Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет інформаційних технологій та комп'ютерної інженерії
(повне найменування інституту, назва факультету (відділення))

Кафедра програмного забезпечення
(повна назва кафедри (предметної, циклової комісії))

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА
на тему:

«Методи та програмні засоби захисту біомедичних зображень від несанкціонованого доступу»

Виконав: студент 2 курсу, групи 1ПІ-23м спеціальності 121 – Інженерії програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Грицишин В. О.
(прізвище та ініціали)

Керівник: к.т.н., доцент кафедри ПЗ
Майданюк В. П.
(прізвище та ініціали)
«05» 12 2024 р.

Рецензент: к.т.н., доцент кафедри ОТ
Богомолов С. В.
(прізвище та ініціали)
«05» 12 2024 р.

Допущено до захисту
Зав. кафедри ПЗ Романюк О. Н.
«05» 12 2024р.

Вінниця ВНТУ – 2024 рік

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти другий (магістерський)
Галузь знань 12 – Інформаційні технології
Спеціальність 121 – Інженерія програмного забезпечення
Освітньо-професійна програма – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О.Н.
«18» вересня 2024 р.

ЗАВДАННЯ
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Грицишину Василю Олександровичу

1. Тема роботи – «Методи та програмні засоби захисту біомедичних зображень від несанкціонованого доступу»
Керівник роботи: Майданюк Володимир Павлович, к. т. н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від «17» вересня 2024 р. № 310.
2. Термін подання студентом роботи : 3 грудня 2024 р.
3. Вихідні дані: метод захисту – стеганографія; тип контейнера – зображення; тип перетворення - Уолша-Адамара; операційна система –Windows; середовище розробки – MS Visual Studio; мова програмування – C/C++.
4.Зміст текстової частини: вступ; аналіз методів захисту біомедичних даних; розробка методів і алгоритмів для захисту біомедичних зображень; розробка програмного забезпечення для захисту біомедичних зображень; тестування додатку; економічна частина; висновки; додатки.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): мета і задачі роботи; наукова новизна та практична цінність; загальна схема захисту біомедичних зображень від несанкціонованого доступу; блок-схема алгоритму шифрування зображень та приховування молодших біт пікселя; блок-схема алгоритму приховування в просторовій області; перетворення Уолша-Адамара; блок-схема алгоритму приховування в коефіцієнтах перетворення Уолша-Адамара; оцінка складності стеганографічного методу; головне вікно програми; інші вікна; зображення після криптографічного захисту; приховування молодших біт зображення; вбудовування в просторовій області; вбудовування в частотній області; основні результати роботи.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	завданн прийня
1–4	Майданюк В. П., доц. каф. ПЗ, к.т.н	19.09.2024	02.12.202
5	Причепа І. В., доц. каф. ЕПВМ, к.е.н	17.11.2024	25.11.202

7. Дата видачі завдання 19 вересня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Обґрунтування вибору методу розробки та постановка задач	20.09.24 – 02.10.24	вик.
2	Розробка архітектури та алгоритмів програмного продукту	03.10.24 – 23.10.24	вик.
3	Аналіз і вибір мови програмування та середовища розробки	16.10.24 – 23.10.24	вик.
4	Розробка програмного продукту	24.10.24 – 13.11.24	вик.
5	Тестування програми	14.11.24 – 21.11.24	вик.
6	Розробка економічної частини	17.11.24 –25.11.24	вик.
7	Оформлення матеріалів до захисту МКР	22.11.24 – 30.11.24	вик.

Студент _____ **Грицишин В. О.**
(підпис) (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи _____ **Майданюк В. П.**
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

УДК 004.932

Грицишин В. О. Методи та програмні засоби захисту біомедичних зображень від несанкціонованого доступу. Магістерська кваліфікаційна робота зі спеціальності 121 – Інженерія програмного забезпечення, освітня програма – Інженерія програмного забезпечення. Вінниця: ВНТУ, 2024. - 112 с.

На укр. мові. Бібліогр.: 31 назва; рис.: 23; табл.: 13.

Метою магістерської кваліфікаційної роботи є зменшення обчислювальних витрат для захисту рентгенівських зображень від несанкціонованого доступу за рахунок застосування стеганографічних перетворень.

У роботі проведено детальний аналіз методів і засобів захисту біомедичних зображень, який показав актуальність теми роботи, зокрема, у зв'язку з розвитком віддаленої радіології. Подальшого розвитку отримав метод стеганографічного захисту рентгенівських зображень, у якому на відміну від існуючих, в якості контейнера використано трансформанти перетворення Уолша-Адамара, отримані з рентгенівського зображення, що дозволило збільшити кількість молодших біт, які використовуються для приховування, в кожному пікселі зображення-контейнера до 4 по кожній складовій кольору і зменшити складність захисту у два рази.

З використанням середовища програмування Microsoft Visual Studio мовою програмування C++ розроблено програмне забезпечення для захисту біомедичних зображень, яке виконує захист рентгенівських зображень з використанням стеганографічних методів.

Ключові слова: Windows, Visual Studio, C/C++, стеганографія, перетворення Уолша-Адамара, трансформанти, програма.

ABSTRACT

Hrytsyshyn V. O. Methods and software for protecting biomedical images from unauthorized access. Master's thesis on the specialty 121 - Software engineering, educational program - Software engineering. Vinnytsia: VNTU, 2024. - 112 p.
In Ukrainian language. Bibliographer: 31 titles; fig.: 23; tabl.: 13.

The purpose of the master's qualification work is to reduce the computational costs for protecting X-ray images from unauthorized access by using steganographic transformations.

The work provides a detailed analysis of methods and means of protecting biomedical images, which showed the relevance of the topic of the work, in particular, in connection with the development of remote radiology. The method of steganographic protection of X-ray images was further developed, in which, unlike existing ones, the transforms of the Walsh-Hadamard transformation obtained from the X-ray image were used as a container, which made it possible to increase the number of low-order bits used for concealment in each pixel of the container image to 4 for each color component and reduce the complexity of protection by half.

Using the Microsoft Visual Studio programming environment in the C++ programming language, software for protecting biomedical images was developed, which performs protection of X-ray images using steganographic methods.

Keywords: Windows, Visual Studio, C/C++, steganography, Walsh-Hadamard transform, transformants, program.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. Всі методи захисту біомедичних зображень від несанкціонованого доступу можна розділити на дві групи [1-7]:

- захист файлів зображень виконується як захист будь-яких файлів;
- методи що враховують специфіку зображень.

До методів захисту першої групи можна віднести такі [1-3]:

1. Шифрування даних: Шифрування біомедичних зображень перед їх зберіганням або передачею є одним з найефективніших способів захисту. Використання алгоритмів шифрування, таких як AES (Advanced Encryption Standard), забезпечує високий рівень безпеки.

2. Контроль доступу: Впровадження систем контролю доступу, які обмежують доступ до біомедичних зображень лише авторизованим користувачам. Це може включати використання паролів, біометричної аутентифікації або багатофакторної аутентифікації1.

3. Аудит та моніторинг: Регулярний аудит та моніторинг доступу до біомедичних зображень допомагає виявляти та запобігати несанкціонованому доступу. Це включає ведення журналів доступу та аналіз підозрілої активності1.

4. Використання VPN: Віртуальні приватні мережі (VPN) забезпечують безпечний канал для передачі даних через Інтернет, що допомагає захистити біомедичні зображення від перехоплення під час передачі.

5. Захист хмарних сховищ: Якщо біомедичні зображення зберігаються в хмарі, важливо використовувати хмарні сервіси з вбудованими функціями шифрування та контролю доступу.

Існують специфічні методи захисту біомедичних зображень, які враховують особливості цих даних. Деякі з них такі [4]:

1. Водяні знаки: Вбудовування водяних знаків у біомедичні зображення допомагає ідентифікувати джерело зображення та виявляти несанкціоноване копіювання або модифікацію.

2. Стиснення з шифруванням: Використання методів стиснення даних разом із шифруванням дозволяє зменшити обсяг даних для зберігання та передачі, одночасно забезпечуючи їх захист.

3. Анонімізація даних: Видалення або маскування особистої інформації з біомедичних зображень перед їх обробкою або передачею допомагає захистити конфіденційність пацієнтів.

4. Блокчейн: Використання блокчейн-технологій для зберігання та передачі біомедичних зображень забезпечує високий рівень безпеки та прозорості, оскільки кожна транзакція записується у незмінний реєстр3.

5. Машинне навчання для виявлення аномалій: Використання алгоритмів машинного навчання для аналізу доступу до біомедичних зображень та виявлення підозрілої активності може допомогти запобігти несанкціонованому доступу.

Однак, поки що недостатня увага приділяється стеганографічним методам захисту біомедичних зображень. Стеганографія (пер. з грец, «тайнопис») — це наука про приховану передачу інформації за допомогою збереження в таємниці самого факту передачі [8]. Для стеганографії важливим є вибір контейнера для приховування повідомлення. Найбільшу місткість забезпечують контейнери у вигляді файлів зображень, у яких можна замінити в кожному пікселі по крайній мірі 1 молодший біт по кожній складовій кольору на біт повідомлення, тобто у пікселі можна приховати мінімум 3 біти інформації. . Відомо два методи приховування повідомлень в зображеннях:

- приховування в просторовій області, тобто безпосередньо в пікселях зображення;

- приховування в частотній області, тобто в коефіцієнтах деякого перетворення, які отримують із зображення. Наприклад, перетворення Фур’є, Уолша-Адамара, дискретне косинусне перетворення або інше. Приховування в

частотній області вимагає значно більше обчислень, але забезпечує збільшення ємності контейнера [9-11].

Рентгенівські знімки це зображення з вузьким динамічним діапазоном, з наявністю ділянок з плавною зміною яскравості, тому слід очікувати, що кількість прихованих біт в кожному пікселі можна збільшити без втрати візуальної якості зображення і виявлення факту приховування іншого зображення в зображенні-контейнері.

Ці методи можуть бути ефективно використані для захисту біомедичних зображень від несанкціонованого доступу та забезпечення конфіденційності пацієнтів і саме ці питання розглядаються в роботі, тому її тема є актуальною.

Зв'язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

Мета та завдання дослідження. Метою магістерської кваліфікаційної роботи є зменшення обчислювальних витрат для захисту рентгенівських зображень від несанкціонованого доступу за рахунок застосування стеганографічних перетворень.

Основними задачами дослідження є:

- обґрунтування доцільності розробки нового технічного рішення;
- розробка та модифікація методів приховування рентгенівських зображень;
- розробка алгоритмів і програм для дослідження приховування рентгенівських зображень;
- експериментальні дослідження ефективності стеганографічних перетворень;
- розрахунок економічних показників розробки.

Об'єкт дослідження – процес приховування рентгенівських зображень в зображеннях того ж типу.

Предмет дослідження – методи та програмні засоби приховування рентгенівських зображень в зображеннях того ж типу.

Методи дослідження. У процесі досліджень використовувались: теорія алгоритмів, теорія імовірності та математична статистика, дискретна математика, теорія цифрової обробки сигналів та зображень, методи стеганографії для розробки моделей та методів приховування рентгенівських зображень; комп'ютерне моделювання для аналізу та перевірки отриманих теоретичних положень.

Наукова новизна отриманих результатів.

1. Подальшого розвитку отримав метод стеганографічного захисту рентгенівських зображень, у якому на відміну від існуючих, в якості контейнера використано трансформанти перетворення Уолша-Адамара, отримані з рентгенівського зображення, що дозволило збільшити кількість молодших біт, які використовуються для приховування, в кожному пікселі зображення-контейнера до 4 по кожній складовій кольору і зменшити складність реалізації алгоритму захисту у два рази.

2. Подальшого розвитку отримав метод шифрування даних при стеганографічному захисті рентгенівських зображень, у якому на відміну від існуючих, використано множину генераторів псевдовипадкових чисел для розсіювання даних у площині зображення та їх гамування, що дозволило що дозволило збільшити криптостійкість стеганографічного захисту в 2 рази.

Практична цінність отриманих результатів. Практична цінність одержаних результатів полягає в тому, що на основі отриманих в магістерській кваліфікаційній роботі теоретичних положень запропоновано алгоритми та розроблено програмні засоби для дослідження захисту рентгенівських зображень від несанкціонованого доступу.

Особистий внесок здобувача. Усі результати, викладені магістерській кваліфікаційній роботі, отримані автором особисто. У друкованих працях, опублікованих у співавторстві, автору належать такі результати: [9] - порівняльний аналіз перетворення Уолша-Адамара; [10] - визначення

параметрів генератора псевдовипадкових чисел; [11] - обгрунтування вибору стеганографічного контейнера.

Апробація матеріалів бакалаврської дипломної роботи. Основні положення магістерської кваліфікаційної роботи доповідалися та обговорювалися на науково-технічній конференції підрозділів Вінницького національного технічного університету (НТКП ВНТУ) (Вінниця, 2023 р.); XVII Міжнародної науково-практичної конференції «Інформаційні технології і автоматизація – 2024» (Одеса, 2024); Міжнародної науково-практичної Інтернет-конференції «Електронні інформаційні ресурси: створення, використання, доступ» (Суми/Вінниця, 2024).

Публікації. Основні результати досліджень опубліковано в 3-х наукових працях - у матеріалах конференцій.

1 АНАЛІЗ МЕТОДІВ ЗАХИСТУ БІОМЕДИЧНИХ ДАНИХ

1.1 Основні задачі захисту біомедичних даних

Віддалена радіологія стала невід'ємною частиною сучасної медичної практики, забезпечуючи швидкий доступ до діагностичних зображень та висновків, незалежно від географічного положення пацієнта або лікаря. Однак, разом із розвитком технологій з'являються нові виклики, пов'язані з безпекою та конфіденційністю медичної інформації [1].

Основною загрозою є можливість несанкціонованого доступу до конфіденційної інформації. Дані медичних обстежень, особливо радіологічні зображення, можуть містити чутливу інформацію про стан здоров'я пацієнта, що робить їх цінною мішенню для кіберзлочинців. Несанкціонований доступ до таких даних може призвести до їх неправомірного використання, наприклад, для шантажу, незаконного продажу або зміни діагнозу.

Забезпечити захист даних у віддаленій радіології в Україні покликана КСЗІ – комплексна система захисту інформації. Вона включає технічні та нетехнічні засоби, що дозволяють запобігти чи ускладнюють доступ до медичних даних.

Для захисту конфіденційності інформації у віддаленій радіології використовуються різні методи шифрування даних. Наприклад, стандарт шифрування AES-256 широко застосовується для захисту переданих зображень і текстових звітів. Крім того, важливим аспектом є автентифікація користувачів, яка може включати багатофакторну автентифікацію (MFA). Це знижує ризик несанкціонованого доступу навіть у випадку компрометації пароля.

Система охорони здоров'я зобов'язана дотримуватися численних норм і стандартів, таких як Загальний регламент щодо захисту даних (GDPR) у Європейському Союзі та Закон про переносимість і підзвітність медичного страхування (HIPAA) у США. Ці закони накладають жорсткі вимоги щодо захисту персональних даних, включаючи радіологічні зображення. Порушення

цих норм може призвести до серйозних юридичних наслідків і значних штрафів для медичних установ.

Ще один важливий аспект захисту конфіденційності у віддаленій радіології — це анонімізація даних. Перед відправленням радіологічних зображень і звітів на віддалену обробку або консультацію, особисті дані пацієнта, такі як ім’я, дата народження або номер медичної картки, можуть бути видалені або замінені кодами. Це допомагає знизити ризики, пов’язані з ідентифікацією пацієнтів.

Збереження даних — ще один важливий аспект безпеки у віддаленій радіології. Дані повинні бути надійно захищені як під час передачі, так і під час зберігання. Використання захищених серверів, регулярне резервне копіювання та моніторинг доступу до даних є необхідними заходами для забезпечення їхньої безпеки.

Забезпечення безпеки та конфіденційності у віддаленій радіології є критично важливим завданням, що вимагає комплексного підходу. Використання сучасних методів шифрування, автентифікації та анонімізації, а також дотримання законодавчих норм, допомагають знизити ризики та захистити конфіденційну медичну інформацію від загроз. У майбутньому розвиток технологій може надати ще більш ефективні інструменти для захисту даних у сфері телемедицини.

Актуальним є питання захисту конфіденційних даних і при користуванні цифровими сервісами для медичних установ. Зокрема, при роботі з платформою RadioLance для пошуку фахівців-радіологів впроваджуються всі необхідні заходи для забезпечення конфіденційності даних. Це гарантує інформаційну безпеку для пацієнта без ризиків для радіолога.

Сервіс RadioLance забезпечує безпеку та конфіденційність у роботі з медичними даними:

- Всі дані, включно із зображеннями і результатами розшифровки, зашифровані як у стані спокою, так і під час передавання.

- Всі дані зберігаються у дата-центрі, який має сертифікат комплексної системи захисту інформації (КСЗІ) від Державної служби спеціального зв'язку та захисту інформації України.
- Архітектура платформи забезпечує стандартизований і безпечний обмін медичними даними, покращує сумісність між різними системами та підвищує загальну безпеку обробки медичних зображень.
- Медичні висновки скріплюються цифровими підписами лікарів, що дають змогу упевнитися в автентичності даних, підтверджуючи, що дані надійшли від довіреного джерела [1].

1.2 Огляд методів захисту біомедичних зображень

Всі методи захисту біомедичних зображень від несанкціонованого доступу можна розділити на дві групи [1-7]:

- захист файлів зображень виконується як захист будь-яких файлів;
- методи що враховують специфіку зображень.

До методів захисту першої групи можна віднести такі [1-3]:

1. Шифрування даних: Шифрування біомедичних зображень перед їх зберіганням або передачею є одним з найефективніших способів захисту. Використання алгоритмів шифрування, таких як AES (Advanced Encryption Standard), забезпечує високий рівень безпеки.

2. Контроль доступу: Впровадження систем контролю доступу, які обмежують доступ до біомедичних зображень лише авторизованим користувачам. Це може включати використання паролів, біометричної автентифікації або багатофакторної автентифікації1.

3. Аудит та моніторинг: Регулярний аудит та моніторинг доступу до біомедичних зображень допомагає виявляти та запобігати несанкціонованому доступу. Це включає ведення журналів доступу та аналіз підозрілої активності1.

4. Використання VPN: Віртуальні приватні мережі (VPN) забезпечують безпечний канал для передачі даних через Інтернет, що допомагає захистити біомедичні зображення від перехоплення під час передачі.

5. Захист хмарних сховищ: Якщо біомедичні зображення зберігаються в хмарі, важливо використовувати хмарні сервіси з вбудованими функціями шифрування та контролю доступу.

Існують специфічні методи захисту біомедичних зображень, які враховують особливості цих даних. Деякі з них такі [4]:

1. Водяні знаки: Вбудовування водяних знаків у біомедичні зображення допомагає ідентифікувати джерело зображення та виявляти несанкціоноване копіювання або модифікацію.

2. Стиснення з шифруванням: Використання методів стиснення даних разом із шифруванням дозволяє зменшити обсяг даних для зберігання та передачі, одночасно забезпечуючи їх захист.

3. Анонімізація даних: Видалення або маскування особистої інформації з біомедичних зображень перед їх обробкою або передачею допомагає захистити конфіденційність пацієнтів.

4. Блокчейн: Використання блокчейн-технологій для зберігання та передачі біомедичних зображень забезпечує високий рівень безпеки та прозорості, оскільки кожна транзакція записується у незмінний реєстр3.

5. Машинне навчання для виявлення аномалій: Використання алгоритмів машинного навчання для аналізу доступу до біомедичних зображень та виявлення підозрілої активності може допомогти запобігти несанкціонованому доступу.

Ці методи можуть бути ефективно використані для захисту біомедичних зображень від несанкціонованого доступу та забезпечення конфіденційності пацієнтів.

1.3 Аналіз криптографічних методів і вибір базового

У криптографії використовуються дві прості форми шифрування, що називаються заміною і перестановкою. Жодна з них не забезпечує високого ступеня надійності, але вони можуть бути покладені в основу розробки більш складних шифрів, у яких при сполученні обох форм надійність коду підвищується.

Шифри заміни заміняють елементи відкритих даних на інші елементи за визначеним правилом. Шифри заміни поділяються на моноалфавітні і поліалфавітні.

У моноалфавітних шифрах замінний символ початкового тексту заміняється на інший, заздалегідь визначений. Наприклад, у коді Цезаря буква заміняється на іншу, віддалену від неї в латинському алфавіті на деяке число позицій [12].

Приклад:

A B C D E

B C D E F

Такий шифр дешифрується зовсім просто. Потрібно підрахувати, як часто зустрічаються букви в зашифрованому тексті, і зіставити результат з відомою для кожної мови частотою букв.

У поліалфавітних перестановках для заміни деякого символу початкового повідомлення в кожнім випадку його появи послідовно використовуються різні символи з деякого набору. Заміну можна організувати так, що кожному вхідному символові можна поставити у відповідність кожний з інших 255 символів таким чином, що загальне число перестановок буде дорівнювати $(2^8-1)!=255!$. Використовуючи криптографічний ключ, можна створити нескладну процедуру генерації однієї з цих замін. Жодна з замін не є ефективною, але як компонент більш складного алгоритму вона може бути корисною.

Шифри перестановок переставляють елементи відкритих даних (біти, букви, символи) у деякому новому порядку. Розрізняють шифри

горизонтальної, вертикальної, подвійної перестановки, ґрати й інші. У класичній криптографії метод перестановок застосовується до текстів природною мовою [8].

Приклад:

A B C D E | K L M N O | A B K P G | T F C D O

C A D E B | M K N O L | K A P G L | C T D O F

До рядка символів застосована перестановка в групі з 5 символів. Сучасна технологія звичайно використовує підстановку в рамках групи двійкових кодів, а не символів. Число перестановок у групі з 8 біт дорівнює 8!, що значно менше числа можливих замін, рівних 255!. Розумніше використовувати криптографічний ключ для модифікації замін, а не перестановок. Це підсилює практичну значимість перестановок, однак обидва методи кодування взаємно доповнюють один одного. Через експонентний ріст розмірів таблиць замін, можливості їхнього практичного застосування обмежені, останні можна компенсувати, використовуючи біти перестановки.

Принцип шифрування полягає в генерації гами-шифру (гамування) за допомогою датчика псевдовипадкових чисел (ПВЧ) і накладенні отриманої гами на відкриті дані (наприклад, при використанні логічної операції «виключне або»). Процес дешифрування даних зводиться до повторної генерації гами шифру при відомому ключі і накладенні такої гами на зашифровані дані. Отриманий зашифрований текст є досить важким для розкриття в тому випадку, якщо гама шифру не містить повторюваних бітових послідовностей. По суті справи, гама шифру повинна змінюватися випадковим чином для кожного слова, що шифрується. Фактично, якщо період гами перевищує довжину всього зашифрованого тексту і невідома ніяка частина початкового тексту, то шифр можна розкрити тільки прямим перебором (підбором ключа). У цьому випадку криптостійкість визначається розміром ключа.

Щоб одержати лінійні послідовності елементів гами, довжина яких перевищує розмір даних, що шифруються, використовуються датчики ПВЧ. На основі теорії груп було розроблено кілька типів таких датчиків.

В даний час найбільш доступні й ефективними є конгруентні генератори ПВЧ. Для цього класу генераторів ПВЧ можна зробити математично строгий висновок про те, якими властивостями володіють вихідні сигнали цих генераторів з погляду періодичності.

Одним з конгруентних генераторів є лінійний конгруентний датчик ПВЧ. Він формує послідовності псевдовипадкових чисел T(i) у відповідності з співвідношенням [12]:

$$T(i+1) = (A*T(i)+C) \bmod M \qquad (2.4)$$

де T(0) – початкова величина, обрана як породжувальне число;

A і C – константи. Такий датчик ПВЧ генерує псевдовипадкові числа з визначеним періодом повторення, що залежить від обраних значень A и C. Значення M звичайне встановлюється рівним 2^n, де n – довжина слова ЕОМ у бітах [8]. Фактично ж, якщо період гамми перевищує довжину всього зашифрованого тексту і невідома ніяка частина початкового тексту, то шифр можна розкрити тільки прямим перебором (пробій на ключ).

Одним з найбільш розповсюджених криптографічних стандартів для шифрування даних є стандарт США – DES [13-14]. Спочатку метод, що лежить в основі даного стандарту був розроблений фірмою IBM. Він був перевірений Агентством Національної Безпеки США, що не знайшло в ньому статистичних або математичних вад. Це означає, що дешифрування даних, захищених за допомогою DES, не може бути виконано статистичними (наприклад, за допомогою частотного словника) або математичними («прокручуванням» у зворотному напрямку) методами.

Після цей метод фірми IBM був прийнятий як федеральний стандарт [13]. Стандарт DES ґрунтується на дешифруванні (шифровці) блоків даних довжиною 64 біта під керуванням 64-бітового ключа.

Розшифровка виконується з використанням того ж ключа, що і при шифруванні, але зі зворотним порядком адресації бітів ключа, так що процес розшифровці зворотний процесові розшифровки [13].

Новим стандартом з більшою довжиною ключа став Advanced Encryption Standard (AES), також відомий під назвою Rijndael — симетричний алгоритм блочного шифрування (розмір блока 128 біт, ключ 128/192/256 біт). Державний інститут стандартів і технологій (англ. National Institute of Standards and Technology, NIST) США опублікував попередню специфікацію AES 26 жовтня 2001 року, після п'ятирічної підготовки. 26 травня 2002 року AES оголошено стандартом шифрування. AES є одним з найпоширеніших алгоритмів симетричного шифрування [15-17].

Через фіксований розмір блоку AES оперує із масивом 4 × 4 байт, що називається станом (версії алгоритму із більшим розміром блоку мають додаткові колонки).

Слабким місцем симетричної криптографічної системи при практичній реалізації є проблема розподілу ключів [18]. Для того, щоб був можливий обмін конфіденційною інформацією між двома суб'єктами інформаційної системи (ІС), ключ повинен згенерувати один з них, а потім якимсь чином знову ж таки в конфіденційному порядку передати іншому. Тобто в загальному випадку для передачі ключа знову ж таки потрібне використання якоїсь криптосистеми.

Для вирішення цієї проблеми на основі результатів, отриманих класичною і сучасною алгеброю, були запропоновані системи з відкритим ключем.

Суть їх полягає в тому, що кожним адресатом ІС генеруються два ключі, зв'язані між собою за певним правилом. Один ключ оголошується відкритим, а інший закритим. Відкритий ключ публікується і доступний будь-кому, хто бажає послати повідомлення адресату. Секретний ключ зберігається в таємниці.

Початковий текст шифрується відкритим ключем адресата і передається йому. Зашифрований текст в принципі не може бути розшифрований тим же

відкритим ключем. Дешифровка повідомлення можлива тільки з використанням закритого ключа, який відомий тільки самому адресатові.

Криптографічні системи з відкритим ключем використовують так звані необоротні або односторонні функції, які володіють наступною властивістю: при заданому значенні x відносно просто обчислити значення f(x), проте якщо відомо y=f(x), то немає простого шляху для обчислення значення x.

Безліч класів необоротних функцій породжує різноманітність систем з відкритим ключем. Проте не всяка необоротна функція годиться для використання в реальних ІС.

У самому визначенні безповоротності присутня невизначеність. Під безповоротністю розуміється не теоретична безповоротність, а практична неможливість обчислити зворотне значення використовуючи сучасні обчислювальні засоби за досяжний інтервал часу.

Тому щоб гарантувати надійний захист інформації, до систем з відкритим ключем (СВК) пред'являються дві важливих і очевидних вимоги:

1. Перетворення початкового тексту повинно бути необоротним і виключати його відновлення на основі відкритого ключа.

2. Визначення закритого ключа на основі відкритого також повинно бути неможливим на сучасному технологічному рівні. При цьому бажана точна нижня оцінка складності (кількості операцій) розкриття шифру.

Алгоритми шифрування з відкритим ключем набули широкого поширення в сучасних інформаційних системах. Так, алгоритм RSA став світовим стандартом де-факто для відкритих систем і рекомендований МККТТ.

Взагалі ж всі пропоновані сьогодні криптосистеми з відкритим ключем спираються на один з наступних типів необоротних перетворень [19-20]:

- розкладання великих чисел на прості множники;
- обчислення логарифма в скінченному полі;
- обчислення коренів алгебраїчних рівнянь.

Криптосистеми з відкритим ключем (СВК) можна використовувати в трьох напрямках:

1. Як самостійні засоби захисту даних, що передаються або зберігаються.
2. Як засоби для розподілу ключів. Алгоритми СВК більш трудомісткі, ніж традиційні криптосистеми. Тому часто на практиці раціонально за допомогою СВК розподіляти ключі, об'єм яких незначний. А потім за допомогою звичайних алгоритмів здійснювати обмін великими інформаційними потоками.
3. Як засоби автентифікації користувачів (електронний підпис).

Шифри перестановки і заміни не забезпечують високого ступеня надійності, але вони можуть бути покладені в основу розробки більш складних шифрів, у яких при сполученні обох форм надійність коду підвищується [8].

Перевагою методу DES є те, що він є стандартом і за даними Національного Бюро Стандартів США, алгоритм має наступні властивості:

- високий рівень захисту даних проти дешифрування і можливої модифікації даних;
- простота в розумінні;
- високий ступінь складності, що робить його розкриття дорожче одержуваної при цьому прибутку;
- метод захисту ґрунтується на ключі і не залежить від «таємності» алгоритму;
- економічний у реалізації й ефективний у швидкодії.

Важливою характеристикою цього алгоритму є його гнучкість при реалізації. Кожен блок даних шифрується незалежно від інших, що дозволяє розшифрувати окремий блок у зашифрованому повідомленні. Тому можна здійснювати незалежну передачу блоків даних і довільний доступ до зашифрованих даних. Ні тимчасова, ні позиційна синхронізація для операції шифрування не потрібна. Алгоритм виробляє зашифровані дані, у яких кожен біт є функцією від усіх бітів відкритих даних і всіх бітів ключа. Розходження лише в одному біті даних дає в результаті рівні імовірності зміни для кожного біта зашифрованих даних. Недоліки цього алгоритму такі:

− маленький розмір ключа. Для дешифрування інформації досить виконати 2^{56} операцій дешифрування, (тобто усього близько $7*10^{16}$ операцій). Хоча в даний час немає апаратури, що могла б виконати в доступний для огляду період часу подібні обчислення, ніхто не гарантує, що вона не з'явиться в майбутньому;

− окремі блоки, що містять однакові знаки (наприклад, пробіли), будуть однаково виглядати в зашифрованому тексті, що з погляду криптоаналізу невірно.

Щодо алгоритму AES, то необхідно відзначити його стійкість до зламу, хоча обчислювальні затрати перевищують DES.

RSA є перспективним, тому що для шифрування інформації не потрібна передача ключа іншим користувачам. Він має високу криптостійкість, просту програмну і апаратну реалізацію. Але його криптостійкість заснована на обчислювальній складності задачі розкладання великого числа на прості множники і може бути знайдений ефективний алгоритм визначення дільників цілих чисел, у результаті чого метод шифрування стане абсолютно незахищеним. Крім того, не існує строгого доведення, що немає іншого способу визначення секретного ключа по відомому, крім як визначення дільників цілих чисел [21].

Метод шифрування з використанням датчика ПВЧ [12] найбільше часто використовується в програмній реалізації систем криптографічного захисту даних. Тому що він досить простий для програмування, дозволяє створювати алгоритми з дуже високої криптостійкістю й ефективність цього методу шифрування дуже висока. Системи, засновані на методі шифрування з використанням датчика ПВЧ, дозволяють у секунду зашифрувати від декількох десятків до декількох сотень Кбайт.

Приведений вище аналіз показує, що для шифрування інформації, яка вимагає середнього ступеня захисту найбільш прийнятним є використання гамування і лінійного конгруентного датчика ПВЧ.

1.4 Аналіз методів стеганографічного захисту

Стеганографія (у перекладі з грецького - «тайнопис»; steganos – таємниця, секрет; graphy – запис) на відміну від криптографії розглядає методи приховування не лише інформації, але і самого факту її передачі. З цією метою приховувана інформація деякого об'єму вкладається в яке-небудь загальнодоступне повідомлення, наприклад в текст, зображення, звук та інше. При вкладенні приховуваного повідомлення відбувається як би його розсіяння за всім обсягом основного або покриваючого повідомлення. При цьому останнє не повинне зазнавати видимих змін (спотворень), які можуть бути легко виявлені сторонніми особами [8, 22-23].

Якщо криптографічні методи, швидше за все, слід відносити до оборонних інструментів інформаційного протиборства, то стеганографія через свою специфіку може стати в XXI столітті технологією створення нової високоефективної інформаційної зброї. Хоча такий поділ є досить умовним.

Джерела сучасної стеганографії втрачаються в глибокій старовині. З книг грецького історика Геродота (V століття до н. э.), що дійшли до нас, відомий приклад прихованої відправки секретного повідомлення спартанцем Демаркусом. Знаходячись на території Персії, яка готувалася вступити у війну з Грецією, він дізнався про план майбутнього нападу персів. Про що і відправив повідомлення до Спарти, написавши його на дерев'яній дощечці, яку зверху покрив воском. На нім, як тоді було прийнято, написав інше цілком нешкідливе повідомлення. Перси ні про що не здогадалися, але їх план нападу став відомий грекам [8].

Загальне уявлення про етапи розвитку стеганографії дає схема, наведена на рис. 1.1.

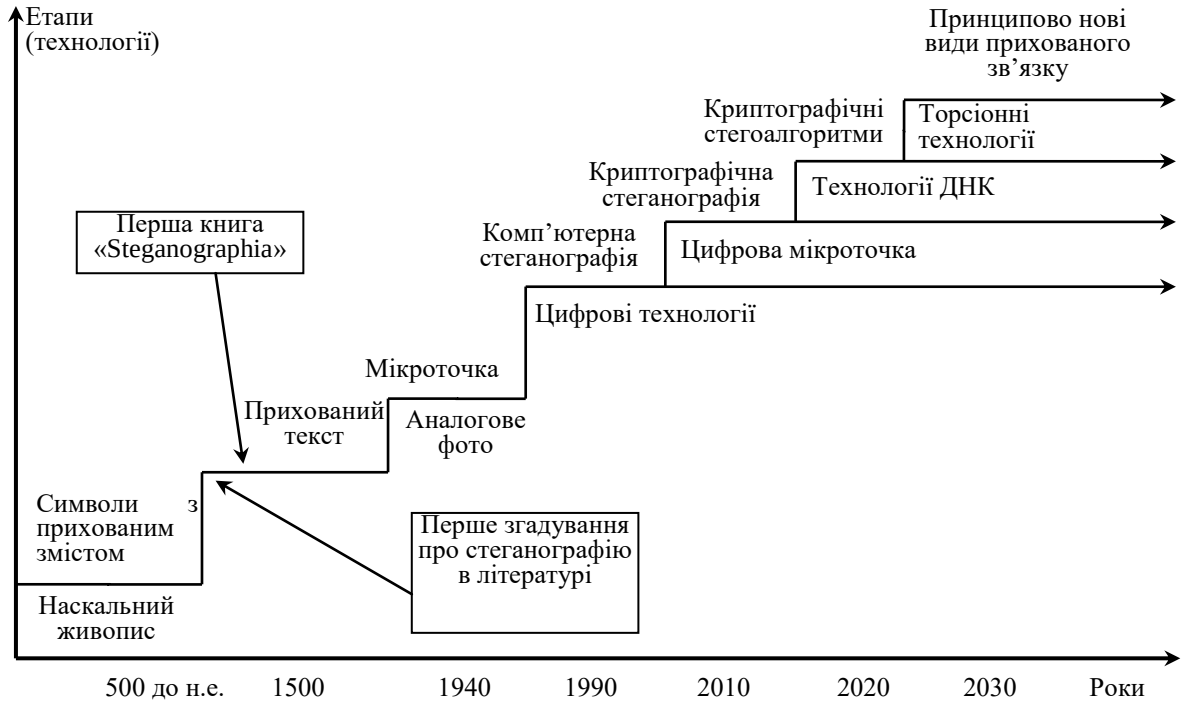


Рисунок 1.1 – Етапи розвитку стеганографії

В даний час стеганографія нестримно розвивається на базі досягнень теорії інформації, статистичної теорії зв'язку, цифрової обробки сигналів, новітніх інформаційних технологій та ін.

У загальному вигляді стеганографічний канал передачі (зберігання) інформації можна представити наступною схемою, показаною на рис. 1.2. Конфіденційні повідомлення $m \in M$ (аудіо-, видео-, комп'ютерні дані, нерухоме зображення і ін.), які підлягають приховуванню, заздалегідь можуть бути оброблені криптографічними методами, тобто зашифровані за допомогою ключа шифрування $k_ш \in K_ш$, де M, $К_ш$ – множина можливих повідомлень і ключів шифрування відповідно. (Далі великі букви використані для позначення множини криптограм, стеганограм і так далі). В результаті маємо криптограму $e \in E$, яка потім за допомогою стеганографічних перетворень, керованих стегоключом $k_{св} \in К_{св}$, вкладається в покриваюче (маскуюче) повідомлення $c \in$

C (cover-message).

Безумовно, це повідомлення, зване також повідомленням – контейнером, за своїм обсягом в бітах повинно значно перевищувати об'єм інформації, що вкладається [8]. Інакше ніякого «таємного розсіювання» секретного повідомлення не вийде. Стеганограма s Є *S* далі передається по відкритому каналу телекомунікацій або підлягає зберіганню в доступному будь-якому користувачеві місці. Саме на цьому етапі стегоаналітик (порушник, що атакує) намагається здійснити успішну атаку з метою:

- визначення факту вкладення конфіденційного повідомлення;
- видобування і прочитання вкладеного повідомлення;
- заміни вкладеного повідомлення на інше правдоподібне повідомлення (імітація);
- видалення вкладеного конфіденційного повідомлення «про всяк випадок».

Рисунок 1.2 – Загальна схема стеганографічної системи: ДСІ – джерело секретної інформації; ОСІ – отримувач секретної інформації; МП – маскуюче повідомлення

Перші дві атаки є пасивними, оскільки в цьому випадку відсутні спроби змінити дані, доступні для атакуючого. Решта атак може бути названо активними.

На приймальній стороні із стеганограми $\hat{S}$ (знак «$\hat{S}$» означає наслідок атаки або дії перешкод в процесі передачі) за допомогою стегоключа k_{si} Є K_{si} і ключа дешифровки $k_д$ Є $K_д$ отримують спочатку криптограму, а потім

витягується вкладене повідомлення .

Під загальну схему, представлену на рис. 1.2, підпадає власне стеганографія, а також процес приховування і витягання так званих цифрових водяних знаків (watermarking). Їх призначення - ідентифікація інформації, інтелектуальної власності, що належить власникові, тобто захист його прав. Таку інформацію містять, наприклад, компакт-диски з музичними записами, відеофільмами і іншими ексклюзивними програмами. Це може бути також картографічна або криміналістична інформація (фотографії злочинців, відбитки пальців) і тому подібне. Крім того, цифрові водяні знаки (ЦВЗ) застосовуються для автентифікації (встановлення достовірності) інформації, яка передається в каналах телекомунікацій. Так, в ході воєн на Близькому Сході в 1970-х роках ізраїльські радисти неодноразово входили в радіомережі управління танковими підрозділами і на арабській мові передавали помилкові накази на передислокацію танків в райони, зайняті ізраїльськими військами. В результаті ці підрозділи знищувалися або не виконували завдання, поставлені перед ними арабським командуванням [5]. При використанні ЦВЗ цього могло б не статися.

Іншими словами, при стеганографії головну цінність представляє конфіденційна інформація, що вкладається, а покриваюче повідомлення використовується лише для її маскування. На відміну від цього вкладення ЦВЗ не ховається, а зловмисник не повинен мати можливості витягувати або підмінити його з метою обману власника інформації, захист якої і забезпечується в даному випадку. При цьому цифрові водяні знаки бувають «0»-бітові або «m»-бітові, m = 1, 2 ... У першому випадку значення має сам факт вкладення ЦВЗ, в другому - такий знак містить деякі ідентифікаційні відомості про власника або автора даного інформаційного об'єкту.

Всі системи стеганографії і з цифровими водяними знаками по аналогії з криптографією реалізуються на основі правила Керхгоффа, коли зловмиснику може бути відомо все, окрім ключів вкладення і видобування інформації. Якщо ключі однакові, то в цьому випадку система називається симетричною. Коли ключі відмінні один від одного, вона стає несиметричною. Такі системи

зазвичай реалізують на основі алгоритму RSA (R. Rivest, A. Shamir, L. Adleman), названого по початкових літерах прізвищ його авторів, або алгоритму Ель-гамаля [20-21].

Ефективність систем стеганографії і цифрових водяних знаків прийнято оцінювати за допомогою наступних імовірнісно-часових характеристик. Це – імовірність не виявлення або помилкового виявлення вкладеної конфіденційної інформації в маскуючому повідомленні, ймовірності помилкового прийому прихованих повідомлень і ЦВЗ, а також швидкості передачі вкладеної інформації по відношенню до швидкості передачі, маскуючого повідомлення. Ці системи також можна оцінювати величиною відношення сигнал-шум і якістю сприйняття маскуючого повідомлення на слух або візуально людиною. Важливим показником є і складність реалізації [8].

Прийняті терміни:

Контейнер – будь-яка інформація, призначена для захована таємних повідомлень.

Порожній контейнер – контейнер без вбудованого повідомлення; заповнений контейнер або стего-контейнер, що містить вбудовану інформацію.

Вбудоване (приховане) повідомлення - повідомлення, що вбудовується в контейнер.

Стеганографічний канал або просто стегоканал – канал передачі стего-повідомлення.

Стегоключ або просто ключ – секретний ключ, необхідний для приховання інформації. У залежності від кількості рівнів захисту (наприклад, вбудовування попереднє зашифрованого повідомлення) у стегосистемі може бути один або декілька стегоключів [22].

По аналогії з криптографією, за типом стегоключа стегосистеми можна підрозділити на два типи [22-23]:

- з закритим ключем;
- з відкритим ключем.

У стегосистемі з закритим ключем використовується один ключ, який має бути визначений або до початку обміну секретними повідомленнями, або переданий по захищеному каналу.

У стегосистемі з відкритим ключем для вбудовування і видобування повідомлення використовуються різні ключі, які розрізняються таким чином, що за допомогою обчислень неможливо вивести один ключ з іншого. Тому один ключ може передаватися вільно по незахищеному каналу зв'язку. Крім того, дана схема добре працює і при взаємній недовірі відправника та адресатаодержувача. У загальному випадку вбудоване повідомлення може бути зашифроване іншими методами криптографії.

Основні вимоги до стеганосистеми такі:

1. Методи приховування повинні забезпечувати автентичність і цілісність файлу.
2. Передбачається, що криптографу повністю відомі можливі стеганографічні методи.
3. Безпека методів ґрунтується на збереженні стеганографічним перетворенням основних властивостей файлу, що відкрито передається при внесенні до нього секретного повідомлення і деякою невідомою противникові інформації - ключа.
4. Навіть якщо факт повідомлення став відомий противникові, витягання секретного повідомлення представляє складне обчислювальне завдання.

Сфери застосування стеганографії:

- захист від копіювання – електронна комерція, контроль за копіюванням (DVD), поширення мультимедійної інформації (відео за запитом);
- прихована анотація документів – медичні знімки, картографія, мультимедійні бази даних;
- автентифікація – системи відеоспостереження, електронної комерції,

голосової пошти, електронне конфіденційне діловодство;

- прихований зв'язок – військове і розвідувальне застосування, а також вживання у випадках, коли криптографію використовувати не можна.

Класифікація методів стеганографії наведена на рисунку 1.3. До технологічних методів відносять ті, що базуються на використанні хімічних або фізичних властивостей різних матеріальних носіїв інформації. На сьогодні в цьому плані найбільше зацікавлення викликають стандартні носії інформації: аудіо, відео та обчислювальної техніки. Наприклад, багато мультимедійних форматів даних мають поля розширень, які заповнюються нульовою інформацією та не враховуються програмою; можна записати інформацію й на не використовуваних місцях гнучких магнітних дисків (на нульовій доріжці тощо).

Рисунок 1.3 – Класифікація стеганографічних методів

Подібні методи прості у використанні, проте вони мають низький рівень стійкості та забезпечують передачу порівняно невеликих об’ємів даних. Клас інформаційних методів містить у собі лінгвістичні та цифрові.

Лінгвістичні методи базуються на використанні надмірності людської мови або іншого середовища, що не містить букв та цифр (фотографії, графіка,

креслення та інше). До цього класу можна також віднести метод генерації псевдо осмисленого тексту, необхідного для приховання таємного повідомлення, а також методи, що базуються на зміні положення рядків на сторінці чи слів у реченні тощо.

Більшість цифрових методів базується на тому, що, з одного боку, файли, які не потребують абсолютної точності, можуть бути дещо видозмінені без втрати функціональності, а з іншого – на відсутності спеціального інструментарію або нездатності органів чуття людини надійно розрізняти незначні зміни в таких файлах.

Методи цифрової стеганографії прийнято класифікувати за різними основами. За способом вкраплення інформації в контейнер розрізняють методи сурогатної, селектуючої та конструюючої стеганографії. Сурогатна стеганографія може бути застосована для досить «шумних» контейнерів, у яких заміна частини бітів контейнера на біти таємного повідомлення не буде помітною для стороннього спостерігача. Що ж до методів селектуючої стеганографії, то вважається, що кодування захованого повідомлення повинно відповідати спеціальним статистичним характеристикам шуму контейнера. Для цього генерується велика кількість альтернативних контейнерів, з яких потім вибирається найбільш підходящий для «вкраплення» даного повідомлення.

У конструюючій стеганографії вибір контейнера залежить від таємного повідомлення, тобто в цьому випадку контейнер генерується самою стегосистемою.

За способом доступу до інформації розрізняють потокові методи та методи з довільним доступом. Перші з них працюють із потоками неперервних даних, коли біти повідомлення необхідно включати в контейнер у режимі реального часу. Другі використовують контейнери фіксованої довжини, якими служать файли певного наповнення (текст, програми, графіка, звук тощо.). На практиці частіше всього використовують саме контейнери фіксованої довжини, як більш зручні та доступні.

За способом організації контейнерів методи цифрової стеганографії

поділяються на систематичні та не систематичні. У систематично організованих контейнерах інформаційні біти можна відокремити від шумових, у яких і буде розміщуватися таємне повідомлення. При несистематичній організації контейнера такого розділення не існує і для виділення повідомлення необхідно опрацьовувати всі біти контейнера.

При видобуванні таємної інформації в деяких стеганографічних методах необхідно мати еталон використовуваного контейнера. У цьому випадку потрібно забезпечити його надійне зберігання та захист від несанкціонованого використання. Більшість сучасних методів не потребують наявності еталона контейнера, а для його отримання використовується спеціальна обробка стеганограми.

За методом обробки контейнера цифрові методи поділяються на дві групи: безпосередні та спектральні. При використанні безпосередніх методів обробці підлягають біти самого контейнера, як, наприклад, у методі найменшого значущого біта, методі заміни палітри кольорів або методі сортування палітри. Спектральні методи базуються на використанні дискретних унітарних перетворень: Фур'є, Уолша, Карунена-Лоєва, слент, вейвлет та інших. При використанні методів цієї групи обробці підлягає не вихідний, а відповідні спектральні контейнери [11].

Можна навести спрощену класифікацію стеганографічних методів. Згідно цього підходу криптографічні методи поділено на 3 розділи:

- класична стеганографія - включає всі «некомп'ютерні методи»;

- комп'ютерна стеганографія - напрям класичної стеганографії, заснований на особливостях комп'ютерної платформи і використання спеціальних властивостей комп'ютерних форматів даних;

- цифрова стеганографія - напрям класичної стеганографії, заснований на приховуванні або впровадженні додаткової інформації в цифрові об'єкти, викликаючи при цьому деякі спотворення цих об'єктів. Використовується надмірність кодування аудио- і візуальної інформації.

Прикладом комп'ютерної стеганографії може бути використання регістра

букв. Нехай нам необхідно заховати букву "A" в тексті "stenography". Для цього беремо двійкове представлення коду символу "A" - "01000001". Нехай для позначення біта, що містить одиницю, використовується символ нижнього регістра, а для нуля – верхнього. Тому після накладення маски "01000001" на текст "stenography", результат буде "sTenogrAphy". Закінчення "phy" нами не використано оскільки для захована одного символу використо-вується 8 байт (по байту на кожен біт), а довжина рядка 11 символів, ось і вийшло, що останні 3 символи "зайві". Використовуючи таку технологію можна заховати в текст довжиною N символів, повідомлення з N/8 символів.

Використання пропусків. Пропуск позначений символом з кодом 32, але в тексті його можна замінити також символом що має код 255 або TAB. Також як і в минулому прикладі, передаємо біти шифрованого повідомлення| використовуючи звичайний текст. Але цього разу 1 - це пропуск, а 0 - це пропуск з кодом 255.

Використання специфіки файлових систем - при зберіганні файл завжди займає ціле число блоків - це зроблено з міркувань зручності адресації. Через це зберігання маленьких файлів реалізується украй нераціонально - в системі Fat32 файл розміром 1 байт займає на диску 4 Кб. При цьому очевидно, що всі 4 Кб, за виключенням 1 байта, забито сміттям і ніяк не використовуються. Але вони можуть бути зайняті корисними даними. Такий метод широко використовується для довготривалого прихованого зберігання невеликих об'ємів інформації - проте він досить легко виявляється і украй небезпечний. Також існує можливість запису даних просто на невживані області накопичувачів (наприклад, нульову доріжку).

Використання зарезервованих полів комп'ютерних форматів даних - Поля розширення є в багатьох мультимедійних форматах, вони заповнюються нульовою інформацією і не враховуються програмою.

На відміну від комп’ютерної стеганографії цифрова стеганографія передбачає використання методів обробки сигналів і приховування даних в файлах зображень, аудіофайлах або інших файлах, що містять оцифровані дані.

Саме цифрова стеганографія найбільш цікава з точки зору приховування інформації, оскільки забезпечує найбільші об’єми даних, що приховуються та найвищий ступінь захисту.

Цифрова стеганографія включає наступні напрями [8]:

1) вбудовування інформації з метою її прихованої передачі;

2) вбудовування цифрових водяних знаків (ЦВЗ) (watermarking);

3) вбудовування ідентифікаційних номерів (fingerprinting);

4) вбудовування заголовків (captioning).

ЦВЗ можуть застосовуватися, в основному, для захисту від копіювання і несанкціонованого використання. У зв'язку з бурхливим розвитком технологій мультимедіа гостро встало питання захисту авторських прав і інтелектуальної власності, представленого в цифровому вигляді. Прикладами можуть бути фотографії, аудіо і відеозаписи і так далі Переваги, які дають подання і передача повідомлень в цифровому вигляді, можуть виявитися перекресленими легкістю, з якою можлива їх крадіжка або модифікація. Тому розробляються різні заходи захисту інформації, організаційного і технічного характеру. Один з найбільш ефективних технічних засобів захисту мультимедійної інформації і полягає у вбудовуванні в об'єкт, що захищається, невидимих міток – ЦВЗ. Розробки в цій області ведуть найбільші фірми у всьому світі. Оскільки методи ЦВЗ почали розроблятися абсолютно недавно, тут є багато неясних проблем, що вимагають свого вирішення [1]. Назву цей метод отримав від всім відомого способу захисту цінних паперів, у тому числі і грошей, від підробки. Термін «Digital watermarking» був вперше застосований в роботі [5]. На відміну від звичайних водяних знаків ЦВЗ можуть бути не лише видимими, але і (як правило) невидимими. Невидимі ЦВЗ аналізуються спеціальним декодером, який виносить рішення про їх коректність. ЦВЗ можуть містити деякий автентичний код, інформацію про власника, або яку-небудь керуючу інформацію. Найбільш часто об'єктами захисту за допомогою ЦВЗ є нерухомі зображення, файли аудіо і відеоданих.

Технологія вбудовування ідентифікаційних номерів виробників має багато спільного з технологією ЦВЗ. Відмінність полягає в тому, що в першому випадку кожна захищена копія має свій унікальний вбудований номер (звідси і назва – дослівно «відбитки пальців»). Цей ідентифікаційний номер дозволяє виробникові відстежувати подальшу долю свого дітища: чи не зайнявся хто-небудь з покупців незаконним тиражуванням. Якщо так, то «відбитки пальців» швидко вкажуть на винного.

Вбудовування заголовків (невидиме) може застосовуватися, наприклад, для підпису медичних знімків, нанесення легенди на карту і так далі Метою є зберігання різнорідно представленої інформації в єдиному цілому. Це, мабуть, єдиний додаток стеганографії, де в явному вигляді відсутній потенційний порушник.

Оскільки цифрова стеганографія є молодою наукою, то її термінологія не до кінця устоялася. Основні поняття стеганографії були узгоджені на першій міжнародній конференції з приховування даних. Проте, навіть само поняття «стеганографія» трактується по різному. Так, деякі дослідники розуміють під стеганографією тільки приховану передачу інформації. Інші відносять до стеганографії такі застосування як, наприклад, метеорний радіозв'язок, радіозв'язок з псевдовипадковою перебудовою радіочастоти, широкосмуговий радіозв'язок. На наш погляд, неформальне визначення того, що таке цифрова стеганографія, могло б виглядати таким чином: «наука про непомітне і надійне приховування одних бітових послідовностей в інших, що мають аналогову природу». Під це визначення якраз підпадають всі чотири наведені вище напрями приховування даних, а радіозв'язок – ні [8]. Крім того, у визначенні міститься дві головні вимоги до стеганографічного перетворення: непомітність і надійність, або стійкість до різного роду спотворень. Згадку про аналогову природу цифрових даних підкреслює той факт, що вбудовування інформації виконується в оцифровані безперервні сигнали. Таким чином, в рамках цифрової стеганографії не розглядуються питання впровадження даних в заголовки IP-пакетів і файли різних форматів, в текстові повідомлення.

Які б не були різні напрями стеганографії, вимоги, що пред'являються ними, багато в чому збігаються, як це буде показано далі. Найбільш істотна відмінність постановки завдання прихованої передачі даних від постановки завдання вбудовування ЦВЗ полягає в тому, що в першому випадку порушник повинен виявити приховане повідомлення, тоді як в другому випадку про його існування всі знають. Більш того, у порушника на законних підставах може бути пристрій виявлення ЦВЗ (наприклад, у складі DVD-програвача).

Слово «непомітному» в нашому визначенні цифрової стеганографії має на увазі обов'язкове включення людини в систему стеганографічної передачі даних. Людина тут може розглядуватися як додатковий приймач даних, що пред'являє до системи передачі вимоги, що важко формалізуються.

Істотний вплив на надійність стегосистеми і можливість виявлення факту передачі прихованого повідомлення має вибір контейнера.

Наприклад, досвідчене око цензора з художньою освітою легко виявить зміну колірної гамми при впровадженні повідомлення в репродукцію "Мадонни" Рафаеля або "Чорного квадрата" Малевича.

По протяжності контейнери можна підрозділити на два типи: безперервні (потокові) і обмеженої (фіксованою) довжини. Особливістю потокового контейнера є те, що неможливо визначити його початок або кінець. Більш того, немає можливості дізнатися заздалегідь, якими будуть подальші шумові біти, що приводить до необхідності включати біти, що приховують повідомлення в потік в реальному масштабі часу, а самі приховуючі біти вибираються за допомогою спеціального генератора, який задає відстань між послідовними бітами в потоці.

У безперервному потоці даних найбільша трудність для одержувача – визначити, коли починається приховане повідомлення. За наявності в потоковому контейнері сигналів синхронізації або меж пакету, приховане повідомлення починається відразу після одного з них. У свою чергу, для відправника можливі проблеми, якщо він не упевнений в тому, що потік контейнера буде достатньо довгим для розміщення цілого таємного

повідомлення.

При використанні контейнерів фіксованої довжини відправник заздалегідь знає розмір файлу і може вибрати приховуючі біти у відповідній псевдовипадковій послідовності. З іншого боку, контейнери фіксованої довжини, як це вже наголошувалося вище, мають обмежений об'єм і інколи вбудовуване повідомлення може не помістится у файл-контейнер.

Інший недолік полягає в тому, що відстані між приховуючими бітами рівномірно розподілені між найбільш коротким і найбільш довгим заданими відстанями, тоді як дійсний випадковий шум матиме експоненціальний розподіл довжин інтервалу. Звичайно, можна породити псевдовипадкові, експоненціально розподілені числа, але цей шлях зазвичай дуже трудомісткий. Проте на практиці найчастіше використовуються саме контейнери фіксованої довжини, як найбільш поширені і доступні [8].

Можливі наступні варіанти контейнерів:

- контейнер генерується самою стегосистемою. Прикладом може служити програма MandelSteg, в якій як контейнер для вбудовування повідомлення генерується фрактал Мандельброта. Такий підхід можна назвати конструюючою стеганографією;
- контейнер вибирається з деякої множини контейнерів. В цьому випадку генерується велике число альтернативних контейнерів, щоб потім вибрати найбільш відповідний для приховування повідомлення. Такий підхід можна назвати селектуючою стеганографією. В даному випадку при виборі оптимального контейнера з множини, що згенерували, найважливішою вимогою є природність контейнера. Єдиною ж проблемою залишається те, що навіть оптимально організований контейнер дозволяє заховати незначну кількість даних при дуже великому об'ємі самого контейнера;
- контейнер поступає ззовні. В даному випадку відсутня можливість вибору контейнера і для захования повідомлення береться перший

контейнер, що попався, не завжди відповідний до вбудовуваного повідомлення. Назвемо це безальтернативною стеганографією.

З урахуванням вище сказаного в даній роботі будемо використовувати фіксований контейнер у вигляді файлів зображень формату BMP.

Більшість досліджень присвячена використанню як стегоконтейнерів зображення [10-11]. Це обумовлено наступними причинами:

– існуванням практичного завдання захисту фотографій, картин, відео від незаконного тиражування і розповсюдження;

– відносно великим об'ємом цифрового представлення зображень, що дозволяє впроваджувати дані великого об'єму;

– заздалегідь відомим розміром контейнера, відсутністю обмежень, що накладаються вимогами реального часу;

– наявністю в більшості реальних зображень областей текстур, що мають шумову структуру і добре придатних для вбудовування інформації;

– слабою чутливістю людського ока до незначних змін кольорів зображення, його яскравості, контрастності, вмісту шуму, спотворень поблизу контурів;

– добре розробленими останнім часом методами цифрової обробки зображень.

Загальна схема вбудовування інформації в зображення представлена на рис. 1.4.

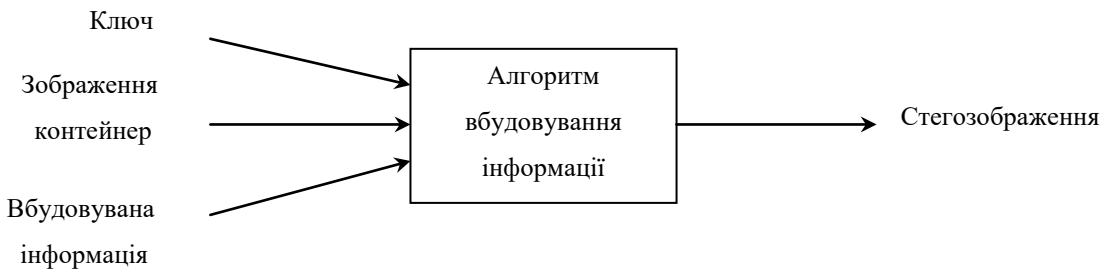


Рисунок 1.4 – Загальна схема вбудовування інформації в зображення

Загальна схема видобування інформації із зображення представлена на рис. 1.5.

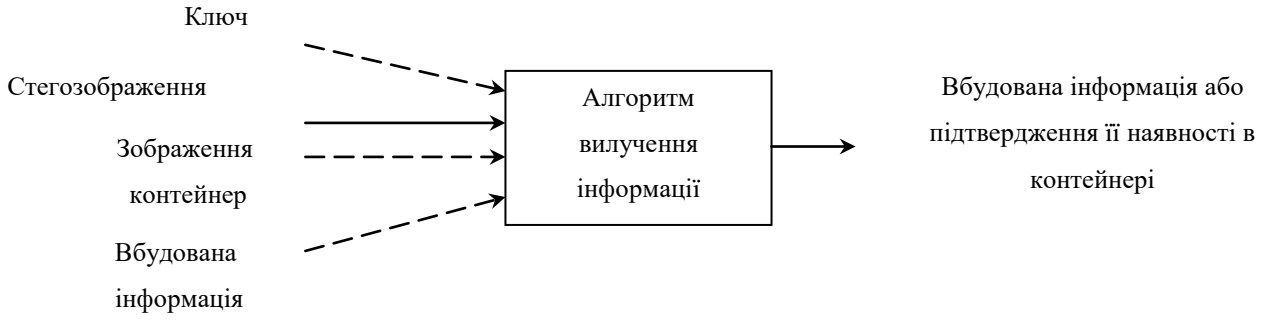


Рисунок 1.5 – Загальна схема вилучення інформації

Зображення володіють великою психовізуальною надмірністю. Око людини подібне до низькочастотного фільтру, якому непомітні спотворення у високочастотній області зображень. Стеганографія заснована на використанні наявної в зображеннях психовізуальної надмірності [8]. Класифікація властивостей системи людського зору приведена в табл. 1. 1.

Узагальнена схема впровадження даних в зображення ґрунтується на наступних процедурах:

1. Фільтрації зображення на основі орієнтованих смугових фільтрів.
2. Обчисленні порогу маскування.
3. Приведенні енергії впроваджуваного сигналу до значення, меншого, ніж поріг маскування для кожного компоненту [8].

Таблиця 1.1 – Властивості системи людського зору

Низькорівневі	**Високорівневі**
1. Чутливість до зміни яскравості зображення 2. Частотна чутливість 3. Ефект маскування	1. Чутливість до: - контрасту - розміру - місцю розташування

	- кольору - формі - зовнішнім подразникам 2. Підвищена увага до зображень переднього плану

До теперішнього часу реалізована велика кількість методів впровадження інформації в зображення, які можуть бути розділені на невелику кількість груп, схожих по використовуваній ідеї.

Приховування даних в нерухомих зображеннях здійснюється на підставі наступних двох груп методів:

1. Прямі методи модифікації зображення в просторовій області.

2. Методи, що модифікують зображення, заздалегідь перетворене в іншу форму.

Прямі методи вбудовують інформацію безпосередньо в підмножину пікселів зображення. Вона впроваджується за рахунок маніпуляцій яскравістю і колірними складовими без обчислювання громіздких лінійних перетворень зображення.

Методи даного класу розрізняються тільки вибором підмножини пікселів, що модифікується, і стратегією зміни значень пікселів.

Найбільш відомими представниками прямих методів є:

– LSB-методи (Least Significant Bits);

– методи модифікації палітри.

LSB-методи. У найбільш простому випадку як підмножина пікселів, що модифікується, вибираються найменш значущі біти зображення.

Один з методів поліпшення якості LSB-методів полягає у використанні надмірності і управління місцем розміщення повідомлення за допомогою секретного ключа.

LSB-методи не мають серйозної перспективи, оскільки вони нестійкі до

таких дій, як фільтрація, конвертація кольорів, ущільненню з втратами [8].

Методи модифікації палітри. Вбудовування даних в палітру ґрунтується на перенумерації кольорів палітри, при цьому самі кольори не змінюються. Вбудовані за таким принципом дані будуть невидимими. Недоліком цього методу є обмеженість об'єму інформації, яка може бути вбудована в палітру. Крім того, модифікована палітра може бути втрачена при повторному збереженні файлу [22].

Найбільш перспективні прямі алгоритми: Kutter, Bruyndonckx, Langelaar, Pitas, Rongen, Patchwork, Bender, Marvel.

В методах, що використовують попереднє перетворення, інформація впроваджується за рахунок декомпозиції зображення-контейнера. Ці методи використовують переваги, якими володіє представлення зображення скінченним набором коефіцієнтів. Як правило такі методи мають хороші характеристики.

У цій групі використовуються достатньо|досить| різноманітні|всілякі| трансформації:

– дискретне косинусне перетворення (ДКП);
– вейвлет-перетворення;
– дискретне перетворення Фурье;
– перетворення Карунена-Лоєва;
– сингулярне розкладання.

Дослідження в області сигнальних перетворень ведуться надзвичайно інтенсивно. Постійно з'являються все нові і нові алгоритми, серед яких слід зазначити: Koch, Benham, Podilchuk, Hsu, Tao, Cox, Barni, Fridrich [1].

Коротка класифікація стегоалгоритмів за способом вбудовування інформації в нерухомі зображення приведена в табл. 1.2.

Таблиця 1.2 – Стегоагоритми впровадження інформації в зображення

Лінійні (аддитивні)	**Нелінійні**	**Інші**
На основі лінійного вбудовування даних: Cox, Barni, Nicchiotti, Dugad, Kim, Loo, Lu Podilchuk, Xia, Wang. На основі злиття цифрового водяного знаку і контейнера: Chae, Kundur.	На основі скалярного квантування: Chu, Hsu. На основі векторного квантування: Chae.	На основі фрактального перетворення: Bas, Puate, Davern.

В даний час розроблено безліч стеганографічних програм, що дозволяють вбудовувати інформацію в нерухомі зображення. Їх короткий опис розміщений в табл. 1.3.

Методи з модифікацією зображень дозволяють вмонтовувати в контейнер більше інформації у порівнянні з методами прямого приховування, однак, вимагають значних обчислювальних ресурсів.

Найбільш популярними стандартами кодування відео є MPEG-2 і MPEG-4.

Стеганографічні методи, які використовують для вбудовування інформації у відео, ущільнене за стандартом MPEG-2 (далі – MPEG), повинні працювати в реальному часі.

Способи вбудовування ЦВЗ, що працюють в реальному часі, повинні відповідати декільком вимогам і, насамперед вони мають бути сліпими і володіти малою обчислювальною складністю.

Таблиця 1.3 – Стеганографічне ПЗ

Операційна система Windows	
Steganos Security Suite	Пакет утиліт приховування і шифрування даних. Програма дозволяє приховувати заздалегідь зашифровані дані будь-якого типа у файлах .BMP і .WAV. Приховування даних у файлах .BMP і .WAV досягається шляхом зміни кожного LSB (least significant bit - молодший біт) одиниці інформації.
S-Tools	Стеганографічна програма, що дозволяє ховати інформацію в графічних (формати BMP і JPG) і аудіо-файлах (формат WAV). Інформація також може бути зашифрована за допомогою криптографічних алгоритмів.
FS	Програма використовується для приховування файлів в BMP, GIF і JPEG зображеннях.
Steganos	Легка у використанні, але все таки потужна програма для шифрування файлів і приховування їх усередині форматів файлів BMP, DIB, VOC, WAV, ASCII, HTML. Для зручності використання програма виконана у вигляді майстра. Це 32-розрядне застосування містить власний Shredder — програму, яка знищує файли з жорсткого диска. З новими властивостями і додатковими можливостями Steganos є серйозним конкурентом на ринку ПЗ для забезпечення інформаційної безпеки приховування файлів.
Hide and Seek	Ця стегопрограма приховує будь-які дані в GIF зображенні
Contraband	Програмне забезпечення, що дозволяє приховувати будь-які файли в 24 бітових графічних файлах формату BMP.
Операційна система DOS	
Jsteg	Програма призначена для приховуання інформації в популярному форматі JPG
StegoDos	Пакет програм, що дозволяє вибирати зображення, приховувати в них повідомлення, відображувати і зберігати зображення в іншому графічному форматі.

Wnstorm	Пакет програм, який дозволяє шифрувати повідомлення і приховувати його усередині графічного файлу PCX формату.
Операційна система OS/2	
Hide4PGP v1.1	Програма, яка дозволяє приховувати інформацію у файлах формату BMP, WAV і VOC. При цьому для приховування можна використовувати будь-яке число самих молодших бітів.
Stego	Дозволяє впроваджувати дані у файли формату PICT без зміни зовнішнього вигляду і розміру PICT -файла.
Операційна система Unix	
Gifshuffle	Кодування тексту в GIF зображеннях
JPHS	Програма приховує файл в JPG зображенні

Таким чином, єдино допустимими є методи, що вбудовують дані безпосередньо в потік ущільнених даних, щоб уникнути зайвих обчислень, як це показано на рис.1.6.

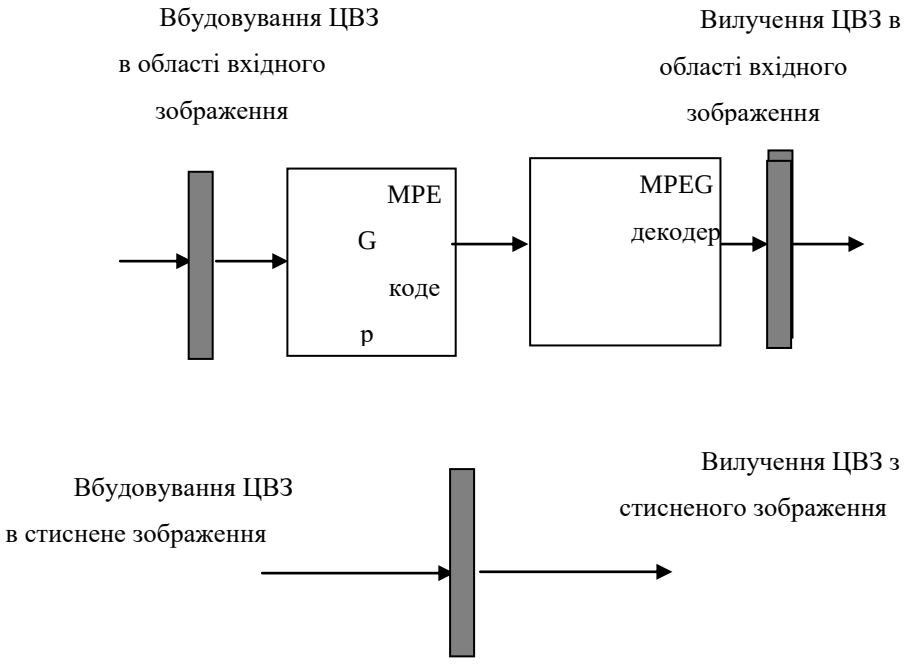


Рисунок 1.6 – Вбудовування / вилучення ЦВЗ в розгорнені дані і здійснення цієї ж операції із стислими даними

Крім того, операція по впровадженню ЦВЗ не повинна збільшувати розмір ущільнених відеоданих. Якщо розмір даних збільшується, то можуть

виникнути проблеми при передачі потоку відео даних по каналу фіксованої швидкості.

У будь-якому випадку задача приховування даних в файлах MPEG доволі складна і вимагає знання як особливостей ущільнення відеозображень цим методом, так і форматів файлів MPEG [22-23].

Для того, щоб перейти до обговорення питань впровадження інформації в аудіосигнали, необхідно визначити вимоги, які можуть бути пред'явлені до стегосистем, що використовуються для приховування інформації в аудіосигналах:

- приховувана інформація має бути стійкою до наявності різних шумів, ущільнення з втратами, фільтрування, аналогово-цифрового і цифро-аналогового перетворень;
- приховувана інформація не повинна вносити до сигналу спотворення, що сприймаються системою слуху людини;
- спроба видалення приховуваної інформації повинна приводити до помітного пошкодження контейнера (для ЦВЗ);
- приховувана інформація не повинна вносити помітних змін до статистики контейнера.

Для приховуваної інформації в аудіосигнали можна використовувати методи, що застосовуються в інших видах стеганографії. Наприклад, можна вбудовувати інформацію, заміщаючи найменш значущі біти (всі або деякі). Або можна створювати стегосистеми, ґрунтуючись на особливостях аудіосигналів і системи слуху людини.

Систему слуху людини можна представити, як аналізатор частотного спектру, який може виявляти і розпізнавати сигнали в діапазоні 10 – 20000 Гц. Систему слуху людини можна змоделювати, як 26 смугових фільтри, смуга пропускання, яких збільшується із збільшенням частоти. Система слуху людини розрізняє зміни фази сигналу слабше, ніж зміни амплітуди або

частоти.

Аудіосигнали можна розділити на три класи [1]:

- розмова телефонної якості, діапазон 300 – 3400 Гц;
- широкосмугова мова 50 – 7000 Гц;
- широкосмугові аудіосигнали 20 – 20000 Гц.

Практично всі аудіосигнали мають характерну особливість. Будь-який з них має достатньо великий об'єм даних, для того, щоб використовувати статистичні методи впровадження інформації. Впровадження може вестись як в часовій області так і в частотній.

У порівнянні з зображеннями звукові файли менш поширені в мережі Інтернет, тому за цим показником перевагу мають зображення.

1.5 Постановка задачі досліджень

1. Аналіз методів захисту біомедичних даних від несанкціонованого доступу показав, що для їх захисту використовуються традиційні криптографічні перетворення, які не враховують специфіку зображень, тому актуальною є розробка методів захисту біомедичних зображень, орієнтованих на врахування природи цих зображень і такими методами є методи цифрової стеганографії. Ці методи у порівнянні з такими стандартами шифрування як DES або AES забезпечують достатній рівень захисту при простій програмній реалізації.
2. У результаті аналізу основних методів цифрової стеганографії в якості контейнера найбільш прийнятним є використання зображень. Перевагами зображень є відносно великі об'єми цифрового представлення зображень, що дозволяє вбудовувати дані великого об'єму; наявністю в більшості реальних зображень областей текстур, що мають шумову структуру і добре придатних для вбудовування інформації; слабкою чутливістю людського ока до незначних змін кольорів зображення, його яскравості, контрастності, вмісту шуму,

спотворень поблизу контурів; добре розробленими останнім часом методами цифрової обробки зображень.

3. Подальших досліджень вимагають методи вбудовування біомедичних зображень в біомедичні зображення, зокрема, вбудовування рентгенівських зображень в контейнер типу рентгенівське зображення.
4. Необхідно дослідити вбудовування зображень, що захищаються не тільки безпосередньо в пікселі зображення, тобто вбудовування в просторовій області, але і в трансформанти двовимірних ортогональних перетворень, тобто вбудовування в частотній області, з метою збільшення об’єму інформації, що вмонтовується в зображення-контейнер.
5. Саме ці питання і розглядаються в роботі.

2 РОЗРОБКА МЕТОДІВ І АЛГОРИТМІВ ДЛЯ ЗАХИСТУ БІОМЕДИЧНИХ ЗОБРАЖЕНЬ

2.1 Розробка загальної схеми захисту

Розглядаються два варіанти вирішення задачі захисту біомедичних зображень:

- традиційний – шифрування файлів зображень;
- врахування специфіки зображення – будь-які перетворення зображення, які виключають відтворення початкового зображення без спеціальних засобів. Після перетворення зображення зберігається у початковому форматі.

Другий підхід має переваги, оскільки дозволяє реалізувати специфічні алгоритми перетворення простіші в реалізації, а збереження зображення у тому ж форматі приховує факт перетворення – ущільнення, шифрування та інше.

Загальна схема системи захисту наведена на рис. 2.1.

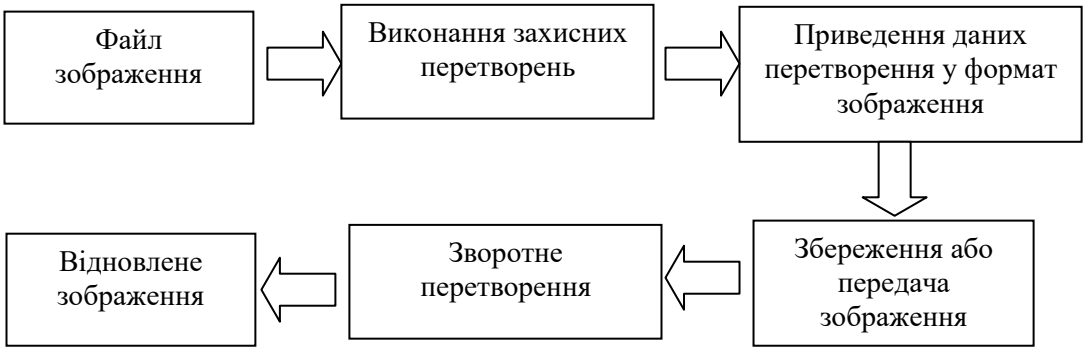


Рисунок 2.1 – Загальна схема захисту зображень від несанкціонованого доступу

Тобто вхідними і вихідними даними є файли у форматі зображення. В якості перетворень можуть застосовуватись криптографічні, стеганографічні або інші перетворення в площині зображення, а не з файлами.

2.2 Розробка криптографічного методу захисту зображень

Захист зображень виконується за рахунок перемішування пікселів зображення з використанням генератора псевдовипадкових чисел (ПВЧ), а також накладенні отриманої гами на відкриті дані (наприклад, при використанні логічної операції «виключне або») [9].

Оскільки кожний піксель кольорового зображення представлений 3 складовими кольору R (червоний), G (зелений), B (синій) до яких є доступ програмним способом, то перемішування будемо вести по кожній складовій кольору. Крім того, зображення це двовимірний масив, тому по кожній координаті буде використовуватись свій генератор ПВЧ. Тобто будемо використовувати 6 різних генераторів ПВЧ.

В даний час найбільш доступні й ефективними є конгруентні генератори ПВЧ. Для цього класу генераторів ПВЧ можна зробити математично строгий висновок про те, якими властивостями володіють вихідні сигнали цих генераторів з погляду періодичності. Одним з конгруентних генераторів є лінійний конгруентний датчик ПВЧ. Він формує послідовності псевдовипадкових чисел T(i) у відповідності з співвідношенням:

$$T(i+1) = (A*T(i)+C) \bmod M \qquad (2.1)$$

де T(0) – початкова величина, обрана як твірне число;

A і C – константи.

Такий датчик ПВЧ генерує псевдовипадкові числа з визначеним періодом повторення, що залежить від обраних значень A и C. Лінійний конгруентний генератор має максимальну довжину $M=2^n$ тільки тоді, коли C - непарне; A mod 4 = 1. Значення T(0) будемо вводити як ключ шифру.

Якщо вибрати M більшим кількості стовпців у зображенні, то з’являється можливість записати значення пікселів у нові позиції.

При використанні перемішування та гамування принцип шифрування полягає в генерації гами шифру за допомогою датчика псевдовипадкових чисел (ПВЧ), яка надає нові позиції кожного байта складових зображення і

накладенні отриманої гами на відкриті дані (наприклад, при використанні логічної операції «виключне або»). Процес дешифрування даних зводиться до повторної генерації гами шифру при відомому ключі, відновлення позиції пікселів зображення і накладенні такої гами на зашифровані дані.

При розробці алгоритму шифрування основними моментами є:

- вибір довжини ключа, тобто розрядності T(0);
- вибір значення M;
- вибір параметрів A і C.

Оскільки M задає період генератора ПВЧ, то для забезпечення високої крипостійкості воно повинно бути максимально можливим. З іншого боку – M може бути константою і змінною. Розглянемо обидва варіанти.

При виборі M як константи зменшується кількість можливих варіантів генерації послідовності, що в свою чергу зменшує криптостійкість алгоритму шифрування. Крім того розмір M при виборі будь-якого значення може бути меншим від розміру вхідного файлу, що призводить до генерації повторної послідовності в межах одного файлу. Тому більші переваги має вибір M як змінної.

Оскільки шифруються файли, розмір яких може бути прочитаний, то при шифруванні кожного файлу можна було б брати M рівним розміру файлу. Але з урахуванням того, що $M=2^n$, вибір будемо виконувати так:

- читається розмір файлу, що шифрується;
- вибирається найближче $M= 2^n$ більше або рівне розміру файлу.

Це забезпечує достатній період генератора ПВЧ, тобто період гами шифру перевищує розмір файлу, що забезпечує високу криптостійкість.

Вибір розміру ключа також пов’язаний з крипостійкістю алгоритму. Чим більший розмір ключа шифру тим вища криптостійкість. Раніше вважалось, що достатнім є 64-бітний ключ. Однак, зростання швидкодії обчислювальних засобів ставить під сумнів ці висновки.

Розрахуємо крипстостійкість для 128-бітного ключа. Критерієм криптостійкості є час необхідний для дешифрування файлів прямим перебором ключів, причому, не враховується обчислювальна складність алгоритму дешифрування і час необхідний для виконання алгоритму дешифрування при заданому ключі вважається рівним часу виконання однієї інструкції процесора типу регістр-регістр. Таким чином, для дешифрування шляхом перебору всіх ключів необхідна кількість часу визначається так:

$$Time = t*2^k \quad (2.2)$$

Де t – час виконання однієї інструкції процесора; k – розрядність ключа.

Якщо прийняти t = 1 нс, отримаємо:

$$Time=(1*2^{128})/(10^9*60*60*24*365) =$$

$$10790283070806014188970,52915499(років)$$

Тобто 128-бітний ключ повністю задовольняє вимогам до крипостійкості на багато років вперед. Сумнівно, що в майбутньому електронні обчислювальні засоби зможуть досягнути швидкодії достатньої для дешифрування даного алгоритму прямим перебором ключів.

Для вибору параметрів A і C генератора ПВЧ розглянемо три варіанти:

- A і C – константи для всіх файлів, що шифруються;
- A і C різні для кожного окремого файлу, що шифрується;
- A і C змінюються як від файлу до файлу, так і процесі шифрування.

З точки зору підвищення криптостійкості найбільш прийнятним є третій варіант або другий варіант. При цьому, параметри A і C можуть бути прив’язані до ключа шифру. Наприклад, можна попередньо генерувати їх використовуючи два інших генератори ПВЧ з T(0) рівним ключу шифру. Якщо вони обчислюються лише один раз перед шифруванням, то отримаємо другий варіант, а якщо генерувати їх в процесі шифрування паралельно з формуванням вихідної послідовності основного генератора, то отримаємо третій варіант.

Однак, враховуючи, що ключ шифру вибраний в 128 біт, а також те що період генератора більший розміру файлу, що шифрується, прийнятним є перший варіант. Це підтверджується і розрахунком криптостійкості наведеним вище. Тому в якості A і C вибираємо константи, що задовольняють умовам забезпечення максимального періоду генератора ПВЧ. Оскільки зображення двовимірний масив, значення пікселя містить три складові, параметри генераторів такі:

- для складової R зображення A=13, C=11, M рівне найближчому $M=2^n >$ кількості пікселів в зображенні.

- для складової G зображення A=25, C=19, M рівне найближчому $M=2^n >$ кількості пікселів в зображенні.

- для складової B зображення A=29, C=29, M рівне найближчому $M=2^n >$ кількості пікселів в зображенні.

Застосування 3-х генераторів ПВЧ підвищує криптостійкість як мінімум у 2 рази у порівнянні з одним.

При шифруванні поточне значення пікселя записується в позицію, що визначається виходами відповідних генераторів ПВЧ, а при дешифруванні навпаки значення пікселя в позиції. що визначається виходами відповідних генераторів ПВЧ записується в поточну позицію. Оскільки генератор ПВЧ генерує числа від 0 до M і $M=2^n >$ Width*Height (Width, Height – ширина і висота зображення), то може виникнути ситуація, що двовимірні координати в площині зображення перевищують допустимі значення. Тому такі випадки необхідно пропускати і виконувати генерацію псевдовипадкових чисел до тих пір поки не отримаємо прийнятний результат.

Для перетворення виходу генератора ПВЧ в двовимірні координати необхідно виконати такі обчислення:

- координата y=T div Width, де div – цілочислове ділення;
- координата x= T-y* Width.

Блок-схема алгоритму шифрування для однієї складової кольору приведена на рис. 2.2.

2.3 Розробка стеганографічних методів захисту зображень

Стеганографія (пер. з грец, «тайнопис») — це наука про приховану передачу інформації за допомогою збереження в таємниці самого факту передачі [10]. Для стеганографії важливим є вибір контейнера для приховування повідомлення. Найбільшу місткість забезпечують контейнери у вигляді файлів зображень, у яких можна замінити в кожному пікселесі по крайній мірі 1 біт по кожній складовій кольору на біт повідомлення, тобто у пікселі можна приховати мінімум 3 біти інформації. А якщо приховувати по 2 біти на кожну складову кольору, то можна приховати 6 біт у пікселі, з 24 біт на пік сель. Тобто розмір файлу з прихованим зображенням повинен бути в 4 рази більшим зображення, що приховується, що часто є не прийнятним.

Тому пропонується приховувати важливу частину інформації в тому ж самому зображенні без збільшення його розміру. Метод приховування такий:

- виконуються дії аналогічні попередньому алгоритму, але перемішуються і гамуються лише молодші 4-5 біт по кожній складовій кольору.

При відновлені зображення ці біти знову розміщуються за початковими позиціями.

Однак, у цьому підході є важливий недолік в плані захисту зображень від несанкціонованого доступу. Зображення спотворюється, що може виликати інтерес зловмисник до причин спотворення, тобто порушується головний принцип стеганографії - збереження в таємниці самого факту передачі. Тому розглянемо чисто стеганографічні методи приховування одного зображення в іншому з урахуванням специфіки рентгенівських зображень.

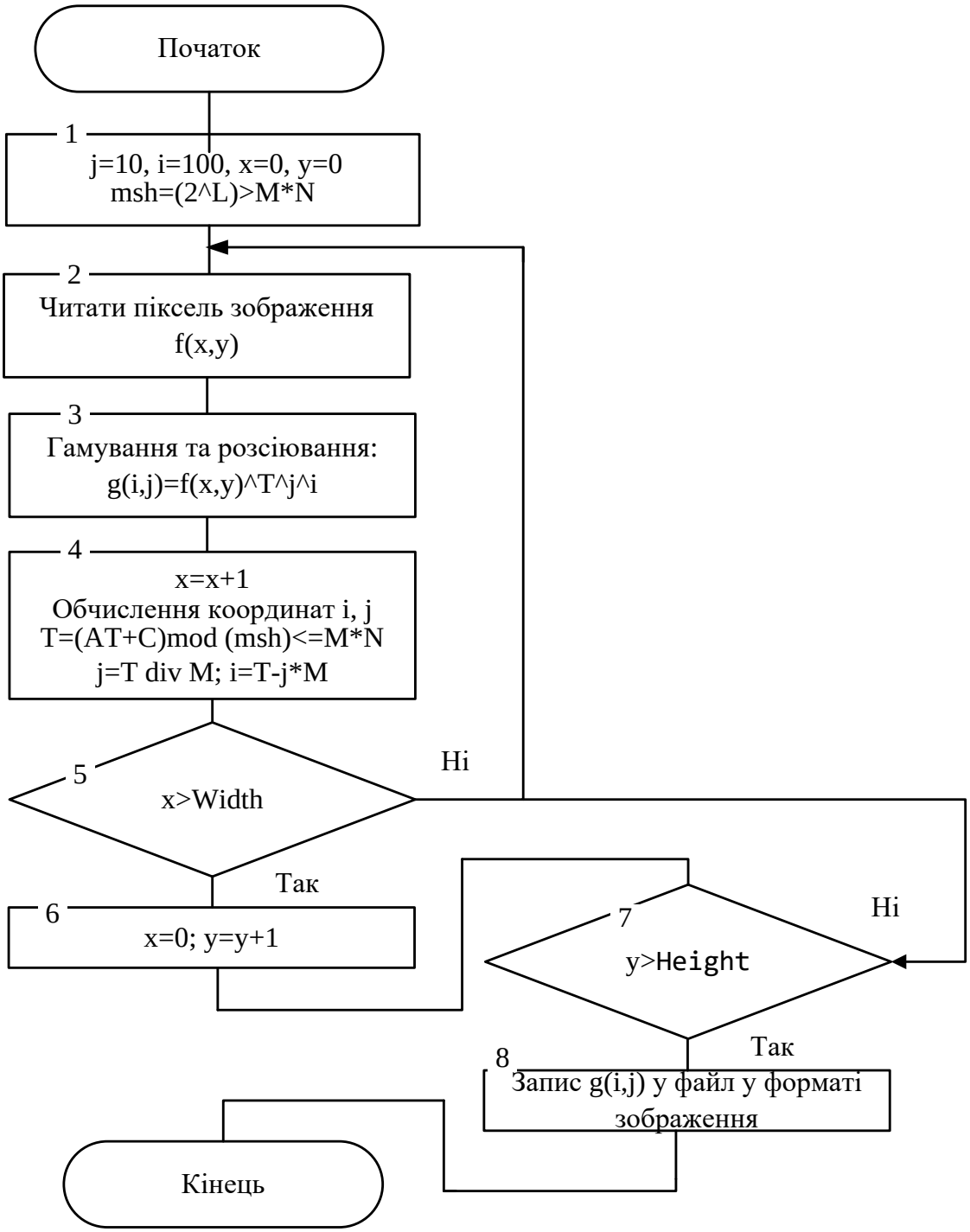


Рисунок 2.2 - Блок-схема алгоритму шифрування зображень

Пропонуються два методи приховування одного зображення в іншому:

- приховування в просторовій області, тобто безпосередньо в пікселях зображення;

- приховування в частотній області, тобто в коефіцієнтах деякого перетворення, які отримують із зображення. Наприклад, перетворення Фур'є, Уолша-Адамара, дискретне косинусне перетворення або інше.

Приховування в частотній області вимагає значно більше обчислень, але забезпечує збільшення ємності контейнера [11]. Однак, значення цього недоліку можна значно зменшити, якщо використовувати, наприклад, перетворення Уолша-Адамара для обчислення якого використовуються лише операції додавання та віднімання.

Пряме і зворотне перетворення, як правило, визначається з однаковою константою нормування так [9]:

$$[F(u,v)] = \frac{1}{N}[H(u,v)][f(x,y)][H(u,v)], \qquad (2.2)$$

$$f(x,y) = \frac{1}{N}[H(u,v)][F(u,v)][H(u,v)],$$

де H(u, v) – матриця Адамара, упорядкована за Уолшом;

f(x,y) – значення пікекселя початкового зображення;

F(u,v) – коефіцієнти (трансформанти) перетворення Уолша-Адамара.

Швидке обчислення перетворення Уолша-Адамара у двовимірному випадку зручно виконувати у два етапи:

1. Перший етап: Виконується одномірне перетворення кожного рядка (або стовпця) фрагмента розміром 8x8 елементів. Отримані коефіцієнти зберігаються в пам'яті.
2. Другий етап: Після обробки всіх рядків фрагмента виконується одномірне перетворення кожного стовпця (або рядка) масиву коефіцієнтів, отриманих на першому етапі.

Загальна кількість операцій для перетворення масиву розміром N×N при цьому становить:

$$m_0 = 2N^2 (N-1).$$

Без застосування швидкого алгоритму обчислення перетворення Уолша-Адамара двовимірне перетворення виконується шляхом прямого перемноження вихідного масиву на матрицю базисних функцій. Це передбачає виконання великої кількості операцій через повне матричне множення. Для масиву розміру N×N:

$$m = N^2 (N^2-1).$$

При N=8, $m/m_0 = 4{,}5$.

Тобто, виконання двовимірного перетворення через два послідовних одномірних дозволяє суттєво зменшити кількість арифметичних операцій.

Матриця Адамара розмірності 8×8 елементів для одновимірного перетворення упорядкована за Уолшом має такий вигляд:

$$H = \begin{vmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \end{vmatrix}$$

Саме ця матриця буде використана для виконання перетворення Уолша-Адамара фрагментів розміром 8×8. Процедура передбачає застосування одномірного перетворення до кожного рядка даних, а потім застосування одномірного перетворення до кожного стовпця стовпця даних, отриманих після першого етапу. Пряме і зворотне перетворення симетричні. Різниця тільки в тому, що пряме перетворення передбачає квантування трансформант, тобто цілочисельне ділення кожного значення на відповідний коефіцієнт квантування, зворотне перетворення включає деквантування, коли кожне значення трансформанти множиться на відповідний коефіцієнт квантування. Ця симетрія та чітка структура алгоритму забезпечують його ефективність у

багатьох задачах, зокрема в обробці зображень, стисканні даних і спектральному аналізі.

Незалежно від типу даних (просторові або частотні) контейнера алгоритм приховування однаковий в обох випадках.

Рентгенівські знімки це зображення з вузьким динамічним діапазоном, з наявністю ділянок з плавною зміною яскравості, тому слід очікувати, що кількість прихованих біт в кожному пікселі можна збільшити без втрати візуальної якості зображення і виявлення факту приховування іншого зображення в зображенні-контейнері.

Для проведення досліджень в якості контейнера використовується зображення типу рентгенівський знімок, приховується зображення того ж типу.

Алгоритм захисту включає такі кроки:

1. Розсіювання бітів зображення, що захищається в площині зображення контейнера (або площині коефіцієнтів перетворення) з використанням конгруентного генератора псевдовипадкових чисел (ПВЧ) та їх гамування. Для кожної складової кольору (RGB) використовується свій генератор ПВЧ. Він формує послідовності псевдовипадкових чисел T(i) у відповідності з співвідношенням [12]:

$$T(i+1) = (A*T(i)+C) \bmod M \qquad (1)$$

де T(0) – початкова величина, обрана як твірне число; A і C – константи.

Такий датчик ПВЧ генерує псевдовипадкові числа з визначеним періодом повторення, що залежить від обраних значень A и C. Лінійний конгруентний генератор має максимальну довжину $M=2^n$ тільки тоді, коли C - непарне; A mod 4 = 1. Значення T(0), A, C можуть бути ключем шифру. A в якості значення M вибирається найближче число кратне «2» більше кількості пікселів в зображенні-контейнері, що підвищує криптостійкість шифрування.

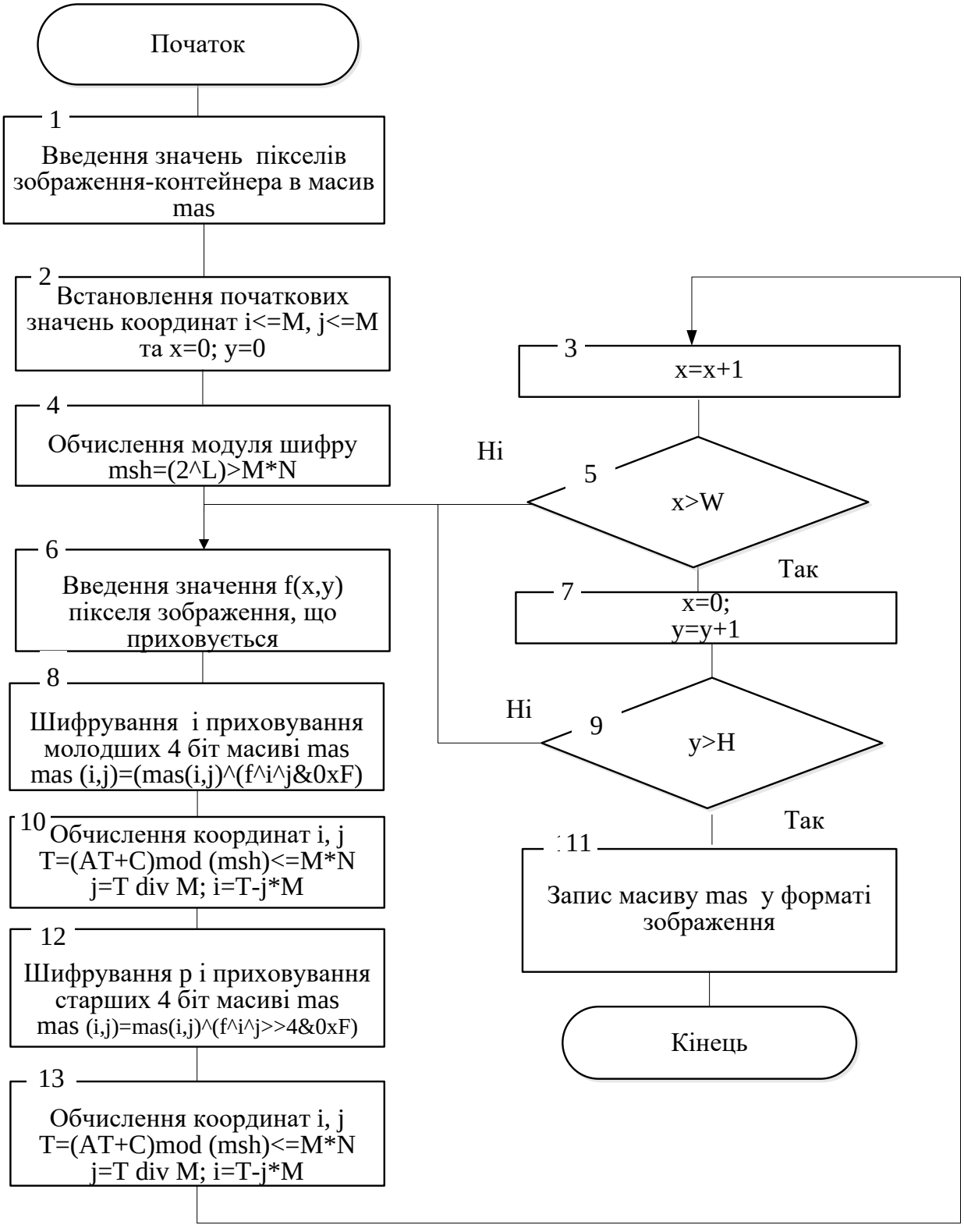


Рисунок 2.3 – Блок-схема алгоритму приховування в просторовій області

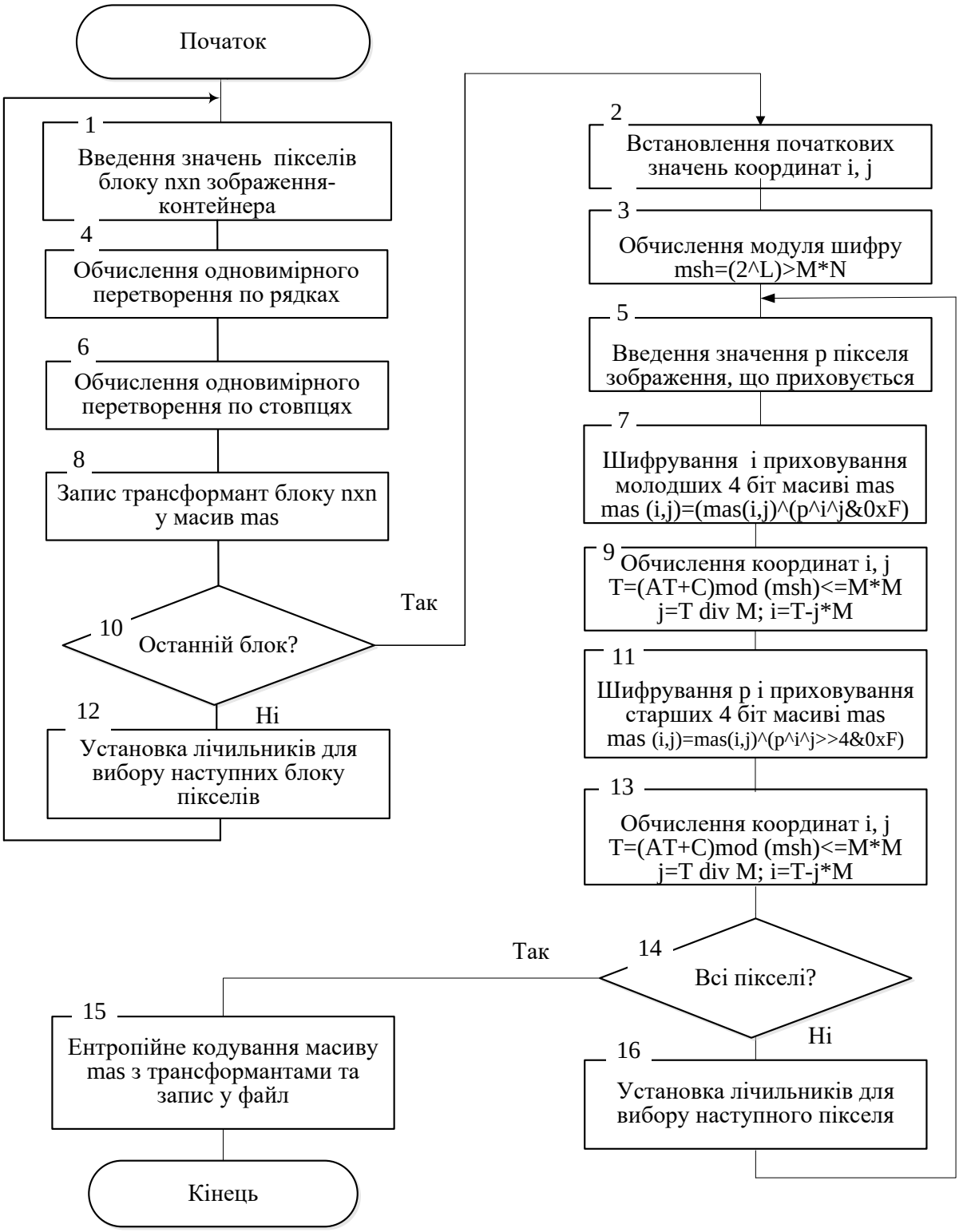


Рисунок 2.4 – Загальна блок-схема приховування в коефіцієнтах перетворення Уолша-Адамара

2. В кожному пікселі (або коефіцієнтах перетворення Уолша Адамара для складових кольору) контейнера приховується 12 біт зображення що захищається Для приховування використовуються 4 молодших біти в кожній складовій кольору. Причому дані, що приховуються в поточному пікселі контейнера належать різним пікселям зображення, що захищається.

Блок-схема алгоритму приховування для однієї складової кольору просторовій області приведена на рис. 2.3, а в частотній на рис. 2.4.

2.4 Оцінка складності стеганографічного методу захисту зображень

У програмуванні, обчислювальну складність алгоритмів зазвичай оцінюють за кількістю дій, які виконує алгоритм та за обсягом задіяної пам’яті. У переважній більшості випадків, для позначення оцінки складності алгоритмів використовують так звану О-нотацію; в математиці таке позначення застосовують для порівняння асимптотичної поведінки функцій/ О-нотація визначає функцію (назвемо її g(n)), яка показує, як буде змінюватися обчислювальна складність алгоритму зі зміною кількості вхідних даних у найгіршому для алгоритму випадку [24-26].

Знайдемо оцінку складності для стандарту шифрування DES. В цьому стандарті до блоку, що підлягає шифруванню, застосовується початкова перестановка (IP), потім складна обчислювальна процедура в залежності від ключа, а в кінці обернена перестановка (IP^{-1}). Обчислювальна процедура виконується з використанням мережі Фейстеля, що містить 16 каскадів.

На кожному етапі виконується зчитування блоку з пам’яті, операція додавання за модулем «2» і запис в пам’ять. У сучасних процессорах найбільш повільною є операції читання-запису оперативного запам'ятовуючого пристрою (ОЗП), оскільки внутрішні операції в процесорі виконуються на частотах 2 ГГц або більших на декількох конвеєрах, а запис-читання пам’яті на частотах лише 200-300 МГц, тому оцінимо кількість звертань до ОЗП. Будемо вважати, що процессор може читати і писати пам'ять блоками по n біт за один машинний

цикл читання, а також не будемо враховувати складність генерації ключів шифрування для кожного етапу.

Тобі в алгоритмі DES початкова перестановка вимагає як мінімум 2 операції – читати і писати. Якщо кількість каскадів мережі Фейстеля k, то на кожному каскаді також потрібно виконати 2 операції – читати і писати. І нарешті обернена перестановка також 2 операції. Таким чином:

$$g_1(k)=2k+4$$

Тобто, будемо вважати, що складність алгоритму DES O(2k+4) лінійно залежить лише від кількості каскадів мережі Фейстеля.

Оцінимо складність стеганографічних методів при тих же початкових умовах. Читання n біт зображення, що приховується вимагає 1 операції читання. Якщо в кожному пікселі приховується m біт, де m<r, де r – кількість біт на піксель, то потрібно 2*(n/(m*r)) операцій читання-запису. Тоді:

$$g_2(m)=1+2*(n/(m*r))$$

Наприклад, якщо k=16, то $g_1(k)=36$. А для $g_2(m)$ при n=64, r=8, m=1(найгірший випадок) маємо:

$$g_2(m)=1+2*(64/(1*8))=17.$$

Тобто запропонований метод забезпечує меншу складність алгоритму і відповідно вищу швидкість роботи. Залежність $g_2(m)$ наведена на рис. 2.5.

Рисунок 2.5 – Залежність складності алгоритму від кількості біт, що приховуються в одному пікселі

2.5 Висновки

1. Запропоновано методи захисту біомедичних зображень, відмінністю яких є виконання операцій захисту не з файлами, а у площині зображення з даними самих зображень.

2. Розроблено метод та алгоритм захисту біомедичних зображень, у якому позиція пікселя по кожній складовій кольору окремо та по кожній координаті окремо визначається з використанням генератора ПВЧ.

3. Розроблено метод приховування важливої інформації, що міститься в зображенні у цьому ж зображенні за рахунок перемішування і гамування молодших 4-5 розрядів кожної складової пікселя.

4. Розроблено метод приховування біомедичних зображень в іншому зображені контейнері.

5. Розроблено метод приховування біомедичних зображень в коефіцієнтах перетворення Уолша-Адамара, які отримані із зображення-контейнера.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ЗАХИСТУ БІОМЕДИЧНИХ ЗОБРАЖЕНЬ

3.1 Аналіз та вибір середовища і мови програмування

Серед мов програмування під Windows найбільш широко застосовуються мови C++ та C# [27-28]. Порівняємо ці мови.

Швидкість розробки. В коротких малобюджетних проектах C # матиме перевагу в швидкості розробки, але в довгих і відносно дорогих проектах дана перевага буде незначною.

Кросплатформеність. C++ кросплатформний за фактом, хоча і з деякими застереженнями, додатковими витратами, а також бінарною несумісністю між платформами. C#, за фактом, виявився не кросплатформним, незважаючи на існування неофіційних .net оточень під різними платформами і навіть потенційну бінарну сумісність між платформами. C # спроектований як кросплатформний, однак його розвиток не пішов в цьому напрямку. Під Windows утворилася досить повна .net інфраструктура, а на інших платформах рівноцінної інфраструктури не з'явилося. При цьому для розробки на C++ склалася практично рівноцінна інфраструктура на більшості існуючих платформ, є маса бібліотек, які скомпільовані або можуть бути скомпільовані під будь-які існуючі платформи. Існує величезна кількість міжплатформних додатків і бібліотек різного масштабу, написаних мовою C++, наряду з кросплатформними бібліотеками є і бібліотеки специфічні для окремих платформ. Все це дає практичний рівноцінні шанси для розвитку додатків на різних платформах.

Бібліотеки. Відмінність асортименту C++ і C# бібліотек в тому, що C++ бібліотек більше, вони мають велику історію, за яку стали непогано налагоджені і оптимізовані, часто кросплатформні, багато з відкритим кодом.

Однак при всіх позитивних сторонах C++ бібліотеки як мають дуже різну, часто навіть архаїчну архітектуру, часто не об'єктний, а структурно-процедурний інтерфейс. Пов'язано це з тим, що багато C++ бібліотеки це C бібліотеки. У C# перерахованих вище проблем значно менше. В силу того, що бібліотеки C# порівняно молоді, - інтерфейси бібліотек, як правило, краще вписуються в ті чи інші шаблони проектування, що спрощує їх вивчення. Однак, при найближчому розгляді великий шанс, що під вашу специфічне завдання C# бібліотеки відсутні, більш того, може виявитися, що і вирішувати таке завдання на C# досить неефективно. Друга неприємна особливість бібліотек C# в тому, що багато з них є просто обгорткою над unmanaged бібліотеками, що буде завжди приводити до втрат продуктивності на конверсіях типів, і створювати додаткові проблеми налагодження та поширення.

Зручність налагодження. Можна було б просто сказати, що під Window, C# помітно зручніше налагоджувати і на цьому зупинитися. Однак якщо з якоїсь причини у вас на ряду з mananged кодом з C # збірки використовується unmanaged, то його налагодження стане більш складним у порівнянні зі звичайним налагодженням unmamanged коду з C ++.

Продуктивність коду і вимогливість до ресурсів. Очевидним є факт того, що можливості по оптимізації некерованого (unmanaged) коду значно ширші, ніж можливості по оптимізації керованого (managed) коду. Пікова продуктивність коду досяжна тільки в unmanaged виконанні, тобто будь-яке завдання мовою C++ може бути вирішена з меншими вимогами до ресурсів. Тому в важких завданнях, пов'язаних з обробкою великої кількості даних, C++ має сильні переваги перед C#. Фундаментальні основи переваг C++ в можливості писати код, який буде виконуватися безпосередньо процесором, і можливості прямої роботи з пам'яттю.

Мова і синтаксис. З першого погляду код C++ і C# дуже схожі зовні. Але різноманіття коду на C++ більше, адже C++ є одночасно і C і C++. І все це може використовуватись одночасно (звичайно, якщо це підтримує компілятор).

Вартість підтримки. У підтримці додатків великої різниці між C++ і C# немає.

Ризики. Основний ризик використання C# - це сильна зав'язка на Microsoft. Навряд чи Microsoft кудись зникне в найближчому майбутньому, але варто розуміти, що Microsoft - це комерційна організація, метою якої є отримання прибутку, а для прибутку потрібні продажі своєї продукції. Тому в інтересах Microsoft розгортати розробку C# і .net так, щоб це приводило до продажу нової продукції Microsoft. Якщо інтереси вашої розробки будуть не відповідати інтересам Microsoft, рано чи пізно це може призвести до проблем.

Самодостатність додатків. Повної самодостатності додатків немає ні у C++ ні у C#. Однак хотілося б відзначити, що рантайм C++, як і будь-яка інша бібліотека, може статично лінкуватися у виконуваний модуль, таким чином виконуваний модуль може містити все необхідне для роботи, і за рахунок цього стане самодостатнім, в разі C# стандартними методами таке не піддається реалізації.

Враховуючи вище сказане для задач обробки зображень перевагу має мова програмування C++, яка і буде використовуватись в розробці.

Щодо середовища розробки, то доцільно було б порівняти два середовища Microsoft Visual Studio та Borland C++ Builder. Тут можна привести два простих аргументи на користь MS Visual Studio:

1. Visual C ++ - найкраще середовище розробки для Win32. Хто може написати краще середовище розробки додатків під деяку ОС? Тільки сам розробник ОС. Такий і є Visual C ++.

2. Borland не зможе створити хороше середовище розробки під Windows, оскільки Borland не розробник цієї ОС.

Тому в якості середовища розробки обрано середовище Microsoft Visual Studio.

3.2 Розробка архітектури та інтерфейсу користувача додатку

Будемо використовувати проект Windows Forms Application [29]. У відповідності з визначеним набором функцій програми, що розробляється головне вікно програми повинно включати поля для виведення початкового і обробленого зображення, кнопки для подачі команд відкриття початкового файлу, збереження обробленого файлу, подачі команди виведення інформації про програму та інші. Крім того, необідно враховувати експериментальний характер роботи.

З урахуванням цього і розроблена головна форма програми, яка наведена на рис. 3.1 і реалізує концепцію одновіконної програми, що є максимально зручним для користувача.

Використовуються такі керуючі компоненти (System.Windows.Forms) Visual Studio:

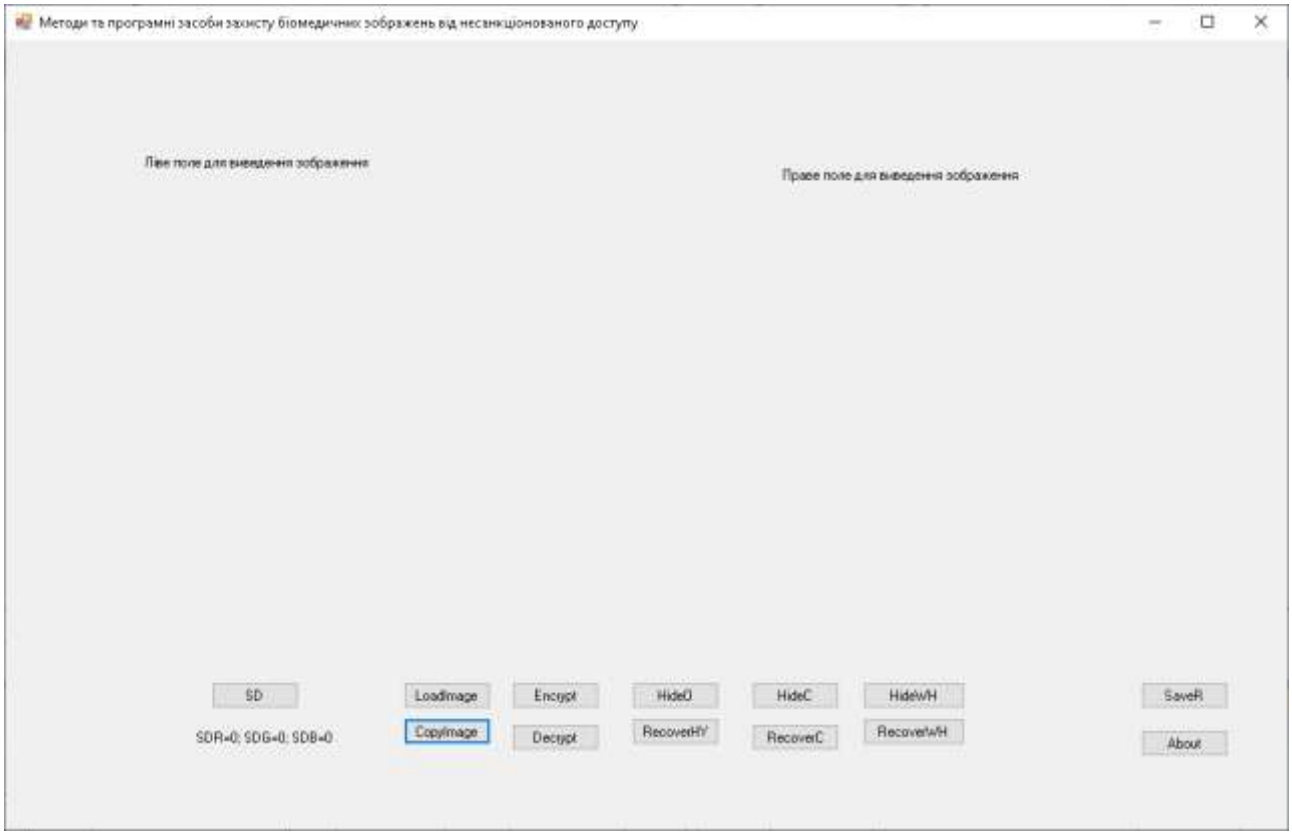


Рисунок 3.1 – Головна форма програми

1. Форма Windows Form – головна форма додатку, на якій розміщуються інші компоненти.
2. Дві форми PictureBox, які призначені для завантаження вхідного зображення та виведення зображення після перетворень.
3. Дев’ять форм Button для подачі команд.
4. Вікно About (про програму) – містить такі форми Windows:
   - Form;
   - форму Label – стандартні повідомлення про ім’я продукту, версію, права та ім’я компанії;
   - форма Button для закриття вікна About.

. Призначення основних класів таке:

- клас About, відповідає за виведення повідомлення Про програму;
- клас Form1 - методи цього класу відповідають за роботу з вікнами, інтерфейс користувача, виконують виклики методів захисту біомедичних зображень при необхідності.

За виконання основних функцій програми відповідають наступні методи класів наведених вище:

1. Метод класу Form1 Form1_Load () – формує вікно відкриття програми.

2. Метод класу Form1 button8_Click () - відкриття зображення і виведення його на форму PictureBox1 (ліва), призначену для вхідного зображення. Викликається при виборі команди LoadImage (рис. 3.1).

3. Метод класу Form1 button2_Click () – копіює зображення з PictureBox1 в PictureBox2. Викликається при виборі команди CopyImage (рис. 3.1):

```
pictureBox2->Image = pictureBox1->Image;
```

4. Метод класу Form1 button7_Click () – зберігає перетворене зображення в форматі .BMP.

5. Метод класу Form1 button6_Click () – виводить повідомлення «Про програму». Викликається при виборі команди About:

6. Методи класу Form1 button12_Click () виконує обчислення середньоквадратичного відхилення декодованого зображення від початкового зображення.

7. Методи класу Form1 button26_Click () виконує криптографічне перетворення зображення.

8. Методи класу Form1 button27_Click () виконує зворотне криптографічне перетворення зображення.

9. Методи класу Form1 button27_Click () виконує приховування важливої частини інформації, що міститься у зображенні у тому ж зображенні.

10. Методи класу Form1 button28_Click () виконує відновлення приховуваної інформації.

Текст програми наведений в додатку А і містить необхідні для розуміння роботи коментарії.

3.3 Розробка модулів шифрування-дешифрування

Вхідними даними для модуля шифрування (метод button26_Click) є файли зображень у форматах, які підтримує Windows. При відкритті файлу відкривається стандартне вікно вибору файлу і дані зображення присвоюються структурі даних image типу Bitmap:

```
if (openFileDialog1->ShowDialog() ==
System::Windows::Forms::DialogResult::OK)
    {
        image1 = gcnew Bitmap(openFileDialog1->FileName, true);
        image2 = gcnew Bitmap(openFileDialog1->FileName, true);
    }
```

Далі визначається модуль шифру. Це число кратне степені 2 більше кількості пікселів в зображенні:

```
mod = 2;
while (mod  < (image1->Width * image1->Height)) mod = mod * 2;
```

Це забезпечує формування генератором ПВЧ послідовності довжина якої перевищує довжину повідомлення, що забезпечує практично необмежену криптостійскість.

Шифрування виконується шляхом розсіювання даних зображення у площині зображення з використанням генераторів ПВЧ та об'єднанні їх з гамою шифру, що формується цими генераторами з використанням операції додавання за модулем два (xor). Для кожної складової кольору використовується свій генератор ПВЧ. В подвійному циклі дані зображення послідовно зчитуються з використанням методу GetPixel():

```
pixelColor = image1->GetPixel(x, y);
```

Далі записуються у нову позицію в масиви gwhr[j][i] (складова кольору R), gwhg[l][k] (складова кольору G), gwhb[n][t] (складова кольору B). Координати j, I, l, k, n, t формуються трьома генераторами ПВЧ. Для складової кольору R фрагмент програми такий:

```
gwhr[j][i] = pixelColor.R;
gwhr[j][i] = (gwhr[j][i] ^ jt ^ lt ^ nt ^ it ^ kt ^ tt)&0x00ff;
```

Далі формуються координати для запису чергового значення. Але тут необхідно мати на увазі, що значнення змінної mod більше пікселів в зображенні, тому можуть бути ситуації, коли сформовані генератором ПВЧ координати виходять за межі зображення. У цьому випадку потрібно пропускати цикли генератора ПВЧ, у яких формуються значення більші кількості пікселів у зображенні:

```
jt = (13 * jt + 11) % mod;
while (jt >= (image1->Width * image1->Height)) jt = (13 * jt + 11) % mod;
```

Цикл while якраз і реалізує це. Якщо значення змінної jt більше добутку ширини на висоту зображення, то формується нове значення jt.

Для отримання нових координат потрібно значення змінної jt перевести у формат рядок-стовпець. Для

```
i = (jt / image1->Width);  //рядок
j = jt - i * image1->Width; //стовпець
```

Для кожної складової кольору виконуються аналогічні дії. В результаті складові кольору RGB поточного пікселя мають різні координати, а також операції xor виконувалась з різними виходами генераторів ПВЧ.

Після завершення шифрування масиви gwhr, gwhg, gwhb в циклі записуються в структуру image2 з використанням методів FromArgb(), SetPixel() з перевіркою на переповнення:

```
t = gwhr[x][y]; if (t < 0)  t = 0; if (t > 255)  t = 255; r = t;
t = gwhg[x][y]; if (t < 0)  t = 0; if (t > 255)  t = 255; g = t;
t = gwhb[x][y]; if (t < 0)  t = 0; if (t > 255)  t = 255;b = t;
newColor1 = Color::FromArgb(r, g, b);
image2->SetPixel(x, y, newColor1);
```

І виводяться в праве поле для виведення зображень:

```
pictureBox2->Width = image2->Width;
pictureBox2->Height = image2->Height;
pictureBox2->Image = image2;
```

Збереження зображення виконує метод button7_Click (), який викликається командою SaveD.

Дешифрування виконує метод button27_Click (). Вхідними даними є зображення у форматах, що підтримуються Windows. Виконуються аналогічні операції, але дані зчитуються з площини зображення за координатами j, i, l, k, n, t, що формуються генераторами ПВЧ, а записуються в масиви gwhr, gwhg, gwhb за координатами x, y, тобто повертаються на свої позиції.

3.4 Розробка модулів приховування молодших розрядів

Алгоритм приховування молодших розрядів ґрунтуються на тому, що якщо по крайній мірі поміняти позиції 4-5 молодших розрядів по кожній складовій кольору зашифрувавши їх з використанням опеації xor, то це неуможливить отримати достовірну інформацію про діагноз хворого. Цей

алгоритм реалізує метод button28_Click (), який запускається на виконання командою HideO.

Аналогічно попередньому модулю дані зображення спочатку записуються в структуру image1. Далі обчислюється модуль шифру і значення кожної складової кольору записуються в масиви:

```
Color pixelColor = image1->GetPixel(x, y);
        gwhr[x][y] = pixelColor.R ;
            gwhg[x][y] = pixelColor.G;
                gwhb[x][y] = pixelColor.B;
```

Процес приховування реалізується у подвійному циклі шляхом послідовного зчитування даних зображення із структури image1 за координатами x, y; виділенням старших трьох розрядів і додаванням молодших зашифрованих 5-розрядів пікселя, координати якого визначаються виходами генераторів ПВЧ і запису результату за координатами x, y:

```
Color pixelColor = image1->GetPixel(x, y);
gwhr[j][i] = (gwhr[j][i]&0xE0)|((pixelColor.R ^ jt ^ lt ^ nt ^ it ^ kt ^ tt) &0x1F);
gwhg[l][k] = (gwhg[l][k] & 0xE0) | ((pixelColor.G ^ jt ^ lt ^ nt ^ it ^ kt ^ tt) & 0x1F);
gwhb[n][t] = (gwhb[n][t] & 0xE0) | ((pixelColor.B ^ jt ^ lt ^ nt ^ it ^ kt ^ tt) & 0x1F);
```

Кількість і робота з генераторами ПВЧ виконується также як і в модулі шифрування. Кількість молодших розрядів, які приховуються і шифруються буде визначено в процесі досліджень.

При відновлені початкового зображення дані читаються з структури image1 за координатами, що формуються генераторами ПВЧ (метод button29_Click ()). Тобто, координатами j, i, l, k, n, t. Наприклад для складової кольору R:

```
Color pixelColor = image1->GetPixel(j, i);
```

Всі інші операції симетричні алгоритму приховування. Таким чином відновлюються значення і початкове положення молодших розрядів кожного пікселя.

3.5 Розробка модулів стеганографічного захисту біомедичних зображень в просторовій області

При виборі команди HideC (метод button30_Click()) виводиться діагностичне повідомлення з інформацією про необхідність обрати контейнер для приховування, а також вказівки щодо розміру контейнера:

MessageBox::Show("Виберіть контейнер. Розмір контейнера повинен бути у два рази більшим файлу, що приховується");

Далі відкривається стандартне вікно для вибору контейнера. Після вибору контейнера пропонується обрати файл зображення, який буде приховуватись в контейнері, і виводиться попередження про розмір цього файлу:

MessageBox::Show("Виберіть файл для приховування. Розмір файлу повинен бути у два рази меншим файлу контейнера");

Якщо розмір файлу, що приховується не відповідає вимогам, то виводиться повідомлення про необхідність обрання іншого контейнера або файлу що приховується і виконання модуля завершується:

```
if ((image2->Width * image2->Height) >= ((image1->Width * image1->Height) / 2))
    { MessageBox::Show("Виберіть інший контейнер або файл для
приховування. Розмір контейнера повинен бути у два рази більшим");
        return;}
```

Модуль шифру повинен бути кратним степені «2», але більшим кількості пікселів в зображенні. Початкове значення змінної mod, яка призначена для зберігання модуля шифру встановлюється рівним 2, а далі обчислюється шукане значення в циклі while^:

```
while (mod < (image1->Width * image1->Height)) mod = mod * 2;
```

Значення складових кольору пікселів зображення контейнера записуються в масиви gwhr для компоненти R, gwhg для компоненти G, gwhb для компоненти B.

Пікселі, зображення що приховуються послідовно читаються з структури image, розсіюються з використанням генератора ПВЧ по кожній складовій

кольору окремо в площині зображення контейнера по 4 біти на байт. Наприклад, для складової R для молодших 4 біт виконуються такі операції:

```
Color pixelColor = image2->GetPixel(x, y);
gwhr[j][i] = (gwhr[j][i] & 0xF0) | ((pixelColor.R ^ jt ^ lt ^ nt ^ it ^ kt ^ tt) & 0x0F);
```

Крім розсіювання тут виконується і гамування цих розрядів, тобто з використанням операції додавання за модулем 2 гама шифру об'єднується з відкритими даними. Зашифровані дані записуються за координатами, що формуються генераторами ПВЧ – для кожної складової кольору свій генератор ПВЧ.

Якщо генератор ПВЧ формує на виході значення, що перевищує кількість пікселів в зображенні, то запускається новий цикл генерації псевдовипадкового числа. Операція повторюється до тих пір доки jt не стане меншим кількості пікселів в контейнері:

```
jt = (13 * jt + 11) % mod;
while (jt >= (image1->Width * image1->Height)) jt = (13 * jt + 11) % mod;
```

Це необхідно, оскільки змінна jt використовується для формування нової позиції компонент піселя і якщо її значення буде перевищувати кількість пікселів в зображенні, то при спробі запису будут формуватись виключення по запису за межами масиву.

Координати пікселя в площині контейнера формуються із значення змінної jt так:

```
i = (jt / image1->Width);
j = jt - i * image1->Width;
```

Приховування старших 4 біт змінної pixelColor виконується в молодших 4 бітах пікселя контейнера за цими новими координатами:

```
gwhr[j][i] = (gwhr[j][i] & 0xF0) | (((pixelColor.R ^ jt ^ lt ^ nt ^ it ^ kt ^ tt)>>4) &
0x0F);
```

Для вилучення прихованого зображення з контейнера необхідно обрати команду RecoveryHY. Цю операцію виконує метод button31_Click(). Виконуються операції симетричні алгоритму приховування за виключенням

напрямку передачі даних та напряму зсуву при вилученні старших 4 біт пікселя прихованого зображення:

```
gwhr[x][y] = (gwhr[x][y] & 0x0F) | (((pixelColor.R << 4) ^ jt ^ lt ^ nt ^ it ^ kt ^ tt)  &
                                                    0x00F0);
```

3.6 Розробка модулів стеганографічного захисту біомедичних зображень в частотній області

Для приховування зображень в частотній області використаємо перетворення Уолша-Адамара для зображення-контейнера. В коефіцієнтах цього перетворення будемо приховувати зображення, що захищається, оскільки це перетворення вимагає найменше обчислень з відомих ортогональних перетворень. Приховування виконує метод button13_Click (), який викликається на виконання командою HideWH.

На початку роботи методу виконуються дії аналогічні методу приховування в просторовій області, тобто перевіряються на відповідність розміри контейнера і зображення, що приховується.

Далі виконується перетворення Уолша-Адамара для всіх сусідніх фрагментів зображення розміром 8x8.

Спочатку у вихідний файл записуються значення матриць коефіцієнтів квантування для кожної складової кольору, тобто 192 байти у послідовності RGB.

Далі у массив kv8[k][l] завантажуються значення коефіцієнтів квантування для складової кольору R:

```
        for (int k = 0; k < n; k++) {
                    for (int l = 0; l < n; l++)
                    {
kv8[k][l] = Convert::ToInt32(dataGridView2->Rows[k]->Cells[l]->Value);
                    }
                }
```

У масив tr[i][j] завантажуються значення чергового блоку компоненти кольору R зображення розміром 8x8:

```
for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++)
        {
                tr[i][j] = gwhr[8 * x + j][8 * y + i];
        }
}
```

Обчислюються значення коефіцієнтів перетворення для блоку 8x8 (в даному випадку для компоненти R кольорового зображення), виконується їх квантування і запис у вихідний файл. Причому, спочатку виконується одновимірне перетворення по рядках, а потім з отриманими даними одновимірне перетворення по стовпцях. Отримані дані кантуються шляхом ділення кожної трансформанти на свій коефіцієнт квантування, записуються у масив gwhr1:

```
for (k = 0; k < n; k++) {
        for (l = 0; l < n; l++) {

                tr[k][l] = tr[k][l] / kv8[k][l];
                gwhr1[8 * x + k][8 * y + l] = tr[k][l];

        }
        }
```

Аналогічні операції виконуються для блоків пікселів компонент G, B кольорового зображення і результати записуються в масиви gwhg1та gwhb1 відповідно.

У кожному коефіцієнті перетворення Уолша-Адамара приховується по 4 біти в кожній складовій кольору. Виконується шифрування кожних 4 біт початкового зображення в використанням операції «виключне або» і

розсіювання їх в площині коефіцієнтів перетворення. Для кожної складової кольору використовується свій генератор ПВЧ. Приховування складової R пікселя в коефіцієнтах перетворення виконується так:

- приховування молодших 4 біт пікселя:

gwhr1[jtt][itt] = (gwhr1[jtt][itt] & 0xFFF0) | ((gwhr2[x][y] ^ jt ^ lt ^ nt ^ it ^ kt ^ tt) & 0x000F);

gwhr1 – масив коефіцієнтів для складової кольору R;

gwhr2 – масив значень пікселів складової R зображення, що приховується;

- приховування старших 4 біт пікселя:

jt = (13 * jt + 11) % mod;

while (jt >= (image1->Width * image1->Height)) jt = (13 * jt + 11) % mod;

itt = (jt / image1->Width);

jtt = jt - itt * image1->Width;

gwhr1[jtt][itt] = (gwhr1[jtt][itt] & 0xFFF0) | (((gwhr2[x][y] ^ jt ^ lt ^ nt ^ it ^ kt ^ tt) >> 4) & 0x000F);

обчислюються координати для старших 4-х біт пікселя, а потім виконується запис за новими координатами.

Далі виконується зворотне перетворення Уолша-Адамара для фрагменту розміром 8x8 компоненти R і результати записуються в масив tr[k][l]. Алгоритм виконання перетворення аналогічний прямому.

Аналогічні обчислення виконуються для блоків пікселів компонент G, B кольорового зображення і результати записуються в масиви tg[k][l] та tb[k][l] відповідно. А потім відновлені відліки блоку 8x8 виводяться в структуру Image2 типу Bitmap:

```
t = tr[i][j]; if (t < 0)  t = 0; if (t > 255)  t = 255; r = t;
t = tg[i][j]; if (t < 0)  t = 0; if (t > 255)  t = 255; g = t;
t = tb[i][j]; if (t < 0)  t = 0; if (t > 255)  t = 255; b = t;
        newColor1 = Color::FromArgb(r, g, b);
    image2->SetPixel(8 * x+j, 8 * y+i, newColor1);
```

По завершенню перетворення відновлене зображення виводиться pictureBox2 і закривається вихідний файл з коефіцієнтами:

```
pictureBox2->Image = image2;
out_file.close();
```

Коефіцієнти перетворення ущільнюються арифметичним кодером (модуль dmc.exe):

```
if (!CreateProcess(NULL, (LPSTR)name2, NULL, NULL, FALSE, 0, NULL,
NULL, &si, &pi))
    {
        printf("CreateProcess failed (%d).\n", GetLastError());
        return;
        }
```

Метод класу Form1 button14_Click () – виконує вилучення прихованого зображення з коефіцієнтів перетворення, а потім обернене перетворення Уолша-Адамара при розмірах блоків 8x8 і виводить вилучене зображення в ліве поле, а декодоване в праве поле для виведення зображень. Викликається при виборі команди RecoverWH (рис. 3.1).

Виконується декодування ущільненого файлу арифметичним декодером і результат записується в файл коефіцієнтів перетворення:

```
if (!CreateProcess(NULL, (LPSTR)name2, NULL, NULL, FALSE, 0, NULL,
NULL, &si, &pi))
        {
            printf("CreateProcess failed (%d).\n", GetLastError());
            return;
        }
```

Далі з файлу коефіцієнтів перетворення зчитуються значення двовимірних матриць квантування, тобто 192 байти – 64 байт для R, 64 байт для G, 64 байт для B.

А потім з файлу коефіцієнтів перетворення зчитуються коефіцієнти для поточного блоку пікселів розміром 8x8 (спочатку коефіцієнти для компоненти

кольору R, записуються в масиви gwhr1 і tr[k][l] і виконується їх деквантування:

```
                for (k = 0; k < n; k++) {
            for (l = 0; l < n; l++) {
                in_file.read((char*)&tr[k][l], 2);
                gwhr1[8 * x + k][8 * y + l] = tr[k][l];
                tr[k][l] = tr[k][l] * kv8[k][l];
            }
        }
```

Аналогічні обчислення виконуються для компонент G і B, результати записуються в масиви gwhg1, tg[k][l] та gwhb1 ,tb[k][l].

Далі виконується зворотне перетворення Уолша-Адамара для фрагменту розміром 8x8 компонент R, G, B. А потім відновлені відліки блоку 8x8 виводяться в структуру Image2 типу Bitmap. Алгоритм виконання перетворення аналогічний прямому.

По завершенню перетворення відновлене зображення виводиться в pictureBox2 і закривається вхідний файл з коефіцієнтами:

```
pictureBox2->Image = image2;
in_file.close();
```

Після обробки всіх блоків 8x8 масиви gwhr1, gwhg1, gwhb1 містять коефіцієнти перетворення Уолша-Адамара по кожній складовій кольрору. Аналогічно алгоритму приховування в просторовій області з цих коефіцієнті вилучається приховане зображення і виводиться в структуру image3 і візуалізується в ліве поле для виведення зображень:

```
pictureBox1->Image = image3;
```

Текст коду медодів наведено в додатку В.

3.7 Висновки

1. Виконано аналіз середовищ розробки та мов програмування, який показав, що для цієї розробки доцільним є використання середовища розробки Microsoft Visual Studio та мови програмування C++, оскільки ці засоби забезпечують найкращі швидкісні характеристики для програм, що розробляються, а також мають розвинені сервісні функції, які пришвидшують розробку.

4. Розроблено програму для дослідження методів захисту біомедичних зображень від несанкціонованого доступу шляхом виконання криптографічних та стеганографічних перетворень.

4 ТЕСТУВАННЯ ДОДАТКУ

4.1 Керівництво користувача

Програмний продукт розроблений під платформу Windows. Комп’ютер повинен не менше 4 Гб оперативної пам’яті та 40 Мб вільного місця на жорсткому диску.

Представлена програма призначена для захисту біомедичних зображень від несанкціонованого доступу шляхом виконання криптографічних та стеганографічних перетворень розроблена в середовищі Visual Studio 2019 мовою програмування C++. Тому програма може працювати на комп’ютері з операційною системою Windows XP, Windows 7, Windows 10. Програма може знайти застосування в навчальному процесі при вивченні методів захисту біомедичних зображень.

Для використання програми потрібно завантажити папку ProtectBM на будь-який диск комп’ютера. Файловий склад програми такий:

- файл ProtectBM.exe – головний модуль програми;
- image1GK.bmp, image2Chr.bmp – технологічні зображення.

Щоб запустити програму на виконання потрібно вибрати файл ProtectBM.exe, що знаходиться у папці ProtectBM. Одразу після запуску програми з’явиться головне вікно програми (рис. 4.1).

Головне вікно програми містить такі елементи:

- два поля для виведення зображень;
- кнопка LoadImage для завантаження зображення в ліве поле для виведення зображень;
- кнопка CopyImage для копіювання зображення з лівого поля в праве;
- кнопка Encrypt для виконання криптографічного перетворення з використанням генераторів ПВЧ;
- кнопка Dencrypt для відновлення зображення з файлу зображення захищеного криптографічним методом. Виводиться в праве поле, Для

його збереження у файл у форматі зображення необхідно скористатись командою SaveR;

- кнопка HideO приховує молодші 5 біт кожної компоненти кольору шляхом їх перемішування і гамування. В праве поле виводиться зображення з спотворене зображення, що захищається. Для його збереження у файл у форматі зображення необхідно скористатись командою SaveR;
- кнопка RecoverHY відновлює зображення з файлу зображення, захищеного командою HideO, виводиться в праве поле;
- кнопка HideC приховує зображення, що захищається, в зображенні контейнері. В праве поле виводиться зображення-контейнер з вбудованим зображенням, що захищається. Для його збереження у файл у форматі зображення необхідно скористатись командою SaveR;
- кнопка RecoverC відновлює зображення з файлу зображення-контейнера, відновлене зображення виводиться в праве поле для виведення зображень;
- кнопка HideWH приховує зображення, що захищається, в коефіцієнтах перетворення Уолша-Адамара;
- кнопка RecoverWH відновлює зображення з файлу коефіцієнтів перетворення Уолша-Адамара, декодоване зображення виводиться в праве поле для виведення зображень, вилучене з контейнера зображення виводиться в ліве поле для виведення зображень;
- кнопка SaveR записує зображення з правого поля у файл у форматі зображення, в це поле виводиться зображення після перетворення або відновлення;
- кнопка About для виведення відомостей про роботу і автора розробки.

При виборі команд LoadImage, Encrypt, Dencrypt, HideO, SaveR відкриється стандартне вікно для вибору файлу. Необхідно вибрати файл і клацнути мишею по кнопці «Відкрити». Після цього виконуються відповідні перетворення.

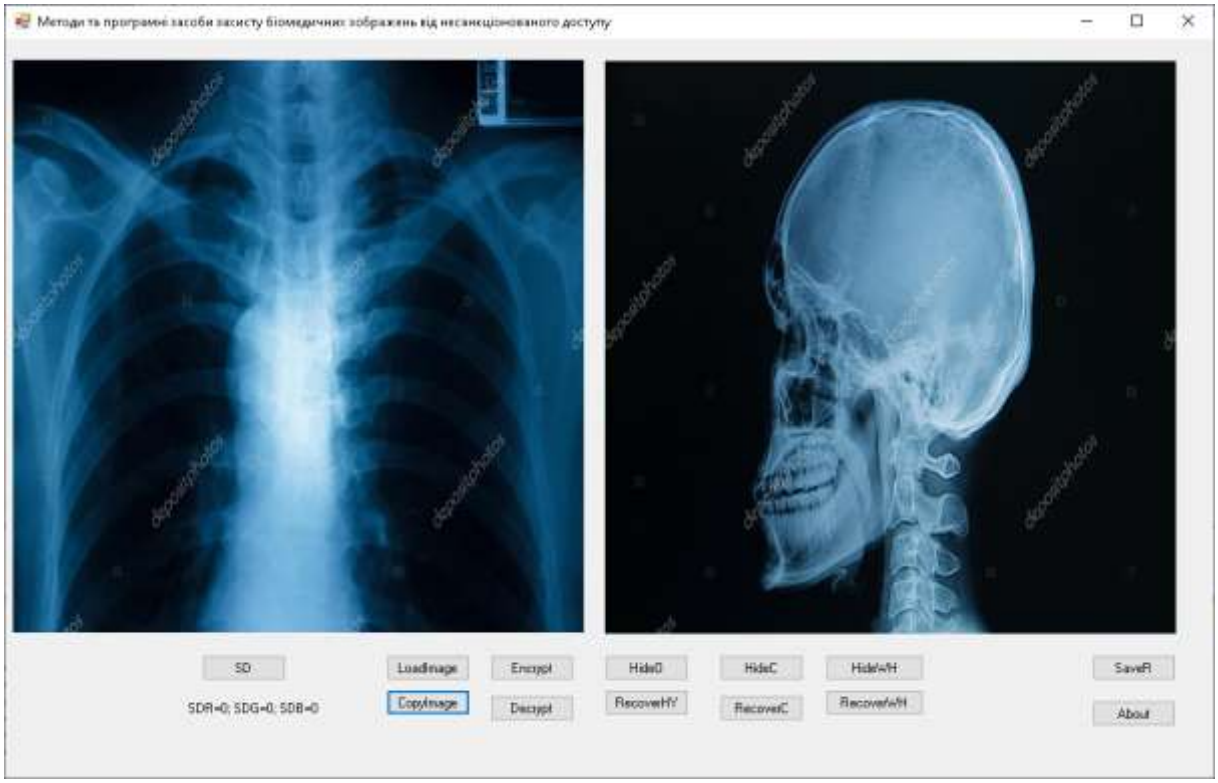


Рисунок 4.1 – Головне вікно програми

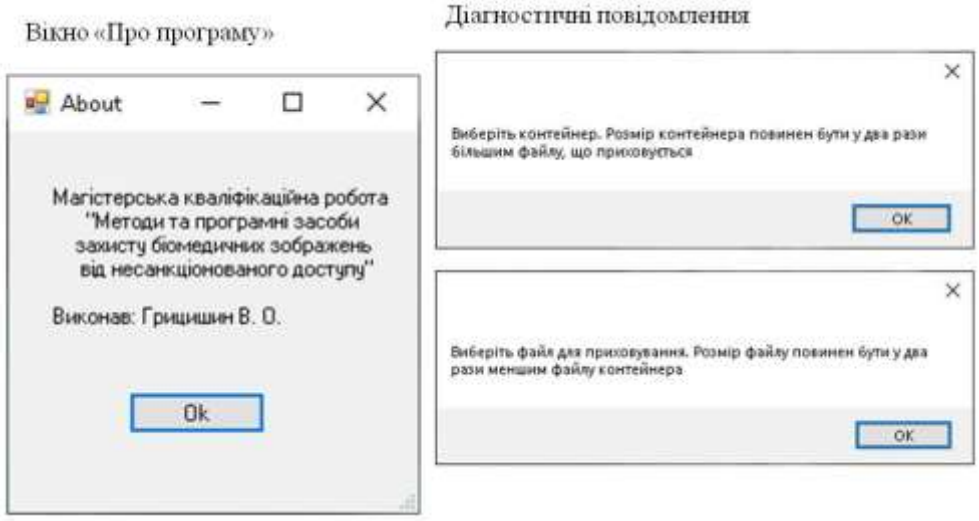


Рисунок 4.2 – Інші вікна програми

При виборі команд HideC, HideWH спочатку з'являються діагностичні повідомлення про те, що розмір контейнера повинен бути у два рази більшим зображення, що приховується, а зображення, що приховується ву два рази меншим (рис. 4.2). Якщо заявлені умови не виконуються, то пропонується обрати інший контейнер.

При виборі команди About (рис. 4.2) з’являється вікно з інформацією про роботу та автора роботи.

В цілому, використання команд програми є інтуїтивно зрозумілим і не вимагає додаткових пояснень.

4.2 Тестування роботи програми в режимі шифрування та режимі приховування молодших розрядів

При проведенні експериментальних досліджень застосовуються наступні технічні засоби :

Монітор - класу SVGA

Процесор - AMD Athlon ™ II, 3 GHz

Початкове напівтонове або кольорове зображення представлене у форматі 512x512 з глибиною 24 Біт/піксель.

Дослідимо працездатність програми в таких режимах:

- робота програма при виборі команд Encrypt та Dencrypt;
- робота програма при виборі команд Hide та RecoverH.

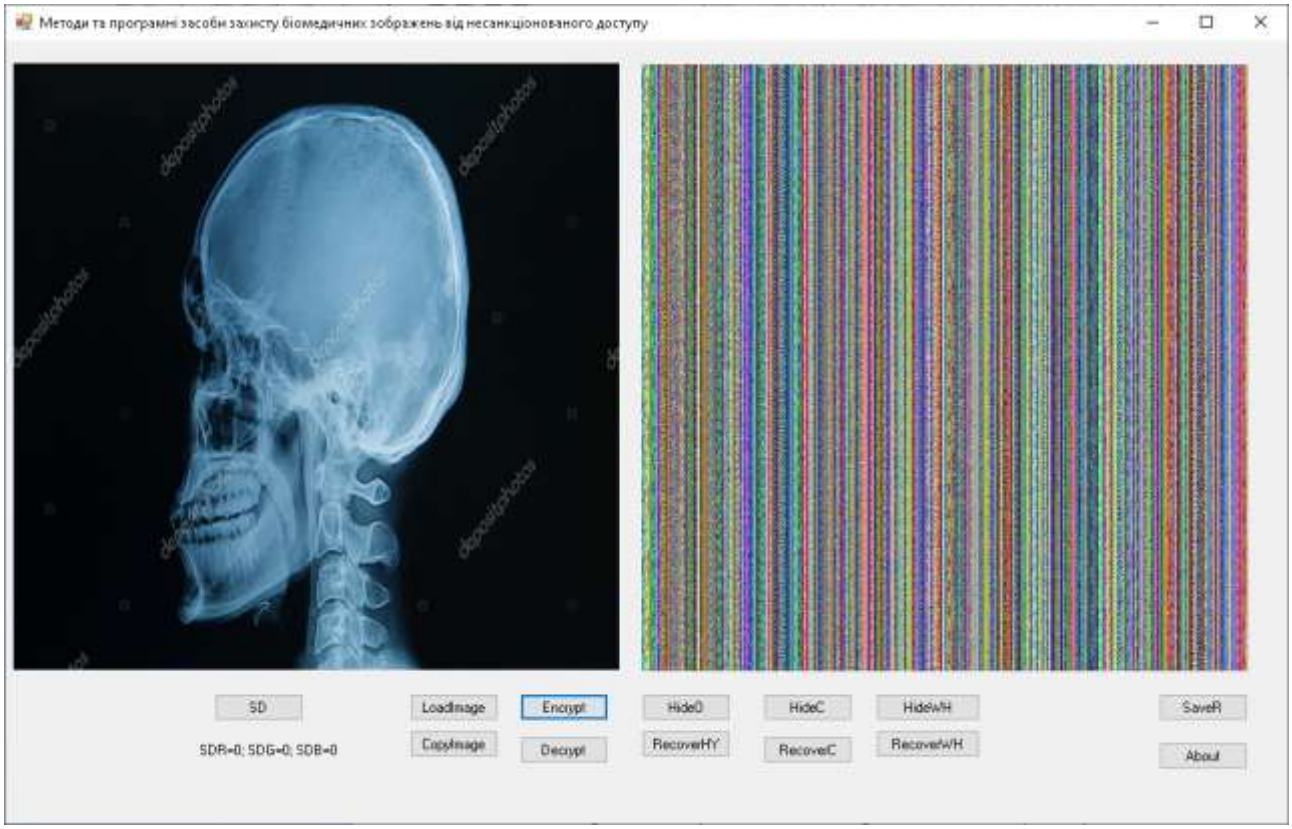


Рисунок 4.3 – Зображення (праве поле) після криптографічного перетворення

Робота програма при виборі команд Encrypt та Decrypt. При виборі команди Encrypt відкривається вікно вибору файлу. Виберемо файл image2Chr.bmp. В результаті виконання перетворення зображення буде зашифровано і ведемо в праве поле для виведення зображень (рис. 4.3).

Збережемо це зображення командою SaveR. При виборі команди Decrypt відкривається вікно вибору файлу. Виберемо зашифрований файл. В результаті виконання перетворення зображення буде відновлено (рис. 4.4). Завантажимо початкове зображення в ліве поле (команда LoadImage) і виберемо команду SD.

Середньо квадратичне відхилення рівне нулю, тобто модуль працює коректно, зображення відтворюється з абсолютною точністю після виконання перетворень.

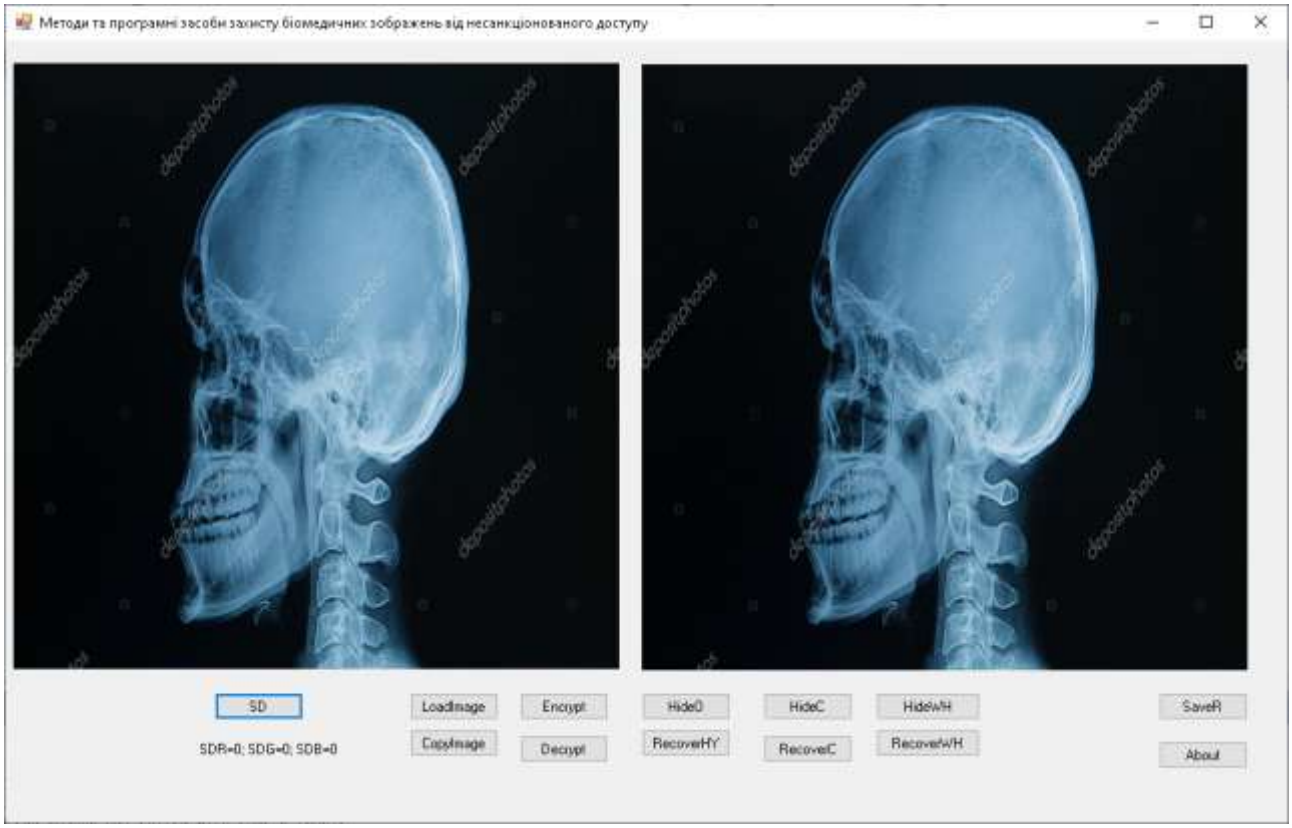


Рисунок 4.4 - Відновлене зображення (праве поле) після криптографічного перетворення

Робота програма при виборі команд HideO та RecoverHY. При виборі команди Hide відкривається вікно вибору файла. Виберемо файл image1GK.bmp. В результаті виконання перетворення зображення буде приховано молодші 5 біт в кожній компоненті пікселя і виведемо в праве поле для виведення зображень (рис. 4.5). Середньоквадратичне відхилення між початковим і спотвореним зображеннями 14.

Збережемо це зображення командою SaveR. При виборі команди RecoverHY відкривається вікно вибору файлу. Виберемо перетворений файл. В результаті виконання перетворення зображення буде відновлено зображення (рис. 4.6). Завантажимо початкове зображення в ліве поле (команда LoadImage) і виберемо команду SD.

Середньо квадратичне відхилення рівне нулю, тобто модуль працює коректно, зображення відтворюється з абсолютною точністю після виконання перетворень.

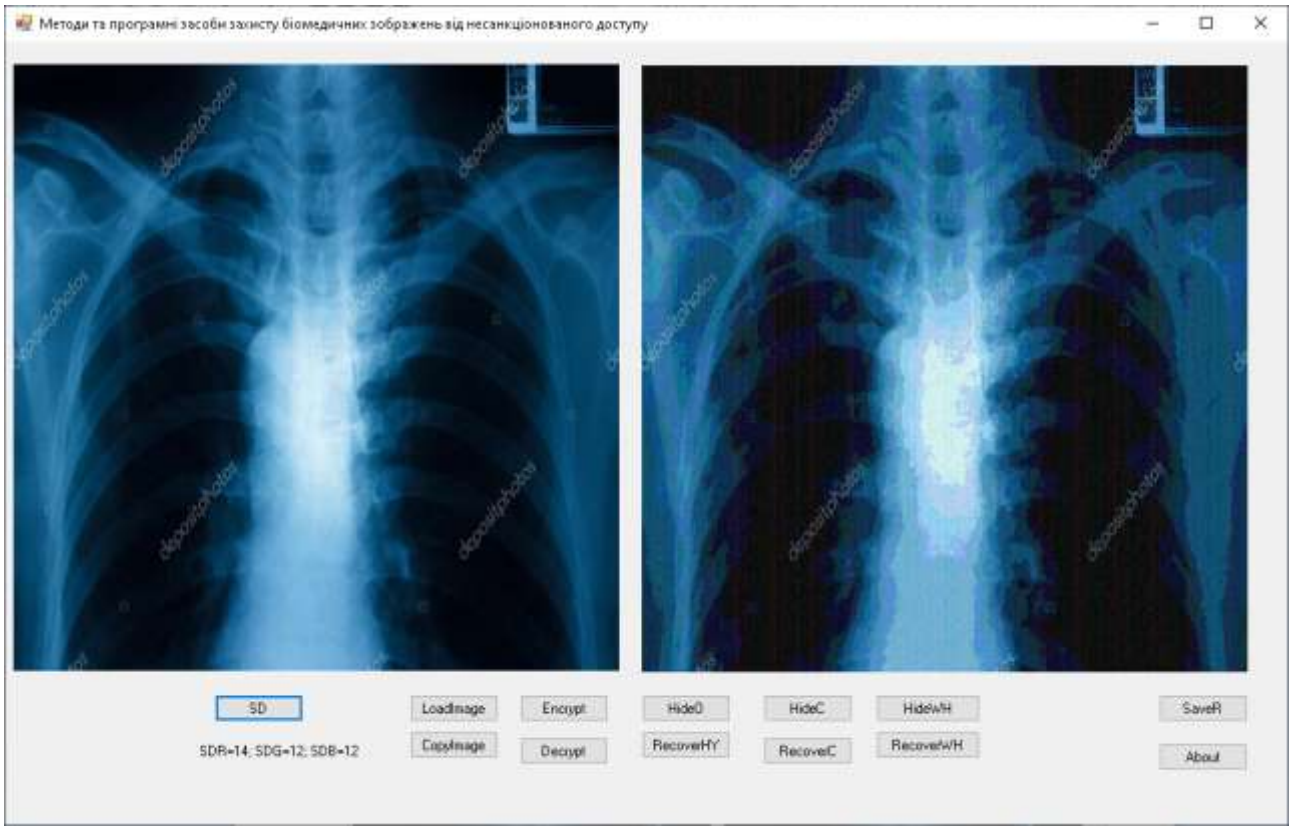


Рисунок 4.5 - Зображення (праве поле) з прихованими молодшими бітами

Рисунок 4.6 – Відновлене зображення (праве поле). СКВ=0

4.3 Тестування роботи програми в режимі стеганографічного захисту в просторовій області

При приховуванні зображення в зображенні-контейнері головною вимогою є забезпечення таємності самого факту приховування. Тобто візуально зобаження-контейнер не повинно відрізнятись після приховування від початкового. При проведенні досліджень в якості контейнера виберемо зображення типу рентгенівське зображення та довільне зображення. Будемо досліджувати залежність візуальної якості зображення після вбудовування зображення, що захищається від кількості біт вбудованих у кожний піксель.

Вбудуємо по 5 біт в кожну складову кольору в зображення контейнер image1GK.bmp (рентгенівський знімок) та lena512.bmp (зображення в градаціях сірого). Вибір довільного зображення в градаціях сірого пояснюється тим, що на такому зображенні спотворення більш помітні у порівнянні з кольоровим. Вбудовується зображення image2Chr256.bmp (рентгенівське зображення) розміром 256x256. Результати наведено на рис. 4.7.

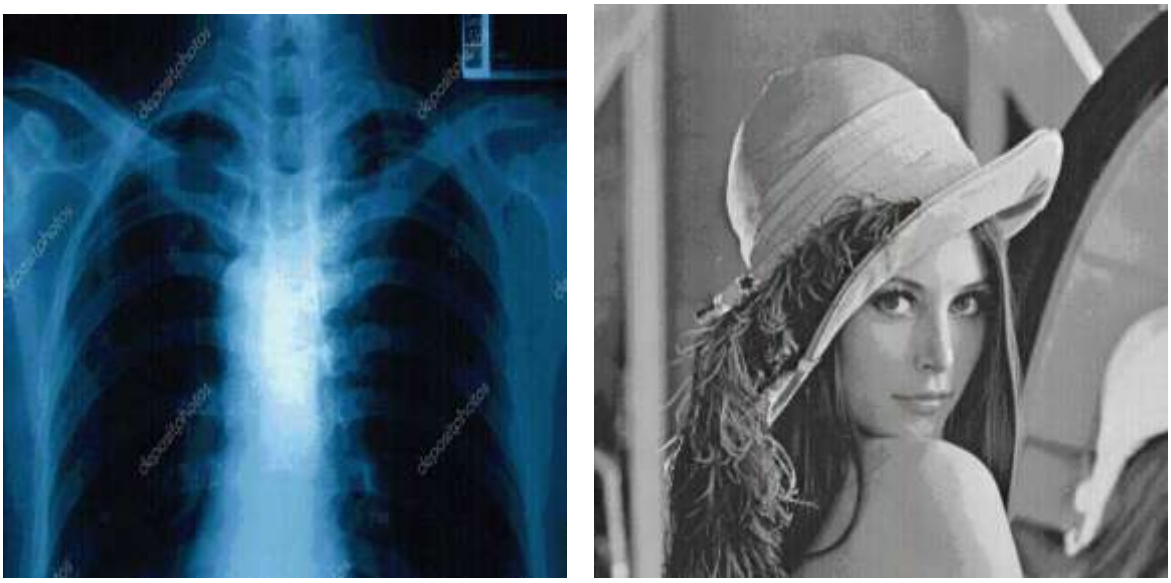


Рисунок 4.7 – Зображення-контейнер з вбудованим зображенням (5біт) СКВ=9-10

Спотворення помітні (рис. 4.7), тому необхідно зменшити кількість біт, що приховуються до 4. Для дослідження використаємо ті ж самі зображення в якості контейнера і зображення що приховується.

Аналіз результатів дослідження (рис. 4.8) показує, що спотворення візуально непомітні, тому будемо приховувати по 4 біти в кожній складовій кольору пікселя, тобто 12 біт на піксель.

Рисунок 4.8 – Зображення-контейнер з вбудованим зображенням (4біт, наповнення контейнера 0,5), СКВ=4

Але при розмірах зображення, що приховується, 256x256, контейнер розміром 512x512 заповнюється лише наполовину. Тому сформуємо зображення розміром 362x362, яке забезпечує повне заповнення контейнера розміром 512x512. Дійсно

512x512=262144пікселя.

Оскільки на 1 піксель приховуваного зображення потрібно 2 пікселя контейнера, то це значення поділити на 2:

262144:2=131072.

Тоді сторона квадратного зображення не повинна перевищувати квадратний корінь з цього числа, тобто бути рівною 362.

Результати досліджень для такого зображення наведено на рис. 4.9.

Рисунок 4.9 – Фрагмент зображення-контейнер з вбудованим зображенням (4біт). Повне заповнення контейнера, СКВ=6

Спотворення візуально помітні, про що свідчить і зростання СКВ до 6.

Тому можна зробити такі висновки: при приховуванні 4 біт в кожному пікселі кожної складової кольору кількість пікселів в контейнері повинна перевищувати кількість пікселів зображення, що приховуються, у 4 рази. Тоді заповнюваність контейнера не перевищуватиме половини можливої місткості і спотворення будуть непомітні.

4.4 Тестування роботи програми в режимі стеганографічного захисту в частотній області

Досліджується залежність якості відновленого зображення від кількості біт вбудованих в коефіцієнти перетворення Уолша-Адамара при розмірах фрагментів, які вибираються з зображення, 8х8. Всі коефіцієнти квантування встановлюються в значення «1», тобто використовуються режим «майже без втрат».

Очікувані результати при коефіцієнтах квантування рівних «1» такі:

- якщо використовувати класичний варіант, ділити кожний коефіцієнт на 8 (нормалізовані коефіцієнти) при прямому та зворотному перетворенні, то тут кількість біт, що заміщується при перетворенні не повинна перевищувати 4, а при заміщенні 3 біт СКВ=0.
- після обчислення коефіцієнтів перетворення Уолша-Адамара кожний з них націло ділиться на розмір сторони фрагмента в межах якого виконується перетворення, в нашому випадку на 8 і після виконання зворотного перетворення отримані значення також діляться на 8. Тому слід очікувати, що при заміщенні молодших розрядів коефіцієнта перетворення спотворення не будуть помітні, якщо не виконувати ділення коефіцієнтів на 8 (не нормалізовані коефіцієнти) при прямому перетворенні, а виконати ділення на 64 при зворотному.

Вбудуємо по 4 біт в нормалізовані коефіцієнти кожної складової кольору в зображення контейнер image1GK.bmp (рентгенівський знімок) та lena512.bmp

(зображення в градаціях сірого). Вбудовується зображення image2Chr256.bmp (рентгенівське зображення) розміром 256x256.

Спотворення візуально непомітні, але зросло на 1 СКВ у порівнянні з приховуванням в просторовій області (рис. 4.8). Це пояснюється тим, що при зворотному перетворенні похибка розповсюджується на весь фрагмент зображення.

Але при повному заповненні контейнера спотворення у вигляді регулярної структури точок помітні (рис. 4.10). Тому змінимо умови проведення експерименту, ділення коефіцієнтів на розмір фрагмента перетворення виконується тільки при зворотному перетворенні.

Рисунок 4.10 – Фрагмент відновленого зображення-контейнер з вбудованим в нормалізовані коефіцієнти Уолша-Адамара зображенням. Повне заповнення контейнера, СКВ=7

Результати експерименту такі (рис. 4.11, табл. 4.1): спотворення непомітні, СКВ=1, що є повністю прийнятним з точки зору стеганостійкості алгоритму приховування. Але при цьому збільшується розмір файлу з

коефіцієнтами після ущільнення методом бех втрат. Це пояснюється тим, що коефіцієнти перетворення не нормалізуються і відповідно зменшується вплив на якість відновленого зображення вбудованого повідомлення, оскільки нормалізація виконується після виконання зворотного перетворення шляхом цілочислового ділення кожного коефіцієнта на 64, тобто 6 молодших розрядів спотворених вбудованим зображенням відкидаються.

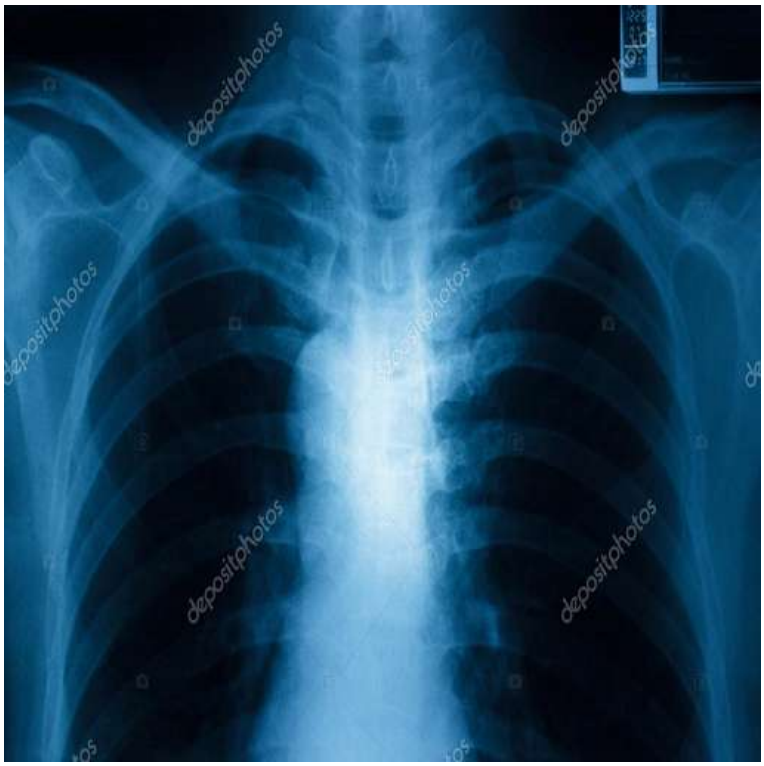


Рисунок 4.11 – Відновлене зображення-контейнер з вбудованим в не нормалізовані коефіцієнти Уолша-Адамара зображенням. Повне заповнення контейнера, СКВ=1

Щодо збільшення розміру файлу з вбудованими коефіцієнтами, то його розмір все ж менший розміру файлу початкового зображення-контейнера на 9 відсотків.

Таким чином, приховування зображень в коефіцієнтах двовимірного ортогонального перетворення дозволяє збільшити інформаційну ємність контейнера при збереженні високої якості зображення-контейнера з

вбудованим повідомленням, що в свою чергу підвищує стеганостійкість методу.

Однак, потрібні додаткові дослідження можливостей збільшення ємності контейнера за рахунок врахування частот коефіцієнтів перетворення. Наприклад, у високочастотних коефіцієнтах можна б було приховувати більше біт повідомлення, оскільки ці коефіцієнти відповідають за дрібні деталі зображення до спотворення яких око людини менш чутливе.

Таблиця 4.1 – Стеганографічний захист в частотній області при повному заповненню контейнера (4біти на кожний коефіцієнт перетворення Уолша-Адамара)

Розмір початкового зображення, байт		Розмір зображення після приховування та ущільнення без втрат, байт	Середньо-квадратичне відхилення (СКВ)	Візуальна оцінка якості
786486 (color)	Нормалізація коефіцієнтів при прямому і зворотному перетворенні	532541	7	Погана, помітні зміни
	Нормалізація коефіцієнтів тільки при зворотному перетворенні	717101	1	Відмінна

4.5 Висновки та перспективи подальших досліджень

Основні результати досліджень такі:

1. Розроблено керівництво користувача, яке містить необхідні відомості для установки програми на комп’ютер та її експлуатації.

2. Тестування програми показало її повну працездатність та відповідність завданню на роботу.

Таким чином можна зробити висновок, що поставлені задачі та цілі роботи виконано.

Дослідження показали, що приховування в коефіцієнтах перетворення Уолша-Адамара забезпечує найбільший об’єм контейнера та найвищу якість зображення-контейнера з вбудованим зображенням. Однак, потрібні додаткові дослідження:

1. Дослідження можливості збільшення об’єму контейнера за рахунок врахування частот коефіцієнтів перетворення.

2. Дослідження застосування інших перетворень для приховування зображень в зображеннях. Перспективними є дослідження застосування дискретного косинусного перетворення, перетворення Фур’є та інших ортогональних перетворень. На особливу увагу заслуговують Wavelet перетворення.

3. Дослідження квантування коефіцієнтів цих перетворень при стеганографічному захисті зображень, що дозволить збільшити коефіцієнт ущільнення зображень після стеганографічного захисту.

4. Дослідження застосування методів штучного інтелекту при стеганографічному захисті біомедичних зображень, зокрема, застосування Deep Learning.

5 ЕКОНОМІЧНА ЧАСТИНА

Науково-технічна розробка має право на існування та впровадження, якщо вона відповідає вимогам часу, як в напрямку науково-технічного прогресу та і в плані економіки. Тому для науково-дослідної роботи необхідно оцінювати економічну ефективність результатів виконаної роботи.

Магістерська кваліфікаційна робота за темою «Методи та програмні засоби захисту біомедичних зображень від несанкціонованого доступу » відноситься до науково-технічних робіт, які орієнтовані на виведення на ринок (або рішення про виведення науково-технічної розробки на ринок може бути прийнято у процесі проведення самої роботи), тобто коли відбувається так звана комерціалізація науково-технічної розробки. Цей напрямок є пріоритетним, оскільки результатами розробки можуть користуватися інші споживачі, отримуючи при цьому певний економічний ефект. Але для цього потрібно знайти потенційного інвестора, який би взявся за реалізацію цього проекту і переконати його в економічній доцільності такого кроку.

Для наведеного випадку нами мають бути виконані такі етапи робіт:

1) проведено комерційний аудит науково-технічної розробки, тобто встановлення її науково-технічного рівня та комерційного потенціалу;

2) розраховано витрати на здійснення науково-технічної розробки;

3) розрахована економічна ефективність науково-технічної розробки у випадку її впровадження і комерціалізації потенційним інвестором і проведено обґрунтування економічної доцільності комерціалізації потенційним інвестором.

5.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного і технологічного аудиту дослідження за темою «Методи та програмні засоби захисту біомедичних зображень від несанкціонованого доступу » є оцінювання науково-технічного рівня та рівня

комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями, наведеними в табл. 5.1 [30].

Таблиця 5.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних умовах
Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної	Необхідно наймати фахівців або витрачати значні кошти та	Необхідне незначне навчання фахівців та збільшення їх	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї

	реалізації ідеї	час на навчання наявних фахівців	штату		
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь- які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки потрібно звести до таблиці.

Таблиця 5.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	1	2	3
	Бали:		
1. Технічна здійсненність концепції	4	3	4
2. Ринкові переваги (наявність аналогів)	3	3	4
3. Ринкові переваги (ціна продукту)	4	4	3
4. Ринкові переваги (технічні властивості)	3	3	4
5. Ринкові переваги (експлуатаційні витрати)	3	3	3
6. Ринкові перспективи (розмір ринку)	3	4	3
7. Ринкові перспективи (конкуренція)	3	3	4

8. Практична здійсненність (наявність фахівців)	3	3	3
9. Практична здійсненність (наявність фінансів)	3	3	4
10. Практична здійсненність (необхідність нових матеріалів)	3	3	3
11. Практична здійсненність (термін реалізації)	3	3	3
12. Практична здійсненність (розробка документів)	3	3	3
Сума балів	38	38	41
Середньоарифметична сума балів $СБ_c$	39		

За результатами розрахунків, наведених в таблиці 5.2, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в табл. 5.3 [30].

Таблиця 5.3 – Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ ,розрахована на основі висновків експертів	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вище середнього
21…30	Середній
11…20	Нижче середнього
0…10	Низький

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Методи та програмні засоби захисту біомедичних зображень від несанкціонованого доступу » становить 39 бали, що, відповідно до таблиці 5.3, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

5.2 Розрахунок витрат на проведення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи на тему «Методи та програмні засоби захисту біомедичних зображень від несанкціонованого доступу », під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за відповідними статтями.

5.2.1 Витрати на оплату праці

До статті «Витрати на оплату праці» належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням

конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці.

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою [30]:

$$З_о = \sum_{i=1}^{k} \frac{М_{ni} \cdot t_i}{Т_р}, \quad (5.1)$$

де k – кількість посад дослідників залучених до процесу досліджень;

$М_{ni}$ – місячний посадовий оклад конкретного дослідника, грн;

t_i – число днів роботи конкретного дослідника, дні;

$Т_р$ – середнє число робочих днів в місяці, $Т_р$=22 дні.

$$З_о = 19000{,}00 \cdot 91 / 22 = 78590{,}91 \text{ грн.}$$

Проведені розрахунки зведемо до таблиці.

Таблиця 5.4 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник проекту	19000	863,64	91	78590,91
Інженер-розробник програмного забезпечення	17000	772,73	91	70318,18
Консультант в сфері машинного навчання	17000	772,73	30	23181,82
Консультант в сфері інвестицій	17000	772,73	30	23181,82
Всього				195272,73

Основна заробітна плата робітників

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт НДР розраховуємо за формулою:

$$З_р = \sum_{i=1}^{n} С_i \cdot t_i, \quad (5.2)$$

де $С_i$ – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою:

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}}, \quad (5.3)$$

де M_M – розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства), приймемо *Мм*=8000,00 грн;

K_i – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду (табл. Б.2, додаток Б) [30];

K_c – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об'єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати.

T_p – середнє число робочих днів в місяці, приблизно T_p = 22 дні;

$t_{зм}$ – тривалість зміни, год.

$C_1 = 8000,00 \cdot 1,10 \cdot 1,65 / (22 \cdot 8) = 82,50$грн.

$З_{р1} = 82,50 \cdot 7,00 = 577,50$ грн.

Таблиця 5.5 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
Встановлення обладнання	7	2	1,1	82,50	577,50
Підготовка робочого місця	5	2	1,1	82,50	412,50
Інсталяція програмного забезпечення	5	5	1,7	127,50	637,50
Всього					1627,50

Додаткова заробітна плата дослідників та робітників

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (5.4)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Приймемо 12%.

$З_{дод}$ = (195272,73+ 1627,50) · 12 / 100% = 21659,03 грн.

5.2.2 Відрахування на соціальні заходи

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \quad (5.5)$$

де $Н_{зп}$ – норма нарахування на заробітну плату. Приймаємо 22%.

Зн = (195272,73+ 1627,50+ 21659,03) · 22 / 100% = 48083,03 грн.

5.2.3 Сировина та матеріали

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень.

Витрати на матеріали (*М*), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою:

$$М = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j - \sum_{j=1}^{n} В_j \cdot Ц_{вj}, \quad (5.6)$$

де $Н_j$ – норма витрат матеріалу *j*-го найменування, кг;

n – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу *j*-го найменування, грн/кг;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ = *1,1 … 1,15*);

$В_j$ – маса відходів *j*-го найменування, кг;

$Ц_{вj}$ – вартість відходів *j*-го найменування, грн/кг.

$М_1$ = 2 · 200,00 · 1,1 - 0,000 · 0,00 = 440 грн.

Проведені розрахунки зведемо до таблиці.

Таблиця 5.6 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за од., грн	Норма витрат, од	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Офісний папір Calipso Plus A4-500-80	200	2	0	0	440
Папір для записів Calipso Parers Light A5	110	1	0	0	121
Органайзер офісний Calipso Office	210	1	0	0	231
Канцелярське приладдя (набір офісного працівника)	150	2	0	0	330
Flesh-пам'ять Kingston 16 GB	350	1	0	0	385
Тека для паперів CALIPSO BOX	82	1	0	0	90,2
Всього					1597,2

5.2.4 Розрахунок витрат на комплектуючі

Витрати на комплектуючі *($К_в$)*, які використовують при проведенні НДР на тему «Методи та програмні засоби захисту біомедичних зображень від несанкціонованого доступу » відсутні.

5.2.5 Спецустаткування для наукових (експериментальних) робіт

До статті «Спецустаткування для наукових (експериментальних) робіт» належать витрати на виготовлення та придбання спецустаткування необхідного для проведення досліджень, також витрати на їх проектування, виготовлення, транспортування, монтаж та встановлення в даному дослідженні не використовувалось.

5.2.6 Програмне забезпечення для наукових (експериментальних) робіт

До статті «Програмне забезпечення для наукових (експериментальних) робіт» належать витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення, (програм, алгоритмів, баз даних) необхідних для проведення досліджень, також витрати на їх проектування, формування та встановлення.

Балансову вартість програмного забезпечення розраховуємо за формулою:

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_i \ , \qquad (5.7)$$

де $Ц_{iпрг}$ – ціна придбання одиниці програмного засобу даного виду, грн;

$C_{прг.i}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, (K_i = *1,10…1,12);*

k – кількість найменувань програмних засобів.

$B_{прг} = 8300 \cdot 1 \cdot 1{,}12 = 9296$ *грн.*

Отримані результати зведемо до таблиці:

Таблиця 5.7– Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Інтегроване середовище розробки Visual Studio 2022	1	25000	28000
Windows 10 Pro	1	8300	9296
Всього			37296

5.2.7 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації за формулою:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12} , \qquad (5.8)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$A_{обл} = (70000{,}00 \cdot 3) / (5 \cdot 12) = 3500$*грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 5.8 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Ноутбук LG, 2 шт	70 000	5	3	3500,00
Монітор samsung, 2шт	12 000	5	3	600,00
Маршрутизатор TP-Link	2 500	2	3	312,50
Робоче місце дослідника	20000	5	3	1000,00
Оргтехніка	7000	4	3	437,50
Приміщення лабораторії	250000	20	3	3125,00
ОС Windows 10	9296	3	3	774,67
Хмарна інфраструктура AWS EC2	3000	3	2	166,67
Всього				9916,33

5.2.8 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію (B_e) розраховуємо за формулою:

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_e \cdot K_{вni}}{\eta_i}, \quad (5.9)$$

де W_{yi} – встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_e$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо $Ц_e = 10,5$ грн;

$K_{вni}$ – коефіцієнт, що враховує використання потужності, $K_{вni} < 1$;

η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

$B_e = 2 \cdot 0,17 \cdot 7280,0 \cdot 10,50 \cdot 0,95 / 0,97 = 2545,37$ грн.

Проведені розрахунки зведемо до таблиці.

Таблиця 5.9 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Ноутбук LG, 2 шт	0,17	1456	2545,37
Монітор samsung, 2шт	0,065	1456	973,23
Робоче місце дослідника	0,15	1456	2245,92
Маршрутизатор	0,01	728	74,86
Оргтехніка	0,45	16	74,04
Всього			5913,43

5.2.9 Службові відрядження

До статті «Службові відрядження» дослідної роботи належать витрати на відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, аспірантів, зайнятих розробленням досліджень, відрядження, пов'язані з проведенням випробувань машин та приладів, а також витрати на відрядження на наукові з'їзди, конференції, наради, пов'язані з виконанням конкретних досліджень.

Витрати за статтею «Службові відрядження» розраховуємо як 20…25% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{св} = (З_о + З_р) \cdot \frac{Н_{св}}{100\%}, \quad (5.10)$$

де $Н_{св}$ – норма нарахування за статтею «Службові відрядження», приймемо $Н_{св}$ = 20%.

$B_{св}$ = (195272,73+ 1627,50) · 20 / 100% = 39380,05 грн.

5.2.10 Витрати на роботи, які виконують сторонні підприємства, установи і організації

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» розраховуємо як 30…45% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{сп} = (З_о + З_р) \cdot \frac{Н_{сп}}{100\%}, \quad (5.11)$$

де $Н_{сп}$ – норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації», приймемо $Н_{сп}$= 35%.

$В_{сп}$ = (195272,73+ 1627,50) · 35 / 100% = 68915,08 грн.

5.2.11 Інші витрати

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \qquad (5.12)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 50%.

$І_в$ = (195272,73+ 1627,50) · 50 / 100% = 98450,11 грн.

5.2.12 Накладні (загальновиробничі) витрати

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов’язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов’язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов’язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$В_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \qquad (5.13)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 120%.

$В_{нзв}$ = (195272,73+ 1627,50) · 120 / 100% = 236 280,27 грн.

Витрати на проведення науково-дослідної роботи розраховуємо як суму всіх попередніх статей витрат за формулою:

$$B_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + В_{спец} + В_{прг} + А_{обл} + В_е + В_{св} + В_{сп} + І_в + В_{нзв}. \quad (5.14)$$

$B_{заг=}$ *858990,76 грн.*

Загальні витрати *ЗВ* на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою:

$$ЗВ = \frac{B_{заг}}{\eta}, \quad (5.15)$$

де η - коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η=0,8.

ЗВ = 858990,76 / 0,8 = 1073738,45 грн.

5.3 Розрахунок економічної ефективності науково-технічної розробки при її можливій комерціалізації потенційним інвестором

В ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку.

Результати дослідження передбачають комерціалізацію протягом 3-х років реалізації на ринку.

В цьому випадку майбутній економічний ефект буде формуватися на основі таких даних:

ΔN – збільшення кількості споживачів продукту, у періоди часу, що аналізуються, від покращення його певних характеристик;

1-й рік – 4000 користувачів;

2-й рік – 3200 користувачів;

3-й рік – 1500 користувачів.

N – кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки, приймемо 1000000 користувачів;

$Ц_б$ – вартість програмного продукту у році до впровадження результатів розробки, приймемо 8000 грн;

$\pm\Delta Ц_о$ – зміна вартості програмного продукту від впровадження результатів науково-технічної розробки, приймемо (-1000,00) грн.

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із 3-х років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою [30]:

$$\Delta П_i = (\pm\Delta Ц_о \cdot N + Ц_о \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (5.16)$$

де λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2024 році ставка податку на додану вартість складає 20%, а коефіцієнт $\lambda = 0,8333$;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту. Приймемо $\rho = 30\%$;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2024 році $\vartheta = 18\%$;

Збільшення чистого прибутку 1-го року:

$\Delta П_1 = (-1000 \cdot 10000,00 + 7000 \cdot 4000) \cdot 0,83 \cdot 0,3 \cdot (1-0,18/100\%) = 3688524$грн.

Збільшення чистого прибутку 2-го року:

$\Delta П_2 = (-1000 \cdot 10000,00 + 7000 \cdot (4000+3200)) \cdot 0,83 \cdot 0,3 \cdot (1-0,18/100\%) =$ 8278687,2 грн.

Збільшення чистого прибутку 3-го року:

$\Delta П_3 = (-1000 \cdot 10000,00 + 7000 \cdot (4000+3200+1500)) \cdot 0,83 \cdot 0,3 \cdot (1-0,18/100\%) =$ 10430326,2 грн.

Приведена вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \quad (5.17)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

T – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ=0,25;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$$ПП = 3688524/(1+0,25)^1 + 8278687,2/(1+0,25)^2 + 10430326,2/(1+0,25)^3 =$$

13589506,02 грн.

Величина початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки:

$$PV = k_{інв} \cdot ЗВ, \quad (5.18)$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$=4;

ЗВ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 1073738,45 грн.

$PV = k_{інв} \cdot ЗВ = 4 \cdot 1073738,45 = 4294953,796$ грн.

Абсолютний економічний ефект $E_{абс}$ для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV \quad (5.19)$$

де $ПП$ – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 13589506,02 грн;

PV – теперішня вартість початкових інвестицій, 4294953,796 грн.

$E_{абс} = ПП - PV$ = 13589506,02 - 4294953,796 = 9294552,23 грн.

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1, \quad (5.20)$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, 9294552,23 грн;

PV – теперішня вартість початкових інвестицій, 4294953,796 грн;

$T_ж$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 3 роки.

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1 = (1+9294552,23/\ 4294953,796)^{1/3} = 0,47.$$

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$:

$$\tau_{мін} = d + f, \quad (5.21)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2024 році в Україні d =0,1;

f – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,25.

$\tau_{мін}$ = 0,11+0,25 = 0,36 < 0,47 свідчить про те, що внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну

роботу за темою «Методи та програмні засоби захисту біомедичних зображень від несанкціонованого доступу » доцільно.

Період окупності інвестицій $T_{ок}$ які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_в}, \tag{5.22}$$

де $E_в$ – внутрішня економічна дохідність вкладених інвестицій.

$T_{ок}$ = 1 / 0,47 = 2,14 року.

$T_{ок} <$ 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

5.4 Висновки

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Методи та програмні засоби захисту біомедичних зображень від несанкціонованого доступу » становить 39 балів, що, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу вище середнього).

Також термін окупності становить 2,14 р., що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Отже можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Методи та програмні засоби захисту біомедичних зображень від несанкціонованого доступу».

ВИСНОВКИ

У магістерській кваліфікаційній роботі проводилося дослідження захисту біомедичних зображень з використанням стеганографічних та криптографічних перетворень. Основні результати роботи такі:

1. Аналіз методів захисту біомедичних даних від несанкціонованого доступу показав, що для їх захисту, зокрема і зображень, використовуються традиційні криптографічні перетворення, які не враховують специфіку зображень.
2. Актуальною є розробка методів захисту біомедичних зображень, орієнтованих на врахування природи цих зображень.
3. Запропоновано методи захисту біомедичних зображень, відмінністю яких є виконання операцій захисту не з файлами, а у площині зображення з даними самих зображень.
4. Розроблено метод та алгоритм захисту біомедичних зображень, у якому позиція пікселя при розсіюванні та гамуванні по кожній складовій кольору визначається з використанням генератора ПВЧ.
5. Розроблено метод приховування важливої інформації, що міститься в зображенні у цьому ж зображенні за рахунок перемішування і гамування молодших 4-5 розрядів кожної складової пікселя.
6. Розроблено метод приховування важливої інформації, що міститься в рентгенівському зображенні, в контейнері того ж типу по 4 біти на піксель в кожній складовій кольору при заповненні контейнера наполовину.
7. Розроблено метод приховування важливої інформації, що міститься в рентгенівському зображенні, в трансформантах перетворення Уолша-Адамара, отриманих з зображення-контейнера, по 4 біти на піксель в кожній складовій кольору.
8. Виконано аналіз середовищ розробки та мов програмування, який показав, що для цієї розробки доцільним є використання середовища розробки Microsoft Visual Studio та мови програмування C++, оскільки ці засоби забезпечують

найкращі швидкісні характеристики для програм, що розробляються, а також мають розвинені сервісні функції, які пришвидшують розробку.

9. Розроблено програму для дослідження методів захисту біомедичних зображень від несанкціонованого доступу шляхом виконання криптографічних та стеганографічних перетворень.

10. Розроблено керівництво користувача, яке містить необхідні відомості для установки програми на комп'ютер та її експлуатації.

11. Тестування програми показало її повну працездатність та відповідність завданню на роботу.

12. Термін окупності менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Таким чином можна зробити висновок, що поставлені задачі та цілі магістерської кваліфікаційної роботи виконано.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Безпека та конфіденційність у віддаленій радіології. [Електронний ресурс]. https://radiolance.ua/bezpeka-ta-konfidentsijnist-u-viddalenij-radiologiyi/
2. Як захистити дані від несанкціонованого доступу? [Електронний ресурс]. URL:https://cybercalm.org/novyny/yak-zahystyty-dani-vid-nesanktsionovanogo-dostupu-porady/
3. Конспект лекцій з вивчення дисципліни «Обробка біомедичних зображень та реконструкція об'єктів» для студентів спеціальності 163 - Біомедична інженерія освітня програма Інтелектуальні штучні імпланти та медичні апарати в біоінженерії / Уклад. Л.Г. Коваль. – Вінниця : ВНТУ, 2020.
4. Методичні вказівки до виконання практичних робіт з дисципліни «Обробка біомедичних зображень та реконструкція об'єктів» для студентів спеціальності 163 - Біомедична інженерія освітня програма Інтелектуальні штучні імпланти та медичні апарати в біоінженерії / Уклад. Л.Г.Коваль. – Вінниця : ВНТУ, 2020.
5. Bin Zhang, Bahbibi Rahmatullah, Shir Li Wang, A. A. Zaidan, B. B. Zaidan & Penghui Liu. A review of research on medical image confidentiality related technology coherent taxonomy, motivations, open challenges and recommendations // Multimedia Tools and Applications, 2020, № 14, p. 21867-21906, https://doi.org/10.1007/s11042-020-09629-4.
6. Олександр РОМАНЮК, Володимир МАЙДАНЮК, Сергій ПАВЛОВ, Наталія ТІТОВА , Сергій РОМАНЮК. Ущільнення медичних зображень // Медико-технічна співпраця заради перемоги: Актуальні завдання медичної, біологічної фізики та інформатики. Матеріали доповідей та виступів III всеукраїнської науково-практичної конференції з міжнародною участю 5-6 квітня 2024 року Вінниця. – Вінниця: Едельвейс. – С. 89-93.

7. Олександр Романюк, Володимир Майданюк, Сергій Павлов, Наталія Тітова , Сергій Романюк. Шифрування медичних зображень // Медико-технічна співпраця заради перемоги: Актуальні завдання медичної, біологічної фізики та інформатики. Матеріали доповідей та виступів III всеукраїнської науково-практичної конференції з міжнародною участю 5-6 квітня 2024 року Вінниця. – Вінниця: Едельвейс. – С. 86-89.
8. Хорошко, В.О. К Комп'ютерна стеганографія: навчальний посібник / В.О. Хорошко, Ю.Є. Яремчук, В.В. Карпінець. − Вінниця: ВНТУ, 2017. − 155 с.
9. Грицишин В. О., Майданюк В. П. Ущільнення зображень з використанням перетворення Уолша-Адамара. Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2023/paper/view /17603 /14625 (дата звернення 21.10.2023). - Назва з екрана.
10. Майданюк В. П., Грицишин В. О. Захист біомедичних зображень від несанкціонованого доступу. / Інформаційні технології і автоматизація – 2024 / Матеріали XVII міжнародної науково-практичної конференції. Одеса, 31 жовтня - 1 листопада 2024 р. - Одеса, Видавництво ОНТУ, 2024 р. – С. 200-201.
11. Грицишин В. О., Майданюк В. П. Використання стеганографії для захисту рентгенівських знімків / Електронні інформаційні ресурси: створення, використання, доступ. Збірник матеріалів Міжнародної науково-практичної Інтернет конференції 20-21 листопада 2024 р. – Суми/Вінниця: НІКО/ КЗВО «Вінницька академія безперервної освіти», 2024. – 336 с.
12. Майданюк, В. П. Основи теорії інформації та кодування : електронний навчальний посібник комбінованого (локального та мережного) використання [Електронний ресурс] / Майданюк В. П., Романюк О. Н., Тужанський С. Є. – Вінниця : ВНТУ, 2022. – 133 с.
13. Data Encryption Standard. [Електронний ресурс]. URL: https://uk.wikipedia.org/wiki/Data_Encryption_Standard.

14. Hlynchuk, L., Hryshanovych, T. ., & Stupin, A. (2021). Реалізація стандарту симетричного шифрування des мовою програмування с та порівняння часу його роботи з відомими утилітами. Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка», 2(14), 118–130. https://doi.org/10.28925/2663-4023.2021.14.118130.
15. Advanced Encryption Standard. [Електронний ресурс]. URL: https://uk.wikipedia.org/wiki/Advanced_Encryption_Standard.
16. Що таке розширений стандарт шифрування (AES) і як він пов’язаний з NIST? [Електронний ресурс]. https://lazarusalliance.com/uk/what-is-advanced-encryption-standard-aes-and-how-is-it-related-to-nist/.
17. Biryukov, Alex and Khovratovich, Dmitry. Related-key Cryptanalysis of the Full AES-192 and AES-256. — Advances in Cryptology – ASIACRYPT.
18. Проблеми управління криптографічними ключами. [Електронний ресурс]. URL: https://cqr.company/ua/web-vulnerabilities/cryptographic-key-management-issues/.
19. Асиметричні алгоритми шифрування. [Електронний ресурс]. URL: https://uk.wikipedia.org/wiki/Асиметричні_алгоритми_шифрування.
20. Public-key cryptography. [Електронний ресурс]. URL: https://en.wikipedia.org/wiki/Public-key_cryptography.
21. Алгоритм шифрування RSA, види атак на нього. Реалізація мовою Python. [Електронний ресурс]. URL: https://dou.ua/forums/topic/43026/.
22. Кузнецов О. О. Стеганографія : навчальний посібник / О. О. Кузнецов, С. П. Євсеєв, О. Г. Король. – Х. : Вид. ХНЕУ, 2011. – 232 с.
23. Лісовська Ю. П. Воєнна стеганографія: квантова симуляція та космічна гіперспектроскопія навчальний посібник / П. М. Лісовський. – К. : Вид. Ліра-К, 2024. – 134 с.
24. Обчислювальна складність. [Електронний ресурс]. URL: https://uk.wikipedia.org/wiki/Обчислювальна_складність.
25. Як це працює? Оцінка складності алгоритмів. [Електронний ресурс]. URL: https://campus.epam.ua/ua/blog/420.

26. Чому важливо оцінювати складність алгоритму. [Електронний ресурс]. URL: https://foxminded.ua/skladnist-alhorytmu/.
27. Порівняння C++ з іншими мовами програмування. [Електронний ресурс]. URL: https://nikvesti.com/news/business/porivnannia-c-plus-plus-z-inshymi-movami-programuvannia-ua.
28. 6 сучасних мов програмування та їх мінуси. [Електронний ресурс]. URL: https://uk.itpedia.nl/2023/07/13/6-moderne-programmeertalen-en-hun-downsides/.
29. MSDN Library for Visual Studio 2019. [Електронний ресурс]. URL: https://msdn.microsoft.com.
30. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.
31. Кавецький В. В. Економічне обґрунтування інноваційних рішень: практикум / В. В. Кавецький, В. О. Козловський, І. В. Причепа. Вінниця : ВНТУ, 2016. 113 с.

Додаток А

(обов'язковий)

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ

Завідувач кафедрою ПЗ

д.т.н., проф.

_____ О. Н. Романюк

"19" вересня 2024 р.

Технічне завдання

на магістерську кваліфікаційну роботу «Методи та програмні засоби захисту біомедичних зображень від несанкціонованого доступу» за спеціальністю

121 – Інженерія програмного забезпечення

Керівник магістерської кваліфікаційної роботи:

_____ к.т.н., доцент В. П. Майданюк

"19" вересня 2024 р.

Виконав:

_____ студент гр. 1ПІ-23м В. О. Грицишин

"19" вересня 2024 р.

Вінниця – 2024 року

1. Найменування та галузь застосування

Магістерська кваліфікаційна робота: «Методи та програмні засоби захисту біомедичних зображень від несанкціонованого доступу».

Галузь застосування – стеганографія, навчальний процес.

2. Підстава для розробки.

Підставою для виконання магістерської кваліфікаційної роботи (МКР) є індивідуальне завдання на БДР та наказ № 310 від «17» вересня 2024 р. ректора ВНТУ про закріплення тем МКР.

3. Мета та призначення розробки.

Метою магістерської кваліфікаційної роботи є зменшення обчислювальних витрат для захисту рентгенівських зображень від несанкціонованого доступу за рахунок застосування стеганографічних перетворень.

Основними задачами дослідження є:

- обґрунтування доцільності розробки нового технічного рішення;
- розробка та модифікація методів приховування рентгенівських зображень;
- розробка алгоритмів і програм для дослідження приховування рентгенівських зображень;
- експериментальні дослідження ефективності стеганографічних перетворень;
- розрахунок економічних показників розробки.

4. Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись МКР:

1. Хорошко, В.О. К Комп’ютерна стеганографія: навчальний посібник / В.О. Хорошко, Ю.Є. Яремчук, В.В. Карпінець. − Вінниця: ВНТУ, 2017. − 155 с.

2. Кобилін О.А., Творошенко І. С. Методи цифрової обробки зображень: навч. посібник. – Харків: ХНУРЕ, 2021. – 124 с.
3. Перелигін Б.В. Цифрова обробка зображень: конспект лекцій. Одеса : Одеський державний екологічний університет, 2023. 186 с.
4. Майданюк В. П. Обробка сигналів. Навчальний посібник. - Вінниця: ВНТУ. Режим доступу: https://iq.vntu.edu.ua/method/getfile.php?fname = 63704.pdf&x=1&card_id=7303&id=63704 (дата звернення 11.01.2023). — Назва з екрана.
5. Грицишин В. О., Майданюк В. П. Ущільнення зображень з використанням перетворення Уолша-Адамара. Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2023/paper/view /17603 /14625 (дата звернення 21.10.2023). - Назва з екрана.
6. Майданюк В. П., Грицишин В. О. Захист біомедичних зображень від несанкціонованого доступу. / Інформаційні технології і автоматизація – 2024 / Матеріали XVII міжнародної науково-практичної конференції. Одеса, 31 жовтня - 1 листопада 2024 р. - Одеса, Видавництво ОНТУ, 2024 р. – С. 200-201.
7. Грицишин В. О., Майданюк В. П. Використання стеганографії для захисту рентгенівських знімків / Електронні інформаційні ресурси: створення, використання, доступ. Збірник матеріалів Міжнародної науково-практичної Інтернет конференції 20-21 листопада 2024 р. – Суми/Вінниця: НІКО/ КЗВО «Вінницька академія безперервної освіти», 2024. – 336 с.

5. Технічні вимоги

- середовище розробки – Microsoft Visual Studio;
- мова програмування – C++;
- метод захисту – стеганографія, шифрування;
- ступінь захисту – середній;

6. Конструктивні вимоги

Графічна та текстова документація повинна відповідати діючим стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

1. Пояснювальна записка до МКР;
2. Технічне завдання;
3. Лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки:

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи
1	Обґрунтування вибору методу розробки та постановка задач	20.09.24 – 02.10.24
2	Розробка архітектури та алгоритмів програмного продукту	03.10.24 – 23.10.24
3	Аналіз і вибір мови програмування та середовища розробки	16.10.24 – 23.10.24
4	Розробка програмного продукту	24.10.24 – 13.11.24
5	Тестування програми	14.11.24 – 21.11.24
6	Розробка економічної частини	17.11.24 –25.11.24
7	Оформлення матеріалів до захисту МКР	22.11.24 – 30.11.24

10. Порядок контролю та прийняття

Порядок контролю і приймання роботи регламентується відповідними документами ВНТУ і державними стандартами.

Додаток Б

(обов'язковий)

ПРОТОКОЛ
ПЕРЕВІРКИ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Методи та програмні засоби захисту біомедичних зображень від несанкціонованого доступу.

Тип роботи: магістерська кваліфікаційна робота

Підрозділ : кафедра програмного забезпечення, ФІТКІ

Керівник: Майданюк В. П.

Показники звіту подібності

Turnitin	
Оригінальність	91 %
Загальна схожість	9 %

Аналіз звіту подібності

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

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

Автор _____ Грицишин В. О
(підпис) (прізвище, ініціали)

Опис прийнятого рішення

_____

Особа, відповідальна за перевірку _____ Черноволик Г. О.
(підпис) (прізвище, ініціали)

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

Додаток В

(довідниковий)

Лістинг програмного коду

```
//Модуль шифрування зображення

private: System::Void button26_Click(System::Object^ sender, System::EventArgs^ e) {

	Bitmap^ image1;
	Bitmap^ image2;
	Bitmap^ image3;
	Color newColor1;
	String^ name;

	if (openFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK)

	{
		image1 = gcnew Bitmap(openFileDialog1->FileName, true);
		image2 = gcnew Bitmap(openFileDialog1->FileName, true);
	}

	for (int x = 0; x < image2->Width; x++)
	{
		for (int y = 0; y < image2->Height; y++)
		{
			Color newColor = Color::FromArgb(0, 0, 0);
			image2->SetPixel(x, y, newColor);
		}
	}

	Byte r, g, b;
	int  i, j, k, l, t, n,jt, lt, nt, it, kt, tt, mod;

	n = 8;

	i = 10;
	j = 10;
	k = 13;
	l = 13;
	t = 201;
	n = 201;

	it = 10;
	jt = 10;
	kt = 13;
	lt = 13;
	tt = 201;
	nt = 201;

	//Модуль шифру
	mod = 2;

	while (mod  < (image1->Width * image1->Height)) mod = mod * 2;

	for (int y = 0; y < image1->Height; y++)
	{

		for (int x = 0; x < image1->Width; x++)
		{
```

```
                Color pixelColor = image1->GetPixel(x, y);

                gwhr[j][i] = pixelColor.R;
                gwhr[j][i] = (gwhr[j][i] ^ jt ^ lt ^ nt ^ it ^ kt ^ tt)&0x00ff;
                gwhg[l][k] = pixelColor.G;
                gwhg[l][k] = (gwhg[l][k] ^ jt ^ lt ^ nt ^ it ^ kt ^ tt) & 0x00ff;
                gwhb[n][t] = pixelColor.B;
                gwhb[n][t] = (gwhb[n][t] ^ jt ^ lt ^ nt ^ it ^ kt ^ tt) & 0x00ff;


                jt = (13 * jt + 11) % mod;
                while (jt >= (image1->Width * image1->Height)) jt = (13 * jt + 11) % mod;
                i = (jt / image1->Width);
                j = jt - i * image1->Width;


                lt = (25 * lt + 19) % mod;
                while (lt  >= (image1->Width * image1->Height)) lt = (25 * lt + 19) % mod;
                k = (lt / image1->Width);
                l = lt - k * image1->Width;

                nt = (29 * nt + 29) % mod;
                while (nt  >= (image1->Width * image1->Height)) nt = (29 * nt + 29) % mod;
                t = (nt / image1->Width);
                n = nt - t * image1->Width;

        }

}


//-----Виведення відновленого зображення----
for (int x = 0; x < image1->Width; x++)
{
        for (int y = 0; y < image1->Height; y++)
        {

                t = gwhr[x][y];
                if (t < 0)  t = 0;
                if (t > 255)  t = 255;
                r = t;

                t = gwhg[x][y];
                if (t < 0)  t = 0;
                if (t > 255)  t = 255;
                g = t;

                t = gwhb[x][y];
                if (t < 0)  t = 0;
                if (t > 255)  t = 255;
                b = t;

                newColor1 = Color::FromArgb(r, g, b);
                image2->SetPixel(x, y, newColor1);

        }
}

pictureBox2->Width = image2->Width;
pictureBox2->Height = image2->Height;
pictureBox2->Image = image2;
```

```
	}

//Модуль дешифрування зображення

private: System::Void button27_Click(System::Object^ sender, System::EventArgs^ e) {
	Bitmap^ image1;
	Bitmap^ image2;
	Bitmap^ image3;
	Color newColor1;
	String^ name;

	if (openFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK)

	{
		image1 = gcnew Bitmap(openFileDialog1->FileName, true);

		image2 = gcnew Bitmap(openFileDialog1->FileName, true);
	}

	for (int x = 0; x < image2->Width; x++)
	{
		for (int y = 0; y < image2->Height; y++)
		{



			Color newColor = Color::FromArgb(0, 0, 0);
			image2->SetPixel(x, y, newColor);

		}

	}


	Byte r, g, b;
	int  i, j, k, l, t, n, jt, lt, nt, it,kt,tt, mod;



	n = 8;

	i = 10;
	j = 10;
	k = 13;
	l = 13;
	t = 201;
	n = 201;

	it = 10;
	jt = 10;
	kt = 13;
	lt = 13;
	tt = 201;
	nt = 201;

	//Модуль шифру
	mod = 2;

	while (mod < (image1->Width * image1->Height)) mod = mod * 2;

	for (int y = 0; y < image1->Height; y++)
	{
```

```
		for (int x = 0; x < image1->Width; x++)
		{
		Color pixelColor = image1->GetPixel(j, i);

			gwhr[x][y] = pixelColor.R;
			gwhr[x][y] = (gwhr[x][y] ^ jt ^ lt ^ nt ^ it ^ kt ^ tt) & 0x00ff;
		pixelColor = image1->GetPixel(l, k);
			gwhg[x][y] = pixelColor.G;
			gwhg[x][y] = (gwhg[x][y] ^ jt ^ lt ^ nt ^ it ^ kt ^ tt) & 0x00ff;
		pixelColor = image1->GetPixel(n, t);
			gwhb[x][y] = pixelColor.B;
			gwhb[x][y] = (gwhb[x][y] ^ jt ^ lt ^ nt ^ it ^ kt ^ tt) & 0x00ff;

			jt = (13 * jt + 11) % mod;
			while (jt >= (image1->Width * image1->Height)) jt = (13 * jt + 11) % mod;
			i = (jt / image1->Width);
			j = jt - i * image1->Width;


			lt = (25 * lt + 19) % mod;
			while (lt >= (image1->Width * image1->Height)) lt = (25 * lt + 19) % mod;
			k = (lt / image1->Width);
			l = lt - k * image1->Width;

			nt = (29 * nt + 29) % mod;
			while (nt >= (image1->Width * image1->Height)) nt = (29 * nt + 29) % mod;
			t = (nt / image1->Width);
			n = nt - t * image1->Width;


		}

}


//-----Виведення відновленого зображення----
for (int x = 0; x < image1->Width; x++)
{
		for (int y = 0; y < image1->Height; y++)
		{

			t = gwhr[x][y];
			if (t < 0)  t = 0;
			if (t > 255)  t = 255;
			r = t;

			t = gwhg[x][y];
			if (t < 0)  t = 0;
			if (t > 255)  t = 255;
			g = t;

			t = gwhb[x][y];
			if (t < 0)  t = 0;
			if (t > 255)  t = 255;
			b = t;

			newColor1 = Color::FromArgb(r, g, b);
			image2->SetPixel(x, y, newColor1);

		}
}

pictureBox2->Width = image2->Width;
```

```
	pictureBox2->Height = image2->Height;

	pictureBox2->Image = image2;


}


	//Модуль приховування молодших біт пікселів зображення

	private: System::Void button28_Click(System::Object^ sender, System::EventArgs^ e) {

		Bitmap^ image1;
		Bitmap^ image2;
		Bitmap^ image3;
		Color newColor1;
		String^ name;

		if (openFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK)
		{
			image1 = gcnew Bitmap(openFileDialog1->FileName, true);
			image2 = gcnew Bitmap(openFileDialog1->FileName, true);
		}
		for (int y = 0; y < image1->Height; y++)
		{
			for (int x = 0; x < image1->Width; x++)
			{



				Color newColor = Color::FromArgb(0, 0, 0);
				image2->SetPixel(x, y, newColor);

			}

		}



		Byte r, g, b;
		int  i, j, k, l, t, n, it, jt, kt, lt, tt, nt, mod;


		n = 8;

		i = 10;
		j = 10;
		k = 13;
		l = 13;
		t = 201;
		n = 201;

		it = 10;
		jt = 10;
		kt = 13;
		lt = 13;
		tt = 201;
		nt = 201;

		//Модуль шифру
		mod = 2;

		while (mod < (image1->Width * image1->Height)) mod = mod * 2;
```

```
	for (int x = 0; x < image1->Width; x++)
	{

		for (int y = 0; y < image1->Height; y++)
		{
			Color pixelColor = image1->GetPixel(x, y);

			gwhr[x][y] = pixelColor.R ;
			gwhg[x][y] = pixelColor.G;
			gwhb[x][y] = pixelColor.B;

		}
	}

	for (int y = 0; y < image1->Height; y++)
	{

		for (int x = 0; x < image1->Width; x++)
		{
			Color pixelColor = image1->GetPixel(x, y);
			//image2->SetPixel(i, j, pixelColor);
			gwhr[j][i] = (gwhr[j][i]&0xE0)|((pixelColor.R ^ jt ^ lt ^ nt ^ it ^ kt ^ tt) &0x1F);
			gwhg[l][k] = (gwhg[l][k] & 0xE0) | ((pixelColor.G ^ jt ^ lt ^ nt ^ it ^ kt ^ tt) &
0x1F);
			gwhb[n][t] = (gwhb[n][t] & 0xE0) | ((pixelColor.B ^ jt ^ lt ^ nt ^ it ^ kt ^ tt) &
0x1F);
			//i = (21 * i + 13) % image1->Width;

			jt = (13 * jt + 11) % mod;
			while (jt >= (image1->Width * image1->Height)) jt = (13 * jt + 11) % mod;
			i = (jt / image1->Width);
			j = jt - i * image1->Width;

			//			k = (17 * k + 23) % image1->Width;
			lt = (25 * lt + 19) % mod;
			while (lt >= (image1->Width * image1->Height)) lt = (25 * lt + 19) % mod;
			k = (lt / image1->Width);
			l = lt - k * image1->Width;

			//		t = (33 * t + 31) % image1->Width;
			nt = (29 * nt + 29) % mod;
			while (nt >= (image1->Width * image1->Height)) nt = (29 * nt + 29) % mod;
			t = (nt / image1->Width);
			n = nt - t * image1->Width;


		}

	}


	//-----Виведення відновленого зображення----
	for (int x = 0; x < image1->Width; x++)
	{
		for (int y = 0; y < image1->Height; y++)
		{

			t = gwhr[x][y];
			if (t < 0)  t = 0;
			if (t > 255)  t = 255;
			r = t;
```

```
                        t = gwhg[x][y];
                        if (t < 0)  t = 0;
                        if (t > 255)  t = 255;
                        g = t;

                        t = gwhb[x][y];
                        if (t < 0)  t = 0;
                        if (t > 255)  t = 255;
                        b = t;

                        newColor1 = Color::FromArgb(r, g, b);
                        image2->SetPixel(x, y, newColor1);

                }
        }


        pictureBox2->Image = image2;
}


//Модуль відновлення молодших біт пікселів зображення

private: System::Void button29_Click(System::Object^ sender, System::EventArgs^ e) {

        Bitmap^ image1;
        Bitmap^ image2;
        Bitmap^ image3;
        Color newColor1;
        String^ name;

        if (openFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK)
        {
                image1 = gcnew Bitmap(openFileDialog1->FileName, true);
                image2 = gcnew Bitmap(openFileDialog1->FileName, true);
        }
        for (int y = 0; y < image1->Height; y++)
        {
                for (int x = 0; x < image1->Width; x++)
                {

                        Color newColor = Color::FromArgb(0, 0, 0);
                        image2->SetPixel(x, y, newColor);

                }

        }


        Byte r, g, b;
        int  i, j, k, l, t, n, it, jt, kt, lt, tt, nt, mod;


        n = 8;

        i = 10;
        j = 10;
        k = 13;
        l = 13;
        t = 201;
        n = 201;

        it = 10;
```

```
jt = 10;
kt = 13;
lt = 13;
tt = 201;
nt = 201;

//Модуль шифру
mod = 2;

while (mod < (image1->Width * image1->Height)) mod = mod * 2;

for (int y = 0; y < image1->Height; y++)
{

	for (int x = 0; x < image1->Width; x++)
	{
		Color pixelColor = image1->GetPixel(x, y);

		gwhr[x][y] = pixelColor.R;
		gwhg[x][y] = pixelColor.G;
		gwhb[x][y] = pixelColor.B;

	}
}

for (int y = 0; y < image1->Height; y++)
{

	for (int x = 0; x < image1->Width; x++)
	{
		Color pixelColor = image1->GetPixel(j, i);
		//image2->SetPixel(i, j, pixelColor);
		gwhr[x][y] = (gwhr[x][y] & 0xE0) | ((pixelColor.R ^ jt ^ lt ^ nt ^ it ^ kt ^ tt) &
0x1F);
		pixelColor = image1->GetPixel(l, k);
		gwhg[x][y] = (gwhg[x][y] & 0xE0) | ((pixelColor.G ^ jt ^ lt ^ nt ^ it ^ kt ^ tt) &
0x1F);
		pixelColor = image1->GetPixel(n, t);
		gwhb[x][y] = (gwhb[x][y] & 0xE0) | ((pixelColor.B ^ jt ^ lt ^ nt ^ it ^ kt ^ tt) &
0x1F);
		//i = (21 * i + 13) % image1->Width;

		jt = (13 * jt + 11) % mod;
		while (jt >= (image1->Width * image1->Height)) jt = (13 * jt + 11) % mod;
		i = (jt / image1->Width);
		j = jt - i * image1->Width;

		//		k = (17 * k + 23) % image1->Width;
		lt = (25 * lt + 19) % mod;
		while (lt >= (image1->Width * image1->Height)) lt = (25 * lt + 19) % mod;
		k = (lt / image1->Width);
		l = lt - k * image1->Width;

		//	t = (33 * t + 31) % image1->Width;
		nt = (29 * nt + 29) % mod;
		while (nt >= (image1->Width * image1->Height)) nt = (29 * nt + 29) % mod;
		t = (nt / image1->Width);
		n = nt - t * image1->Width;


	}

}
```

```
	//-----Виведення відновленого зображення----
	for (int x = 0; x < image1->Width; x++)
	{
		for (int y = 0; y < image1->Height; y++)
		{

			t = gwhr[x][y];
			if (t < 0)  t = 0;
			if (t > 255)  t = 255;
			r = t;

			t = gwhg[x][y];
			if (t < 0)  t = 0;
			if (t > 255)  t = 255;
			g = t;

			t = gwhb[x][y];
			if (t < 0)  t = 0;
			if (t > 255)  t = 255;
			b = t;

			newColor1 = Color::FromArgb(r, g, b);
			image2->SetPixel(x, y, newColor1);

		}
	}


	pictureBox2->Image = image2;
}


//Модуль приховування в просторовій області

private: System::Void button30_Click(System::Object^ sender, System::EventArgs^ e) {

	Bitmap^ image1;
	Bitmap^ image2;
	Bitmap^ image3;
	Color newColor1;
	String^ name;

	MessageBox::Show("Виберіть контейнер. Розмір контейнера повинен бути у два рази більшим файлу, що приховується");
	if (openFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK)
				image1 = gcnew Bitmap(openFileDialog1->FileName, true);

	MessageBox::Show("Виберіть файл для приховування. Розмір файлу повинен бути у два рази меншим файлу контейнера");
	if (openFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK)

		image2 = gcnew Bitmap(openFileDialog1->FileName, true);

	image3 = gcnew Bitmap(image1->Width, image1->Height);
	if ((image2->Width * image2->Height) >= ((image1->Width * image1->Height) / 2))
	{
		MessageBox::Show("Виберіть інший контейнер або файл для приховування. Розмір контейнера повинен бути у два рази більшим");
		return;
	}
	for (int y = 0; y < image3->Height; y++)
```

```
{
	for (int x = 0; x < image3->Width; x++)
	{

		Color newColor = Color::FromArgb(0, 0, 0);
		image3->SetPixel(x, y, newColor);

	}

}



Byte r, g, b;
int  i, j, k, l, t, n, it, jt, kt, lt, tt, nt, mod, xt;



n = 8;

i = 10;
j = 10;
k = 13;
l = 13;
t = 201;
n = 201;

it = 10;
jt = 10;
kt = 13;
lt = 13;
tt = 201;
nt = 201;

//Модуль шифру
mod = 2;

while (mod < (image1->Width * image1->Height)) mod = mod * 2;

for (int y = 0; y < image1->Height; y++)
{

	for (int x = 0; x < image1->Width; x++)
	{
		Color pixelColor = image1->GetPixel(x, y);

		gwhr[x][y] = pixelColor.R;
		gwhg[x][y] = pixelColor.G;
		gwhb[x][y] = pixelColor.B;

	}
}



for (int y = 0; y < image2->Height; y++)
{

	for (int x = 0; x < image2->Width; x++)
	{
		Color pixelColor = image2->GetPixel(x, y);
		//image2->SetPixel(i, j, pixelColor);
		gwhr[j][i] = (gwhr[j][i] & 0xF0) | ((pixelColor.R ^ jt ^ lt ^ nt ^ it ^ kt ^ tt) & 0x0F);
```

```
                        jt = (13 * jt + 11) % mod;
                        while (jt >= (image1->Width * image1->Height)) jt = (13 * jt + 11) % mod;
                        i = (jt / image1->Width);
                        j = jt - i * image1->Width;
                        gwhr[j][i] = (gwhr[j][i] & 0xF0) | (((pixelColor.R ^ jt ^ lt ^ nt ^ it ^ kt ^ tt)>>4) &
0x0F);

                        gwhg[l][k] = (gwhg[l][k] & 0xF0) | ((pixelColor.G ^ jt ^ lt ^ nt ^ it ^ kt ^ tt) &
0x0F);
                        lt = (25 * lt + 19) % mod;
                        while (lt >= (image1->Width * image1->Height)) lt = (25 * lt + 19) % mod;
                        k = (lt / image1->Width);
                        l = lt - k * image1->Width;
                        gwhg[l][k] = (gwhg[l][k] & 0xF0) | (((pixelColor.G ^ jt ^ lt ^ nt ^ it ^ kt ^ tt)>>4) &
0x0F);

                        gwhb[n][t] = (gwhb[n][t] & 0xF0) | ((pixelColor.B ^ jt ^ lt ^ nt ^ it ^ kt ^ tt) &
0x0F);
                        nt = (29 * nt + 29) % mod;
                        while (nt >= (image1->Width * image1->Height)) nt = (29 * nt + 29) % mod;
                        t = (nt / image1->Width);
                        n = nt - t * image1->Width;
                        gwhb[n][t] = (gwhb[n][t] & 0xF0) | (((pixelColor.B ^ jt ^ lt ^ nt ^ it ^ kt ^ tt)>>4) &
0x0F);

                        jt = (13 * jt + 11) % mod;
                        while (jt >= (image1->Width * image1->Height)) jt = (13 * jt + 11) % mod;
                        i = (jt / image1->Width);
                        j = jt - i * image1->Width;

                        //                      k = (17 * k + 23) % image1->Width;
                        lt = (25 * lt + 19) % mod;
                        while (lt >= (image1->Width * image1->Height)) lt = (25 * lt + 19) % mod;
                        k = (lt / image1->Width);
                        l = lt - k * image1->Width;

                        //          t = (33 * t + 31) % image1->Width;
                        nt = (29 * nt + 29) % mod;
                        while (nt >= (image1->Width * image1->Height)) nt = (29 * nt + 29) % mod;
                        t = (nt / image1->Width);
                        n = nt - t * image1->Width;

                }

        }


        //-----Виведення відновленого зображення----
        for (int x = 0; x < image1->Width; x++)
        {
                for (int y = 0; y < image1->Height; y++)
                {

                        t = gwhr[x][y];
                        if (t < 0)  t = 0;
                        if (t > 255)  t = 255;
                        r = t;

                        t = gwhg[x][y];
                        if (t < 0)  t = 0;
                        if (t > 255)  t = 255;
                        g = t;
```

```
                        t = gwhb[x][y];
                        if (t < 0)  t = 0;
                        if (t > 255)  t = 255;
                        b = t;

                        newColor1 = Color::FromArgb(r, g, b);
                        image3->SetPixel(x, y, newColor1);

                }
        }


        pictureBox2->Image = image3;

}

//Модуль відновлення зображення в просторовій області

private: System::Void button31_Click(System::Object^ sender, System::EventArgs^ e) {

        Bitmap^ image1;
        Bitmap^ image2;
        Bitmap^ image3;
        Color newColor1;
        String^ name;

        if (openFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK)

                image1 = gcnew Bitmap(openFileDialog1->FileName, true);

        image2 = gcnew Bitmap(image1->Width, image1->Height);

        for (int y = 0; y < ((image1->Height) / 1); y++)
        {
                for (int x = 0; x < ((image1->Width) / 1); x++)
                {

                        Color newColor = Color::FromArgb(0, 0, 0);
                        image2->SetPixel(x, y, newColor);

                }

        }


        Byte r, g, b;
        int  i, j, k, l, t, n, it, jt, kt, lt, tt, nt, mod;


        n = 8;

        i = 10;
        j = 10;
        k = 13;
        l = 13;
        t = 201;
        n = 201;

        it = 10;
        jt = 10;
        kt = 13;
        lt = 13;
        tt = 201;
```

```
	nt = 201;

	//Модуль шифру
	mod = 2;

	while (mod < (image1->Width * image1->Height)) mod = mod * 2;

	for (int y = 0; y < image1->Height; y++)
	{

		for (int x = 0; x < image1->Width; x++)
		{

			gwhr[x][y] = 0;
			gwhg[x][y] = 0;
			gwhb[x][y] = 0;

		}
	}

	for (int y = 0; y < ((image1->Height) / 2); y++)
	{

		for (int x = 0; x < ((image1->Width) / 2); x++)
		{
			Color pixelColor = image1->GetPixel(j, i);
			//image2->SetPixel(i, j, pixelColor);
			gwhr[x][y] = (gwhr[x][y] & 0x00) | ((pixelColor.R ^ jt ^ lt ^ nt ^ it ^ kt ^ tt) &
0x0F);
			jt = (13 * jt + 11) % mod;
			while (jt >= (image1->Width * image1->Height)) jt = (13 * jt + 11) % mod;
			i = (jt / image1->Width);
			j = jt - i * image1->Width;
			pixelColor = image1->GetPixel(j, i);
			gwhr[x][y] = (gwhr[x][y] & 0x0F) | (((pixelColor.R << 4) ^ jt ^ lt ^ nt ^ it ^ kt ^ tt)
& 0x00F0);

			pixelColor = image1->GetPixel(l, k);
			gwhg[x][y] = (gwhg[x][y] & 0x00) | ((pixelColor.G ^ jt ^ lt ^ nt ^ it ^ kt ^ tt) &
0x0F);
			lt = (25 * lt + 19) % mod;
			while (lt >= (image1->Width * image1->Height)) lt = (25 * lt + 19) % mod;
			k = (lt / image1->Width);
			l = lt - k * image1->Width;
			pixelColor = image1->GetPixel(l, k);
			gwhg[x][y] = (gwhg[x][y] & 0x0F) | (((pixelColor.G<< 4) ^ jt ^ lt ^ nt ^ it ^ kt ^ tt)
& 0x00F0);

			pixelColor = image1->GetPixel(n, t);
			gwhb[x][y] = (gwhb[x][y] & 0x00) | ((pixelColor.B ^ jt ^ lt ^ nt ^ it ^ kt ^ tt) &
0x0F);
			nt = (29 * nt + 29) % mod;
			while (nt >= (image1->Width * image1->Height)) nt = (29 * nt + 29) % mod;
			t = (nt / image1->Width);
			n = nt - t * image1->Width;
			pixelColor = image1->GetPixel(n, t);
			gwhb[x][y] = (gwhb[x][y] & 0x0F) | (((pixelColor.B<< 4) ^ jt ^ lt ^ nt ^ it ^ kt ^ tt)
& 0x00F0);
			//i = (21 * i + 13) % image1->Width;

			jt = (13 * jt + 11) % mod;
			while (jt >= (image1->Width * image1->Height)) jt = (13 * jt + 11) % mod;
			i = (jt / image1->Width);
```

```
                        j = jt - i * image1->Width;

                        //                      k = (17 * k + 23) % image1->Width;
                        lt = (25 * lt + 19) % mod;
                        while (lt >= (image1->Width * image1->Height)) lt = (25 * lt + 19) % mod;
                        k = (lt / image1->Width);
                        l = lt - k * image1->Width;

                        //          t = (33 * t + 31) % image1->Width;
                        nt = (29 * nt + 29) % mod;
                        while (nt >= (image1->Width * image1->Height)) nt = (29 * nt + 29) % mod;
                        t = (nt / image1->Width);
                        n = nt - t * image1->Width;

                }

        }
        pictureBox2->Width = (image1->Width) / 2;
        pictureBox2->Height = (image1->Height) / 2;


        //-----Виведення відновленого зображення----
        for (int x = 0; x < (image1->Width)/2; x++)
        {
                for (int y = 0; y < (image1->Height)/2; y++)
                {

                        t = gwhr[x][y];
                        if (t < 0)  t = 0;
                        if (t > 255)  t = 255;
                        r = t;

                        t = gwhg[x][y];
                        if (t < 0)  t = 0;
                        if (t > 255)  t = 255;
                        g = t;

                        t = gwhb[x][y];
                        if (t < 0)  t = 0;
                        if (t > 255)  t = 255;
                        b = t;

                        newColor1 = Color::FromArgb(r, g, b);
                        image2->SetPixel(x, y, newColor1);

                }
        }


        pictureBox2->Image = image2;
}


//Модуль приховування зображення в частотній області

private: System::Void button13_Click(System::Object^ sender, System::EventArgs^ e) {


        Bitmap^ image1;
        Bitmap^ image2;
        Bitmap^ image3;
        Color newColor1;
        String^ name;
```

```
MessageBox::Show("Виберіть контейнер. Розмір контейнера повинен бути у два рази більшим файлу, що приховується");
if (openFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK)
	image1 = gcnew Bitmap(openFileDialog1->FileName, true);

MessageBox::Show("Виберіть файл для приховування. Розмір файлу повинен бути у два рази меншим файлу контейнера");
if (openFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK)

	image3 = gcnew Bitmap(openFileDialog1->FileName, true);


if ((image3->Width * image3->Height) > ((image1->Width * image1->Height) / 2))
{
	MessageBox::Show("Виберіть інший контейнер або файл для приховування. Розмір контейнера повинен бути у два рази більшим");
	return;
}

image2 = gcnew Bitmap("lena512color.tiff", true);

for (int x = 0; x < image1->Width; x++)
{
	for (int y = 0; y < image1->Height; y++)
	{



		Color newColor = Color::FromArgb(0, 0, 0);
		image2->SetPixel(x, y, newColor);

	}

}

if (saveFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK)

	name = saveFileDialog1->FileName;


string name1 = "";
for (int i = 0; i < name->Length; i++)
	name1 += (char)name[i];
   name1 = name1 + ".wht";

//Відкриття файлу для запису в бінарному режимі
ofstream out_file(name1, ios::binary);

//Перевірка на помилку відкриття



if (!out_file) {
	cout << "\nCouldn't open file";
	return;
}


Byte r, g, b;
int  i, j, k, l, t, n, it, jt, kt, lt, tt, nt, itt, jtt, ktt, ltt, ttt, ntt, mod, xt;


short int a[8][8] = { {1,1,1,1,1,1,1,1},
```

```
                                        {1,1,1,1,-1,-1,-1,-1},
                                        {1,1,-1,-1,-1,-1,1,1},
                                        {1,1,-1,-1,1,1,-1,-1},
                                        {1,-1,-1,1,1,-1,-1,1},
                                        {1,-1,-1,1,-1,1,1,-1},
                                        {1,-1,1,-1,-1,1,-1,1},
                                        {1,-1,1,-1,1,-1,1,-1} };
short int tr[8][8];
short int tr1[8][8];
short int tg[8][8];
short int tg1[8][8];
short int tb[8][8];
short int tb1[8][8];


n = 8;

it = 10;
jt = 10;
kt = 13;
lt = 13;
tt = 201;
nt = 201;

itt = 10;
jtt = 10;
ktt = 13;
ltt = 13;
ttt = 201;
ntt = 201;


//Модуль шифру
mod = 2;

while (mod < (image1->Width * image1->Height)) mod = mod * 2;


for (int y = 0; y < image1->Height; y++)
{

        for (int x = 0; x < image1->Width; x++)
        {
                Color pixelColor = image1->GetPixel(x, y);

                gwhr[x][y] = pixelColor.R;
                gwhg[x][y] = pixelColor.G;
                gwhb[x][y] = pixelColor.B;

        }
}
for (int y = 0; y < image3->Height; y++)
{

        for (int x = 0; x < image3->Width; x++)
        {
                Color pixelColor = image3->GetPixel(x, y);

                gwhr2[x][y] = pixelColor.R;
                gwhg2[x][y] = pixelColor.G;
                gwhb2[x][y] = pixelColor.B;

        }
```

```
	}

	for (int k = 0; k < n; k++) {
		for (int l = 0; l < n; l++)
		{

			kv8[k][l] = Convert::ToInt32(dataGridView2->Rows[k]->Cells[l]->Value);
			out_file.write((char*)&kv8[k][l], 1);
			kv8[k][l] = Convert::ToInt32(dataGridView5->Rows[k]->Cells[l]->Value);
			out_file.write((char*)&kv8[k][l], 1);
			kv8[k][l] = Convert::ToInt32(dataGridView6->Rows[k]->Cells[l]->Value);
			out_file.write((char*)&kv8[k][l], 1);

		}
	}

	for (int x = 0; x <= 63; x++)
	{
		for (int y = 0; y <= 63; y++)
		{

			//------RRRRRRR-------

			for (int k = 0; k < n; k++) {
				for (int l = 0; l < n; l++)
				{

					kv8[k][l] = Convert::ToInt32(dataGridView2->Rows[k]->Cells[l]-
>Value);
				}
			}

			for (int i = 0; i < n; i++) {
				for (int j = 0; j < n; j++)
				{
					tr[i][j] = gwhr[8 * x + j][8 * y + i];
				}
			}

			for (j = 0; j < n; j++)
			{
				for (int k = 0; k < n; k++) {
					t = 0;
					for (int l = 0; l < n; l++)
					{
						t = t + tr[j][l] * a[k][l];
					}
					tr1[j][k] = t;
				}
			}

			for (int i = 0; i < n; i++)
			{
				for (k = 0; k < n; k++) {
					t = 0;
					for (l = 0; l < n; l++) {
						t = t + tr1[l][i] * a[k][l];
					}
					tr[k][i] = t / 1;


				}
			}
```

```
for (k = 0; k < n; k++) {
	for (l = 0; l < n; l++) {

		tr[k][l] = tr[k][l] / kv8[k][l];
		gwhr1[8 * x + k][8 * y + l] = tr[k][l];

	}
}

//-------GGGGGGGGGG--------

for (int k = 0; k < n; k++) {
	for (int l = 0; l < n; l++)
	{

		kv8[k][l] = Convert::ToInt32(dataGridView5->Rows[k]->Cells[l]-
>Value);
	}
}

for (int i = 0; i < n; i++) {
	for (int j = 0; j < n; j++)
	{
		tg[i][j] = gwhg[8 * x + j][8 * y + i];
	}
}

for (j = 0; j < n; j++)
{
	for (int k = 0; k < n; k++) {
		t = 0;
		for (int l = 0; l < n; l++)
		{
			t = t + tg[j][l] * a[k][l];
		}
		tg1[j][k] = t;
	}
}

for (int i = 0; i < n; i++)
{
	for (k = 0; k < n; k++) {
		t = 0;
		for (l = 0; l < n; l++) {
			t = t + tg1[l][i] * a[k][l];
		}
		tg[k][i] = t / 1;


	}
}

for (k = 0; k < n; k++) {
	for (l = 0; l < n; l++) {

		tg[k][l] = tg[k][l] / kv8[k][l];
		gwhg1[8 * x + k][8 * y + l] = tg[k][l];


	}
}
//-----BBBBBBB----------
```

```
                        for (int k = 0; k < n; k++) {
                                for (int l = 0; l < n; l++)
                                {

                                        kv8[k][l] = Convert::ToInt32(dataGridView6->Rows[k]->Cells[l]-
>Value);
                                }
                        }

                        for (int i = 0; i < n; i++) {
                                for (int j = 0; j < n; j++)
                                {
                                        tb[i][j] = gwhb[8 * x + j][8 * y + i];
                                }
                        }

                        for (j = 0; j < n; j++)
                        {
                                for (int k = 0; k < n; k++) {
                                        t = 0;
                                        for (int l = 0; l < n; l++)
                                        {
                                                t = t + tb[j][l] * a[k][l];
                                        }
                                        tb1[j][k] = t;
                                }
                        }

                        for (int i = 0; i < n; i++)
                        {
                                for (k = 0; k < n; k++) {
                                        t = 0;
                                        for (l = 0; l < n; l++) {
                                                t = t + tb1[l][i] * a[k][l];
                                        }
                                        tb[k][i] = t / 1;



                                }
                        }


                        for (k = 0; k < n; k++) {
                                for (l = 0; l < n; l++) {

                                        tb[k][l] = tb[k][l] / kv8[k][l];
                                        gwhb1[8 * x + k][8 * y + l] = tb[k][l];


                                }
                        }
                }
        }

        //Приховування в коефіцієнтах
        for (int y = 0; y < image3->Height; y++)
                        {

                                for (int x = 0; x < image3->Width; x++)
                                {
```

```
                                        gwhr1[jtt][itt] = (gwhr1[jtt][itt] & 0xFFF0) | ((gwhr2[x][y] ^ jt ^ lt
^ nt ^ it ^ kt ^ tt) & 0x000F);
                                        jt = (13 * jt + 11) % mod;
                                        while (jt >= (image1->Width * image1->Height)) jt = (13 * jt +
11) % mod;
                                        itt = (jt / image1->Width);
                                        jtt = jt - itt * image1->Width;
                                        gwhr1[jtt][itt] = (gwhr1[jtt][itt] & 0xFFF0) | (((gwhr2[x][y] ^ jt ^
lt ^ nt ^ it ^ kt ^ tt) >> 4) & 0x000F);
                                        gwhg1[ltt][ktt] = (gwhg1[ltt][ktt] & 0xFFF0) | ((gwhg2[x][y] ^ jt ^
lt ^ nt ^ it ^ kt ^ tt) & 0x000F);
                                        lt = (25 * lt + 19) % mod;
                                        while (lt >= (image1->Width * image1->Height)) lt = (25 * lt +
19) % mod;
                                        ktt = (lt / image1->Width);
                                        ltt = lt - ktt * image1->Width;
                                        gwhg1[ltt][ktt] = (gwhg1[ltt][ktt] & 0xFFF0) | (((gwhg2[x][y] ^ jt
^ lt ^ nt ^ it ^ kt ^ tt) >> 4) & 0x000F);
                                        gwhb1[ntt][ttt] = (gwhb1[ntt][ttt] & 0xFFF0) | ((gwhb2[x][y] ^ jt ^
lt ^ nt ^ it ^ kt ^ tt) & 0x000F);
                                        nt = (29 * nt + 29) % mod;
                                        while (nt >= (image1->Width * image1->Height)) nt = (29 * nt +
29) % mod;
                                        ttt = (nt / image1->Width);
                                        ntt = nt - ttt * image1->Width;
                                        gwhb1[ntt][ttt] = (gwhb1[ntt][ttt] & 0xFFF0) | (((gwhb2[x][y] ^ jt
^ lt ^ nt ^ it ^ kt ^ tt) >> 4) & 0x000F);
                                        jt = (13 * jt + 11) % mod;
                                        while (jt >= (image1->Width * image1->Height)) jt = (13 * jt +
11) % mod;
                                        itt = (jt / image1->Width);
                                        jtt = jt - itt * image1->Width;
                                        lt = (25 * lt + 19) % mod;
                                        while (lt >= (image1->Width * image1->Height)) lt = (25 * lt +
19) % mod;
                                        ktt = (lt / image1->Width);
                                        ltt = lt - ktt * image1->Width;
                                        nt = (29 * nt + 29) % mod;
                                        while (nt >= (image1->Width * image1->Height)) nt = (29 * nt +
29) % mod;
                                        ttt = (nt / image1->Width);
                                        ntt = nt - ttt * image1->Width;
                                }
                        }
//RRRRRRRRRRR
                        for (int x = 0; x <= 63; x++)
                        {
                                for (int y = 0; y <= 63; y++)
                                {
                                        for (int k = 0; k < n; k++) {
```

```
                for (int l = 0; l < n; l++)
                {

                        kv8[k][l]   =   Convert::ToInt32(dataGridView2-
>Rows[k]->Cells[l]->Value);
                }
        }
        for (k = 0; k < n; k++) {
                for (l = 0; l < n; l++) {

                        tr[k][l] = gwhr1[8 * x + k][8 * y + l];
                        out_file.write((char*)&tr[k][l], 2);
                        tr[k][l] = tr[k][l]* kv8[k][l];


                }
        }
        for (j = 0; j < n; j++)
        {
                for (int k = 0; k < n; k++) {
                        t = 0;
                        for (int l = 0; l < n; l++)
                        {
                                t = t + tr[l][j] * a[k][l];
                        }
                        tr1[k][j] = t;
                }
        }

        for (int i = 0; i < n; i++)
        {
                for (k = 0; k < n; k++) {
                        t = 0;
                        for (l = 0; l < n; l++) {
                                t = t + tr1[i][l] * a[k][l];
                        }
                        tr[i][k] = t / 64;
                }
        }
//GGGGGGGGGGG
        for (int k = 0; k < n; k++) {
                for (int l = 0; l < n; l++)
                {

                        kv8[k][l]   =   Convert::ToInt32(dataGridView5-
>Rows[k]->Cells[l]->Value);
                }
        }

        for (k = 0; k < n; k++) {
                for (l = 0; l < n; l++) {

                        tg[k][l] = gwhg1[8 * x + k][8 * y + l];
                        out_file.write((char*)&tg[k][l], 2);
                        tg[k][l] = tg[k][l] * kv8[k][l];

                }
        }
        for (j = 0; j < n; j++)
        {
                for (int k = 0; k < n; k++) {
                        t = 0;
                        for (int l = 0; l < n; l++)
```

```
                {
                    t = t + tg[l][j] * a[k][l];
                }
                tg1[k][j] = t;
            }
        }

        for (int i = 0; i < n; i++)
        {
            for (k = 0; k < n; k++) {
                t = 0;
                for (l = 0; l < n; l++) {
                    t = t + tg1[i][l] * a[k][l];
                }
                tg[i][k] = t / 64;
            }
        }



//BBBBBBBBBB
        for (int k = 0; k < n; k++) {
            for (int l = 0; l < n; l++)
            {

                kv8[k][l] = Convert::ToInt32(dataGridView6->Rows[k]->Cells[l]->Value);

            }
        }
        for (k = 0; k < n; k++) {
            for (l = 0; l < n; l++) {

                tb[k][l] = gwhb1[8 * x + k][8 * y + l];
                out_file.write((char*)&tb[k][l], 2);
                tb[k][l] = tb[k][l] * kv8[k][l];

            }
        }

        for (j = 0; j < n; j++)
        {
            for (int k = 0; k < n; k++) {
                t = 0;
                for (int l = 0; l < n; l++)
                {
                    t = t + tb[l][j] * a[k][l];
                }
                tb1[k][j] = t;
            }
        }

        for (int i = 0; i < n; i++)
        {
            for (k = 0; k < n; k++) {
                t = 0;
                for (l = 0; l < n; l++) {
                    t = t + tb1[i][l] * a[k][l];
                }
                tb[i][k] = t / 64;
            }
        }
```

```
                        //-----Виведення відновленого зображення----

                        for (int i = 0; i < n; i++) {
                                for (int j = 0; j < n; j++)
                                {
                                        t = tr[i][j];
                                        if (t < 0)  t = 0;
                                        if (t > 255)  t = 255;
                                        r = t;

                                        t = tg[i][j];
                                        if (t < 0)  t = 0;
                                        if (t > 255)  t = 255;
                                        g = t;

                                        t = tb[i][j];
                                        if (t < 0)  t = 0;
                                        if (t > 255)  t = 255;
                                        b = t;

                                        newColor1 = Color::FromArgb(r, g, b);
                                        image2->SetPixel(8 * x + j, 8 * y + i, newColor1);

                                }
                        }


                }

        }
        pictureBox2->Image = image2;
        out_file.close();

        String^ name3 = "d:/ProtectBM/dmc.exe c " + name + ".wht " + name + ".whc";
        char name2[512];
        i = 0;
        for (i = 0; i < name3->Length; i++) name2[i] = (char)name3[i];
        name2[i] = NULL;

        PROCESS_INFORMATION pi;
        STARTUPINFO si;

        ZeroMemory(&si, sizeof(si));
        si.cb = sizeof(si);
        ZeroMemory(&pi, sizeof(pi));

        if (!CreateProcess(NULL, (LPSTR)name2, NULL, NULL, FALSE, 0, NULL, NULL, &si, &pi))
        {
                printf("CreateProcess failed (%d).\n", GetLastError());
                return;
        }

        WaitForSingleObject(pi.hProcess, INFINITE);
        CloseHandle(pi.hProcess);
        CloseHandle(pi.hThread);
}


//Модуль відновлення зображення в частотній області

private: System::Void button14_Click(System::Object^ sender, System::EventArgs^ e) {

        Bitmap^ image2;
```

```
Bitmap^ image3;
Color newColor1;
String^ name;

if (openFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK)

	name = openFileDialog1->FileName;

String^ name3 = "d:/ProtectBM/dmc.exe e " + name + " " + name + ".wht";
char name2[512];
int i = 0;
for (i = 0; i < name3->Length; i++) name2[i] = (char)name3[i];
name2[i] = NULL;

PROCESS_INFORMATION pi;
STARTUPINFO si;

ZeroMemory(&si, sizeof(si));
si.cb = sizeof(si);
ZeroMemory(&pi, sizeof(pi));

if (!CreateProcess(NULL, (LPSTR)name2, NULL, NULL, FALSE, 0, NULL, NULL, &si, &pi))
{
	printf("CreateProcess failed (%d).\n", GetLastError());
	return;
}

WaitForSingleObject(pi.hProcess, INFINITE);
CloseHandle(pi.hProcess);
CloseHandle(pi.hThread);

name3 = name + ".wht";
string name1 = "";
for (int i = 0; i < name3->Length; i++)
	name1 += (char)name3[i];

//Відкриття файлу для читання в бінарному режимі
ifstream in_file(name1, ios::binary);

if (!in_file) {
	cout << "\nCouldn't open file";
	return;
}



image2 = gcnew Bitmap("image1GK.bmp", true);
image3 = gcnew Bitmap(image2->Width, image2->Height);

for (int y = 0; y < image2->Height; y++)
{

	for (int x = 0; x < image2->Width; x++)
	{

		gwhr[x][y] = 0;
		gwhg[x][y] = 0;
		gwhb[x][y] = 0;

	}
}
```

```
for (int x = 0; x < image2->Width; x++)
{
	for (int y = 0; y < image2->Height; y++)
	{



		Color newColor = Color::FromArgb(0, 0, 0);
		image2->SetPixel(x, y, newColor);

	}

}

Byte r, g, b;
//int  j, k, l, t, n;
int   j, k, l, t, n, it, jt, kt, lt, tt, nt, itt, jtt, ktt, ltt, ttt, ntt, mod, xt;

short int a[8][8] = { {1,1,1,1,1,1,1,1},
                                                {1,1,1,1,-1,-1,-1,-1},
                                                {1,1,-1,-1,-1,-1,1,1},
                                                {1,1,-1,-1,1,1,-1,-1},
                                                {1,-1,-1,1,1,-1,-1,1},
                                                {1,-1,-1,1,-1,1,1,-1},
                                                {1,-1,1,-1,-1,1,-1,1},
                                                {1,-1,1,-1,1,-1,1,-1} };
short int tr[8][8];
short int tr1[8][8];
short int tg[8][8];
short int tg1[8][8];
short int tb[8][8];
short int tb1[8][8];

n = 8;

it = 10;
jt = 10;
kt = 13;
lt = 13;
tt = 201;
nt = 201;

itt = 10;
jtt = 10;
ktt = 13;
ltt = 13;
ttt = 201;
ntt = 201;
int cph = (image3->Width * image3->Height) * 3;
int lht = 0;

//Модуль шифру
mod = 2;

while (mod < (image2->Width * image2->Height)) mod = mod * 2;

for (int k = 0; k < n; k++) {
	for (int l = 0; l < n; l++)
	{

		in_file.read((char*)&kv8[k][l], 1);
		dataGridView2->Rows[k]->Cells[l]->Value = kv8[k][l];
```

```
                        in_file.read((char*)&kv8[k][l], 1);
                        dataGridView5->Rows[k]->Cells[l]->Value = kv8[k][l];
                        in_file.read((char*)&kv8[k][l], 1);
                        dataGridView6->Rows[k]->Cells[l]->Value = kv8[k][l];



                }
        }

        for (int x = 0; x <= 63; x++)
        {
                for (int y = 0; y <= 63; y++)
                {

//----RRRR-----
                        for (int k = 0; k < n; k++) {
                                for (int l = 0; l < n; l++)
                                {

                                        kv8[k][l] = Convert::ToInt32(dataGridView2->Rows[k]->Cells[l]-
>Value);
                                }
                        }

                        for (k = 0; k < n; k++) {
                                for (l = 0; l < n; l++) {
                                        in_file.read((char*)&tr[k][l], 2);
                                        gwhr1[8 * x + k][8 * y + l] = tr[k][l];
                                        tr[k][l] = tr[k][l] * kv8[k][l];
                                }
                        }

//----GGGG-----

                        for (int k = 0; k < n; k++) {
                                for (int l = 0; l < n; l++)
                                {

                                        kv8[k][l] = Convert::ToInt32(dataGridView5->Rows[k]->Cells[l]-
>Value);
                                }
                        }

                        for (k = 0; k < n; k++) {
                                for (l = 0; l < n; l++) {
                                        in_file.read((char*)&tg[k][l], 2);
                                        gwhg1[8 * x + k][8 * y + l] = tg[k][l];
                                        tg[k][l] = tg[k][l] * kv8[k][l];

                                }
                        }
//----BBBB-----

                        for (int k = 0; k < n; k++) {
                                for (int l = 0; l < n; l++)
                                {

                                        kv8[k][l] = Convert::ToInt32(dataGridView6->Rows[k]->Cells[l]-
>Value);
                                }
                        }
```

```
                for (k = 0; k < n; k++) {
                        for (l = 0; l < n; l++) {
                                in_file.read((char*)&tb[k][l], 2);
                                gwhb1[8 * x + k][8 * y + l] = tb[k][l];
                                tb[k][l] = tb[k][l] * kv8[k][l];

                        }
                }
//Відновлення RRRRR
                for (j = 0; j < n; j++)
                {
                        for (int k = 0; k < n; k++) {
                                t = 0;
                                for (int l = 0; l < n; l++)
                                {
                                        t = t + tr[l][j] * a[k][l];
                                }
                                tr1[k][j] = t;
                        }
                }

                for (int i = 0; i < n; i++)
                {
                        for (k = 0; k < n; k++) {
                                t = 0;
                                for (l = 0; l < n; l++) {
                                        t = t + tr1[i][l] * a[k][l];
                                }
                                tr[i][k] = t / 64;
                        }
                }
//Відновлення GGGG
                for (j = 0; j < n; j++)
                {
                        for (int k = 0; k < n; k++) {
                                t = 0;
                                for (int l = 0; l < n; l++)
                                {
                                        t = t + tg[l][j] * a[k][l];
                                }
                                tg1[k][j] = t;
                        }
                }

                for (int i = 0; i < n; i++)
                {
                        for (k = 0; k < n; k++) {
                                t = 0;
                                for (l = 0; l < n; l++) {
                                        t = t + tg1[i][l] * a[k][l];
                                }
                                tg[i][k] = t / 64;
                        }
                }

//Відновлення BBBB
                for (j = 0; j < n; j++)
                {
                        for (int k = 0; k < n; k++) {
                                t = 0;
                                for (int l = 0; l < n; l++)
                                {
                                        t = t + tb[l][j] * a[k][l];
```

```
                                        }
                                        tb1[k][j] = t;
                                }
                        }

                        for (int i = 0; i < n; i++)
                        {
                                for (k = 0; k < n; k++) {
                                        t = 0;
                                        for (l = 0; l < n; l++) {
                                                t = t + tb1[i][l] * a[k][l];
                                        }
                                        tb[i][k] = t / 64;
                                }
                        }


                        //-----Виведення декодованого зображення----

                        for (int i = 0; i < n; i++) {
                                for (int j = 0; j < n; j++)
                                {
                                        t = tr[i][j];
                                        if (t < 0)  t = 0;
                                        if (t > 255)  t = 255;
                                        r = t;

                                        t = tg[i][j];
                                        if (t < 0)  t = 0;
                                        if (t > 255)  t = 255;
                                        g = t;

                                        t = tb[i][j];
                                        if (t < 0)  t = 0;
                                        if (t > 255)  t = 255;
                                        b = t;

                                        newColor1 = Color::FromArgb(r, g, b);
                                        image2->SetPixel(8 * x + j, 8 * y + i, newColor1);

                                }
                        }

                }

        }
        pictureBox2->Image = image2;
        in_file.close();

        for (int y = 0; y < 362/*((image2->Height) / 2)*/; y++)
        {

                for (int x = 0; x <362/* ((image2->Width) / 2)*/; x++)
                {
                        gwhr[x][y] = (gwhr[x][y] & 0x0000) | ((gwhr1[jtt][itt]^ jt ^ lt ^ nt ^ it ^ kt ^ tt) &
0x000F);
                        jt = (13 * jt + 11) % mod;
                        while (jt >= (image2->Width * image2->Height)) jt = (13 * jt + 11) % mod;
                        itt = (jt / image2->Width);
                        jtt = jt - itt * image2->Width;
                        gwhr[x][y] = (gwhr[x][y] & 0x0F) | (((gwhr1[jtt][itt] << 4) ^ jt ^ lt ^ nt ^ it ^ kt ^ tt)
& 0x00F0);
```

```
                        gwhg[x][y] = (gwhg[x][y] & 0x0000) | ((gwhg1[ltt][ktt] ^ jt ^ lt ^ nt ^ it ^ kt ^ tt) &
0x000F);
                        lt = (25 * lt + 19) % mod;
                        while (lt >= (image2->Width * image2->Height)) lt = (25 * lt + 19) % mod;
                        ktt = (lt / image2->Width);
                        ltt = lt - ktt * image2->Width;
                        gwhg[x][y] = (gwhg[x][y] & 0x0F) | (((gwhg1[ltt][ktt] << 4) ^ jt ^ lt ^ nt ^ it ^ kt ^
tt) & 0x00F0);

                        gwhb[x][y] = (gwhb[x][y] & 0x0000) | ((gwhb1[ntt][ttt] ^ jt ^ lt ^ nt ^ it ^ kt ^ tt) &
0x000F);
                        nt = (29 * nt + 29) % mod;
                        while (nt >= (image2->Width * image2->Height)) nt = (29 * nt + 29) % mod;
                        ttt = (nt / image2->Width);
                        ntt = nt - ttt * image2->Width;
                        gwhb[x][y] = (gwhb[x][y] & 0x0F) | (((gwhb1[ntt][ttt] << 4) ^ jt ^ lt ^ nt ^ it ^ kt ^
tt) & 0x00F0);

                        jt = (13 * jt + 11) % mod;
                        while (jt >= (image2->Width * image2->Height)) jt = (13 * jt + 11) % mod;
                        itt = (jt / image2->Width);
                        jtt = jt - itt * image2->Width;

                        lt = (25 * lt + 19) % mod;
                        while (lt >= (image2->Width * image2->Height)) lt = (25 * lt + 19) % mod;
                        ktt = (lt / image2->Width);
                        ltt = lt - ktt * image2->Width;

                        nt = (29 * nt + 29) % mod;
                        while (nt >= (image2->Width * image2->Height)) nt = (29 * nt + 29) % mod;
                        ttt = (nt / image2->Width);
                        ntt = nt - ttt * image2->Width;

                }

        }

        //-----Виведення відновленого зображення----
        for (int x = 0; x < 362/*(image2->Width) / 2*/; x++)
        {
                for (int y = 0; y < 362/*(image2->Height) / 2*/; y++)
                {

                        t = gwhr[x][y];
                        if (t < 0)  t = 0;
                        if (t > 255)  t = 255;
                        r = t;

                        t = gwhg[x][y];
                        if (t < 0)  t = 0;
                        if (t > 255)  t = 255;
                        g = t;

                        t = gwhb[x][y];
                        if (t < 0)  t = 0;
                        if (t > 255)  t = 255;
                        b = t;

                        newColor1 = Color::FromArgb(r, g, b);
                        image3->SetPixel(x, y, newColor1);

                }
        }
```

```
        pictureBox1->Image = image3;
}
```

Додаток Г
(довідниковий)

Формати файлу з вбудованим в трансформанти зображенням

Зсув	**Назва параметру**	**Тип**	**Призначення**
0	r	byte	Матриці квантування розміром 8x8 для RGB
1	g	Byte	
2	b	Byte	
…	…	…	
189	r	Byte	
190	g	byte	
191	b	byte	
192-319	R	Short int (2 байти)	Значення коефіцієнтів ДОП для складової R першого блоку 8x8 з вбудованою складовою R зображення (молодші 4 біти)
320-447	G	Short int (2 байти)	Значення коефіцієнтів ДОП для складової G першого блоку 8x8 з вбудованою складовою G зображення (молодші 4 біти)
448-575	B	Short int (2 байти)	Значення коефіцієнтів ДОП для складової B першого блоку 8x8 з вбудованою складовою B зображення (молодші 4 біти)
…	…	…	…

Додаток Д

(обов’язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

МЕТОДИ ТА ПРОГРАМНІ ЗАСОБИ ЗАХИСТУ БІОМЕДИЧНИХ ЗОБРАЖЕНЬ ВІД НЕСАНКЦІОНОВАНОГО ДОСТУПУ

Магістерська кваліфікаційна робота “Методи та програмні засоби захисту біомедичних зображень від несанкціонованого доступу”

Виконав: студент 2 курсу,
Групи 1ПІ-23м Грицишин В. О.
Керівник: Майданюк В. П.

Рисунок Д.1 – Титульний слайд

2

МЕТА І ЗАДАЧІ РОБОТИ

Мета роботи:

- метою магістерської кваліфікаційної роботи є зменшення обчислювальних витрат для захисту рентгенівських зображень від несанкціонованого доступу за рахунок застосування стеганографічних перетворень.

Основні задачі дослідження:

- обґрунтування доцільності розробки нового технічного рішення;
- розробка та модифікація методів приховування рентгенівських зображень;
- розробка алгоритмів і програм для дослідження приховування рентгенівських зображень;
- експериментальні дослідження ефективності стеганографічних перетворень;
- розрахунок економічних показників розробки.

Об’єкт дослідження – процес приховування рентгенівських зображень в зображеннях того ж типу.

Предмет дослідження – методи та програмні засоби приховування рентгенівських зображень в зображеннях того ж типу.

Рисунок Д.2 – Мета і задачі роботи

3

НАУКОВА НОВИЗНА ТА ПРАКТИЧНА ЦІННІСТЬ РОБОТИ

Наукова новизна отриманих результатів.

1. Подальшого розвитку отримав метод стеганографічного захисту рентгенівських зображень, у якому на відміну від існуючих, в якості контейнера використано трансформанти перетворення Уолша-Адамара, отримані з рентгенівського зображення, що дозволило збільшити кількість молодших біт, які використовуються для приховування, в кожному пікселі зображення-контейнера до 4 по кожній складовій кольору і зменшити складність реалізації алгоритму захисту у два рази.

2. Подальшого розвитку отримав метод шифрування даних при стеганографічному захисті рентгенівських зображень, у якому на відміну від існуючих, використано множину генераторів псевдовипадкових чисел для кожної складової кольору, для розсіювання даних у площині зображення та їх гамування, що дозволило збільшити криптостійкість стеганографічного захисту в 2 рази.

Практична цінність отриманих результатів. Практична цінність одержаних результатів полягає в тому, що на основі отриманих в магістерській кваліфікаційній роботі теоретичних положень запропоновано алгоритми та розроблено програмні засоби для дослідження захисту рентгенівських зображень від несанкціонованого доступу.

Особистий внесок здобувача. Усі результати, викладені магістерській кваліфікаційній роботі, отримані автором особисто. У друкованих працях, опублікованих у співавторстві, автору належать такі результати: [9] - порівняльний аналіз перетворення Уолша-Адамара; [10] - визначення параметрів генератора псевдовипадкових чисел; [11] - обгрунтування вибору стеганографічного контейнера.

Рисунок Д.3 – Наукова новизна та практична цінність

4

ЗАГАЛЬНА СХЕМА ЗАХИСТУ БІОМЕДИЧНИХ ЗОБРАЖЕНЬ ВІД НЕСАНКЦІОНОВАНОГО ДОСТУПУ

Вирішення задачі захисту біомедичних зображень:

- традиційний – шифрування файлів зображень;
- врахування специфіки зображення – виконуються будь-які перетворення в площині зображення, які виключають відтворення початкового зображення без спеціальних засобів. Після перетворення зображення зберігається у форматі зображення

Захисні перетворення

- криптографічні;
- стеганографічні;
- комбіновані

Рисунок Д.4 - Загальна схема захисту біомедичних зображень від несанкціонованого доступу

Рисунок Д.5 - Блок-схема алгоритму шифрування зображень та приховування молодших біт пікселя

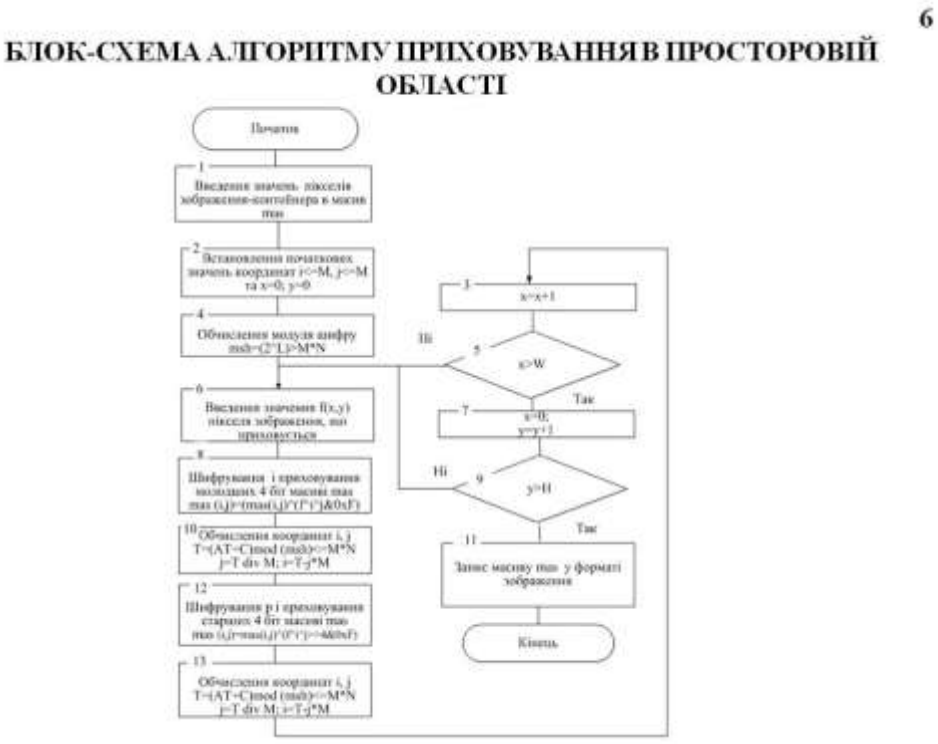


Рисунок Д.6 – Блок-схема алгоритму приховування в просторовій області

7

ДИСКРЕТНЕ ПЕРЕТВОРЕННЯ УОЛША-АДАМАРА

Перетворення Уолша-Адамара

$$[F(u,v)] = \frac{1}{N}[H(u,v)][f(x,y)][H(u,v)],$$

$$f(x,y) = \frac{1}{N}[H(u,v)][F(u,v)][H(u,v)],$$

H(u, v) – матриця Адамара, упорядкована за Уолшом;
f(x,y) – значення пікселя початкового зображення;
F(u,v) – коефіцієнти (трансформанти) перетворення Уолша-Адамара.

Кількість операцій при перетворенні масиву N*N при цьому складає:

$$m_0 = 2N^2 (N-1)$$

Без застосування швидкого обчислення перетворення Уолша-Адамара:

$$m = N^2 (N^2-1).$$

При N=8, $m/m_0 = 4,5$.

Матриця Адамара упорядкована за Уолшом розмірності 8x8

$$H = \begin{vmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \end{vmatrix}$$

Рисунок Д.7 – Перетворення Уолша-Адамара

8

БЛОК-СХЕМА ПРИХОВУВАННЯ В КОЕФІЦІЄНТАХ ПЕРЕТВОРЕННЯ УОЛША-АДАМАРА

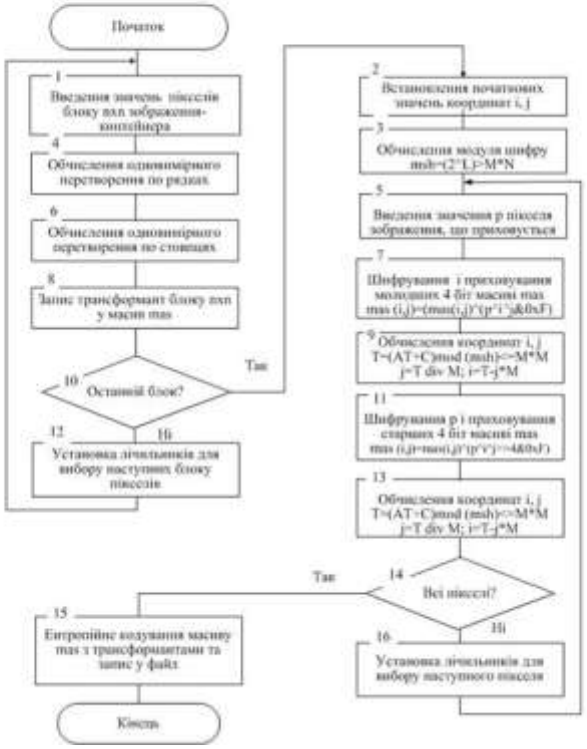


Рисунок Д.8 – Блок-схема приховування в коефіцієнтах перетворення Уолша-Адамара

9

ОЦІНКА СКЛАДНОСТІ СТЕГАНОГРАФІЧНОГО МЕТОДУ

DES

$g_1(k)=2k+4$

k – кількість каскадів мережі Фейстеля;

k=16

$g_1(16)=36$

СТЕГАНОГРАФІЧНИЙ МЕТОД

$g_2(m)=1+2*(n/(m*r))$

n – кількість біт, які читаються за 1 цикл читання;

m – кількість біт, що приховуються в 1 пікселі;

r – кількість біт на піксель;

n=64, r=8, m=1 (найгірший випадок);

$g_2(m)=17$

Рисунок Д.9 – Оцінка складності стеганографічного захисту

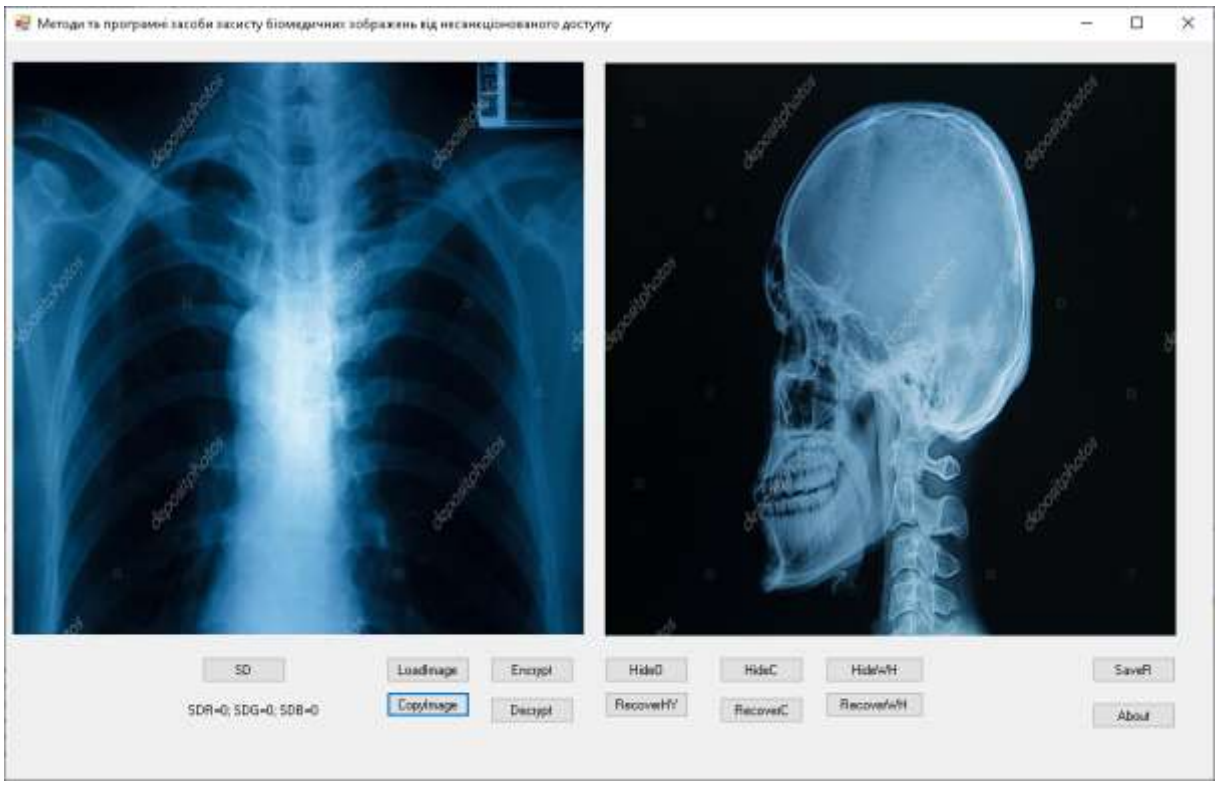
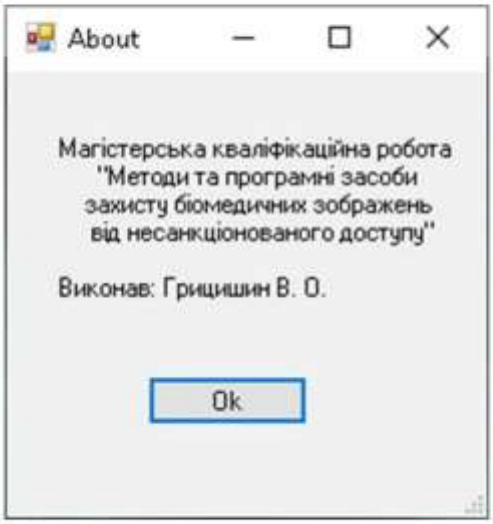


Рисунок Д.10 – Головне вікно програми

11

ІНШІ ВІКНА ДОДАТКУ

Вікно «Про програму»



Діагностичні повідомлення

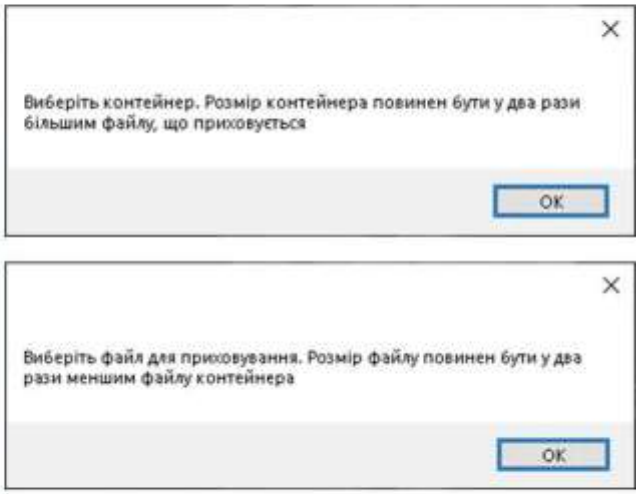


Рисунок Д.11 – Інші вікна

12

ЗОБРАЖЕННЯ (ПРАВЕ ПОЛЕ) ПІСЛЯ КРИПТОГРАФІЧНОГО ПЕРЕТВОРЕННЯ

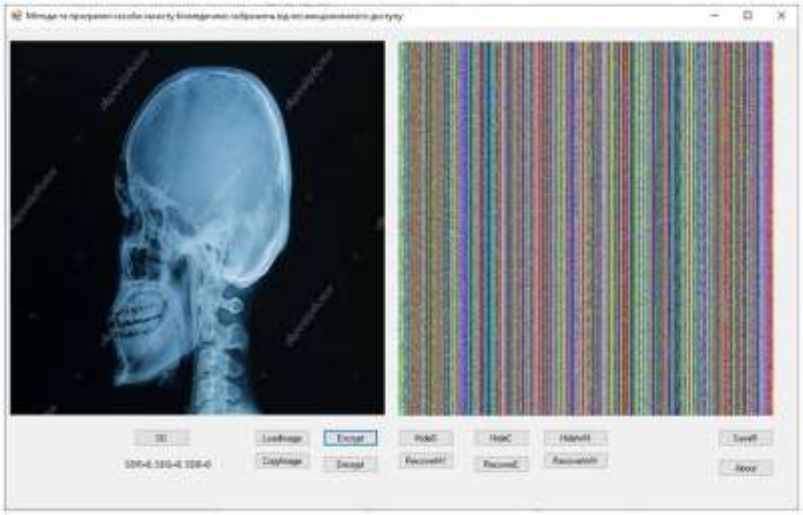


Рисунок Д.12 – Зображення після криптографічного захисту

13

ЗОБРАЖЕННЯ (ПРАВЕ ПОЛЕ) З ПРІХОВАНИМИ МОЛОДШИМИ БІТАМИ

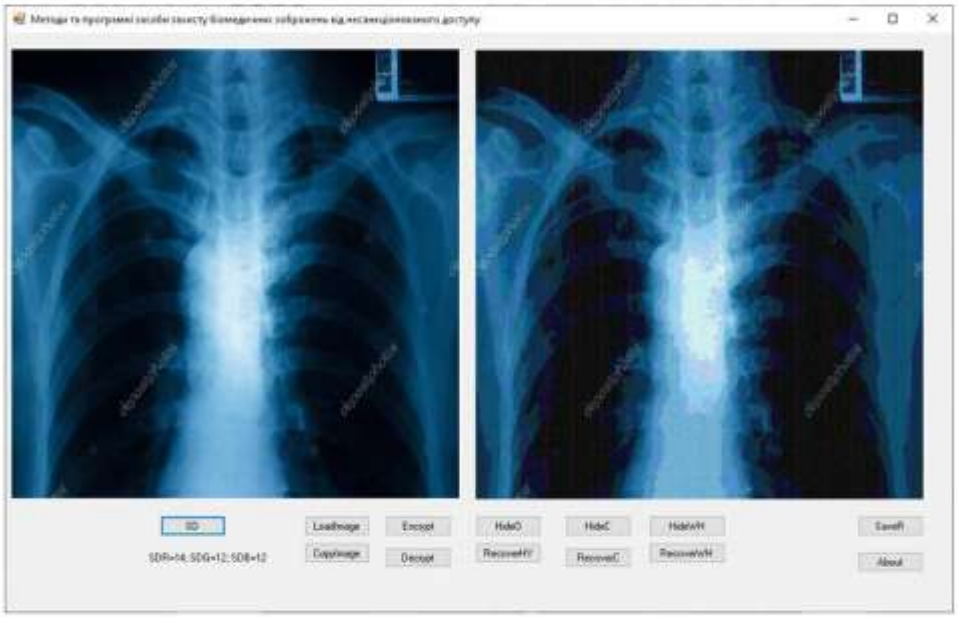


Рисунок Д.13 – Приховування молодших біт зображення

14

ЗОБРАЖЕННЯ-КОНТЕЙНЕР З ВБУДОВУВАНИМ ЗОБРАЖЕННЯМ В ПРОСТОРОВІЙ ОБЛАСТІ

5 БІТ на піксель.
Наповнення контейнера 0,5 від вмісту,
СКВ=9-10

4 БІТ на піксель.
Наповнення контейнера 0,5 від вмісту,
СКВ=4

Рисунок Д.14 – Вбудовування в просторовій області

15

ЗОБРАЖЕННЯ-КОНТЕЙНЕР З ВБУДОВУВАНИМ В НЕНОРМАЛІЗОВАНІ КОЕФІЦІЄНТИ УОЛША-АДАМАРА ЗОБРАЖЕННЯМ

Стеганографічний захист в частотній області при повному заповненню контейнера (4біти на кожний коефіцієнт перетворення Уолша-Адамара)

Розмір початкового зображення, байт		Розмір зображення після приховування та ущільнення без втрат, байт	Середньо-квадратичне відхилення (СКВ)	Візуальна оцінка якості
786486 (color)	Нормалізація коефіцієнтів при прямому і зворотному перетворенні	532541	7	Погана, помітні зміни
	Нормалізація коефіцієнтів тільки при зворотному перетворенні	717101	1	Відмінна

Рисунок Д.15 – Вбудовування в частотній області

16

ОСНОВНІ РЕЗУЛЬТАТИ РОБОТИ

1. Аналіз методів захисту біомедичних даних від несанкціонованого доступу показав, що для їх захисту, зокрема і зображень, використовуються традиційні криптографічні перетворення, які не враховують специфіку зображень.
2. Актуальною є розробка методів захисту біомедичних зображень, орієнтованих на врахування природи цих зображень.
3. Запропоновано методи захисту біомедичних зображень, відмінністю яких є виконання операцій захисту не з файлами, а у площині зображення з даними самих зображень.
4. Розроблено метод та алгоритм захисту біомедичних зображень, у якому позиція пікселя при розсіюванні та гамуванні по кожній складовій кольору визначається з використанням генератора ШВЧ.
5. Розроблено метод приховування важливої інформації, що міститься в зображенні у цьому ж зображенні за рахунок перемішування і гамування молодших 4-5 розрядів кожної складової пікселя.
6. Розроблено метод приховування важливої інформації, що міститься в рентгенівському зображенні, в контейнері того ж типу по 4 біти на піксель в кожній складовій кольору при заповненні контейнера наполовину.
7. Розроблено метод приховування важливої інформації, що міститься в рентгенівському зображенні, в трансформантах перетворення Уолша-Адамара, отриманих з зображення-контейнера, по 4 біти на піксель в кожній складовій кольору.
8. Виконано аналіз середовищ розробки та мов програмування, який показав, що для цієї розробки доцільним є використання середовища розробки Microsoft Visual Studio та мови програмування C++, оскільки ці засоби забезпечують найкращі швидкісні характеристики для програм, що розробляються, а також мають розвинені сервісні функції, які пришвидшують розробку.
9. Розроблено програму для дослідження методів захисту біомедичних зображень від несанкціонованого доступу шляхом виконання криптографічних та стеганографічних перетворень.
10. Розроблено керівництво користувача, яке містить необхідні відомості для установки програми на комп'ютер та її експлуатації.
11. Тестування програми показало її повну працездатність та відповідність завданню на роботу.
12. Термін окупності менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Рисунок Д.16 – Основні результати роботи

Рисунок Д.17 – Фінальний слайд