Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 20__ р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи кібербезпеки для стеганографічного захисту інформації”

Виконав здобувач вищої освіти
IV курсу, групи КБ-19
ОПП «Кібербезпека»
спеціальності 125 «Кібербезпека»
_____ Пащенко А. В.
«____» _____ 20 р.

Керівник проекту
доктор технічних наук, професор
_____ Мелешко Є. В.
«____» _____ 20 р.
Рецензент _____
_____

м. Кропивницький

Міністерство освіти і науки України
Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*
Кафедра *Кібербезпеки та програмного забезпечення*
Освітній ступінь *бакалавр*
Спеціальність *125 Кібербезпека*

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____О.А.Смірнов
« __ » _____ 20__ року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Пащенку Андрію Віталійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи кібербезпеки для стеганографічного захисту інформації*

керівник роботи *Мелешко Єлизавета Владиславівна, д-р техн. наук, професор*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу №12-02 від 05.01.2023 року

2. Строк подання студентом роботи до захисту *20.05.2023 р.*

3. Мета та завдання кваліфікаційної бакалаврської роботи*: Метою розробки є програмне забезпечення системи кібербезпеки для стеганографічного захисту інформації*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.
2.Перегляд аналогічних існуючих систем.
3. Опис і обґрунтування проектних рішень.
4. Етапи програмування системи.
5. Впровадження системи в промислову експлуатацію.
6. Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Структурна схема системи	*1 аркуш*
Функціональна схема системи	*2 аркуша*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*3 аркуша*

6. Дата видачі завдання « 21 » грудня 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної бакалаврської роботи	Строк виконання етапів кваліфікаційної бакалаврської роботи	Примітка
1.	Аналіз існуючих систем	10.03.2023 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2023 р.	
3.	Розробка моделі компонента	20.03.2023 р.	
4.	Розробка структур даних	25.03.2023 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2023 р.	
6.	Програмування алгоритмів	10.04.2023 р.	
7.	Оформлення ПЗ	17.04.2023 р.	
8.	Попередній захист роботи	22.05.2023 р.	

Студент _____ _____
(підпис) (прізвище та ініціали)

Керівник роботи _____ _____
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Пащенко А.В. Програмне забезпечення системи кібербезпеки для стеганографічного захисту інформації. 125 Кібербезпека. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

У даній кваліфікаційній бакалаврській роботі розроблено програмне забезпечення, яке призначено для системи кібербезпеки для стеганографічного захисту інформації.

Метою роботи є створення системи кібербезпеки для стеганографічного захисту інформації.

Результат роботи – програмна реалізація системи кібербезпеки для стеганографічного захисту інформації.

В процесі роботи над реалізацією системи виконано дослідження існуючих методів, алгоритмів та програмних засобів. Розроблено та реалізовано власне програмне забезпечення, здійснено опис всіх його компонентів.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ архітектури IBM PC з ОС Windows 10/11.

Програму розроблено на мові програмування C#.

Ключові слова: кібербезпека, захист інформації, стеганографія

ABSTRACT

Pashchenko A.V. Cyber security system software for steganographic protection of information. 125 Cybersecurity. Central Ukrainian National Technical University. Kropyvnytskyi 2023

In this bachelor's qualification work, software that is designed for system for steganographic protection of information was developed.

The purpose of the development is the software of the system for steganographic protection of information.

The result of the work is the software implementation of the system for steganographic protection of information.

In the process of working on a software model an analysis of existing hardware and software was performed. All components of the software developed are fully described.

A user-friendly interface is developed. The instructions for working with the software are provided.

The program can be used on an IBM PC running Windows 10/11.

The program was developed in the C# programming language.

Keywords: cybersecurity, information protection, steganography

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

LSB	–	Least Significant Bit.
BMP	–	bitmap-формат операційної системи Windows, який запам’ятовує одно і багатокольорові (RGB) ілюстрації у формі Pixel.
GIF	–	8-бітний растровий графічний формат, що використовує до 256 чітких кольорів із 24-бітного діапазону RGB.
RGB	–	адитивна модель кольорів, що описує спосіб синтезу кольору, за якою червоне, зелене та синє світло накладаються разом, змішуючись у різноманітні кольори.
JPEG	–	растровий формат збереження графічної інформації, що використовує стиснення з втратами.
НЗБ	–	найменш значущий біт.
Стего	–	стегоповідомлення, текст, який вбудовується в графічний файл методами стеганографії.

ВСТУП

Стеганографічний захист інформації у комп'ютерних системах та мережах має велику актуальність у сучасному світі. Зростання кількості та значущості цифрової інформації, що передається, обробляється та зберігається в мережі, створює потребу в збільшенні методів її захисту від різних інформаційних атак та загроз її цілісності та конфіденційності. Стеганографія є одним з методів забезпечення інформаційної безпеки, який дозволяє приховати наявність інформаційного повідомлення, як правило це реалізується за допомогою передачі захищеного повідомлення в іншій інформації, яка відіграє роль контейнера. Що може бути використано для захисту від різних видів кібератак, зокрема, від перехоплення та аналізу трафіку, від атак методом «людина посередині» тощо. Сучасні цілі вжитку стеганографії наступні: забезпечення конфіденційності даних, захист авторських прав на цифрові зображення, перевірка легітимності використання ліцензійного програмного забезпечення тощо.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи кібербезпеки для стеганографічного захисту інформації.

Для досягнення поставленої мети треба вирішити наступні задачі:

– Дослідження існуючих методів стеганографічного захисту інформації.

– Розробка алгоритмів системи стеганографічного захисту інформації.

– Програмна реалізація системи стеганографічного захисту інформації.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно реалізувати стеганографічний захист даних.

Тож, розробка програмного забезпечення системи кібербезпеки для стеганографічного захисту інформації має велике значення для підвищення безпеки обміну та зберігання цифрової інформації, зменшення можливостей незаконного доступу до цієї інформації, а також забезпечення конфіденційності та цілісності даних, та вирішувалася у даній кваліфікаційній роботі.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Застосування стеганографії має велике значення для захисту приватної та авторської закритої інформації у мережі Інтернет від незаконного доступу та використання.

Призначення стеганографічних систем захисту інформації може бути наступним:

– *Захист конфіденційності інформації*: збереження даних або передача повідомлення приховано, без будь-якої підозри від зловмисників, що можуть отримати доступ до носіїв інформації та перехоплювачів повідомлень в Інтернеті.

– *Захист авторських прав на цифровий контент*: за допомогою стеганографії можна вбудовувати в зображення або відео інформацію про авторство або права на використання цього контенту, ця техніка називається вбудовуванням «прихованих цифрових водяних знаків» та дозволяє виявляти незаконне використання цифрового контенту.

– *Захист від зловмисного програмного забезпечення*: приховування ідентифікаційних кодів або інших даних, які використовуються для визначення легітимності та незмінності (відсутності змін, викликаних наявністю комп'ютерних вірусів) програмного забезпечення, що дозволяє запобігти використанню зловмисного програмного забезпечення.

– *Захист від атак типу «людина посередині»*: приховування інформації в такий спосіб, що її може виявити та прочитати тільки особа, яка має потрібний ключ доступу, що забезпечує захист від атак типу «людина посередині».

– *Збереження конфіденційності відкритих ключів*: приховування відкритих ключів від незаконних користувачів, що забезпечує більш високий рівень безпеки системи шифрування.

– *Передача конфіденційної інформації в обмеженому каналі зв'язку*: приховування повідомлення в зображенні, що передається в обмеженому каналі зв'язку, що дозволяє зберегти обмежену пропускну здатність каналу.

Таким чином, існує багато напрямків та способів застосування систем стеганографічного захисту інформації.

1.2 Область застосування

Стеганографічний захист інформації може використовуватися, як організаціями та фірмами, так і окремими користувачами. Застосування стеганографії може бути корисним для будь-якої організації чи фірми, що має важливу та конфіденційну інформацію, прикладами таких організацій можуть бути фінансові установи, державні органи, наукові центри, тощо. Також стеганографічний захист інформації можна використовувати у сфері військової та державної безпеки для забезпечення конфіденційності важливої інформації та захисту від шпигунської діяльності. Окремі користувачі також можуть використовувати стеганографію для захисту власної приватності або своїх авторських прав на цифровий контент.

Отже, стеганографія надає широкі можливості застосування в різних сферах життєдіяльності та може бути використана будь-якою організацією або приватним користувачем, що має потребу в захисті інформації.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми кваліфікаційної бакалаврської роботи

Стеганографія – це спосіб організації прихованого зберігання чи передачі повідомлень, у якому ховається саме наявність інформації, на відміну від криптографії, де противник точно може визначити, чи є захищене повідомлення, але не може його прочитати. Таким чином, якщо мета криптографії полягає у блокуванні несанкціонованого доступу до інформації шляхом шифрування змісту секретних повідомлень, то мета стеганографії – приховати сам факт існування секретного повідомлення. При необхідності обидва способи можуть бути об'єднані та використані для підвищення ефективності захисту інформації.

Комп'ютерна стеганографія базується на двох основних засадах:

1. Файли, що містять оцифроване зображення або звук, можуть бути певною мірою видозмінені без втрати їх функціональності на відміну від інших типів даних, що вимагають абсолютної точності.

2. Нездатність органів чуття людини розрізняти незначні зміни в кольорі зображення або якості звуку. Цей принцип особливо легко застосовувати до зображення або звуку, що несе надмірну інформацію.

Згодом ці принципи були поширені й на інші типи інформації. Наприклад, у текстову інформацію можна додавати зайві пробіли, і це буде непомітно людині, але нести приховане повідомлення.

Стеганографічна система, або стегосистема – це сукупність засобів та методів, які використовуються для формування прихованого каналу передачі інформації.

При побудові стегосистеми повинні враховуватися такі положення:

– методи приховування повинні забезпечувати автентичність та цілісність інформації, в якій ховається повідомлення;

– супротивник має повне уявлення про стеганографічну систему і деталі її реалізації. Єдиною інформацією, невідомою потенційному противнику, є ключ, за допомогою якого тільки його власник може встановити факт присутності та змісту прихованого повідомлення;

– якщо противник якимось чином дізнається про факт існування прихованого повідомлення, це не повинно дозволити йому витягти приховану інформацію з інших даних до тих пір, поки ключ зберігається в таємниці;

– потенційний противник повинен бути позбавлений будь-яких технічних та інших переваг у розпізнаванні або розкритті змісту прихованих повідомлень.

Вбудоване (приховане) повідомлення – це повідомлення, вбудоване у стеганографічний контейнер.

Стеганографічний контейнер – це будь-яка інформація, призначена для приховування повідомлення, наприклад, файл зображення, аудіофайл, текстовий файл або цифровий сигнал. Вибір виду контейнера істотно впливає на надійність стегосистеми і можливість виявлення факту передачі прихованого повідомлення. За розміром (протяжністю) контейнери можна розділити на два типи: безперервні (струмові) та обмеженої (фіксованої) довжини.

Особливістю потокового контейнера (це найчастіше цифровий трафік чи сигнал) є те, що неможливо визначити його початок та кінець. У такому контейнері біти інформації, що використовуються для приховування повідомлення, включаються до загального потоку в реальному масштабі часу і вибираються за допомогою спеціального генератора, що задає відстані між ними. У безперервному потоці даних найбільша складність для одержувача – визначити, коли починається приховане повідомлення. За наявності потокового контейнера сигналів синхронізації або меж пакета, приховане повідомлення починається відразу після одного з них. У свою чергу для відправника

повідомлення можливі проблеми, якщо він не впевнений, що потік контейнера буде досить довгим для розміщення всього повідомлення.

При використанні контейнера обмеженої довжини (найчастіше файлу) відправник заздалегідь знає розмір файлу і може вибрати біти, що приховують, у відповідній псевдовипадковій послідовності. З іншого боку, такі контейнери мають обмежений об'єм, і повідомлення, що вбудовується, іноді може не поміститися у файл-контейнер. Інший недолік полягає в тому, що відстані між бітами, що приховують, рівномірно розподілені між найбільш короткими і найбільш довгими заданими відстанями, в той час як істинний випадковий шум буде мати експоненційний розподіл довжин інтервалу. При необхідності можна задати псевдовипадкові експоненційно розподілені числа, проте цей шлях є найбільш трудомістким. На практиці найчастіше використовуються контейнери обмеженої довжини як найбільш поширені та доступні.

Можливі наступні варіанти контейнерів:

– контейнер генерується самою стегосистемою. Такий підхід називається конструюючою стеганографією;

– контейнер вибирається з деякої множини контейнерів, що генеруються стегосистемою. Такий підхід називається селектуючою стеганографією;

– контейнер надходить ззовні стегосистеми. Такий підхід називається безальтернативною стеганографією.

Залежно від виду інформації, яка використовується для вбудовування повідомлень, контейнери можуть бути візуальними, звуковими та текстовими.

Візуальний контейнер є картинкою або фотографією, в якій для вбудовування повідомлень використовуються невеликі зміни яскравості заздалегідь визначених точок растра зображення.

Звуковий контейнер є мовленнєвим або музичним сигналом, в якому для вбудовування повідомлень використовуються молодші біти аудіосигналу, що практично не відбивається на якості звуку.

Текстовий контейнер є текстовим файлом, підготовленим до друку на принтері, в якому для вбудовування повідомлень використовуються невеликі зміни стандартів друку (відстань між літерами, словами та рядками, розміри літер, рядків та ін.).

При виборі того чи іншого виду контейнера необхідно мати на увазі, що при збільшенні обсягу повідомлення, що вбудовується, знижується надійність стегосистеми (при незмінному розмірі контейнера). Таким чином, контейнер, що використовується в стегосистемі, накладає обмеження на розмір вбудованого повідомлення.

Порожній контейнер – це контейнер без вбудованого повідомлення.

Заповнений контейнер або стегоконтейнер – це контейнер, що містить вбудовану інформацію.

Стеганографічний канал (стегоканал) – це канал передачі прихованого повідомлення.

Ключ (стегоключ) – це секретний ключ, необхідний приховування повідомлення. Залежно від кількості рівнів захисту у стегосистемі може бути один або кілька стегоключів.

За аналогією з криптографією стегоключі стегосистеми можуть поділятися на два види: секретний ключ і відкритий ключ.

У стегосистемі із секретним ключем використовується один ключ, який має бути визначений або до початку обміну секретними повідомленнями, або передано захищеним каналом.

У стегосистемі з відкритим ключем для вбудовування та отримання повідомлення використовуються різні ключі, відмінність яких полягає в тому, що за допомогою обчислень неможливо визначити один ключ з іншого. Тому один ключ (відкритий) може передаватися вільно незахищеним каналом зв’язку.

Будь-яка стегосистема повинна відповідати таким вимогам:

– властивості контейнера повинні бути модифіковані таким чином, щоб зміну неможливо було виявити під час візуального контролю;

– стегоповідомлення має бути стійким до спотворень, які можуть мати місце при його передачі, включаючи різні трансформації (зменшення, збільшення, перетворення в інший формат, стиснення без втрати інформації, стиснення з втратою інформації тощо);

– для збереження цілісності повідомлення, що вбудовується, необхідно використовувати коди з виправленням помилок;

– для підвищення надійності вбудовуване повідомлення має бути продубльовано.

Дослідження сучасних методів комп'ютерної стеганографії

Методи комп'ютерної стеганографії можна розділити загалом на два види: методи, засновані на надмірності візуальної та аудіоінформації; методи, що ґрунтуються на використанні спеціальних властивостей комп'ютерних форматів.

Методи, що ґрунтуються на надмірності візуальної та аудіоінформації, для приховання інформації використовують молодші розряди цифрових відрахувань цифрового зображення та звуку, які містять дуже мало корисної інформації. Їхнє заповнення додатковою інформацією практично не впливає на якість сприйняття, що і дає можливість приховувати конфіденційну інформацію.

Перевагою цих методів є можливість прихованої передачі великого обсягу інформації та можливість захисту авторського права шляхом створення прихованого зображення товарної марки, реєстраційного номера тощо.

Недолік методу у тому, що рахунок введення додаткової інформації спотворюються статистичні характеристики цифрових потоків. Для зниження компрометуючих ознак потрібна корекція статистичних показників.

Методи, що ґрунтуються на використанні спеціальних властивостей комп'ютерних форматів, поділяються на: методи використання зарезервованих для розширення полів комп'ютерних форматів даних; методи спеціального форматування текстових файлів; методи приховування в місцях, що не використовуються, гнучких дисків; методи використання функцій, що імітують; методи видалення файлу заголовка, що ідентифікує.

Методи використання зарезервованих для розширення полів комп'ютерних форматів даних засновані на тому, що багато мультимедійних форматів мають поля розширення, які заповнюються нульовою інформацією та не враховуються програмою. У ці поля і записується інформація, що приховується.

Методи спеціального форматування текстових файлів у свою чергу поділяються на: методи використання відомого усунення рядків, слів, речень, абзаців; методи вибору певних позицій букв; методи використання спеціальних властивостей, які не відображаються на екрані полів форматів.

Розглянемо найбільш поширені стеганографічні методи.

Методи заміни молодших бітів. Одним із найпоширеніших методів стеганографії, що використовують психофізичні особливості людини, є метод заміни молодших біт інформації, або ***LSB-метод*** (Least Signifi-cant Bits). Поширеність цього методу обумовлена функціональною простотою, великою ємністю та високим ступенем захищеності від стегоаналізу.

Суть методу полягає у заміні кількох молодших біт у байтах даних. Він застосовується у графічних файлах, що використовують для формування кольору кожного елемента зображення (піксела) значення деяких складових (наприклад, значення складових основних кольорів – червоного, зеленого та синього), або у звукових файлах, що використовують для формування звуку значення дискретизованих амплітуд сигналу.

При оцифруванні зображення або звуку завжди існує похибка дискретизації, яка зазвичай знаходиться на рівні молодшого біта. Це означає, що фактично невідомо, що стоятиме в молодшому розряді цифрового представлення кольору або звуку. Тому при заміні тільки наймолодшого біта говорити про якесь спотворення зображення або звуку не має сенсу. Однак при заміні тільки одного молодшого біта такий метод має досить малу ємність, близько 10% обсягу файла-контейнера, тому на практиці використовують заміну більше одного біта.

Метод заміни палітри кольорів. Метод заснований на використанні специфічних особливостей формату файлу-контейнера і призначений для

приховання текстової інформації у графічних файлах, які використовують палітри кольорів.

Такими файлами є, наприклад, файли BMP, PCX та GIF.

Палітра є деяким числом тріад байт (не більше 256), які описують колір точки за тим же принципом, що і в файлах True Color. За палітрою слідує масив байт, кожен з яких описує одну точку зображення і містить номер кольору в палітрі.

При використанні цього методу як контейнер слід вибирати файли, що містять деякий колір у надлишку. Наприклад, це можуть бути схеми, малюнки, чорний текст на білому тлі.

Таке поєднання кольорів є оптимальним для реалізації даного методу, проте можна використовувати інші кольори.

Метод сортування палітри кольорів. Метод ґрунтується як на використанні особливостей формату контейнера, так і на використанні психофізичних особливостей сприйняття кольору людиною. При цьому методі як контейнер використовуються файли з індексованими кольорами, що містять монохромне (зазвичай градації сірого) зображення. Суть методу полягає у спеціальній попередній підготовці файлу-контейнера.

Палітра файлу впорядковується таким чином, щоб кольори із сусідніми номерами мінімально відрізнялися один від одного та рівномірно змінювалися від чорного кольору для нульового номера до білого для 255-го номера. Після цього інформація заноситься в молодші розряди точок зображення. Тобто спотворюються не самі кольори, а номери кольорів, але завдяки заздалегідь відсортованій палітрі колір точки замінюється на схожий, який практично неможливо відрізнити від вихідного через їхню малу відмінність.

Інформація, прихована цим методом, легко виявляється засобами програмного аналізу.

Описані вище методи не збільшують розмір файлу-контейнера, але їх застосування обмежене растровими форматами, що використовують стиснення інформації без втрати якості, наприклад, RLE- або LZW-стиск.

Методи комп'ютерної стеганографії у JPEG-файлах. Кольорові зображення, представлені у цифровій формі, досить великі і займають великий обсяг пам'яті (до кількох мегабайт). Тому для їх стиснення існує низка методів. Так, формати BMP та GIF використовують алгоритми стиснення без втрат, що забезпечують точне відновлення вихідного зображення. Існують також алгоритми стиснення зі втратою (спотворенням) інформації.

Таким прикладом може бути формат JPEG (Joint Photographic Experts Group).

У випадку методи обробки (стиснення) зображень можна розділити на дві групи: безпосередні і спектральні. При використанні безпосередніх методів обробки піддаються самі вихідні зображення (пікселі). Спектральні методи ґрунтуються на застосуванні дискретних унітарних перетворень Фур'є, Адамара та ін. При цьому обробляється не вихідне зображення, а відповідні коефіцієнти перетворення.

Методи приховування інформації в JPEG-файлах мають досить високий ступінь захищеності від стегоаналізу, оскільки можливість варіювання якості стисненого зображення в широкому діапазоні не дозволяє легко встановити, чи виникають в результаті стиснення похибки наслідком приховування даних або наслідком використання високих коефіцієнтів квантування.

Комп'ютерна стеганографія у PRN-файлах. Файли друку кольорових графічних зображень на принтерах, які підтримують точковий висновок, містять опис бітової картки відбитка.

Процедура друку передбачає отримання бітових карток відбитків, кодування їх на мові керування принтером (PRN-файли) та пересилання на принтер для безпосереднього отримання відбитка.

В цьому випадку як контейнер стегосистеми може бути використана бітова карта відбитка.

Структурну схему стегосистеми для приховування інформації в бітових картах наведено на рис. 2.1.

До отримання відбитка бітова карта модифікується таким чином, щоб не дозволити виявити наявність вбудованого повідомлення візуальним контролем самої карти та відповідного відбитка. Для приховування інформації в цьому випадку застосовується один із методів комп'ютерної стеганографії, заснований на надмірності даних. У бітових картах надмірність даних створюється рахунок великої варіантності взаємного розташування кольорових плям всередині однієї й тієї ж одиничного фрагмента відбитка зображення.

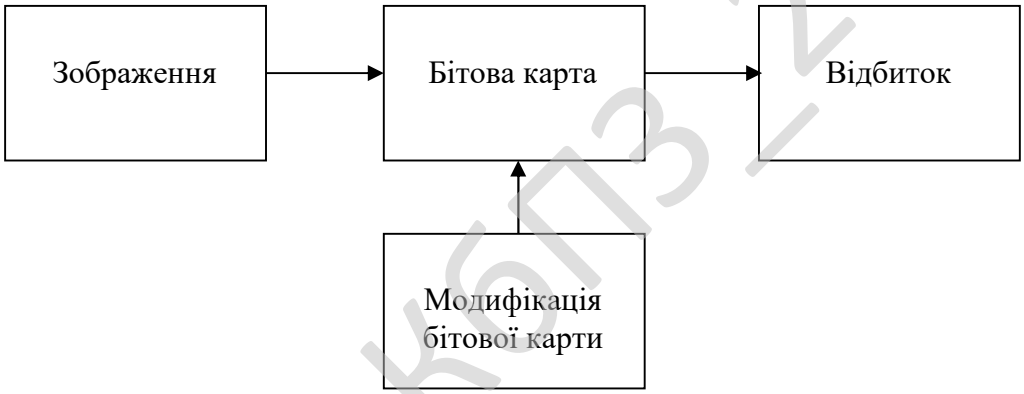

Рисунок 2.1 – Структурна схема стегосистеми для прихования інформації у бітових картах

Кількість бітових карт дорівнює кількості первинних барвників принтера. Кожна бітова карта, що відповідає одному первинному барвнику принтера, може бути представлена прямокутною матрицею з нульовими та одиничними компонентами, яка має 1 відповідає наявності барвника у відповідній позиції, а 0 – його відсутності. Сукупність бітових матриць з відповідними значеннями 0 та 1 і відповідає конкретному відбитку. Градації кольору на відбитку відтворюються за допомогою підмножин (поодиноких фрагментів) бітової карти - растрових точок з різною щільністю барвистих плям. Для

збереження правильного градаційного відтворення під час друку і, отже, для приховання факту вбудовування конфіденційної інформації, модифікація бітових карток відбитків не повинна призводити до зміни щільності барвистих плям (дотів) усередині растрових точок.

Серед усіх можливих візерунків, що відповідають деякій щільності заповнення растрової точки барвистими плямами, для цілей стеганографії можна користуватися регуляризованими, не створюють муар растрами, тобто такими, доти яких розподілені досить рівномірно по площі растрової точки.

Слід зазначити, що розміри растрової точки жорстко не визначені і можуть змінюватись у розумних межах. Наприклад, при роздільній здатності 300 dpi (dot per inch) можна використовувати квадратну матрицю розмірами 6x6, 7x7, 8x8 та 9x9. Форма растрової точки (прямокутна, кругла, овальна і т.д.) також може змінюватись при вбудовуванні інформації.

Отримати відбитки з прихованою інформацією можна лише за допомогою спеціального програмного забезпечення, оскільки при друку драйвер принтера, використовуючи фірмову технологію растрування, «розмиє» ті символи прихованої інформації, які за своїми розмірами можна порівняти з розмірами растрової точки. Для отримання прихованої інформації необхідно використовувати програму та алфавітні таблиці.

Методи комп'ютерної стеганографії, що використовують особливості форматів файлів-контейнерів, неможливо виявити шляхом суб'єктивного аналізу (переглядом, прослуховуванням), але досить легко виявити, використовуючи різні програмні засоби стегоаналізу. У той самий час методи, засновані на використанні психо-фізичних особливостей людини, неможливо виявити простим програмним аналізом щодо відповідності формату і досить складно, а деяких випадках неможливо, виявити шляхом суб'єктивного аналізу.

Дослідження існуючого програмного забезпечення для стеганографії

ImageSpyer G2 – утиліта для приховування інформації у графічних файлах із використанням криптографії. При цьому підтримується близько 30 алгоритмів

шифрування та 25 хеш-функцій для шифрування контейнера. Приховує обсяг інформації, що дорівнює числу пікселів зображення. Опціонально доступна компресія даних, що приховуються. Як вихідні графічні файли можуть використовуватися формати BMP, JPEG, WMF, EMF, TIFF. ImageSpyer ховає секретні файли тільки в картинки і навіть дозволяє ставити пароль, щоб зашифрувати дані перед використанням. Великий плюс – наявність графічного інтерфейсу зі зрозумілими поясненнями та безліччю налаштувань. Вихідних форматів лише два: BMP і TIFF.

RedJPEG – 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. Професійна розширена версія RedJPEG XT доповнена маскуванням факту впровадження та посиленою процедурою ініціалізації потокового шифру на основі характеристик зображення. Включено x86 та x86-64 зборки.

DarkCryptTC та Проект «Зоря» – цю програму можна назвати найбільш потужним стеганографічним рішенням. Вона підтримує більше сотні різних симетричних та асиметричних криптоалгоритмів. Включає підтримку власної системи плагінів, призначеної для блокових шифрів (BlockAPI), текстову, аудіо та графічну стеганографію (включаючи реальну стеганографію JPEG), потужний генератор паролів та систему знищення інформації та ключів. Список форматів, що підтримуються, справді вражає: *.txt, *.html, *.xml, *.docx, *. odt, *.bmp, *jpg, *.tiff, *.png, *.jp2, *.psd, tga, *.mng, *.wav, *.exe, *.dll. Набір програм для стеганографії не дуже великий, але цілком достатній для того, щоб ефективно приховувати інформацію у файлах різних форматів.

OpenStego – проект, реалізований на Java, має підтримку шифрування AES і дуже популярний серед бажаючих познайомитися зі стеганографією. Підтримує плагіни, щоб ти сам зміг реалізувати якийсь стеганографічний алгоритм. Є версії і

Windows, і Linux. Як і ImageSpyer, OpenStego значно роздмухує розміри файлу, хоч і не настільки сильно. Підтримується також один спосіб упаковки, але це легко поправити плагінами. Вихідні файли можуть бути тільки у форматі PNG, але це не можна назвати дуже великим мінусом, тим більше що на вхід можна подавати майже будь-який формат. Є й цікава функція, що дозволяє потай помітити фотографію «невидимими цифровими знаками», щоб легко знайти злодія у випадку незаконного поширення файлу. Для цього програма впроваджує у картинку непомітний ідентифікатор користувача, який надалі можна буде дістати та перевірити, хто поширив картинку без дозволу. Програма не потребує встановлення.

SilentEye – крос-платформний софт з простим інтерфейсом. Має безліч плагінів і приємним GUI. Використовує сучасні алгоритми стеганографії та маскування. З очевидних переваг є те, що введення тексту прихованого повідомлення відбувається прямо у вікні програми замість завантаження текстових файлів зі стороннього редактора. Підтримуються формати вихідних файлів картинок – BMP, JPEG, PNG, GIF, TIFF, звуку – лише WAV. Можна налаштувати якість вихідного зображення – воно визначає, скільки втрат буде при кодуванні в JPEG. Для шифрування впроваджених даних застосовується AES, але налаштувань набагато більше, ніж у OpenStego.

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. Програма підтримує впровадження у формати BMP, GIF, WEBP, PNG та PDF.

StegoTC G2 TC – стеганографічний архіваторний плагін (wcx) для Total Comander дозволяє приховувати дані в будь-якому зображенні, при цьому підтримуються формати BMP, TIFF та PNG.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Розробка програмного забезпечення для реалізації стеганографічного захисту інформації була реалізована мовою програмування Microsoft Visual C#, що є інтегрованим середовищем створення додатків, яке було розроблене компанією Microsoft і є частиною пакету Microsoft Visual Studio.

Visual C# включає в себе компілятор і середовище розробки, компоненти якого при взаємодії покращують процес розробки додатків.

Компілятор мови Visual C# має чимало нових інструментальних засобів і покращених можливостей для створення додатків під Windows. Такі додатки відзначаються простотою у використанні, хоча процес їх створення досить не простий. Тому, для полегшення роботи програмістів, розроблена бібліотека Microsoft Foundation Classes, що істотно полегшує програмування під Windows. Ще одна перевага MFC – можливість допрацьовувати класи чи створювати власні. Користуючись класами C#, можна швидко і легко вирішувати більшість задач програмування. Ці класи використовуються для управління об'єктами Windows, для вирішення конкретних і загальних задач. В MFC є класи керування файлами, часом, рядами, обробкою виключень, є засоби оброблювання повідомлень та діагностування помилок, що суттєво спрощує розробку і створення додатків під Windows. Отже в MFC представлена більшість функцій WindowsAPI.

Вибір мови програмування Visual C# пояснюється досить великою кількістю можливостей, що було описано вище, розгорнутою роботою з класами, потужною роботою з бібліотеками, необхідними для розробки даного програмного забезпечення по темі даної магістерської роботи, в особливості з бібліотеками читання/запису із/у файл. Підключений модуль crc забезпечує дві реалізації CRC (циклічний надлишковий код) об'єктів обчислення і дві реалізації функцій обчислення CRC. Реалізації засновані на шаблоні. Перша об'єктна реалізація необхідна для теоретичного використання. Може оброблювати окремі

біти, але вважається повільним для практичного застосування. Друга об'єктна реалізація – байтова і використовує таблиці пошуку для швидкої роботи. Оптимізована реалізація повинна підійти для загального використання. Перша функціональна реалізація використовує оптимізований об'єкт. Друга функціональна реалізація дозволяє використання CRC, який безпосередньо слідує за його даними.

Microsoft Visual Studio – необходний засіб для розробників, який дозволяє вирішувати різноманітні задачі розробки. Система спрощує створення, налагодження та розгортання додатків на різноманітних платформах, включаючи SharePoint та хмарне середовище. Visual Studio включає вбудовану підтримку моделі «розробка через тестування», а також інструментів налагодження, які забезпечують створення високоякісних рішень.

Написання програмного коду часто потребує одночасної роботи з декількома конструкторами і редакторами. Visual Studio допомагає розробнику організувати цифрове оточення завдяки підтримці декількох моніторів, що спрощує роботу над програмами.

Новий компонент підтримки розробки додатків SharePoint дозволяє створювати спеціалізовані засоби спільної роботи, включаючи веб-модулі, списки, робочі процеси, події та багато іншого.

Visual Studio включає вбудовані інструменти розробки для Windows, в тому числі, такі компоненти інтерфейсу користувача, як мультисенсорний ввод та строка, які створюють основу провідної технології Windows.

Нові функції прив’язки даних пересуванням (в Windows Presentation Foundation) та конструктори Silverlight спрощують і прискорюють побудову додатків Windows та багатофункціональних інтернет-додатків (Rich Internet Applications, RIA) для спеціалістів з проектування та розробки.

Спрощення розгортання веб-додатків. Переміщення веб-додатків у виробниче середовище за один клік миші. Visual Studio виконує перенесення коду, параметрів IIS та схеми бази даних на цільовий сервер.

Інтегроване середовище розробки. З Visual Studio все знаходиться під контролем програміста. Можна користуватися функціями, які налаштовуються, наприклад, такими як можливість використання декількох моніторів, для організації і управління робочим процесом так, як це необхідно саме для програміста. Можна повністю розкрити свій творчий потенціал з допомогою візуальних конструкторів, які призначені для сучасних платформ Windows.

Підтримка платформи розробки. Незалежно від складності проекту Visual Studio можна використовувати для реалізації ідеї та рішення для широкого спектру платформ, включаючи Windows, Windows Server, веб-середовище, хмарне середовище, Office та SharePoint та багато іншого.

Засоби тестування. Visual Studio оснащена усіма вдосконаленими засобами тестування для забезпечення стабільної якості коду. Можна скористатися кодованими тестами інтерфейсу користувача, за допомогою яких можна автоматизувати тестування інтерфейсу веб-сайтів та додатків Windows, засобами ручного тестування, функцією Test Professional, тестуванням продуктивності веб-додатків, тестуванням навантаження та іншими корисними функціями. Visual Studio включає функції модульного тестування в інтегрованому середовищі розробки, які можуть створювати будь-які заглушки методів, які необхідні для компіляції модульних тестів, які дозволяють гарантувати правильне виконання усіх модулів коду.

Управління життєвим циклом додатків (ALM). Під створенням успішних додатків мається на увазі чіткий та безперервний процес, зручний для всіх учасників робочої групи. Вбудовані в Visual Studio засоби ALM допомогають компаніям організувати ефективну спільну роботу і систему комунікації на будь-якому рівні, зробити видимим фактичний стан проекту, забезпечивши доставку високоякісних рішень з меншими витратами.

Налагодження та діагностика. У Visual Studio представлена IntelliTrace, корисна функція налагодження, яка дозволяє залишити усі проблеми з відтворюваністю помилок у минулому. Тестери можуть детально і ефективно

описувати помилки і розробники завжди зможуть відтворити їх у тому стані, у якому їх знайшли. До інших можливостей відносяться статичний аналіз коду, метрика коду і профілювання.

Проектування і моделювання. Architecture Explorer у Visual Studio допоможе розпізнати активи існуючого коду і їх взаємозалежності. Структурні діаграми забезпечують архітектурну узгодженість і дозволяють оцінювати код. Крім того, Visual Studio підтримує п’ять основних видів UML-діаграм, які використовуються разом із кодом.

Розробка баз даних. Розробка баз даних потребує тої самої ретельності і уваги, що й розробка додатків. У Visual Studio це враховується: користувачам надаються надійні засоби розробки і управління змінами, які дозволяють синхронізувати додаток і базу даних.

Team Foundation Server. Team Foundation Server (TFS) — це платформа спільної роботи на основі рішення Майкрософт з управління життєвим циклом додатків. TFS автоматизує і спрощує розробку програмного забезпечення, а також, забезпечує можливість відстежувати всі елементи проекту і бачити у реальному часі його стан для всіх учасників проекту. Платформа також надає потужні засоби панелі моніторингу і створення звітів.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на кваліфікаційну бакалаврську роботу, реалізації підлягає програмне забезпечення, яке призначено для системи кібербезпеки для стеганографічного захисту інформації.

В процесі розробки бакалаврської роботи необхідно виконати наступні за:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей, результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи, розробити функціональну та структурну схеми системи;

в) розробити алгоритми для реалізації програмного забезпечення стосовно теми роботи, побудувати блок-схеми алгоритмів програми та підпрограм;

г) розробити програмне забезпечення системи, що дозволить реалізувати задачу, поставлену технічним завданням задачу;

д) організувати інтерфейс користувача та обробку виключних ситуацій;

е) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

ж) сформувати висновки про виконаний обсяг робіт та одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Інформація є одним з найцінніших ресурсів суспільства поруч з традиційними матеріальними видами ресурсів, як нафта, метал, корисні копалини тощо, тому, процес переробки інформації, подібно до процесів переробки матеріальних ресурсів можна сприймати як технологію. Інформаційна технологія передбачає вміння правильно працювати з інформацією і обчислювальною технікою.

Інформаційна технологія – процес, що використовує сукупність засобів і методів збору, отримання, накопичення, зберігання, обробки, аналізу і передачі даних (інформації) в організаційній структурі з використанням засобів обчислювальної техніки.

Інформаційні процеси – це процеси, які пов'язані з отриманням, зберіганням, обробкою та передачею інформації, в ході яких змінюється зміст інформації або форма її подання.

Передача інформації – це фізичний процес, при якому відбувається переміщення певної інформації в просторі. Цей процес складається з наступних компонентів:

- джерела інформації;
- приймача інформації;
- носія інформації;
- середовища передачі.

Зараз поширеним став спосіб передачі інформації за допомогою мережі Інтернет. Процес передачі інформації можна розглянути узагальнено, адже яким би не був спосіб інформація завжди передається у вигляді певних повідомлень від деякого джерела інформації до її приймача за допомогою каналу зв’язку між ними. Джерело посилає передане повідомлення, яке кодується в певний сигнал.

Цей сигнал переміщується каналом зв'язку. В результаті в приймачі з'являється сигнал, який декодується і стає прийнятим повідомленням. Варто також зазначити, що для більш точної та економної передачі інформації, її необхідно відповідним чином закодувати.

Інформація не може існувати самостійно без матеріального носія та без передачі енергії. Закодоване повідомлення переходить до статусу сигналів, які виступають носіями інформації, і ці сигнали відправляються через комунікаційні канали. При досягненні приймача, ці сигнали мають бути повторно перетворені в легкорозумілий формат за допомогою пристрою для декодування.

Сукупність всіх цих пристроїв, предметів, об'єктів, які призначені для передачі інформації, називається **каналом інформації**, або **інформаційним каналом** (рисунок 3.1).

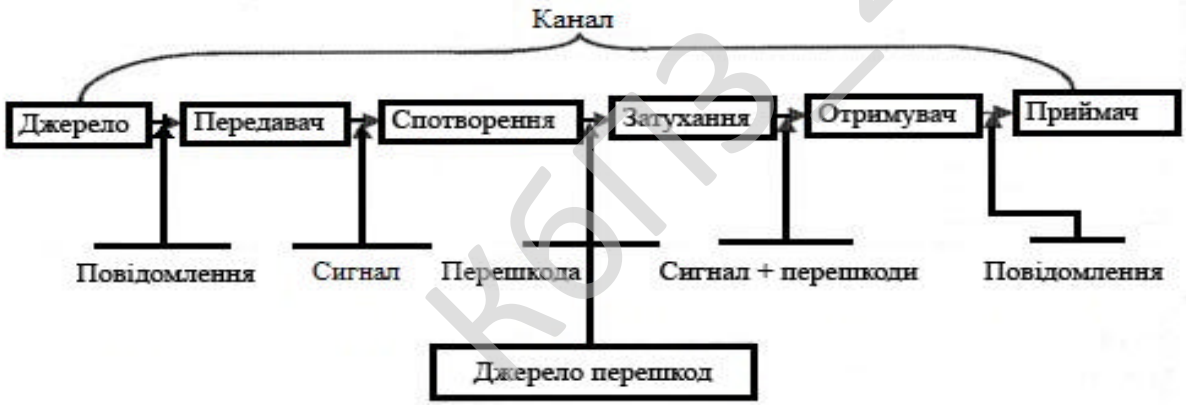

Рисунок 3.1 – Схема каналу передачі інформації

Стеганографія – це наука, що стосується прихованої передачі інформації, з метою приховування самого процесу передачі. Основна мета стеганографії полягає в тому, щоб переконати людину, що в переданій інформації, яка зовні не має жодної цінності, фактично міститься цінна прихована інформація. Таким чином, стеганографія дозволяє передавати конфіденційні дані через відкриті канали, при цьому приховуючи факт її передачі. Криптографія зосереджується на захисті повідомлень, роблячи їх непридатними для використання в разі перехоплення, тоді як стеганографія намагається приховати сам акт передачі повідомлення. Криптографія і стеганографія можуть бути використані спільно:

спочатку повідомлення шифрується, а потім передається приховано. Використання тільки криптографії залишає ризик того, що спостерігач, який перехопив повідомлення, може змусити відправника або отримувача його розшифрувати.

Часто для приховування інформації використовуються графічні файли. Існує велика кількість методів приховування різного рівня складності, але одним з найпростіших є метод найменшого значущого біта (НЗБ, LSB – Least Significant Bit).

Будь-яке зображення являє собою сукупність пікселів. Молодший значущий біт зображення несе в собі менше всього інформації. Відомо, що людина в більшості випадків не здатен помітити змін у цьому біті. Фактично, НЗБ – це шум, тому його можна використовувати для вбудовування інформації шляхом заміни менш значущих бітів пікселів зображення бітами повідомлення, що вбудовується. При цьому, для зображення у градаціях сірого (кожен піксель зображення кодується одним байтом) обсяг вбудованих даних може становити 1/8 від загального обсягу контейнера. Наприклад, в зображення розміром 512x512 можна вбудувати приблизно 32 Кбайт інформації. Якщо ж модифікувати два молодших біта (що також практично непомітно), то дану пропускну здатність можна збільшити ще вдвічі.

Для вбудовування використовується інформація про колір кожного пікселя зображення. Кожна точка кольорового зображення кодується комбінацією з 3-х байт, що задають рівень червоного (R), зеленого (G) і блакитного кольорів (B) – RGB. Кожній з них відповідає своє значення інтенсивності, яке може змінюватися від 0 до 255. Отже, за кожен з колірних каналів відповідає 8 бітів (1 байт), а глибина кольору зображення в цілому - 24 біта (3 байти).

Популярність даного методу обумовлена його простотою і тим, що він дозволяє приховувати у відносно невеликих файлах досить великі обсяги інформації. Метод найчастіше працює з растровими зображеннями,

представленими у форматі без компресії (наприклад, GIF і BMP).

Метод НЗБ відрізняється від інших низькою стеганографічною витривалістю перед атаками пасивних та активних зловмисників. Його ключовий недолік полягає в високій реактивності на спотворення контейнера. Щоб знизити цю чутливість, зазвичай додатково використовують метод кодування, що витривалий до шуму.

Цей метод використовується для вбудовування будь-яких даних у зображення шляхом зміни бітів точок. Ці дані можуть бути додатково зашифровані, а потім вбудовані в зображення (стегофайл), яке може бути передано через відкритий канал без ризику. Однак, не рекомендується обробляти стегофайл в графічному редакторі, оскільки це може знищити приховану інформацію. Також необхідно обережно завантажувати їх на файлообмінники, оскільки вони також можуть обробляти зображення під час завантаження.

Одним із недоліків методу є обмеження обсягу вбудовуваних даних, яке безпосередньо залежить від розміру зображення: чим більше зображення, тим більше даних можна в нього вставити. Один із варіантів розв'язання цієї проблеми полягає в тому, що дані вставляються в різні службові поля і змінні формату, які не обов'язково мають бути графічними. Обсяг повідомлення може бути майже будь-яким, але така вставка легко виявляється, достатньо просто переглянути значення службових полів. Виявлення методу НЗБ можливе за допомогою статистичних методів.

Основні засоби, якими володіє порушник – статистичні та візуальні атаки. Якщо візуальна атака спрямована на виявлення секретного повідомлення шляхом дослідження стегоповідомлення людиною, її органами почуттів, то статистична атака ґрунтується на дослідженні статистичних характеристик цифрового об'єкта, які порушуються в результаті приховування інформації.

Один із способів підвищити стійкість стеганографічної системи – адаптивний вибір елементів стегоконтейнера для вбудовування інформації. Наприклад, інформація може вбудовуватися в найбільш зашумлені ділянки

зображення, що ускладнює її виявлення порушником. Слід зауважити, однак, що адаптивне правило вибору елементів контейнера найчастіше не залежить або мало залежить від секретного ключа. Ця властивість дозволяє і порушнику провести успішну атаку на стегосистему. Проблема може бути вирішена, якщо при вбудовуванні використовується інформація, доступ до якої порушник отримати не може. Такий підхід до приховування інформації, дозволяючи використовувати переваги адаптивного стеганографічного перетворення, не знижує стійкість системи, так як порушник не має можливості застосувати правило вибору елементів стегоконтейнера. Вилучення повідомлення одержувачем можливо завдяки так званим «wet paper codes».

Ідея стеганографії з порушенням квантування полягає у використанні інформації, що відкидається в результаті обробки стегоконтейнера з втратами. Йдеться про зменшення розмірів зображення, про скорочення колірної гами.

В процесі проведення обробки зображення застосовується дискретне косинусне перетворення (Discrete Cosine Transform), коефіцієнти якого піддаються поділу на елементи матриці квантування і округленню до найближчого цілого. Для вбудовування інформації можна використовувати коефіцієнти, що лежать в інтервалі (n +0; 5 "; n +0; 5 +"), де " – деяке невелике додатне число (наприклад, " = 0; 1), а n – ціле. Для цих коефіцієнтів, званих змінними, помилка округлення максимальна і не змінюється значно при зміні коефіцієнта (якщо значення округленого коефіцієнта змінюється з n на n + 1 або, навпаки, з n + 1 на n).

Автори цього методу мають на увазі високу стійкість стеганографії з порушенням квантування до атак, спрямованим на виявлення наявності прихованого повідомлення. Основний недолік методу полягає в наступному: відправник змушений або використовувати стиснені зображення в якості стеганографічних контейнерів (що не завжди зручно), або пересилати одержувачу двічі стиснуте з допомогою компресії зображення (що підвищує підозрілість стегоповідомлення).

Аналіз стегосистем, що використовують порушення квантування, передбачає пошук методів виявлення модифікованих коефіцієнтів, виходячи з квантованих коефіцієнтів, або на основі самого зображення. Зміни в коефіцієнтах зображення впливають лише на відповідний блок зображення розміром 8x8 пікселів. Результати досліджень підтверджують це припущення для ненульових коефіцієнтів зображення. Дослідження також показують, що пропонований метод дозволяє отримати вибірки, в яких до 70% коефіцієнтів були модифіковані.

Ці результати можуть бути використані для розробки стегосистем, які використовують переваги стеганографії з порушенням квантування, але при цьому не потрібно обмежуватися нестиснутими зображеннями. Замість цього, в якості стеганографічного контейнера може бути використано зображення в форматі BMP.

BMP (від англ. **Bitmap Picture**) – формат зберігання растрових зображень, розроблений компанією Microsoft. З форматом BMP працює величезна кількість програм, так як його підтримка інтегрована в операційні системи Windows і OS/2. Файли формату BMP можуть мати розширення .bmp, .dib та .rle. Також структури з цього формату використовуються деякими WinAPI-функціями підсистеми GDI.

Глибина кольору в даному форматі може бути 1, 4, 8, 16, 24, 32, 48 біт на піксель. При цьому для глибини кольору менше 16 біт використовується палітра з повнокольоровими компонентами глибиною 24 біта.

BMP-файл складається з чотирьох частин:

1) Тема файлу (BITMAPFILEHEADER)

2) Заголовок зображення (BITMAPINFOHEADER, може бути відсутнім). BITMAPV4HEADER (Win95, NT4.0) BITMAPV5HEADER (Win98/Me, 2000/XP і т.д.)

3) Палітра (може бути відсутньою)

4) Зображення

BITMAPFILEHEADER. Ця структура містить інформацію про тип, розмір та представлення даних у файлі. Розмір структури становить 14 байт.

```
typedef struct tagBITMAPFILEHEADER
```

```
{ WORD    bfType;        // зсув 0 байт від початку файлу
  DWORD   bfSize;        // зсув 2 байти від початку файлу, довжина 4 байти
  WORD    bfReserved1;
  WORD    bfReserved2;
  DWORD   bfOffBits;     // зсув 10 байт від початку файлу, довжина 4 байти
} BITMAPFILEHEADER, *PBITMAPFILEHEADER;
```

Тип `WORD` повинен мати розмір 16 біт, типи `DWORD` і `LONG` - 32 біта, тип `LONG` - знаковий, порядок байтів мається на увазі little endian.

`bfType` - тип файлу, символи «BM» (у HEX: `0x42 0x4d`).

`bfSize` - розмір всього файлу в байтах.

`bfReserved1` і `bfReserved2` - зарезервовані, повинні містити нулі.

`bfOffBits` - містить зсув в байтах від початку структури `BITMAPFILEHEADER` до безпосередньо бітів зображення.

Палітра може містити послідовність чотирьохбайтових полів за кількістю доступних кольорів (256 для 8-бітного зображення). Три молодші байти кожного поля визначають інтенсивність червоного, зеленого і синього компонентів кольору, старший байт не використовується. Кожен піксель зображення описаний в такому випадку одним байтом, що містить номер поля палітри, в якому збережений колір цього пікселя.

Якщо піксель зображення описується 16-бітним числом, палітра може зберігати три двобайтових значення, кожне з яких визначає маску для вилучення з 16-бітного пікселя червоної, зеленої і синьої компоненти кольору.

Файл BMP може не містити палітри, якщо в ньому зберігається нестиснене повнокольорове зображення.

Дані зображення – послідовність пікселів, записаних в тому чи іншому вигляді. Пікселі зберігаються рядками, знизу вгору. Кожен рядок зображення доповнюється нулями до довжини, кратної чотирьом байтам.

У bmp-файлах з глибиною кольору 24 біта, байти кольору кожного пікселя зберігаються в зворотному порядку BGR (Blue, Green, Red)

В bmp-файлах з глибиною кольору 32 біта, байти кольору кожного пікселя зберігаються в прямому порядку ARGB (Alpha, Red, Green, Blue)

Бітність зображення. Залежно від кількості представлених кольорів, на кожну точку відводиться від 1 до 48 бітів:

– 1 біт – монохромне зображення (два кольори).

– 2 біти – 4 можливих кольору (режими роботи CGA) (2-бітовий режим офіційно не стандартизований, але використовується).

– 4 біти – 16-кольорове зображення (режими роботи EGA).

– 8 біт (1 байт) – 256 кольорів, останній з режимів, які підтримували індексовані кольори.

– 16 біт (2 байти) – режим HiColor, Для 5-6-5 = 65536 можливих відтінків, для 5-5-5 = 32768 можливих відтінків.

– 24 біти (3 байти) – TrueColor. У зв'язку з тим, що 3 байти не дуже добре співвідносяться зі ступенями двійки (особливо при зберіганні даних у пам'яті, де вирівнювання даних по межі слова має значення), замість нього часто використовують 32-бітове зображення. У режимі TrueColor на кожен з трьох каналів (в режимі RGB) відводиться по 1 байту (256 можливих значень), загальна кількість кольорів дорівнює 16777216.

– 32 біти (4 байти) – цей режим практично аналогічний TrueColor, четвертий байт зазвичай не використовується, або в ньому розташовується альфа-канал (прозорість).

– 48 біт (6 байт) – рідко використовуваний формат з підвищеною точністю передачі кольору (16 біт на канал), підтримується відносно малою кількістю програм і обладнання.

Індексовані кольори. При кількості біт 1 (2 кольори), 2 (4 кольори), 4 (16 кольорів) або 8 (256 кольорів) на кожен піксель, може використовуватися спеціальний режим індексованих кольорів. У цьому випадку число, що відповідає кожному пікселю, вказує не на колір, а на номер кольору в палітрі. Завдяки використанню палітри є можливість адаптувати зображення до кольорів, присутніх на зображенні. У такому випадку зображення обмежене не заданими кольорами, а максимальною кількістю одночасно використовуваних кольорів.

3.2 Розробка структурної схеми

На рисунку 3.2 зображена структурна схема розроблюваної системи.

Рисунок 3.2 – Структурна схема системи

На структурній схемі показано два процеси, що відбуваються у розроблюваній системі:

1. Процес вбудовування невидимих стегоповідомлень у графічні файли.

2. Процес вилучення та читання невидимих стегоповідомлень з графічних файлів.

Процес вбудовування невидимих стегоповідомлень складається з безпосередньо створення стего та його вбудовування у файл, дані дії реалізовані за допомогою наступних модулів розроблюваного програмного забезпечення:

1. Формування тексту та/або зображення для стего.
2. Генератор ключів для шифратора.
3. Шифратор AES.
4. Перешкодостійке кодування CRC-32.
5. Генератор ключів для стегокодера.
6. Стегокодер LSB.

Процес перевірки невидимих стегоповідомлень складається з вилучення стего та перевірки авторських прав, дані дії реалізовані за допомогою наступних модулів розроблюваного програмного забезпечення:

1. Стегодекодер LSB.
2. Перевірка цілісності стего (CRC-32).
3. Дешифратор AES.
4. Читання тексту та/або зображення стего.
5. Порівняння виявленої інформації з інформацією у БД.

Загальний принцип роботи програмного забезпечення наступний: 1) в стегоконтейнери перед їх відправкою через Інтернет для прихованої передачі інформації вбудовується зашифроване невидиме стегоповідомлення, 2) отримувач стегоконтейнеру робить спробу вилучити приховане стегоповідомлення, 3) для успішного вилучення стегоповідомлення одержувач повинен був раніше отримати стегоключ та ключ шифрування від відправника інформації надійним каналом зв’язку, наприклад, при особистій зустрічі через флеш-носій. Також обмін стегоключем та ключем шифрування може здійснюватися на основі асиметричного шифрування по захищеному каналу зв’яку.

Таким чином розроблене програмне забезпечення дозволяє здійснювати приховану передачу інформації по незахищеному каналу зв’язку.

3.3 Розробка функціональної схеми

На рисунку 3.3 зображена функціональна схема підпрограми вбудовування стегоповідомлення.

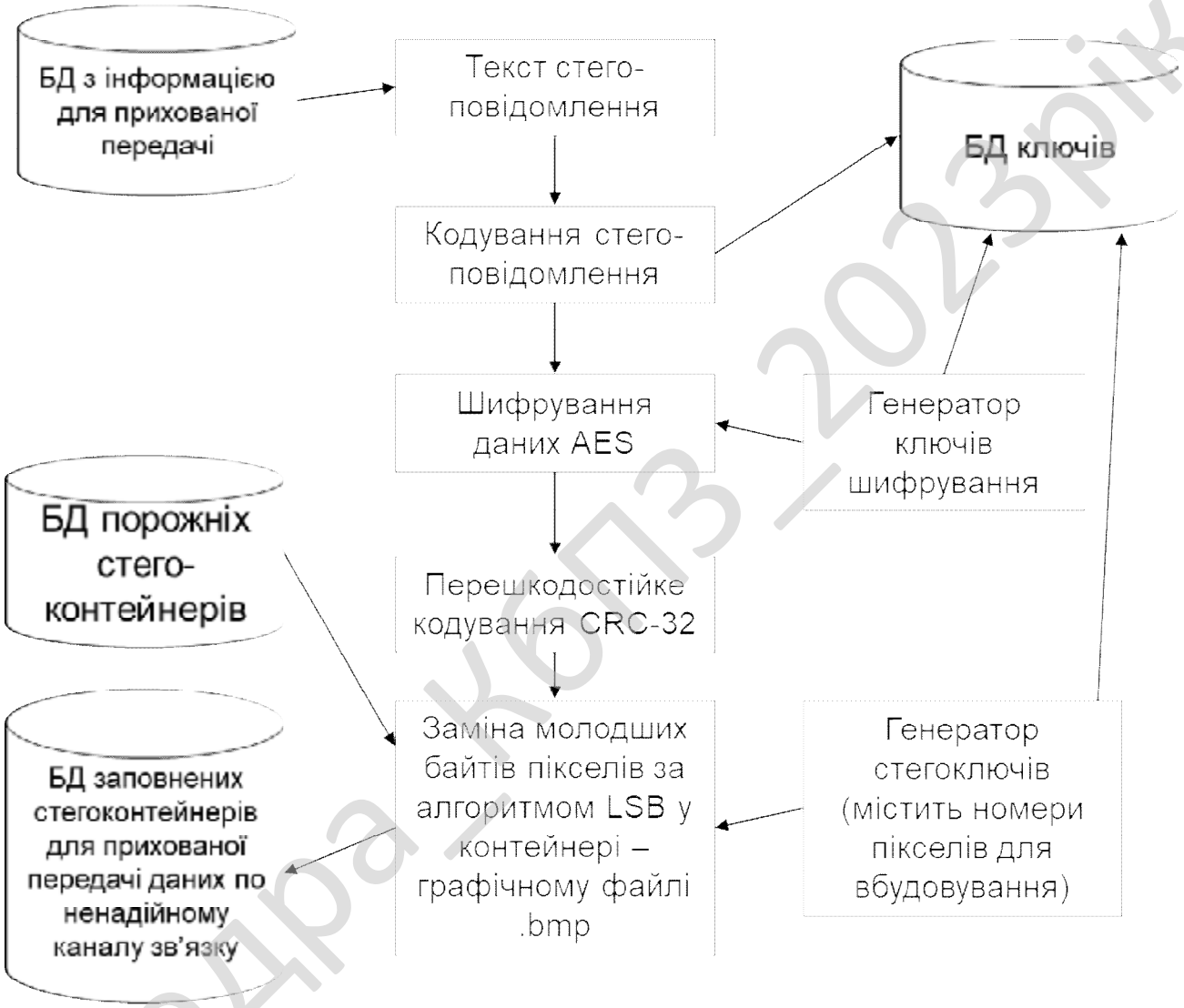

Рисунок 3.3 – Функціональна схема підпрограми вбудовування стегоповідомлення

Як видно з рисунку, стегокодер складається з наступних частин:

1. БД з інформацією для прихованої передачі.
2. БД порожніх стего-контейнерів.

3. БД заповнених стегоконтейнерів для прихованої передачі даних по ненадійному каналу зв’язку.

4. БД ключів.

5. Текст стего-повідомлення.

6. Кодування стего-повідомлення.

7. Шифрування даних AES.

8. Перешкодостійке кодування CRC-32.

9. Заміна молодших байтів пікселів за алгоритмом LSB у контейнері – графічному файлі .bmp.

10. Генератор стегоключів (містить номери пікселів для вбудовування).

11. Генератор ключів шифрування.

На рисунку 3.4 зображена функціональна схема підпрограми вилучення стегоповідомлення.

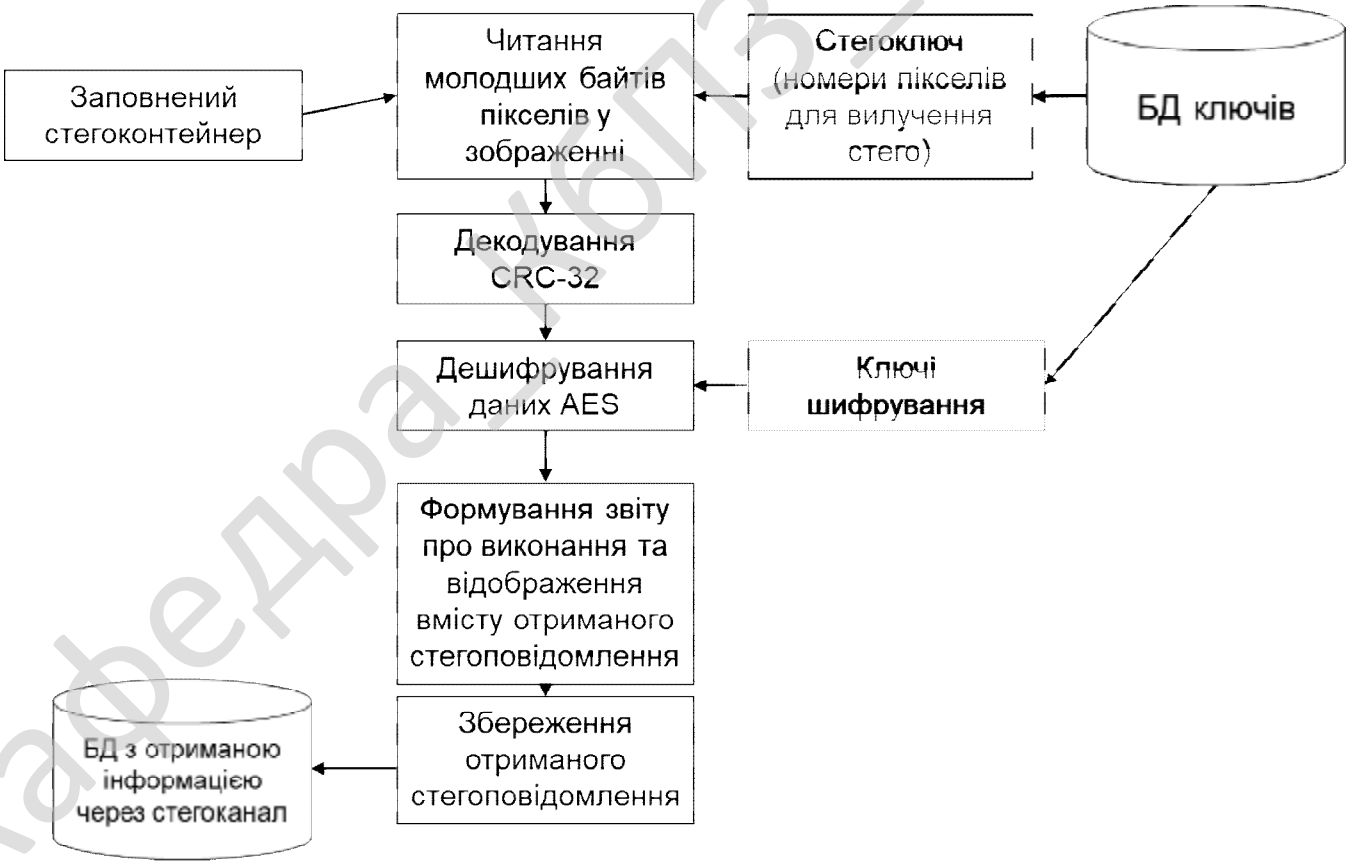

Рисунок 3.4 – Функціональна схема підпрограми вилучення стегоповідомлення

Як видно з рисунку, стегодекодер складається з наступних частин:

1. БД з отриманою інформацією через стегоканал.
2. БД ключів.
3. Читання молодших байтів пікселів у зображенні.
4. Стегоключ (номери пікселів для вилучення стего).
5. Ключі шифрування.
6. Читання молодших байтів пікселів у зображенні.
7. Декодування CRC-32.
8. Дешифрування даних AES.
9. Формування звіту про виконання та відображення вмісту отриманого стегоповідомлення.
10. Збереження отриманого стегоповідомлення.

Стегокодер та стегодекодер дозволяють забезпечити приховану передачу даних між користувачами через ненадійний канал зв’язку. Для того, щоб система адекватно виконувала свої функції – важливо попередньо обмінятися стегоключем та ключем шифрування через надійний канал зв’язку, а також періодично їх оновлювати.

3.4 Розробка діаграми процесів

На рисунку 3.5 зображена діаграма взаємодії процесів у розробленій системі.

Як видно з наведеного рисунку, система складається з наступних процесів:

1. Завантаження файлів для вбудовування стего.
2. Формування стегоповідомлення.
3. Вбудовування стего у файли.
4. Стегокодер LSB.
5. Кодер CRC-32.

6. Шифратор AES.

7. Передача даних по ненадійному каналу зв’язку у заповнених стегоконтейнерах.

8. Передача ключів надійним каналом зв’язку.

9. Завантаження файлу для читання стего.

10. Вилучення стегоповідомлення.

11. Стегодекодер LSB.

12. Декодер CRC-32.

13. Дешифратор AES.

14. Виведення на екран тексту стего.

15. Збереження вилученого стегоповідомлення.

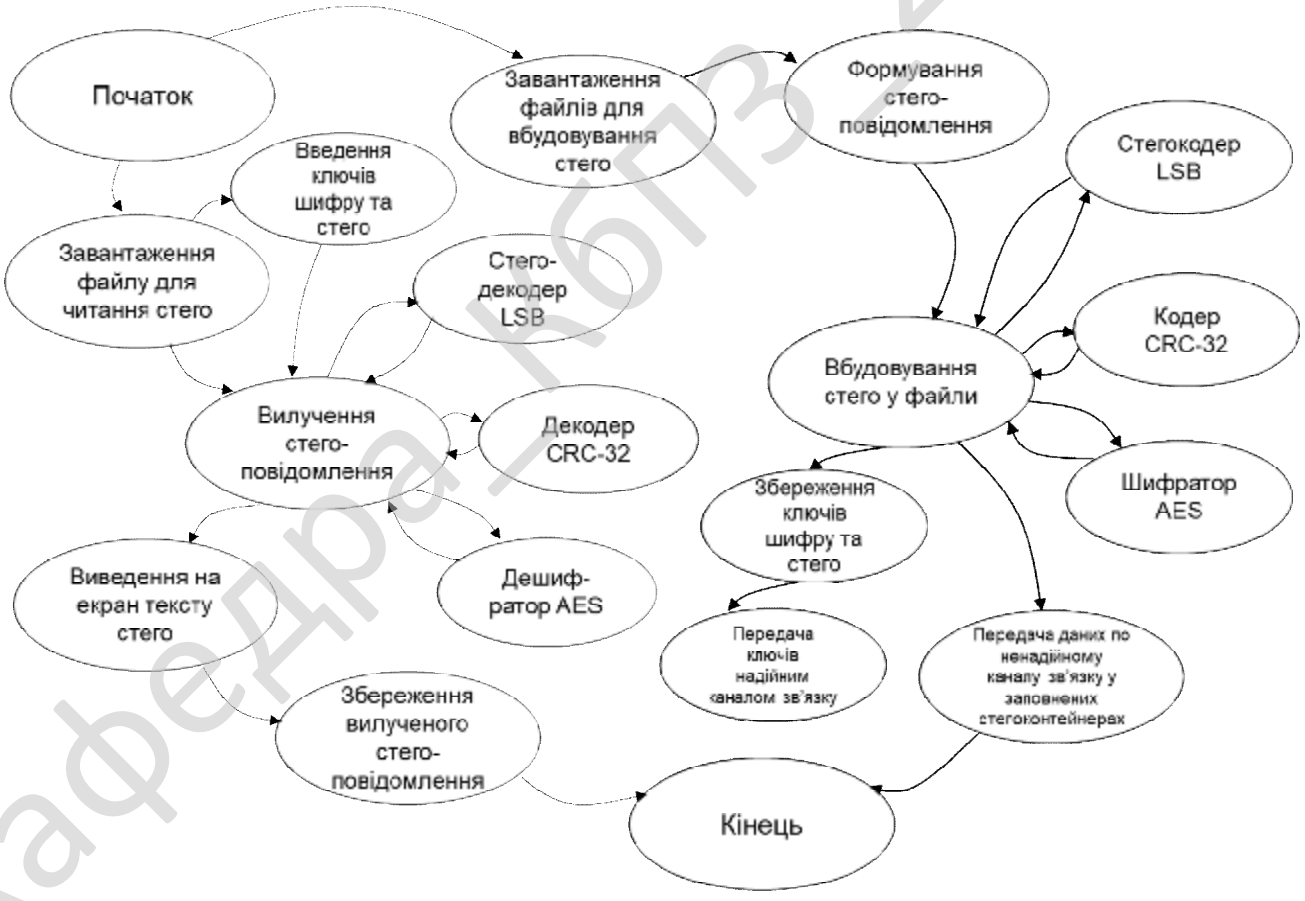

Рисунок 3.5 – Діаграма взаємодії процесів

Перераховані процеси взаємодіють між собою наступним чином. Після запуску системи користувач може ініціювати один з двох процесів або процес завантаження файлів для вбудовування стего, або процес завантаження файлу для читання стего.

Процес завантаження файлів для вбудовування стего викликає процес формування стего для вбудовування у стегоконтейнер та для прихованої передачі ненадійним каналом зв'язку, після якого запускається процес вбудовування стего у файли. Процес вбудовування стего у файли безпосередньо пов'язаний з процесами Стегокодер LSB, Кодер CRC-32 та Шифратор AES.

Процес завантаження файлу для читання стего після свого виконання запускає процес вилучення стего, за яким слідують процеси вилучення стего з графічного файлу, виведення та збереження влученого стегоповідомлення. Процес вилучення стего пов'язаний з процксами Стегодекодер LSB, Декодер CRC-32 та Дешифратор AES.

4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Розглянемо алгоритм роботи основної програми. Його блок-схема зображена на рисунку 4.1. З рисунку видно, що після запуску програми спочатку відбувається вивід основного вікна програми. Потім користувач обирає одну з наступних дій:

– Вбудовування стего.

– Читання стего.

– Вихід з програмного забезпечення.

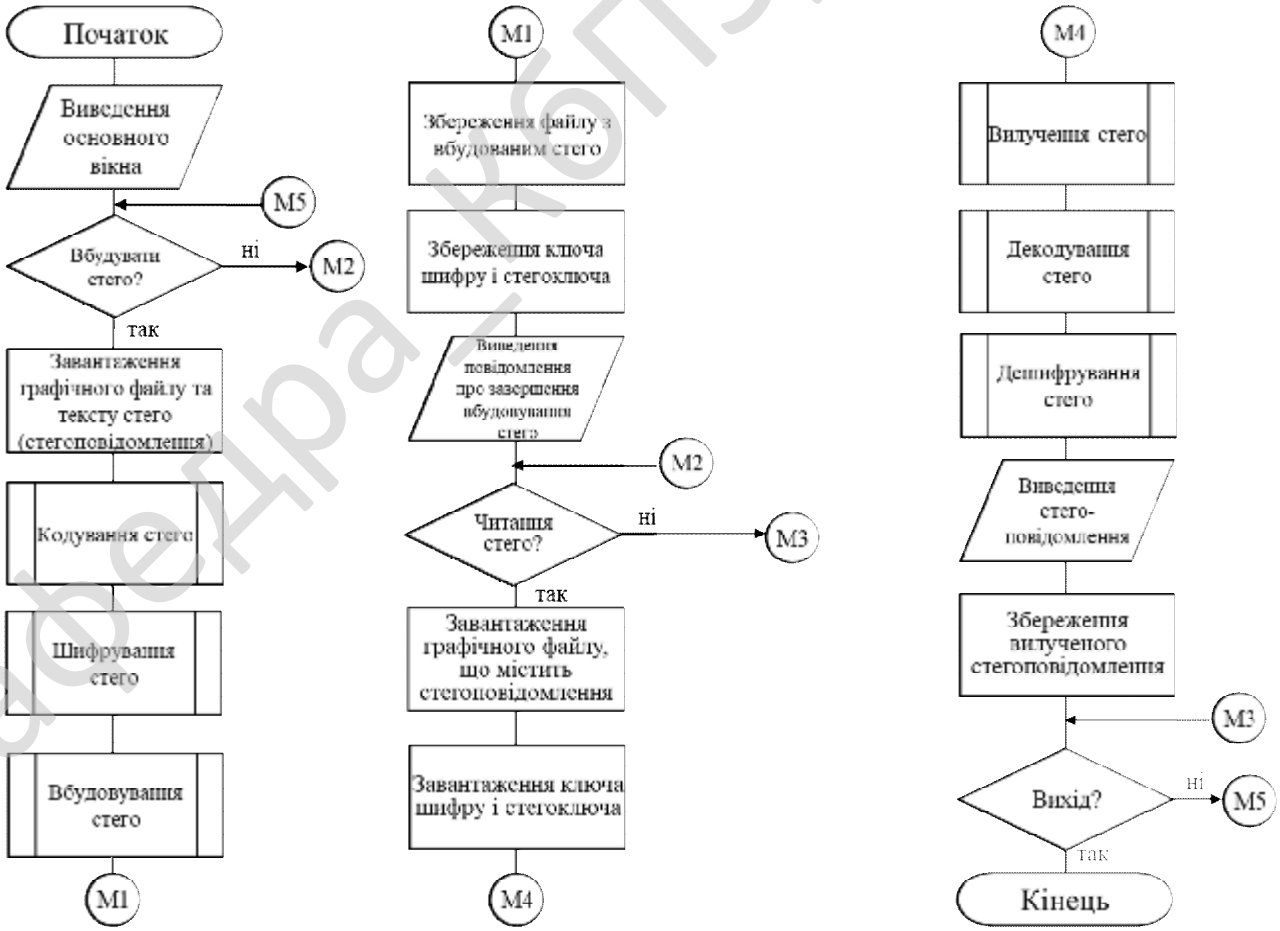

Рисунок 4.1 – Блок-схема роботи основної програми

Якщо користувач обирає вбудовування стего, тоді виконуються наступні дії:

1. Завантаження файлу-контейнеру для вбудовування стего та файлу з текстом стегоповідомлення.

2. Кодування стего.

3. Шифрування стего.

4. Вбудовування стего.

5. Збереження файлу з вбудованим стего.

6. Збереження ключа шифру і стегоключа.

7. Виведення повідомлення про завершення вбудовування стего.

Якщо користувач обирає перевірка стего, тоді виконуються наступні дії:

1. Завантаження графічного файлу, що містить стегоповідомлення.

2. Завантаження ключа шифру і стегоключа.

3. Вилучення стего.

4. Декодування стего.

5. Дешифрування стего.

6. Виведення стегоповідомлення.

7. Збереження вилученого стегоповідомлення.

Найважливішими підпрограмами розробленої програми являються підпрограми стегокодера та стегодекодера, що дозволяють вбудовувати та вилучати приховані стегоповідомлення. Принципи роботи даних підпрограм і будуть розглянуті нижче.

На рисунку 4.2 зображена блок-схема роботи алгоритму стекгокодера. Вона працює наступним чином:

– Спершу відбувається читання стегоключа, повідомлення та файлу для вбудовування стего.

– Після цього відбувається розбиття зображення на фрагменти.

– Наступним кроком є ініціалізація лічильника фрагментів.

– Після цього відбувається вибір пікселя з поточного фрагменту

відповідно до стегоключа.

– Наступним етапом є розрахунок значення яскравості обраного пікселя.

– Якщо у піксель потрібно записати «0», толі збільшуємо яскравість пікселя. У противному випадку, зменшуємо яскравість пікселя.

– Далі збільшується лічильник на одиницю, й переходимо до наступного пікселя згідно стегоключа. Якщо оброблено останній фрагмент, тоді відбувається завершення процедури.

– У іншому випадку, відбувається перехід до наступного фрагменту, й процедура повертається на етап вибору пікселя з поточного фрагменту, відповідно до стегоключа.

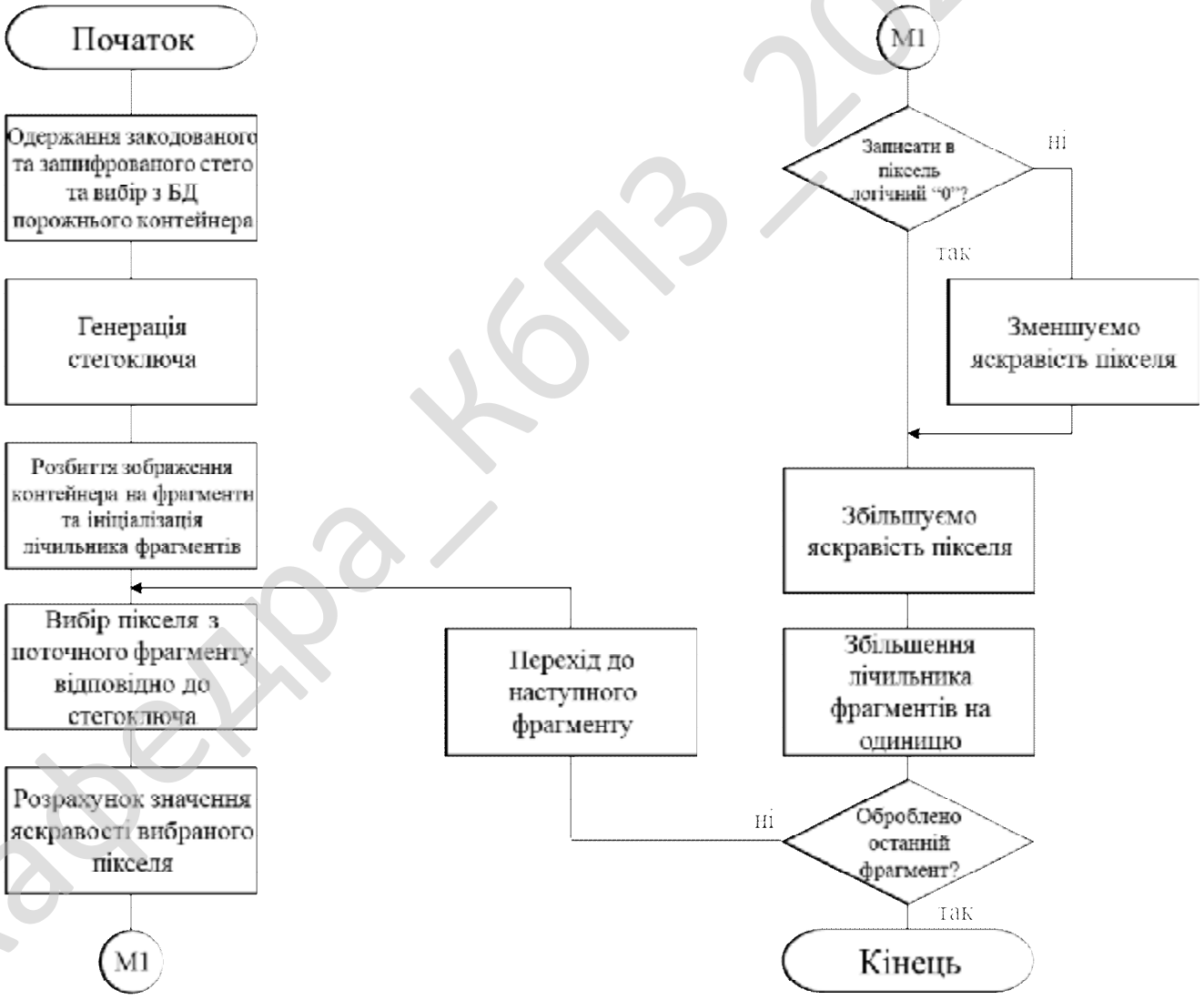

Рисунок 4.2 – Блок-схема роботи підпрограми стегокодера

На рисунку 4.3 зображено блок-схему роботи алгоритму стегодекодера. Вона працює наступним чином:

– Спершу відбувається відкриття стегоключа та зображення.

– Після цього відбувається розбиття контейнера на фрагменти.

– Наступним кроком є ініціалізація лічильника фрагментів.

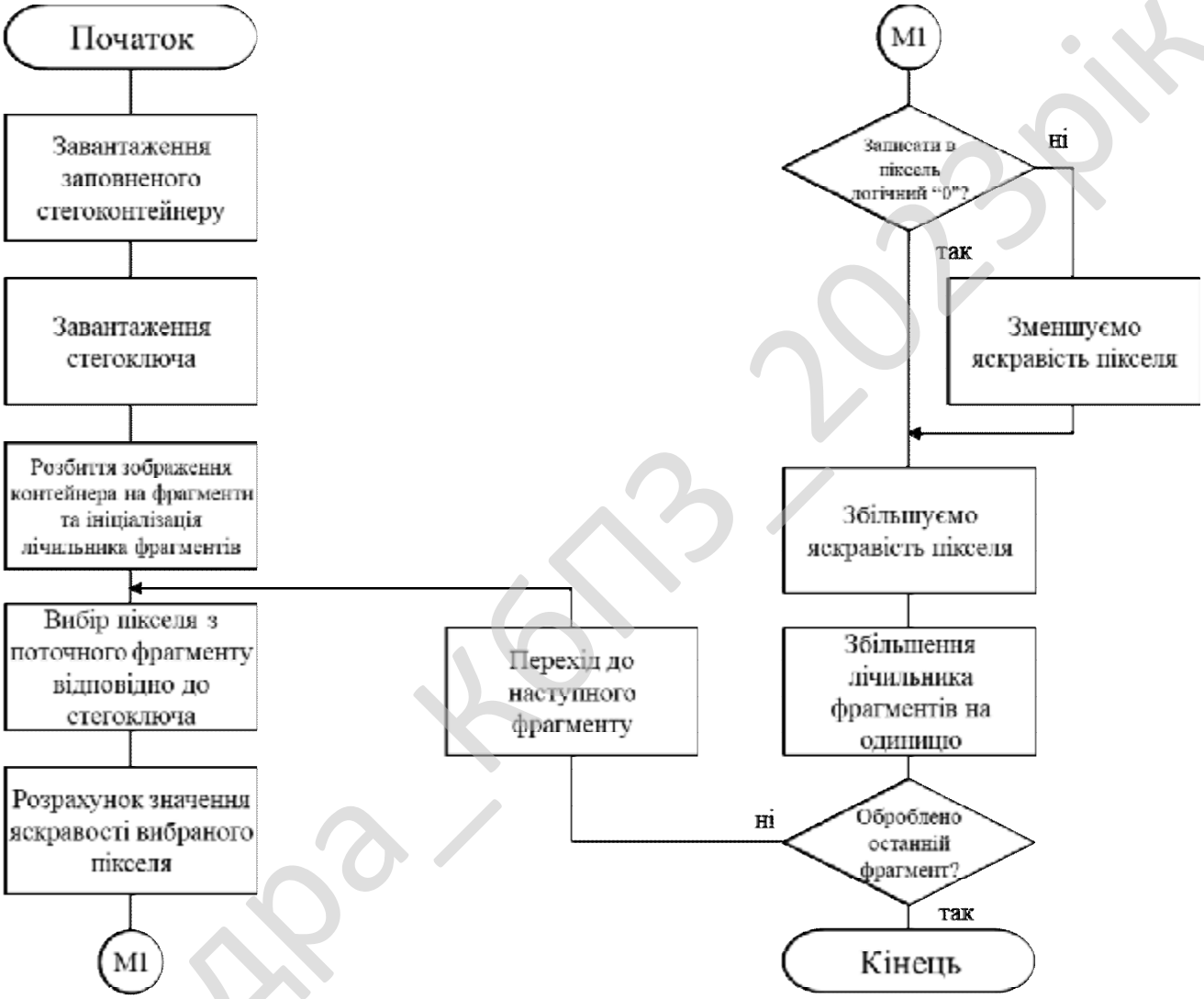

Рисунок 4.3 – Блок-схема роботи підпрограми стегодекодера

– Після цього відбувається вибір пікселя з поточного фрагменту відповідно до стегоключа.

– Наступним кроком є розрахунок середнього значення яскравості інших пікселів у фрагменті.

– Якщо обраний піксель більш яскравий, тоді відбувається читання з пікселя логічного «0», у іншому випадку відбувається читання з пікселя логічної «1».

– Далі збільшується лічильник фрагментів на одиницю, й переходимо до наступного пікселя згідно стегоключа. Якщо оброблено останній фрагмент, тоді відбувається завершення процедури.

– У іншому випадку, відбувається перехід до наступного фрагменту, й процедура повертається на етап вибору пікселя з поточного фрагменту, відповідно до стегоключа.

Наведемо частину коду програми, що реалізує вбудовування та читання стегоповідомлень:

```
    private void BackgroundWorker_DoWork(object sender, DoWorkEventArgs e)
    {
        string str = "FileTagging.exe";
        if (((int) e.Argument) == 0)
        {
            string str2 = "\"" + this.FileToHide.Text + "\" \"" +
(this.UseBase.Checked ? "0" : this.BaseBitmap.Text) + "\" \"" +
this.OutputFile.Text + "\" " + this.GetBits().ToString();
            if (this.Pass1.Text.Length > 0)
            {
                str2 = str2 + " \"" + this.Pass1.Text + "\"";
            }

     byte[] encrypted = AesEncryption.Encrypt(str2, aesAlg.Key, aesAlg.IV);
     SaveKeyAndIV(keyFile, aesAlg.Key, aesAlg.IV);

     string keyFile = "key.dat";
     str2 = Convert.ToBase64String(encrypted);

      str2 = CRC32.ComputeChecksum(str2);

            try
            {
                MethodInvoker method = null;
                using (Process ProcessObj = new Process())
```

```
                {
                    ProcessObj.StartInfo.FileName = str;
                    ProcessObj.StartInfo.Arguments = str2;
                    ProcessObj.StartInfo.UseShellExecute = false;
                    ProcessObj.StartInfo.CreateNoWindow = true;
                    ProcessObj.StartInfo.RedirectStandardOutput = true;
                    ProcessObj.Start();
                    ProcessObj.WaitForExit();


                    if (method == null)
                    {

                        method = (MethodInvoker) (() => (this.ExecLog.Text =
ProcessObj.StandardOutput.ReadToEnd()));
                    }
                    this.ExecLog.Invoke(method);
                    if (ProcessObj.ExitCode != 0)
                    {
                        MessageBox.Show("Сталася помилка приховування стего",
"Помилка", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                    }
                    else
                    {
                        MessageBox.Show("Ваші дані стего були успішно
вбудовані", "Успішно!", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);
                    }
                }
            }


            catch (Exception)
            {
                MessageBox.Show("Сталася помилка виконання процесу",
"Помилка", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
            }
        }


        else
        {
```

```
            string str3 = (this.ImageType.Checked ? "1" : "0") + " \"" +
this.BitmapWithHiddenData.Text + "\"";
            if (this.Pass.Text.Length > 0)
            {
                str3 = str3 + " \"" + this.Pass.Text + "\"";
            }

      uint crc = CRC32.ComputeChecksum(str3)
      p = CRC32.CheckChecksum(str3, crc)
      if(p==false)
      {
            Console.WriteLine("Перевірка CRC-32 виявила порушення цілісності
      даних стего. ");
      }

      string keyFile = "key.dat";
      string decrypted = AesEncryption.Decrypt(encrypted, aesAlg.Key,
aesAlg.IV);
      str3 = decrypted;
      byte[] loadedKey, loadedIV;
      LoadKeyAndIV(keyFile, out loadedKey, out loadedIV);

            try
            {
                using (Process process = new Process())
                {
                    process.StartInfo.FileName = str;
                    process.StartInfo.Arguments = str3;
                    process.StartInfo.UseShellExecute = false;
                    process.StartInfo.CreateNoWindow = true;
                    process.StartInfo.RedirectStandardOutput = true;
                    process.Start();
                    process.WaitForExit();
                    this.ExecLog.Text = process.StandardOutput.ReadToEnd();
                    if (process.ExitCode != 0)
                    {
                        MessageBox.Show("Сталася помилка вилучення
прихованого стего", "Помилка", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);
                    }
```

```
                    else
                    {
                        MessageBox.Show("Ваші дані стего були успішно
вилучені", "Успіх!", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);
                    }
                }
            }


            catch (Exception)
            {
                MessageBox.Show("Сталася помилка виконання процесу",
"Помилка", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
            }
        }
    }
```

Для контролю цілісності вбудовуваних у файли контейнери стегоповідомлень було застосовано алгоритм обчислення контрольної суми CRC-32, що дозволяє виправляти частково пошкоджені стегоповідомлення. Пошкодження стегоповідомлень може виникати внаслідок помилок у каналі зв'язку, або атак зловмисників накладанням шуму, стисненням з втратами тощо, що мають ціллю знищення прихованих стегоповідомлень. Розглянемо детальніше принцип дії алгоритму CRC-32.

Алгоритм обчислення контрольної суми CRC-32

CRC (англ. *Cyclic redundancy check*) – циклічний надлишковий код – алгоритм обчислення контрольної суми, призначений для перевірки цілісності даних. CRC є практичним додатком завадостійкого кодування, заснованого на певних математичних властивостях циклічного коду.

Завадостійке кодування

Перші спроби створення кодів з надлишковою інформацією почалися задовго до появи сучасних ПК. Наприклад, ще в шістдесятих роках минулого століття Рідом і Соломоном була розроблена ефективна методика кодування – Код Ріда-Соломона. Використання її в ті часи не представлялося можливим,

тому що зробити операцію декодування за розумний час першими алгоритмами не вдавалося. Точку в цьому питанні поставила фундаментальна робота Берлекампа, опублікована в 1968 році. Ця методика, на практичне застосування якої вказав через рік Мессі, і донині використовується в цифрових пристроях, що забезпечують прийом RS-кодованих даних. Більше того: дана система дозволяє не тільки визначати позиції, але й виправляти невірні кодові символи (найчастіше октети).

Але далеко не скрізь від коду потрібна корекція помилок. Багато сучасних каналів зв'язку мають прийнятні характеристики, і найчастіше досить лише перевірити, чи успішно пройшла передача або виникли які-небудь складності; структура ж помилок і конкретні позиції невірних символів зовсім не цікавлять приймаючу сторону. І в цих умовах дуже вдалим рішенням виявилися алгоритми, що використовують контрольні суми. CRC як можна найкраще підходить для подібних завдань: невисокі витрати ресурсів, простота реалізації й уже сформований математичний апарат з теорії лінійних циклічних кодів забезпечили їй величезну популярність.

Контрольна сума

У найзагальнішому вигляді контрольна сума являє собою деяке значення, побудоване за певною схемою на основі кодуємого повідомлення. Перевірочна інформація при **систематичному кодуванні** дописується в кінець повідомлення – після корисних даних. На приймаючій стороні абонент знає алгоритм обчислення контрольної суми: відповідно, програма має можливість перевірити коректність прийнятих даних.

При передачі пакетів по реальному каналу, зрозуміло, можуть виникнути викривлення передаваємої інформації внаслідок різних зовнішніх впливів: електричних наведень, поганих погодних умов і багатьох інших. Сутність методики в тому, що при гарних характеристиках хэш-функції в переважному числі випадків помилка в повідомленні призведе до зміни обчисленого на приймальній стороні значення CRC. Якщо вихідна й обчислена суми не рівні між

собою, приймається рішення про невірогідність прийнятих даних, і можна запросити повторну передачу пакета.

Математичний опис

Алгоритм CRC базується на властивостях ділення з остачею двійкових багаточленів, тобто багаточленів над кінцевим полем CF(2^N). Значення CRC є по суті остачею від ділення багаточлена, що відповідає вхідним даним, на якийсь фіксований породжуючий багаточлен.

Кожній кінцевій послідовності бітів a_0, a_1, … , a_{n-1} взаємно однозначно зіставляється двійковий поліном $\sum_{n=0}^{N-1} a_n x^n$, послідовність коефіцієнтів якого являє собою вихідну послідовність. Наприклад, послідовність бітів 1011010 відповідає багаточлену:

$$P(x) = 1 \cdot x^6 + 0 \cdot x^5 + 1 \cdot x^4 + 1 \cdot x^3 + 0 \cdot x^2 + 1 \cdot x^1 + 0 \cdot x^0 = x^6 + x^4 + x^3 + x^1$$

Кількість різних багаточленів ступеня меншого N дорівнює 2^N, що збігається із числом всіх двійкових послідовностей довжини N.

Значення контрольної суми в алгоритмі з породжуючим багаточленом G(x) ступеня N визначається як бітова послідовність довжини N, що представляє багаточлен R(x), який вийшов у залишку при діленні багаточлена P(x), що представляє вхідний потік бітів, на багаточлен G(x):

$$R(x) = P(x) \cdot x^N \bmod G(x)$$

де:

R(x) – багаточлен, що представляє значення CRC.

P(x) – багаточлен, коефіцієнти якого представляють вхідні дані.

G(x) – породжуючий багаточлен.

N – ступінь породжуючого багаточлена.

Множення x^n здійснюється приписуванням N нульових бітів до вхідної послідовності, що поліпшує якість хешування для коротких вхідних послідовностей.

При діленні з остачею вхідного багаточлена на породжуючий поліном G(x) ступеня N можна одержати 2^N різних залишків від ділення. G(x) завжди є

не привідним многочленом. Звичайно його підбирають відповідно до вимог до хеш-функції в контексті кожного конкретного застосування.

Проте, існує безліч стандартизованих утворюючих багаточленів, що володіють гарними математичними й кореляційними властивостями (мінімальне число колізій, простота обчислення).

Обчислення CRC. Параметри алгоритму

Говорячи про формування контрольної суми CRC, у першу чергу потрібно згадати про поліноми-генератори. Існує велика множина багаточленів, що беруть участь у формуванні *cyclic reduntancy code*.

Іншим параметром конкретного алгоритму обчислення контрольної суми є розмір слова, або сумарна кількість регістрів – інформаційних осередків, використовуваних для обчислення чисельного значення хешу. При цьому обов'язково враховується те, що розмір слова й степінь утворюючого контрольну суму полінома збігаються. На практиці найбільш поширені 8, 16 і 32 – бітові слова, що є наслідком особливостей архітектури сучасної обчислювальної техніки.

І останній параметр, важливий при описі певної методики – початкові стани регістрів (стартове слово). Це остання із трьох значимих характеристик; знаючи їх у сукупності, можна відновити алгоритм обчислення CRC, якщо дана модифікація методики не має специфічних особливостей, таких, як зворотний порядок обробки бітів.

Опис процедури формування CRC-32

З файлу береться перше 32-бітне слово. Якщо старший біт у слові «1», то слово зсувається вліво на один розряд з наступним виконанням операції XOR. Відповідно, якщо старший біт у слові «0», то після зсуву операція XOR не виконується. Після зсуву губиться старий старший біт, а молодший біт звільняється – його значення встановлюється рівним нулю. На місце молодшого біта завантажується черговий біт з файлу, і операція повторюється доти, поки не

завантажиться останній біт файлу. Після проходження всього файлу, у слові залишається залишок, що і є контрольною сумою.

Найбільш використовувані й стандартизовані поліноми

У той час, як циклічні надлишкові коди є частиною стандартів, у цього терміна не існує загальноприйнятого визначення – трактування різних авторів нерідко суперечать один одному.

Цей парадокс стосується й вибору багаточлена-генератора: найчастіше стандартизовані поліноми не є найефективнішими в плані статистичних властивостей відповідного ним *check reduntancy code* .

Найпопулярніший і рекомендуємий IEEE поліном для CRC-32 використовується в Ethernet, FDDI; також цей багаточлен є генератором коду Хеммінга. Використання іншого полінома – CRC-32C – дозволяє досягти такої ж продуктивності при довжині вихідного повідомлення від 58 біт до 131 Кбіт, а в деяких діапазонах довжини вхідного повідомлення може бути навіть вище – тому в наші дні він теж користується популярністю.

Наведемо приклад коду для розрахунку та перевірки CRC-32:

```
using System.Security.Cryptography;

public class CRC32
{
    private static uint[] table;

    public static uint ComputeChecksum(string input)
    {
        if (table == null)
        {
            InitializeTable();
        }

        uint crc = 0xffffffff;
        byte[] bytes = Encoding.UTF8.GetBytes(input);

        for (int i = 0; i < bytes.Length; ++i)
        {
```

```
            byte index = (byte)(((crc) & 0xff) ^ bytes[i]);
            crc = (uint)((crc >> 8) ^ table[index]);
        }

        return ~crc;
    }

    public static bool CheckChecksum(string input, uint checksum)
    {
        return ComputeChecksum(input) == checksum;
    }

    private static void InitializeTable()
    {
        uint poly = 0xedb88320;
        table = new uint[256];

        for (uint i = 0; i < table.Length; ++i)
        {
            uint temp = i;
            for (int j = 8; j > 0; --j)
            {
                if ((temp & 1) == 1)
                {
                    temp = (uint)((temp >> 1) ^ poly);
                }
                else
                {
                    temp >>= 1;
                }
            }
            table[i] = temp;
        }
    }
}
```

Для шифрування та дешифрування стегоповідомлень було розроблено наступні функції:

```
public class AesEncryption
{
```

```
        public static byte[] Encrypt(string plainText, byte[] Key, byte[] IV)
        {
            byte[] encrypted;

            using (Aes aesAlg = Aes.Create())
            {
                aesAlg.Key = Key;
                aesAlg.IV = IV;

                ICryptoTransform encryptor = aesAlg.CreateEncryptor(aesAlg.Key,
aesAlg.IV);

                using (MemoryStream msEncrypt = new MemoryStream())
                {
                    using (CryptoStream csEncrypt = new CryptoStream(msEncrypt,
encryptor, CryptoStreamMode.Write))
                    {
                        using      (StreamWriter      swEncrypt      =      new
StreamWriter(csEncrypt))
                        {
                            swEncrypt.Write(plainText);
                        }

                        encrypted = msEncrypt.ToArray();
                    }
                }
            }

            return encrypted;
        }

        public static string Decrypt(byte[] cipherText, byte[] Key, byte[] IV)
        {
            string plaintext = null;

            using (Aes aesAlg = Aes.Create())
            {
                aesAlg.Key = Key;
                aesAlg.IV = IV;

                ICryptoTransform decryptor = aesAlg.CreateDecryptor(aesAlg.Key,
aesAlg.IV);
```

```
                using (MemoryStream msDecrypt = new MemoryStream(cipherText))
                {
                    using (CryptoStream csDecrypt = new CryptoStream(msDecrypt,
decryptor, CryptoStreamMode.Read))
                    {
                        using (StreamReader srDecrypt = new
StreamReader(csDecrypt))
                        {
                            plaintext = srDecrypt.ReadToEnd();
                        }
                    }
                }
            }

            return plaintext;
        }
    }
```

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення включає в себе різні техніки та методи, щоб запобігти несанкціонованому копіюванню та поширенню програми. Один з таких методів – це використання алгоритму шифрування RSA та введення ключа ліцензії перед використанням програми.

Ключ ліцензії – це спеціальний код, який вводиться користувачем перед використанням програмного забезпечення, і який забезпечує легальне використання програми. Ключ може бути згенерований на основі унікальної ідентифікаційної інформації про комп’ютер або користувача. Ключ ліцензії вводиться перед початком використання програмного забезпечення і перевіряється на правильність, що забезпечує легальність використання програми.

Ключі ліцензії зазвичай складаються з послідовності символів або чисел, що можуть бути згенеровані з використанням різних алгоритмів. Користувач

отримує один ключ, який може бути введений перед використанням програмного забезпечення для перевірки легальності використання програми.

Алгоритм RSA використовується для шифрування програмного забезпечення. Пропонується використовувати як ключ ліцензії закритий RSA ключ, а відкритий ключ використовувати для шифрування програмного забезпечення. Після введення ключа ліцензії програмне забезпечення розшифровується та користувач отримує доступ до його легального використання.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Інтерфейс розробленого програмного забезпечення зображено на рисунках 5.1-5.4. З рисунків видно, що головне вікно програми містить наступні вкладки:

1. Вбудовування стего.
2. Вилучення ЦЗВ.
3. Про програму.

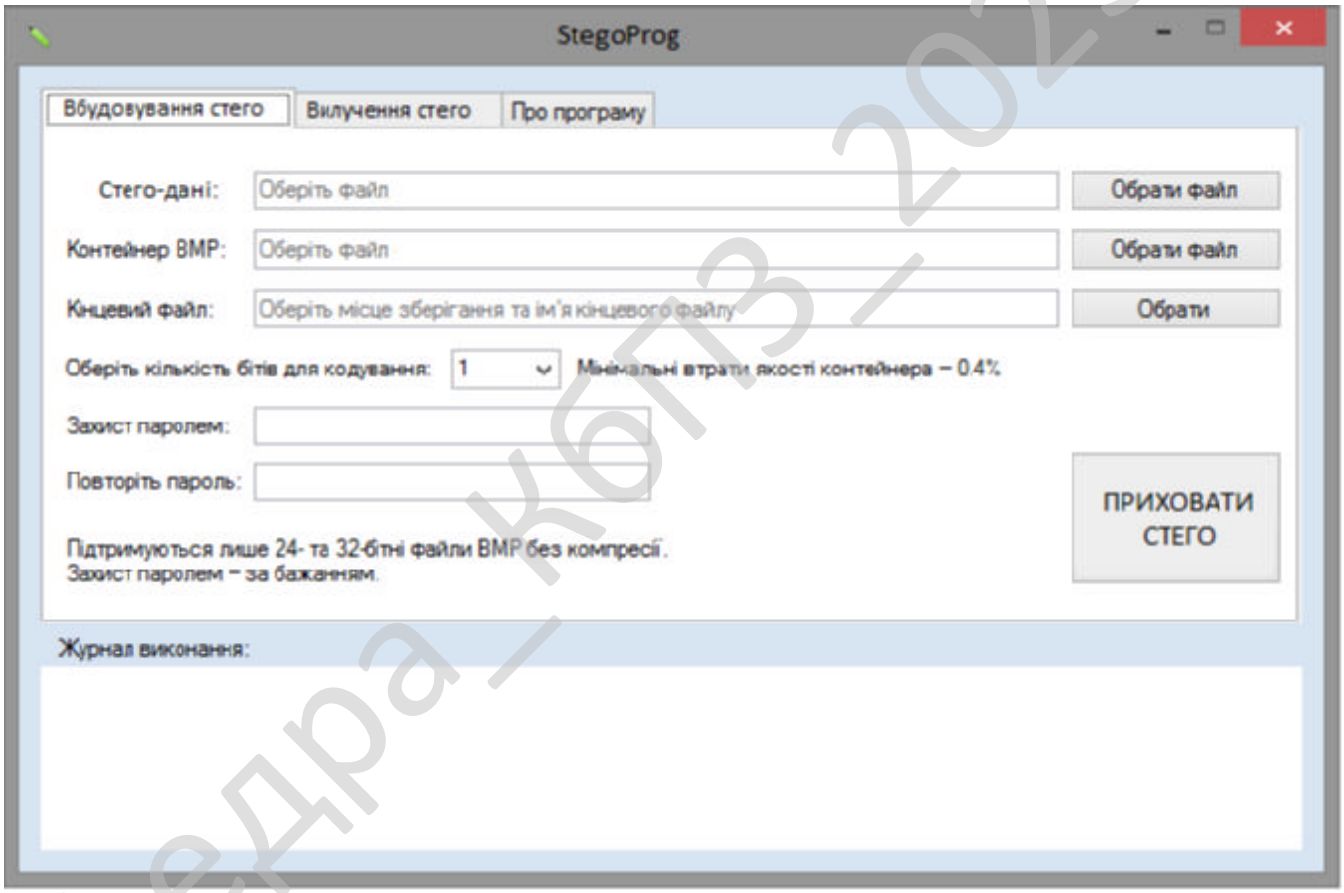

Рисунок 5.1 – Головне вікно програми, вкладка «Прикріплення стего»

Розроблене програмне забезпечення працює з файлами у форматі BMP, що мають 24-х або 32-х бітну розрядність.

Для вбудовування стего у файл слід виконати наступну послідовність кроків (рисунок 5.1):

1. Обрати файл, що мітить стего-дані.

2. Обрати порожній файл-контейнер.

3. Обрати назву вихідного файлу та шлях, куди слід зберегти заповнений стегоповідомленням контейнер.

4. Обрати кількість найменш значущих бітів у пікселях контейнера для вбудовування стегоповідомлення (від 1 до 3 бітів).

5. Ввести стегопароль (за бажанням).

6. Натиснути кнопку «Приховати стего».

Після вказаних дій стего буде вбудовано у файл-контейнер. На рисунку 5.2 зображене головне вікно програми після успішного вбудовування стего. На рисунку видно, що кожна обрана користувачем дія заноситься у Журнал виконання.

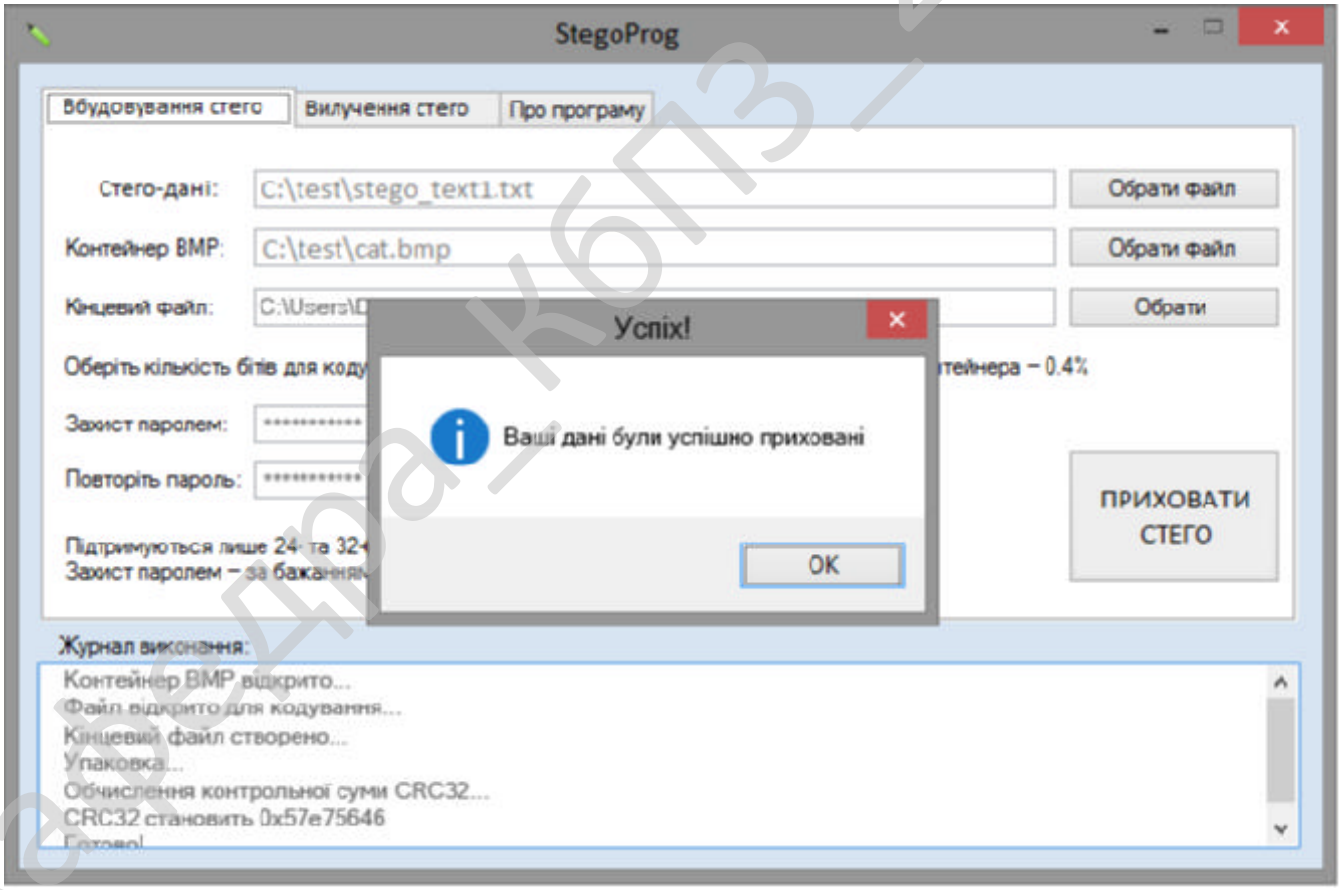

Рисунок 5.2 – Головне вікно програми, вкладка «Прикріплення стего» після успішного вбудовування стего

Для більш надійного збереження стего у файлах-контейнерах в програмі застосовано алгоритм обчислення контрольної суми CRC-32 для перевірки цілісності даних, що дозволяє виправляти частково пошкоджені стего.

Для вилучення стегоповідомлення із файлу слід виконати наступну послідовність кроків:

1. Обрати заповнений файл-контейнер.

2. Обрати назву вихідного файлу та шлях, куди слід зберегти вилучений з файлу-контейнера стего.

3. Обрати кількість найменш значащих бітів у пікселях контейнера, в які було вбудовано стего (від 1 до 3 бітів).

4. Ввести стегопароль (якщо він використовувався при вбудовуванні).

5. Натиснути кнопку «Вилучити прихований стего».

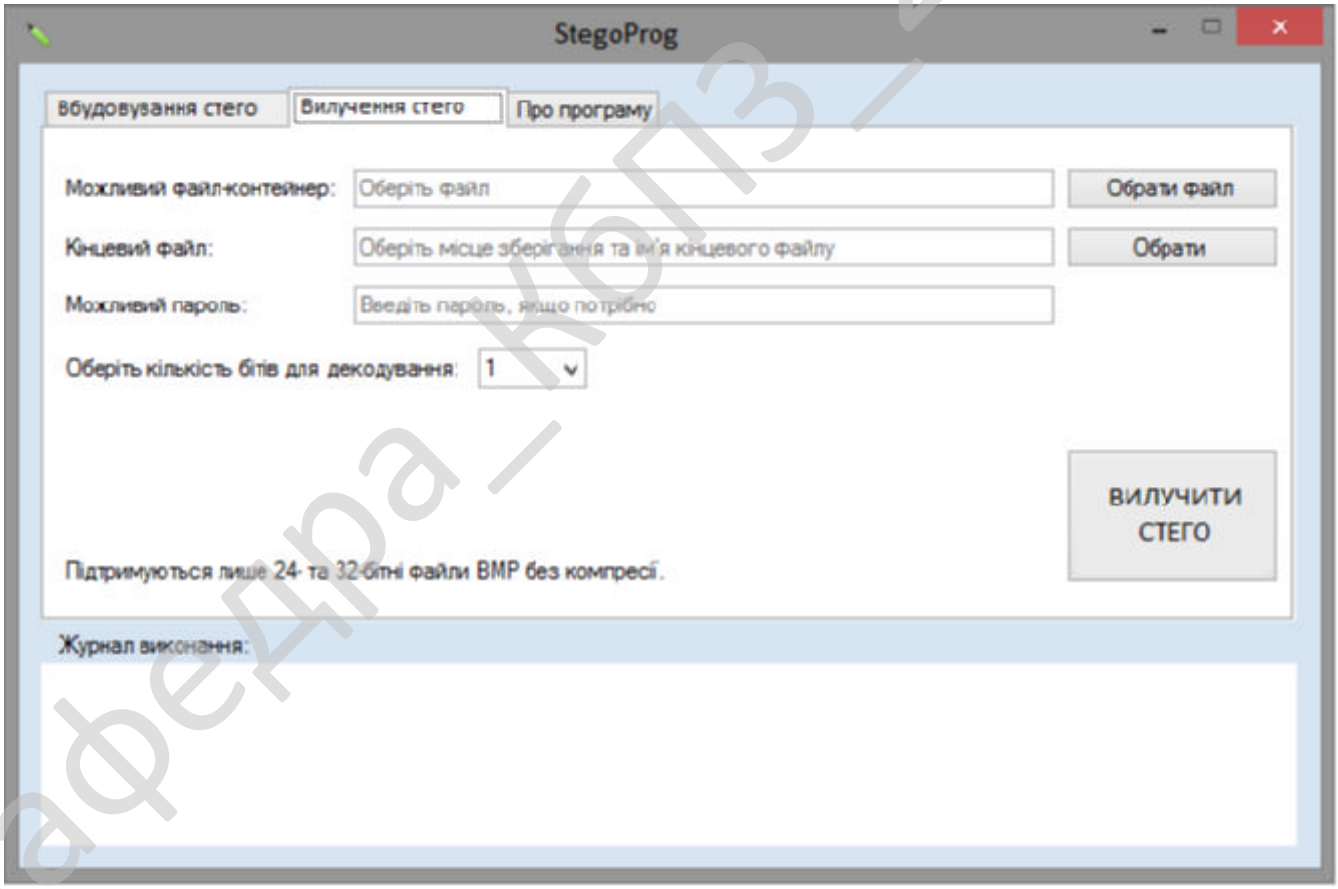

Рисунок 5.3 – Головне вікно програми, вкладка «Вилучення стего»

Після вказаних дій стего буде вилучено із файлу-контейнера. На рисунку

5.4 зображене головне вікно програми після успішного вилучення прихованого стего. Виконані для вилучення стего дії відображені в Журналі виконання. При вилученні відбувається перевірка контрольної суми CRC-32, якщо перевірка проходить успішно, програма повідомляє користувача про те, що вилучений стего цілий, інакше виводиться повідомлення про те, що вилучений стего було пошкоджено.

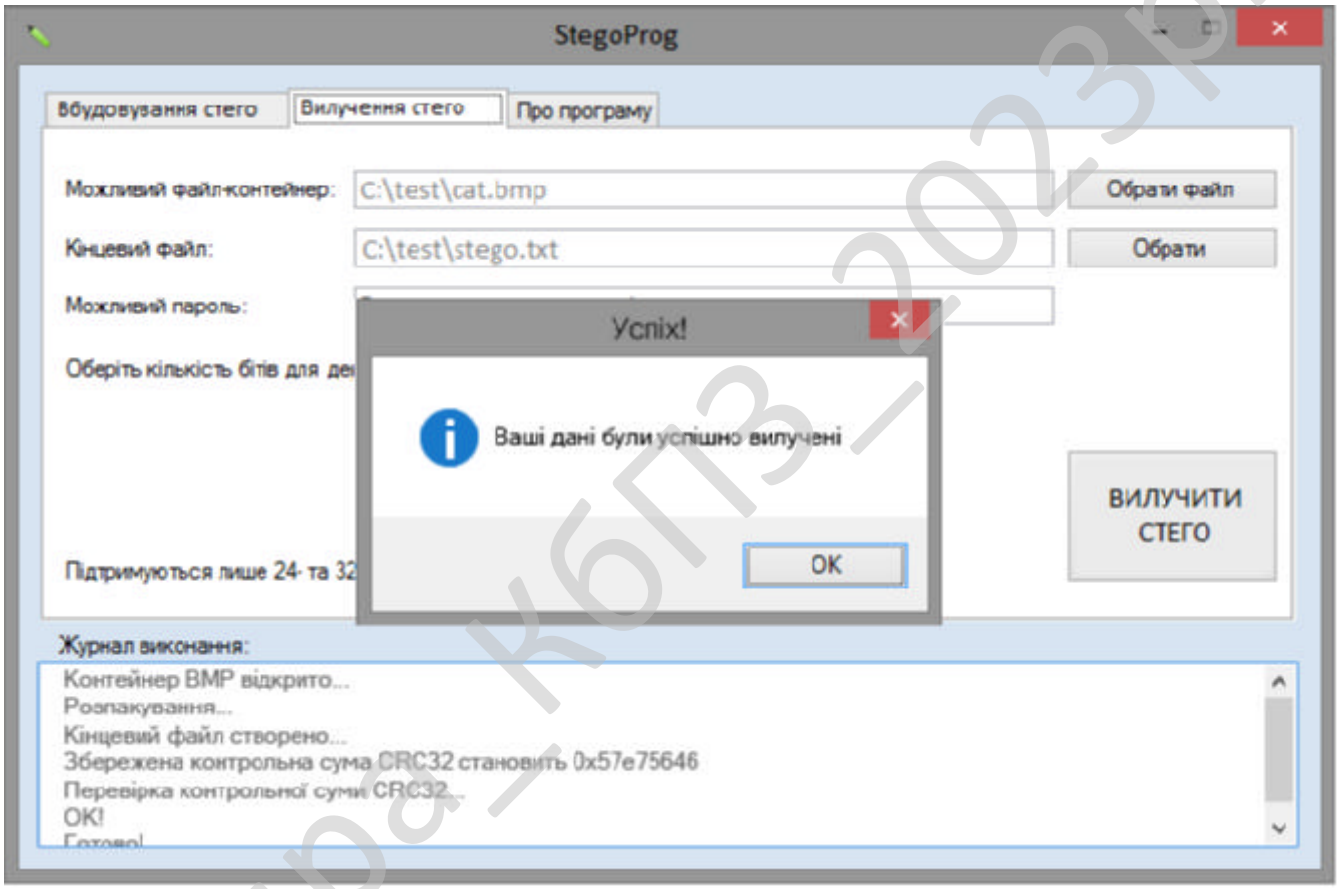

Рисунок 5.4 – Головне вікно програми, вкладка «Вилучення стего» після успішного вилучення стего

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання бакалаврської кваліфікаційної роботи, призначено для системи стеганографічного захисту інформації. Пропонується вбудовувати стеганографічне повідомлення в цифрові зображення.

Виходячи з поставленої у даній роботі мети було досягнуто:

– розглянуто та проаналізовано існуючі на даний момент рішення в області вбудовування інформації в графічні об’єкти методом стеганографії;

– реалізовано задачу вбудовування приховуваної інформації у графічний файл-контейнер методом LSB;

– реалізовано задачу вилучення прихованої інформації з файлу-контейнеру методом LSB;

– створено простий інтуїтивно зрозумілий інтерфейс вітчизняної програми стеганографічного захисту інформації.

При створені програмного забезпечення було використано об’єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем. Програма реалізована середовищі програмування Visual Studio на мові C#. Дана мова програмування дозволяє найбільш ефективно обробляти дані, що використовуються у розробленому програмному забезпеченні. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як наслідок, зменшити витрати на його створення.

Розроблена програма призначена для виконання під управлінням багатозадачної операційної системи Windows 10/11. Даються необхідні рекомендації з роботи з системою. Для підвищення рівня безпеки запропоновано застосовувати алгоритм шифрування заснований на еліптичних кривих.

Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Simmons G. J. The Prisoners Problem and the Subliminal Channel // CRYPTO83 — Advances in Cryptology. New York: Springer, 1984. P. 51–67.

2. Wayner P. Disappearing Cryptography, Second Edition — Information Hiding: Steganography and Watermarking. San Francisco: Morgan Kaufmann, 2002. 413 p.

3. Hassaballah M. (Ed.), Digital Media Steganography: Principles, Algorithms, and Advances, 1st ed. Elsevier, 2020

4. Fridrich J., Goljan M., Soukal D. Perturbed Quantization Steganography // ACM Multimedia & Security J. 2005. V. (11)2. P. 98–107.

5. Fridrich J., Goljan M., Soukal D. Perturbed Quantization Steganography with Wet Paper Codes // ACM Multimedia and Security Workshop. New York: ACM Press, 2004. P. 4–15.

6. Fridrich J., Pevny T., Kodovsky J. Statistically Undetectable JPEG Steganography: Dead Ends, Challenges, and Opportunities // ACM Multimedia and Security Workshop. New York: ACM Press, 2007. P. 3–14.

7. Fridrich J., Goljan M., Lisonek P., Soukal D. Writing on Wet Paper // IEEE Trans. on Sig. Proc., Special Issue on Media Security. 2005. V. 53. P. 3923–3935.

8. Dalal M. and Juneja M., "Steganography and Steganalysis (in digital forensics): a Cybersecurity guide," Multimedia Tools and Applications, vol. 80, no. 4, pp. 5723-5771, Feb. 2021

9. Hussain A.A., Ridzuan A.H. and Mokhtar M.R., "Digital audio steganography: Systematic review, classification, and analysis of the current state of the art," Computer Science Review, vol. 40, p. 100316, Nov. 2020

10. Singh S.K. and Singh S.K., "Social Media and Steganography: Use, Risks and Current Status," IEEE Access, vol. 9, pp. 153656-153665, 2021

11. Conway M. Steganography, Signals Intelligence, and Terrorism //

Knowledge, Technology and Policy. – 2003. – V.16, №2. – P. 45-47.

12. Buckland M., Goldberg E. Emanuel Goldberg and His Knowledge Machine. – Libraries Unlimited. – 2006. – 70 p.

13. An Overview of Steganography for the Computer Forensics Examiner. 2004. [Електронний ресурс]. – Режим доступу: http://www.garykessler.net/library/fsc_stego.html.

14. The Third International Conference on Vailability, Reliability and Security A Statistical Algorithm for Linguistic Steganography Detection Based on Distribution of Words. [Електронний ресурс]. – Режим доступу: http://www.giac.unibel.by/sm_full.aspx?guid=7933.

15. Imai H. Quantum Computation and Information. From Theory to Experiment / H. Imai, M. Hayashi – Springer-Verlag: Berlin, Heidelberg. – 2006. – P. 235.

16. Корченко О.Г., Василіу Є.В., Гнатюк С.О. Сучасні квантові технології захисту інформації // Науково-технічний журнал "Захист інформації". – 2010, № 1. – С. 77-89.

17. Bilal A. Shaw. Quantum steganography and quantum error-correction // University of Southern California. – 2010. – P.137.

18. Гомонай О.В. Лекції з квантової інформатики: Навчальний посібник. – Вінниця : О.Власюк. – 2006. – 146 с.

19. Curty M., Santos D.J. Quantum steganography // In 2nd Bielefeld Workshop on Quantum Information and Complexity. – 2000. – P. 12-14.

20. Ben-Aroya A., Ta-Shma A. On the complexity of approximating the diamond norm – 2009. – V.3. – P. 51-58.

21. Mogos G. Stego Quantum Algorithm // International Symposium on Computer Science and its Applications. – 2008. – P. 187-190.

22. Teleporting an unknown quantum state via dual classical and Einstein-Podolsky Rosen channels / Bennett C.H., Brassard G., Crepeau C., Jozsa R., Peres A., Wootters W.K. – 1993. – 128 p.

23. Bennett C., Wiesner S. Communication via one- and two-particle operators on Einstein-Podolsky-Rosen states. – 1992. – V. 69. – P. 2881-2884.

24. Martin K. Steganographic communication with quantum information / Lecture Notes in Computer Science, – 2008. – 130 p.

25. Koch E., Zhao J. Towards Robust and Hidden Image Copyright Labeling // IEEE Workshop on Nonlinear Signal and Image Processing. 1995. P. 123-132.

26. Wu T., Wu S. Selective encryption and watermarking of MPEG video // International Conference on Image Science, Systems, and Technology. 1997.

27. Langelaar G., van der Lubbe J., Biemond J. Copy Protection for Multimedia Data based on Labeling Techniques // 17th Symposium on Information Theory in the Benelux. 1996.

28. Bassia P., Pitas I., Robust audio watermarking in the time domain // Department of Informatics, University of Tressaloniki.

29. Simmons G. The prisoners problem and the subliminal channel // Proc. Workshop on Communications Security (Crypto483), 1984. P. 51-67.

30. Барсуков В.С. Стеганографические технологии защиты документов, авторских прав и информации // Обзор специальной техники.- 2000.- №2.-С. 31-40.

31. Ramkumar M. Data Hiding in Multimedia. PhD Thesis. New Jersey Institute of Technology, 1999. 72p.

32. Wayner P. Strong Theoretical Steganography //Cryptologia. - 1995. - Vol. XIX, 3 - PP.285-299.

33. Sharp J. Microsoft Visual C# Step by Step, 10th ed. Microsoft Press, 2022

34. Miles R., Exam Ref 70-483 Programming in C#, 2nd ed. Microsoft Press, 2018.

35. Mayo J. Programming the Microsoft Bot Framework: A Multiplatform Approach to Building Chatbots. Microsoft Press, 2017

37. Hall G.M. Adaptive Code: Agile coding with design patterns and SOLID principles, 2nd ed. Microsoft Press, 2017.

38. Miles R. Begin to Code with C#. Microsoft Press, 2016.

39. Whitman M.E. and Mattord H.J. Principles of Information Security, 6th ed. Cengage Learning, 2018.

40. Smith R.E. Elementary Information Security, 2nd ed. Jones & Bartlett Learning, 2015.

41. Kim D. and. Solomon M.G. Fundamentals of Information Systems Security, 3rd ed. Jones & Bartlett Learning, 2018.

42. Andress J. Foundations of Information Security: A Straightforward Introduction. No Starch Press, 2019.

43. Stamp M. Information Security: Principles and Practice, 2nd ed. Wiley-IEEE Press, 2011.

44. Стасюк О. І., Гнатюк С. О., Довгич Н. І., Літош М. С. Сучасні стеганографічні методи захисту інформації. Захист інформації, 2011, 13.1 (50).

45. Коханович Г. Ф., Шевченко О. В., Кінзерявий В. М., Хохлачова Ю. Є. Сучасні методи квантової стеганографії. 2011.

46. Мелешко Є.В., Якименко М.С., Поліщук Л.І. Алгоритми та структури даних: Навчальний посібник для студентів технічних спеціальностей денної та заочної форми навчання. – Кропивницький: Видавець – СПД ФО Лисенко В.Ф., 2019. – 156 с.

47. Мелешко, Є. В. Метод визначення джерела витоку таємної державної інформації на основі стеганографічних маркерів // Інформаційні технології та комп'ютерна інженерія : зб. тез доп. наук.-практ. конф., 4 груд. 2014 р., м. Кіровоград / М-во освіти і науки України, Кіровоград. нац. техн. ун-т . – Кіровоград: КНТУ, 2014. – С. 168.

48. Смірнов О.А. Мелешко Є.В. Дослідження методів стегоаналізу цифрових зображень // Наука і техніка Повітряних Сил Збройних Сил України. – Харків.: ХУПС, 2012. – Вип. 2 (8). – С. 92–99.

49. Мелешко Є. В. Метод вбудовування дворівневих стего в медіафайли для захисту авторських прав // Збірник наукових праць Харківського

університету Повітряних Сил. – Харків: ХУПС, 2013. – Вип. 4 (37). – С. 127-131.

50. Основи захисту інформації : метод. вказ. до викон. лаб. робіт для студ. за спец. 123 “Комп’ютерна інженерія”, 122 “Комп’ютерні науки”/ [уклад. : О. А. Смірнов, Є. В. Мелешко, О. К. Конопліцька-Слободенюк, В. Д. Хох, С. А. Смірнов]; М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т. – Кропивницький : ЦНТУ, 2017. – 53 с.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

					ВКРБ-125.23.0015.00.00.ТЗ		
Вим.	Арк.	№ документа	Підпис	Дата			
Розробив		Пащенко А.В.			Програмне забезпечення системи кібербезпеки для стеганографічного захисту інформації	Літ. Б	Аркуш 1 / Аркушів 6
Перевірив		Мелешко Є.В.					
Н. Контр.		Гермак В.С.				ЦНТУ КБ-19	
Затв.		Смірнов О.А.					

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи кібербезпеки для стеганографічного захисту інформації.

2 Підстава для розробки

Підставою для розробки служить завдання на кваліфікаційну бакалаврську роботу №12-02 від 05.01.2023 року, видане на кафедрі кібербезпеки та програмного забезпечення.

3 Мета та призначення розробки

Метою кваліфікаційної бакалаврської роботи є розробка програмного забезпечення системи кібербезпеки для стеганографічного захисту інформації.

4 Джерела розробки

Джерелом цієї кваліфікаційної бакалаврської роботи є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– систему кібербезпеки для стеганографічного захисту інформації;
– цілісність даних у процесі роботи та при зберіганні;
– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ архітектури IBM PC, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпеченням.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows 10/11.

5.8.1 Обладнання

Комп’ютер Intel Core i7/8 ГБ /1 Tb/ GeForce GT 1030 2GB або сумісні з ним.

5.8.2 Мова програмування

Програму розроблено на мові програмування C#.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у вигляді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД.

7 Перелік документів, що розробляються

– Структурна схема системи.
– Функціональна схема системи.
– Діаграма процесів.
– Блок-схема алгоритму роботи програми.
– Пояснювальна записка.

8 Етапи розробки

8.1 Збір і обробка інформації по темі кваліфікаційної бакалаврської роботи. Постановка задачі на виконання кваліфікаційної бакалаврської роботи (складання ТЗ).

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень кваліфікаційної бакалаврської роботи.

8.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання кваліфікаційної бакалаврської роботи на попередній захист 20.05.2023 р.

11.2 Подання кваліфікаційної бакалаврської роботи на захист 8.06.2023 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи

за першим (бакалаврським) рівнем вищої освіти

_____ Є.В. Мелешко

Програмне забезпечення системи кібербезпеки для стеганографічного захисту інформації

Лістинг програми

Код документу 12

Носій: CD/DVD-диск

Загальна кількість аркушів: 28

Літера: РП

Кропивницький – 2023 року

// Form1.cs – файл головної програми

```
namespace FileTagging
{
      using System;
      using System.Collections.Generic;
      using System.ComponentModel;
      using System.Data;
      using System.Diagnostics;
      using System.Drawing;
      using System.Drawing.Imaging;
      using System.IO;
      using System.Linq;
      using System.Runtime.CompilerServices;
      using System.Text;
      using System.Threading.Tasks;
      using System.Windows.Forms;
      using System.Security.Cryptography;

public class FileTagging : Form

{

    // Поля

    private BackgroundWorker BackgroundWorker;
    private TextBox BaseBitmap;
    private Label BitInfo;
    private Label BitInfo2;
    private Label BitInfo3;
    private TextBox BitmapWithHiddenData;
    private double Capacity;
    private Button ChooseBitmap;
    private Button ChooseFileToHide;
    private Button ChooseOutputFile;
    private IContainer components;
    private TextBox ExecLog;
    private TextBox FileToHide;
    private ComboBox ImageBitsUsage;
    private CheckBox ImageType;
    private Label label1;
    private Label label10;
    private Label label11;
    private Label label2;
    private Label label3;
    private Label label4;
    private Label label5;
    private Label label6;
    private Label label7;
    private Label label8;
    private Label label9;
    private OpenFileDialog OpenFile;
    private TextBox OutputFile;
    private TextBox Pass;
    private TextBox Pass1;
    private TextBox Pass2;
    private Button Run;
    private SaveFileDialog SaveFile;
    private Button SelectBitmapWithHiddenData;
    private TabControl tabControl1;
    private TabPage tabPage1;
    private TabPage tabPage2;
    private Button Unpack;
    private CheckBox UseBase;
    private bool ValidBitmap;
```

```
    // методи

    public FileTagging()
    {
        this.InitializeComponent();
    }

//вбудовування прихованих стегоповідомлень у файли

    private void BackgroundWorker_DoWork(object sender, DoWorkEventArgs e)
    {
        string str = "FileTagging.exe";
        if (((int) e.Argument) == 0)
        {
            string str2 = "\"" + this.FileToHide.Text + "\" \"" +
(this.UseBase.Checked ? "0" : this.BaseBitmap.Text) + "\" \"" +
this.OutputFile.Text + "\" " + this.GetBits().ToString();
            if (this.Pass1.Text.Length > 0)
            {
                str2 = str2 + " \"" + this.Pass1.Text + "\"";
            }

            byte[] encrypted = AesEncryption.Encrypt(str2, aesAlg.Key,
      aesAlg.IV);
            SaveKeyAndIV(keyFile, aesAlg.Key, aesAlg.IV);

            string keyFile = "key.dat";
            str2 = Convert.ToBase64String(encrypted);

            str2 = CRC32.ComputeChecksum(str2);

            try
            {
                MethodInvoker method = null;
                using (Process ProcessObj = new Process())
                {
                    ProcessObj.StartInfo.FileName = str;
                    ProcessObj.StartInfo.Arguments = str2;
                    ProcessObj.StartInfo.UseShellExecute = false;
                    ProcessObj.StartInfo.CreateNoWindow = true;
                    ProcessObj.StartInfo.RedirectStandardOutput = true;
                    ProcessObj.Start();
                    ProcessObj.WaitForExit();


                    if (method == null)
                    {

                        method = (MethodInvoker) (() => (this.ExecLog.Text =
ProcessObj.StandardOutput.ReadToEnd()));
                    }
                    this.ExecLog.Invoke(method);
                    if (ProcessObj.ExitCode != 0)
                    {
                        MessageBox.Show("Сталася помилка приховування стего",
"Помилка", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                    }
                    else
                    {
                        MessageBox.Show("Ваші дані стего були успішно
вбудовані", "Успішно!", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);
                    }
                }
            }


            catch (Exception)
            {
```

```
                MessageBox.Show("Сталася помилка виконання процесу", "Помилка",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
            }
        }


        else
        {
            string str3 = (this.ImageType.Checked ? "1" : "0") + " \"" +
this.BitmapWithHiddenData.Text + "\"";
            if (this.Pass.Text.Length > 0)
            {
                str3 = str3 + " \"" + this.Pass.Text + "\"";
            }

            uint crc = CRC32.ComputeChecksum(str3)
            p = CRC32.CheckChecksum(str3, crc)
            if(p==false)
            {
                  Console.WriteLine("Перевірка CRC-32 виявила порушення
            цілісності даних стего. ");
            }

            string keyFile = "key.dat";
            string decrypted = AesEncryption.Decrypt(encrypted, aesAlg.Key,
      aesAlg.IV);
            str3 = decrypted;
            byte[] loadedKey, loadedIV;
            LoadKeyAndIV(keyFile, out loadedKey, out loadedIV);

            try
            {
                using (Process process = new Process())
                {
                    process.StartInfo.FileName = str;
                    process.StartInfo.Arguments = str3;
                    process.StartInfo.UseShellExecute = false;
                    process.StartInfo.CreateNoWindow = true;
                    process.StartInfo.RedirectStandardOutput = true;
                    process.Start();
                    process.WaitForExit();
                    this.ExecLog.Text = process.StandardOutput.ReadToEnd();
                    if (process.ExitCode != 0)
                    {
                        MessageBox.Show("Сталася помилка вилучення прихованої
стегоінформації", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                    }


                    else
                    {
                        MessageBox.Show("Ваші дані стего були успішно вилучені",
"Успіх!", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);
                    }
                }
            }


            catch (Exception)
            {
                MessageBox.Show("Сталася помилка виконання процесу", "Помилка",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
            }
        }
    }


    private void BackgroundWorker_RunWorkerCompleted(object sender,
RunWorkerCompletedEventArgs e)
```

```
    {
        this.tabControl1.Enabled = true;
    }

//відбір контейнерів

    private void BaseBitmap_TextChanged(object sender, EventArgs e)
    {
        this.ValidBitmap = false;


        try
        {
            this.BitInfo2.Text = "";
            using (Image image = Image.FromFile(this.BaseBitmap.Text))
            {
                if (!image.RawFormat.Equals(ImageFormat.Bmp))
                {
                    this.BitInfo2.Text = "Ви не можете використовувати це
зображення для приховуваня даних стего";
                }


                else
                {
                    int num;
                    if (image.PixelFormat == PixelFormat.Format24bppRgb)
                    {
                        num = 3;
                    }
                    else if (image.PixelFormat == PixelFormat.Format32bppRgb)
                    {
                        num = 4;
                    }
                    else
                    {
                        this.BitInfo2.Text = "Ви не можете використовувати це
зображення для приховуваня даних стего";
                        return;
                    }
                    this.Capacity = (((1.0 * image.Height) * image.Width) * num)
/ 8192.0;
                    this.ValidBitmap = true;
                    this.RefreshDescription();
                }
            }

        }
        catch (Exception)
        {
            this.BitInfo2.Text = "Формат зображення некоректний";
        }
    }

// відкриття контейнера для вбудовування прихованого стего

    private void ChooseBitmap_Click(object sender, EventArgs e)
    {
        this.OpenFile.Title = "Відкриття BMP контейнера...";
        this.OpenFile.Filter = "Bitmaps (*.bmp)|*.bmp|All files (*.*)|*.*";


        if (DialogResult.OK == this.OpenFile.ShowDialog())
        {
            this.BaseBitmap.Text = this.OpenFile.FileName;
        }
    }
```

```
//вибір контейнера для вбудовування прихованого стего

    private void ChooseFileToHide_Click(object sender, EventArgs e)
    {
        this.OpenFile.Title = "Оберіть файл-стего...";
        this.OpenFile.Filter = "All files (*.*)|*.*";
        if (DialogResult.OK == this.OpenFile.ShowDialog())
        {
            this.FileToHide.Text = this.OpenFile.FileName;
        }
    }



    private void ChooseOutputFile_Click(object sender, EventArgs e)
    {
        this.SaveFile.Filter = "Bitmaps (*.bmp)|*.bmp|All files (*.*)|*.*";
        if (DialogResult.OK == this.SaveFile.ShowDialog())
        {
            this.OutputFile.Text = this.SaveFile.FileName;
        }
    }



    protected override void Dispose(bool disposing)
    {
        if (disposing && (this.components != null))
        {
            this.components.Dispose();
        }
        base.Dispose(disposing);
    }


// розрахунок та перевірка CRC-32

public class CRC32
{
    private static uint[] table;

    public static uint ComputeChecksum(string input)
    {
        if (table == null)
        {
            InitializeTable();
        }

        uint crc = 0xffffffff;
        byte[] bytes = Encoding.UTF8.GetBytes(input);

        for (int i = 0; i < bytes.Length; ++i)
        {
            byte index = (byte)(((crc) & 0xff) ^ bytes[i]);
            crc = (uint)((crc >> 8) ^ table[index]);
        }

        return ~crc;
    }

    public static bool CheckChecksum(string input, uint checksum)
    {
        return ComputeChecksum(input) == checksum;
    }

    private static void InitializeTable()
    {
        uint poly = 0xedb88320;
        table = new uint[256];
```

```
        for (uint i = 0; i < table.Length; ++i)
        {
            uint temp = i;
            for (int j = 8; j > 0; --j)
            {
                if ((temp & 1) == 1)
                {
                    temp = (uint)((temp >> 1) ^ poly);
                }
                else
                {
                    temp >>= 1;
                }
            }
            table[i] = temp;
        }
    }
}


// функції шифрування та дешифрування для стегоповідомлень методом AES

using System;
using System.IO;
using System.Security.Cryptography;
using System.Text;

public class AesEncryption
{
    public static byte[] Encrypt(string plainText, byte[] Key, byte[] IV)
    {
        byte[] encrypted;

        using (Aes aesAlg = Aes.Create())
        {
            aesAlg.Key = Key;
            aesAlg.IV = IV;

            ICryptoTransform encryptor = aesAlg.CreateEncryptor(aesAlg.Key,
aesAlg.IV);

            using (MemoryStream msEncrypt = new MemoryStream())
            {
                using (CryptoStream csEncrypt = new CryptoStream(msEncrypt,
encryptor, CryptoStreamMode.Write))
                {
                    using (StreamWriter swEncrypt = new StreamWriter(csEncrypt))
                    {
                        swEncrypt.Write(plainText);
                    }

                    encrypted = msEncrypt.ToArray();
                }
            }
        }

        return encrypted;
    }

    public static string Decrypt(byte[] cipherText, byte[] Key, byte[] IV)
    {
        string plaintext = null;

        using (Aes aesAlg = Aes.Create())
        {
            aesAlg.Key = Key;
            aesAlg.IV = IV;
```

```
            ICryptoTransform decryptor = aesAlg.CreateDecryptor(aesAlg.Key,
aesAlg.IV);

            using (MemoryStream msDecrypt = new MemoryStream(cipherText))
            {
                using (CryptoStream csDecrypt = new CryptoStream(msDecrypt,
decryptor, CryptoStreamMode.Read))
                {
                    using (StreamReader srDecrypt = new StreamReader(csDecrypt))
                    {
                        plaintext = srDecrypt.ReadToEnd();
                    }
                }
            }
        }

        return plaintext;
    }

public class Program
{
    public static void SaveKeyAndIV(string fileName, byte[] key, byte[] iv)
    {
        using (FileStream fs = new FileStream(fileName, FileMode.Create))
        {
            using (BinaryWriter bw = new BinaryWriter(fs))
            {
                bw.Write(key.Length);
                bw.Write(key);
                bw.Write(iv.Length);
                bw.Write(iv);
            }
        }
    }

    public static void LoadKeyAndIV(string fileName, out byte[] key, out byte[]
iv)
    {
        using (FileStream fs = new FileStream(fileName, FileMode.Open))
        {
            using (BinaryReader br = new BinaryReader(fs))
            {
                int keyLength = br.ReadInt32();
                key = br.ReadBytes(keyLength);
                int ivLength = br.ReadInt32();
                iv = br.ReadBytes(ivLength);
            }
        }
    }

    public static void Main()
    {
        string text = "Текст для шифрування";
        string keyFile = "key.dat";

        using (Aes aesAlg = Aes.Create())
        {
            // Зберігання ключа та IV в файл
            SaveKeyAndIV(keyFile, aesAlg.Key, aesAlg.IV);

            // Шифрування тексту
            byte[] encrypted = AesEncryption.Encrypt(text, aesAlg.Key,
aesAlg.IV);
            Console.WriteLine("Шифрований текст: " +
Convert.ToBase64String(encrypted));

            // Завантаження ключа та IV з файлу
            byte[] loadedKey, loadedIV;
            LoadKeyAndIV(keyFile, out loadedKey, out loadedIV);
```

```
            // Дешифрування тексту
            string decrypted = AesEncryption.Decrypt(encrypted, loadedKey,
loadedIV);
            Console.WriteLine("Дешифрований текст: " + decrypted);
        }
    }
}

public static void SaveKeyAndIV(string fileName, byte[] key, byte[] iv)
    {
        using (FileStream fs = new FileStream(fileName, FileMode.Create))
        {
            using (BinaryWriter bw = new BinaryWriter(fs))
            {
                bw.Write(key.Length);
                bw.Write(key);
                bw.Write(iv.Length);
                bw.Write(iv);
            }
        }
    }

    public static void LoadKeyAndIV(string fileName, out byte[] key, out byte[]
iv)
    {
        using (FileStream fs = new FileStream(fileName, FileMode.Open))
        {
            using (BinaryReader br = new BinaryReader(fs))
            {
                int keyLength = br.ReadInt32();
                key = br.ReadBytes(keyLength);
                int ivLength = br.ReadInt32();
                iv = br.ReadBytes(ivLength);
            }
        }
    }
}


    private void FileToHide_TextChanged(object sender, EventArgs e)
    {


        try
        {

            if (Path.GetDirectoryName(this.FileToHide.Text).Length == 0)
            {
                this.OutputFile.Text =
Path.GetFileNameWithoutExtension(this.FileToHide.Text) + ".bmp";
            }
            else
            {
                this.OutputFile.Text =
(Path.GetDirectoryName(this.FileToHide.Text) + @"\" +
Path.GetFileNameWithoutExtension(this.FileToHide.Text) + ".bmp").Replace(@"\\",
@"\");
            }
        }
        catch (Exception)
        {
        }


        try
        {
            FileInfo info = new FileInfo(this.FileToHide.Text);
```

```
            this.BitInfo3.Text = "Розмір обраних файлів становить: " +
Math.Round((double) (((double) info.Length) / 1024.0), 2).ToString() + "
KBytes";

        }
        catch (Exception)
        {
        }
    }

    private int GetBits()
    {
        switch (this.ImageBitsUsage.SelectedIndex)
        {
            case 0:
                return 1;

            case 1:
                return 2;

            case 2:
                return 4;
        }
        return 0;
    }

    private ImageCodecInfo GetEncoder(ImageFormat format)
    {
        foreach (ImageCodecInfo info in ImageCodecInfo.GetImageDecoders())
        {
            if (info.FormatID == format.Guid)
            {
                return info;
            }
        }
        return null;
    }


    private void ImageBitsUsage_SelectedIndexChanged(object sender, EventArgs e)
    {
        switch (this.ImageBitsUsage.SelectedIndex)
        {
            case 0:
                this.BitInfo.Text = "Мінімальні втрати якості контейнера —
0.4%\r\nProbability to hide up to (W*H*3)/(8*1024) KBytes of data for 24-bit
bitmap and up to (W*H*4)/(8*1024) KBytes for 32-bit bitmap.";
                break;

            case 1:
                this.BitInfo.Text = "Невеликі втрати якості контейнера —
1.2%\r\nProbability to hide up to (W*H*3)/(4*1024) KBytes of data for 24-bit
bitmap and up to (W*H*4)/(4*1024) KBytes for 32-bit bitmap.";
                break;

            case 2:
                this.BitInfo.Text = "Видимі втрати якості контейнера —
5.9%\r\nProbability to hide up to (W*H*3)/(2*1024) KBytes of data for 24-bit
bitmap and up to (W*H*4)/(2*1024) KBytes for 32-bit bitmap.";
                break;
        }
        this.RefreshDescription();
    }

    private void FileTagging_Load(object sender, EventArgs e)
    {
        this.ImageBitsUsage.SelectedIndex = 0;
    }
```

```
    private void InitializeComponent()
    {
        this.label1 = new Label();
        this.FileToHide = new TextBox();
        this.OpenFile = new OpenFileDialog();
        this.ChooseFileToHide = new Button();
        this.ChooseBitmap = new Button();
        this.BaseBitmap = new TextBox();
        this.label2 = new Label();
        this.ChooseOutputFile = new Button();
        this.tabControl1 = new TabControl();
        this.tabPage1 = new TabPage();
        this.UseBase = new CheckBox();
        this.label9 = new Label();
        this.Pass2 = new TextBox();
        this.label10 = new Label();
        this.Pass1 = new TextBox();
        this.Run = new Button();
        this.BitInfo3 = new Label();
        this.BitInfo2 = new Label();
        this.BitInfo = new Label();
        this.label5 = new Label();
        this.ImageBitsUsage = new ComboBox();
        this.label4 = new Label();
        this.OutputFile = new TextBox();
        this.label3 = new Label();
        this.tabPage2 = new TabPage();
        this.ImageType = new CheckBox();
        this.label11 = new Label();
        this.Pass = new TextBox();
        this.Unpack = new Button();
        this.label7 = new Label();
        this.label8 = new Label();
        this.SelectBitmapWithHiddenData = new Button();
        this.BitmapWithHiddenData = new TextBox();
        this.SaveFile = new SaveFileDialog();
        this.label6 = new Label();
        this.ExecLog = new TextBox();
        this.BackgroundWorker = new BackgroundWorker();
        this.tabControl1.SuspendLayout();
        this.tabPage1.SuspendLayout();
        this.tabPage2.SuspendLayout();
        base.SuspendLayout();
        this.label1.AutoSize = true;
        this.label1.Location = new Point(6, 15);
        this.label1.Name = "label1";
        this.label1.Size = new Size(0x3d, 13);
        this.label1.TabIndex = 1;
        this.label1.Text = "Стего-дані:";
        this.FileToHide.Location = new Point(0x5c, 12);
        this.FileToHide.Name = "FileToHide";
        this.FileToHide.Size = new Size(0xe5, 20);
        this.FileToHide.TabIndex = 2;
        this.FileToHide.TextChanged += new
EventHandler(this.FileToHide_TextChanged);
        this.OpenFile.Filter = "All files|*.*";
        this.OpenFile.Title = "Оберіть файл-стего...";
        this.ChooseFileToHide.Location = new Point(0x147, 10);
        this.ChooseFileToHide.Name = "ChooseFileToHide";
        this.ChooseFileToHide.Size = new Size(0x19, 0x17);
        this.ChooseFileToHide.TabIndex = 3;
        this.ChooseFileToHide.Text = "...";
        this.ChooseFileToHide.UseVisualStyleBackColor = true;
        this.ChooseFileToHide.Click += new
EventHandler(this.ChooseFileToHide_Click);
        this.ChooseBitmap.Location = new Point(0x147, 0x24);
        this.ChooseBitmap.Name = "ChooseBitmap";
        this.ChooseBitmap.Size = new Size(0x19, 0x17);
```

```
            this.ChooseBitmap.TabIndex = 6;
            this.ChooseBitmap.Text = "...";
            this.ChooseBitmap.UseVisualStyleBackColor = true;
            this.ChooseBitmap.Click += new EventHandler(this.ChooseBitmap_Click);
            this.BaseBitmap.Location = new Point(0x5c, 0x26);
            this.BaseBitmap.Name = "BaseBitmap";
            this.BaseBitmap.Size = new Size(0xe5, 20);
            this.BaseBitmap.TabIndex = 5;
            this.BaseBitmap.TextChanged += new
    EventHandler(this.BaseBitmap_TextChanged);
            this.label2.AutoSize = true;
            this.label2.Location = new Point(6, 0x29);
            this.label2.Name = "label2";
            this.label2.Size = new Size(0x44, 13);
            this.label2.TabIndex = 4;
            this.label2.Text = "BMP контейнер:";
            this.ChooseOutputFile.Location = new Point(0x147, 0x3e);
            this.ChooseOutputFile.Name = "ChooseOutputFile";
            this.ChooseOutputFile.Size = new Size(0x19, 0x17);
            this.ChooseOutputFile.TabIndex = 9;
            this.ChooseOutputFile.Text = "...";
            this.ChooseOutputFile.UseVisualStyleBackColor = true;
            this.ChooseOutputFile.Click += new
    EventHandler(this.ChooseOutputFile_Click);
            this.tabControl1.Controls.Add(this.tabPage1);
            this.tabControl1.Controls.Add(this.tabPage2);
            this.tabControl1.Location = new Point(12, 12);
            this.tabControl1.Name = "tabControl1";
            this.tabControl1.SelectedIndex = 0;
            this.tabControl1.Size = new Size(0x174, 0x196);
            this.tabControl1.TabIndex = 10;
            this.tabPage1.Controls.Add(this.UseBase);
            this.tabPage1.Controls.Add(this.label9);
            this.tabPage1.Controls.Add(this.Pass2);
            this.tabPage1.Controls.Add(this.label10);
            this.tabPage1.Controls.Add(this.Pass1);
            this.tabPage1.Controls.Add(this.Run);
            this.tabPage1.Controls.Add(this.BitInfo3);
            this.tabPage1.Controls.Add(this.BitInfo2);
            this.tabPage1.Controls.Add(this.BitInfo);
            this.tabPage1.Controls.Add(this.label5);
            this.tabPage1.Controls.Add(this.ImageBitsUsage);
            this.tabPage1.Controls.Add(this.label4);
            this.tabPage1.Controls.Add(this.label2);
            this.tabPage1.Controls.Add(this.ChooseOutputFile);
            this.tabPage1.Controls.Add(this.label1);
            this.tabPage1.Controls.Add(this.OutputFile);
            this.tabPage1.Controls.Add(this.FileToHide);
            this.tabPage1.Controls.Add(this.label3);
            this.tabPage1.Controls.Add(this.ChooseFileToHide);
            this.tabPage1.Controls.Add(this.ChooseBitmap);
            this.tabPage1.Controls.Add(this.BaseBitmap);
            this.tabPage1.Location = new Point(4, 0x16);
            this.tabPage1.Name = "tabPage1";
            this.tabPage1.Padding = new Padding(3);
            this.tabPage1.Size = new Size(0x16c, 380);
            this.tabPage1.TabIndex = 0;
            this.tabPage1.Text = "Приховати стего";
            this.tabPage1.UseVisualStyleBackColor = true;
            this.UseBase.AutoSize = true;
            this.UseBase.Location = new Point(9, 0xb8);
            this.UseBase.Name = "UseBase";
            this.UseBase.Size = new Size(0x12f, 0x11);
            this.UseBase.TabIndex = 0x15;
            this.UseBase.UseVisualStyleBackColor = true;
            this.UseBase.CheckedChanged += new
    EventHandler(this.UseBase_CheckedChanged);
            this.label9.AutoSize = true;
            this.label9.Location = new Point(6, 0x9d);
```

```
            this.label9.Name = "label9";
            this.label9.Size = new Size(0x5c, 13);
            this.label9.TabIndex = 0x13;
            this.label9.Text = "Повторіть пароль:";
            this.Pass2.Location = new Point(0x87, 0x9a);
            this.Pass2.Name = "Pass2";
            this.Pass2.PasswordChar = '*';
            this.Pass2.Size = new Size(0xd9, 20);
            this.Pass2.TabIndex = 20;
            this.label10.AutoSize = true;
            this.label10.Location = new Point(6, 0x83);
            this.label10.Name = "label10";
            this.label10.Size = new Size(0x6a, 13);
            this.label10.TabIndex = 0x11;
            this.label10.Text = "Захист паролем:";
            this.Pass1.Location = new Point(0x87, 0x80);
            this.Pass1.Name = "Pass1";
            this.Pass1.PasswordChar = '*';
            this.Pass1.Size = new Size(0xd9, 20);
            this.Pass1.TabIndex = 0x12;
            this.Run.Location = new Point(0x87, 0x15d);
            this.Run.Name = "Run";
            this.Run.Size = new Size(0x5e, 0x17);
            this.Run.TabIndex = 50;
            this.Run.Text = "Приховати стего";
            this.Run.UseVisualStyleBackColor = true;
            this.Run.Click += new EventHandler(this.Run_Click);
            this.BitInfo3.Location = new Point(7, 0x124);
            this.BitInfo3.Name = "BitInfo3";
            this.BitInfo3.Size = new Size(0x15d, 13);
            this.BitInfo3.TabIndex = 15;
            this.BitInfo3.TextAlign = ContentAlignment.MiddleRight;
            this.BitInfo2.Location = new Point(7, 0x115);
            this.BitInfo2.Name = "BitInfo2";
            this.BitInfo2.Size = new Size(0x15d, 13);
            this.BitInfo2.TabIndex = 14;
            this.BitInfo2.TextAlign = ContentAlignment.MiddleRight;
            this.BitInfo.Location = new Point(9, 0xd0);
            this.BitInfo.Name = "BitInfo";
            this.BitInfo.Size = new Size(0x157, 0x36);
            this.BitInfo.TabIndex = 13;
            this.BitInfo.TextAlign = ContentAlignment.MiddleRight;
            this.label5.Location = new Point(8, 0x139);
            this.label5.Name = "label5";
            this.label5.Size = new Size(0x158, 0x1d);
            this.label5.TabIndex = 12;
            this.label5.Text = "Підтримуються лише 24- та 32-бітні файли BMP без
компресії. Захист паролем — за бажанням.";
            this.ImageBitsUsage.DropDownStyle = ComboBoxStyle.DropDownList;
            this.ImageBitsUsage.FormattingEnabled = true;
            this.ImageBitsUsage.Items.AddRange(new object[] { "1", "2", "4" });
            this.ImageBitsUsage.Location = new Point(0x124, 0x5d);
            this.ImageBitsUsage.Name = "ImageBitsUsage";
            this.ImageBitsUsage.Size = new Size(60, 0x15);
            this.ImageBitsUsage.TabIndex = 11;
            this.ImageBitsUsage.SelectedIndexChanged += new
EventHandler(this.ImageBitsUsage_SelectedIndexChanged);
            this.label4.AutoSize = true;
            this.label4.Location = new Point(6, 0x60);
            this.label4.Name = "label4";
            this.label4.Size = new Size(0x112, 13);
            this.label4.TabIndex = 10;
            this.label4.Text = "Оберіть кількість бітів для кодування стего:";
            this.OutputFile.Location = new Point(0x5c, 0x40);
            this.OutputFile.Name = "OutputFile";
            this.OutputFile.Size = new Size(0xe5, 20);
            this.OutputFile.TabIndex = 8;
            this.label3.AutoSize = true;
            this.label3.Location = new Point(6, 0x43);
```

```
            this.label3.Name = "label3";
            this.label3.Size = new Size(0x3a, 13);
            this.label3.TabIndex = 7;
            this.label3.Text = "Кінцевий файл:";
            this.tabPage2.Controls.Add(this.ImageType);
            this.tabPage2.Controls.Add(this.label11);
            this.tabPage2.Controls.Add(this.Pass);
            this.tabPage2.Controls.Add(this.Unpack);
            this.tabPage2.Controls.Add(this.label7);
            this.tabPage2.Controls.Add(this.label8);
            this.tabPage2.Controls.Add(this.SelectBitmapWithHiddenData);
            this.tabPage2.Controls.Add(this.BitmapWithHiddenData);
            this.tabPage2.Location = new Point(4, 0x16);
            this.tabPage2.Name = "tabPage2";
            this.tabPage2.Padding = new Padding(3);
            this.tabPage2.Size = new Size(0x16c, 380);
            this.tabPage2.TabIndex = 1;
            this.tabPage2.Text = "Вилучення стего";
            this.tabPage2.UseVisualStyleBackColor = true;
            this.ImageType.AutoSize = true;
            this.ImageType.Location = new Point(9, 0x41);
            this.ImageType.Name = "ImageType";
            this.ImageType.Size = new Size(0x14d, 0x11);
            this.ImageType.TabIndex = 0x34;
            this.ImageType.UseVisualStyleBackColor = true;
            this.label11.AutoSize = true;
            this.label11.Location = new Point(6, 0x2a);
            this.label11.Name = "label11";
            this.label11.Size = new Size(0x66, 13);
            this.label11.TabIndex = 0x16;
            this.label11.Text = "Можливий пароль:";
            this.Pass.Location = new Point(0x87, 0x27);
            this.Pass.Name = "Pass";
            this.Pass.PasswordChar = '*';
            this.Pass.Size = new Size(0xd9, 20);
            this.Pass.TabIndex = 0x17;
            this.Unpack.Location = new Point(0x7d, 0x15d);
            this.Unpack.Name = "Unpack";
            this.Unpack.Size = new Size(0x72, 0x17);
            this.Unpack.TabIndex = 0x33;
            this.Unpack.Text = "Вилучити прихований стего";
            this.Unpack.UseVisualStyleBackColor = true;
            this.Unpack.Click += new EventHandler(this.Unpack_Click);
            this.label7.AutoSize = true;
            this.label7.Location = new Point(6, 0x146);
            this.label7.Name = "label7";
            this.label7.Size = new Size(0x10d, 13);
            this.label7.TabIndex = 20;
            this.label7.Text = "Підтримуються лише 24- та 32-бітні файли BMP без
компресії.";
            this.label8.AutoSize = true;
            this.label8.Location = new Point(6, 15);
            this.label8.Name = "label8";
            this.label8.Size = new Size(0x7b, 13);
            this.label8.TabIndex = 0x11;
            this.label8.Text = "BMP контейнер з прихованим стего:";
            this.SelectBitmapWithHiddenData.Location = new Point(0x147, 10);
            this.SelectBitmapWithHiddenData.Name = "SelectBitmapWithHiddenData";
            this.SelectBitmapWithHiddenData.Size = new Size(0x19, 0x17);
            this.SelectBitmapWithHiddenData.TabIndex = 0x13;
            this.SelectBitmapWithHiddenData.Text = "...";
            this.SelectBitmapWithHiddenData.UseVisualStyleBackColor = true;
            this.SelectBitmapWithHiddenData.Click += new
EventHandler(this.SelectBitmapWithHiddenData_Click);
            this.BitmapWithHiddenData.Location = new Point(0x87, 12);
            this.BitmapWithHiddenData.Name = "BitmapWithHiddenData";
            this.BitmapWithHiddenData.Size = new Size(0xba, 20);
            this.BitmapWithHiddenData.TabIndex = 0x12;
            this.SaveFile.Title = "Зберегти кынцевий файл як...";
```

```
        this.label6.AutoSize = true;
        this.label6.Location = new Point(9, 0x1a5);
        this.label6.Name = "label6";
        this.label6.Size = new Size(0x4e, 13);
        this.label6.TabIndex = 15;
        this.label6.Text = "Журнал виконання:";
        this.ExecLog.Location = new Point(12, 0x1b5);
        this.ExecLog.Multiline = true;
        this.ExecLog.Name = "ExecLog";
        this.ExecLog.ReadOnly = true;
        this.ExecLog.ScrollBars = ScrollBars.Vertical;
        this.ExecLog.Size = new Size(0x170, 0x65);
        this.ExecLog.TabIndex = 0x10;
        this.BackgroundWorker.DoWork += new
DoWorkEventHandler(this.BackgroundWorker_DoWork);
        this.BackgroundWorker.RunWorkerCompleted += new
RunWorkerCompletedEventHandler(this.BackgroundWorker_RunWorkerCompleted);
        base.AutoScaleDimensions = new SizeF(6f, 13f);
        base.AutoScaleMode = AutoScaleMode.Font;
        base.ClientSize = new Size(0x18c, 550);
        base.Controls.Add(this.ExecLog);
        base.Controls.Add(this.label6);
        base.Controls.Add(this.tabControl1);
        base.FormBorderStyle = FormBorderStyle.FixedSingle;
        base.MaximizeBox = false;
        base.Name = "FileTagging";
        base.StartPosition = FormStartPosition.CenterScreen;
        this.Text = "Paul © 2013";
        base.Load += new EventHandler(this.FileTagging_Load);
        this.tabControl1.ResumeLayout(false);
        this.tabPage1.ResumeLayout(false);
        this.tabPage1.PerformLayout();
        this.tabPage2.ResumeLayout(false);
        this.tabPage2.PerformLayout();
        base.ResumeLayout(false);
        base.PerformLayout();
    }



    private void RefreshDescription()
    {
        if (!this.ValidBitmap)
        {
            this.BitInfo2.Text = "";
        }
        else
        {
            this.BitInfo2.Text = "Ви можете зберегти" + Math.Round((double)
(this.GetBits() * this.Capacity), 2).ToString() + " КБайт прихованих даних";
        }
    }



    private void Run_Click(object sender, EventArgs e)
    {
        this.ExecLog.Text = "";
        if (((this.FileToHide.Text.Length == 0) || (!this.UseBase.Checked &&
(this.BaseBitmap.Text.Length == 0))) || (this.OutputFile.Text.Length == 0))
        {
            MessageBox.Show("Заповніть усі необхідні поля", "Помилка",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
        }
        else if ((this.Pass1.Text.Length > 0) && (this.Pass1.Text !=
this.Pass2.Text))
        {
            MessageBox.Show("Повторіть пароль правильно", "Помилка",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
```

```
        }
        else
        {
            this.BackgroundWorker.RunWorkerAsync(0);
            this.tabControl1.Enabled = false;
        }
    }


// вибір зображення з прихованими даними

    private void SelectBitmapWithHiddenData_Click(object sender, EventArgs e)
    {
        this.OpenFile.Title = "Відкрити BMP контейнер з прихованим стего...";
        this.OpenFile.Filter = "Bitmaps (*.bmp)|*.bmp|All files (*.*)|*.*";
        if (DialogResult.OK == this.OpenFile.ShowDialog())
        {
            this.BitmapWithHiddenData.Text = this.OpenFile.FileName;
        }
    }







// вилучення даних

    private void Unpack_Click(object sender, EventArgs e)
    {

        this.ExecLog.Text = "";

        if (this.BitmapWithHiddenData.Text.Length == 0)
        {

            MessageBox.Show("Заповніть усі необхідні поля", "Помилка",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
        }

        else
        {
            this.BackgroundWorker.RunWorkerAsync(1);
            this.tabControl1.Enabled = false;
        }
    }



    private void UseBase_CheckedChanged(object sender, EventArgs e)
    {

        this.BaseBitmap.Enabled = !this.UseBase.Checked;
        this.ChooseBitmap.Enabled = !this.UseBase.Checked;

        this.label2.Enabled = !this.UseBase.Checked;
        this.label4.Enabled = !this.UseBase.Checked;

        this.ImageBitsUsage.Enabled = !this.UseBase.Checked;
        this.BitInfo.Enabled = !this.UseBase.Checked;

        this.BitInfo2.Enabled = !this.UseBase.Checked;
    }


    // властивості
    public int PropertyTagCompression { get; set; }
}
```

```
internal static class Program
{
    // методи
    [STAThread]
    private static void Main()
    {
        Application.EnableVisualStyles();
        Application.SetCompatibleTextRenderingDefault(false);
        Application.Run(new ImageEncoder.ImageEncoder());
    }
}
}
```

// Form1.Designer.cs – файл конструктора форми головного вікна програми

```
namespace WindowsFormsApplication1
{


 partial class Form1
    {
        /// <summary>
        /// Потрібна змінна конструктора
        /// </summary>
        private System.ComponentModel.IContainer components = null;



         /// <summary>
        /// Звільнити всі використовувані ресурси.
        /// </summary>
        /// <param name="disposing">істинно, якщо керований ресурс повинен бути
видалений; інакше хибно.</param>
        protected override void Dispose(bool disposing)
        {

            if (disposing && (components != null))
            {

                components.Dispose();

            }

            base.Dispose(disposing);
        }



        #region Код, автоматично створений конструктором форм Windows

        /// <summary>
        /// Обов'язковий метод для підтримки конструктора
        /// не можна міняти вміст даного методу за допомогою редактора коду.
        /// </summary>



        private void InitializeComponent()
        {


            this.components = new System.ComponentModel.Container();
            System.ComponentModel.ComponentResourceManager resources = new
System.ComponentModel.ComponentResourceManager(typeof(Form1));
            this.tab1 = new System.Windows.Forms.TabControl();
            this.tabPage1 = new System.Windows.Forms.TabPage();
            this.button4 = new System.Windows.Forms.Button();

            this.label9 = new System.Windows.Forms.Label();
            this.label8 = new System.Windows.Forms.Label();
            this.textBox5 = new System.Windows.Forms.TextBox();
            this.label7 = new System.Windows.Forms.Label();
            this.textBox4 = new System.Windows.Forms.TextBox();
            this.label6 = new System.Windows.Forms.Label();
            this.label5 = new System.Windows.Forms.Label();
            this.comboBox1 = new System.Windows.Forms.ComboBox();
            this.label4 = new System.Windows.Forms.Label();
            this.button3 = new System.Windows.Forms.Button();

            this.textBox3 = new System.Windows.Forms.TextBox();
```

```
            this.label3 = new System.Windows.Forms.Label();
            this.button2 = new System.Windows.Forms.Button();
            this.textBox2 = new System.Windows.Forms.TextBox();
            this.label2 = new System.Windows.Forms.Label();
            this.button1 = new System.Windows.Forms.Button();
            this.textBox1 = new System.Windows.Forms.TextBox();
            this.label1 = new System.Windows.Forms.Label();
            this.tabPage2 = new System.Windows.Forms.TabPage();
            this.label13 = new System.Windows.Forms.Label();
            this.button7 = new System.Windows.Forms.Button();

            this.button5 = new System.Windows.Forms.Button();
            this.textBox7 = new System.Windows.Forms.TextBox();
            this.textBox6 = new System.Windows.Forms.TextBox();
            this.label11 = new System.Windows.Forms.Label();
            this.label10 = new System.Windows.Forms.Label();
            this.label12 = new System.Windows.Forms.Label();
            this.richTextBox1 = new System.Windows.Forms.RichTextBox();
            this.tabPage3 = new System.Windows.Forms.TabPage();
            this.pictureBox1 = new System.Windows.Forms.PictureBox();

            this.label14 = new System.Windows.Forms.Label();
            this.AboutTextBox2 = new System.Windows.Forms.RichTextBox();
            this.form1BindingSource = new
System.Windows.Forms.BindingSource(this.components);
            this.tab1.SuspendLayout();
            this.tabPage1.SuspendLayout();
            this.tabPage2.SuspendLayout();
            this.tabPage3.SuspendLayout();

((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.form1BindingSource)).BeginInit(
);

            this.SuspendLayout();

            //
            // tab1
            //

            this.tab1.Controls.Add(this.tabPage1);
            this.tab1.Controls.Add(this.tabPage2);
            this.tab1.Controls.Add(this.tabPage3);
            this.tab1.Location = new System.Drawing.Point(12, 12);
            this.tab1.Name = "tab1";
            this.tab1.SelectedIndex = 0;
            this.tab1.Size = new System.Drawing.Size(658, 279);
            this.tab1.TabIndex = 0;

            //
            // tabPage1
            //

            this.tabPage1.Controls.Add(this.button4);
            this.tabPage1.Controls.Add(this.label9);
            this.tabPage1.Controls.Add(this.label8);
            this.tabPage1.Controls.Add(this.textBox5);
            this.tabPage1.Controls.Add(this.label7);
            this.tabPage1.Controls.Add(this.textBox4);
            this.tabPage1.Controls.Add(this.label6);
            this.tabPage1.Controls.Add(this.label5);
            this.tabPage1.Controls.Add(this.comboBox1);
            this.tabPage1.Controls.Add(this.label4);
            this.tabPage1.Controls.Add(this.button3);
            this.tabPage1.Controls.Add(this.textBox3);
            this.tabPage1.Controls.Add(this.label3);
            this.tabPage1.Controls.Add(this.button2);
            this.tabPage1.Controls.Add(this.textBox2);
```

```
            this.tabPage1.Controls.Add(this.label2);
            this.tabPage1.Controls.Add(this.button1);
            this.tabPage1.Controls.Add(this.textBox1);
            this.tabPage1.Controls.Add(this.label1);
            this.tabPage1.Location = new System.Drawing.Point(4, 22);
            this.tabPage1.Name = "tabPage1";
            this.tabPage1.Padding = new System.Windows.Forms.Padding(3);
            this.tabPage1.Size = new System.Drawing.Size(650, 253);
            this.tabPage1.TabIndex = 0;
            this.tabPage1.Text = "Прикріплення стего";
            this.tabPage1.UseVisualStyleBackColor = true;

            //
            // button4
            //

            this.button4.Location = new System.Drawing.Point(533, 167);
            this.button4.Name = "button4";
            this.button4.Size = new System.Drawing.Size(111, 70);
            this.button4.TabIndex = 18;
            this.button4.Text = "ПРИХОВАТИ СТЕГО";
            this.button4.UseVisualStyleBackColor = true;

            //
            // label9
            //

            this.label9.AutoSize = true;
            this.label9.Location = new System.Drawing.Point(6, 224);
            this.label9.Name = "label9";
            this.label9.Size = new System.Drawing.Size(171, 13);
            this.label9.TabIndex = 17;
            this.label9.Text = "Захист паролем - за бажанням.";

            //
            // label8
            //

            this.label8.AutoSize = true;
            this.label8.Location = new System.Drawing.Point(6, 211);
            this.label8.Name = "label8";
            this.label8.Size = new System.Drawing.Size(321, 13);
            this.label8.TabIndex = 16;
            this.label8.Text = "Підтримуються лише 24- та 32-бітні файли BMP без
компресії.";

            //
            // textBox5
            //

            this.textBox5.Location = new System.Drawing.Point(107, 173);
            this.textBox5.Name = "textBox5";
            this.textBox5.Size = new System.Drawing.Size(207, 20);
            this.textBox5.TabIndex = 15;

            //
            // label7
            //

            this.label7.AutoSize = true;
            this.label7.Location = new System.Drawing.Point(6, 176);
            this.label7.Name = "label7";
            this.label7.Size = new System.Drawing.Size(99, 13);
            this.label7.TabIndex = 14;
            this.label7.Text = "Повторіть пароль:";

            //
            // textBox4
            //
```

```
            this.textBox4.Location = new System.Drawing.Point(107, 144);
            this.textBox4.Name = "textBox4";
            this.textBox4.Size = new System.Drawing.Size(207, 20);
            this.textBox4.TabIndex = 13;

            //
            // label6
            //

            this.label6.AutoSize = true;
            this.label6.Location = new System.Drawing.Point(6, 147);
            this.label6.Name = "label6";
            this.label6.Size = new System.Drawing.Size(92, 13);
            this.label6.TabIndex = 12;
            this.label6.Text = "Захист паролем:";

            //
            // label5
            //

            this.label5.AutoSize = true;
            this.label5.Location = new System.Drawing.Point(320, 117);
            this.label5.Name = "label5";
            this.label5.Size = new System.Drawing.Size(35, 13);
            this.label5.TabIndex = 11;
            this.label5.Text = "label5";

            //
            // comboBox1
            //

            this.comboBox1.DataBindings.Add(new
System.Windows.Forms.Binding("SelectedItem", this.form1BindingSource, "Cursor",
true));
            this.comboBox1.FormattingEnabled = true;
            this.comboBox1.Items.AddRange(new object[] {
            "1",
            "2",
            "4"});
            this.comboBox1.Location = new System.Drawing.Point(257, 114);
            this.comboBox1.Name = "comboBox1";
            this.comboBox1.Size = new System.Drawing.Size(57, 21);
            this.comboBox1.TabIndex = 10;
            this.comboBox1.Tag = "";
            this.comboBox1.SelectedIndexChanged += new
System.EventHandler(this.comboBox1_SelectedIndexChanged);

            //
            // label4
            //

            this.label4.AutoSize = true;
            this.label4.Location = new System.Drawing.Point(6, 117);
            this.label4.Name = "label4";
            this.label4.Size = new System.Drawing.Size(245, 13);
            this.label4.TabIndex = 9;
            this.label4.Text = "Оберіть кількість бітів для кодування стего:";

            //
            // button3
            //

            this.button3.Location = new System.Drawing.Point(533, 82);
            this.button3.Name = "button3";
            this.button3.Size = new System.Drawing.Size(111, 22);
            this.button3.TabIndex = 8;
            this.button3.Text = "Зверни";
            this.button3.UseVisualStyleBackColor = true;
```

```
            //
            // textBox3
            //

            this.textBox3.ForeColor = System.Drawing.SystemColors.GrayText;
            this.textBox3.Location = new System.Drawing.Point(107, 83);
            this.textBox3.Name = "textBox3";
            this.textBox3.Size = new System.Drawing.Size(420, 20);
            this.textBox3.TabIndex = 7;
            this.textBox3.Text = "Оберіть місце зберігання та ім\'я кінцевого
файлу";

            //
            // label3
            //

            this.label3.AutoSize = true;
            this.label3.Location = new System.Drawing.Point(6, 86);
            this.label3.Name = "label3";
            this.label3.Size = new System.Drawing.Size(84, 13);
            this.label3.TabIndex = 6;
            this.label3.Text = "Кінцевий файл:";

            //
            // button2
            //

            this.button2.Location = new System.Drawing.Point(533, 51);
            this.button2.Name = "button2";
            this.button2.Size = new System.Drawing.Size(111, 22);
            this.button2.TabIndex = 5;
            this.button2.Text = "Зверни файл";
            this.button2.UseVisualStyleBackColor = true;

            //
            // textBox2
            //

            this.textBox2.ForeColor = System.Drawing.SystemColors.GrayText;
            this.textBox2.Location = new System.Drawing.Point(107, 52);
            this.textBox2.Name = "textBox2";
            this.textBox2.Size = new System.Drawing.Size(420, 20);
            this.textBox2.TabIndex = 4;

            this.textBox2.Text = "Оберіть файл формату BMP для вбудовування
стего";

            //
            // label2
            //

            this.label2.AutoSize = true;
            this.label2.Location = new System.Drawing.Point(6, 55);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(90, 13);
            this.label2.TabIndex = 3;
            this.label2.Text = "Контейнер BMP:";

            //
            // button1
            //

            this.button1.Location = new System.Drawing.Point(533, 20);
            this.button1.Name = "button1";
            this.button1.Size = new System.Drawing.Size(111, 22);
            this.button1.TabIndex = 2;
            this.button1.Text = "Зверни файл";
            this.button1.UseVisualStyleBackColor = true;
```

```
            //
            // textBox1
            //

            this.textBox1.ForeColor = System.Drawing.SystemColors.GrayText;
            this.textBox1.Location = new System.Drawing.Point(107, 21);
            this.textBox1.Name = "textBox1";
            this.textBox1.Size = new System.Drawing.Size(420, 20);
            this.textBox1.TabIndex = 1;
            this.textBox1.Text = "Оберіть файл, що буде використовуватись як
стего-дані";

            //
            // label1
            //

            this.label1.AutoSize = true;
            this.label1.Location = new System.Drawing.Point(6, 24);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(72, 13);
            this.label1.TabIndex = 0;
            this.label1.Text = "Стего-Дані:";


            //
            // tabPage2
            //

            this.tabPage2.Controls.Add(this.label13);
            this.tabPage2.Controls.Add(this.button7);
            this.tabPage2.Controls.Add(this.button5);
            this.tabPage2.Controls.Add(this.textBox7);
            this.tabPage2.Controls.Add(this.textBox6);
            this.tabPage2.Controls.Add(this.label11);
            this.tabPage2.Controls.Add(this.label10);
            this.tabPage2.Location = new System.Drawing.Point(4, 22);
            this.tabPage2.Name = "tabPage2";
            this.tabPage2.Padding = new System.Windows.Forms.Padding(3);
            this.tabPage2.Size = new System.Drawing.Size(650, 253);
            this.tabPage2.TabIndex = 1;
            this.tabPage2.Text = "Вилучення стего";
            this.tabPage2.UseVisualStyleBackColor = true;

            //
            // label13
            //

            this.label13.AutoSize = true;
            this.label13.Location = new System.Drawing.Point(6, 224);
            this.label13.Name = "label13";
            this.label13.Size = new System.Drawing.Size(321, 13);
            this.label13.TabIndex = 20;
            this.label13.Text = "Підтримуються лише 24- та 32-бітні файли BMP
без компресії.";

            //
            // button7
            //

            this.button7.Location = new System.Drawing.Point(533, 167);
            this.button7.Name = "button7";
            this.button7.Size = new System.Drawing.Size(111, 70);
            this.button7.TabIndex = 19;
            this.button7.Text = "ВИЛУЧИТИ ПРИХОВАНИЙ СТЕГО";
            this.button7.UseVisualStyleBackColor = true;

            //
            // button5
            //
```

```
            this.button5.Location = new System.Drawing.Point(533, 20);
            this.button5.Name = "button5";
            this.button5.Size = new System.Drawing.Size(111, 22);
            this.button5.TabIndex = 4;
            this.button5.Text = "Зверни файл";
            this.button5.UseVisualStyleBackColor = true;

            //
            // textBox7
            //

            this.textBox7.ForeColor = System.Drawing.SystemColors.GrayText;
            this.textBox7.Location = new System.Drawing.Point(160, 52);
            this.textBox7.Name = "textBox7";
            this.textBox7.Size = new System.Drawing.Size(367, 20);
            this.textBox7.TabIndex = 3;
            this.textBox7.Text = "Введіть пароль, якщо потрібно";

            //
            // textBox6
            //

            this.textBox6.ForeColor = System.Drawing.SystemColors.GrayText;
            this.textBox6.Location = new System.Drawing.Point(160, 21);
            this.textBox6.Name = "textBox6";
            this.textBox6.Size = new System.Drawing.Size(367, 20);
            this.textBox6.TabIndex = 2;
            this.textBox6.Text = "Оберіть файл для перевірки на наявність
стего";

            //
            // label11
            //

            this.label11.AutoSize = true;
            this.label11.Location = new System.Drawing.Point(6, 55);
            this.label11.Name = "label11";
            this.label11.Size = new System.Drawing.Size(102, 13);
            this.label11.TabIndex = 1;
            this.label11.Text = "Можливий пароль:";

            //
            // label10
            //

            this.label10.AutoSize = true;
            this.label10.Location = new System.Drawing.Point(6, 24);
            this.label10.Name = "label10";
            this.label10.Size = new System.Drawing.Size(148, 13);
            this.label10.TabIndex = 0;
            this.label10.Text = "Можливий файл-контейнер:";

            //
            // label12
            //

            this.label12.AutoSize = true;
            this.label12.Location = new System.Drawing.Point(18, 297);
            this.label12.Name = "label12";
            this.label12.Size = new System.Drawing.Size(107, 13);
            this.label12.TabIndex = 1;
            this.label12.Text = "Журнал виконання:";

            //
            // richTextBox1
            //
```

```
            this.richTextBox1.BorderStyle =
System.Windows.Forms.BorderStyle.None;
            this.richTextBox1.ForeColor = System.Drawing.SystemColors.GrayText;
            this.richTextBox1.Location = new System.Drawing.Point(12, 313);
            this.richTextBox1.Name = "richTextBox1";
            this.richTextBox1.Size = new System.Drawing.Size(658, 96);
            this.richTextBox1.TabIndex = 2;
            this.richTextBox1.Text = "";

            //
            // tabPage3
            //

            this.tabPage3.Controls.Add(this.AboutTextBox2);
            this.tabPage3.Controls.Add(this.label14);
            this.tabPage3.Controls.Add(this.pictureBox1);
            this.tabPage3.Location = new System.Drawing.Point(4, 22);
            this.tabPage3.Name = "tabPage3";
            this.tabPage3.Padding = new System.Windows.Forms.Padding(3);
            this.tabPage3.Size = new System.Drawing.Size(650, 253);
            this.tabPage3.TabIndex = 2;
            this.tabPage3.Text = "Про програму";
            this.tabPage3.UseVisualStyleBackColor = true;


            //
            // pictureBox1
            //

            this.pictureBox1.Image =
((System.Drawing.Image)(resources.GetObject("pictureBox1.Image")));
            this.pictureBox1.Location = new System.Drawing.Point(6, 6);
            this.pictureBox1.Name = "pictureBox1";
            this.pictureBox1.Size = new System.Drawing.Size(128, 128);
            this.pictureBox1.TabIndex = 0;
            this.pictureBox1.TabStop = false;

            //
            // label14
            //

            this.label14.AutoSize = true;
            this.label14.Font = new System.Drawing.Font("Microsoft Sans Serif",
11F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point,
((byte)(204)));
            this.label14.Location = new System.Drawing.Point(140, 6);
            this.label14.Name = "label14";
            this.label14.Size = new System.Drawing.Size(122, 18);
            this.label14.TabIndex = 3;
            this.label14.Text = "FileTagging 1.0";
            this.label14.Click += new System.EventHandler(this.label14_Click);

            //
            // AboutTextBox2
            //

            this.AboutTextBox2.AutoWordSelection = true;
            this.AboutTextBox2.BackColor = System.Drawing.Color.White;
            this.AboutTextBox2.BorderStyle =
System.Windows.Forms.BorderStyle.None;
            this.AboutTextBox2.DetectUrls = false;
            this.AboutTextBox2.Font = new System.Drawing.Font("Microsoft Sans
Serif", 8.25F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            this.AboutTextBox2.Location = new System.Drawing.Point(143, 27);
            this.AboutTextBox2.Name = "AboutTextBox2";
            this.AboutTextBox2.ReadOnly = true;
            this.AboutTextBox2.ScrollBars =
System.Windows.Forms.RichTextBoxScrollBars.None;
```

```
            this.AboutTextBox2.ShortcutsEnabled = false;
            this.AboutTextBox2.Size = new System.Drawing.Size(501, 162);
            this.AboutTextBox2.TabIndex = 4;
            this.AboutTextBox2.TabStop = false;
            this.AboutTextBox2.Text = resources.GetString("AboutTextBox2.Text");

            //
            // form1BindingSource
            //

            this.form1BindingSource.DataSource =
typeof(WindowsFormsApplication1.Form1);

            //
            // Form1
            //

            this.AutoScaleDimensions = new System.Drawing.Size(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.BackColor =
System.Drawing.SystemColors.GradientInactiveCaption;
            this.ClientSize = new System.Drawing.Size(682, 421);
            this.Controls.Add(this.richTextBox1);
            this.Controls.Add(this.label12);
            this.Controls.Add(this.tab1);
            this.FormBorderStyle =
System.Windows.Forms.FormBorderStyle.FixedSingle;
            this.Icon =
((System.Drawing.Icon)(resources.GetObject("$this.Icon")));
            this.MaximizeBox = false;
            this.Name = "Form1";
            this.StartPosition =
System.Windows.Forms.FormStartPosition.CenterScreen;
            this.Text = "FileTagging";
            this.tab1.ResumeLayout(false);
            this.tabPage1.ResumeLayout(false);
            this.tabPage1.PerformLayout();
            this.tabPage2.ResumeLayout(false);
            this.tabPage2.PerformLayout();
            this.tabPage3.ResumeLayout(false);
            this.tabPage3.PerformLayout();

((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.form1BindingSource)).EndInit();
            this.ResumeLayout(false);
            this.PerformLayout();

        }

        #endregion

        private System.Windows.Forms.TabControl tab1;
        private System.Windows.Forms.TabPage tabPage1;
        private System.Windows.Forms.TabPage tabPage2;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.Button button1;
        private System.Windows.Forms.TextBox textBox1;
        private System.Windows.Forms.Button button3;
        private System.Windows.Forms.TextBox textBox3;
        private System.Windows.Forms.Label label3;
        private System.Windows.Forms.Button button2;
        private System.Windows.Forms.TextBox textBox2;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.ComboBox comboBox1;
        private System.Windows.Forms.Label label4;
        private System.Windows.Forms.BindingSource form1BindingSource;
        private System.Windows.Forms.Label label5;
        private System.Windows.Forms.TextBox textBox5;
```

```
            private System.Windows.Forms.Label label7;
            private System.Windows.Forms.TextBox textBox4;
            private System.Windows.Forms.Label label6;
            private System.Windows.Forms.Label label9;
            private System.Windows.Forms.Label label8;
            private System.Windows.Forms.Button button4;
            private System.Windows.Forms.Button button5;
            private System.Windows.Forms.TextBox textBox7;
            private System.Windows.Forms.TextBox textBox6;
            private System.Windows.Forms.Label label11;
            private System.Windows.Forms.Label label10;
            private System.Windows.Forms.Label label12;
            private System.Windows.Forms.RichTextBox richTextBox1;
            private System.Windows.Forms.Button button7;
            private System.Windows.Forms.Label label13;
            private System.Windows.Forms.TabPage tabPage3;
            private System.Windows.Forms.PictureBox pictureBox1;
            private System.Windows.Forms.Label label14;
            private System.Windows.Forms.RichTextBox AboutTextBox2;
        }
    }
```

// Resources.Designer.cs – конструктор ресурсів

```
namespace FileTagging.Properties {
    using System;


    /// <summary>
    ///   Клас ресурсу зі строгою типізацією для пошуку локалізованих рядків і
т.д.
    /// </summary>

[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Resources.Tools.
StronglyTypedResourceBuilder", "4.0.0.0")]
    [global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
    [global::System.Runtime.CompilerServices.CompilerGeneratedAttribute()]
    internal class Resources {

        private static global::System.Resources.ResourceManager resourceMan;

        private static global::System.Globalization.CultureInfo resourceCulture;


[global::System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Per
formance", "CA1811:AvoidUncalledPrivateCode")]
        internal Resources() {
        }

        /// <summary>
        ///   Повертає хешований екземпляр ResourceManager, використаний цим
класом.
        /// </summary>

[global::System.ComponentModel.EditorBrowsableAttribute(global::System.Component
Model.EditorBrowsableState.Advanced)]
        internal static global::System.Resources.ResourceManager ResourceManager
{
            get {
                if (object.ReferenceEquals(resourceMan, null)) {
                    global::System.Resources.ResourceManager temp = new
global::System.Resources.ResourceManager("FileTagging.Properties.Resources",
typeof(Resources).Assembly);
                    resourceMan = temp;
                }
                return resourceMan;
            }
        }

        /// <summary>
        ///   Перезаписує властивість CurrentUICulture поточного потоку для всіх
        ///   звертань до ресурсу за допомогою цього класу ресурсу зі строгою
типізацією.
        /// </summary>

[global::System.ComponentModel.EditorBrowsableAttribute(global::System.Component
Model.EditorBrowsableState.Advanced)]
        internal static global::System.Globalization.CultureInfo Culture {
            get {
                return resourceCulture;
            }
            set {
                resourceCulture = value;
            }
        }
    }
}
```