Міністерство освіти і науки України
Одеський національний політехнічний університет
Інститут інформаційної безпеки, радіоелектроніки та телекомунікацій
Кафедра кібербезпеки та програмного забезпечення

Конофольський Василь Васильович
студент групи РЗ-151

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

Модифікація методу хеш-стеганографії, заснованого на передачі послідовності цифрових аудіофайлів

Напрям:
125
Кібербезпека

Керівник:
Зоріло Вікторія Вікторівна,
к.т.н., ст. викл.

Одеса – 2020

Міністерство освіти і науки України
Одеський національний політехнічний університет
Інститут інформаційної безпеки, радіоелектроніки та телекомунікацій
Кафедра кібербезпеки та програмного забезпечення

Рівень вищої освіти: другий (магістерський)
Спеціальність: 125 - Кібербезпека
Освітня програма - КІбербезпека

ЗАТВЕРДЖУЮ
Завідувач. кафедри ІУЗІС

_____ Кобозєва А.А.

«____» _____ 2020р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Конофольського Василя Васильовича, РЗ-151

1. Тема проекту (роботи) Модифікація методу хеш-стеганографії, заснованого на передачі послідовності цифрових аудіофайлів
   Керівник роботи: к.т.н., ст. викл. Зоріло В.В.

   затверджені наказом ректора від „16” листопада 2020р. № 468-в
2. Зміст пояснювальної записки: аналіз відомих засобів для обчисленнястійких хеш-кодів аудіо-файлів, розробка хеш-алгоритму та його адаптація для задачі хеш-стеганографії, модифікація алгоритму хеш-стеганографії на основі проведеного аналізу, розробка програмного інтерфейсу для алгоритмічної реалізації запропонованого методу, охорона праці.

3. Перелік графічного матеріалу: схема вбудовування тексту у контейнер, схема витягу тексту із стеганоконтейнеру, блок-схема для техніки аудіостеганографії з використанням фазового кодування, схема відповідності «символ-аудіо», схема розташування світильників.

4. Консультанти розділів роботи

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Охорона праці та безпека в надзвичайних ситуаціях	Ярова І.А. к.т.н., доцент	05.11.2020	26.11.2020

5. Дата видачі завдання “_____”_____20___ р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання	Примітка
1	*Аналіз літератури з теми квалфікаційної роботи*	01-09-2020	*виконано*
2	*Модифікація методу хеш-стеганографії*	01-10-2020	*виконано*
3	*Розробка програмного інтерфейсу*	16-11-2020	*виконано*
4	*Підготовка тексту роботи*	02-11-2020	*виконано*
5	*Підготовка презентації та доповіді*	18-12-2020	*виконано*
6	*Попередній захист*	01-12-2020	*виконано*
7	*Нормоконтроль, рецензування*	15-12-2020	*виконано*
8	*Занесення роботи в електронний архів*	25-12-2020	*виконано*
9	*Допуск до захисту у завідувача кафедри*	25-12-2020	*виконано*

Здобувач вищої освіти _____ Конофольський В.В.

Керівник роботи _____ Зоріло В.В.

ЗАВДАННЯ

на розробку розділу “Охорона праці та безпека в надзвичайних ситуаціях”

Конофольському Василю Васильовичу, група РЗ-151

Інститут інформаційної безпеки, радіоелектроніки та телекомунікацій

Кафедра кібербезпеки та програмного забезпечення

Тема роботи *Модифікація методу хеш-стеганографії, заснованого на передачі послідовності цифрових аудіофайлів*

Зміст розділу:

1 Аналіз умов праці і вибір основних заходів захисту від небезпечних і шкідливих виробничих факторів.

2 Аналіз техногенних небезпек, вибір заходів і засобів забезпечення безпеки у надзвичайних ситуаціях.

3 Розрахунок штучного освітлення.

Керівник роботи	Консультант з охорони праці
_____ (_____)	_____ (_____)
«____» _____ 2020 р.	«____» _____ 2020 р.

АНОТАЦІЯ

Кваліфікаційна робота на тему «Модифікація методу хеш-стеганографії, заснованого на передачі послідовності цифрових аудіофайлів» на здобуття освітньо-кваліфікаційного рівня “Магістр” з спеціальності 125 – «Кібербезпека» містить 19 рисунків, 2 таблиці, 2 додатків, 50 літературних джерел за переліком посилань. Робота виконана на 74 сторінці загального тексту і 45 сторінках основного тексту.

Метою даної роботи є підвищення ефективності передачі секретного повідомлення шляхом модифікації метода хеш-стеганографії.

У результаті виконання кваліфікаційної роботи розроблено модифікацію методу хеш-стеганографії та створено програмний продукт для кодування та декодування повідомлення із стеганоповідомлення. Проведено аналіз результатів нового методу та порівняння їх із початковим, після чого зроблено висновок, що модифікований метод краще забезпечує стійкість стеганоповідомлення до різних атак.

Результати даної роботи можуть бути використані для захисту текстової інформації при передачі по каналу зв’язку від несанкціонованого її використання сторонніми особами.

СТЕГАНОГРАФІЯ, ХЕШ-КОД, АУДІО-ФАЙЛ, ШВИДКЕ ПЕРЕТВОРЕННЯ ФУР’Є, ХЕШ-СТЕГАНОГРАФІЯ.

ANNOTATION

Qualification work on "Modification of hash-steganography method based on digital audio files sequence transmission" for educational qualification level "Master" from specialty 125 - "Cybersecurity" contains 19 figures, 2 tables, 2 appendix, 50 references according to the list of references. The work is written on 74 pages of general text and 45 pages of the main text.

The purpose of this work is to increase the efficiency of the transmission of secret messages by modifying the method of hash steganography.

As a result of the qualification work a modification of hash steganography method was developed and a software product for encoding and decoding a message with steganopovidolenie was created. The results of the new method were analyzed and compared with the original method, after which it was concluded that the modified method better provides steganopovidolenie resistance to various attacks.

The results of this work can be used to protect textual information during transmission over the communication channel from its unauthorized use by unauthorized persons.

STEGANOGRAPHY, HASH, AUDIO FILES, FAST FOURIER TRANSFORM, HASH STEGANOGRAPHY.

ЗМІСТ

ВСТУП

У сучасному світі, передача величезного обсягу інформації є нормою завдяки великій пропускній спроможності каналів зв'язку. Для передачі секретних повідомлень використовують криптографію, стеганографію та комбінацію тих і інших методів. При використанні криптографії в чистому вигляді зашифровані повідомлення привертають увагу. При використанні стеганографічних методів завжди постає питання про пропускну спроможність сигналу та про стійкість метода до атак. Однак сьогодні є альтарнатива, хеш-стеганографія. Хеш-стеганографія – передача повідомлення у вигляді послідовності цифрових файлів, хеш-коди яких в результаті аналізу формують повідомлення. Таким чином, не треба вбудовувати додаткову інформацію в контейнер. Послідовність контейнерів – це і є інформація.

На даний момент у відкритому друці не знайдено робіт та посилань на використання аудіо-сигналів у хеш-стеганографії. Цифрові зображення – зручний та добре досліджений варіант. Оскільки аудіофайли є одним з основних форматів даних для передачі інформації в останні роки, питання приховування інформації в аудіофайл набуває все більшої актуальності. Існує безліч сервісів, що надають людям можливість прослуховувати і завантажувати аудіофайли різних форматів і якості, а також дедалі популярнішим стає спосіб спілкування голосовими повідомленнями. Усе це робить актуальним дослідження аудіо-файлів для передачі секретних повідомлень.

Мета даної роботи – підвищення ефективності передачі секретного повідомлення шляхом модифікації метода хеш-стеганографії.

Для досягнення поставленої мети потрібно вирішити наступні задачі:

- провести огляд відомих алгоритмів хеш-стеганографії;
- обґрунтувати вибір алгоритму для модифікації;
- реалізувати програмно модифікований алгоритм;
- провести аналіз його ефективності.

Об’єкт дослідження методи отримання хеш-кодів аудіо-файлів.

Предметом дослідження є метод хеш-стеганографії.

Для досягнення поставленої у кваліфікаційній роботі мети, були використані чисельні методи, методи обробки цифрових аудіо-файлів.

Розробка складається з чотирьох розділів. У першому розділі наведений короткий опис предметної області та аналіз різних методів вбудовування додаткової інформації у аудіо-файли. У другому розділі детально розглянуто метод та запропоновано його модифікація алгоритмом. Третій розділ – реалізація програмного продукту, у якому описано програмне середовище програмування, безпосередньо розробку програмного коду та інтерфейс користувача. Четвертий розділ – «Охорона праці», у якому наведений аналіз умов праці і вибір основних заходів виробничої безпеки на робочому місці користувача персонального комп’ютеру та аналіз пожежної безпеки і вибір заходів і засобів пожежної безпеки на робочому місці користувача персонального комп’ютеру.

Практичне значення кваліфікаційної роботи полягає в доведенні запропонованої модифікації хеш-стеганографічного методу до його алгоритмічної реалізації та отриманні високих показників стійкості до різних видів атак.

Публікації.

1 ОГЛЯД АУДІО СТЕГАНОГРАФІЇ

Швидкий розвиток Інтернету та цифрова інформаційна революція викликали серйозні зміни в загальній культурі.

Стеганографія - це мистецтво і наука написання прихованих повідомлень таким чином, щоб ніхто, крім відправника та передбачуваного одержувача, не підозрював про існування повідомлення, форма захисту через неясність. Слово «стеганографія» має грецьке походження і означає «покрите лист», від грецьких слів steganos означає «закритий або захищений», a graphy означає «лист»[1].

Головна мета стеганографії – це вільне спілкування в особовій манері та уникнення підозри у передачі прихованих даних[1,2]. Завдяки цьому зменшується ймовірність того, що у зловмисника виникнуть підозри щодо передачі секретного повідомлення.

Гнучке і просте у використанні програмне забезпечення, і зниження цін на цифрові пристрої (наприклад, портативні програвачі компакт-дисків і MP3, DVD-плеєри, CD- і DVD-рекордери, ноутбуки, КПК) зробили можливим для споживачів з усього світу створювати, редагувати і обмінюватися мультимедійними даними. Широкосмугові Інтернет-з'єднання, що забезпечують практично безпомилкову передачу даних, допомагають людям поширювати великі мультимедійні файли і роблять їх ідентичні цифрові копії [3]. У сучасній системі зв'язку приховування даних є найбільш важливим фактором безпеки мережі. Відправка конфіденційних повідомлень і файлів через Інтернет передається в незахищеною формі, але у кожного є що тримати в секреті. Використовуються три основні методи: шифрування, водяні знаки і стеганографія.

Обкладинки можуть бути різних типів, включаючи файли зображень, аудіо та відео, текст.

Запропоновано та продемонстровано кілька методів приховування аудіо- та відеоданих, як у часовій, так і в частотній областях.

Загальні принципи технології приховування даних, а також термінологія, прийнята на Першому міжнародному семінарі з приховування інформації, Кембридж, Великобританія [4] проілюстровані на рисунку 1.1 та на рисунку 1.2.

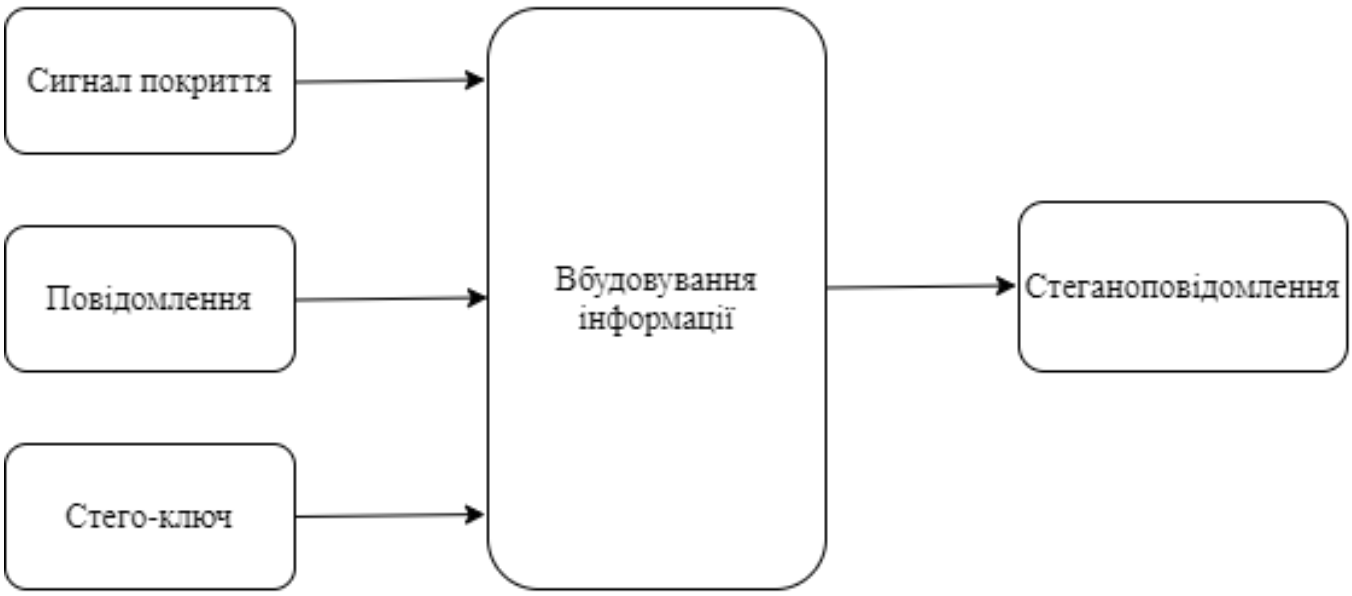

Рисунок 1.1 – Схема вбудовування тексту у контейнер

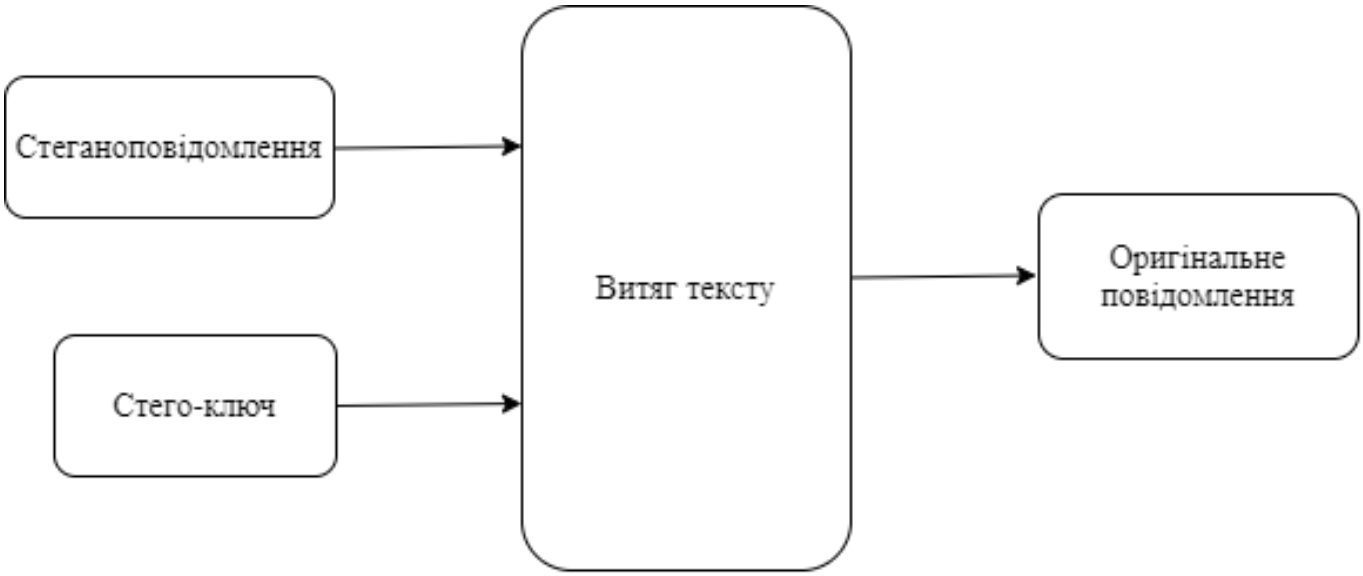

Рисунок 1.2 – Схема витягу тексту із стеганоконтейнеру

Ефективна стеганографічна схема повинна мати наступні бажані характеристики [5]:

1. Секретність: особа не повинна мати змогу витягувати приховані дані із хост-носія без відома відповідного секретного ключа, що використовується в процедурі вилучення.

2. Невідчутність: Носій після вбудовування в приховані дані повинен бути невідмінним від оригінального. Не слід викликати підозри щодо існування прихованих даних у середовищі.

3. Висока ємність: Максимальна довжина прихованого повідомлення, яке можна вбудувати, повинна бути якомога довшою.

4. Опір: приховані дані повинні мати можливість вижити, коли носій середовища маніпулюється, наприклад, за допомогою якоїсь схеми стиснення з втратами.

5. Точне вилучення: вилучення прихованих даних із середовища має бути точним та надійним. В основному, мета стеганографії полягає в забезпеченні таємного спілкування, як криптографія.

1.1 Кодування найменш значущих бітів (LSB)

Кодування найменш значущого біта (LSB) [6-9]: Одним з найперших методів, що вивчались при приховуванні інформації цифрового аудіо (як і інших типів носіїв), є кодування LSB. Кодування найменш значущих бітів (LSB) - це найпростіший спосіб вбудування інформації в цифровий аудіофайл. Замінюючи найменш значущий біт кожної точки вибірки двійковим повідомленням, кодування LSB дозволяє кодувати великий обсяг даних. У кодуванні LSB ідеальна швидкість передачі даних становить 1 кбіт / с на 1 кГц. У деяких реалізаціях кодування LSB, однак, два найменш значущі біти вибірки замінюються двома бітами повідомлення. Це збільшує обсяг даних, які можна закодувати, а також збільшує обсяг отриманих шумів в аудіофайлі[10].

Для отримання секретного повідомлення із звукового файлу, кодованого LSB, приймачеві потрібен доступ до послідовності вибіркових індексів, що використовуються в процесі вбудовування. Зазвичай довжина секретного

повідомлення, яке кодується, менше загальної кількості зразків у звуковому файлі. Тоді слід вирішити, як вибрати підмножину зразків, яка буде містити секретне повідомлення, і повідомити це рішення одержувачу.

Один тривіальний прийом - розпочати на початку звукового файлу та виконати кодування LSB, поки повідомлення не буде повністю вбудовано, а решта зразків залишаться незмінними. Це створює проблему безпеки, однак, оскільки перша частина звукового файлу матиме інші статистичні властивості, ніж друга частина звукового файлу, яка не була змінена. Одним із рішень цієї проблеми є заповнення секретного повідомлення випадковими бітами так, щоб довжина повідомлення дорівнювала загальній кількості вибірок. У цій техніці LSB двійкової послідовності кожного зразка оцифрованого аудіофайлу замінюється двійковим еквівалентом секретного повідомлення. Це, як правило, ефективна техніка у випадках, коли заміна LSB не спричиняє значного погіршення якості. Наприклад, LSB представляє значення 1.

Наприклад, щоб приховати букву "D" (код ASCII 68, а це 01000100) всередині восьми байтів аудіофайлу, встановіть LSB кожного байта, виділивши по одному біту текстових даних за один раз і виправивши LSB (рис. 1.3).

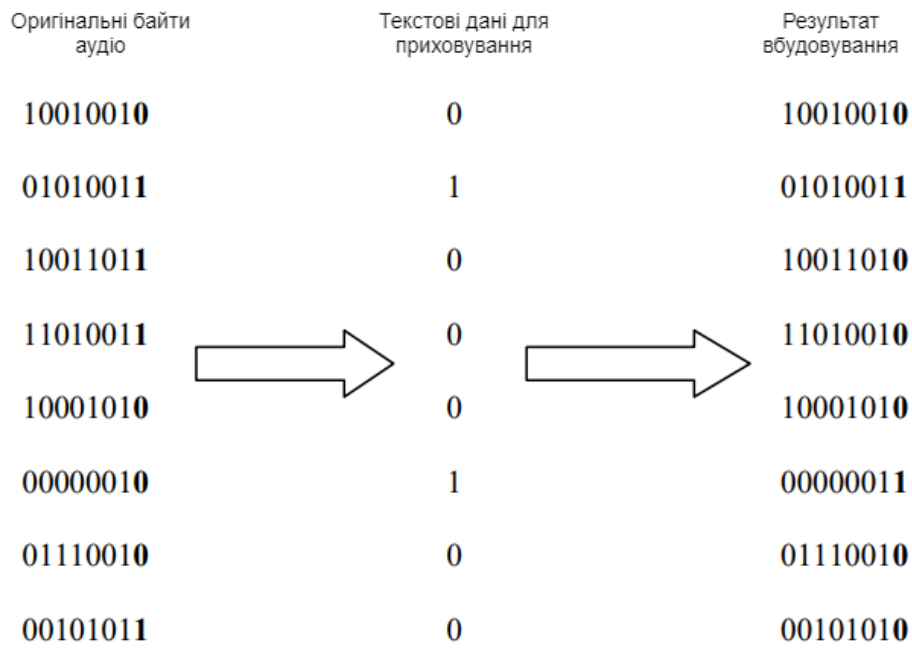

Рисунок 1.3 –Кодування LSB методом

Переваги LSB методу наступні.

1. Розмір файлу-контейнера залишається незмінним.

2. Можливість варіювати пропускну здатність, змінюючи кількість замінних біт.

Недоліком методу LSB являється вразливість до всіх видів афінних атак. Метод може використовуватись тільки при відсутності шумів у каналі зв'язку. Виявлення стеганоповідомлення LSB-кодуванням здійснюється по аномальним характеристикам розподілення значень діапазону молоших бітів відліків цифрового сигналу.

1.2 Кодування за паритетом

Однією з робіт у техніці приховування аудіоданих є техніка кодування за паритетом[8]. При кодуванні по паритету звуковий сигнал розбивається на окремі ділянки вибірки і приховує секретне повідомлення в біті парності кожної області вибірки. Якщо біт парності області вибірки не відповідає секретному біту повідомлення, який потрібно вбудувати, LSB одного з вибірки в області інвертується. Отже, це дасть широкий вибір варіантів, де приховати секретний біт, і збереже зміну сигналу більш непомітною.

Перевірка парності - дуже простий метод для виявлення помилок в переданому пакеті даних. За допомогою даного коду ми не можемо відновити дані, але можемо виявити тільки одиночну помилку.

Таким чином, відправник має більше вибору для кодування секретного біта, і сигнал може бути змінений більш ненав'язливо. За допомогою методу кодування парності перші три біти повідомлення "HEY" кодуються на рисунку 1.4.

Зразок регіону 1.

- Повідомлення біт m1=0.
- Регіони біт парності p1 = 1.
- Перевернути LSB одного із зразків у регіоні.
- p1=m1.

Зразок регіону 2.

- Повідомлення біт m2=1.
- Регіони біт парності p2 = 1.
- p2=m2.

Зразок регіону 3.

- Повідомлення біт m3=0.
- Регіони біт парності p3 = 0.
- p3=m2.

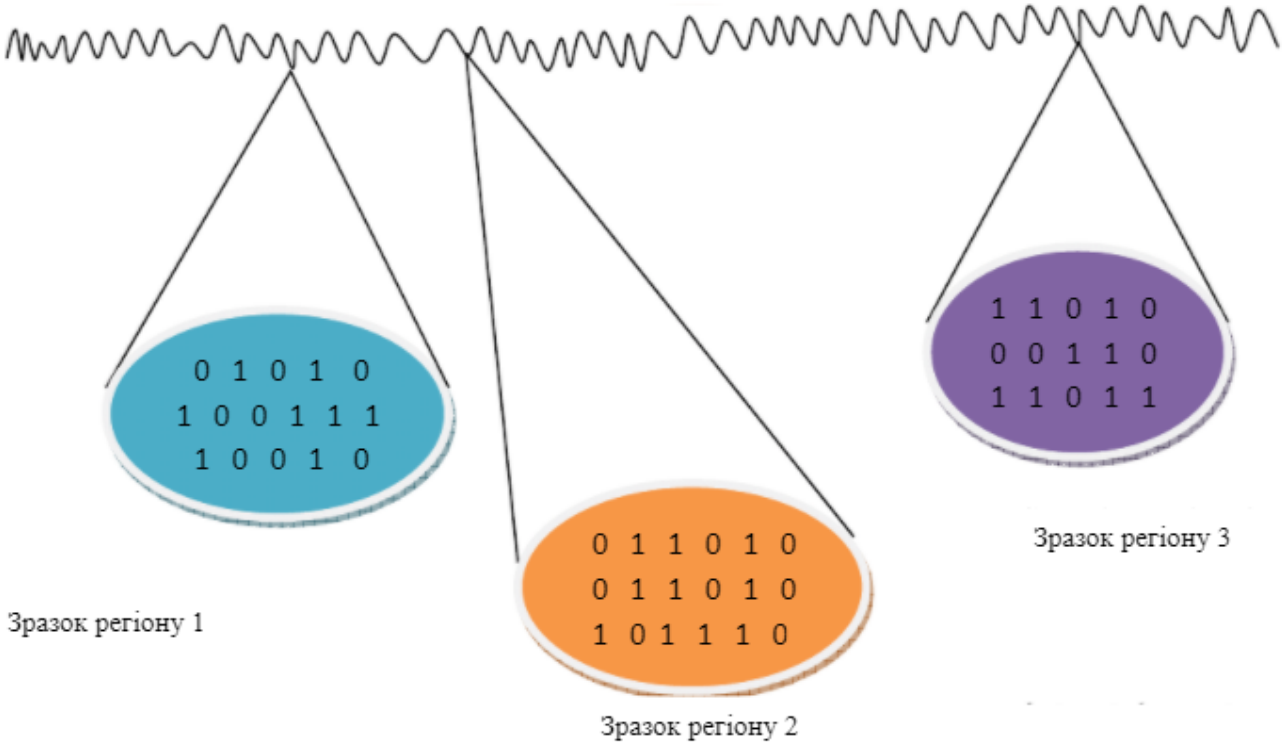

Рисунок 1.4 –Кодування за паритетом

Процес декодування витягує секретне повідомлення шляхом обчислення та вибудовування бітів парності регіонів, що використовуються в процесі кодування. Ще раз, відправник і одержувач можуть використовувати спільний секретний ключ як насіння в генераторі псевдовипадкових чисел, щоб створити той самий набір зразків областей.

Є два основні недоліки, пов’язані з використанням таких методів, як кодування LSB або кодування паритету. Людське вухо дуже чутливе і часто може виявити навіть найменший шматочок шуму, що вводиться у звуковий файл, хоча метод кодування паритету наближається до того, щоб зробити введений шум нечутним. Проте обидва методи мають другий недолік, оскільки вони не є надійними.

Якби звуковий файл, вбудований із секретним повідомленням, використовуючи або кодування LSB, або кодування парності, був перепробований, вбудована інформація буде втрачена [11]. Надійність можна дещо покращити, використовуючи техніку надмірності під час кодування секретного повідомлення. Однак методи надмірності значно знижують швидкість передачі даних.

У кожному пакеті даних є один біт парності, або, так званий, паритетний біт. Цей біт встановлюється під час запису (або відправки) даних, і потім розраховується і порівнюється під час читання (отримання) даних. Він дорівнює сумі по модулю 2 всіх біт даних в пакеті. Тобто число одиниць в пакеті завжди буде парне. Зміна цього біта (наприклад з 0 на 1) повідомляє про виниклу помилку. Приклад процесу перевірки парності

1 Дані 10101 отримують біт парної парності 1, в результаті чого отримується послідовність бітів 101011.

2 Ці дані передаються на інший комп’ютер. При передачі дані пошкоджені, і комп'ютер отримує неправильні дані 100011.

3 Комп’ютер-одержувач обчислює парність: 1 + 0 + 0 + 0 + 1 + 1 = 3. Потім він виконує 3 за модулем 2 (залишок від 3 ділиться на 2), очікуючи результату 0, який би означав, що число парне.

4 Натомість він отримує результат 3 за модулем 2 = 1, що вказує на те, що число непарне. Оскільки він шукає цифри з рівномірною парністю, він просить оригінальний комп’ютер надіслати дані знову.

5 Цього разу дані надходять без помилок: 101011. Комп’ютер, що приймає, обчислює 1 + 0 + 1 + 0 + 1 + 1 = 4. 4 за модулем 2 = 0, що вказує на парність парності. Біт парності вилучається з кінця послідовності, і дані 10101 приймаються.

1.3 Метод фазового кодування

Основна ідея полягає в тому, щоб розділити вихідний аудіопотік або файл контейнер на блоки та вбудувати всю послідовність даних повідомлень у фазовий спектр першого блоку.

Він заснований на заміні обраних фазових складових з вихідного спектра аудіосигналу прихованими даними. Однак, щоб гарантувати нечутні, модифікація фазових компонентів повинна бути невеликою [12].

Одним недоліком методу фазового кодування є значно низька корисна навантаження, оскільки для вбудовування секретного повідомлення використовується лише перший блок. Крім того, повідомлення не розподіляється по контейнеру - це означає, що це локалізовані дані, і, отже, їх легко видалити атакою обрізання. [13]

Фазове кодування пояснюється в наступній процедурі:

1. Оригінальний звуковий сигнал сегментований для вилучення заголовка. Частина решти розділена на менші сегменти, довжина яких дорівнює розміру кодованого повідомлення.

2. Дискретне перетворення Фур'є (DFT) застосовується до кожного сегменту для створення матриці фаз.

3. Принцип вбудовування секретного повідомлення у фазовий вектор першого сегмента сигналу вказаний на рисунку 1.5.

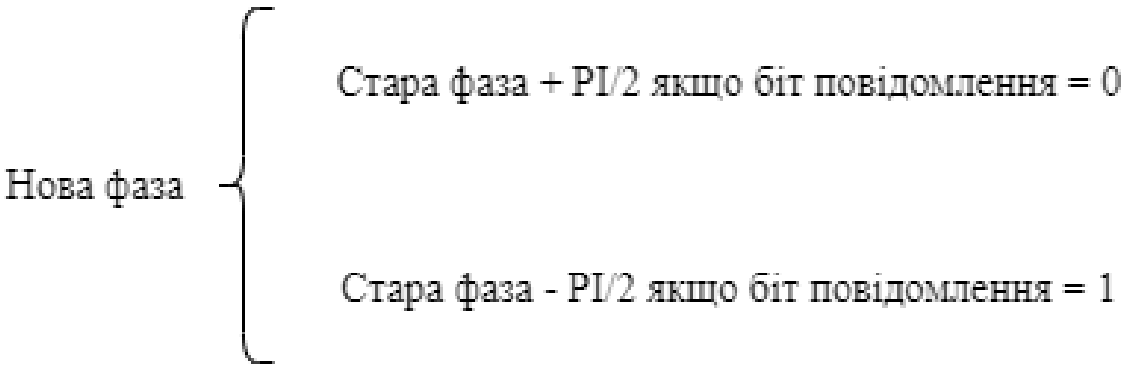

Рисунок 1.5 – Генерація нової фази

4. Нова матриця фаз створюється з використанням нової фази першого сегмента та вихідної матриці фаз.

5. За допомогою нової фазової матриці звуковий сигнал відновлюється шляхом застосування зворотного DFT, а потім конкатенації звукових сегментів з оригінальним заголовком.

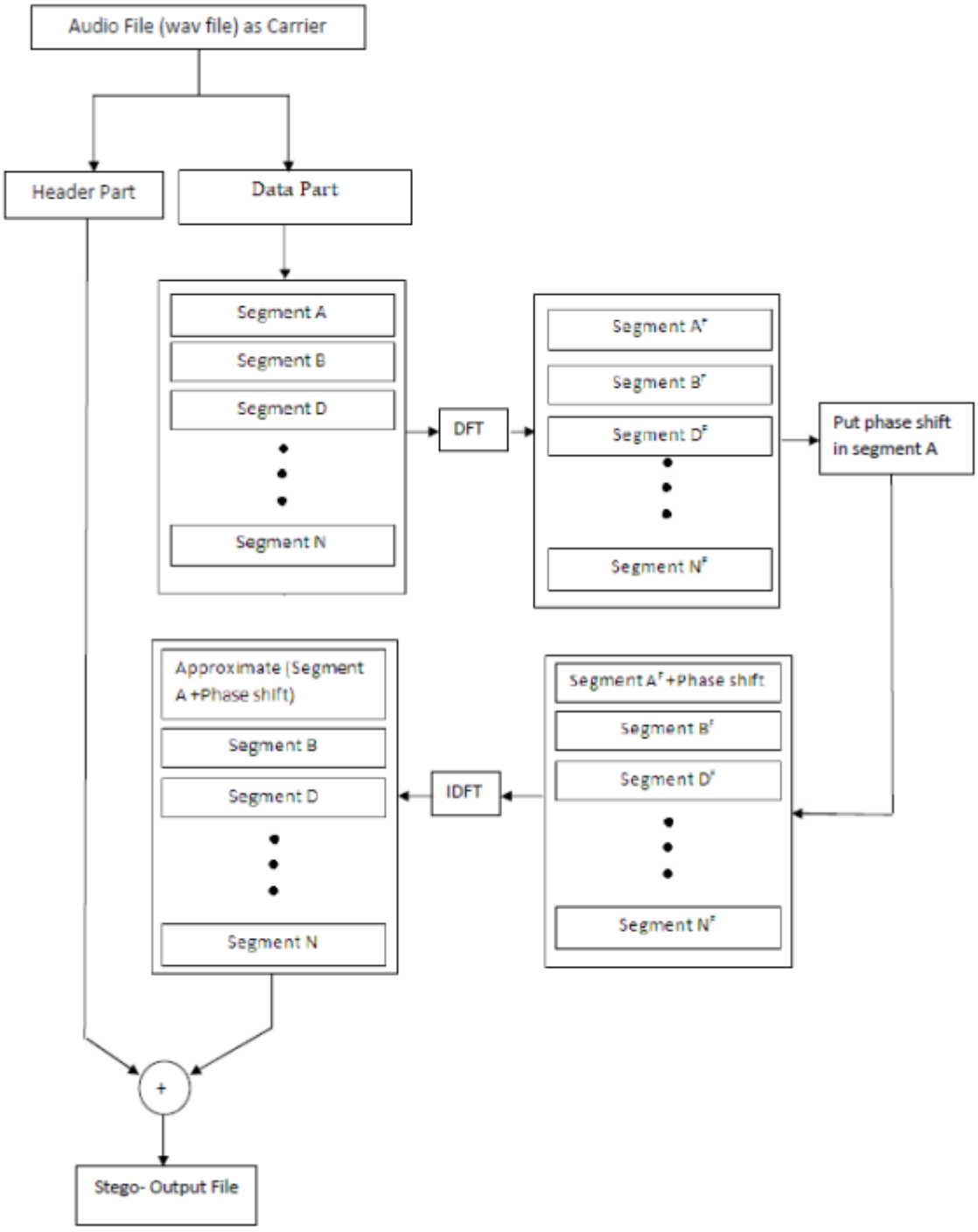

Рисунок 1.6 - Блок-схема для техніки аудіостеганографії з використанням фазового кодування

Фазове кодування ґрунтується на реальності, що на відміну від шумів, компоненти звукової фази непомітні для людського вуха. Замість того, щоб додавати шуми, ця техніка кодує біти секретних даних до фазових зсувів у фазовому спектрі звукового сигналу, отримуючи нечутні кодування з точки зору відношення сигнал / шум.

У фазовому кодуванні фаза початкового аудіосегменту замінюється еталонною фазою, яка представляє дані. Фаза наступних сегментів модифікується назад, щоб підтримувати відносну фазу між сегментами. Фазове кодування, коли можливо застосувати, є одним з найефективніших аудіостеганографічних методів з точки зору відношення сигнал / шум (SNR). Коли різко зміниться фазова залежність між кожною частотною складовою, відбудеться помітна дисперсія фаз. З іншого боку, за умови, що зміна фази досить мала, може бути здійснена нечутна стеганографія.

2 МОДИФІКАЦІЯ МЕТОДУ ХЕШ-СТЕГАНОГРАФІЇ

2.1 Основні відомості

Зростаюче використання Інтернету серед широких мас населення і широка доступність загальнодоступних і приватних цифрових даних спонукали професіоналів галузі та дослідників звернути особливу увагу на захист даних. В даний час використовуються три основні методи: криптографія, водяні знаки і стеганографія. Методи криптографії засновані на відображенні спотвореного змісту повідомлення для неавторизованих людей. У водяних знаках дані приховані, щоб передати деяку інформацію про покривному носії, наприклад про право власності та авторські права. Незважаючи на те, що методи криптографії і водяних знаків важливі для підвищення безпеки даних, підвищений інтерес до вивчення більш досконалих або додаткових нових методів знаходиться в центрі уваги багатьох поточних досліджень. На рисунку 2.1 продемонстровано відмінності і подібності між стеганографії, водяними знаками і криптографією.

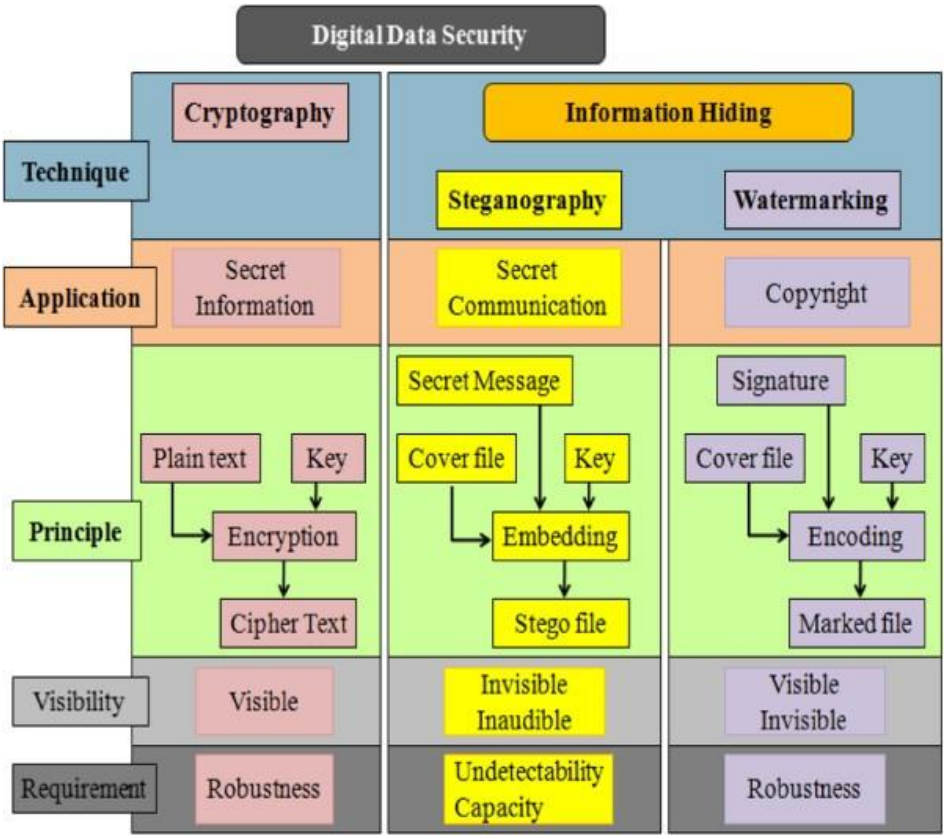

Рисунок 2.1 – Дисципліни безпеки цифрових даних

Термінологія, яка використовується для блоків стеганографії, вперше була введена на першій міжнародній конференції з приховування інформації [11].

Під час передачі конфіденційної інформації зловмисний користувач може незаконно копіювати, модифікувати або знищувати інформацію, що передається в Інтернеті. Як результат, інформаційна безпека стає життєво важливим питанням. Певного ступеня безпеки можна досягти за допомогою криптографічних методів. Але отриманий в результаті шифрувальний текст, який видається незрозумілим, може залучити зловмисників більше, ніж звичайний текст. Також текст шифру легко виявити, а також існує безліч методів його зламу.

Стеганографія - це також техніка захисту інформації, яка приховує секретну інформацію в межах звичайних носіїв інформації, таких як цифрові зображення, аудіо, відео тощо [12]. У стеганографії секретне повідомлення маскується, щоб приховати існування, і робиться «невидимим», таким чином приховуючи той факт, що повідомлення відправляється взагалі, лише відправник та уповноважений одержувач можуть виявити наявність секретної інформації.

Завдяки стеганографії люди можуть надсилати повідомлення, не маючи жодного знання про існування спілкування. Стеганографія може бути рішенням, яке дозволяє надсилати новини та інформацію без цензури та без страху перехоплення та відстеження повідомлень.

Стеганографія стає все більш важливою, оскільки більше людей приєднується до кіберпросторової революції [13]. В аудіостеганографії слабке місце слухової системи людини (HAS) використовується для приховування інформації в аудіо[14]. Оскільки слухова система людини має точнішу, ніж зорова система людини (HVS), аудіостеганографія є більш складною, ніж стеганографія зображення.

Аудіофайл, що використовується для приховування секретного повідомлення, називається аудіо-контейнер, і як тільки секретні дані вбудовуються в аудіо-контейнери, отриманий звук називається стего аудіо. Трьома важливими параметрами при проектуванні методу стеганографії є прозорість сприйняття,

здатність приховувати та надійність. Прихована інформація непомітна, якщо слухач не може виділити між обкладинкою та стего-аудіосигналом.

Потужність приховування - це кількість затемнених даних (у бітах) в аудіосигналі обкладинки. Критерії надійності оцінюються шляхом виживання прихованих даних проти шуму та маніпуляцій із звуковим сигналом [15].

Було досліджено три видатних підходи до вбудовування даних, а саме приховування в тимчасовому домені, в трансформаційних доменах та в кодованому домені. Вихід з цих вейвлет-перетворень забезпечує більший рівень безпеки та надійності, ніж інші підходи.

Сучасні методи стеганографії використовують характеристики цифрових носіїв, використовуючи їх в якості носіїв (прикриттів) для зберігання прихованої інформації.

Обкладинки можуть бути різних типів, включаючи зображення, аудіо, відео, текст і дейтаграми IP. Приклад аудіостеганографії наведений на рисунку 2.2, де в якості контейнера виступає цифровий аудіо-файл.

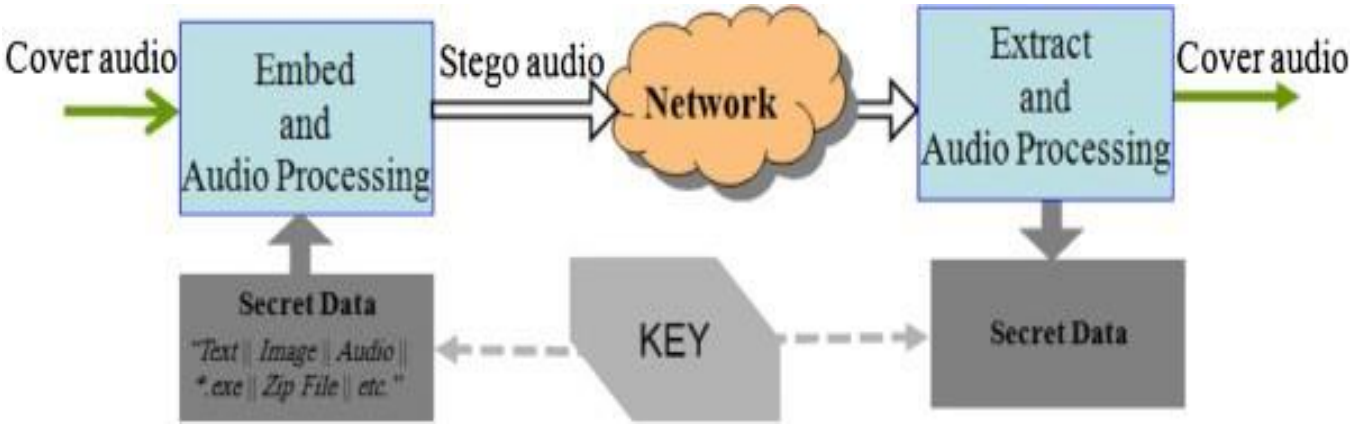

Рисунок 2.2 – Робочий процес аудіо-стеганографії

На якість аудіостеганографічних методів впливають різні функції. Важливість і вплив кожної функції залежать від програми та середовища передачі. Найбільш важливі властивості включають стійкість до шуму, стиску і маніпулювання сигналами, а також безпеку і здатність приховувати приховані дані.

Вимога стійкості тісно пов'язане з додатком, а також є найбільш складним вимогою, яке необхідно виконати в стеганографічної системі, коли торгується з можливістю приховування даних. Як правило, надійність і ємність навряд чи співіснують в одній і тій же стеганографічної системі через дисбаланс компромісів між цими двома критеріями, коли підвищені рівні стійкості призводять до зменшення здатності приховувати дані[16].

Особлива важливість приховування даних в аудіофайл виникає з переважаючого присутності аудіосигналів як інформаційних векторів в нашому людському суспільстві.

Розсудлива практика стеганографії передбачає, що прикриття, що використовується для приховування повідомлень, не повинно викликати підозр у опонентів. Фактично, доступність і популярність аудіофайлів робить їх придатними для зберігання прихованої інформації.

Крім того, велика частина зусиль по стеганалізу більше спрямована на цифрові зображення, залишаючи аудіо стеганаліз щодо невивченим. Приховування даних в аудіофайл особливо складно через чутливість HAS. Однак HAS як і раніше допускає загальні зміни в невеликих діапазонах диференціала. Наприклад, гучні звуки часто заглушають тихі звуки.

Крім того, є деякі загальні спотворення навколишнього середовища, до такої міри, що слухачі в більшості випадків їх ігнорують. Ці властивості спонукали дослідників вивчити можливість використання аудіосигналів в якості носіїв для приховування даних [17 - 22]. Зміни аудіосигналів для цілей впровадження даних можуть вплинути на якість цих сигналів

2.2 Модифікація методу

В роботі [23] описано метод хеш-стеганографії, заснований на використанні цифрових зображень. В основі даного методу лежить використання хеш-кодів зображення для формування послідовності зображень, кожне з яких відповідають певному символу секретного повідомлення.

На даний момент у відкритому друці не знайдено робіт та посилань на використання аудіо-сигналів у хеш-стеганографії. Цифрові зображення – зручний та добре досліджений варіант.

Оскільки аудіофайли є одним з основних форматів даних для передачі інформації в останні роки, питання приховування інформації в аудіофайл набуває все більшої актуальності. Існує безліч сервісів, що надають людям можливість прослуховувати і завантажувати аудіофайли різних форматів і якості, а також дедалі популярнішим стає спосіб спілкування голосовими повідомленнями. Усе це робить актуальним дослідження аудіо-файлів для передачі секретних повідомлень.

Хеш-коди є дуже чутливими до будь-яких змін інформації, тому в роботі описано алгоритм модифікації використовуваних файлів перед обчисленням їх хеш-кодів таким чином, щоб вони були стійкими до атак. Описаний метод взято за основу в даній роботі, проте проведемо дослідження на предмет використання замість цифрових зображень аудіо-файлів.

На відміну від перерахованих у попередньому розділі методів, даний метод не використовує вбудовування додаткової інформації в аудіофайл як контейнер. В даному методі запропоновано використовувати хеш-код (далі хеш), отриманий з аудіофайлів для побудови стеганоповідомлення.

При передачі та конвертації аудіофайлів можливе порушення цілісності їх хеш-коду через можливі втрати якості і шуми. Це може стати перешкодою при встановленні відповідності отриманих файлів та їх хеш-кодів та при відновленні секретного повідомлення.

Запропоновано використовувати низькі частоти аудіо-файлів для побудови хеш-коду, що дозволяє вірно ідентифікувати аудіо-треки навіть за наявності шумів, сторонніх звуків тощо. Розглянемо основні кроки даного алгоритму для використання його при отриманні стійких до атак хеш-кодів аудіо-сигналів.

З мережі Ethernet скачуються аудіо-файли. Для кожного аудіо-файлу використовуються наведені нижче кроки в результаті чого отримаємо унікальний хеш-код.

1.Зчитування та аналіз даних аудіо-зразків у байт-масив. Байт-масив даних у вигляді лінійного графіку зображено на рисунку 2.3.

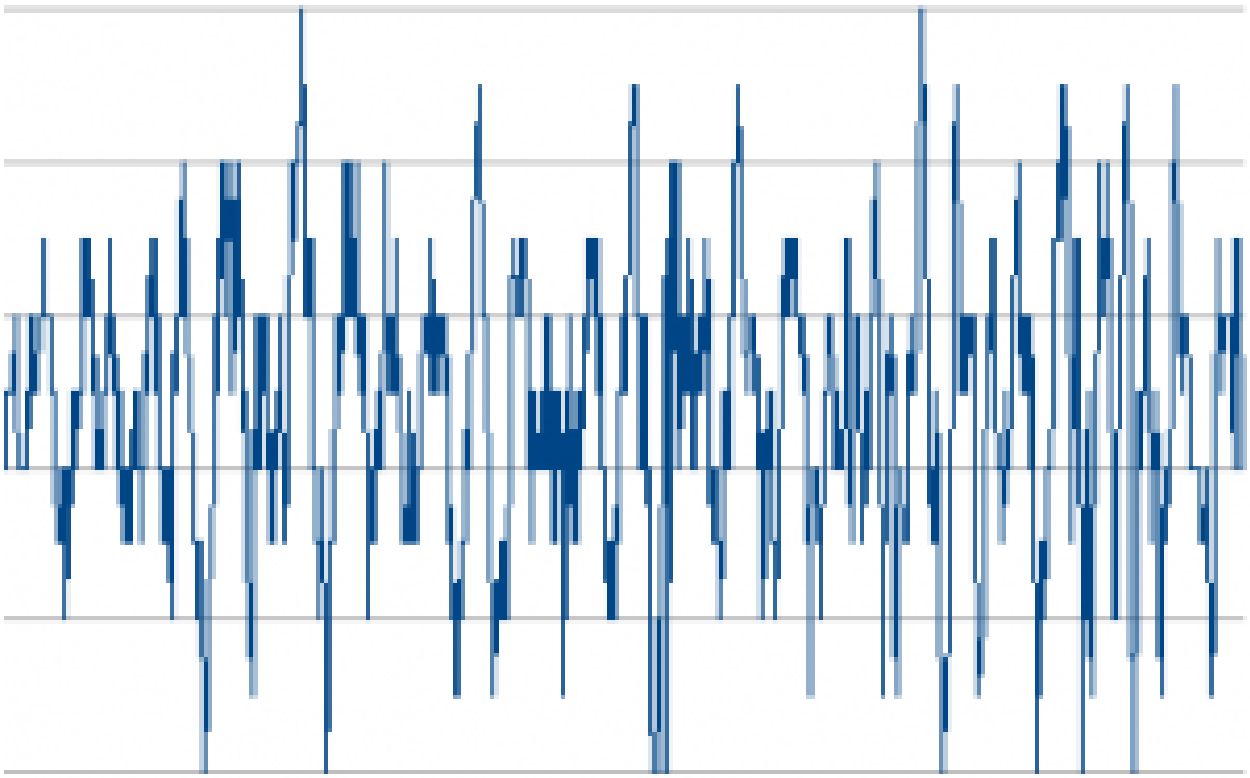

Рисунок 2.3 – Візуалізація отриманих даних

Ці дані зазвичай називають часовою областю. Представлення звукового файлу в такому вигляді дозволяє проводити його аналіз у відношенні до часу [24]. Але отримання цих даних є лише початком, бо далі необхідно з прямих даних часової області отримати спектральний аналіз.

2. Спектральний аналіз. На даному етапі до файлу застосовують швидке перетворення Фур'є, що перетворить дані с часової області в частотну.

Коли дані перетворюються в частотну область, втрачається кожний біт інформації про час. Таким чином, отримується величина всіх частот, але без інформації про те, коли вони з'являються. Для вирішення цієї проблеми дані розподіляються на сегменти та перетворюються у цих сегментах по одному. Завдяки цьому можна визначити дані всіх частот на утвореному сегменті.

3. Отримані дані кодуємо за допомогою хеш-функції MD5, результатом чого на виході отримаємо хеш-код аудіо-файлу(рис. 2.4).

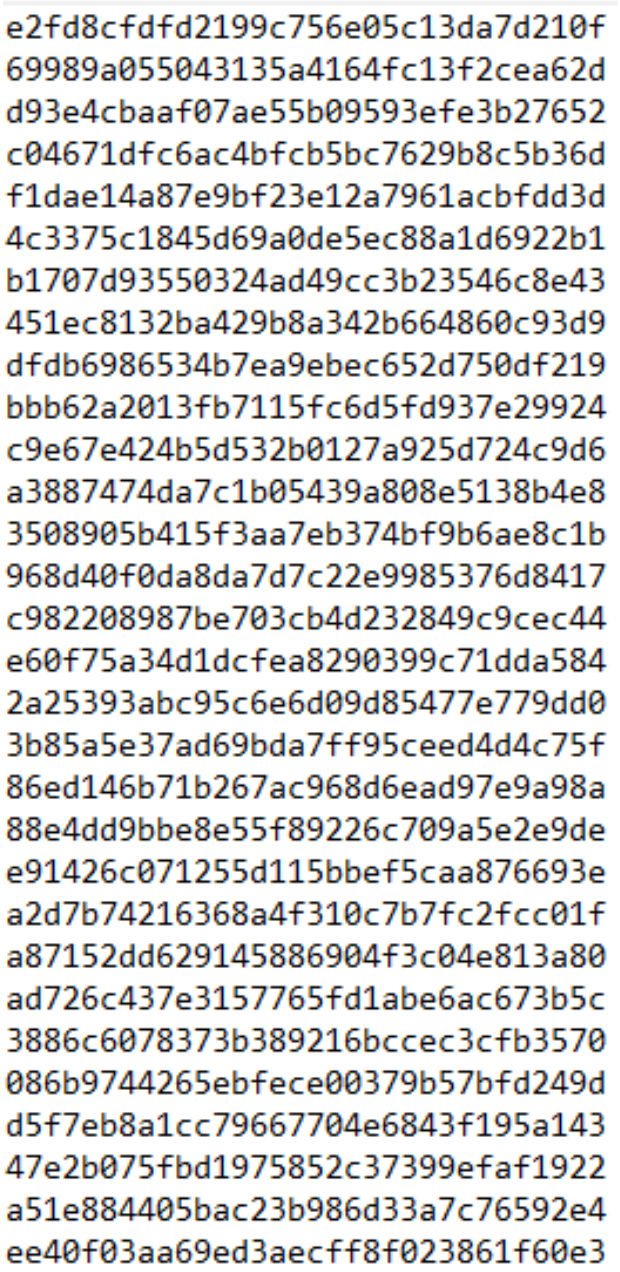

Рисунок 2.4 – Хеш-коди аудіо-файлів

Розглянемо можливість застосування даного алгоритму з метою передати секретне повідомлення через послідовність аудіо-файлів. Створюємо базу даних з аудіофайлів і отриманих з них хеш-кодів. Для бази даних необхідна велика кількість унікальних аудіофайлів. Хеш-код аудіофайлів отримуємо наступним чином.

1. За допомогою вище вказаного алгоритму обчислюємо хеш-коди аудіо-файлів. Одному символу повідомлення поставимо у відповідність декілька зразків

аудіо-сигналів таким чином, щоб ASCII-код символу повідомлення співпадав з двома першими символами хеш-кодів аудіофайлів (рис. 2.5).

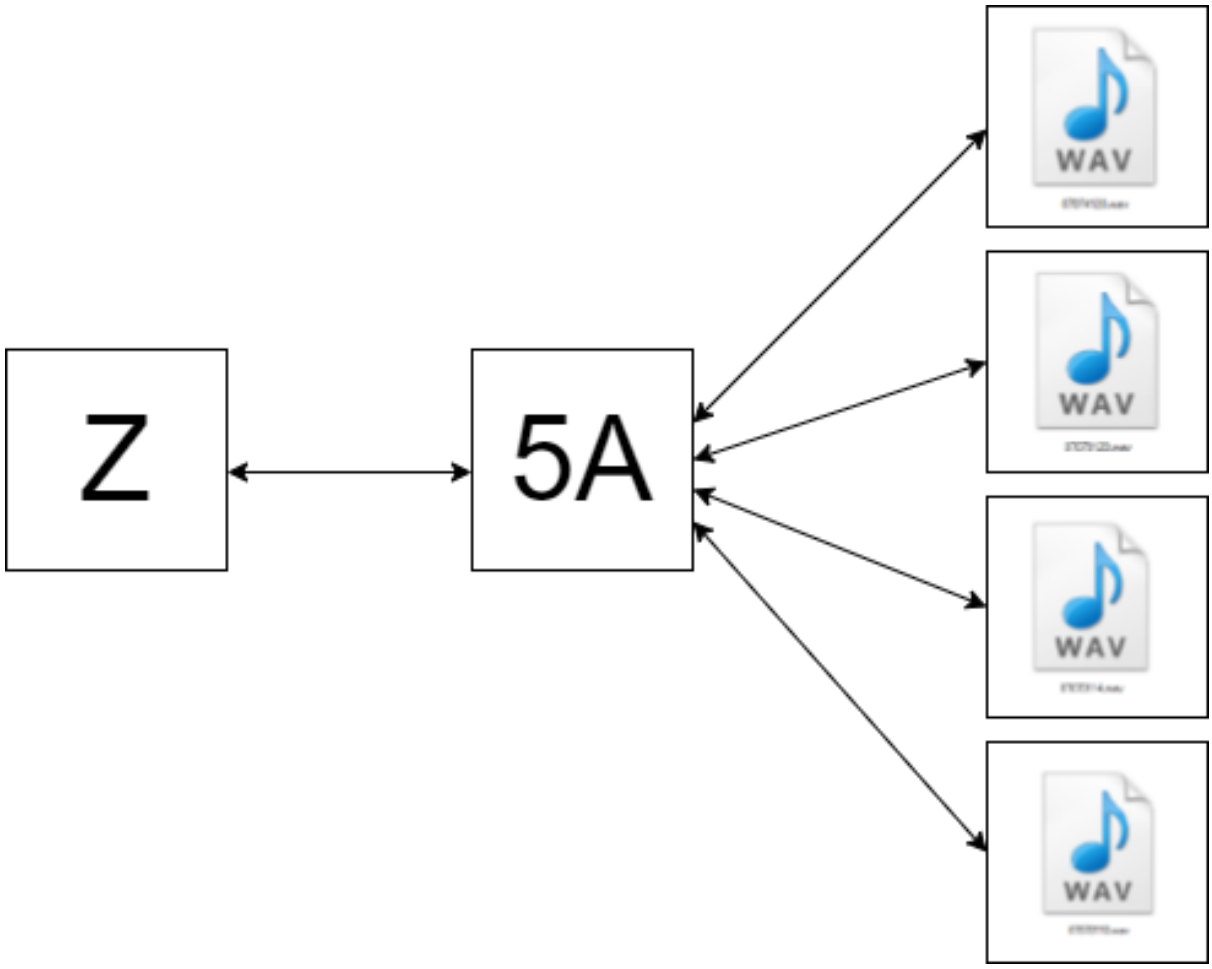

Рисунок 2.5 – Схема відповідності «символ-аудіо»

2. Переводимо наше повідомлення відповідно до таблиці кодування символів ASCII та поділяємо її на рівні частини по два символи.

3. Кожній парі символів ASCII-послідовності ставимо у відповідність аудіо-файл. Отримаємо послідовність аудіо-файлів, що є контейнером стеганоповідомлення та готова до передачі.

Декодування повідомлення відбувається наступним чином. Отримане стеганоповідомлення у вигляді послідовності аудіо-файлів, отримувач зчитую перші два символи з хеш-кодів кожного файлу, в результаті чого отримуємо хеш-код повідомлення. Далі залишається перевести дані в текст для отримання оригінального повідомлення за допомогою таблиці ASCII(рис. 2.6).

DECIMAL	HEXADECIMAL	CHARACTER	DECIMAL	HEXADECIMAL	CHARACTER
64	40	@	96	60	`
65	41	A	97	61	a
66	42	B	98	62	b
67	43	C	99	63	c
68	44	D	100	64	d
69	45	E	101	65	e
70	46	F	102	66	f
71	47	G	103	67	g
72	48	H	104	68	h
73	49	I	105	69	i
74	4A	J	106	6A	j
75	4B	K	107	6B	k
76	4C	L	108	6C	l
77	4D	M	109	6D	m
78	4E	N	110	6E	n

Рисунок 2.6 – Таблиця ASCII для кодування символів

Надійність, безпеку і покриваність - три основних критерії ефективності, які обертаються навколо існуючих методів стеганографії. Щоб класифікувати і оцінити методи з урахуванням цих критеріїв, розглядаються середовище передачі і використовуваним додатком. Наприклад, для прихованої зв'язку потрібен високий рівень стійкості через проходження даних одним з існуючих кодерів, що може сильно вплинути на цілісність переданих даних[25].

На основі розглянутих у попередньому розділі методів сформовано таблицю порівняння даного методу з LSB методом на основі вибраних критеріїв(таблиця 2.1).

Таблиця 2.1 Порівняння методів

Вид атаки	LSB	FFT
Стиснення	52	71
Шуми	68	76

У експерименті спостерігали за стійкістю методів до різних видів атак. Було закодовано повідомлення чисельністю загальною кількістю 100 повідомлень. Перед декодуванням на кожне повідомлення була проведена одна з атак (стиснення або накладення шумів). В результаті спостерігали, що запропонований у даній роботі метод являється більш стійким до атак ніж LSB.

У повному обсязі запропонований метод має наступні переваги:

- Стійкість до таких атак, як стиснення та накладення шумів.
- Повна стійкість проти стеганоаналізу.
- Можливість передавати повідомлення великих об’ємів.
- На даний час не має аналогів для порівняння у сфері хеш-стеганографії.

3 РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

3.1 Середовище програмування

NET Framework - програмна платформа, випущена компанією Microsoft в 2002 році. Основою платформи є загальномовне середовище виконання Common Language Runtime (CLR), яка підходить для різних мов програмування. Функціональні можливості CLR доступні в будь-яких мовах програмування, що використовують цю середу.

C # - об'єктно-орієнтована мова програмування. Розроблено в компанії Microsoft як мова розробки додатків для платформи Microsoft.NET Framework і згодом був стандартизований як ECMA-334 і ISO / IEC 23270.

Особливістю C # є те, що він розроблявся як мова програмування прикладного рівня для CLR і, як такий, залежить, перш за все, від можливостей самої CLR. Виконуючого середовища CLR надає C #, як і всім іншим. NET-орієнтованим мовам, багато можливостей.

Вибір мови C # обумовлений його відносною простотою у вивченні, високою розширюваністю системи, тобто в C # можна імпортувати класи і об'єкти з інших програм.

Microsoft Visual Studio - лінійка продуктів компанії Майкрософт, що включають інтегроване середовище розробки програмного забезпечення і ряд інших інструментальних засобів.

Дані продукти дозволяють розробляти як консольні додатки, так і додатки з графічним інтерфейсом, в тому числі з підтримкою технології Windows Forms, а також веб-сайти, веб-додатки, веб-служби як в рідному, так і в керованому кодах для всіх платформ, підтримуваних Microsoft Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone.NET Compact Framework і Microsoft Silverlight.

Visual Studio включає в себе редактор вихідного коду з підтримкою технології IntelliSense і можливістю найпростішого рефакторінга коду. Вбудований відладчик може працювати як відладчик рівня вихідного коду, так і як відладчик

машинного рівня. Решта вбудовуються інструменти включають в себе редактор форм для спрощення створення графічного інтерфейсу додатку, веб-редактор, дизайнер класів і дизайнер схеми бази даних.

Visual Studio дозволяє створювати і підключати сторонні додатки (плагіни) для розширення функціональності практично на кожному рівні, включаючи додавання підтримки систем контролю версій вихідного коду (як наприклад, Subversion і Visual SourceSafe), додавання нових наборів інструментів (наприклад, для редагування і візуального проектування коду на предметно-орієнтованих мовах програмування або інструментів для інших аспектів процесу розробки програмного забезпечення (наприклад, клієнт Team Explorer для роботи з Team Foundation Server).

Головними причинами вибору продукту Visual Studio в даному проекті є:

- інтегрований мову C #;
- проста реалізація спільних завдань;
- технологія IntelliSense, що дозволяє прискорити кодування, оскільки зменшується кількість тексту, що набирається на клавіатурі, а також зменшується необхідність звертатися до зовнішньої документації;
- розширення Resharper, яке передбачає синтаксичний аналіз коду, додаткові кошти автозаповнення і підсвічування синтаксису;
- Entity Framework - об'єктно-орієнтована технологія доступу до даних, інтегріреумая в лінійку продуктів Visual Studio починаючи з Visual Studio 2008 Service Pack 1.

3.2 Розробка програмного продукту

Розробка програмного продукту насамперед починається безпосередньо з реалізації алгоритму отримання хеш-коду аудіо-файлу. Після завантаження аудіо-файлів у систему, кожен файл підлягає обробці для отримання індивідуального хеш-коду.

Так як дані аудіо-файлу записуються у часовій області, необхідно їх конвертувати у частотну. Для цього масив даних обробляється за допомогою швидкого перетворення Фур'є, код даної функції наведений нижче.

```
public static Complex[] fft(Complex[] x)
        {
            Complex[] X;
            int N = x.Length;
            if (N == 2)
            {

                X = new Complex[2];
                X[0] = x[0] + x[1];
                X[1] = x[0] - x[1];

            }
            else
            {
                Complex[] x_even = new Complex[N / 2];
                Complex[] x_odd = new Complex[N / 2];
                for (int i = 0; i < N / 2; i++)
                {
                    x_even[i] = x[2 * i];
                    x_odd[i] = x[2 * i + 1];
                }
                Complex[] X_even = fft(x_even);
                Complex[] X_odd = fft(x_odd);
                X = new Complex[N];
                for (int i = 0; i < N / 2; i++)
                {
                    X[i] = X_even[i] + w(i, N) * X_odd[i];
                    X[i + N / 2] = X_even[i] - w(i, N) *
X_odd[i];
                }
            }
            return X;
}
```

Отримані дані кодуємо за допомогою хеш-функції MD5. В результаті отримаємо хеш-код аудіо-файлу. Коли закінчимо розрахунки, отримаємо базу даних хеш-кодів усіх використаних аудіо-файлів. Приклади хеш-кодів наведені на рисунку 3.1.

Рисунок 3.1 – Хеш-коди аудіо-файлів

Для полегшення використання даних, хеш-коди записуються в .txt файл для подальшого використання.

Далі програмуємо алгоритм кодування повідомлення та підбір відповідних аудіо-файлів. Користувач вводить повідомлення, програма зчитує та шифрує його за допомогою таблиці кодування ASCII. В результаті отримаємо ASCII-послідовність, а саме хеш-код повідомлення. Далі розбиваємо по два символи та записуємо у масив для наступного етапу кодування.

На кожен елемент масиву необхідно знайти файл з відповідним хеш-кодом. Для першого елементу повідомлення у заздалегідь створеній базі даних хеш-кодів шукаємо аудіо-файл у якого початок хеш-коду співпадаємо с нашим елементом. Підібраний файл відправляється у нову папку. Якщо відібраний файл вже використовувався в даному повідомлені, то пошук продовжується, в іншому випадку береться попередній файл. Програмна реалізація пошуку відповідного аудіо-файлу наведена нижче.

```
string                          targetPath                          =
"C:\\Users\\Администратор\\Desktop\\encMes";
                if (Directory.Exists(targetPath))
                {
                    Directory.Delete(targetPath, true);
                }
                Directory.CreateDirectory(targetPath);
                int NI = 1;
                string[] odd = new string[mas.Length];
                for (int i = 0; i < mas.Length; i++)
                {
                    int j = 0;
                    int count = 0;
                    for (j = 0; j < audioGEO.Length; j++)
                    {
                        if   (Convert.ToString(mas[i])   ==
audioHash[j])
                        {
                            string          newName          =
Convert.ToString(NI).PadLeft(2, '0');
                            string fileName = audioGEO[j];
                            int A = 0;
                            for (int b = 0; b < mas.Length;
b++)
                            {
                                if (odd[b] == fileName)
                                {
                                    A = 1;
                                    count = b;
                                    break;
                                }
                            }
                            odd[i] = fileName;
                            if (A != 1)
                            {

Directory.CreateDirectory(targetPath);
                                File.Copy(fileName,
"C:\\Users\\Администратор\\Desktop\\encMes\\0" + newName +
".wav");
                                NI++;
                                break;
                            }
                        }
```

```
                        else if (j == audioGEO.Length - 1)
{
                            string newName =
Convert.ToString(NI).PadLeft(2, '0');
                            string fileName = odd[count];

Directory.CreateDirectory(targetPath);
                            File.Copy(fileName,
"C:\\Users\\Администратор\\Desktop\\encMes\\0" + newName +
".wav");
                            NI++;
                            break;
                        }
                    }
                }

Process.Start("C:\\Users\\Администратор\\Desktop\\encMes",
"-p");
```

Після завершення кодування повідомлення відкривається вікно із відібраними аудіо-файлами, готовими для подальшого використання.

Кінцевим етапом являється декодування стеганоповідомлення. Адресат повинен вказати шлях до папки, в якій знаходяться послідовність аудіо-файлів.

```
public void LoadFiles_Click(object sender, EventArgs e)
        {
            using (var dialog = new FolderBrowserDialog())
                if (dialog.ShowDialog() == DialogResult.OK)
                    path = dialog.SelectedPath;
            }
```

За допомогою алгоритму визначаються хеш-коди аудіо-файлів. В масив записуються перші два символи з кожного хеш-коду, отримуємо ASCII-послідовність. Далі декодуємо повідомлення та виводимо результат на екран.

Детально ознайомитись з кодом програмного продукту можна у додатку А Код програмного продукту.

3.3 Програмний інтерфейс

В інтерфейс даної програми були включені такі елементи управління: 6 функціональних кнопок(Encode, Decode, Load Folder, Clear Text, Update Base,

Discrete Fourier transform), 1 спливаюче вікно, 1 текстове поле вводу та 1 текстове поле виводу. Інтерфейс програмного продукту зображено на рисунку 3.2.

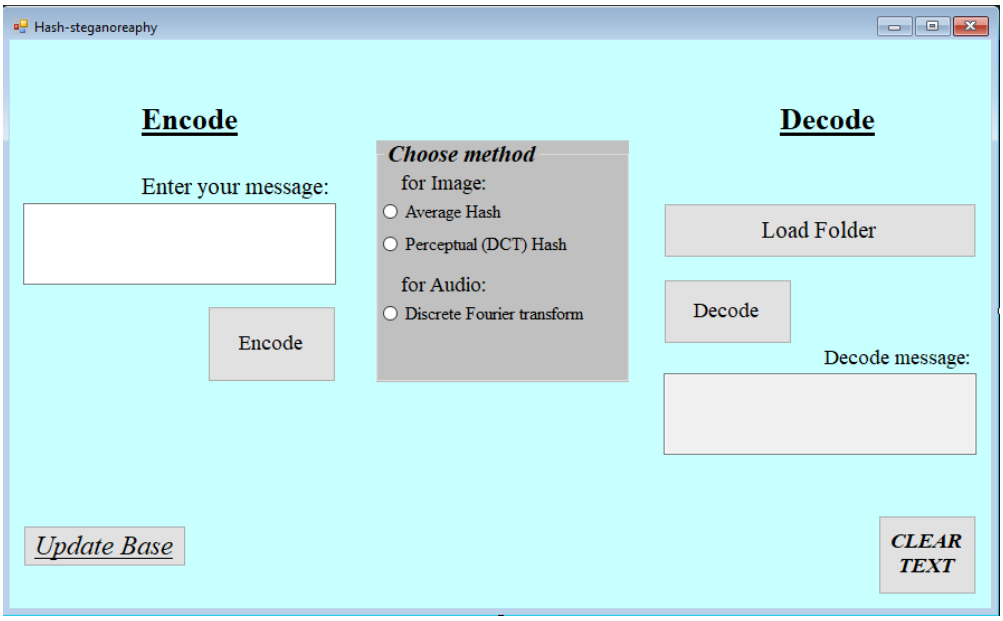

Рисунок 3.2 – Інтерфейс програмного продукту

При першому використані програми необхідно створити або оновити базу хеш-кодів. Для цього користувач натискає на «Discrete Fourier transform», потім обирає «Update Base»(рис.3.3).

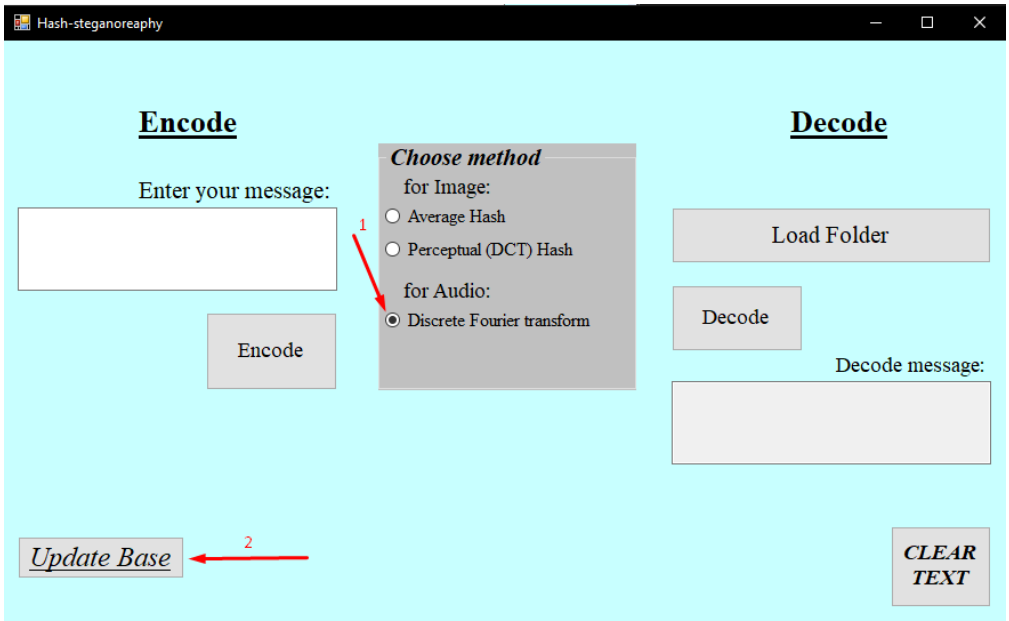

Рисунок 3.3 – Підготовка до створення бази даних

Після чого відкривається вікно, де необхідно обрати папку з аудіо-файлами, що будуть використовуватись для шифрування повідомлення(рис.3.4).

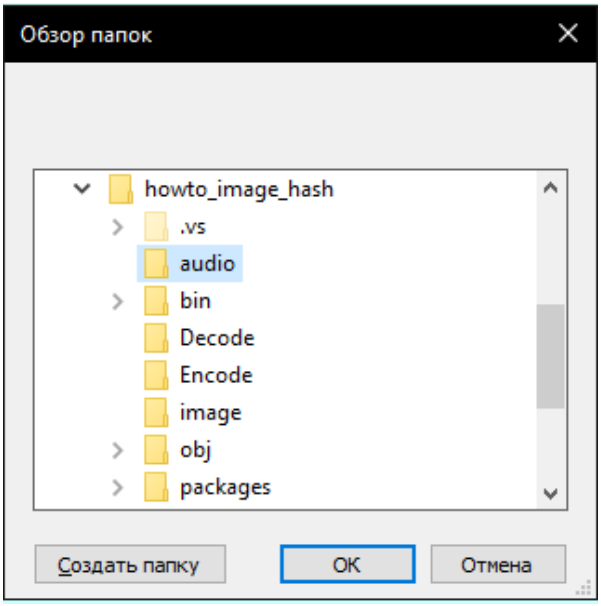

Рисунок 3.4 – Вибір кореневої папки аудіо-файлів

В полі «Enter your message» користувач вводить повідомлення, яке бажає зашифрувати, після чого натискає кнопку «Encode». Перед користувачем спливає вікно із аудіо-файлами, які представляють оригінальне повідомлення(рис.3.5).

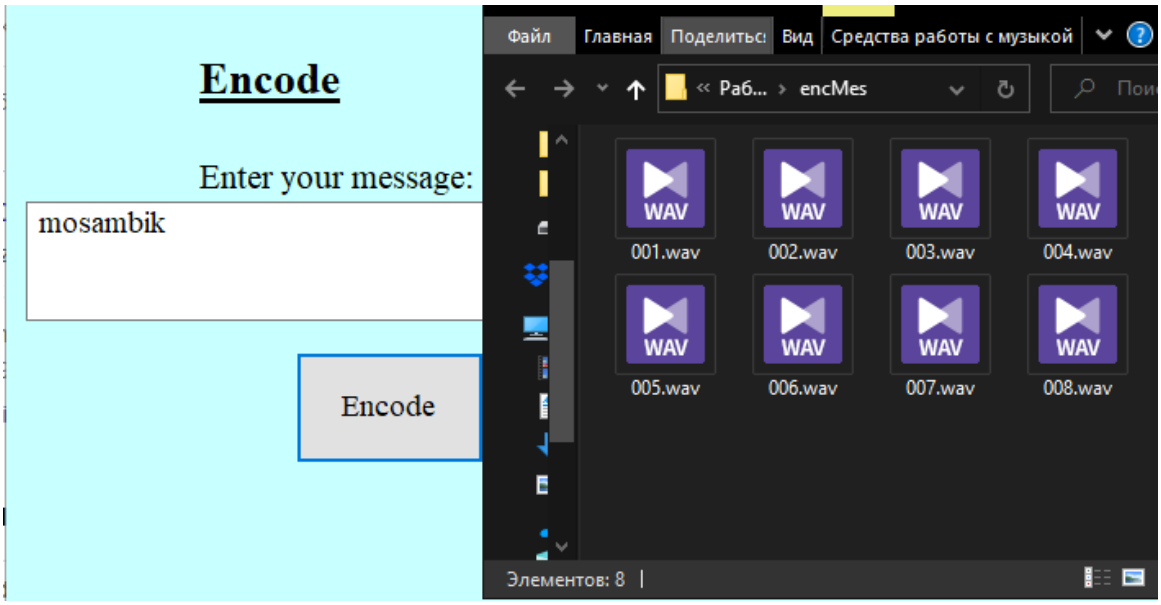

Рисунок 3.5 – Закодоване повідомлення «mosambik»

Декодування. Натиснути кнопку «Load Files», спливає вікно, де користувач вказує папку із стеганоповідомленням (рис.3.6).

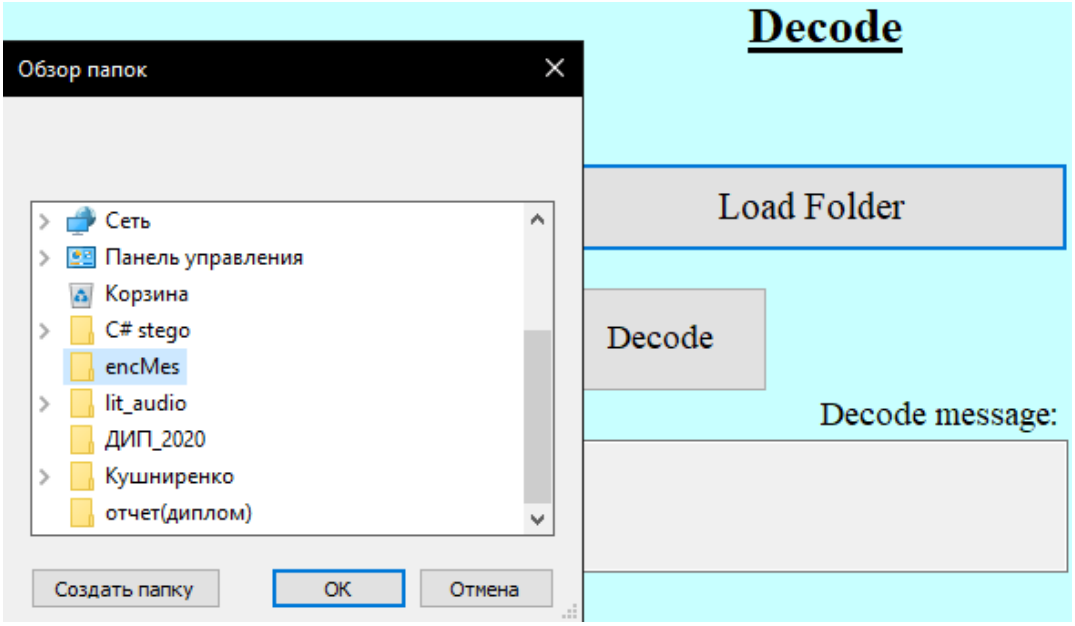

Рисунок 3.6 – Завантаження папки із стеганоповідомленням

Тільки після цього натискаємо на кнопку «Decode» і у текстовому полі нижче буде відображене зашифроване повідомлення(рис.3.7).

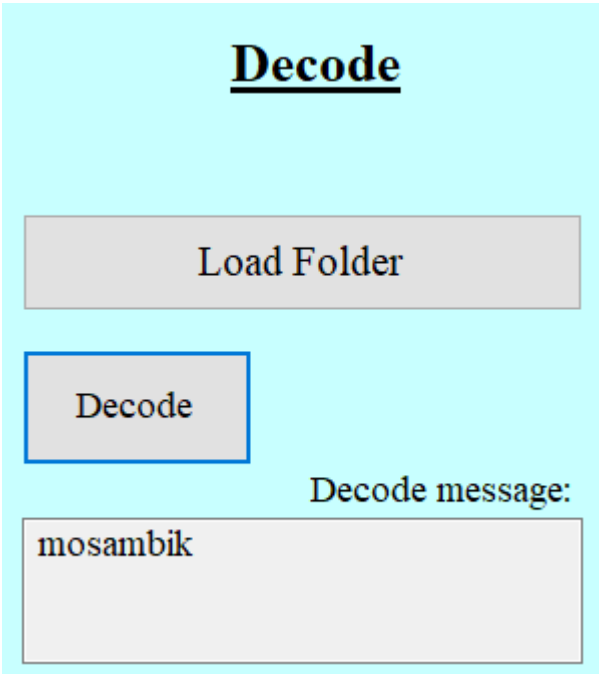

Рисунок 3.7 – Декодування стеганоповідомлення

Після закінчення шифрування/дешифрування рекомендується натиснути кнопку «Clear Text», щоб очистити текстові поля введення/виведення для подальшої роботи.

У даному розділі детально розглянули процес розробки програмного продукту та інтерфейсу користувача, описано причини вибору середовища програмування.

Інтерфейс зроблений максимально зручним і легким як у використанні, так і розумінні, жодних зайвих елементів. Код програмного продукту розділений на блоки, що безпосередньо полегшує читабельність та розуміння порядку виконання процесів прописаних у ньому.

ВИСНОВОК

В ході виконання кваліфікаційної роботи проведено аналіз сучасних алгоритмів аудіо-стеганографії та існуючих хеш-алгоритмів для обчислення хеш-коду аудіо-файлів.

Для модифікації обрано алгоритм, що використовую низькі частоти отримані в результаті швидкого перетворення Фур’є.

Модифікований алгоритм хеш-стеганографії дозволяє підвищити стійкість хеш-кодів аудіо-файлів до різноманітних атак –стиснення та накладення шумів.

Алгоритм реалізовано на мові об’єктно-орієнтовного програмування C# (Visual Studio 2019). За допомогою Windows Form сконструйовано легкий та зрозумілий інтерфейс користувача.

Мету досягнуто, усі поставлені в роботі задачі вирішено.

ПЕРЕЛІК ПОСИЛАНЬ

1. Тієн Ч. та Лін Я., —Розподіл секрету в зображеннях. *Computers & Graphics.* 2011. vol. 26. no. 5, P.765–770.

2. Barsukov, V.S. and Romantsov, A.P. , “Computer steganography yesterday, today, tomorrow”, *Tehnika dlja specsluzhb.* URL: http://www.bnti.ru/showart.asp?aid=330&lvl=03.07.06

3. J. Johnston and K. Brandenburg. Wideband Coding Perceptual Consideration for Speech and Music. *Advances in Speech Signal Processing.* S. Furoi and M. Sondhi, Eds. New York: Marcel Dekker. 1992.

4. Pfitzmann. “Information Hiding Terminology”. *First International Workshop on Information Hiding*. Ma 30 – June 1.1996.Cambridge. P.347- 350

5. Pramatha Nath Basu. Embedding of Text in Audio – A case of Steganography. *International Conference on Recent Trends in Information, Telecommunication and Computing*. 2010

6. Jayram P., Ranganatha H. R.,Anupama H. S. Inforamation Hiding Using Audio Steganography- A Survey. *The international journal of Multimedia & Its application (IJMA)*. August 2011. V.3, #3.

7. Cvejic N., Seppanen T. Increasing Robustness of LSB Audio Steganography Using a Novel Embedding Method, *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC04).* V. 2. (Washington, DC, USA, 2004), P. 533

8. Cvejic N., Seppanen T. Reduced distortion bit-modification for LSB audio steganography. *J. Universal Comput. Sci* .2005. 11(1).P.56-65.

9. Ahmed M.A., Kiah L.M., Zaidan B.B., Zaidan A.A. A Novel Embedding Method to Increase Capacity and Robustness of Low-bit Encoding Audio Steganography Technique Using Noise Gate Software Logic Algorithm. *J. Appl. Sci* 2010, 10.P.59-64.

10. Nedeljko Cvejic, Tapio Seppben(IEEE 2002) Increasing The Capacity Of LSB-Based Audio Steganography

11. Ranganatha H. R., Anupama H. S. Inforamation Hiding Using Audio Steganography- A Survey. *The international journal of Multimedia & Its application (IJMA)* 2011.V.3, #3.

12. Gang L., Akansu A.N., Ramkumar M. MP3 resistant oblivious steganography, Proceedings of, *IEEE International Conference on Acoustics, Speech, and Signal Processing.* 2001. V.3. P.1365–1368.

13. Kriti Saroha, Pradeep Kumar Singh. A Variant of LSB Steganography for Hiding Images in Audio. *International Journal of Computer Applications* (0975 – 8887) V.11. #6. December 2010.

14. Bret Dunbar, *SANS Institute InfoSec Reading Room* .A Detailed look at Steganographic Techniques and their use in an Open-Systems Environment.

15. Андерсон. *Скрытие информации: 1-й международный семинар, том 1174 конспектов лекций по информатике, Институт Исаака Ньютона* . Шпрингер-Верлаг, Берлин, Германия; 1996 г.

16. K. Ramani. Steganography using BPCS to the integer wavelet transformed image. *IJCSNS International Journal of Computer Science and Network Security.* V.7. #7. 2007. P.293- 302.

17. Fatiha Djebbar. Comparative study of digital audio steganography techniques", *EURASIP Journal on Audio, Speech, and Music Processing*. 2012, P.1192-1203

18. Zwicker E., Fastl H. *Psychoacoustics*. Springer Verlag, Berlin. 1990.

19. Abbas Cheddad. Digital image steganography: Survey and analysis of current methods. *Signal Processing.* V.90. #3. March 2010. P.727-752.

20. Bender W., Gruhl D., Morimoto N., Lu A. Techniques for Data Hiding. *IBM Syst. J* 1996. 35(3 and 4). P.313-336.

21. Джеббар Ф., Аяд Б., Хамам Х., Абед-Мераим К. Взгляд на новейшие методы аудиостеганографии. *Инновации в информационных технологиях (ИИТ)*, Международная конференция 2011 р. С.409–414.

22. Fallahpour M., Megias D. Водяные знаки высокой емкости с использованием интерполяции амплитуды FFT. *Специалисты по информации и коммуникациям (IEICE). Электрон Дж. Экспресс* 2009, 6 (14). P.1057-1063.

23. Джеббар Ф., Герчи Д., Абед-Мараим К., Хамам Х. Скрытие текста в высокочастотных компонентах спектра речи. *Информационная обработка сигналов и их приложения (ISSPA)*, 10-я Международная конференция 2010. С. 666–669.

24. Джеббар Ф., Аяд Б., Абед-Мераим К., Хабиб Х . Унифицированный алгоритм сокрытия данных фазового и амплитудного спектров речи. Принято в Журнале безопасности и сетей связи. (John Wiley and Sons, Ltd, 4 апреля 2012 г.)

25. Джеббар Ф., Абед-Мараим К., Гуэрчи Д., Хамам Х. Скрытие спектра текста в речи на основе динамической энергии с использованием свойств маскировки речи. *Промышленная мехатроника и автоматизация (ICIMA).* 2-я Международная конференция 2010 г., том 2, С. 422–426.

26. Джеббар Ф., Хамам Х., Абед-Мераим К., Гуэрчи Д. Контролируемое искажение для стеганографии спектра данных в речи с высокой пропускной способностью, *Международная конференция по интеллектуальному сокрытию информации и обработке мультимедийных сигналов (IEEE-IIHMSP).* С. 212–215

27. Бохонько М.В Модифікація методу хеш-стеганографії / М.В. Бохонько, В.В. Зоріло. VII Міжнародна науково-практична інтернет-конференція «Сучасний рух науки». 6-7 червня 2019 р. С.605-608.

28. Lee, Y. W., Cheatham, T. P., Wiesner, J. B. Application of Correlation Analysis to the Detection of Periodic Signals in Noise. 1950.P.1165–1171.

29. Djebbar F., Hamam H., Abed-Meraim K., Guerchi D. Controlled distortion for high capacity data-in-speech spectrum steganography. *International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IEEE-IIHMSP)*. 2010. P. 212–215

30. Создание приложения Windows Forms на C# в Visual Studio https://docs.microsoft.com/ru-ru/visualstudio/ide/create-csharp-winform-visual-studio?view=vs-2019

31. Березуцький, В. В. Навч. пос. Т. С. Бондаренко, Г. Г. Валенко та ін. / за заг. ред. В. В. Березуцького. — 2-ге вид., перероб. і доп. — Х.: Факт, 2007. — 480 с.

32. Жидецький В.Ц., Джигирей В.С., Мельник О.В. Основи охорони праці. Львів: Афіша, 2001. 350 с

33. Санітарні норми виробничого шуму, ультразвуку та інфразвуку ДСН 3.3.6.037-99

34. ПРИРОДНЕ І ШТУЧНЕ ОСВІТЛЕННЯ ДБН В.2.5-28-2006

35. Санітарні норми мікроклімату виробничих приміщень ДСН 3.3.6.042-99

36. . ДСН 3.3.6.042-99. Державні санітарні норми мікроклімату виробничих приміщень. [Чинний від 1999.12.01]. Київ, 1999. 63 с.

37. Електрична енергія. Сумісність технічних засобів. Норми якості електричної енергії в системах електропостачання загального призначення ДСТУ 13109-97

38. правила охорони праці під час експлуатації електронно-обчислювальних машин ДНАОП 0.00.1.33-99

39. Правила безпеки у прокатному виробництві ДНАОП 0.03-3.05-77

40. Про затвердження Державних санітарних норм та правил при роботі з джерелами електромагнітних полів ДСНіП 3.3.6.096

41. Система стандартів безпеки праці в будівництві. ДСТУ-Н Б А.3.2-1

42. Санітарні норми мікроклімату виробничих приміщеньСН 4088-86

43. ДСН 3.3.6.042-99. Державні санітарні норми мікроклімату виробничих приміщень. [Чинний від 01.12.1999]. Київ, 1999. 19 с

44. Гігієна. Терміни та визначення основних понять ДСТУ 3038-85

45. ДСанПІН 3.3.2.007-98. Гігієнічні вимоги до організації роботи з візуальними дисплейними терміналами електронно-обчислювальних машин. [Чинний від 1998.12.10]. Київ, 1998.

46. Кодекс Законів «Про працю» (ст. 14. Обов’язки працівника щодо додержання вимог нормативно-правових актів з охорони праці)

47. ДСТУ ISO 9241-5:2004. Ергономічні вимоги до роботи з відеотерміналами в офісі. Частина 5. Вимоги до компонування робочого місця та до робочої пози (2286). [Чинний від 01.01.2006]. Київ, 2004. 27 с.

48. ДСН 3.3.6.037-99. Державні санітарні норми виробничого шуму, ультразвуку та інфразвуку. [Чинний від 01.12.1999]. Київ, 1999. 36 с.

49. ДБН В.1.1-7-2002 С 25. ПОЖЕЖНА БЕЗПЕКА ОБ'ЄКТІВ БУДІВНИЦТВА. [Чинний від 01.01.2007]. Київ, 2007. 42 с.

50. ОНТП 24-86. Визначення категорій приміщень і будинків по вибухопожежної і пожежної небезпеки [Електронний ресурс]. - Режим доступу: World Wide Web. - URL: https://dnaop.com/html/2590/doc-ОНТП_24-86

Додаток А

Код програмного продукту

```
using System;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Drawing.Drawing2D;
using System.Drawing.Imaging;
using System.IO;
using System.Security.Cryptography;
using System.Diagnostics;
using System.Numerics;
using System.Collections.Generic;

namespace howto_image_hash
{
    public partial class Form1 : Form
    {
        private float[][] _dctMatrix;
        public        static        string[]        imGEO        =
Directory.GetFiles(@"C:\\Users\\Администратор\\Desktop\\C#
stego\\howto_image_hash\\image",                                  "*.jpg",
SearchOption.AllDirectories);
        public        static        string[]        audioGEO        =
Directory.GetFiles(@"C:\\Users\\Администратор\\Desktop\\C#
stego\\howto_image_hash\\audio",                                  "*.wav",
SearchOption.AllDirectories);
        public     static     string[]     Average_Hash     =     new
string[imGEO.Length];
        public    static    string[]    Perceptual_Hash    =    new
string[imGEO.Length];
        public     static     string[]     audioHash     =     new
string[audioGEO.Length];
        public static string path = null;


        public Form1()
        {
            InitializeComponent();
        }

        public void Update_Click(object sender, EventArgs e)
        {
            if (aHash.Checked)
            {
                using (var dialog = new FolderBrowserDialog())
                    if (dialog.ShowDialog() == DialogResult.OK)
```

```
                    path = dialog.SelectedPath;
                string[] files = Directory.GetFiles(path, "*.jpg",
SearchOption.AllDirectories);
                Bitmap[] bmpArr = new Bitmap[files.Length];
                for (int i = 0; i < bmpArr.Length; i++)
                {
                    bmpArr[i] = new Bitmap(files[i]);
                }
                File.Delete("C:\\Users\\Администратор\\Desktop\\C#
stego\\howto_image_hash\\Encode\\Average_Hash.txt");

                for (int i = 0; i < bmpArr.Length; i++)
                {
                    Bitmap shrunk_bm = ScaleTo(bmpArr[i], 8, 8,
InterpolationMode.High);
                    Bitmap grayscale_bm = ToMonochrome(shrunk_bm);
                    Bitmap bmpMat = new Bitmap(grayscale_bm);
                    Color[,] color = new Color[bmpMat.Width,
bmpMat.Height];
                    int count = 0;
                    string[] lll = new string[64];
                    int[,] colorS = new int[bmpMat.Width,
bmpMat.Height];
                    for (int y = 0; y < bmpMat.Height; y++)
                    {
                        for (int x = 0; x < bmpMat.Width; x++)
                        {
                            color[x, y] = bmpMat.GetPixel(x, y);
                            colorS[x, y] = int.Parse(color[x,
y].B.ToString("G"));
                            lll[count++] = color[x,
y].B.ToString("G");
                        }
                    }
                    int summ = 0;
                    for (int t = 0; t < lll.Length; t++)
                    {
                        summ += int.Parse(lll[t]);
                    }
                    int mid = summ / lll.Length;
                    for (int y = 0; y < bmpMat.Height; y++)
                    {
                        for (int x = 0; x < bmpMat.Width; x++)
                        {
                            if (colorS[x, y] > mid)
                                colorS[x, y] = 1;
                            else if (colorS[x, y] <= mid)
                                colorS[x, y] = 0;
                        }
                    }
                    string pixel_s = "";
                    for (int x = 0; x < 8; x++)
                    {
```

```
                        for (int y = 0; y < 8; y++)
                        {
                            if (x == y)
                                pixel_s += Convert.ToString(colorS[x,
y]);
                        }

                    }
                    for (int x = 0; x < 8; x++)
                    {
                        for (int y = 7; y > 0; y--)
                            if (x + y == 7)
                            {
                                pixel_s += Convert.ToString(colorS[x,
y]);
                                break;
                            }
                    }
                    Average_Hash[i] = GetHash(pixel_s);

File.AppendAllText("C:\\Users\\Администратор\\Desktop\\C#
stego\\howto_image_hash\\Encode\\Average_Hash.txt",  Average_Hash[i]+
Environment.NewLine);

                }

                //Process.Start(@"C:\Users\Администратор\Desktop\C#
stego\howto_image_hash\Encode\imHash.txt");
                MessageBox.Show("DataBase updated!");

            }
            else if (pHash.Checked)
            {
                using (var dialog = new FolderBrowserDialog())
                    if (dialog.ShowDialog() == DialogResult.OK)
                        path = dialog.SelectedPath;
                string[]  files  =  Directory.GetFiles(path,  "*.jpg",
SearchOption.AllDirectories);
                Bitmap[] bmpArr = new Bitmap[files.Length];
                for (int i = 0; i < bmpArr.Length; i++)
                {
                    bmpArr[i] = new Bitmap(files[i]);
                }
                File.Delete("C:\\Users\\Администратор\\Desktop\\C#
stego\\howto_image_hash\\Encode\\Perceptual_Hash.txt");

                for (int i = 0; i < bmpArr.Length; i++)
                {
                    Bitmap  shrunk_bm  =  ScaleTo(bmpArr[i],  32,  32,
InterpolationMode.High);
                    Bitmap grayscale_bm = ToMonochrome(shrunk_bm);
                    Bitmap bmpMat = new Bitmap(grayscale_bm);
```

```
                    Color[,]   color   =   new   Color[bmpMat.Width,
bmpMat.Height];
                    var colorS = new float[1024];
                    int count = 0;
                    for (int y = 0; y < bmpMat.Height; y++)
                    {
                        for (int x = 0; x < bmpMat.Width; x++)
                        {
                            color[x, y] = bmpMat.GetPixel(x, y);
                            colorS[count]    =    int.Parse(color[x,
y].B.ToString("G"))/255.0f;
                            count++;
                        }

                    }
                    _dctMatrix = GenerateDctMatrix(32);
                    var dctPixels = ComputeDct(colorS, _dctMatrix);
                    var dctHashPixels = new float[64];
                    for (var x = 0; x < 8; x++)
                    {
                        for (var y = 0; y < 8; y++)
                        {
                            dctHashPixels[x + y * 8] = dctPixels[x +
1][y + 1];
                        }
                    }
                    var pixelList = new List<float>(dctHashPixels);
                    pixelList.Sort();
                    var median = (pixelList[31] + pixelList[32]) / 2;
                    var hash = 0UL;
                    for (var f = 0; f < 64; f++)
                    {
                        if (dctHashPixels[f] > median)
                        {
                            hash |= (1UL << f);
                        }
                    }
                    Perceptual_Hash[i] = GetHash(hash.ToString());

File.AppendAllText("C:\\Users\\Администратор\\Desktop\\C#
stego\\howto_image_hash\\Encode\\Perceptual_Hash.txt",
Perceptual_Hash[i] + Environment.NewLine);

                }
                MessageBox.Show("Base Done!");
            }
            else if (DFT.Checked)
            {
                using (var dialog = new FolderBrowserDialog())
                    if (dialog.ShowDialog() == DialogResult.OK)
                        path = dialog.SelectedPath;
                string[] files = Directory.GetFiles(path);
```

```
                File.Delete("C:\\Users\\Администратор\\Desktop\\C#
stego\\howto_image_hash\\Encode\\audioHash.txt");
                for (int i = 0; i < files.Length; i++)
                {
                    var audio = File.OpenRead(files[i]);
                    byte[] buffer = new byte[256];
                    audio.Read(buffer, 0, buffer.Length);
                    string audio_Hash = "";
                    Complex[] c = new Complex[buffer.Length];
                    for (int j = 0; j < buffer.Length; j++)
                    {
                        c[j] = buffer[j];

                    }
                    foreach (Complex c1 in fft(c))
                    {
                        audio_Hash                                    +=
Convert.ToInt16((Math.Log(Math.Abs(c1.Real)       +       1)       *
Math.Log(Math.Abs(c1.Imaginary) + 1))).ToString();
                    }
                    audio_Hash = GetHash(audio_Hash);

File.AppendAllText("C:\\Users\\Администратор\\Desktop\\C#
stego\\howto_image_hash\\Encode\\audioHash.txt",     audio_Hash     +
Environment.NewLine);
                }

                MessageBox.Show("DB check");

            }
            else
            {
                MessageBox.Show("Choose your environment / encryption
method");
            }
        }

        private static float[][] ComputeDct(float[] image, float[][]
dctMatrix)
        {
            // Get the size of dct matrix. We assume that the image
is same size as dctMatrix
            var size = dctMatrix.GetLength(0);

            // Make image matrix
            var imageMat = new float[size][];
            for (var i = 0; i < size; i++)
            {
                imageMat[i] = new float[size];
            }

            for (var y = 0; y < size; y++)
            {
```

```
                for (var x = 0; x < size; x++)
                {
                    imageMat[y][x] = image[x + y * size];
                }
            }

            return      Multiply(Multiply(dctMatrix,      imageMat),
Transpose(dctMatrix));
        }
        private static float[][] Multiply(float[][] a, float[][] b)
        {
            var n = a[0].Length;
            var c = new float[n][];
            for (var i = 0; i < n; i++)
            {
                c[i] = new float[n];
            }

            for (var i = 0; i < n; i++)
                for (var k = 0; k < n; k++)
                    for (var j = 0; j < n; j++)
                        c[i][j] += a[i][k] * b[k][j];
            return c;
        }
        private static float[][] Transpose(float[][] mat)
        {
            var size = mat[0].Length;
            var transpose = new float[size][];

            for (var i = 0; i < size; i++)
            {
                transpose[i] = new float[size];
                for (var j = 0; j < size; j++)
                    transpose[i][j] = mat[j][i];
            }
            return transpose;
        }
        private static float[][] GenerateDctMatrix(int size)
        {
            var matrix = new float[size][];
            for (int i = 0; i < size; i++)
            {
                matrix[i] = new float[size];
            }

            var c1 = Math.Sqrt(2.0f / size);

            for (var j = 0; j < size; j++)
            {
                matrix[0][j] = (float)Math.Sqrt(1.0d / size);
            }

            for (var j = 0; j < size; j++)
```

```
            {
                for (var i = 1; i < size; i++)
                {
                    matrix[i][j] = (float)(c1 * Math.Cos(((2 * j + 1)
* i * Math.PI) / (2.0d * size)));
                }
            }
            return matrix;
        }
        private static Complex w(int k, int N)
        {
            if (k % N == 0) return 1;
            double arg = -2 * Math.PI * k / N;
            return new Complex(Math.Cos(arg), Math.Sin(arg));
        }

        public static Complex[] fft(Complex[] x)
        {
            Complex[] X;
            int N = x.Length;
            if (N == 2)
            {
                X = new Complex[2];
                X[0] = x[0] + x[1];
                X[1] = x[0] - x[1];
            }
            else
            {
                Complex[] x_even = new Complex[N / 2];
                Complex[] x_odd = new Complex[N / 2];
                for (int i = 0; i < N / 2; i++)
                {
                    x_even[i] = x[2 * i];
                    x_odd[i] = x[2 * i + 1];
                }
                Complex[] X_even = fft(x_even);
                Complex[] X_odd = fft(x_odd);
                X = new Complex[N];
                for (int i = 0; i < N / 2; i++)
                {
                    X[i] = X_even[i] + w(i, N) * X_odd[i];
                    X[i + N / 2] = X_even[i] - w(i, N) * X_odd[i];
                }
            }
            return X;
        }

        private Bitmap ScaleTo(Bitmap bm, int wid, int hgt,
InterpolationMode interpolation_mode)
        {
            Bitmap new_bm = new Bitmap(wid, hgt);
            using (Graphics gr = Graphics.FromImage(new_bm))
            {
```

```
                RectangleF source_rect = new RectangleF(-0.5f, -0.5f,
bm.Width, bm.Height);
                Rectangle dest_rect = new Rectangle(0, 0, wid, hgt);
                gr.InterpolationMode = interpolation_mode;
                gr.DrawImage(bm,       dest_rect,       source_rect,
GraphicsUnit.Pixel);
            }
            return new_bm;
        }

        private Bitmap ToMonochrome(Image image)
        {
            ColorMatrix cm = new ColorMatrix(new float[][]
            {
                new float[] {0.299f, 0.299f, 0.299f, 0, 0},
                new float[] {0.587f, 0.587f, 0.587f, 0, 0},
                new float[] {0.114f, 0.114f, 0.114f, 0, 0},
                new float[] { 0, 0, 0, 1, 0},
                new float[] { 0, 0, 0, 0, 1}
            });
            ImageAttributes attributes = new ImageAttributes();
            attributes.SetColorMatrix(cm);
            Point[] points =
            {
                new Point(0, 0),
                new Point(image.Width, 0),
                new Point(0, image.Height),
            };
            Rectangle  rect  =  new  Rectangle(0,  0,  image.Width,
image.Height);
            Bitmap bm = new Bitmap(image.Width, image.Height);
            using (Graphics gr = Graphics.FromImage(bm))
            {
                gr.DrawImage(image, points, rect,
                    GraphicsUnit.Pixel, attributes);
            }
            return bm;
        }

        static string GetHash(string input)
        {
            MD5 md5Hasher = MD5.Create();
            byte[]                        data                        =
md5Hasher.ComputeHash(Encoding.Default.GetBytes(input));
            StringBuilder sBuilder = new StringBuilder();
            for (int i = 0; i < data.Length; i++)
            {
                sBuilder.Append(data[i].ToString("x2"));
            }
            return sBuilder.ToString();
        }

        private void Encode_ms_Click(object sender, EventArgs e)
```

```
        {
            if (aHash.Checked)
            {
                string main_text = OriginalText.Text;
                byte[] result = Encoding.ASCII.GetBytes(main_text);
                string                    outStr                    =
BitConverter.ToString(result).Replace("-", "");
                string[] mas = new string[result.Length];
                int startInd = 0;
                for (int t = 0; t < mas.Length; t++)
                {
                    mas[t] = outStr.Substring(startInd, 2).ToLower();
                    startInd = startInd + 2;
                }

                string[]                    imHash                    =
File.ReadAllLines("C:\\Users\\Администратор\\Desktop\\C#
stego\\howto_image_hash\\Encode\\Average_Hash.txt");
                for (int c = 0; c < imHash.Length; c++)
                {
                    imHash[c] = imHash[c].Substring(0, 2);
                }
                string                  targetPath                  =
"C:\\Users\\Администратор\\Desktop\\encMes";

                if (Directory.Exists(targetPath))
                {

Directory.Delete("C:\\Users\\Администратор\\Desktop\\encMes", true);
                }
                Directory.CreateDirectory(targetPath);
                int NI = 1;
                string[] odd = new string[mas.Length];
                for (int i = 0; i < mas.Length; i++)
                {
                    int j = 0;
                    int count = 0;
                    for (j = 0; j < imGEO.Length; j++)
                    {
                        if (Convert.ToString(mas[i]) == imHash[j])
                        {
                            string             newName             =
Convert.ToString(NI).PadLeft(2, '0');
                            string fileName = imGEO[j];
                            int A = 0;
                            for (int b = 0; b < mas.Length; b++)
                            {
                                if (odd[b] == fileName)
                                {
                                    A = 1;
                                    count = b;
                                    break;
                                }
```

```
                            }
                            odd[i] = fileName;
                            if (A != 1)
                            {

Directory.CreateDirectory(targetPath);
                                File.Copy(fileName,
"C:\\Users\\Администратор\\Desktop\\encMes\\0" + newName + ".jpg");
                                NI++;
                                break;
                            }
                        }
                        else if (j == imGEO.Length - 1) {
                            string             newName             =
Convert.ToString(NI).PadLeft(2, '0');
                            string fileName = odd[count];
                            Directory.CreateDirectory(targetPath);
                            File.Copy(fileName,
"C:\\Users\\Администратор\\Desktop\\encMes\\0" + newName + ".jpg");
                            NI++;
                            break;
                        }
                    }
                }

Process.Start("C:\\Users\\Администратор\\Desktop\\encMes", "-p");

            }
            else if (pHash.Checked)
            {
                string main_text = OriginalText.Text;
                byte[] result = Encoding.ASCII.GetBytes(main_text);
                string                    outStr                    =
BitConverter.ToString(result).Replace("-", "");
                string[] mas = new string[result.Length];
                int startInd = 0;
                for (int t = 0; t < mas.Length; t++)
                {
                    mas[t] = outStr.Substring(startInd, 2).ToLower();
                    startInd = startInd + 2;
                }

                string[]                    imHash                    =
File.ReadAllLines("C:\\Users\\Администратор\\Desktop\\C#
stego\\howto_image_hash\\Encode\\Perceptual_Hash.txt");
                for (int c = 0; c < imHash.Length; c++)
                {
                    imHash[c] = imHash[c].Substring(0, 2);
                }
                string                   targetPath                   =
"C:\\Users\\Администратор\\Desktop\\encMes";
                DirectoryInfo           dInf           =           new
DirectoryInfo("C:\\Users\\Администратор\\Desktop\\encMes\\");
```

```
                foreach (FileInfo file in dInf.GetFiles())
                {
                    file.Delete();
                }
                if (Directory.Exists(targetPath))
                {
                    Directory.Delete(targetPath, true);
                }
                Directory.CreateDirectory(targetPath);
                int NI = 1;
                string[] odd = new string[mas.Length];
                for (int i = 0; i < mas.Length; i++)
                {
                    int j = 0;
                    int count = 0;
                    for (j = 0; j < imGEO.Length; j++)
                    {
                        if (Convert.ToString(mas[i]) == imHash[j])
                        {
                            string              newName             =
Convert.ToString(NI).PadLeft(2, '0');
                            string fileName = imGEO[j];
                            int A = 0;
                            for (int b = 0; b < mas.Length; b++)
                            {
                                if (odd[b] == fileName)
                                {
                                    A = 1;
                                    count = b;
                                    break;
                                }
                            }
                            odd[i] = fileName;
                            if (A != 1)
                            {

Directory.CreateDirectory(targetPath);
                                File.Copy(fileName,
"C:\\Users\\Администратор\\Desktop\\encMes\\0" + newName + ".jpg");
                                NI++;
                                break;
                            }
                        }
                        else if (j == imGEO.Length - 1)
                        {
                            string              newName             =
Convert.ToString(NI).PadLeft(2, '0');
                            string fileName = odd[count];
                            Directory.CreateDirectory(targetPath);
                            File.Copy(fileName,
"C:\\Users\\Администратор\\Desktop\\encMes\\0" + newName + ".jpg");
                            NI++;
                            break;
```

```
                        }
                    }
                }

Process.Start("C:\\Users\\Администратор\\Desktop\\encMes", "-p");
            }
            else if (DFT.Checked)
            {
                string main_text = OriginalText.Text;
                byte[] result = Encoding.ASCII.GetBytes(main_text);
                string                    outStr                    =
BitConverter.ToString(result).Replace("-", "");
                string[] mas = new string[result.Length];
                int startInd = 0;
                for (int t = 0; t < mas.Length; t++)
                {
                    mas[t] = outStr.Substring(startInd, 2).ToLower();
                    startInd = startInd + 2;
                }

                string[]                  audioHash                 =
File.ReadAllLines("C:\\Users\\Администратор\\Desktop\\C#
stego\\howto_image_hash\\Encode\\audioHash.txt");
                for (int c = 0; c < audioHash.Length; c++)
                {
                    audioHash[c] = audioHash[c].Substring(0, 2);
                }
                string                  targetPath                  =
"C:\\Users\\Администратор\\Desktop\\encMes";
                if (Directory.Exists(targetPath))
                {
                    Directory.Delete(targetPath, true);
                }
                Directory.CreateDirectory(targetPath);
                int NI = 1;
                string[] odd = new string[mas.Length];
                for (int i = 0; i < mas.Length; i++)
                {
                    int j = 0;
                    int count = 0;
                    for (j = 0; j < audioGEO.Length; j++)
                    {
                        if (Convert.ToString(mas[i]) == audioHash[j])
                        {
                            string              newName             =
Convert.ToString(NI).PadLeft(2, '0');
                            string fileName = audioGEO[j];
                            int A = 0;
                            for (int b = 0; b < mas.Length; b++)
                            {
                                if (odd[b] == fileName)
                                {
                                    A = 1;
```

```
                                    count = b;
                                    break;
                                }
                            }
                            odd[i] = fileName;
                            if (A != 1)
                            {

Directory.CreateDirectory(targetPath);
                                File.Copy(fileName,
"C:\\Users\\Администратор\\Desktop\\encMes\\0" + newName + ".wav");
                                NI++;
                                break;
                            }
                        }
                        else if (j == audioGEO.Length - 1) {
                            string              newName              =
Convert.ToString(NI).PadLeft(2, '0');
                            string fileName = odd[count];
                            Directory.CreateDirectory(targetPath);
                            File.Copy(fileName,
"C:\\Users\\Администратор\\Desktop\\encMes\\0" + newName + ".wav");
                            NI++;
                            break;
                        }
                    }
                }


Process.Start("C:\\Users\\Администратор\\Desktop\\encMes", "-p");


            }
            else
            {
                MessageBox.Show("Choose your environment / encryption
method");
            }
        }

        public void LoadFiles_Click(object sender, EventArgs e)
        {
            using (var dialog = new FolderBrowserDialog())
                if (dialog.ShowDialog() == DialogResult.OK)
                    path = dialog.SelectedPath;
        }

        private void Decode_ms_Click(object sender, EventArgs e)
        {
            if (aHash.Checked)
            {
```

```
                string[] files = Directory.GetFiles(path, "*.jpg",
SearchOption.AllDirectories);
                Bitmap[] bmpArr = new Bitmap[files.Length];
                for (int i = 0; i < bmpArr.Length; i++)
                {
                    bmpArr[i] = new Bitmap(files[i]);
                }
                string[] decHash = new string[files.Length];
                for (int i = 0; i < bmpArr.Length; i++)
                {
                    Bitmap shrunk_bm = ScaleTo(bmpArr[i], 8, 8,
InterpolationMode.High);
                    Bitmap grayscale_bm = ToMonochrome(shrunk_bm);
                    Bitmap bmpMat = new Bitmap(grayscale_bm);
                    Color[,] color = new Color[bmpMat.Width,
bmpMat.Height];
                    int count = 0;
                    string[] lll = new string[64];
                    int[,] colorS = new int[bmpMat.Width,
bmpMat.Height];
                    for (int y = 0; y < bmpMat.Height; y++)
                    {
                        for (int x = 0; x < bmpMat.Width; x++)
                        {
                            color[x, y] = bmpMat.GetPixel(x, y);
                            colorS[x, y] = int.Parse(color[x,
y].B.ToString("G"));
                            lll[count++] = color[x,
y].B.ToString("G");
                        }
                    }
                    int summ = 0;
                    for (int t = 0; t < lll.Length; t++)
                    {
                        summ += int.Parse(lll[t]);
                    }
                    int mid = summ / lll.Length;
                    for (int y = 0; y < bmpMat.Height; y++)
                    {
                        for (int x = 0; x < bmpMat.Width; x++)
                        {
                            if (colorS[x, y] > mid)
                                colorS[x, y] = 1;
                            else if (colorS[x, y] <= mid)
                                colorS[x, y] = 0;
                        }
                    }
                    string pixel_s = "";
                    for (int x = 0; x < 8; x++)
                    {
                        for (int y = 0; y < 8; y++)
                        {
                            if (x == y)
```

```
                                pixel_s += Convert.ToString(colorS[x,
y]);
                            }

                        }
                        for (int x = 0; x < 8; x++)
                        {
                            for (int y = 7; y > 0; y--)
                                if (x + y == 7)
                                {
                                    pixel_s += Convert.ToString(colorS[x,
y]);
                                    break;
                                }
                        }
                        decHash[i]    =    GetHash(pixel_s).Substring(0,
2).Insert(0, "0x");

File.AppendAllText("C:\\Users\\Администратор\\Desktop\\C#
stego\\howto_image_hash\\Decode\\Average_Hash.txt",
                            decHash[i] + Environment.NewLine);
                    }
                    byte[]     bytes     =     decHash.Select(item     =>
Convert.ToByte(item, 16)).ToArray();
                    string  resultConv  =  Encoding.ASCII.GetString(bytes,
0, bytes.Length);
                    Decrypt_ms.Text = resultConv;

                }
                else if (pHash.Checked)
                {
                    string[]  files  =  Directory.GetFiles(path,  "*.jpg",
SearchOption.AllDirectories);
                    Bitmap[] bmpArr = new Bitmap[files.Length];
                    for (int i = 0; i < bmpArr.Length; i++)
                    {
                        bmpArr[i] = new Bitmap(files[i]);
                    }
                    string[] decHash = new string[files.Length];
                    for (int i = 0; i < bmpArr.Length; i++)
                    {
                        Bitmap  shrunk_bm  =  ScaleTo(bmpArr[i],  32,  32,
InterpolationMode.High);
                        Bitmap grayscale_bm = ToMonochrome(shrunk_bm);
                        Bitmap bmpMat = new Bitmap(grayscale_bm);
                        Color[,]   color   =   new   Color[bmpMat.Width,
bmpMat.Height];
                        var colorS = new float[1024];
                        int count = 0;
                        for (int y = 0; y < bmpMat.Height; y++)
                        {
                            for (int x = 0; x < bmpMat.Width; x++)
                            {
```

```
                            color[x, y] = bmpMat.GetPixel(x, y);
                            colorS[count]    =    int.Parse(color[x,
y].B.ToString("G")) / 255.0f;
                            count++;
                        }

                    }
                    _dctMatrix = GenerateDctMatrix(32);
                    var dctPixels = ComputeDct(colorS, _dctMatrix);
                    var dctHashPixels = new float[64];
                    for (var x = 0; x < 8; x++)
                    {
                        for (var y = 0; y < 8; y++)
                        {
                            dctHashPixels[x + y * 8] = dctPixels[x +
1][y + 1];
                        }
                    }
                    var pixelList = new List<float>(dctHashPixels);
                    pixelList.Sort();
                    var median = (pixelList[31] + pixelList[32]) / 2;
                    var hash = 0UL;
                    for (var f = 0; f < 64; f++)
                    {
                        if (dctHashPixels[f] > median)
                        {
                            hash |= (1UL << f);
                        }
                    }
                    decHash[i]                                        =
GetHash(hash.ToString()).Substring(0, 2).Insert(0, "0x");

File.AppendAllText("C:\\Users\\Администратор\\Desktop\\C#
stego\\howto_image_hash\\Decode\\Average_Hash.txt",
                            decHash[i] + Environment.NewLine);
                }
                byte[]     bytes     =     decHash.Select(item     =>
Convert.ToByte(item, 16)).ToArray();
                string  resultConv  =  Encoding.ASCII.GetString(bytes,
0, bytes.Length);
                Decrypt_ms.Text = resultConv;
            }

            else if (DFT.Checked)
            {
                string[] files = Directory.GetFiles(path);
                string[] decHash = new string[files.Length];
                for (int i = 0; i < files.Length; i++)
                {
                    var audio = File.OpenRead(files[i]);
                    byte[] buffer = new byte[256];
                    audio.Read(buffer, 0, buffer.Length);
                    string audio_Hash = "";
```

```
                    Complex[] c = new Complex[buffer.Length];
                    for (int j = 0; j < buffer.Length; j++)
                    {
                        c[j] = buffer[j];

                    }
                    foreach (Complex c1 in fft(c))
                    {
                        audio_Hash                                    +=
Convert.ToInt16((Math.Log(Math.Abs(c1.Real)       +       1)       *
Math.Log(Math.Abs(c1.Imaginary) + 1))).ToString();
                    }
                    decHash[i]   =   GetHash(audio_Hash).Substring(0,
2).Insert(0, "0x"); ;

File.AppendAllText("C:\\Users\\Администратор\\Desktop\\C#
stego\\howto_image_hash\\Decode\\audioHash.txt",     audio_Hash     +
Environment.NewLine);
                }
                byte[]     bytes     =     decHash.Select(item     =>
Convert.ToByte(item, 16)).ToArray();
                string  resultConv  =  Encoding.ASCII.GetString(bytes,
0, bytes.Length);
                Decrypt_ms.Text = resultConv;
            }
            else
            {
                MessageBox.Show("Choose your environment / encryption
method");
            }


        }

        private void btnCLear_Click(object sender, EventArgs e)
        {
            OriginalText.Text = String.Empty;
            Decrypt_ms.Text = String.Empty;
        }

    }
}
```