НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

До захисту допущено:

Завідувач кафедри

_____ Олександр РОЛІК

«___»_____20__ р.

Дипломний проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Інтегровані інформаційні системи»
спеціальності 126 «Інформаційні системи та технології»
на тему: «Цифрова система стеганографії»

Виконав:
студент IV курсу, групи ІА-94
Мартікян Сурен Арменович _____

Керівник:
асистент кафедри ІСТ
Степанов Андрій Сергійович _____

Рецензент:
доцент кафедри ІПІ, к.т.н.
Ліхоузова Тетяна Анатоліївна _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2023 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 126 «Інформаційні системи та технології»

Освітньо-професійна програма «Інтегровані інформаційні системи»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____Олександр РОЛІК

«___»_____20__р.

ЗАВДАННЯ

на дипломний проєкт студенту

Мартікяну Сурену Арменовичу

1. Тема проєкту «Цифрова система стеганографії», керівник проєкту Степанов Андрій Сергійович, асистент кафедри, затверджені наказом по університету від «31»травня 2023р. № 2101-с;

2. Термін подання студентом проєкту: «14» червня 2023

3. Вихідні дані до проєкту: технічна документація, теоритичні дані

4. Зміст пояснювальної записки:

Розділ 1: Дослідження предметної області

Розділ 2: Практична реалізація стеганографії у різних форматах файлів

Розділ 3: Проектування программи

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плака-тів, презентацій тощо): діаграма класів, контекстна діаграма, узагальнена схема стеганографічної системи, загальний алгоритм роботи застосунку;

6. Дата видачі завдання: «5» квітня 2023

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Огляд, аналіз та попередній перегляд стеганографії	17.04-21.04.2023	виконано
2	Опанування одного з методів стеганографії(LSB)	24.04-28.04.2023	виконано
3	Опанування файлів, які мають бути контейнерами	24.04-28.04.2023	виконано
4	Вибір мови та середовища програмування	01.05-05.05.2023	виконано
5	Розробка програми, яка ховає будь який файл в контейнер	08.05-12.05.2023	виконано
6	Тестування програми	15.05-19.05.2023	виконано
7	Подання ДП на попередній захист	09.06.2023	виконано
8	Подання ДП на основний захист	14.06.2023	виконано
	Захист дипломного проєкту		

Студент Сурен МАРТІКЯН

Керівник проєкту Андрій СТЕПАНОВ

АНОТАЦІЯ

Мартікян С.А. Цифрова система стенографії. КПІ ім. Ігоря Сікорського, Київ, 2023.

Проєкт містить 61 с. тексту, 27 рисунків, 3 таблиці, посилання на 19 літературних джерел, додаток та 4 конструкторських документа.

Ключові слова: стеганографія, LSB метод, контейнер, секретний файл, шифрування.

Об’єктом розробки є Цифрова система стенографії.

Мета розробки – розробка цифрової системи стеганографії, яка забезпечує надійне та ефективне вбудовування і витягування прихованої інформації з цифрових зображень. Система повинна бути стійкою до атак, а також забезпечувати збереження якості зображення після вбудовування і видобування даних.

У даному проєкті будуть розглянуті та реалізовані основні принципи стеганографії, зокрема використання методу найменших значущих бітів (LSB) для вбудовування повідомлення в різні типи медіа-файлів, такі як зображення та аудіо. Очікується, що розроблена цифрова система стеганографії буде здатна ефективно приховувати повідомлення всередині медіа-файлів, забезпечуючи при цьому стійкість до аналізу та виявлення. Така система може мати широке застосування в області конфіденційного обміну інформацією, криміналістичних досліджень, кібербезпеки та військовою метою.

SUMMARY

S.A. Martikyan Digital shorthand system. KPI named after Igor Sikorsky, Kyiv, 2023.

The project contains 61 pages. text, 27 figures, 3 tables, references to 19 literary sources, an appendix and 4 design documents.

Keywords: steganography, LSB method, container, secret file, encryption.

The object of development is the Digital Shorthand System.

The purpose of the development is to develop a digital steganography system that provides reliable and effective embedding and extraction of hidden information from digital images. The system must be resistant to attacks, as well as ensure the preservation of image quality after embedding and data extraction.

This project will explore and implement the basic principles of steganography, including the use of Least Significant Bits (LSB) to embed messages in various media types such as images and audio. The developed digital steganography system is expected to be able to effectively hide messages inside media files while providing resistance to analysis and detection. Such a system can be widely used in the field of confidential information exchange, forensic investigations, cyber security and military purposes.

Номер рядка	Формат	Позначення	Найменування	Кільк. аркушів	Номер екзем.	Примітка
1			Документація загальна			
2						
3			Знову розроблена			
4						
5	А4	ІА94.100БАК.004 ПЗ	Пояснювальна записка	61		
6	А3	ІА94.100БАК.004 Д1	Цифрова система	1		
7			стеганографії.			
8			Контекстна діаграма.			
9	А3	ІА94.100БАК.004 Д2	Цифрова система	1		
10			стеганографії.			
11			Діаграма класів			
12	А3	ІА94.100БАК.004 Д3	Цифрова система	1		
13			стеганографії.			
14			Загальний алгоритм роботи			
15			застосунку			
16	А3	ІА94.100БАК.004 Д4	Цифрова система	1		
17			стеганографії.			
18			Узагальнена схема			
19			стеганографічної системи			
20						
21						
22						
23						
24						
25						
26						
27						
28						

Зм.	Аркуш	№ докум.	Підпис	Дата	І А94.100БАК.004 ТП		
Розроб.		Мартікян С.А.			**Цифрова система стеганографії**. Відомість проєкту	Літ.	Аркуш 1 / Аркушів 1
Керівн.		Степанов А.С.				т	
						КПІ ім. Ігоря Сікорського	
						Група ІА-94	
Затв.							

Пояснювальна записка

до дипломного проєкту

на тему: «Цифрова система стеганографії»

Київ – 2023 року

ЗМІСТ

Перелік скорочень та умовних позначень

AES - симетричний алгоритм блочного шифрування.

LSB - Least Significant Bit, найменший значущий біт.

PNG - Portable Network Graphics, растровий формат збереження графічної інформації.

BMP – Bitmap, формат файлу зображень растрової графіки.

JPEG - Joint Photographic Experts Group, растровий формат зберігання графічної інформації.

DCT - Discrete Cosine Transform, дискретне косинусне перетворення

WAV - Waveform audio format, формат аудіофайлу, розроблений компаніями Microsoft та IBM.

Visual studio - інтегроване середовище розробки програмного забезпечення.

C# - Сі шарп, об'єктно-орієнтована мова програмування.

ВСТУП

У сучасному цифровому світі, де обмін інформацією став неодмінною складовою нашого повсякденного життя, безпека та конфіденційність цієї інформації стають особливо важливими. З кожним днем зростає кількість цифрових даних, які обробляються, зберігаються та передаються, тому виникає потреба у захисті цих даних від несанкціонованого доступу, змін та витоку.

Одним із найефективніших способів захисту конфіденційної інформації є стеганографія - наука про таємне приховування інформації. Стеганографія використовує різні методи та алгоритми для вбудовування конфіденційних даних в носії, такі як зображення, аудіо-файли або відео, з метою збереження цілісності та конфіденційності інформації.

Метою даної дипломного проєкту є розробка стеганографічного застосунку, який надає зручні та надійні засоби для приховування даних у різних форматах файлів. Основна мета полягає у створенні потужного та ефективного інструменту, який забезпечує високу стійкість до виявлення та збереження якості носія.

Для досягнення цієї мети, у роботі проводиться дослідження предметної області стеганографії. Розглядаються загальні уявлення про стеганографію, включаючи класичну, комп'ютерну та цифрову стеганографію. Досліджуються існуючі рішення та аналізуються їхні переваги та недоліки. На основі цього аналізу формулюється постановка задачі розробки власного стеганографічного застосунку, який буде включати методи вбудовування та вилучення інформації в різних форматах файлів.

Окрім того, в даній роботі розглядаються розширений стандарт шифрування (AES) та метод останнього біта (LSB), які є основними технологіями для стеганографічного приховування даних. Аналізується структура різних форматів файлів, таких як зображення (BMP, PNG, JPEG) та аудіо-файли (WAV), для встановлення оптимальних місць для вбудовування даних.

Для розробки стеганографічного застосунку використовуються такі технології та інструменти, як мова програмування C#, середовище Visual Studio та .NET Framework. Реалізована архітектура проекту дозволяє забезпечити модульність та розширюваність застосунку, що дозволяє легко додавати підтримку нових форматів файлів у майбутньому.

В цілому, даний дипломний проєкт спрямований на розробку надійного та ефективного стеганографічного застосунку, який може бути використаний для захисту конфіденційної інформації у різних сферах, таких як комунікація, архівування та передача даних. Подальші розділи роботи детально описують розроблену архітектуру проекту, методи вбудовування та вилучення інформації у різних форматах файлів, а також результати експериментів та висновки, зроблені на основі проведених досліджень та розробки програмного забезпечення.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Актуальність проблеми

Проблема стеганографії залишається актуальною і в наш час. Стеганографія - це наука про приховане передавання інформації шляхом вбудовування її в нерозрізнимий вигляд звичайних об'єктів або даних. Ця техніка може бути використана для конфіденційного передавання інформації, а також для зловживання, зокрема, у злочинних цілях.

Ось кілька причин, чому проблема стеганографії залишається актуальною:

– конфіденційне передавання інформації: Стеганографія може бути використана для захисту конфіденційної інформації, наприклад, під час передавання важливих даних через незахищені мережі. Це особливо актуально в сферах, де конфіденційність даних є важливою, наприклад, у фінансовому секторі або сфері національної безпеки;

– кібербезпека: Використання стеганографії може бути складним для виявлення і аналізу, оскільки прихована інформація зазвичай не привертає уваги. Це може створювати проблеми для організацій, які намагаються виявити й запобігти шпигунству, кібератакам або розповсюдженню незаконної інформації;

– зловживання: Стеганографія може бути використана зловмисниками для незаконної передачі інформації, такої як поширення торгівля наркотиками або планування терористичних актів. Це створює виклик для правоохоронних органів, які намагаються виявити і запобігти таким злочинам;

– розвиток технологій: З появою нових технологій і методів аналізу даних, таких як штучний інтелект і машинне навчання, постійно з'являються нові способи виявлення стеганографії. Однак, разом з цим розвиваються й методи стеганографії, які намагаються уникнути виявлення та аналізу.

Отже, проблема стеганографії залишається актуальною в контексті захисту конфіденційної інформації, кібербезпеки та боротьби зі злочинністю. Необхідно продовжувати дослідження і розвивати нові методи виявлення і захисту від стеганографії для забезпечення безпеки та захисту інформації.

1.2 Загальні уявлення про стеганографію

Стеганографія — спосіб передачі або зберігання інформації з урахуванням таємності факту такої передачі (зберігання). На відміну від шифрування, яка приховує зміст секретної інформації, стеганографія приховує факт існування інформації. Стеганографія зазвичай використовується в поєднанні з методами шифрування, тим самим доповнюючи один одного. Перевага стеганографії перед шифруванням полягає в тому, що інформація не привертає уваги. Таким чином, шифрування захищає зміст інформації, а стеганографія захищає факт існування прихованої інформації. Наприкінці 1990-х років виділили 3 види стеганографії:

– Класичний;

– Комп'ютерний;

– Цифровий.

1.2.1 Класична стеганографія

У давнину рабів використовували для приховування інформації. Таємне послання було написано на голеній голові раба. Коли у раба відростало волосся, він йшов до адресата, який після гоління голови читав таємне послання. Одним із найпоширеніших методів класичної стеганографіяграфії є використання невидимого чорнила. Текст, написаний таким чорнилом, проявляється лише за певних умов (нагрівання, освітлення, хімічна обробка тощо). Винайдений ще в I столітті, він продовжував використовуватися протягом Середньовіччя і до наших днів. Існує хімічно нестійке пігментне чорнило. Письмо цими чорнилами виглядає так, ніби це було написано звичайною ручкою, але через деякий час нестійкий пігмент руйнується, і слідів тексту не залишається. Є й інші методи приховування інформації:

– записка біля стопки традиційно розташованих карток;

– написи всередині вареного яйця, за допомогою спеціального матеріалу

інформація пишеться на яйці, потім після його варіння текст переноситься на яєчний білок;

– «коди сленгу», де слова мають інше умовне значення;

– трафарет, який після розміщення на тексті залишає видимими приховані букви та інші методи.

В даний час під стеганографією найчастіше розуміють приховування інформації в текстових, графічних або звукових файлах за допомогою спеціального програмного забезпечення.

1.2.2 Комп'ютерная стеганографія

Комп'ютерна стеганографія - класичний напрямок стеганографія, заснований на властивостях комп'ютерної платформи. Ось кілька прикладів:

– використання полів у форматах комп'ютерних файлів. Суть методу полягає в тому, що частина поля формату не заповнюється інформацією, вона за замовчуванням заповнюється нулями. Відповідно, ми можемо використовувати цю «нульову» частину для запису наших даних. Недоліки: легкість виявлення і малий обсяг інформації, що передається;

– спосіб приховування інформації в невикористаних областях жорстких дисків. При використанні цього методу інформація записується в невикористані частини жорсткого диска в нульових полях. Недоліки: низька продуктивність, мала пропускна здатність повідомлень;

– як використовувати спеціальні властивості полів формату, які не видно на екрані. Цей метод заснований на спеціальних «невидимих» полях для отримання анотацій, покажчиків. Наприклад, написання чорними літерами на чорному тлі. Недоліки: низька продуктивність, малий обсяг інформації, що передається;

– використання можливостей файлових систем. Файли на жорсткому диску завжди займають ціле число кластерів. Наприклад, у широко використовуваній раніше файловій системі FAT32 (використовувалася в Windows98 / Me / 2000) стандартний розмір кластера становить 4 КБ.

Відповідно, для зберігання файлу з пам'яттю 1 КБ використовувався один кластер, який займав 4 КБ пам'яті, а решта 3 КБ ні для чого не використовувалися, відповідно ці 3 КБ можна використовувати для зберігання інформації.
Недолік: Легкість виявлення.

1.2.3 Цифрова стеганографія

Цифрова стеганографія — це розділ класичної стеганографії, який приховує або вбудовує інформацію в цифровий об’єкт, спричиняючи певне спотворення цих об’єктів. Як правило, використовуються зображення, відео, аудіо, текстури 3D об'єктів. Спотворення є нижчими за середньостатистичний поріг людської чутливості й не призводять до помітних змін цих об’єктів. Крім того, в оцифрованих об'єктах аналогового характеру завжди присутні шуми, при відтворенні об'єкта з'являються додаткові аналогові шуми і апаратні нелінійні спотворення, що сприяє більшій непомітності прихованої інформації.

Конфіденційну інформацію можна приховати в будь-якому цифровому об'єкті: текстовому документі, ліцензійному ключі, розширенні файлу тощо. Однак найбільш підходящим контейнером є зображення, аудіо, відео. Зазвичай вони досить великі за розміром, що означає, що можна зберігати великі обсяги даних з невеликими спотвореннями.

Конфіденційна інформація може бути записана в метаданих файлу або безпосередньо в основному вмісті. Наприклад, візьміть зображення, яке містить сотні тисяч пікселів. Кожен піксель має опис, інформацію про колір. У більшості випадків секретна інформація ховається в пікселях зображень і декодується звідти за допомогою спеціальних алгоритмів. Одним з алгоритмів є метод LSB, про який ми поговоримо в наступному розділі.

1.3 Стеганографічна система

Для загального опису стеганографічних систем використовуються

стеганографічні моделі. У 1996 році «Приховування інформації». На конференції «Перша інформаційна майстерня» прийнято єдину термінологію. Імпорт та виділення секретного листа з іншої інформації виконується стеганографічною системою. Стеганографічна система — це комбінація інструментів і методів, що використовуються для створення прихованого повідомлення. При побудові системи реєстрації необхідно враховувати такі фактори:

– стегонографічна система повинна мати прийнятну обчислювальну складність алгоритму;

– методи стеганографії повинні забезпечувати автентичність і цілісність інформації для уповноваженої особи;

– єдиною інформацією, невідомою потенційному супротивнику, є ключ, за яким можна визначити наявність та зміст секретного повідомлення;

– зловмисник не має технічних чи інших переваг для розкриття змісту прихованого повідомлення;

– якщо зловмисник виявляє існування прихованого повідомлення, він не повинен дозволити йому отримати приховану інформацію, поки ключ зберігається в секреті.

Характерною рисою типової стеганографічної системи є існування певного зв’язку між стабільністю вбудованого повідомлення, різними зовнішніми впливами (включаючи стегенаналіз) і обсягом повідомлення, яке показано на рисунку 1.1.

Збільшення обсягу прихованої інформації сприяє зниженню надійності системи (незалежно від розміру контейнера). Таким чином, контейнер, що використовується в стеганографічній системі, накладає певні обмеження на розмір прихованої інформації.

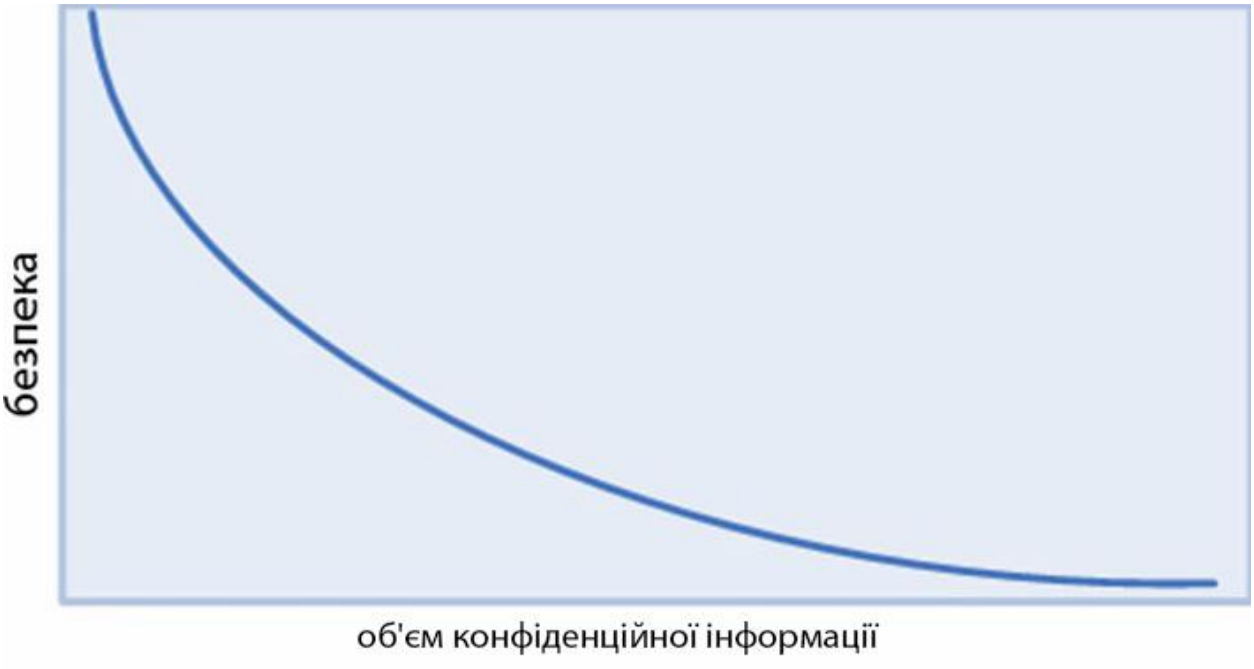

Рисунок 1.1 – Залежність системи стегонаграфії від обсягу секретної інформації

Давайте зараз поговоримо про основні складові стеганографічної системи:

1. контейнер - так називається будь-яка послідовність інформації, яка дозволяє вставити приховану в ній інформацію;

a) порожній контейнер – контейнер, який не містить прихованого повідомлення.

b) заповнений контейнер – контейнер, який містить приховане повідомлення.

2. повідомлення – загальна назва прихованої інформації, тому що повідомлення може бути у вигляді тексту, зображення, а чому б і не звукового типу;

3. token channel – завершений канал передачі контейнера;

4. ключ – секретний ключ, необхідний для розшифровки конфіденційної інформації

5. пристрій виявлення стеганографічного контейнера – модуль, призначений для виявлення стеганографічного контейнера;

6. пристрій шифрування/дешифрування повідомлень – модулі, які служать для шифрування/дешифрування інформації із заповненого контейнера;

7. пристрій виявлення стеганографічного контейнера – модуль, призначений

для виявлення стеганографічного контейнера;

8. пристрій шифрування/дешифрування повідомлень – модулі, які служать для шифрування/дешифрування інформації із заповненого контейнера;

9. конвертер повідомлень - частина модуля виявлення стеганографічного контейнера, яка перетворює повідомлення в необхідну форму (стиснення або шифрування) для приховування/дешифрування.

На рисунку 1.2 показаний загальний вигляд стеганографічної системи.

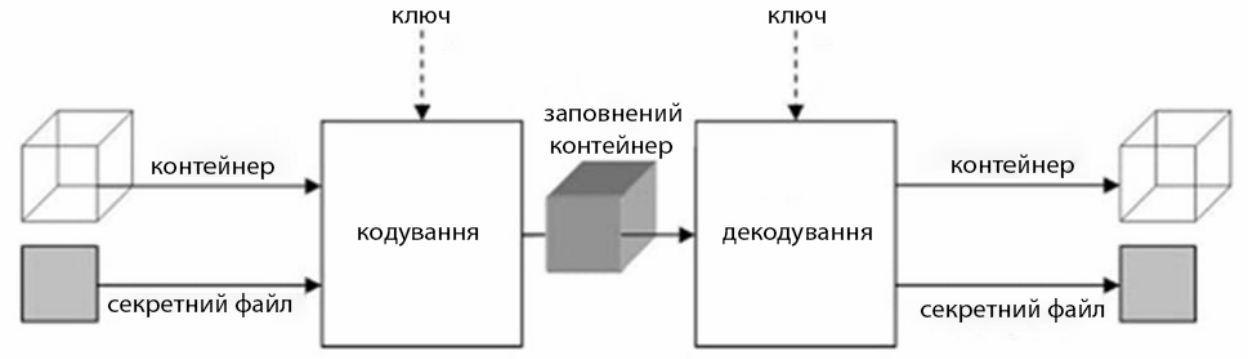

Рисунок 1.2 – Стеганографічная система

1.4 Опис існуючих рішень

1.4.1 ImageSpyer G2

Утиліта для приховування інформації у графічних файлах із використанням шифрування. При цьому підтримується близько 30 алгоритмів шифрування та 25 хеш-функцій для шифрування контейнера. Приховує обсяг, що дорівнює числу пікселів зображення. Опціонально доступна компресія даних, що приховуються.

1.4.2 RedJPEG

Інтерфейс цієї програми, як і випливає з назви, виконаний у червоному стилі. Ця проста у використанні утиліта призначена для приховування будь-яких даних у JPEG у зображенні (фото, картинка) за допомогою стеганографічного методу. Використовує відкриті алгоритми шифрування, потоковий шифр AMPRNG та

Cartman II DDP4 у режимі хеш-функції, LZMA-компресію.

1.4.3 ImageJS

ImageJS - Linux утиліта, яка призначена не зовсім для приховування інформації від людей. Натомість вона допомагає обманювати браузери, які цілком обґрунтовано припускають, що у валідних картинках нічого стороннього бути не повинно.

ImageJS дозволяє створити картинки, які одночасно є справжніми JS-скриптами. Це потрібно, щоб спростити проведення більш небезпечних XSS-атак, де іноді потрібно підвантажити скрипт саме з атакованого домену. Ось тут на допомогу приходить можливість завантажити туди аватарку, яка одночасно містить JavaScript payload для подальшої атаки. Програма підтримує впровадження у формати BMP, GIF, WEBP, PNG та PDF.

1.5 Аналіз існуючих рішень

1.5.1 Переваги та недоліки ImageSpyer G2

– підтримує близько 30 алгоритмів шифрування та 25 хеш-функцій для шифрування контейнера;

– можливість приховати обсяг, що дорівнює числу пікселів зображення;

– опціональна компресія даних, що приховуються;

– не згадується про типи графічних файлів, які підтримуються;

– не вказано, чи можливо видобування прихованих даних безпосередньо з програми.

1.5.2 Переваги та недоліки RedJPEG

– простий у використанні інтерфейс;

– приховування даних у форматі JPEG за допомогою стеганографічного

методу;

– використання відкритих алгоритмів шифрування та компресію LZMA;

– не згадується про підтримку інших графічних форматів;

– не вказано, чи можливе видобування прихованих даних безпосередньо з програми.

1.5.3 Переваги та недоліки ImageJS

– призначена для обманювання браузерів і створення зображень, що одночасно є JS-скриптами;

– використовується для спрощення проведення XSS-атак;

– підтримує формати BMP, GIF, WEBP, PNG та PDF для впровадження скриптів;

– не вказано, чи можливе приховування інших типів даних, окрім JS-скриптів;

– функціональність, пов'язана з приховуванням даних, обмежена налаштуванням скриптів.

У цілому, ImageSpyer G2 та RedJPEG спеціалізуються на приховуванні даних у графічних файлах з використанням криптографії та стеганографії відповідно. З іншого боку ImageJS має зовсім іншу спеціалізацію, а саме, використовується для створення зображень, що одночасно є JS-скриптами, з метою обману браузерів і проведення XSS-атак. Отже, ці три рішення відрізняються функціональністю та способом використання.

1.6 Постановка задачі

Стеганографія - це ефективний спосіб захисту інформації, який стає особливо актуальним, коли складні системи стеганографії з тих чи інших причин важко використовувати. У сфері стеганографії аналіз програмних систем привів до висновку, що таких програм занадто мало, вони не забезпечують необхідну функціональність або непридатні.

З цієї причини виникла потреба в розробці програми для створення та передачі стеганографічних контейнерів. Необхідно організувати додаток таким чином, щоб для стороннього спостерігача процес сприймався як звичайний обмін цифровими файлами.

Метою дипломної роботи є проектування стеганографічної системи, яка реалізує безпечне зберігання або передачу інформації. Для того, щоб зробити систему більш безпечною, інформація буде мати опцію шифрування перед впровадженням стеганографії. Для досягнення поставленої мети необхідно вирішити наступні проблеми:

– вивчати та аналізувати основні методи та принципи цифрової стеганографії;

– вивчити формати, які будуть контейнерами для конфіденційної інформації;

– створити стеганографічний алгоритм, який забезпечить надійність і функціональність нашої системи;

– розробити програму, яка реалізує алгоритм, який матиме функціональність для приховування та дешифрування даних, за допомогою зручного інтерфейсу.

Висновок до розділу 1

В даному розділі було розглянуто теоретичні основи стеганографії, визначено основні поняття та принципи її функціонування. Здійснений історичний огляд розвитку стеганографії дозволив отримати уявлення про її еволюцію та важливі етапи розвитку.. Також була обґрунтована актуальність теми дослідження цифрової системи стеганографії, сформульовано мету та завдання проекту.

2 ПРАКТИЧНА РЕАЛІЗАЦІЯ СТЕГАНОГРАФІЇ

2.1 Розширений стандарт шифрування (AES)

Розширений стандарт шифрування (AES), також відомий як Rijndael (вимовляється як [rɛinda:l] — Reindal) — це симетричний алгоритм шифрування (розмір блоку 128 біт, ключ 128/192/256 біт), прийнятий урядовим стандартом шифрування США за результатами конкурс AES. Цей алгоритм добре проаналізований і зараз широко використовується, як це було з попередньою системою шифрування DES. 26 травня Національний інститут стандартів і технологій (NIST) оголосив AES стандартом шифрування. Станом на 21е століття AES є одним із найпопулярніших алгоритмів симетричного шифрування. Підтримка прискорення AES була представлена Intel у сімействі процесорів x86, починаючи з 2010 року з процесорами Arrandale, а потім Sandy Bridge. 128 біт інформації розбиваються на байти та упаковуються в масив 4x4. Шифрування і дешифрування показано на рисунку 2.1. Таблиця 2.1 містить деякі визначення, щоб краще зрозуміти алгоритм шифрування AES.

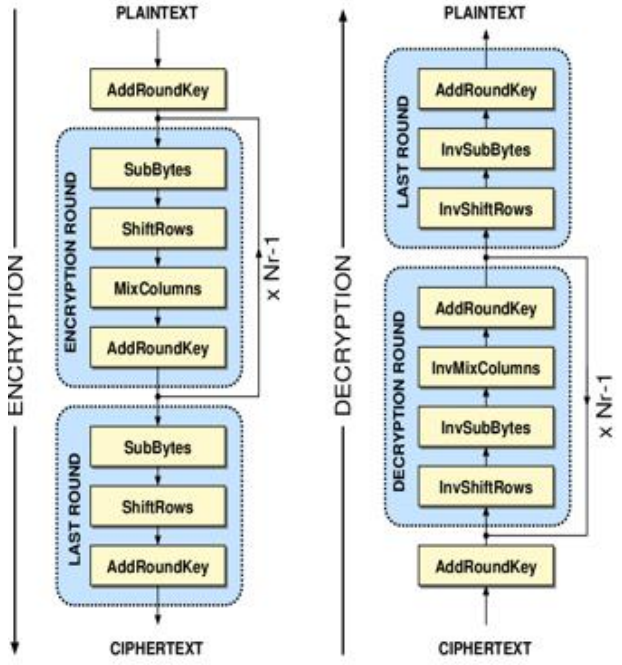

Рисунок 2.1 – Шифрування та дешифрування в AES

Таблиця 2.1 - Деякі визначення AES

Block	Послідовність бітів, які є полями введення, виведення, стану та ключа раунду. Блок також можна розуміти як послідовність байтів.
Cipher Key	Секретний ключ, який використовується в процедурі розширення ключа, де отримані фазові ключі
Ciphertext	Зашифрована інформація
Key Expansion	У процедурі круглий ключ створюється з секретного ключа (Cipher Key).
Round Key	Round Key - отримується з ключа шифру за допомогою процедури розширення ключа. Під час шифрування та дешифрування фазовий ключ призначається STATE.
State	Проміжний результат шифрування, який можна представити у вигляді прямокутного масиву байтів
S-box	Масив заміни, який використовується в процедурах. SBox і InvSBox можна знайти в Інтернеті
Nb	Кількість рядків і стовпців масиву. Для AES Nb=4
Nk	32 - кількість бітових слів, які складають ключ шифрування. Для AES Nk = 4, 6 або 8
Nr	Кількість кроків шифрування. Для AES Nk = 10, 12 або 14
Rcon[]	Це масив, який використовується в процедурі розширення ключа

Отримуємо фазові ключі в блоці розширення ключів. Ключ шифру має довжину 128 біт і упакований у масив 4x4, як показано на малюнку 4. Щоб отримати W5, спочатку береться W1, потім виконується операція RotWord, яка переміщує стовпець вгору, а 1-й елемент опускається, закінчуючи RW. Потім виконуємо операцію SybeBytes з цим стовпцем, в результаті чого отримуємо S1.

W5 отримано є логічним добутком W1, S1 і R1 (масив Rcon, який можна знайти в Інтернеті) у стовпці 1. «Отримано решта елементів: W6 = W2 ⊕ W5, W7 = W3 ⊕ W6, W8 = W4 ⊕ W7

Решта фазових ключів отримують аналогічно, початковий ключ вибирають з останнього отриманого.

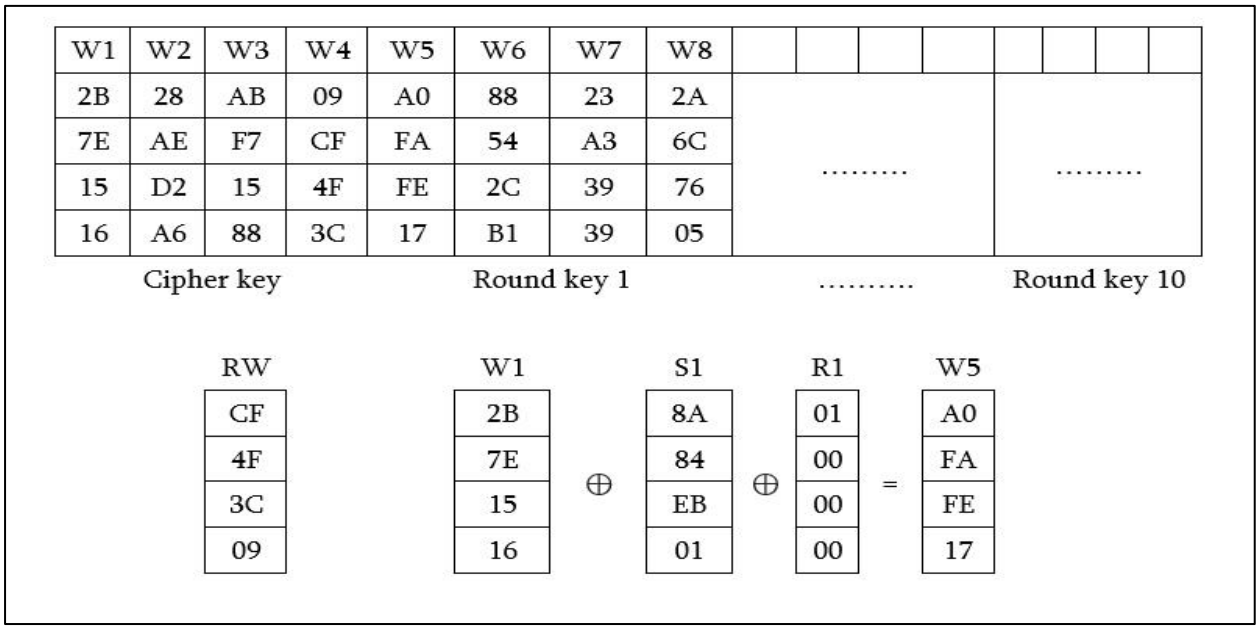

Рисунок 2.2 – Ключова фаза розширення в AES

Як бачимо з рисунку 2.2, шифрування та дешифрування відбувається за допомогою процедур AddRoundKey, SubBytes, ShiftRows, MixColumns, AddRoundKey, InvSubBytes, InvShiftRows, InvMixColumns. Тепер поговоримо про описані вище процедури.

SubBytes: У цій процедурі перша половина нашого байтового масиву вказує на рядок, а друга половина вказує на стовпець. Потім ми замінюємо значення масиву на значення рядка та стовпця, отримані вище в SBox. Підстановка в InvSubBytes виконується таким же чином, значення рядка береться з масиву InvSbox(Рисунок 2.3).

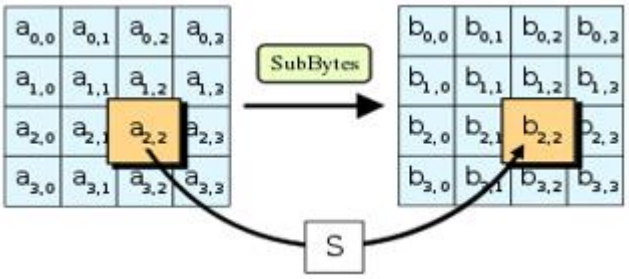

Рисунок 2.3 – Фаза SubBytes в AES

SubBytes: У цій процедурі перша половина нашого байтового масиву вказує на рядок, а друга половина вказує на стовпець. Потім ми замінюємо значення масиву на значення рядка та стовпця, отримані вище в SBox. Підстановка в InvSubBytes виконується таким же чином, значення рядка береться з масиву InvSbox(Рисунок 2.3).

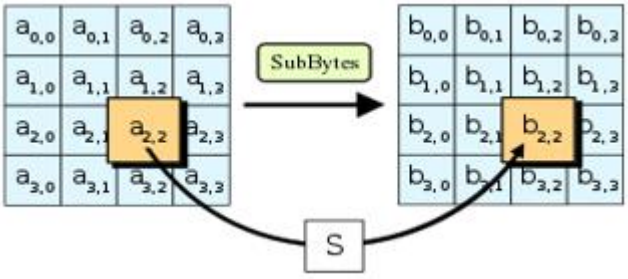

Рисунок 2.3 – Фаза SubBytes в AES

ShiftRows: На цьому етапі значення в першому рядку не зсуваються, у 2-му рядку - на 1 - вліво, в 3-му - на 2 - в 4-му - на 3 - де значення в перший стовпець переміщується в кінець. InvShiftRows також виконується у зворотному порядку(Рисунок 2.4).

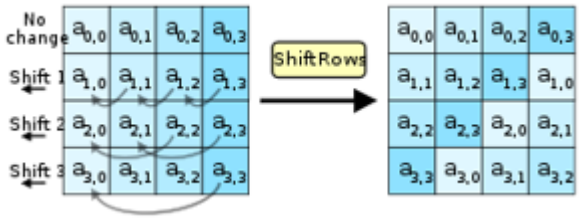

Рисунок 2.4 – Фаза ShiftRows в AES

MixColumns: Процедура MixColumns бере чотири байти кожного стовпця масиву та множить його на фіксований поліном c(x) = 3x3 + x2 + x + 2 за модулем GF(28) x4 + 1 . У InvMixColumns множення виконується на фіксований поліном c(x) = 11x3 + 13x2 + 9x + 14(Рисунок 2.5).

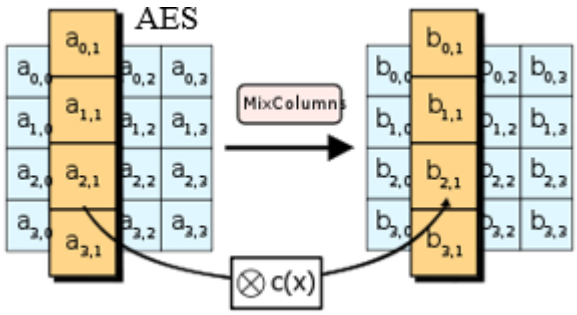

Рисунок 2.5 – Фаза MixColumns в AES

AddRoundKey: На цьому етапі значення в нашому масиві логічно перемножуються (XOR) на відповідний елемент фазового ключа відповідно(Рисунок 2.6)

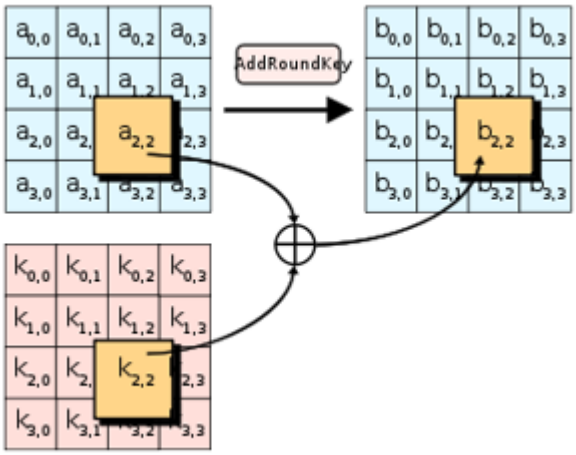

Рисунок 2.6 – Фаза AddRoundKey в AES

2.2 Метод останнього біта (LSB)

У процесі введення інформації також необхідно враховувати властивості системи сприйняття людини. Розглянемо наступний приклад, припустимо, що у нас є зображення в градаціях сірого з 256 відтінками сірого, де один байт (8 біт/піксель) відповідає за відтінок. Добре відомо, що людське око не в змозі виявити зміну в молодшому розряді.

Наприклад, ми маємо послідовність 8-розрядних двійкових чисел, де кожен байт представляє один піксель. Припустімо, ми приховали цифру 11, яка виглядає як 1011 у двійковій системі. Тепер використовуйте останній біт послідовності байтів, щоб приховати цифру. Наша припущена послідовність виглядатиме так: 01101011 00111010 01101111 01100101 ... .

Метод можна застосовувати не тільки до сірих, а й до зображень RGB. У цьому випадку стеганографія може здійснюватися за допомогою байтів червоного, зеленого або синього компонентів пікселя. Суть методу полягає в тому, що за допомогою молодшого біта або бітів здійснюється стеганографія нашого повідомлення. Для реалізації стеганографії повідомлення буде перетворено у двійковий формат, тоді заголовок файлу буде опущено, а в метаданих використовується алгоритм LSB. Після перетворення зображення на двійковий останній біт кожного байта замінюється бітом у нашій двійковій послідовності. В результаті ми отримуємо зображення, яке можна передати по відкритому каналу і зловмисник не може здогадатися про наявність конфіденційної інформації. Одержувач може легко витягнути повідомлення із зображення після отримання контейнера.

2.3 Структура формату PNG

Сьогодні PNG (Portable Network Graphics) є одним із найпопулярніших форматів веб-графіки. У 1995 році на форумі Usenet було запропоновано створити формат на заміну популярному на той час формату GIF, який вимагав ліцензії. Так з'явився PNG, що неофіційно розшифровується як «PNG IS NOT A GIF».

Тепер давайте детально обговоримо переваги та недоліки формату. Зображення піддається стисненню без втрат. Підходить для зберігання проміжних версій зображення. Підтримує велику кількість кольорів. PNG - 8 (256 кольорів) і PNG - 24 (близько 16,7 мільйонів кольорів). Використовується метод під назвою альфа-хвиля.

Вміння працювати з шарами. можливість додавання МЕТА-ДАНИХ до файлу (при необхідності захист авторських прав). невеликі розміри файлів. Це єдиний формат, який дозволяє отримувати зображення з прозорим фоном (наприклад, це дуже важливо при створенні логотипів, для яких зазвичай потрібен прозорий фон). Тепер поговоримо про недоліки. Немає підтримки анімації. Неможливо зберегти кілька зображень в одному файлі. Крім логотипів, цей формат також використовується для створення елементів навігації сторінок, типографіки, літографій, текстів, малюнків з чіткими краями зображень. В цілому PNG використовується для отримання прозорих зображень.

Оскільки формат PNG зберігає зображення без втрати кольору, стеганографія виконується в молодших бітах пікселів зображення за допомогою алгоритму LSB.

2.4 Структура формату BMP

BMP (від слів BitMap - растрове зображення або бітовий масив) - це формат зберігання зображень, розроблений Microsoft з розширенням .bmp. З форматом BMP працює величезна кількість програм, так як його підтримка інтегрована в операційні системи Windows і OS/2. ОС Windows має спеціальні функції API, які допомагають читати та відображати зображення формату BMP. Тепер розглянемо структуру BMP. БМП складається з чотирьох частин:

– BitMapFileHeader;

– BitMapInfoHeader;

– таблиця кольорів;

– масив BitMap.

В таблиці 2.2 представлена будова БМП в цегляному вигляді.

Таблиця 2.2 – Структура BMP

Ім'я:	Довжина	Положення	Опис
BitMapFileHeader			
Type	2	0	Пишеться "BM"
Size	4	2	Довжина файлу
Reserved 1	2	6	Зарезервовані біти
Reserved 2	2	8	Зарезервовані біти
OffsetBits	4	10	Перейти від початку файлу
BitMapInfoHeader			
Size	4	14	Довжина заголовка
Width	4	18	Довжина зображення, крапки
Height	4	22	Висота зображення, балів
Planes	2	26	Кількість рівнів (частіше один)
BitCount	2	28	Глибина кольору
Compression	4	30	Тип стиснення зображення (0 – нестисне зображення)
SizeImage	4	34	Розмір зображення, у байтах
XpelsPerMeter	4	38	Горизонтальна роздільна здатність зображення, точка, на метр
YpelsPerMeter	4	42	Роздільна здатність зображення по вертикалі, точка, на метр

ColorsUsed	4	46	Кількість використаних кольорів (при значенні 0 максимальна кількість кольорів використовується в реєстрі)
ColorsImportant	4	50	Кількість основних кольорів (зі значенням 0 усі кольори вважаються важливими)
ColorTable (палітра)			
ColorTable	1024	54	256 елементів по 4 байти
BitMap Array			
Image	Довжина:	1078	Зображення із записаними лініями

Оскільки формат BMP зберігає зображення без втрати кольору, стеганографія виконується в молодших бітах пікселів зображення за допомогою алгоритму LSB.

2.5 Структура формату JPEG

JPEG — один із популярних графічних форматів, який використовується для зберігання фотозображень і подібних зображень. Файли, що містять дані JPEG, зазвичай мають розширення .jpg, .jfif, .jpe або .jpeg. Однак .jpg є найпопулярнішим із них на всіх платформах. Файл JPEG містить послідовність вказівників, кожен з яких починається з байта 0xFF, що вказує на початок вказівника. Деякі маркери складаються лише з цієї пари байтів, тоді як інші містять додаткові дані, що складаються з двох байтів, що вказують на довжину покажчика, включаючи байти довжини. Така файлова структура дозволяє швидко знайти вказівник з необхідними даними (наприклад, довжина рядка, кількість рядків і кількість колірних компонентів стиснутого зображення). Тепер перерахуємо основні показники JPEG:

– 0xFF 0xD8 SOI - початок закодованої частини зображення;

– 0xFF 0xC0 SOF0 - вказує на те, що зображення закодовано в базовому

режимі за допомогою кодів DCT і Хаффмана. Покажчик містить довжину та ширину зображення, кількість компонентів (рівно 8 біт для кожного компонента) і співвідношення сторін компонентів (наприклад, 4:2:0);

– 0xFF 0xC1 SOF1 - вказує, що зображення закодовано в розширеному режимі з використанням кодів DCT і Хаффмана. Покажчик містить довжину і ширину зображення, кількість компонентів (кількість біт на компонент 8 або 12) і співвідношення компонентів (наприклад, 4:2:0);

– 0xFF 0xC2 SOF2 - вказує, що зображення закодовано в прогресивному режимі з використанням кодів DCT і Хаффмана. Покажчик містить довжину і ширину зображення, кількість компонентів (кількість біт на компонент 8 або 12) і співвідношення компонентів (наприклад, 4:2:0);

– 0xFF 0xC4 DHT – цей покажчик містить одну або більше таблиць Хаффмана;

– 0xFF 0xDB DQT - всередині цього маркера є одна або більше таблиць квантування;

– 0xFF 0xDA SOS – початок першого або наступного сканування зображення, зліва направо, зверху вниз. Якщо ми маємо справу з базовим або розширеним режимом кодування, використовується одне сканування. У прогресивному режимі використовується кілька сканерів. Можливо, маркер SOS розділяє інформаційну (заголовок) і зашифровану (стиснуті дані зображення) частини зображення;

– 0xFF 0xFE COM - Маркер коментаря;

– 0xFF 0xD9 EOI - Кінець закодованої частини зображення.

На відміну від форматів BMP і PNG, формат JPEG зберігає зображення з втратою кольору. З цієї причини ми не можемо виконати стеганографія в пікселях за допомогою методу LSB, який ми можемо використовувати у форматах PNG і BMP. Щоб виконати стеганографія, необхідно розуміти структуру формату JPEG. Існує 3 типи формату JPEG: базовий, розширений і прогресивний. Ми будемо використовувати стеганографію в базовому режимі(extended), оскільки вона більше підходить до нашого завдання. Щоб краще зрозуміти формат, давайте подивимося на невелику (для зручності перегляду) фавіконку Google у форматі jpeg. Як я вже

згадував, зображення розділене на вказівники, довжина яких становить 2 байти, а перший байт — [FF]. Майже всі вказівники зберігають свою довжину в наступних 2 байтах після вказівника. На рисунку 2.9 показано зовнішній вигляд нашого фавікону в 16-бітній системі. Для зручності маркери пофарбовані в червоний колір. Перш ніж розглядати приклад, давайте трохи поговоримо про алгоритм стиснення формату JPEG. Стиснення відбувається в 6 основних етапів:

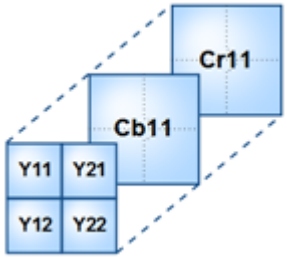

Рисунок 2.7 - Канали Cb і Cr стиснуті в 2 рази

– зображення перетворюється з колірної гами RGB на гамму YCbCr;

– часто канали Cb і Cr стискаються в 2 рази. У цьому випадку кожен канал Y отримує усереднене значення каналів Cb і Cr(Рисунок 2.7);

– на цьому етапі значення каналів розбиваються на блоки 8x8;

– кожен блок піддається дискретному косинусному перетворенню. В результаті ми отримуємо масив коефіцієнтів 8x8, коефіцієнт в лівому куті масиву називається DC (він є найважливішим коефіцієнтом і вважається середнім значенням всіх значень), а решта 63 значення - називається AC;

– отримані коефіцієнти квантуються, кожен з яких множиться на елементи масиву квантування (кожен канал має власний масив квантування);

– на завершальному етапі отримані масиви кодуються методом Хаффмана. Закодовані масиви зберігаються в порядку Yi Cbi Cri, наприклад, якщо канали Cb і Cr стискаються 2 рази Y00 Y10 Y01 Y11 Cb00 Cr00 Y20...(Рисунок 2.8).

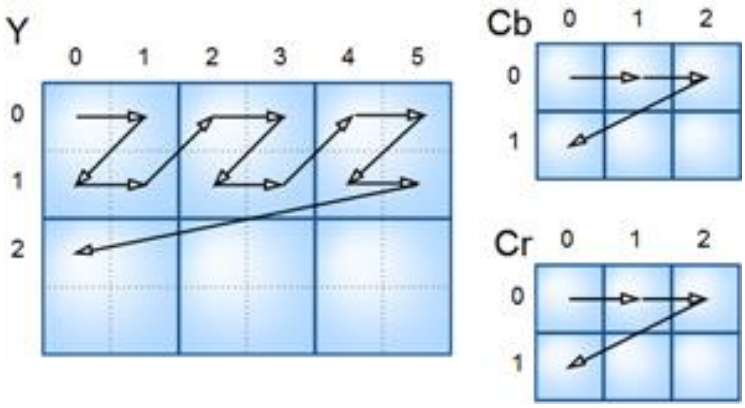

Рисунок. 2.8 – Розташування каналів Y, Cb, Cr у JPEG

Тепер розглянемо наш приклад:

FF	D8	FF	FE	00	04	3A	29	FF	DB	00	43	00	A0	6E	78
8C	78	64	A0	8C	82	8C	B4	AA	A0	BE	F0	FF	FF	F0	DC
DC	F0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	DB	00
43	01	AA	B4	B4	F0	D2	F0	FF	FF	FF	FF	FF	FF	FF	FF
FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
FF	FF	FF	C0	00	11	08	00	10	00	10	03	01	22	00	02
11	01	03	11	01	FF	C4	00	15	00	01	01	00	00	00	00
00	00	00	00	00	00	00	00	00	00	03	02	FF	C4	00	1A
10	01	00	02	03	01	00	00	00	00	00	00	00	00	00	00
00	01	00	12	02	11	31	21	FF	C4	00	15	01	01	01	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	01	FF
C4	00	16	11	01	01	01	00	00	00	00	00	00	00	00	00
00	00	00	00	11	00	01	FF	DA	00	0C	03	01	00	02	11
03	11	00	3F	00	AE	E7	61	F2	1B	D5	22	85	5D	04	3C
82	C8	48	B1	DC	BF	FF	D9								

Рисунок 2.9 – Фавікон Google у 16-кова система числення

Ми розглянемо маркери, за допомогою яких буде здійснюватися стеганографія, це:

– 0xFF 0xC0 SOF0 - Базовий DCT;

– 0xFF 0xDB DQT - таблиця квантування;

– 0xFF 0xC4 DHT - таблиця Хаффмана;

– 0xFF 0xDA SOS (Start of Scan) – початок сканування.

2.5.1 Базовий DCT

```
FF C0 00 11 08 00 10 00 10 03 01 22 00 02
11 01 03 11 01
```

Цей маркер SOF0 означає, що зображення закодовано базовим методом:

– [00 11] Довжина. 17 байт;

– [08] Точність. 8 біт. У цьому режимі воно завжди дорівнює 8. Вказує бітрейт каналу;

– [00 10] Висота зображення. 0x10 = 16;

– [00 10] Довжина зображення. 0x10 = 16;

– [03] Кількість каналів. 3. Переважно Y, Cb, Cr або R, G, B.

1-ий канал:

– [01] індентифікатор. 1;

– [2_] Горизонтальне проріджування (H1). 2;

– [_2] Вертикальне проріджування (V1). 2;

– [00] Ідентифікатор таблиці квантування. 0.

2-ий канал:

– [02] ідентифікатор. 2:

– [1_] Горизонтальне проріджування (H2). 1;

– [_1] Вертикальне проріджування (V2). 1;

– [01] Ідентифікатор таблиці квантування. 1.

3-ій канал:

– [03] ідентифікатор. 3;

– [1_] Горизонтальне проріджування (H3). 1;

– [_1] Вертикальне проріджування (V3). 1.

Ми знаходимо $H_{max} = 2$ і $V_{max} = 2$. Канал і буде стиснутий у H_{max} / H_i разів по горизонталі та у V_{max} / V_i разів по вертикалі.

2.5.2 DQT - таблиця квантування

								FF	DB	00	43	00	A0	6E	78
8C	78	64	A0	8C	82	8C	B4	AA	A0	BE	F0	FF	FF	F0	DC
DC	F0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF			

– [00 43] Довжина покажчика. 0x43 = 67 байтів;

– [0_] Довжина значення. 0 (0 — 1 байт, 1 — 2 байти);

– [_0] Ідентифікатор таблиці. 0.

Решта 64 байти потрібно заповнити в масив 8x8. Значення заповнюються в масиві за допомогою методу зигзагоподібного порядку, як показано на Рисунку 2.10.

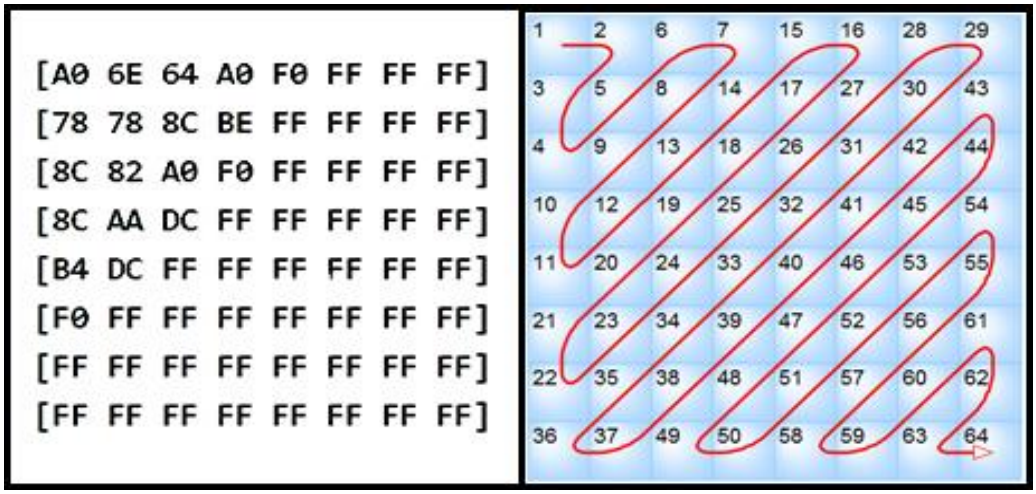

Рисунок 2.10 – Метод порядку зигзаг

2.5.3 DHT - таблиця Хаффмана

У цьому розділі зберігаються значення та коди, закодовані Хаффманом.

FF C4 00 15 00 01 01 00 00 00 00
00 00 00 00 00 00 00 00 00 00 03 02

– [00 15] Довжина покажчика. 21 байт;

– [0_] Вказує на коефіцієнт таблиці. (0 – коефіцієнт DC, 1 - коефіцієнт AC);

– [_0] Ідентифікатор таблиці. 0.

Наступні 16 значень:

Довжина коду 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

Кількість кодів [01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00]

Зверніть увагу, що в розділі зберігаються лише довжини кодів, коди потрібно шукати окремо. Отже, у нас є один код довжини 1 і 1 код довжини 2. Отже, у нас є 2 коди, більше кодів у цій таблиці немає. Потім після 16 байт йдуть самі коди. Значення мають довжину в один байт, тому ми читаємо 2 байти:

– [03] — перше значення коду;

– [02] — друге значення коду.

У нас є ще 3 маркери Хаффмана, вони розшифровуються так само. Маючи всі таблиці Хаффмана, необхідно за допомогою отриманих таблиць побудувати бінарне дерево. Побудувавши дерево, ми вже можемо декодувати коди. Отримавши коди, ми розміщуємо їх у таблиці 8x8 зигзагом. Алгоритм побудови дерева дуже простий, незалежно від того, на якому вузлі ми знаходимося, ми завжди намагаємося додати значення до лівої гілки. Якщо зайнято, то на праву гілку. Якщо на місці немає значення, ми повертаємось на вищий рівень і пробуємо звідти. Зупинитись потрібно на рівні рівному довжині коду. Ліві гілки відповідають 0, а праві – 1.

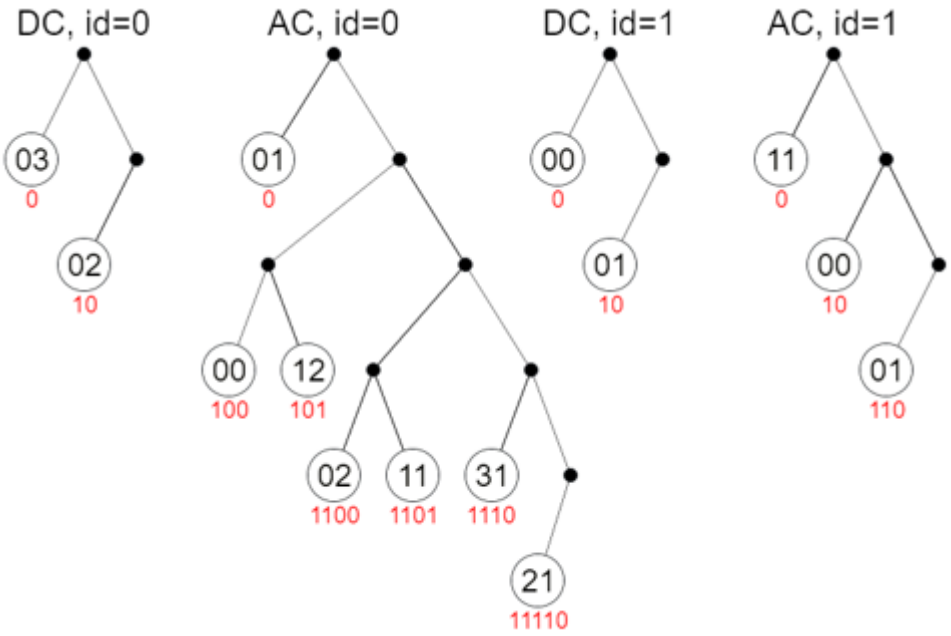

Рисунок 2.11 – Вигляд дерева Хаффмана

У кружечках на рисунку 2.11 вказані значення, отримані в таблицях, під кружечками - коди, якими кодується фактичний вміст картинки.

2.5.4 SOS (Start of Scan) – початок сканування

							FF	DA	00	0C	03	01	00	02	11
03	11	00	3F	00	AE	E7	61	F2	1B	D5	22	85	5D	04	3C
82	C8	48	B1	DC	BF										

– [00 0C] Довжина. 12 байт;

– [03] кількість каналів. У нас є 3 канали: Y, Cb, Cr.

1-ий канал:

– [01] ідентифікатор каналу. 1 (Y);

– [0_] ідентифікатор таблиці Хаффмана для DC.коефіцієнтів: 0;

– [_0] ідентифікатор таблиці Хаффмана для AC коефіцієнтів: 0.

2-ий канал:

– [02] ідентифікатор каналу. 2 (Cb);

– [1_] ідентифікатор таблиці Хаффмана для DC.коефіцієнтів: 1;

– [_1] ідентифікатор таблиці Хаффмана для AC коефіцієнтів: 1.

3-ій канал:

– [03] ідентифікатор каналу. 3 (Cr);

– [1_] ідентифікатор таблиці Хаффмана для DC.коефіцієнтів: 1;

– [_1] ідентифікатор таблиці Хаффмана для AC коефіцієнтів: 1.

[00], [3F], [00] - Start of spectral or predictor selection, End of spectral selection, Successive approximation bit position.. Ці значення використовуються тільки для прогресивного режиму.

Звідси до кінця (вказівник [FF D9]) кодуються дані.

Тепер подивимося на значення після 12-го байта. Перших 33 бітів буде достатньо для побудови першого масиву коефіцієнтів.

Hex:	[AE]	[E7]	[61]	[F2]	[1B]
Bin:	10101110	11100111	01100001	11110010	00011011

2.5.5 Знаходження DC-коефіцієнтів

1. Читаємо послідовність бітів, якщо зустрічаємо 2 байти [FF 00], то це не покажчик, а просто байт [FF]. Після кожного біта ми рухаємося по дереву Хаффмана (з відповідним ідентифікатором). У випадку 0 ми рухаємося вздовж гілки вліво, а у випадку 1 – вправо і так далі, поки не досягнемо кінцевого вузла.

10101110111001110110000111100100

2. Беремо значення вузла. Якщо дорівнює 0, то коефіцієнт дорівнює 0, записуємо його в масив і продовжуємо пошук інших коефіцієнтів. У нашому випадку 02 вказує на довжину коефіцієнта. Тобто зчитуємо наступні 2 біти, які будуть коефіцієнтом:

10101110111001110110000111100100

1. Якщо перша цифра значення дорівнює 1 у двійковій системі, то ми не робимо зміну DC = <значення>. В іншому випадку ми виконуємо таку операцію DC = <значення> - 2 ^ <довжина значення> + 1.

2.5.6 Знаходження AC – коефіцієнтів

DC коефіцієнт зчитується лише один раз, після чого зчитуються коефіцієнти AC.

1. Подібно до знаходження коефіцієнта DC. Продовжуємо читати послідовність.

10101110111001110110000111110010 0

2. Беремо значення вузла. Якщо він дорівнює 0, це означає, що решта значень матриці потрібно заповники нулями. Наступні байти кодують наступну матрицю. У нашому випадку значення вузла дорівнює 0x31:

a) Значення першої половини байта вказує, скільки 0 потрібно додати до матриці. У нашому випадку 3 шт;

b) Значення другої половини байта вказує на довжину коефіцієнта. У нашому випадку 1, тобто наступний біт - це коефіцієнт.

10101110111001110110000111110010 0

3. Подібно до пункту 3 коефіцієнта DC.

Зчитувати коефіцієнт AC необхідно до тих пір, поки ми не зустрінемо нульове значення вузла або поки матриця не заповниться. Таким чином, масив наших коефіцієнтів матиме такий вигляд:

10101110111001110110000111110010 0

$$\begin{bmatrix} 2 & 0 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Таким же чином ми отримуємо решта три зубці Y. Але для хвиль Y коефіцієнти DC є не коефіцієнтами DC, а їх різницею між коефіцієнтами попередніх масивів:

– DC для 2-го масиву. 2 + (-4) = -2;

– DC для 3-го масиву. -2 + 5 = 3;

– DC для масиву 4. 3 + (-4) = -1.

Канали Cb і Cr схожі. Ми розглянемо тільки канали Y1 і Cb, Cr.

$$\begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

2.5.7 Квантування

Ми досягли етапу квантування. Елементи масиву необхідно помножити на елементи масиву квантування. Як ми з'ясували з блоку S0F0, Y-хвилі множаться на масив з ідентифікатором 0, а Cb, Cr на масив з ідентифікатором 1 відповідно. Тож після виконання множення ми отримуємо 4 масиви Y і масиви Cb, Cr. Розглянемо лише хвилі Y_1 і Cb, Cr. Після етапу квантування всі коефіцієнти DC підсумовуються до 1024.

$$\begin{bmatrix} 1344 & 0 & 300 & 0 & 0 & 0 & 0 & 0 \\ 0 & 120 & 280 & 0 & 0 & 0 & 0 & 0 \\ 0 & -130 & -160 & 0 & 0 & 0 & 0 & 0 \\ 140 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 854 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 180 & 210 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1024 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 180 & -210 & 0 & 0 & 0 & 0 & 0 & 0 \\ 240 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Таким чином ми прийшли до оберненого дискретного косинусного перетворення(2.1).

$$S_{yx} = \frac{1}{4} \sum_{u=0}^{7} \sum_{v=0}^{7} C_u C_v S_{vu} \cos\frac{(2x+1)u\pi}{16} \cos\frac{(2y+1)v\pi}{16} \qquad (2.1)$$

де Y і X — номери стовпців і рядків коефіцієнта, U і V є змінними, які змінюються в межах 0-7, якщо U дорівнює 0, тоді $C_u=1/\sqrt{2}$, інакше $C_u = 1$, якщо V дорівнює 0, тоді $C_v=1/\sqrt{2}$, інакше $C_v = 1$, S_{vu} - наш коефіцієнт

Y1

$$\begin{bmatrix} 138 & 92 & 27 & -17 & -17 & 28 & 93 & 139 \\ 136 & 82 & 5 & -51 & -55 & -8 & 61 & 111 \\ 143 & 80 & -9 & -77 & -89 & -41 & 32 & 86 \\ 157 & 95 & 6 & -62 & -76 & -33 & 36 & 86 \\ 147 & 103 & 37 & -12 & -21 & 11 & 62 & 100 \\ 87 & 72 & 50 & 36 & 37 & 55 & 79 & 95 \\ -10 & 5 & 31 & 56 & 71 & 73 & 68 & 62 \\ -87 & -50 & 6 & 56 & 79 & 72 & 48 & 29 \end{bmatrix}$$

Cb

$$\begin{bmatrix} 60 & 52 & 38 & 20 & 0 & -18 & -32 & -40 \\ 48 & 41 & 29 & 13 & -3 & -19 & -31 & -37 \\ 25 & 20 & 12 & 2 & -9 & -19 & -31 & -37 \\ -4 & -6 & -9 & -13 & -17 & -20 & -23 & -25 \\ -37 & -35 & -33 & -29 & -25 & -21 & -18 & -17 \\ -67 & -63 & -55 & -44 & -33 & -22 & -14 & -10 \\ -90 & -84 & -71 & -56 & -39 & -23 & -11 & -4 \\ -102 & -95 & -81 & -62 & -42 & -23 & -9 & -1 \end{bmatrix}$$

Cr

$$\begin{bmatrix} 19 & 27 & 41 & 60 & 80 & 99 & 113 & 120 \\ 0 & 6 & 18 & 34 & 51 & 66 & 78 & 85 \\ -27 & -22 & -14 & -4 & 7 & 17 & 25 & 30 \\ -43 & -41 & -38 & -34 & -30 & -27 & -24 & -22 \\ -35 & -36 & -39 & -43 & -47 & -51 & -53 & -55 \\ -5 & -9 & -17 & -28 & -39 & -50 & -58 & -62 \\ 32 & 26 & 14 & -1 & -18 & -34 & -46 & -53 \\ 58 & 50 & 36 & 18 & -2 & -20 & -34 & -42 \end{bmatrix}$$

Оскільки стиснення проводилося із співвідношенням Cb:Y, Cr:Y 2:1 (оскільки ми не заповнювали SOF0 - базовий блок DCT), то кожен піксель описується такою формулою pixel $(Y_{ij}, Cb_{[i/2,j/2]}, Cr_{[i/2,j/2]})$ останнім етапом залишається перехід від YCbCr до системи RGB.

1. R=Y+1,402*Cr;
2. G=Y-0,34414*Cb-0,71414*Cr;
3. B=Y+1,772*Cb.

Якщо значення виходять за діапазон [0, 255], то беремо граничні значення. В результаті ми отримуємо таблиці каналів R, G, B верхнього лівого квадрата 8x8 нашого зображення.

R

255	248	194	148	169	215	255	255
255	238	172	115	130	178	255	255
255	208	127	59	64	112	208	255
255	223	143	74	77	120	211	255
237	192	133	83	85	118	184	222
177	161	146	132	145	162	201	217
56	73	101	126	144	147	147	141
0	17	76	126	153	146	127	108

G

231	185	117	72	67	113	171	217
229	175	95	39	28	76	139	189
254	192	100	31	15	63	131	185
255	207	115	46	28	71	134	185
255	241	175	125	112	145	193	230
226	210	187	173	172	189	209	225
149	166	191	216	229	232	225	220
72	110	166	216	238	231	206	186

B

255	255	249	203	178	224	255	255
255	255	226	170	140	187	224	255
255	255	192	123	91	138	184	238
255	255	208	139	103	146	188	239
255	255	202	152	128	161	194	232
255	244	215	200	188	205	210	227
108	125	148	172	182	184	172	167
31	69	122	172	191	183	153	134

Ось і закінчимо наш розглянутий приклад. Перш за все, наше зображення JPEG перетворюється на базовий метод. У розглянутому прикладі ми знаходимо коефіцієнти за допомогою дерева Хаффмана. За допомогою цих коефіцієнтів здійснюється стеганографія. Молодші розряди коефіцієнтів замінюються бітами секретної інформації. Спотворення залежить від того, скільки бітів ми використовуємо з кінця коефіцієнта. Коефіцієнт DC не змінюється, оскільки він

спричиняє значні спотворення зображення.

2.6 Структура Аудіоформату Wav

Файли з розширенням WAV або WAVE (Waveform Audio file) є стандартним форматом для зберігання оцифрованої музики та звуку. Найчастіше файли WAV зберігають звук у вихідному вигляді без стиснення, але вони підтримують стиснення, тому звук або музика, що мають розширення WAV, не завжди оригінальні. WAV займає більше пам’яті, ніж популярні MP3 або AAC, тому дуже мало людей використовують цей формат для зберігання власної аудіотеки на комп’ютері. Однак якщо ви займаєтеся монтажем звуку і музики або створюєте звукові ефекти для комп'ютерних ігор, то формат WAV буде найкращим варіантом для зберігання. WAV складається з двох частин:

– Заголовок;

– Дані.

Отже, давайте детально обговоримо частину заголовка файлу .Wav. Таблиця 2.3 показує структуру заголовка.

Таблиця 2.3 - Структура формату WAV

Окупаційні біти	Поле	Опис
0-3 (4 байти)	chunkId	Містить символ "RIFF" у кодуванні ASCII: 0x52494646. Це початок ланцюжка RIFF.
4-7 (4 байти)	chunkSize	Це розмір файлу мінус 8, тобто без урахування полів chunkId і chunkSize.
8-11 (4 байти)	format	Містить символи "WAVE" 0x57415645
12-15 (4 байти)	subchunk1Id	Містить символи "fmt" 0x666d7420

16-19 (4 байти)	subchunk1Size	Містить кількість байтів, які містять дані про файл
20-21 (2 байти)	audioFormat	Містить код, який вказує, чи стиснутий файл, найчастіше це буде 1, що означає PCM/нестиснутий.
22-23 (2 байти)	numChannels	Кількість каналів. Моно = 1, Стерео = 2 тощо.
24-27 (4 байти)	sampleRate	Розмір дисертації. 8000 Гц, 44100 Гц тощо.
28-31 (4 байти)	byteRate	Кількість байтів, переданих за секунду під час випуску.
32-33 (2 байти)	blockAlign	Кількість байтів на вибірку, включаючи всі канали.
34-35 (2 байти)	bitsPerSample	Кількість бітів у вибірці. Так звана «глибина» або точність звуку. 8 біт, 16 біт тощо.
36-39 (4 байти	subchunk2Id	Містить символи "дані" 0x64617461
40-43 (4 байти)	subchunk2Size	Кількість байтів в області даних
44	data	Прямі дані WAV.

В основному заголовок виглядає так, але можуть бути винятки. Після послідовності символів даних починаються дані нашого аудіофайлу, а наступні 4 байти показують довжину даних. Знаючи структуру WAV, береться значення blockAlign (як ми вже згадували, blockAlign показує, скільки байтів описує один семпл).

Якщо у нас 2 канали, перша половина нашого blockAlign показуватиме перший канал, а друга половина – другий канал. Маючи канали, ми використовуємо метод LSB і ховаємо нашу секретну інформацію в молодших бітах каналів.

Висновок до розділу 2

В розділі 2 було детально розглянуто ті формати файлів таких як PNG, BMP, JPEG і WAV, які будуть використовуватися контейнерами для нашої конфіденцальної інформації. Були розглянуті основні алгоритми і методи, використовувані для вбудовування та вилучення прихованої інформації. В нашій стеганографічній системі буде використовуватися метод LSB.

Вивчення розширеного стандарту шифрування (AES) дало нам можливість застосовувати потужні алгоритми шифрування для забезпечення конфіденційності вбудованих даних. Метод останнього біта (LSB) виявився ефективним для вбудовування інформації у незначущі біти пікселів або семплів звукових сигналів без помітних змін.

Дослідження структур форматів файлів PNG, BMP, JPEG і WAV дозволило нам розібратися з їхніми особливостями та можливостями для стеганографічного використання. Ми вивчили, як знаходити та змінювати певні частини цих файлових форматів, при цьому вносячи незначні зміни початкового контейнера

Результати дослідження та аналізу структур форматів файлів були використані при проектуванні стеганографічного застосунку. Розроблені алгоритми та функціональність застосунку дозволяють ефективно вбудовувати та вилучати приховану інформацію у зазначені формати файлів.

Висновок з розділу 2 вказує на успішне виконання цього етапу дипломної роботи, де було зроблено докладний аналіз, вивчено структури та реалізовано практичну реалізацію стеганографії у різних форматах файлів. Цей розділ становить основу для подальшого розроблення та вдосконалення стеганографії.

3 ПРОЕКТУВАННЯ ЗАСТОСУНКУ

3.1 Опис використовуваного середовища

3.1.1 Visual Studio

Visual Studio була створена організацією Microsoft, яка являє собою інтегроване програмне середовище та ряд інших інструментів. Інтегроване середовище розробки (IDE) — це багатоцільова програма, яку можна використовувати для розробки програмного забезпечення. Окрім стандартного редактора та відладчика, які доступні у великій кількості IDE, Visual Studio містить компілятор, систему автозаповнення коду (IntelliSense) та зручний графічний інтерфейс. Visual Studio дозволяє розробляти консоль, GUI, Windows Forms та інші програми(Рисунок 3.1).

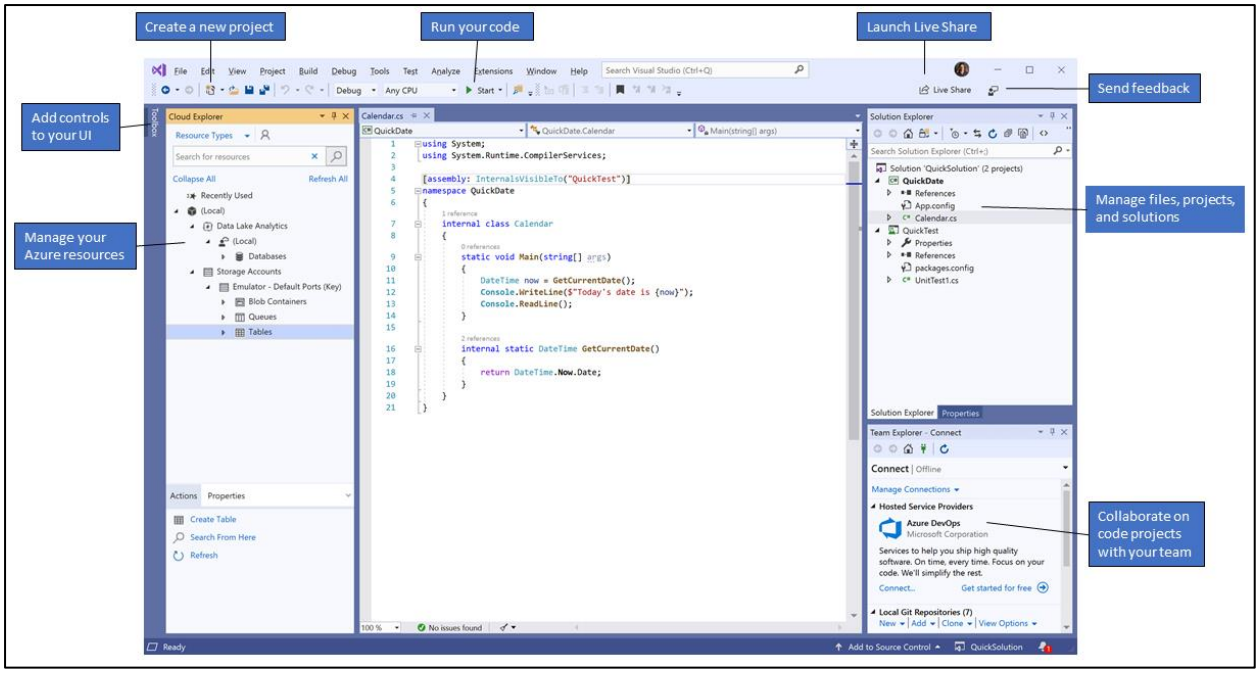

Рисунок 3.1 – Середовище Visual Studio з відкритим проектом і кількома основними вікнами інструментів

3.1.2 Мова програмування C#

C# — це елегантна об’єктно-орієнтована мова, безпечна для типів, яка дозволяє розробникам створювати різноманітні безпечні та надійні програми. C# можна використовувати для створення клієнтських програм Windows, веб-служб XML, клієнт-серверних програм, програм баз даних тощо. C# надає розширений редактор коду, зручні конструктори інтерфейсу користувача, інтегрований налагоджувач та багато інших інструментів.

Мова C# дуже багата, але в той же час проста і легка для вивчення. Фігурні дужки C# миттєво впізнають усі, хто знайомий із мовами програмування C, C++ або Java. Програмісти, які розуміють будь-яку з перерахованих вище мов, зазвичай швидко й ефективно працюють із C#. У C# мові полегшено синтаксис з ++ і додані переселення (enum), делегати, пряма доступність до пам'яті.. C# підтримує універсальні методи та типи, які забезпечують вищий рівень безпеки та продуктивності. Вирази LINQ забезпечують дуже зручну мовну структуру для строго типізованих запитів.

C# вважається об’єктно-орієнтованою мовою, що означає, що C# підтримує інкапсуляцію, успадкування та поліморфізм. Усі змінні та методи, включаючи метод Main, який є початком запису програми, інкапсулюються під час визначення класу. Клас успадковує безпосередньо від одного батька, але може реалізувати будь-яку кількість інтерфейсів. Методи, які замінюють віртуальні методи батьківського класу, повинні містити ключове слово override, щоб уникнути випадкового перевизначення. У C# структура (struct) подібна до полегшених класів. На відміну від класів, структури реалізуються в стеку, і він реалізує інтерфейси, але не підтримує успадкування.

Крім основних принципів об'єктно-орієнтованого програмування, C# спрощує роботу з програмними компонентами за допомогою кількох нових мовних конструкцій, серед яких:

– делегати, які вмикають безпечні типи сповіщень про події;

– властивості, які надають неявний доступ до закритих членів класу;

– атрибути, які надають декларативні метадані про змінні на виході програми;

– внутрішні коментарі для документів XML;

– LINQ, який надає вбудовані можливості для запитів до різних джерел даних.

Для взаємодії з іншими програмами Windows, такими як COM-об’єкти або бібліотеки DLL Win32, ми можемо використовувати процес C# під назвою Interop. Interop дозволяє програмам C# робити майже все, що можливо з машинним кодом C++. C# навіть підтримує покажчики та концепцію «небезпечного» коду для випадків, коли важливий прямий доступ до пам’яті.

Функціональна структура C# простіша, ніж C або C++, і гнучкіша, ніж Java. Не використовуються окремі заголовки файлів, і немає необхідності оголошувати методи та типи в спеціальному порядку. Вихідний файл C# може визначати будь-яку кількість класів, структур, інтерфейсів і подій.

3.1.3 .NET Framework

Програми на C# працюють на платформі .NET Framework, вбудованому компоненті Windows, який включає віртуальну систему середовища виконання під назвою Common Language Runtime (CLR), а також набір інтегрованих класів і бібліотек. CLR від Microsoft є реалізацією міжнародного стандарту Common Language Infrastructure (CLI), який є середовищем створення, запуску та розробки, де мови програмування працюють разом без бар’єрів.

Вихідний код C# компілюється в проміжну мову (IL), яка відповідає специфікації CLI. Код і ресурси IL, включаючи растрові зображення та рядки, зберігаються на диску як виконуваний файл (зазвичай із розширенням .exe або .dll). Такі файли називають колекцією. Колекція містить маніфест, який надає інформацію про типи, версію колекції, вимоги безпеки, мову та регіональні налаштування.

Коли виконується програма C#, CLR завантажує колекцію та виконує різні операції залежно від інформації, що зберігається в маніфесті. Якщо всі вимоги безпеки виконано, середовище CLR виконує JIT-компіляцію з IL-коду в машинний код. CLR також виконує інші операції, такі як автоматичне збирання сміття, обробка винятків і керування ресурсами. На малюнку 2 показано взаємозв'язки між програмними файлами C#, бібліотеками класів .NET Framework, збірками та середовищем CLR під час компіляції та виконання. Взаємодія між мовами програмування вважається ключовою особливістю .NET Framework. Оскільки код IL відповідає специфікації загального типу (CTS), код IL, згенерований з C#, може взаємодіяти з кодом, згенерованим у .NET Visual Basic, Visual C++ та інших з більш ніж 20 мовами, сумісними з CTS. Один набір може містити декілька модулів, які записуються як . На мови NET і всі типи можна посилатися так, ніби вони написані однією мовою.

3.2 Архітектура проекту

Користувач починає з порожнього контейнера, який хоче приховати секретну інформацію. Контейнер проходить процес аналізу формату, щоб визначити, який формат відповідає формату нашого контейнера (bmp/png wav і jpeg). Крім порожнього контейнера користувач також вводить секретний файл, який потрібно приховати. Ключ також вводиться користувачем для використання під час кодування секретного файлу. Саме кодування починається, використовуючи аналізовані формати контейнера та методи кодування. Секретний файл і ключ перетворюються на певний формат, який передається одержувачу. Після кодування створюється заповнений контейнер, що містить приховану секретну інформацію. Контейнер зазнає процесу декодування, використовуючи відповідні методи декодування. Якщо ключ правильний, секретний файл декодується та витягується з контейнера. Потім контейнер перевіряється, чи є в ньому прихована інформація. Якщо стеганодетектор виявляє приховані дані, контейнер містить секретний файл і починає витягувати секретні файли. Якщо стеганодетектор не виявляє жодних прихованих даних, це це означає, що контейнер

був порожній або не містить секретного файлу.

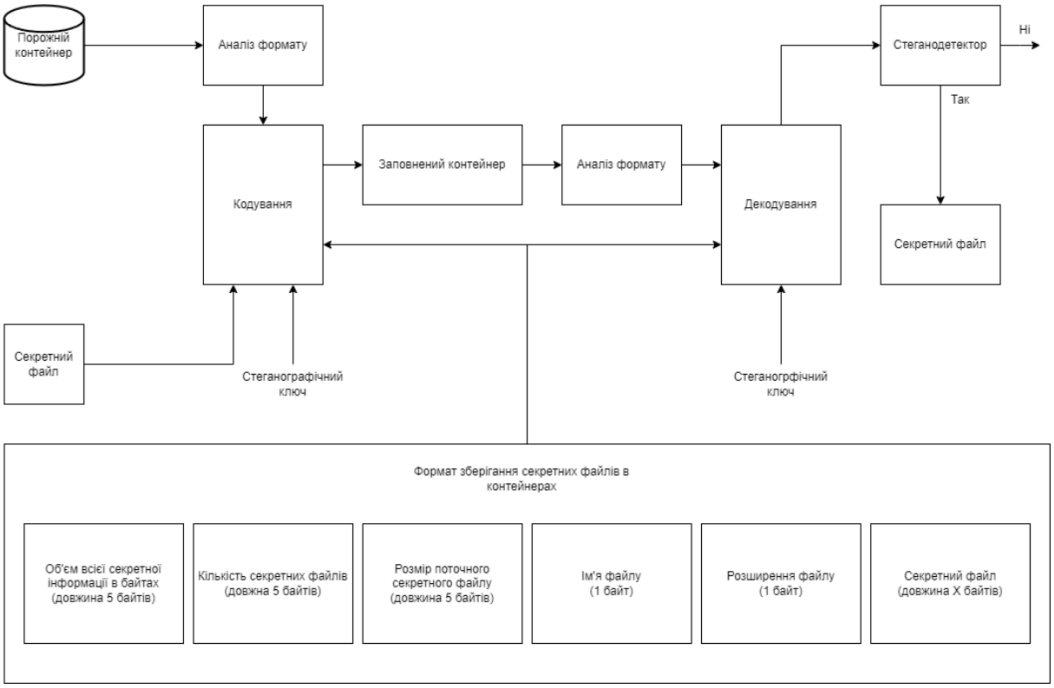

Рисунок 3.2 - Загальна схема роботи програми

Крім декодування секретного файлу, користувач може використовувати той самий або інший ключ для шифрування секретної інформації. Якщо секретна інформація шифрована вона розшифрується безпосередньо після декодування. Ця схема(Рисунок 3.2) описує послідовність дій, що відбуваються у програмі для приховування та вилучення секретної інформації з контейнера, використовуючи методи кодування та декодування.

Секретні файли зберігаються у контейнері структурою, яка відображена на рисунку 3.3, в якому вся наша інформація кодується таким чином:

– Перші 5 байтів зберігають обсяг байтів всіх секретних файлів;

– Після 5 байтів зберігають обсяг конкретного секретного файлу;

– Після 5 байтів зберігають розмір поточного секретного файлу;

– Після 1 байт зберігає ім'я файлу, якщо він більше 255 бітів, то береться перші 255 біти;

– Після 1 б зберігає розширення конкретного секретного файлу. Вона може

бути будь-якого типу;

– Потім іде бітовий потік нашого секретного файлу, розмір якого було визначено перед цим.

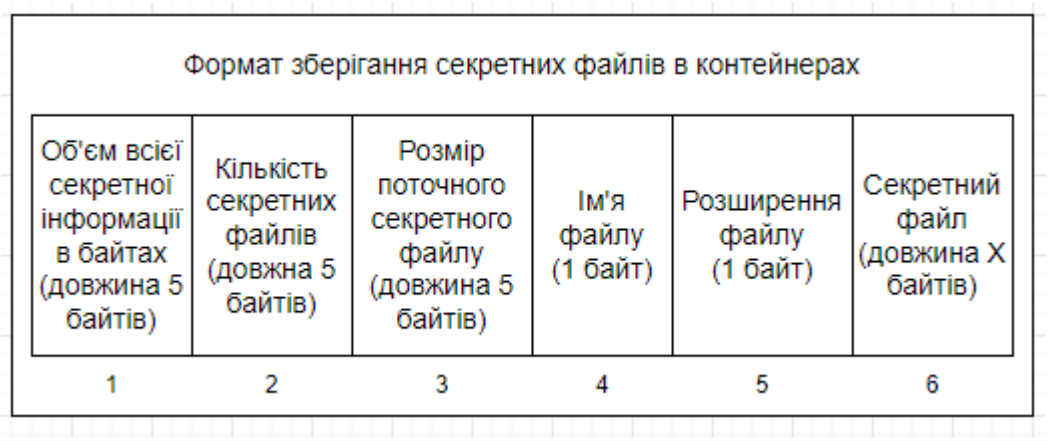

Рисунок 3.3 – Формат зберігання секретних файлів в контейнерах

3.3 Структура діаграми класів

Основним класом є MainForm, що є головним вікном програми. У цьому вікні вставивши ключ і вибравши тип (вбудовування/витяг) ми переходимо в StegPanel, який відповідає за обробку приховування та вилучення інформації зображень. StegPanel має зв'язки із кількома іншими класами. Він може взаємодіяти з класом BMP/PNG, щоб приховати або отримати інформацію зображення у форматі BMP або PNG.

Також він має зв'язок із класом JPEG, який використовує стеганографію у форматі JPEG з деревом Хаффманом (HuffTree) для приховування та вилучення інформації. StegPanel також використовує клас WAV для прихування або вилучення інформації з аудіофайлів у форматі WAV.

Крім того, StegPanel має можливість взаємодіяти з класом AES, що дозволяє зашифрувати та розшифрувати приховану інформацію з використанням алгоритму AES.

У цій діаграмі(Рисунок 3.4) показана взаємодія між класами у програмі, яка використовується для стеганографії та шифрування. Кожен клас відповідає певним

функціональностям і забезпечує можливість взаємодії коїться з іншими класами до виконання необхідних операцій.

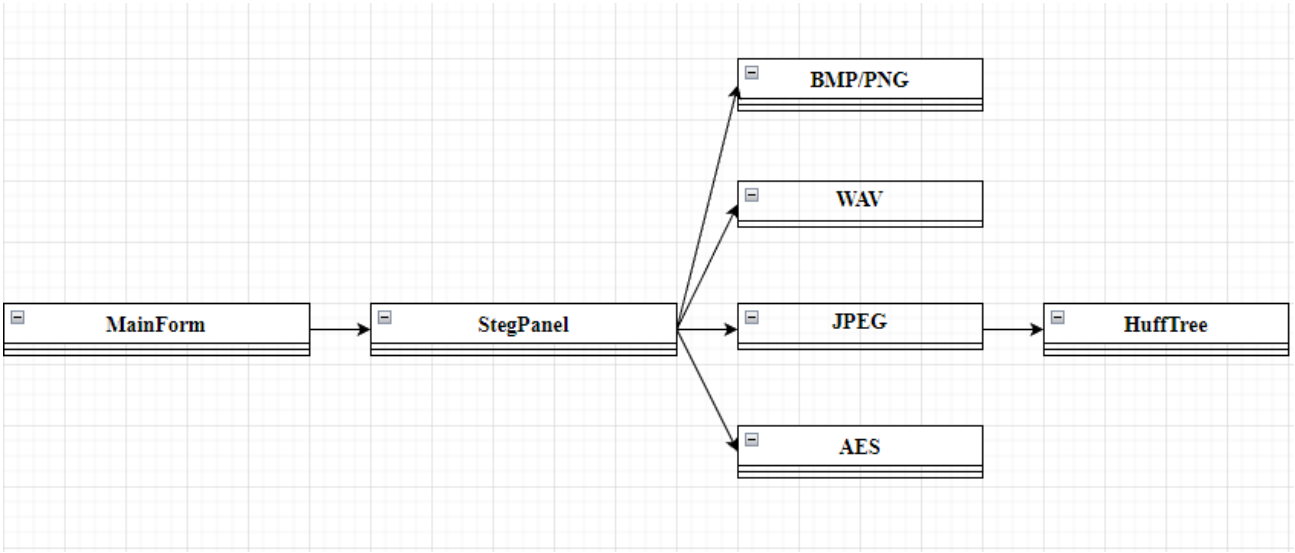

Рисунок 3.4 – Структура діаграми класів

3.3.1 Структура класу MainForm

MainForm представляє головну форму проекту, пов'язаного із цифровою стеганографією. Вона містить елементи керування, такі як кнопки (button1, pbHide, pbUnhide), мітку SteganographyKeyLabel та текстове поле textBoxStegKey.

Клас MainForm визначено відповідними полями та методами. Поля включають button1, components, pbHide, pbUnhide, SteganographyKeyLabel та textBoxStegKey, які, ймовірно, представляють елементи управління форми та інші компоненти, необхідні для функціонування проекту(Рисунок 3.5).

Методи класу MainForm включають Button1_Click, Dispose, InitializeComponent, KeyStringToLSBByte, MainForm, OpenNewForm, PbHide_Click, PbUnHide_Click, textBoxStegKey_TextChanged та ValidateKey. Ці методи відповідають за обробку подій, ініціалізацію компонентів, перетворення ключа на байти, створення нових форм та валідацію ключа. MainForm, є основним інтерфейсом проекту.

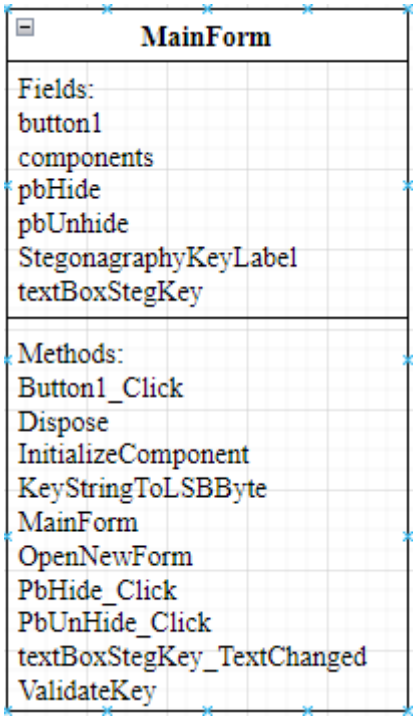

Рисунок 3.5 – Структура класу MainForm

3.3.2 Структура класу StegPanel

StegPanel представляє клас, пов'язаний із панеллю для цифрової стеганографії у проекті. Він містить різні поля та методи для управління стеганографічним процесом.

Поля класу StegPanel включають btnContainerRemove, btnRemove, btnStart, checkBox, cName, components, ContainerBtn, containerGridView, contCoords, cPath, cSize, dataGridView, encryptTextBox, folderBrowserDialog1, labelContainer pbPicture, SelectItemBtn, Size та StegKey. Ці поля представляють елементи управління, дані та стани, пов'язані з панеллю стеганографії. Наприклад, btnContainerRemove та btnRemove кнопки для видалення контейнера та видалення секретних файлів, btnStart - кнопка для запуску процесу стеганографії, checkBox - прапорець для включення AEC шифрування.

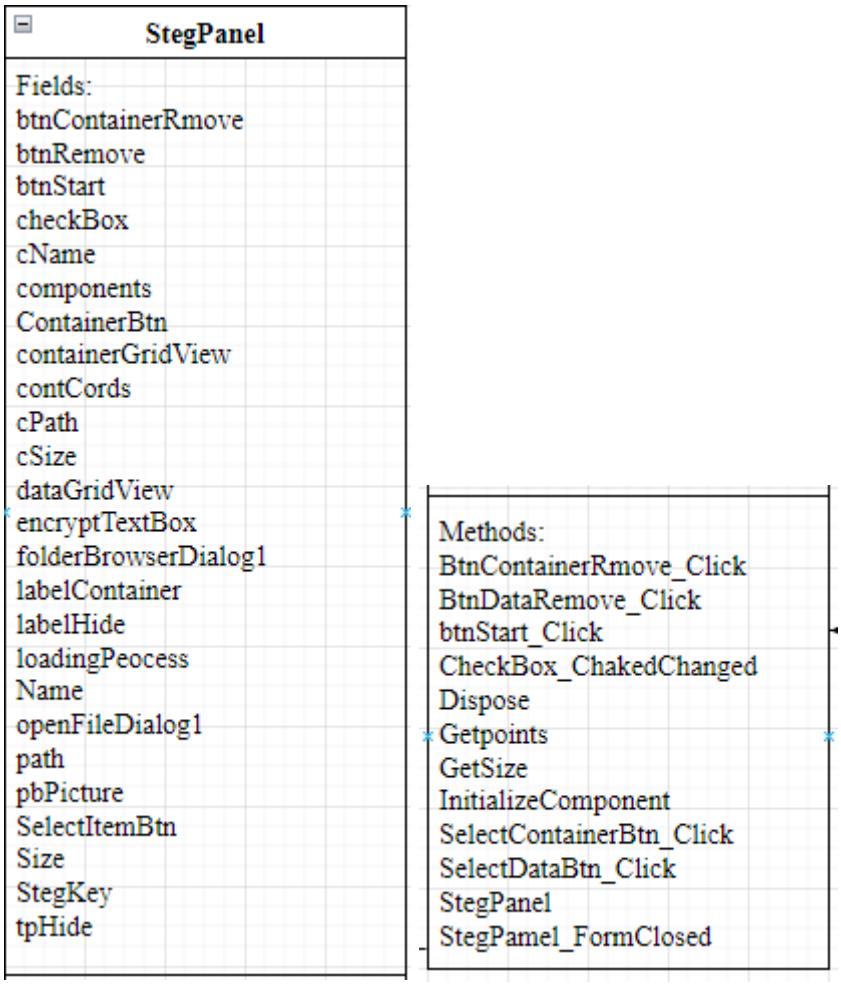

Рисунок 3.6 – Структура класу StegPanel

Методи класу StegPanel включають BtnContainerRemove_Click, BtnDataRemove_Click, btnStart_Click, CheckBox_ChakedChanged, Dispose, GetPoints, GetSize, InitializeComponent, SelectContainerBtn_Click, SelectDataBtn_Click, Steg. Ці методи відповідають за обробку подій, ініціалізацію компонентів, вибір елементів, налаштування, запуск процесу стеганографії та обробку закриття форми панелі(Рисунок 3.6).

3.3.3 Структура класу BMP/PNG

Клас BMP (Bitmap) представляє функціональність, пов'язану з обробкою зображень у форматі BMP та PNG, пов'язаному з цифровою стеганографією. Він містить кілька методів для кодування та декодування інформації у зображеннях BMP та PNG.

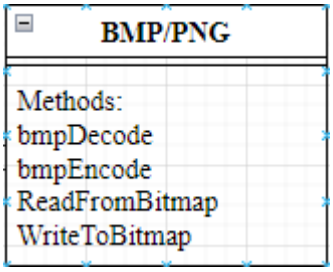

Рисунок 3.7 – Структура класу BMP/PNG

Методи класу BMP включають bmpDecode, bmpEncode, ReadFromBitmap та WriteToBitmap. Метод bmpDecode відповідає за декодування прихованої інформації із зображення BMP та PNG з використанням методу останнього біта. Метод bmpEncode відповідає за кодування інформації та її приховування у зображенні BMP та PNG.

Метод ReadFromBitmap виконує читання даних із зображення BMP, для подальшої обробки чи аналізу. Метод WriteToBitmap виконує запис даних зображення BMP та PNG(Рисунок 3.7).

Клас BMP/PNG, використовується в проекті для роботи із зображеннями у форматі BMP, застосування стеганографічних методів для приховання та вилучення інформації з цих зображень.

3.3.4 Структура класу Wav

Клас WAV являє собою функціональність, пов'язану з обробкою звукових файлів у форматі WAV. Він містить кілька полів та методів для роботи з даними звукових файлів.

Поля класу WAV включають wavFile, який представляє об'єкт або шлях до звукового файлу WAV. Властивості класу WAV включають AudioInfoCount, BitsPerSample, BlockAlignBytes, NumberOfChannels та StartPos. Ці властивості надають інформацію про характеристики звукового файлу WAV, таких як кількість аудіофрагментів, кількість біт на вибірку, розмір блоку в байтах, кількість каналів та початкова позиція у файлі.

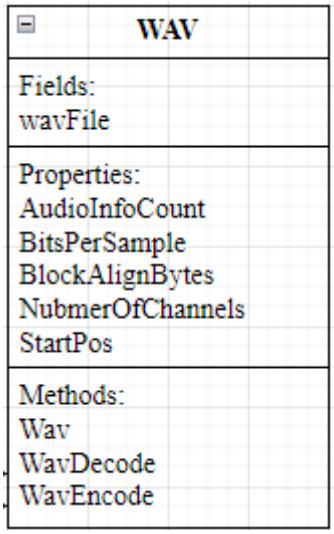

Рисунок 3.8 – Структура класу WAV

Методи класу WAV включають Wav, WavDecode та WavEncode. Метод Wav служить конструктором класу, що ініціалізує об'єкт WAV. Метод WavDecode відповідає за декодування прихованої інформації із звукового файлу WAV. Метод WavEncode відповідає за кодування

Клас WAV використовується в проекті для роботи зі звуковими файлами у форматі WAV, застосування методу останнього біта приховування та вилучення біта приховування та вилучення інформації з цих файлів.

3.3.5 Структура класу JPEG

Клас JPEG являє собою функціональність, пов'язану з обробкою зображень у форматі JPEG. Він містить кілька полів та методів для роботи з даними зображень у форматі JPEG.

Поля класу JPEG включають arrayJpeg, compCount, DHT_AC та DHT_DC. Поле arrayJpeg представляє масив даних, що містять зображення у форматі JPEG. Поле compCount вказує кількість коефіцієнтів зображення. Поля DHT_AC і DHT_DC відносяться до таблиць Хаффмана для кодування значень коефіцієнтів яскравості та кольоровості в JPEG.

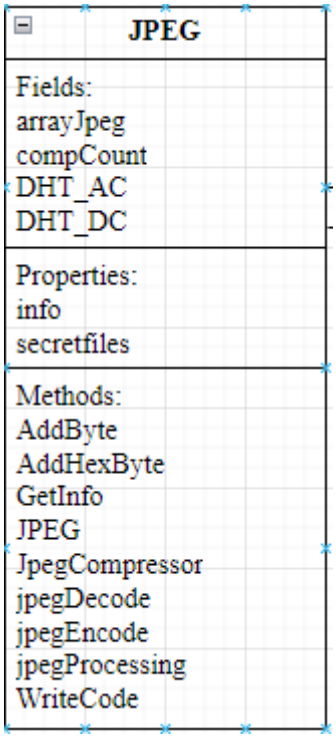

Рисунок 3.9 – Структура класу JPEG

Властивості класу JPEG включають info та secretfiles. Властивість info надає інформацію про JPEG-зображення, таку як розмір. Властивість secretfiles представляє бітовий потік секретних файлів.

Методи класу JPEG включають GetInfo який призначений для отримання інформації про те, скільки може зберігати секретних файлів наше JPEG-зображення. Метод JPEG є конструктором класу, що ініціалізує об'єкт JPEG. Метод JpegCompressor використовується для стиснення зображення та конвертування у базовий режим формату JPEG. Метод jpegDecode відповідає за декодування прихованої інформації із зображення JPEG. Метод jpegEncode відповідає за кодування інформації та її приховування у зображенні JPEG. Метод jpegProcessing виконує обробку зображення JPEG(Рисунок 3.9).

Клас JPEG використовується в проекті для роботи із зображеннями у форматі JPEG, застосування стеганографічних методів для приховання та вилучення інформації з цих зображень.

3.3.6 Структура класу HuffTree

Клас HuffTree представляє функціональність, пов'язану з побудовою та використанням дерев Хаффмана. Він містить властивості Code та Val, а також метод HuffTree. Властивість Code представляє кодове значення дерева Хаффмана. Властивість Val, зберігає значення, що відповідає коду дерева Хаффмана. Метод HuffTree служить для побудови дерева Хаффмана на основі вхідних даних(Рисунок 3.10).

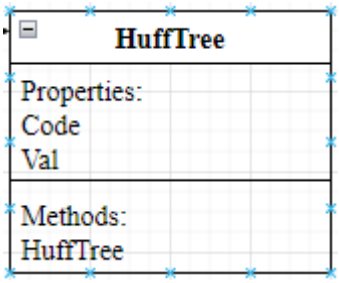

Рисунок 3.10 – Структура класу HuffTree

Клас HuffTree використовується в проекті для реалізації дерев Хаффмана, які можуть бути використані для кодування та декодування секретної інформації у процесі стеганографії JPEG файлів.

3.3.7 Структура класу AES

Клас AES є функціональністю, пов'язаною з алгоритмом шифрування AES (Advanced Encryption Standard). Він містить кілька полів та методів, пов'язаних з процесом шифрування та розшифрування даних.

Поля класу AES включають GMatrix, invGMatrix, invSBox, Rcon, roundKey та SBox. Ці поля містять матриці та таблиці, що використовуються в алгоритмі AES для перетворення даних.

Методи класу AES включають конструктор AES, який виконує ініціалізацію об'єкта класу AES. Методи Decrypt та Encrypt відповідають за розшифрування та

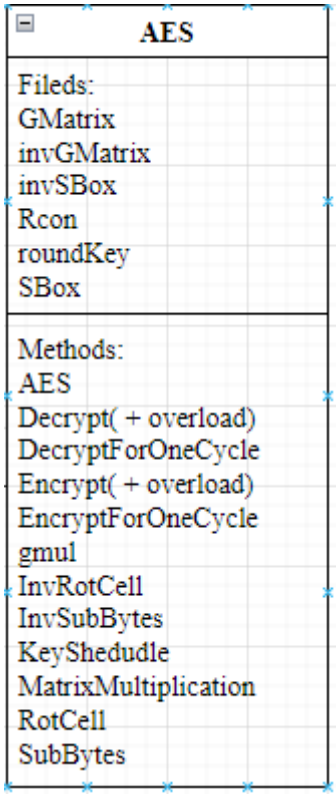

Рисунок 3.11 – Структура класу AES

шифрування даних відповідно, з використанням перевантажень для різних вхідних параметрів. Методи DecryptForOneCycle та EncryptForOneCycle виконують один цикл шифрування або розшифрування. Метод gmul виконує галуа-твір (галуа-множення) для двох чисел. Методи InvRotCell, InvSubBytes, KeySchedule, MatrixMultiplication, RotCell та SubBytes відповідають за різні перетворення, що використовуються в алгоритмі AES(Рисунок 3.11)

Клас AES використовується в проекті для реалізації алгоритму шифрування AES, який є одним з найбільш поширених та надійних методів шифрування, що застосовуються у цифровій стеганографії та інших галузях інформаційної безпеки.

3.4 Огляд програми

Програма розроблена, щоб приховати будь-який тип формату файлів у форматі PNG, BMP, JPEG і WAV. Дані можна зашифрувати за допомогою AES 128. Програма не має обмеження на розмір файлу, але під час роботи з великими

файлами для стегонографії потрібно багато часу. Початковий вигляд програми показано на рисунку 3.12, де:

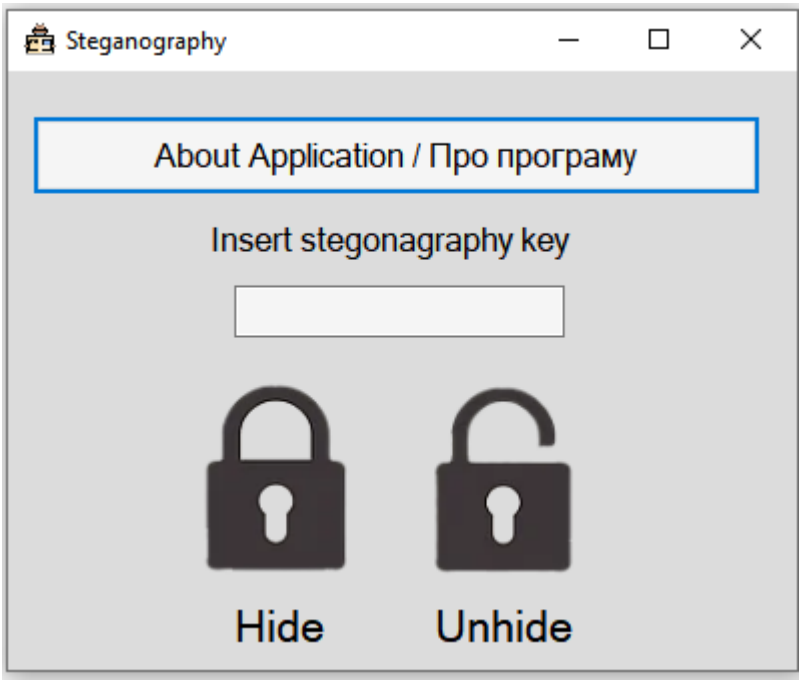

Рисунок 3.12 - Початковий вигляд програми

– Кнопка Hide – при натисканні відкривається нове вікно, в якому виконується кодування;

– Кнопка Unhide – при натисканні відкривається нове вікно, в якому виконується декодування;

– Insert stegonagraphy key – потрібно написати перед початком кодування;

– About Application – після натискання відкривається нове вікно, у якому вказано, як користуватися програмою.

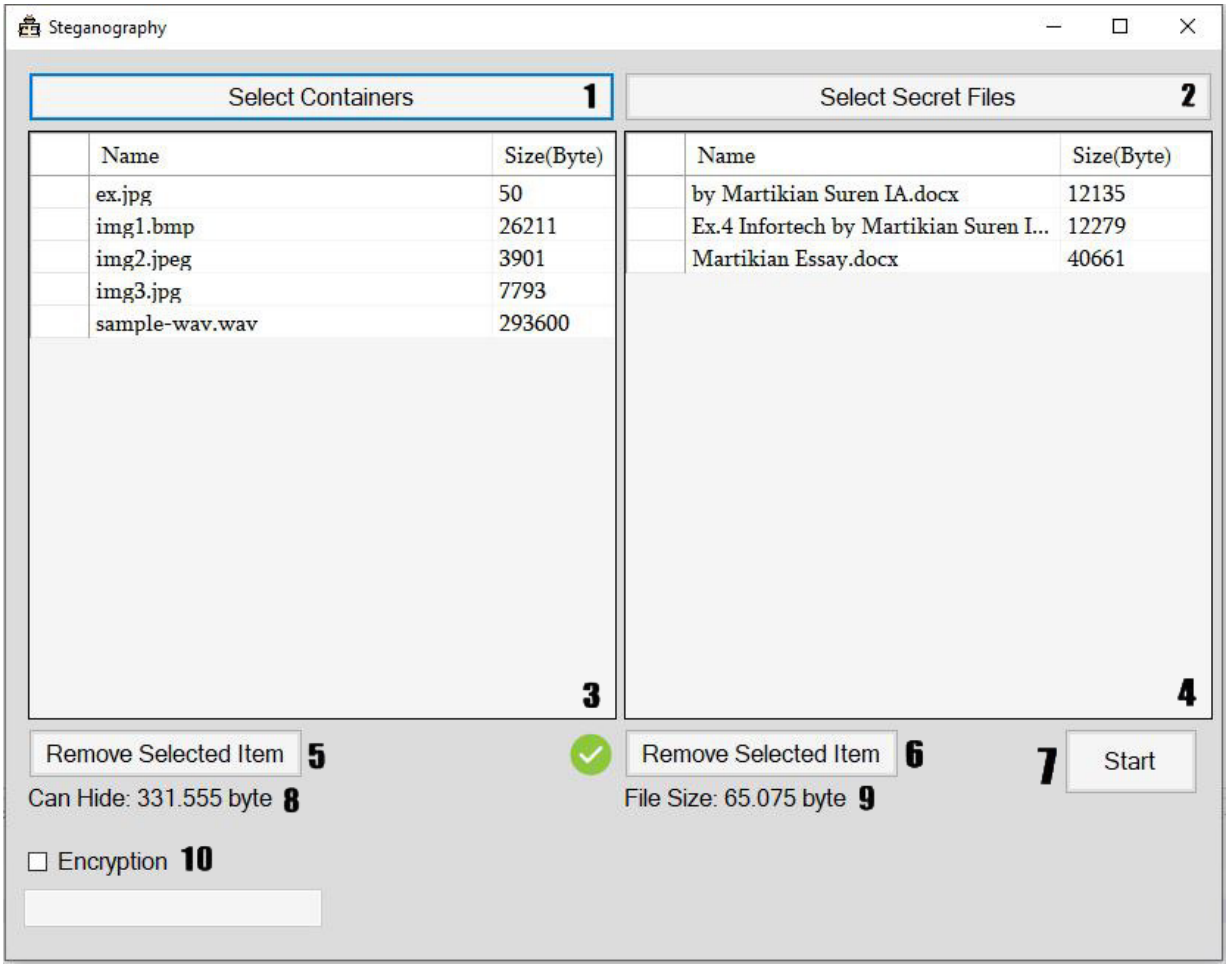

Рисунок 3.13 – Вікно стеганографії

Коли ви натискаєте кнопки «Hide» або «Unhide», відкривається нове вікно, показане на рисунку 3.13.

1. Виберіть контейнери, де виконується стеганографія;

2. Виберіть файли, які ми хочемо приховати;

3. У цій таблиці показано всі вибрані контейнери, які використовуватимуться для стеганографії;

4. У цій таблиці показано файли, які будуть приховані в контейнерах;

5. Ця кнопка видаляє вибрані елементи з таблиці 3;

6. Ця кнопка видаляє вибрані елементи з таблиці 4;

7. При натисканні починається процес стеганографії;

8. Показує, скільки інформації (в байтах) можна приховати;

9. Показує розмір секретної інформації (в байтах);

10. Ми можемо зашифрувати секретні файли завдяки додаткового паролю.

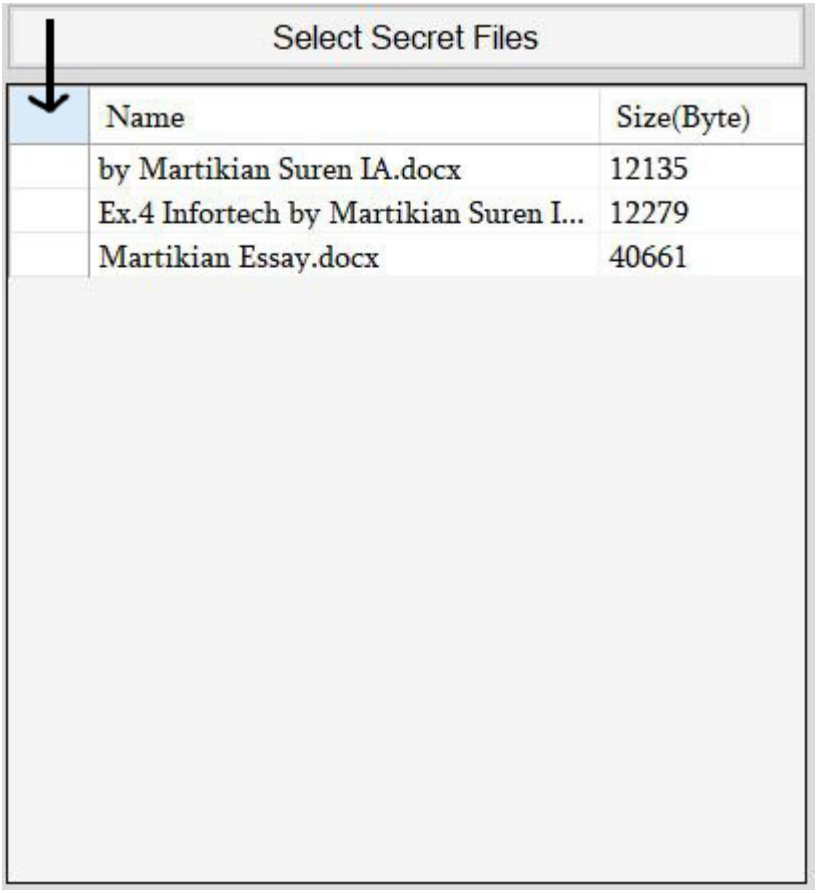

Рисунок 3.14 – Контейнерний стіл

Рисунок 3.14 показує стіл з контейнерами. Щоб вибрати контейнер, вам потрібно вибрати перший стовпець у таблиці, як показано стрілкою на рисунку 3.14. Коли ви натиснете на червону кнопку, усі контейнери будуть вибрані. Кнопка «Видалити вибраний елемент» видаляє всі вибрані контейнери. У таблиці 2-й стовпець показує назву контейнера, а 3-й стовпець показує розмір, який контейнер може приховати.

Висновок до розділу 3

Розділ 3 дипломної роботи була присвячена проектування стеганографії. У цьому розділі було докладно описано середовище, включаючи Visual Studio, мову програмування C# і .NET Framework. Також було розроблено архітектуру проекту та описано структури класів та їх функціональність.

Використання Visual Studio та мови програмування C# дозволило створити функціональний та зручний додаток для стеганографії. .NET Framework надавши широкий набір інструментів та бібліотек, що сприяло ефективній розробці програмного забезпечення.

Архітектура проекту була добре структурована, що дозволило зручно організувати різноманітні функціональні модулі. Розглянуто структури класів, таких як MainForm, StegPanel, BMP/PNG, WAV, JPEG, HuffTree та AES, які забезпечували відповідні функції аналізу стеганографічного та вбудовування даних.

Проектований приклад мав зручний інтерфейс і дозволяв користувачеві ефективно виконувати операції з вбудовування та вилучення прихованої інформації у різних форматах файлів. Всі розроблені модулі та функції були вбудовані в єдину систему, що сприяло зручному використанню та надійній роботі програми.

Висновок з розділу 3 підкреслює успішне проектування та розробку застосування стеганографії. Додаток демонструє готовність та здатність ефективно працювати з різними форматами файлів та виконувати необхідні стеганографічні операції. Його функціональність та логічна структура становлять основу для подальшого розширення та покращення застосування.

ВИСНОВКИ

У цій дипломній роботі було проведено дослідження в галузі стеганографії, що є актуальною проблемою в галузі захисту інформації. Було розглянуто загальні уявлення про стеганографію, включаючи класичну, комп'ютерну та цифрову стеганографію.

У розділі 1 проведено аналіз існуючих стеганографічних систем, таких як ImageSpyer G2, RedJPEG та ImageJS, з'ясовано їх переваги та недоліки. За підсумками цього аналізу було поставлено завдання роботи.

У розділі 2 було розглянуто практичну реалізацію стеганографії у різних форматах файлів. Було вивчено розширений стандарт шифрування (AES), метод останнього біта (LSB) та структуру форматів файлів, таких як PNG, BMP, JPEG і WAV. Були досліджені основні етапи стеганографічного аналізу та вбудовування даних у ці формати файлів.

У розділі 3 було проведено проектування програми для стеганографії. Було описано середовище, що використовується, включаючи Visual Studio, мову програмування C# і .NET Framework. Було розроблено архітектуру проекту, включаючи структури класів та їх функціональність. Також було надано огляд програми.

Загальною метою дипломної роботи було створення функціонального та ефективного інструменту для кодування та декодування конфіденційних даних у контейнерах. За результатами дослідження та розробки було створено додаток, який забезпечує стеганографічну форму в PNG, BMP, JPEG та WAV.

В результаті виконаної дипломної роботи було досягнуто поставлених цілей і вирішено поставлені завдання. Розроблений додаток є ефективним інструментом для вбудовування та вилучення будь-якого формату контейнерів (jpeg, wav, png та bmp). Ця робота може бути використана як основа для подальшого розширення функціональності та поліпшення алгоритмів стеганографії.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Шелухін О.І., Канаєв С.Д. “Стеганографія. Алгоритми та програмна реалізація”

2. Вадим Грибунін - "Цифрова стеганографія короткий зміст"

3. Г. Ф. Конахович, А. Ю. Пузиренко – “Комп'ютерна стеганографія. Теорія та практика"

4. Youssef Bassil - "Steganography: 3 Black Magic до Magic of Science: The Secret Art of Deception"

5. Jessica Fridrich ”Steganography в Digital Media: Principles, Algorithms, and Applications”

6. [Електронний ресурс]. - https://ua.m.wikipedia.org/wiki/Стеганографія#

7. [Електронний ресурс]. - https://tech.wikireading.ru/13221

8. [Електронний ресурс]. - https://docs.microsoft.com/ru-ru/dotnet/csharp/getting-started/introduction-to-the-csharp-language-and-the-net-framework

9. [Електронний ресурс]. - https://knowledge.allbest.ru/programming/3c0a65625b2bc78a4c53a88521216c27_0.html

10. [Електронний ресурс]. - https://ua.wikipedia.org/wiki/Advanced_Encryption_Standard

11. [Електронний ресурс]. - https://ua.bmstu.wiki/BMP_(Bitmap_Picture)

12. [Електронний ресурс]. - https://jenyay.net/Programming/Bmp

13. [Електронний ресурс]. - https://en.wikipedia.org/wiki/Portable_Network_Graphics

14. [Електронний ресурс]. - https://habr.com/ua/post/102521/

15. [Електронний ресурс]. - https://impulseadventure.com/photo/jpeg-huffman-coding.html

16. [Електронний ресурс]. - https://habr.com/ua/post/113611/

17. [Електронний ресурс]. - https://audiocoding.ru/articles/2008-05-22-wav-file-structure/

18. [Електронний ресурс]. - http://enisey.name/umk/pzis/ch18s07.html

19. [Електронний ресурс]. - https://spy-soft.net/cifrovaya-steganografiya-sposoby-realizacii/

ДОДАТОК А

Код програми (репозиторій на GitHub): https://github.com/Suren32/Steganography