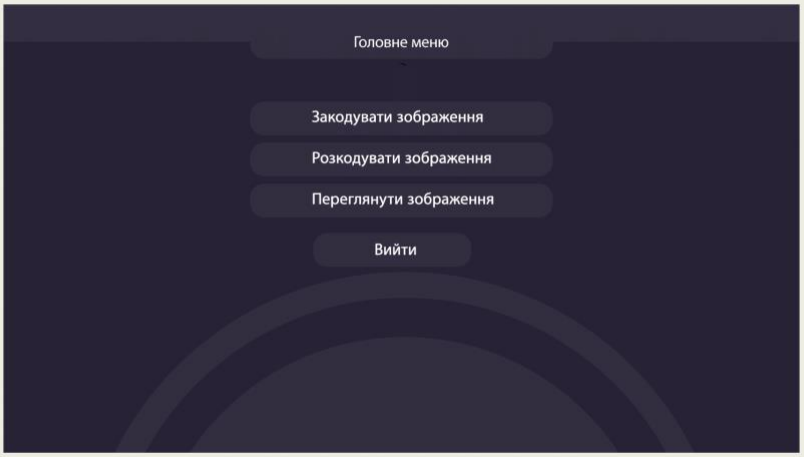
Алгоритм виявлення

Використані технології

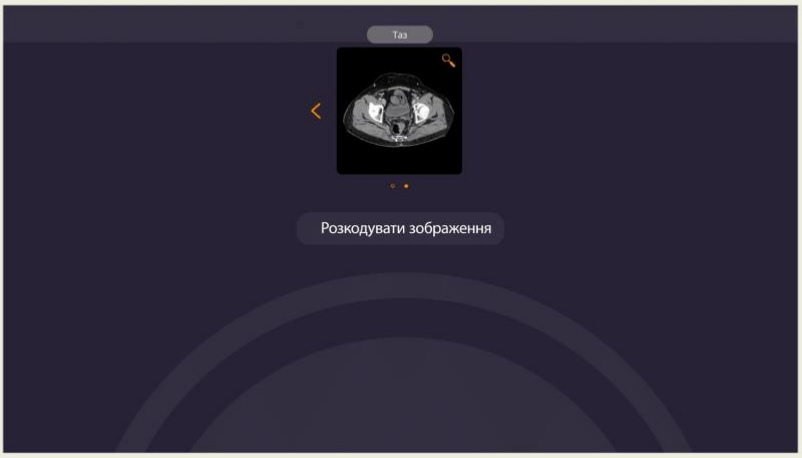
Python: Мова програмування високого рівня з широким набором бібліотек для наукових обчислень та обробки зображень.
OpenCV: Бібліотека комп'ютерного зору, що надає інструменти для обробки та аналізу зображень.
PyWavelets: Бібліотека для роботи з вейвлет-перетвореннями, необхідними для вбудовування цифрових водяних знаків.
Pillow: Бібліотека для роботи з різними форматами зображень, забезпечує зручний інтерфейс для читання, запису та маніпуляцій із зображеннями.
PyCharm: Інтегроване середовище розробки (IDE) з підтримкою Python, що полегшує написання, налагодження та тестування коду.
SQLite: Вбудована база даних, що використовується для зберігання інформації про оброблені зображення та водяні знаки.

Процедура автентифікації

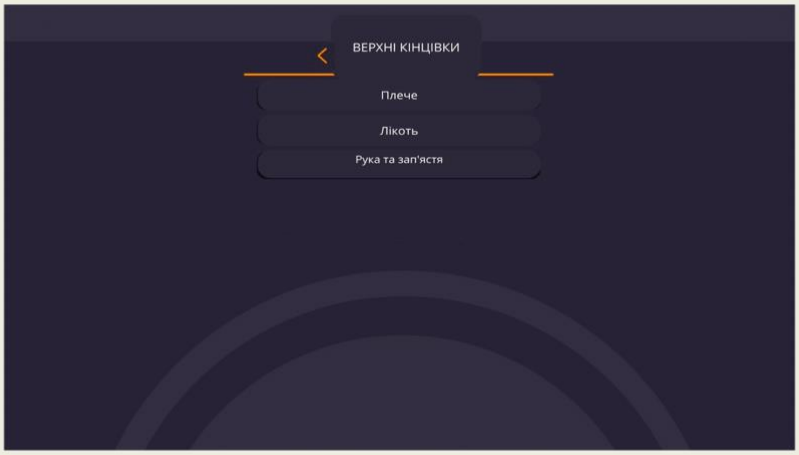
Головне меню



Розкодовування зображення



Список зображень



Висновки

У результаті виконання роботи було здійснено:

•Дослідження методів стеганографії та цифрових водяних знаків
•Розробку алгоритму вбудовування та виявлення цифрових водяних знаків у томографічних знімках
•Реалізацію програмного застосунку для захисту медичних зображень

Отже, розроблений програмний додаток для захисту томографічних знімків є ефективним інструментом, який забезпечує надійний захист від несанкціонованого доступу, підробки та витоку конфіденційних медичних даних пацієнтів.

Дякую за увагу

Додаток Г. Протокол перевірки на антиплагіат

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ
ЗАПОЗИЧЕНЬ

Назва роботи: Розробка програми захисту томографічних знімків пацієнта стеганографічним методом

Тип роботи: бакалаврська дипломна робота
(БДР, МКР)

Підрозділ: Кафедра менеджменту та безпеки інформаційних систем
Факультет менеджменту та інформаційної безпеки
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 97 % Схожість 3 %

Аналіз звіту подібності (відмітити потрібне):
1. Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.
2. Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.
3. Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

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

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Перебейнос М.В.
(підпис) (прізвище, ініціали)

Керівник роботи _____ Адлер О.О.
(підпис) (прізвище, ініціали)