Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

Навчально-науковий інститут деревообробних та
комп'ютерних технологій і дизайну
(повне найменування інституту, назва факультету (відділення))

Кафедра інформаційних систем та комп'ютерного моделювання
(повна назва кафедри (предметної, циклової комісії))

Пояснювальна записка

до дипломної роботи

перший (бакалаврський)
(рівень вищої освіти)

на тему: **Розроблення програмного забезпечення стеганографічного захисту аудіо файлів**

Виконав: студент IV курсу, групи ІСТ-41
спеціальності
126 "Інформаційні системи та технології"
(шифр і назва напряму підготовки, спеціальності)
Турчин В.П.
(прізвище та ініціали)
Керівник Різник О.Я.
(прізвище та ініціали)
Рецензент Дендюк М.В.
(прізвище та ініціали)

Львів – 2023

Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

ННІ деревообробних та комп'ютерних технологій і дизайну
Кафедра інформаційних систем та комп'ютерного моделювання
Рівень вищої освіти перший (бакалаврський)
Спеціальність 126 "Інформаційні системи та технології"
(шифр і назва)

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри ІСКМ
_____Сторожук О.Л.
„ 21 " ____11____ 2022 р.

З А В Д А Н Н Я
НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ
Турчин Володимир Петрович
(прізвище, ім'я, по батькові)

1. Тема роботи **Розроблення програмного забезпечення стеганографічного захисту аудіо файлів**
керівник роботи, канд. техн. наук, доцент Різник О.Я.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від 21.11. 2022 року № С-521
2. Термін подання студентом роботи 10. 06. 2023 р.
3. Вихідні дані до роботи:
- провести огляд стеганографічних алгоритмів захисту аудіо файлів
- дослідити математичну модель стеганографічного захисту аудіо файлів;
- розробити на мові програмування застосунок з інтуїтивним інтерфейсом;
- провести тестування роботи розробленого застосунка.
- провести тестування роботи розробленого програмного продукту.

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
Розділ 1. Стан проблемної області
Розділ 2. Інформаційне забезпечення
Розділ 3. Математичне забезпечення
Розділ 4. Програмне забезпечення
5. Перелік графічного матеріалу (розробка та відображення алгоритмів моніторингу довкілля за допомогою сенсорних давачів, структура програмного рішення, експериментальна частина)
Додаток А. Вихідний код розробленого програмного забезпечення
6. Дата видачі завдання 23 листопада 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Огляд літературних джерел	15.12-31.12.2022	виконано
2	Розділ 1. Стан проблемної області	03.01-15.02.2023	виконано
3	Розділ 2. Інформаційне забезпечення	16.02-22.03.2023	виконано
4	Розділ 3. Математичне забезпечення	23.03-19.04.2023	виконано
5	Розділ 4. Програмне забезпечення	20.04-30.04.2023	виконано
7	Експериментальна частина	02.05-06.05.2023	виконано
8	Оформлення дипломної роботи	09.05-13.05.2023	виконано
9	Підготовка до захисту дипломної роботи, оформлення презентації	16.05-20.05. 2023	виконано

Студент _____ Турчин В.П.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Різник О.Я.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Дипломна робота містить *59* сторінок пояснювальної записки, *29* рисунків, *10* таблиць, *1* додаток, *22* джерел.

Предметом дослідження є спосіб впровадження цифрового водяного знака звукові файли.

Мета роботи - реалізувати вбудовування цифрового водяного знака в аудіофайли методом відлуння.

Результатом є програмна реалізація.

Необхідно провести додаткові дослідження, щоб розширити діапазон типів файлів, в які можна вбудувати цифровий водяний знак.

Ключові слова: стеганографія, звукові файли, захист інформації, приховання інформації, відлуння-сигнали, програмування, реалізація.

ABSTRACT

Thesis contains *59* pages of explanatory note, *29* drawings, *10* tables *1* appendix, *22* sources.

The subject of the study is the method of implementing digital watermarking of sound files.

The purpose of the work is to implement the embedding of a digital watermark in audio files using the echo method.

The result is a software implementation.

More research is needed to expand the range of file types that can be digitally watermarked.

Keywords: steganography, sound files, information protection, information hiding, echo-signals, programming, implementation.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

ACF – автокореляційна функція

ДСТУ – Державний стандарт України

НШВФ – Небезпечні та шкідливі виробничі фактори

Operating System – операційна система

ПК – Персональний електронний комп'ютер

ССЛ – Система слуху людини

США – Сполучені Штати Америки

Флеш-пам'ять – це тип напівпровідникової перезаписуваної енергонезалежної пам'яті (PEPROM).

ЦВЗ – цифровий водяний знак

ISO – Міжнародна організація зі стандартизації, ISO (International Organization for Standardization, ISO) – міжнародна організація, яка розробляє стандарти.

MPEG – (Moving Picture Experts Group - рос. Moving Picture Experts Group) - група фахівців, підпорядкована ISO, орієнтована на розробку стандартів стиснення цифрових зображень і звуку. Перша зустріч відбулася в 1988 році в Ганновері.

SD – це формат картки флеш-пам'яті, призначений для використання в портативних пристроях. Сьогодні він широко використовується в цифрових фотоапаратах, мобільних телефонах, КПК, комунікаторах, GPS-навігаторах та ігрових консолях.

MFC – Microsoft Foundation Classes – це бібліотека C++, розроблена корпорацією Майкрософт і призначена для полегшення розробки програм графічного інтерфейсу користувача для Microsoft Windows за допомогою багатого набору бібліотечних класів.

WAV – (англ. wave «хвиля») – формат файлу-контейнера для зберігання запису цифрового аудіопотоку. У Windows цей формат найчастіше використовується як оболонка для нестиснутого аудіо (PCM), де певна кількість бітів виділяється для кожної вибірки амплітуди сигналу. Однак аудіо, стиснуте майже будь-яким кодеком, можна помістити в контейнер WAV (але відтворення таких файлів може бути проблематичним).

MP3 — (точніше MPEG-1/2/2.5 Layer 3; але не MPEG-3) є третім рівнем формату аудіокодування MPEG, ліцензованого формату файлів для зберігання аудіоінформації.

ВСТУП

Завдання захисту інформації від несанкціонованого доступу вирішувалися протягом всієї історії людства. Вже в стародавньому світі існували два основних напрямки вирішення цієї проблеми і донині: криптографія і стеганографія. Мета криптографії полягає в тому, щоб приховати вміст повідомлень шляхом їх шифрування. Навпаки, стеганографія приховує саме існування таємного повідомлення.

Слово «стеганографія» має грецьке коріння і дослівно означає «написання кодом». Історично цей напрямок з'явився першим, але потім його значною мірою витіснила криптографія. Таємне письмо здійснюється різними способами. Загальною рисою цих методів є те, що приховане повідомлення вкладається в якийсь нешкідливий об'єкт, який не привертає уваги. Потім цей об’єкт публічно транспортується до місця призначення. У криптографії наявність зашифрованого повідомлення сама по собі привертає увагу опонентів, у випадку стеганографії наявність прихованого зв'язку залишається непомітною.

Які тільки стеганографічні методи не використовували люди, щоб захистити свої таємниці! Яскравими прикладами є використання вощених табличок, варених яєць, сірникових коробок і навіть голови раба (повідомлення було прочитано після того, як посланець збрив волосся). У минулому столітті т. зв. гарне чорнило, непомітне за звичайних умов. Приховане повідомлення містилося в деяких листах невинних фраз, переданих шляхом внесення в текст незначних стилістичних, орфографічних чи пунктуаційних помилок. З винаходом фотографії з'явилася технологія мікрофотографії, успішно використана Німеччиною під час світових воєн. Прикладом стеганографії є також заточування карт чітерами.

Під час Другої світової війни уряд США надавав великого значення боротьбі з секретними методами передачі інформації. Введено деякі обмеження на поштову оплату. Листи та телеграми з кросвордами, шаховими ходами, замовленнями на доставку квітів із зазначенням часу та виду квітів не приймаються; стрілки надісланих годинників переведені. Була залучена велика кількість цензорів, які навіть перефразовували телеграми, не змінюючи їх змісту.

Приховування інформації перерахованими способами можливе лише тому, що противник не знає способу приховування. Між тим, ще в 1883 році Кергоф писав, що система захисту інформації повинна виконувати свої функції навіть тоді, коли супротивник повністю знає її структуру та алгоритми функціонування. Весь секрет системи захисту інформації повинен міститися в ключі, тобто в інформації, попередньо (як правило) розділеної між одержувачами. Незважаючи на те, що це правило відомо більше 100 років, і зараз є ті, що його нехтують. Звичайно, їх не можна використовувати в серйозних цілях.

Розвиток комп'ютерних технологій в останнє десятиліття дав новий поштовх розвитку комп'ютерної стеганографії. З'явилося багато нових областей застосування. Повідомлення тепер вбудовані в цифрові дані, як правило, аналогового характеру. Це мова, звукозаписи, зображення, відео. Є також пропозиції щодо вбудовування інформації в текстові файли та файли виконуваних програм.

Щоб перейти до теми вбудовування інформації в аудіосигнали, необхідно вказати вимоги, які можуть пред'являтися до стегосистем для вбудовування інформації в аудіосигнали:

- прихована інформація повинна бути стійкою до наявності різних шумів, стиснення з втратами, фільтрації, аналого-цифрового та цифро-аналогового перетворення;
- прихована інформація не повинна вносити спотворення в сигнал, що приймається слуховим апаратом людини;
- спроба видалити приховану інформацію повинна призвести до помітного пошкодження упаковки (для цифрових водяних знаків (DWM));
- прихована інформація не повинна викликати помітних змін у статистиці контейнера.

Інші типи стеганографії можна використовувати для вбудовування прихованої інформації в аудіосигнали. Наприклад, ви можете вбудовувати інформацію, замінюючи молодші біти (всі або деякі). Або ви можете побудувати стеганосистеми на основі характеристик звукових сигналів і слухового апарату людини.

Слухову систему людини можна розглядати як аналізатор частотного спектру, який може виявляти та розпізнавати сигнали в діапазоні 10-20 000 Гц. Слухову систему людини можна змоделювати як 26 фільтрів передачі, пропускна здатність яких зростає зі збільшенням частоти. Слухова система людини менш здатна розрізняти зміни фази сигналу, ніж зміни амплітуди або частоти.

Звукові сигнали можна розділити на три класи:

- якість телефонного зв'язку, діапазон 300 - 3400 Гц;
- широкосмугова мова 50 - 7000 Гц;
- широкосмугові аудіосигнали 20 - 20000 Гц.

Майже всі звукові сигнали мають характеристику. Кожен з них являє собою досить великий обсяг даних для застосування статистичних методів вбудовування інформації. Перший описаний метод, присвячений цій характеристиці звукових сигналів, працює в часовій області.

Метою дослідження є метод таємного вбудовування інформації в аудіофайли.

Об'єктом дослідження є звукові файли.

Предметом дослідження є характеристики форматів аудіофайлів з погляду вбудовування інформації.

Задачі роботи:

- вбудована інформація повинна бути стійкою до шумів, стиску, фільтрації, аналого-цифрового та цифро-аналогового перетворення;

- спроба видалення вбудованої інформації повинна приводити до пошкодження контейнера;

- вбудована інформація не повинна викликати суттєвих змін у контейнері.

Практична цінність роботи полягає в можливості захисту аудіофайлів від несанкціонованого використання.

РОЗДІЛ 1. СТАН ПРОБЛЕМНОЇ ОБЛАСТІ

1.1. Аналіз методів стеганографічного захисту аудіофайлів на основі перетворення

Алгоритм, запропонований у [1], задовольняє більшості вищевказаних вимог. Цифровий водяний знак вбудовується в аудіосигнали (послідовність 8- або 16-бітових семплів) шляхом невеликої зміни амплітуди кожного семплу. Виявлення ЦВЗ не вимагає оригінального аудіосигналу.

Нехай аудіосигнал складається з N вибірок x(i), i =1, ..., N, де значення N не менше 88200 (що відповідає 1 секунді для стереофонічного аудіосигналу, дискретизованого на частоті 44,1 кГц). Функція f(x(i), w(i)) використовується для вбудовування водяного знака, де w(i) є читанням водяного знака в межах [-α; α], α константа. Функція f повинна враховувати характеристики слухового апарату людини, щоб уникнути помітних спотворень вихідного сигналу. Результуюча кількість сигналів виходить наступним чином:

$$y(i)=x(i)+f(x(i),w(i)) \tag{1.1}$$

Відношення сигнал/шум в цьому випадку розраховується як

$$SNR = 10\log_{10}\frac{\sum_n x^2(n)}{\sum_n [x(n)-y(n)]^2} \tag{1.2}$$

Зверніть увагу, що генератор випадкових чисел, який використовується в схемі, повинен бути рівномірно розподіленим. Стабільність цифрового водяного знака загалом підвищується зі збільшенням енергії цифрового водяного знака, але це збільшення обмежується згори прийнятним співвідношенням сигнал/шум.

Виявлення ЦВЗ відбувається наступним чином. Нехай S буде такою сумою:

$$S = \sum_{i=1}^{N} y(i)w(i). \tag{1.3}$$

Поєднуючи (1.1) і (1.3) отримуємо

$$S = \sum_{i=1}^{N} \left[x(i)w(i) + f(x(i), w(i))w(i) \right]. \tag{1.4}$$

Перша сума в (1.4) дорівнює нулю, якщо числа у виході ГВЧ розподілені рівномірно, а математичне очікування значення сигналу дорівнює нулю. У більшості випадків існує деяка різниця, позначена w Δ, яку також необхідно враховувати. Отже, (1.4) набуває вигляду

$$S = \sum_{i=1}^{N-\Delta w} x(i)\, w(i) + \sum_{i=1}^{\Delta w} x(i)\, w(i) + \sum_{i=1}^{N} f(x(i), w(i))\, w(i) \tag{1.5}$$

Сума $\sum_{i=1}^{N-\Delta w} x(i)w(i)$, як показано вище, приблизно дорівнює нулю. Якщо цифровий водяний знак не був вбудований в звуковий сигнал, то S приблизно дорівнює $\frac{\Delta w}{N}\sum_{i=1}^{N} x(i)w(i)$. З іншого боку, якщо звуковий сигнал надходив із ЦВЗ, то S приблизно дорівнює $\frac{\Delta w}{N}\sum_{i=1}^{N} x(i)w(i) + \sum_{i=1}^{N} f(x(i), w(i))w(i)$. Однак x(i) є вихідним сигналом, який за згодою не можна використовувати в процесі виявлення ЦВЗ. Сигнал можна замінити на y(i), що призведе до обміну $\sum_{i=1}^{\Delta w} x(i)w(i)$ на $\frac{\Delta w}{N} S$, помилка буде незначною. Таким чином, віднімаючи компонент $\frac{\Delta w}{N} S$ з S, та поделив на $\sum_{i=1}^{N} f(y(i), w(i))w(i)$, отримуємо результат r, нормований на 1. Детектор ЦВЗ, використовуваний у цьому методі, обчислює значення r, задане за формулою

$$r \hat{=} \frac{S - \frac{\Delta w}{N}|S|}{\sum_{i=1}^{N} f(y(i), w(i))w(i)}. \quad (1.6)$$

Порогове значення теоретично знаходиться в діапазоні від 0 до 1, з урахуванням апроксимації, цей діапазон зменшується до [0-ε; 1+e]. Виходячи з досвіду, щоб сказати, чи дійсно в сигналі присутній конкретний ОС, граничне значення ОС має бути більше 0,7. Якщо потрібна висока надійність для визначення наявності ЦВЗ в сигналі, поріг необхідно збільшити.

На рис. 1.1 показано емпіричну функцію щільності ймовірності для аудіосигналу з ЦВЗ і без нього. Емпірична функція густини ймовірності звукового сигналу без ЦВЗ показана суцільною кривою, пунктирна крива описує емпіричну густину ймовірності звукового сигналу з вбудованим ЦВЗ. Обидва розподіли були розраховані з використанням 1000 різних значень ЦВЗ для відношення сигнал/шум 26 дБ.

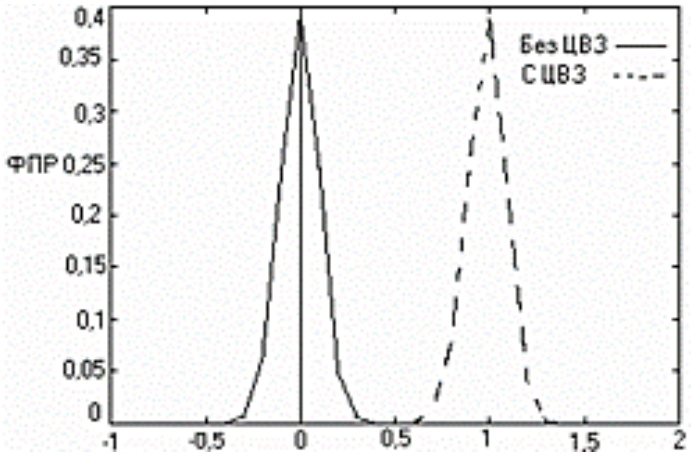


Рис. 1.1. Функція щільності розподілу сигналів за величиною виявлення з CVS та без неї

Використання одного звукового сигналу з великою кількістю різних ЦВЗ призводить до посилення чутності спотворень.

Максимальна кількість ЦВЗ обмежена енергією кожного їх.

Декодер може коректно відтворити будь-який ЦВЗ за умови, що кодувальник використовує унікальні ключі. На рис. 1.2 показано приклад ідентифікації центрального банку з використанням 1000 різних ключів, у тому числі лише одного правильного.

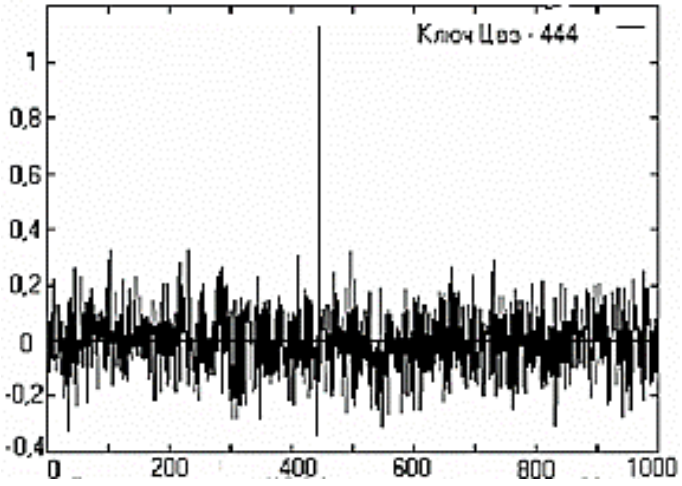


Рис. 1.2. Розпізнавання даного ключа вбудовування ЦВЗ

У вправі [2] була перевірена стійкість методу застосування інформації, що аналізується, до стиснення MPEG на швидкостях від 80 кб/с до 48 кб/с. Після повернення зі стиском до 80 кб/с можна спостерігати незначне зниження граничного значення виявлення звукових сигналів від ЦВЗ (рис. 1.3). При стисканні аудіосигналу до 48 кбіт/с виникають звукові ефекти, які значно знижують якість сигналу.

Стійкість алгоритму вбудовування ЦВЗ до фільтрації була перевірена шляхом застосування до нього фільтра ковзного середніх частот і фільтра нижніх частот. Аудіофайли з реалізованим ЦВЗ фільтруються за допомогою ковзного середньочастотного фільтра довжиною 20, який вносить значні спотворення аудіоінформацію.

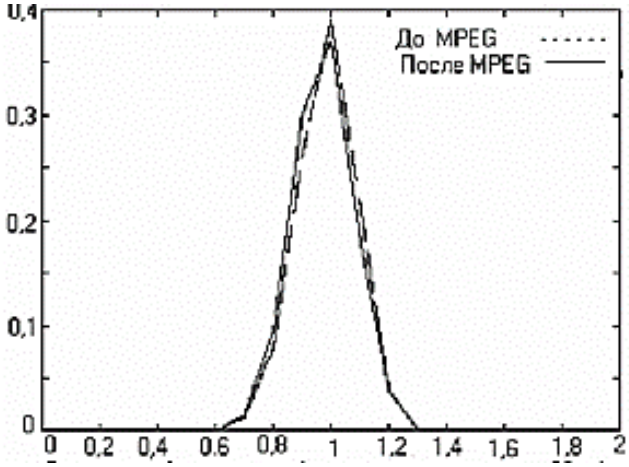


Рис. 1.3. Вплив стиснення даних ЦВЗ

Як правило, поріг виявлення збільшується у відфільтрованих сигналах. Це з тим, що функція щільності розподілу сигналів після фільтрації зміщена вправо стосовно відносної функції розподілу нефільтрованих сигналів.

ЦВЗ зберігається, коли до аудіосигналу застосовується фільтр нижніх частот. Однак при фільтрації звукових сигналів від ЦВЗ низькочастотним фільтром Хеммінга 25-го порядку з частотою зрізу 2205 Гц сталося зниження ймовірності виявлення ЦВЗ.

Щоб перевірити стійкість ЦВЗ до передискретизації Р. Басії та І. Пітаса, аудіосигнали були передискретизовані до 22050 Гц та 11025 Гц та повернуті до вихідної частоти. ЦВЗ зберіг.

При переквантуванні звукового сигналу з 16-бітного на 8-бітне і назад реалізована ЦВЗ зберігається, незважаючи на часткову втрату інформації. На рис. 1.4 показано, наскільки добре зберігається ЦВЗ у 1000 аудіосигналах при їх повторному квантуванні до 8-бітного та назад до 16-бітного.

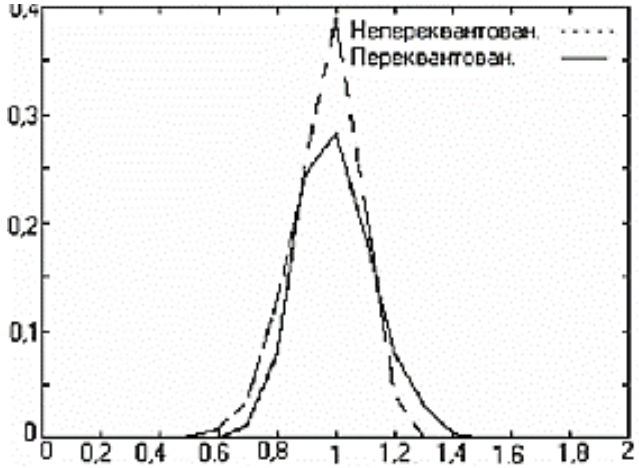


Рис. 1.4. Вплив квантування сигналу на КВН

Відхилення функції густини переквантованого розподілу сигналу збільшується, як і при використанні ФНЧ, тому ефективність виявлення знижується.

1.2. Аналіз методів, заснованих на використанні перетворень

Метод, що передбачає використання слабкої чутливості слухової системи людини до малих змін фази сигналу, було запропоновано, зокрема, Ст Бендер, Н. Моримото.

Введення інформації шляхом зміни фази аудіосигналу є способом, при якому фаза початкового сегмента аудіосигналу модифікується в залежності від вхідних даних. Фаза послідовних сегментів узгоджується з ним, щоб зберегти різницю фаз. Це необхідно, оскільки людське вухо чутливіше до різниці фаз. Фазове кодування, якщо можна застосувати, є одним з найбільш ефективних методів кодування за критерієм відношення сигнал/шум.

Процедура фазового кодування виглядає так:

Звуковий сигнал $s[i], (0 \le i \le I-1)$ ділиться на серію N коротких $s_n[i]\ (0 \le n \le N-1)$ сегментів.

До n-го сегменту сигналу $s_n[i]$ застосовується k-точечне дискретне перетворення Фур'є, де K=I/N, і створюються матриці фаз $\phi_n(w_k)$ та амплитуд $A_n(w_k)$ для $(0 \le k \le K-1)$.

Записується різниця фаз між кожними двома сусідніми сегментами. $(0 \le n \le N-1)$.

$$\Delta\phi_{n+1}(w_k) = \phi_{n+1}(w_k) - \phi_n(w_k) \tag{1.7}$$

Двійкова послідовність даних представлена як π/2 і –π/2, $\phi_0^{/} = \phi_{data}^{/}$.

Враховуючи різницю фаз, для неї створюється нова фазова матриця n>0:

$$\begin{bmatrix} (\phi_1^{/}(w_k) = \phi_0^{/}(w_k) + \Delta\phi_1(w_k)) \\ \dots \\ (\phi_n^{/}(w_k) = \phi_{n-1}^{/}(w_k) + \Delta\phi_n(w_k)) \\ \dots \\ (\phi_0^{/}(w_k) = \phi_{N-1}^{/}(w_k) + \Delta\phi_N(w_k)) \end{bmatrix} \tag{1.8}$$

Стегокодований сигнал, отримують шляхом застосування зворотного дискретного перетворення Фур'є, вихідної амплітудної матриці та модифікованої фазової матриці.

Приймачу необхідно знати: довжину сегмента та точки DPF. Послідовність має бути синхронізована перед декодуванням.

Недоліком цієї схеми є низька пропускна здатність. В експериментах В. Бендера та Н. Морімото смуга пропускання каналу становила від 8 до 32 біт в секунду.

1.3 Аналіз методів стеганографічного захисту аудіофайлів заснованих на використанні надбудови відлуння

Автори запропонували спосіб введення інформації з допомогою відлуння -сигналу.

Цей метод дозволяє вводити дані в сигнал, що маскує, змінюючи параметри луни. Параметри відлуння, що несуть вхідну інформацію: початкова амплітуда, час загасання та зміщення (час затримки між вихідним сигналом та його луною). Коли зсув зменшується, обидва сигнали змішуються. У якийсь момент людське вухо перестає розрізняти ці два сигнали, і відлуння сприймається як додатковий резонанс. Цей момент важко визначити точно, оскільки він залежить від джерела запису, типу звуку та слухача. У загальному випадку, за дослідженнями В. Бендера та Н. Морімото, для більшості видів сигналів і для більшості слухачів злиття двох сигналів відбувається на відстані між ними близько 0,001 секунди.

Кодер використовує дві тимчасові затримки: одну кодування нуля, іншу кодування одиниці. Обидва часи затримки коротші, ніж час, протягом якого людське вухо здатне розпізнавати відлуння. Крім зменшення часу затримки, необхідно забезпечити, щоб інформація, що вводиться, не сприймалася людським слухом, задавши початкову амплітуду і час згасання.

Кодування. Для простоти був обраний приклад всього двох імпульсів (один для копіювання вихідного сигналу, інший для створення ехо-сигналу). Збільшення кількості імпульсів збільшує кількість відлунь.

Нехай на рис. 1.5 показаний «поодинокий» метод кодування, але в рис. 1.5б – «нульовий» метод кодування.

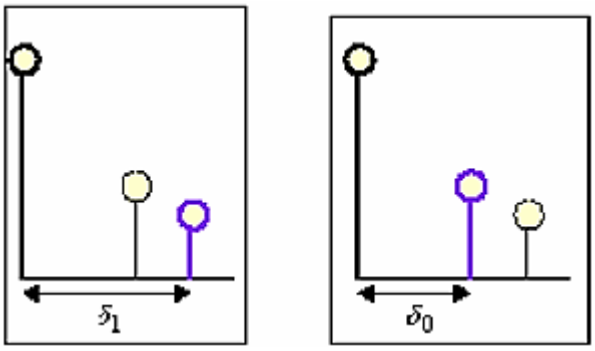


Рис. 1.5. Кодування одного біта інформації (а - одиниця, б - нуль)

Використання даних показано на рис. 1.6.

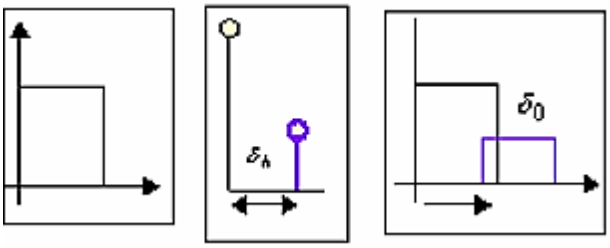


Рис. 1.6. Використання одного біта інформації

Затримка (hδ) між вихідним сигналом та його луною залежить від даних, що вводяться в даний момент. Затримка (1δ) відповідає одиниці, а затримка луна-сигналу (0δ) дорівнює нулю.

Для кодування більше біта вихідний сигнал ділиться на невеликі секції. Кожна секція обробляється як окремий сигнал і до неї вводиться один біт інформації. Результуючий закодований сигнал (що містить кілька бітів вбудованої інформації) є комбінацією окремих ділянок. на рис. 1.11 показаний приклад, де сигнал розбитий на сім ділянок - a, b, c, d, e, f, g..

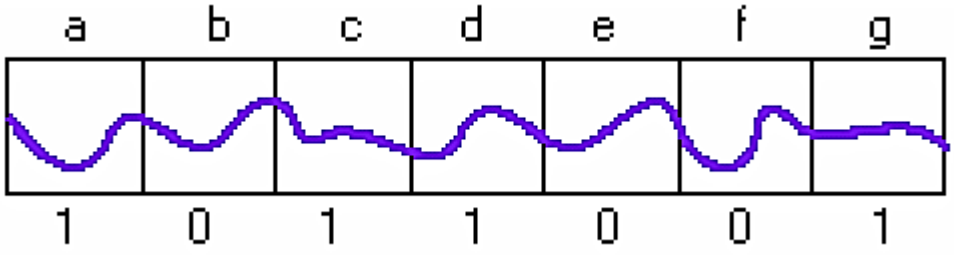


Рис. 1.7. Поділ сигналу на ділянки

Блок вставлятиметься у секції a, c, d, e. Отже, у цих зонах система функціонуватиме так, як показано на рис. 1.7а. У поля b, e, f будуть введені нулі, у цих полях система працюватиме як на рис. 1.7б.

Для мінімальної видимості спочатку створюються два сигнали: один містить лише "одиниці", а інший - тільки нулі. Отримані сигнали показано на рис. 1.8.

Потім створюються два комутаційних сигнали - нульовий та одиничний (рис. 1.9). Кожен з них є двійковим рядком, стан якого залежить від того, який біт потрібно вставити в дану ділянку звукового сигналу.

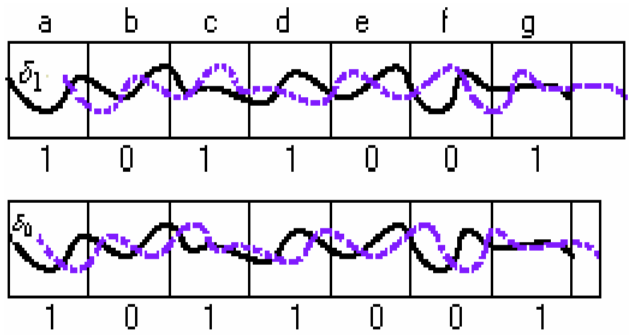


Рис. 1.8. Сигнали, що містять лише одне двійкове значення

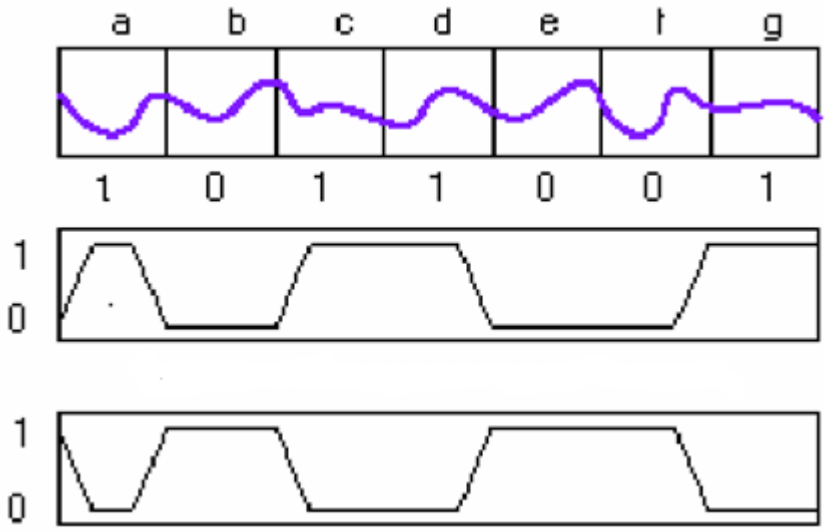


Рис. 1.9. Перемикання сигналів

Потім обчислюється сума творів сигналу нульового мікшування та аудіосигналу з нульовою затримкою, а також сигналу єдиного мікшування та

аудіосигналу з однією затримкою. Іншими словами, коли до звукового сигналу необхідно додати "одиницю", виводиться сигнал затримки "одиниця", інакше виводиться сигнал затримки "нуль". Оскільки сума двох сигналів мікшування завжди дорівнює одиниці, забезпечується плавний перехід між ділянками аудіосигналу, до яких застосовують різні біти.

Розшифровка. Декодування вбудованої інформації полягає у визначенні інтервалу часу між сигналом і луною. Для цього необхідно розглянути амплітуду (у двох точках) автокореляційної функції дискретного косинусного перетворення логарифма спектра потужності (кепстру).

В результаті обчислень кепстра буде отримана послідовність імпульсів (відлуння, що повторюється кожні δ секунд) (рис. 1.10).

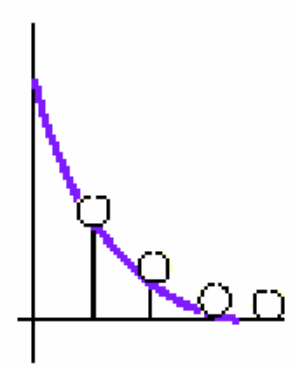


Рис. 1.10. Результат розрахунку кепстра

Для визначення інтервалу часу між сигналом і його відлунням необхідно розрахувати автокореляційну функцію кепстра.

Спалах автокореляційної функції відбудеться через 1 δ або 0 δ секунд після початкового сигналу. Правило декодування засноване на визначенні інтервалу часу між вихідним сигналом і автокореляційним рядом.

Під час декодування «одиниця» приймається, якщо значення функції автокореляції через 1δ секунд більше, ніж через 0δ секунд, інакше воно дорівнює «нулю».

Згідно з дослідженнями В. Бендера і Н. Морімото, ця схема дозволяє непомітно вбудувати 16 біт в одну секунду аудіозапису без втрати якості.

Деякі порівняння вже проводилися такими авторами, як Конахович і Пузиренко [7].

Використовуючи описи даних у попередньому розділі, можна провести порівняння за деякими ознаками, що дозволить зробити вибір на користь того чи іншого способу вбудовування інформації.

Слід зазначити, що метод вбудовування в молодший біт не розглядався через поганий захист від усіх видів атак.

Оскільки методи приховування даних з розширеним спектром і фазового кодування в якості ключових використовують дані області спотворень, які може визначити досвідчений фахівець, використання цих методів не є цілком безпечним.

Використання часу затримки як ключових даних у методі відлуння в поєднанні з даними про області, де інформація прихована, може бути ефективнішим і безпечнішим навіть для спеціалістів.

Якщо оцінювати методи з боку пропускної здатності, то за певними властивостями контейнера фазове кодування перевищує луна-сигнали в 2 рази.Для підвищення пропускної здатності методу луни-сигналів необхідно проводити дослідження в цьому напрямку.

Можна підрахувати місткість звукового файлу середнього розміру (тривалість) - 3 хвилини:

$$16*3*60=2880 \text{ бит}=360 \text{ байт} \quad (1.9)$$

Враховуючи тенденції розвитку інформаційних технологій і збільшення обсягу інформації, яку потрібно повідомити, це дуже мало. Навіть карта micro SD, захована в шкарпетку, є потенційно найкращим рішенням для перенесення інформації. Єдиний недолік – картку неможливо відправити поштою.

Тому потрібно уважно вивчити різні способи підвищення пропускної здатності методу ехо-сигналу.

На закінчення цієї глави можна з упевненістю сказати, що метод вбудовування інформації в аудіофайли за допомогою перетворення ехо-сигналу підходить для захисту аудіофайлів цифровими водяними знаками. Стійкість до амплітудно-частотних атак дозволяє обійти інші методи, нестійкі до цих атак.

РОЗДІЛ 2. ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1. Структура аудіо файлу

Wave-файл включає в себе заголовок, в якому представлений формат та всі характеристики даного аудіо файлу, і також області звукових даних [9].

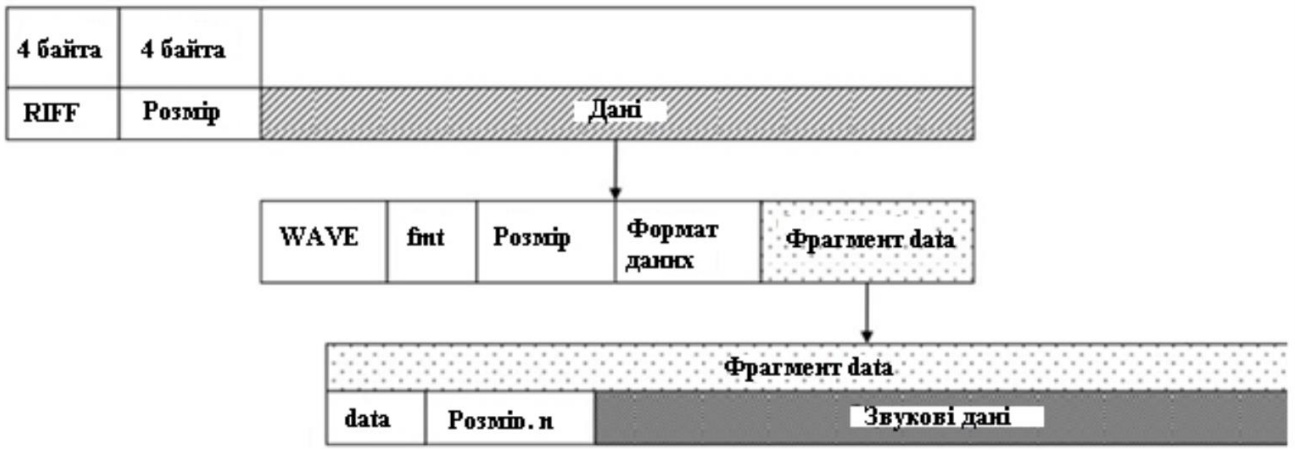


Рис. 2.1. Структура аудіо файлу wave

2.2. Заголовок аудіо файлу

Нижче в таблиці наведено заголовок та опис полів [10] (де PCM – імпульсно-кодова модуляція [11]).

Розташування	Поле	Опис
0..3 (4 байта)	chunkId	Містить символи "RIFF" в ASCII кодуванні (в bigendian представленні). 0x52494646 Є початком RIFF-ланцюжка.
4..7 (4 байта)	chunkSize	Це розмір ланцюжка, який залишився, і починається з цієї позиції. Тобто, це розмір файлу – 8, тут виключені поля chunkId і chunkSize.
8..11 (4 байта)	format	Містить символи "WAVE" 0x57415645 (big-endian представлення)
12..15 (4 байта)	subchunk1Id	Містить символи "fmt " 0x666d7420 (у big-endian представленні)
16..19 (4 байти)	subchunk1Size	16 для формату PCM. Тут це розмір підчіпки, яка залишилась, починаючи з цієї позиції.
20..21 (2 байти)	audioFormat	Аудіо формат. Для PCM = 1 (тобто Лінійне квантування). Значеннями, які є відмінними від 1, позначаються певні формати стиснення. Повний список є у файлі [12].

Розташування	Поле	Опис
22..23 (2 байти)	numChannels	Кількість каналів: моно=1, стерео=2 і т.д.
24..27 (4 байти)	sampleRate	Частота дискретизації: 8000 Гц, 44100 Гц і т.д.
28..31 (4 байти)	byteRate	Кількість байт, які передані за секунду відтворення.
32..33 (2 байти)	blockAlign	Кількість байт для одного семпла, включаючи всі канали.
34..35 (2 байти)	bitsPerSample	Кількість біт у семпле: 8 біт, 16 біт і т.д.
36..39 (4 байти)	subchunk2Id	Містить символи 0x64617461 "data" (у big-endian поданні)
40..43 (4 байти)	subchunk2Size	Кількість байт у сфері даних.
44..	data	Безпосередньо wave-дані.

2.3. Зчитування заголовку аудіо файлу

Для прикладу представлений аудіофайл «WhiteNoise.wav», який є «білим шумом» (рис. 2.2).

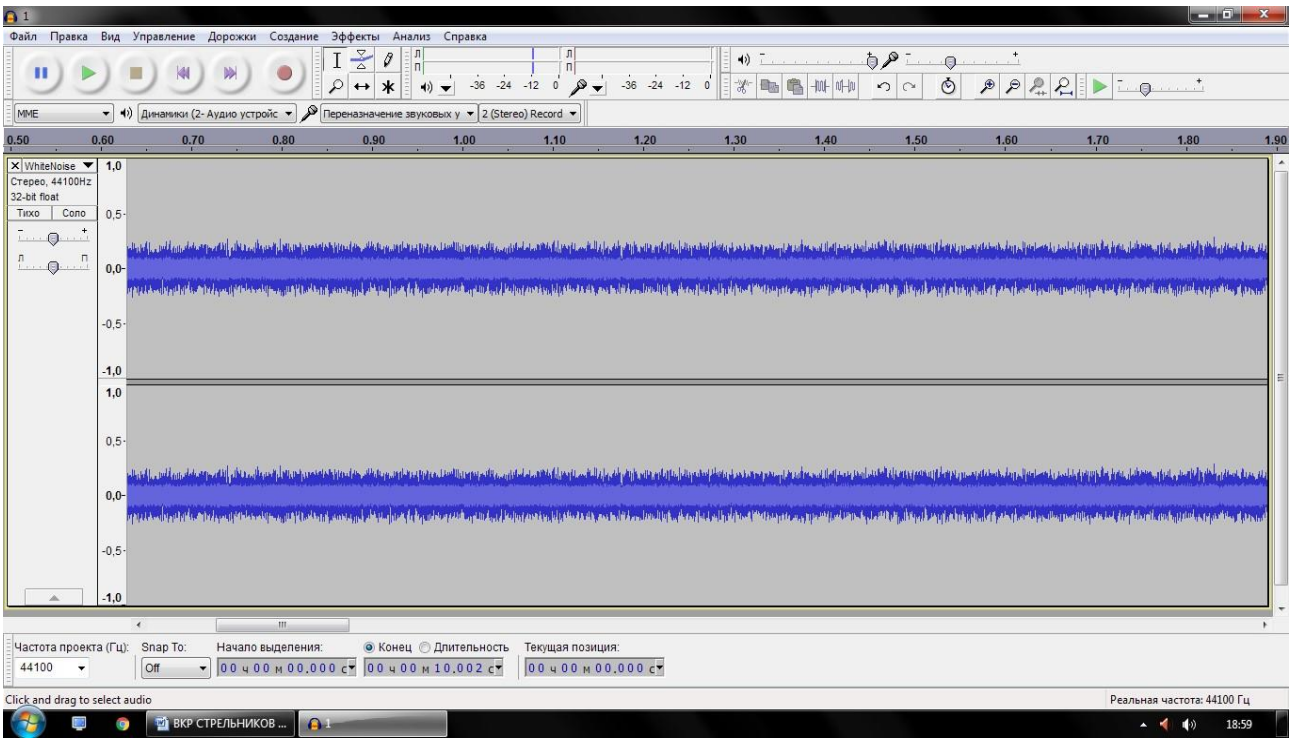
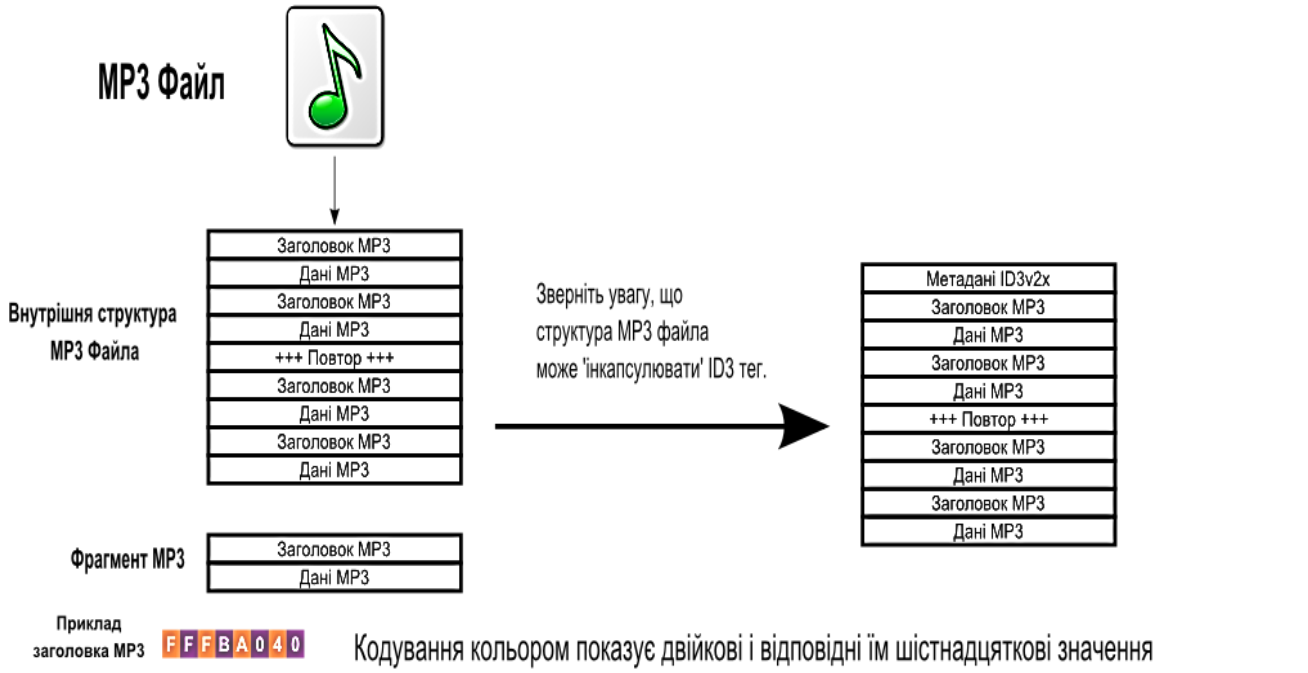


Рис. 2.2. Осцилограма аудіофайлу «WhiteNoise.wav»

1. 0..3 – символи ‘R’ ‘I’ ‘F’ ‘F’.
2. 4..7 – 2538452, розмір файлу в байтах.
3. 8..15 – символи ‘W’ ’A’ ‘V’ ‘E’ ‘f’ ‘m’ ‘t’ ‘ ’.
4. 16..19 – вміщено число 16, яке вказує, що це PCM.
5. 20..21 – число 3, (PCM зі стисненням)
6. 22..23 – число 2 (кількість каналів)
7. 24..27 – число 44100 (частота дискретизації)
8. 28..31 – число 352800 (кількість байт, які передані за секунду)
9. 32..33 – число 8 (кількість байт для одного семпла з включенням всіх каналів
10. 34..35 – число 32 – кількість біт у семпле
11. 36..39 – символи ‘d’ ‘a’ ‘t’ ‘a’
12. 40..43 – число 2538384 – кількість байт у галузі даних

Внутрішню структуру аудіо mp3-файлу показано на рис. 2.3.



Деталі заголовка MP3

Біти	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Біп.	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0
Hex.	F			F			F							B			A						0				4				0	
Позн.	Маркер MP3 фрагмента												Версія	Layer		Захист від помилок	Бітрейт				Частота		Біт вкладень (padding bit)	Служб. біт	Режим		Розширен. режим (використов. для Стерео)		Копія	Оригін.	Акцент (emphasis)	
Значен.	Маркер												1 = MPEG	01 = Layer 3		1 = Вітсутній	1010 = 160				00 = 44100 Гц		0 = Фрагмент без вкладень	Н/Д	01 = Стерео		0 = Інтенс. стерео вимк.	0 = MS стерео вимк.	0 = Без автор. прав	0 = Копія	00 = Нет	

Рис. 2.3. Внутрішня структура аудіо mp3-файлу

2.4. Дерево цілей та проблем реалізації стеганографічних методів

Загальні вимоги до реалізації стеганографічних методів для аудіофайлів буде показано у вигляді дерева цілей (рис. 2.4).

Рис. 2.4. Дерево цілей реалізації стеганографічних методів для аудіо файлів

Роглянемо дерево проблем вбудовання прихованого повідомлення для аудіо файлів (рис. 2.5).

Рис. 2.5. Дерево проблем вбудовання прихованого повідомлення для аудіо файлів

2.5. Дослідження впливу часу затримки відлуння на ефективність стеганографічного захисту інформації

Для повного або хоча б приблизного дослідження впливу часу затримки луна-сигналу на ефективність стеганографічного захисту інформації необхідно зібрати відповідну кількість контейнерів різної якості. Перелік прийнятих для тестування файлів наведено в таблиці 2.1

Таблиця 2.1

Перелік аудіофайлів, використаних у дослідженні

№	Довжина, с	Частота дискретизації, семплів в секунду	Опис
1	3.44	44100	Динамічна електронна музика
2	0.04	16000	Стандартний звук завантаження ОС
3	12.08	96000	Класична музика

Було також прийнято використовувати певний набір затримок для вбудовування бітів даних. Перелічіть їх у таблиці 2.2.

Таблиця 2.2

Перелік тестових затримок

№	Затримка для «0» біта, с	Затримка для «1» біта, с
1	0,0012	0,008
2	0,012	0,08
3	0,003	0,005

З метою дослідження впливу часу затримки відлуння-сигналу на ефективність захисту стеганографічної інформації реалізовано метод вбудовування в середовище САПР Math. Це середовище було вибрано для зручності використання.

У реалізований алгоритм, описаний у [7], внесено зміни відповідно до обраних значень затримки.

Були проведені подальші дослідження, результати яких довгоочікувані. Таким чином, параметри № 1, які повинні бути максимально наближеними до рекомендованих авторами [7], ледве розпізнаються для тренованого вуха.

Параметр 2 виявився абсолютно неприйнятним для використання, незалежно від якості використовуваної тари.

Але параметр 3 показав, що можна використовувати контейнер, подібний за якістю до того, що використовувався в номері 1 (динамічна електронна музика). При використанні цих параметрів затримки ймовірність успішного вилучення досягала 92% (з 1000 спроб - 923 успішних).

Видно, що на даний момент використання контейнера подібної якості до того, що використовувався в № 1, викличе найменшу підозру і бажання перевірити цей контейнер на приховану інформацію. Однак не завжди необхідно захищати аудіофайл цифровим водяним знаком для файлів такого типу.

РОЗДІЛ 3. МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Дослідження впливу амплітуди ехосигналу на продуктивність захист стеганографічної інформації

Для дослідження впливу амплітуди використовувалися ті ж аудіофайли, що і для дослідження впливу часу затримки (табл. 2.2), а значення амплітуди були такими: 30%, 50%, 80%, 100% .

Також слід зазначити, що рекомендоване значення амплітуди накладеного ехо-сигналу становить 80% для сигналу з меншою затримкою та 30% для сигналу з більшою затримкою.

Для наочності вплив значення амплітуди відлуння на кінцевий сигнал було взято з графіків, отриманих у середовищі Math CAD (рис. 3.1 та 3.2).

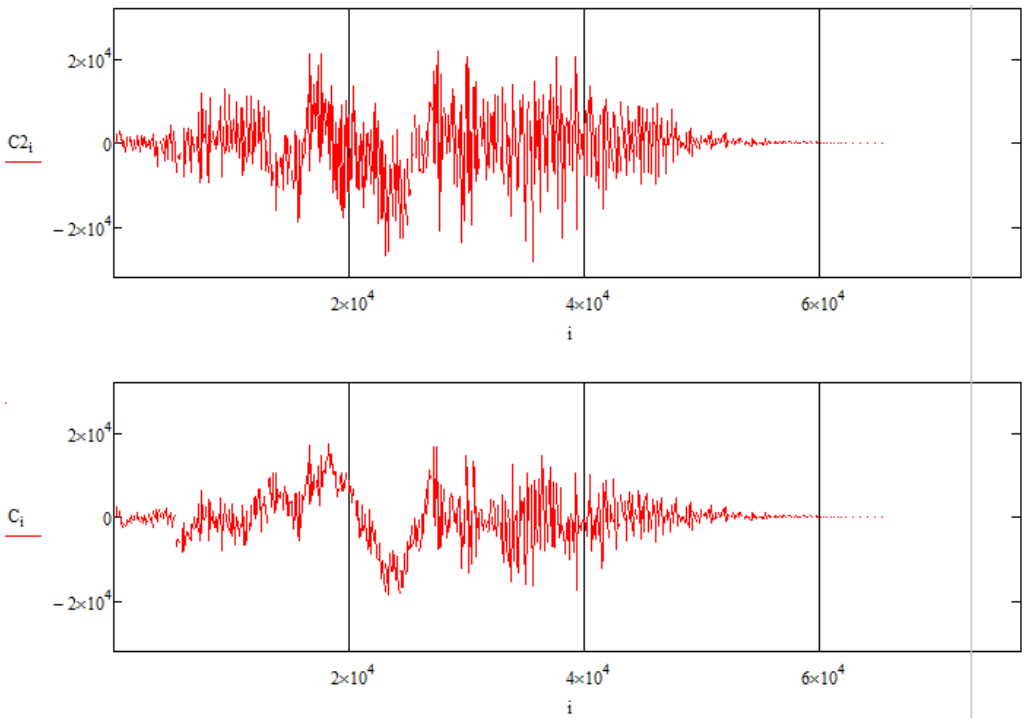


Рис. 3.1. Порівняння результатів вбудовування в аудіофайл без застосування зменшення амплітуди сигналу накладання. Отриманий сигнал вище, вихідний сигнал нижче

Безсумнівно, використання 100% значення амплітуди для відлуння призведе до значної, помітної зміни FFS в аудіофайлі. Також варто відзначити, що зміна значення амплітуди до 30% особливо не впливає на значення ACF значень Cepstra, а отже, не впливає на якість вилучення. У цьому діапазоні можна використовувати значення 50% або навіть 30% без втрати ймовірності правильного вилучення бітів даних.

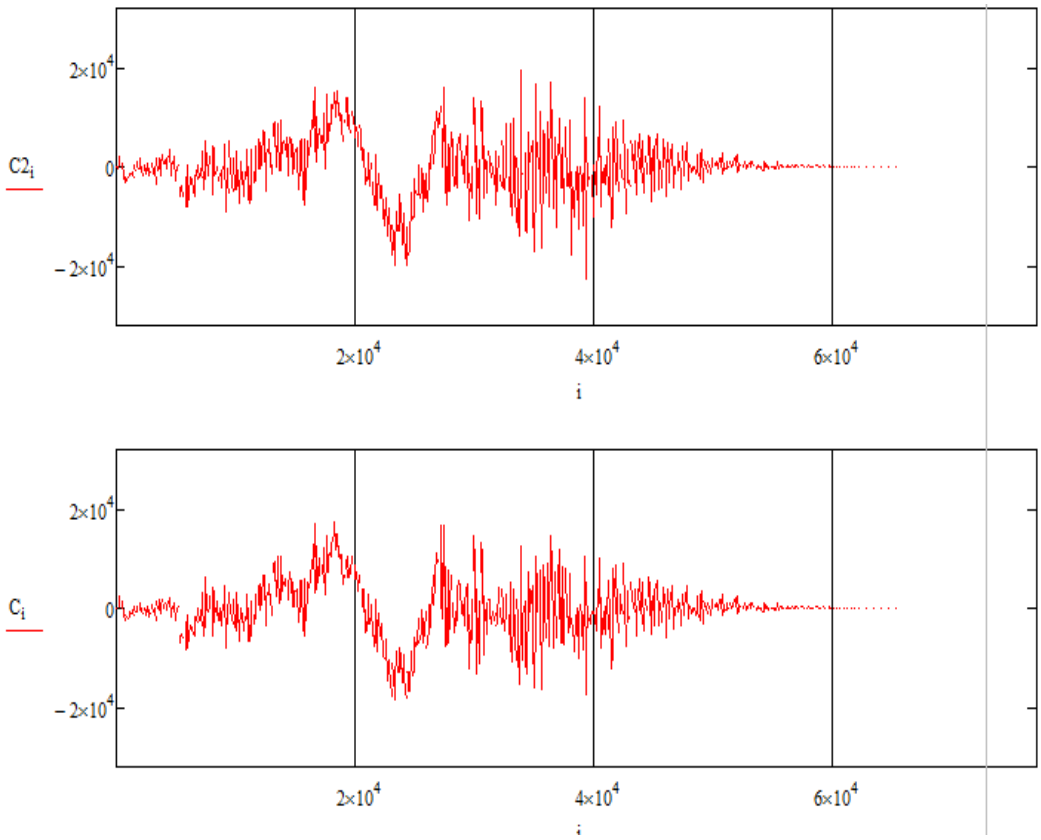


Рис. 3.2. Порівняння результатів вбудовування в аудіофайл шляхом зменшення амплітуди сигналу накладання до 30%. Отриманий сигнал вище, вихідний сигнал нижче

3.2. Обґрунтування параметрів формування відлуння для ефективний захист стеганографічної інформації

Використовуючи результати досліджень впливу величини затримки та амплітуди ехосигналу, доцільно було б відібрати дані, отримані під час цих досліджень.

Однак ці дані не претендують на досконаліть і, можливо, потрібне більш детальне вивчення цих характеристик. У будь-якому випадку, ці дослідження показали, що для подібних типів контейнерів необхідно підбирати певні значення ознак, щоб підвищити ефективність захисту стеганографічної інформації.

Виходячи з результатів тесту, було прийнято використовувати значення затримки відлуння 0,0012 і 0,0008.

Ці значення знижують ефективність алгоритму вилучення, трохи знижуючи ймовірність правильного вилучення біта, але підходять майже для всіх типів бункерів. Значення амплітуди 30% було вибрано, оскільки це допомагає збільшити ймовірність того, що наявність прихованої інформації не буде виявлено.

Як видно на рис. 3.3 вилучення, навіть при зниженні амплітуди до 30%, цілком реально.

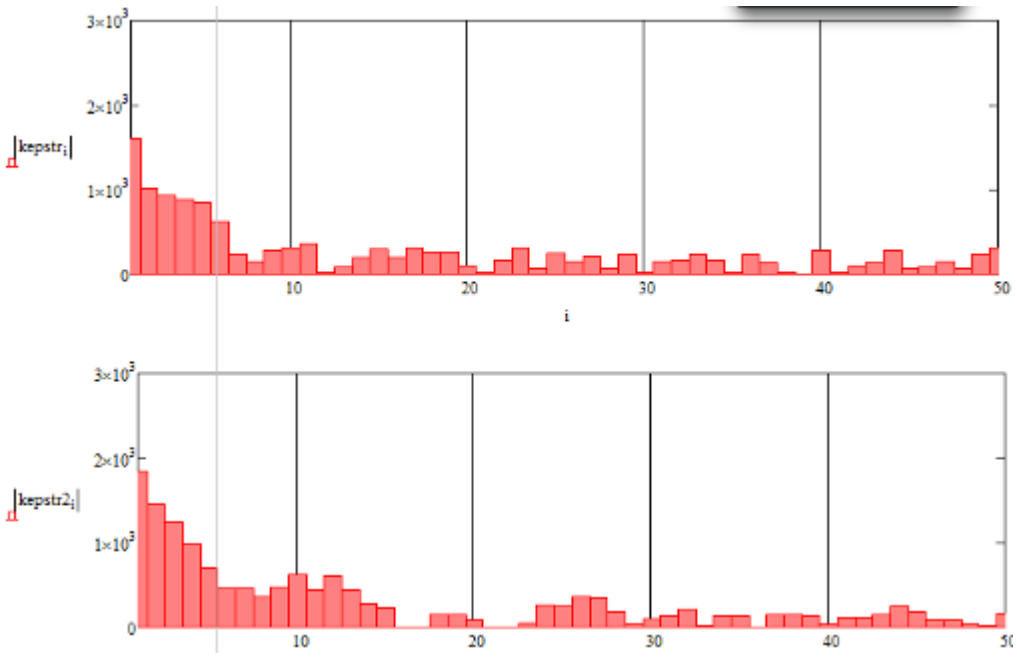


Рис. 3.3 – Порівняння результатів Cepstra ACF до та після вбудовування одного біта при значенні амплітуди 30%.

Проведене дослідження не претендує на ідеальність і правду, т.к. оцінюється однією особою. Тим не менш, застосування отриманих рекомендацій, безумовно, допоможе досягти певної високої ефективності методу вбудовування інформації в звукові файли за допомогою перетворення ехо-сигналу.

3.3 Дослідження впливу амплітуди ехосигналу на продуктивність захист стеганографічної інформації

Для дослідження впливу амплітуди використовувалися ті ж аудіофайли, що і для дослідження впливу часу затримки (табл. 2.2), а значення амплітуди були такими: 30%, 50%, 80%, 100% .

Також слід зазначити, що рекомендоване значення амплітуди накладеного ехо-сигналу становить 80% для сигналу з меншою затримкою та 30% для сигналу з більшою затримкою.

Для наочності вплив значення амплітуди відлуння на кінцевий сигнал було взято з графіків, отриманих у середовищі Math CAD (рис. 2.1 та 2.2).

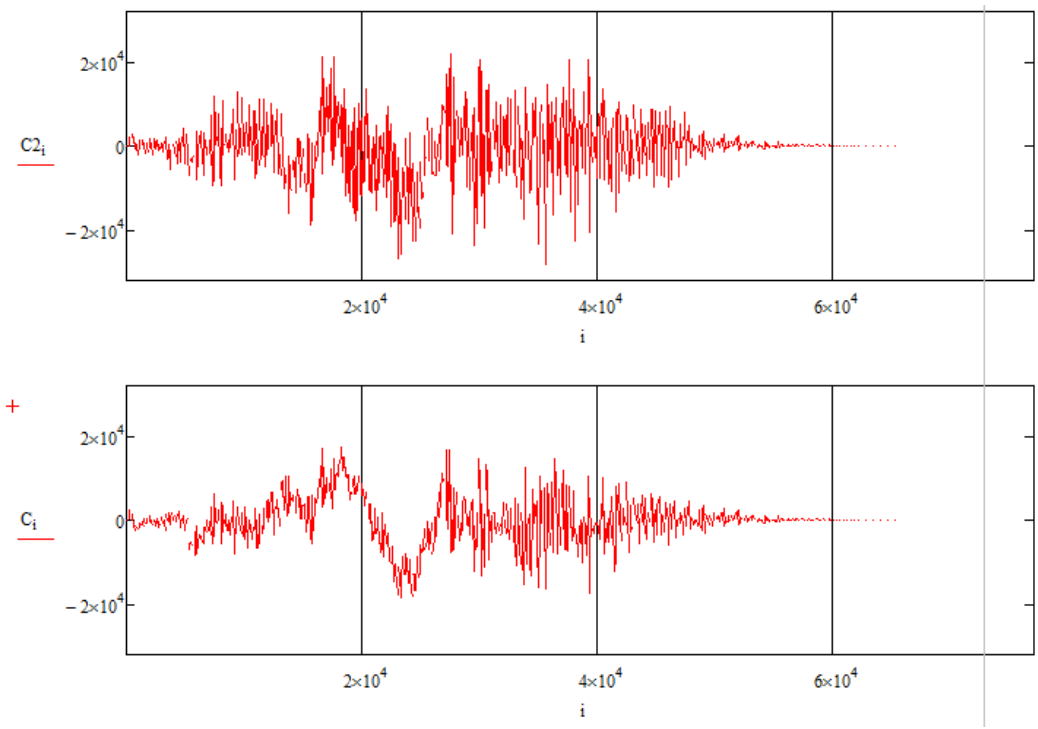


Рис. 3.4. Порівняння результатів вбудовування в аудіофайл без застосування зменшення амплітуди сигналу накладання. Отриманий сигнал вище, вихідний сигнал нижче

Безсумнівно, використання 100% значення амплітуди для відлуння призведе до значної, помітної зміни FFS в аудіофайлі. Також варто відзначити, що зміна значення амплітуди до 30% особливо не впливає на значення ACF значень Cepstra, а отже, не впливає на якість вилучення. У цьому діапазоні можна використовувати значення 50% або навіть 30% без втрати ймовірності правильного вилучення бітів даних.

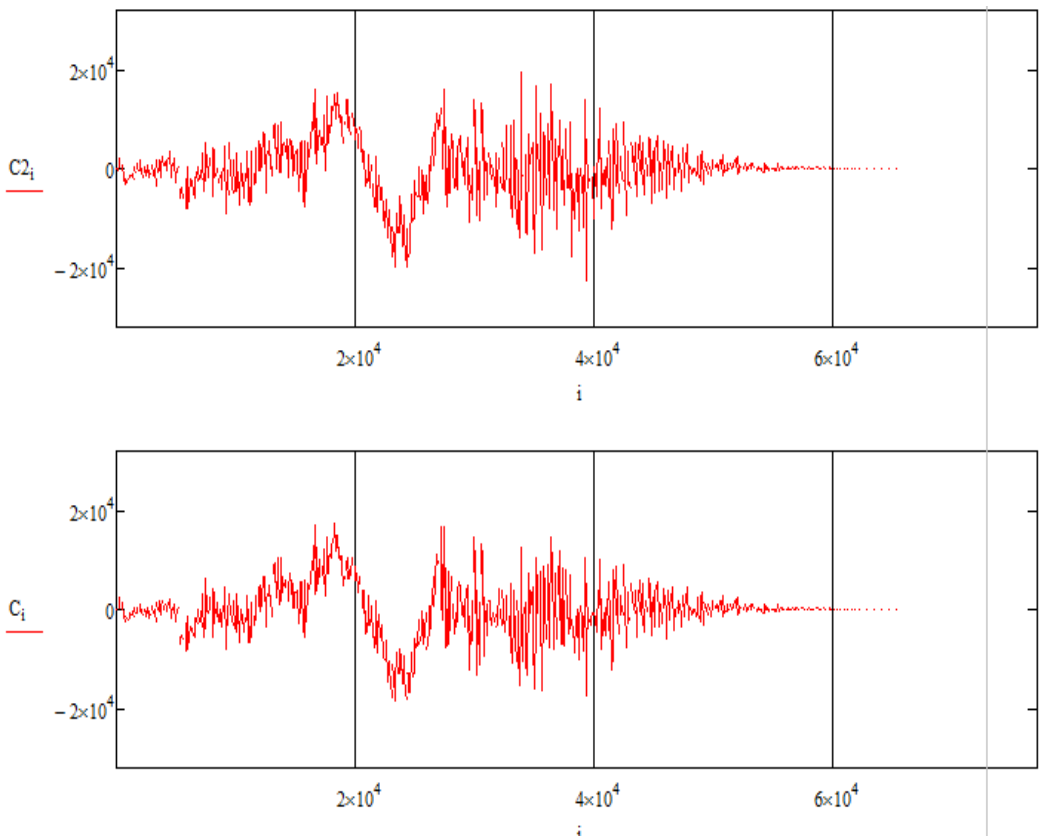


Рис. 3.5. Порівняння результатів вбудовування в аудіофайл шляхом зменшення амплітуди сигналу накладання до 30%. Отриманий сигнал вище, вихідний сигнал нижче

3.4. Обґрунтування параметрів формування відлуння для ефективний захист стеганографічної інформації

Використовуючи результати досліджень впливу величини затримки та амплітуди ехосигналу, доцільно було б відібрати дані, отримані під час цих досліджень.

Однак ці дані не претендують на досконаліть і, можливо, потрібне більш детальне вивчення цих характеристик. У будь-якому випадку, ці дослідження показали, що для подібних типів контейнерів необхідно підбирати певні

значення ознак, щоб підвищити ефективність захисту стеганографічної інформації.

Виходячи з результатів тесту, було прийнято використовувати значення затримки відлуння 0,0012 і 0,0008. Ці значення знижують ефективність алгоритму вилучення, трохи знижуючи ймовірність правильного вилучення біта, але підходять майже для всіх типів бункерів. Значення амплітуди 30% було вибрано, оскільки це допомагає збільшити ймовірність того, що наявність прихованої інформації не буде виявлено.

Як видно на рис. 3.6 вилучення, навіть при зниженні амплітуди до 30%, цілком реально.

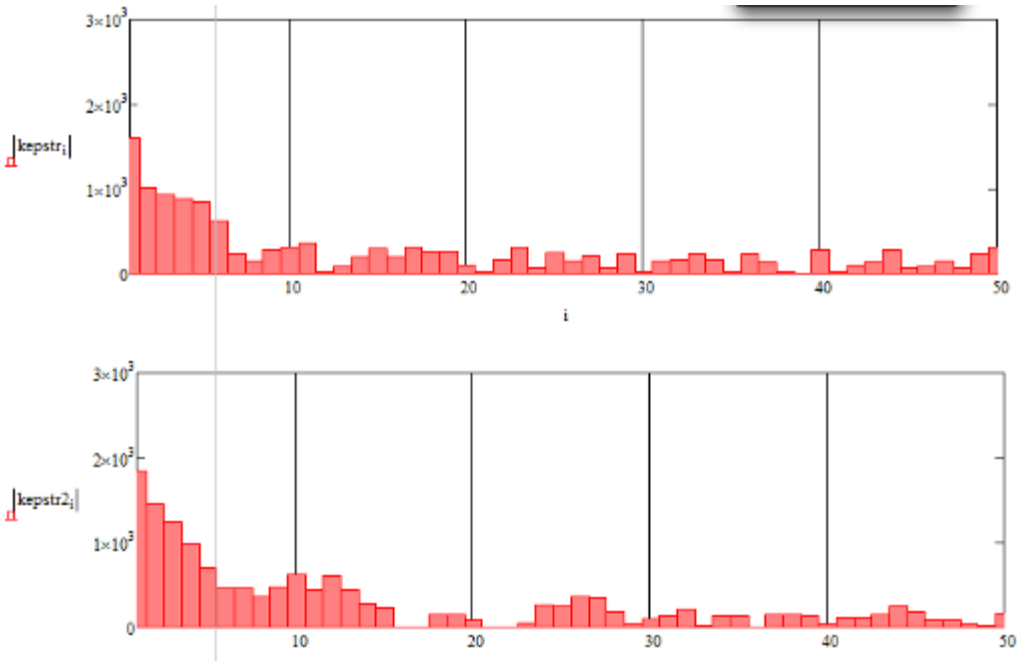


Рис. 3.6. Порівняння результатів Cepstra ACF до та після вбудовування одного біта при значенні амплітуди 30%.

Проведене дослідження не претендує на ідеальність і правду, так як. оцінюється однією особою. Тим не менш, застосування отриманих рекомендацій, безумовно, допоможе досягти певної високої ефективності

методу вбудовування інформації в звукові файли за допомогою перетворення ехо-сигналу.

3.5. Розроблений підхід стегоаналізу

Запропонований підхідбазується на гіпотезі невипадковості молодших значущих бітів аудіофайлів. Хоча існує думка, що LSB (Least Significant Bits) аудіофайлів випадковий.

Насправді це не так. Хоча людське вухо не помітить зміни у звуковому файлі після зміни останніх бітів, статистичні параметри звукового файлу будуть змінені.

Перед зберіганням дані зазвичай архівуються (щоб зменшити обсяг) або шифруються (щоб краще захистити повідомлення від потрапляння в чужі руки). Це робить біти даних дуже близькими до випадкових. У цьому випадку послідовне вбудовування такої інформації замінить NZB аудіофайлу на випадкові біти. Це головна суть підходу - фіксація різниці між розташуванням молодших бітів у порожньому та заповненому контейнері.

Щоб перевірити цей підхід, була розроблена спеціальна методологія з використанням набору статистичних тестів NIST, розробленого Національним інститутом стандартів і технологій. (США, 2014).

Тести пакетів NIST були розроблені для статистичного тестування двійкових послідовностей на випадковість. Ці тести покладаються на статистичні властивості, якими володіють лише випадкові послідовності.

Для розробки методики була створена тестова база даних із 108 різних типів аудіофайлів, припускаючи їх використання як контейнер. База містила шуми (білий, чорний, коричневий, рожевий, зелений тощо), рівні синусоїди, записи мови, інструментів (гітара, саксофон, барабани тощо), а також

повноцінні музичні композиції. База даних складається з файлів з різними бітрейтами (8, 16, 24, 32 біти) і частотами дискретизації. Розмір файлів від 38 КБ до 41 522 КБ.

Щоб відстежувати випадковість низьких бітів, вам потрібно відсортувати файли за деякими атрибутами. Тому було обрано відносне число нуля b у файлі. Відносна кількість нульових байтів визначається як відношення нульових байтів файлу до загальної кількості байтів. За цією функцією було замовлено всі 108 тестових файлів.

Щоб перевірити гіпотезу, частково заповнені файли (крім порожніх контейнерів) були додані до вихідної бази даних, а саме всі файли з бази даних були вкладені в stego (за допомогою розробленого програмного забезпечення Stegora WaveHide) на 10%, 50% і 100% взято в враховувати максимальну місткість стегоконтейнера.

Як тест використовувався бітовий тест частоти пакетів NIST [26], який оцінює відношення нулів до одиниць у двійковій послідовності. Алгоритм цього тесту такий [25]: файл розглядається як послідовність бітів, одиниця як +1, нуль як 0. Враховується сума послідовностей. Потім статистика обчислюється за формулою:

$$S_{obs} = \frac{|S|}{\sqrt{n}}, \tag{3.1}$$

де |S| – сума послідовності,

n – кількість елементів у послідовності.

Обчислюється P-значення через додаткову функцію помилок:

$$P_{value} = erfc(\frac{S_{obs}}{\sqrt{2}}), \tag{3.2}$$

Додаткова функція помилок обчислюється так:

$$erfc(x) = \frac{2}{\sqrt{\pi}} \int_{x}^{\infty} e^{-t^2} dt \tag{3.3}$$

А якщо результат буде більше 0,01, то така послідовність буде випадковою з імовірністю 99%.

3.6. Метод стегоаналізу аудіо файлів на основі стиснення

Щоб перевірити порівняльну продуктивність запропонованої методики стеганалізу, ми реалізували добре відомий метод стеганалізу, заснований на алгоритмі стиснення, описаному в [8].

3.6.1. Особливості стегоаналізу за алгоритмом стиснення

В роботі [8]:

- не вказано кількість змінених молодших бітів.
- немає інформації про тип перевірених звукових файлів.
- швидше за все використовувалися звукові файли подібного типу, тому даний алгоритм не можна вважати універсальним.
- використовувалися непопулярні на даний момент 8- і 16-бітні файли wave.

Зараз більше підходять аудіофайли з бітрейтом 24 і вище..

3.6.2. Реалізація методу стегоаналізу за алгоритмом стиснення

Після впровадження з наведеними вище коментарями приймається:

- види досліджуваних файлів;
- кількість молодших біт, що замінюються на випадкові – 2 (як це було зроблено у стеганографічній частині розробленого додатку);
- бітність файлів від 8 до 32.

У розробленому програмному забезпеченні даний метод реалізовано таким чином:

В інтерфейсі розробленого пакету Stegora WaveHide після натискання кнопки «Альтернативний аналіз» і введення коефіцієнтів сигма і K користувач повинен вибрати файл для аналізу. Спочатку файл розбивається на K рівних частин. Потім кожна частина стискається за допомогою методів стандартної бібліотеки zlib і обчислюється співвідношення розміру стисненого файлу до стисненого. Після зміни двох молодших бітів у вихідному файлі на випадкові біти та повторення процедури обчислюється відношення розміру стисненого до стисненого фрагмента. Для кожного шматка розраховується параметр Δ. Якщо кількість елементів, для яких параметр Δ нижче введеного коефіцієнта сигми, більше половини, то приймається рішення про наявність стего вкладень у розглянутому файлі. В іншому випадку приймається рішення не здійснювати такі операції.

3.7. Висновок

Запропонований підхід базується на випадковості молодших значущих бітів аудіофайлів. У цьому разі людське вухо не помітить зміни у звуковому файлі після зміни останніх бітів. Це головна суть підходу - фіксація практично відсутності різниці між розташуванням молодших бітів у порожньому та заповненому контейнері.

РОЗДІЛ 4. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

4.1 Розробка алгоритмів вбудовування та вилучення цифрових даних водяний знак до аудіофайлу

Щоб вбудувати та витягти цифровий водяний знак у аудіофайл, необхідно розробити два алгоритми, які можна реалізувати мовою програмування C++.

Почнемо з алгоритму вбудовування:

Перетворення вбудованого повідомлення в двійковий код;

Перевірте місткість контейнера та розмір вбудованого повідомлення. Якщо повідомлення не поміщається в контейнер, вийти з помилкою;

Розділіть контейнер на рівні частини, залежно від частоти дискретизації, щоб забезпечити пропускну здатність 16 біт на секунду;

Візьміть першу частину ємності;

Створити ехо-сигнал шляхом просування вихідного сигналу на значення затримки поточного біта;

Зменшити амплітуду відлуння до 30%. Також потрібно створити спадаючий і висхідний фронти в кінці і початку ехо-сигналу, відповідно, для запобігання різких переходів, чутної FFS;

Накласти відлуння-сигнал на вихідний сигнал;

Виконайте кроки 5-7 для всіх бітів вбудованого повідомлення.

Алгоритм вилучення (насправді це алгоритм перевірки цифрових водяних знаків):

- перетворення повідомлення для перевірки у двійковий код;
- перевірте місткість контейнера та розмір перевіреного повідомлення. Якщо повідомлення не поміщається в контейнер, вийти з помилкою;

- розділіть контейнер на рівні частини, залежно від частоти дискретизації, щоб забезпечити пропускну здатність 16 біт на секунду;
- візьміть першу частину ємності;
- розрахувати Cepstra ACF для поточного фрагмента контейнера;
- порівняйте значення ACF Cepstra, що відповідають часу, рівному значенню зсуву exo-сигналів. Якщо значення на рівні зсуву «1» більше за «0», то біт 1 був вбудований;
- порівняти отриманий біт з відповідним бітом перевіреного повідомлення;
- повторіть кроки 5–7 для всіх бітів повідомлення, яке перевіряється.
- крім того, якщо всі біти збігаються (ви можете перевірити відсоток збігу, тому що метод exo-сигналу не має 100% вірогідності правильного виділення бітів), ми робимо висновок, що аудіофайл захищено.

4.2. Структура програмного продукту

Розглянемо програмну структуру пакету (рис. 4.1).

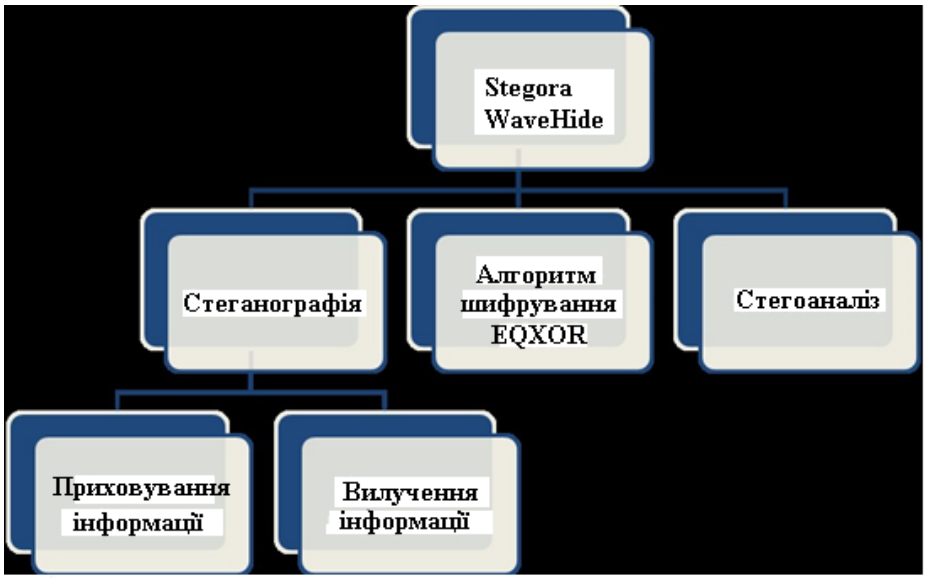


Рис. 4.1. Програмна структура пакету

4.2. UML-диаграма класів

У цій системі є два класи, що реалізують цей інтерфейс: File, Text.

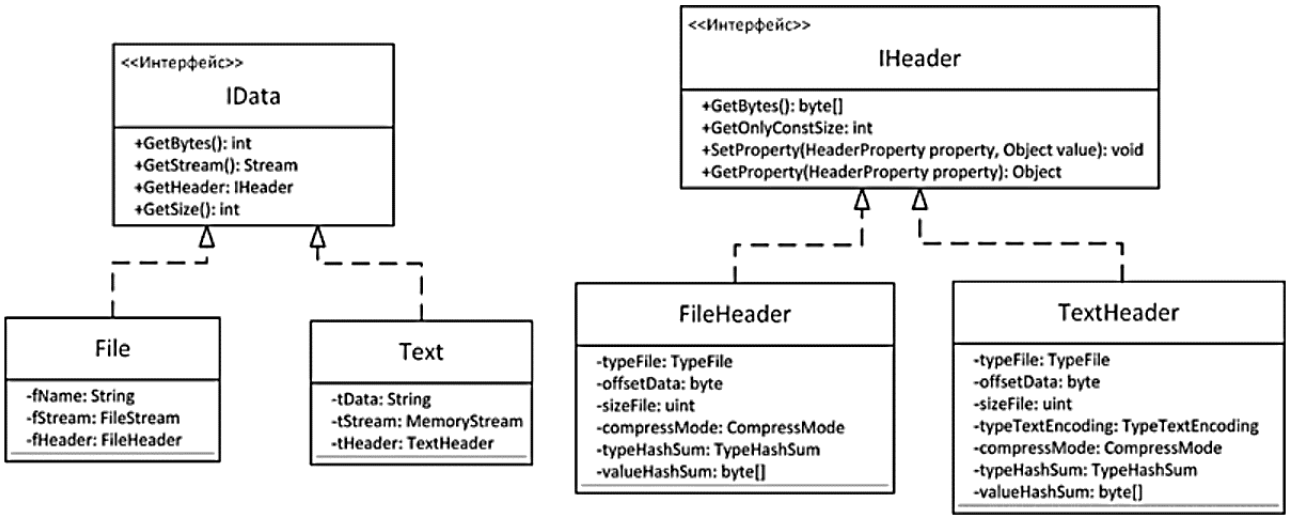


Рис. 4.2. Діаграма класів

Розглянемо структури класу FileHeader (табл. 4.1).

Таблиця 4.1.

Структура класу FileHeader

Ім’я поля	Кількість байт	Опис
typeFile	1	Тип документу, що впроваджується
offsetData	1	Зміщення початку даних (розмір заголовку)
sizeFile	4	Розмір файлу (кількість байт)
compressMode	1	Режим стиснення
typeHashSum	1	Тип хеш-суми
valueHashSum	*n*	Значення хеш-суми

4.3. Опис інтерфейсу розробленої програмної реалізації методу захисту стеганографічної інформації

Для реалізації стеганографічного методу захисту інформації звукового файлу за допомогою ехо-сигналів обрано мову програмування C++. Середовищем розробки є Microsoft Visual Studio 2008. Для спрощення створення інтерфейсу було вирішено використовувати бібліотеку програмування MFC, надану Microsoft для програмування під Windows. Інтерфейс програми наведено на рис. 4.2 - 4.4.

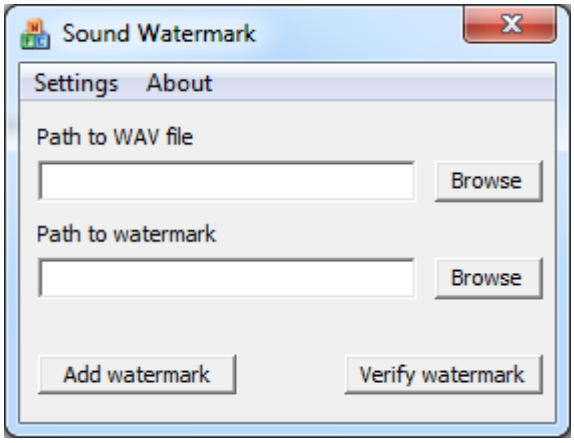


Рис. 4.2. Інтерфейс звукового водяного знака.

Щоб вставити водяний знак в аудіофайл, ви повинні:

Виберіть аудіофайл для вбудовування, ввівши шлях вручну, вибравши файл у вікні вибору файлу, яке з’явиться після натискання кнопки «Огляд»;

Виберіть файл водяного знака (наразі підтримуються лише текстові файли у форматі txt);

Натисніть кнопку «Додати водяний знак».

Після успішного вбудовування користувач побачить повідомлення, показане на рис. 4.3.

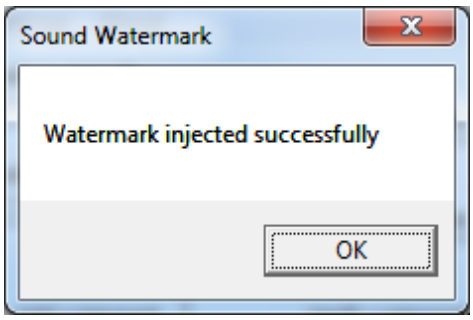


Рис. 4.3. Водяний знак успішно вбудовано в аудіофайл

Щоб перевірити аудіофайл на наявність цифрового водяного знака, необхідно:

Виберіть аудіофайл для перевірки;

Виберіть файл CEH;

Натисніть кнопку Перевірити водяний знак.

Після успішного вилучення користувач побачить повідомлення, показане на рис. 4.4.

Рис. 4.4. Успішно витягнутий цифровий водяний знак відповідає посиланню

Для спрощення використання програми додано такі функції:

- обмеження використання вхідних файлів.
- не можна використовувати файли, відмінні від формату WAV (PCM16 або PCM32). При використанні файлів у форматі, відмінному від цього, користувач побачить вікно з помилкою (рис. 4.5). Крім того, коли ви

натискаєте кнопку перегляду для вибору аудіофайлу, вибір буде обмежено лише файлами wav (рис. 4.5);

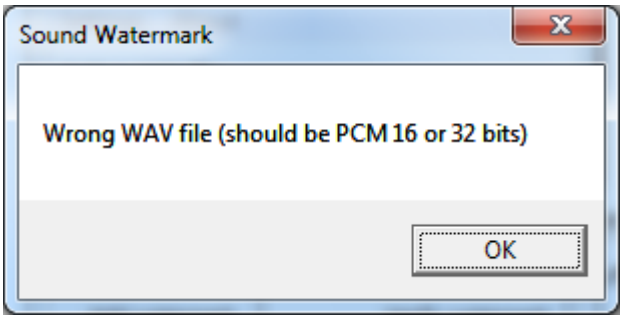


Рис. 4.5. Помилка формату файлу неправильного контейнера.

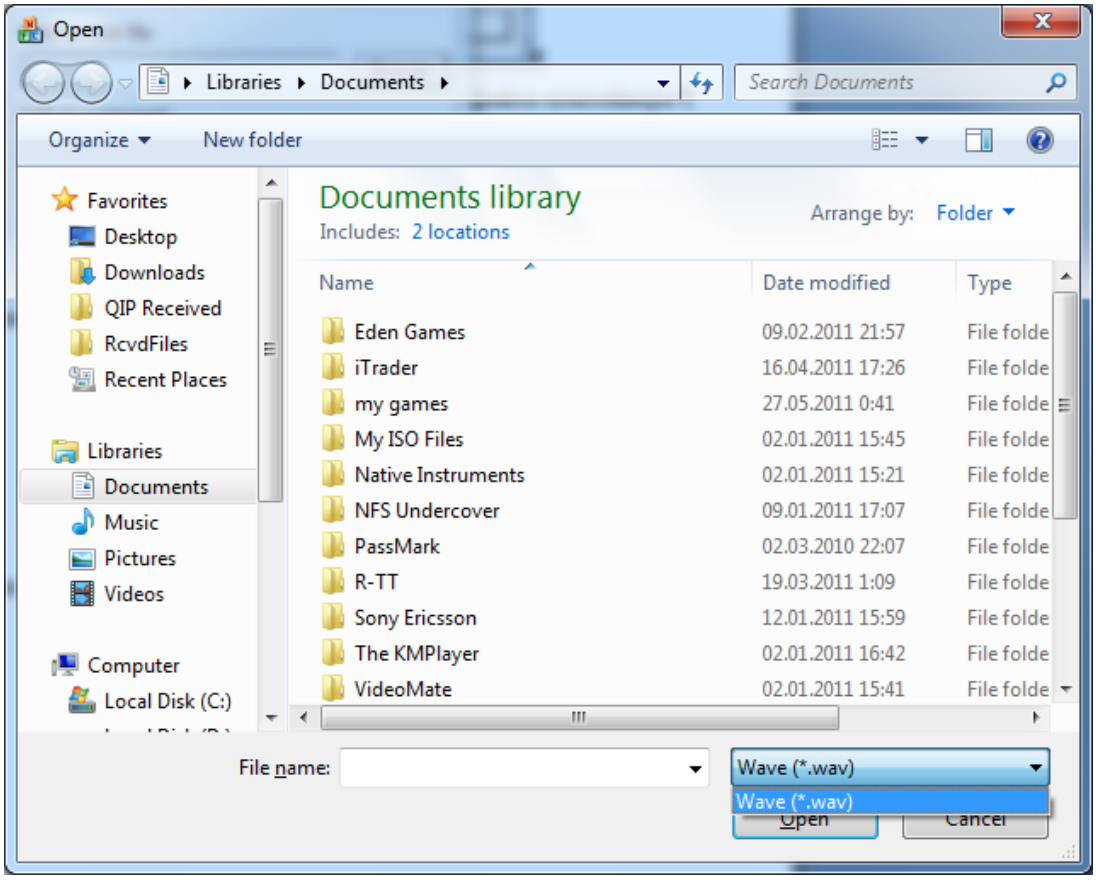


Рис. 4.6. Вікно вибору аудіофайлу з обмеженням типу файлу.

- - можливість простого перетягування файлу в поле введення шляху (drag and drop);
- - при спробі вставити або перевірити водяний знак без вибору файлів з'явиться помилка (рис. 4.7).

Рис. 4.7. Помилка щодо порожніх полів введення шляху до файлу

4.4. Практичні рекомендації щодо використання та вдосконалення програмної реалізації

Загалом програма має завершений вигляд, але є багато способів її розвитку.

Можна значно розширити межі застосування цієї програми, реалізувавши алгоритм вбудовування цифрових водяних знаків для більш популярного формату MP3. Існує багато бібліотек з відкритим кодом, які дозволяють отримати з файлу MP3 по суті те саме, що ви можете отримати з файлу WAV без бібліотек – значення амплітуди. Після отримання не складе труднощів реалізувати алгоритм вбудовування цифрового водяного знака в аудіофайли MP3.

Також необхідно переробити механізм вбудовування, щоб додати можливість використовувати будь-які файли, а не тільки текстові.

На додаток до вдосконалення реалізації програмного забезпечення, алгоритм вбудовування можна вдосконалити, щоб збільшити ймовірність

правильного вилучення бітів вбудованого повідомлення. Цього можна досягти шляхом реалізації алгоритму зміни пропускної здатності файлу в залежності від якості використовуваного контейнера.

Сьогодні наявність такого програмного комплексу покращить стан справ на фронті боротьби з піратством серед аудіофайлів.

Програма розгорнута та готова до використання. Можливість покращувати його та випускати додаткові версії є хорошою основою для комерційного використання.

4.5. Тестування програмного продукту

За допомогою вищезазначеного тесту було перевірено випадковість послідовності молодших бітів із усіх 100 файлів з бази даних.

Результати представлені на рис. 4.8. – 4.10.

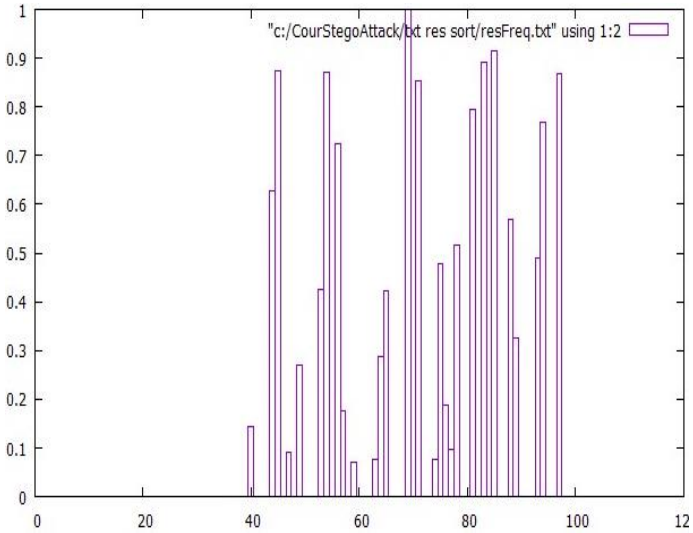


Рис. 4.8. Перевірка на випадковість послідовності LSB 100 аудіо файлів

Як показано на рис. 4.8, послідовності LSB для файлів з відносною кількістю нульових байтів менше 0,08 (значення для файлу 40) не є випадковими.

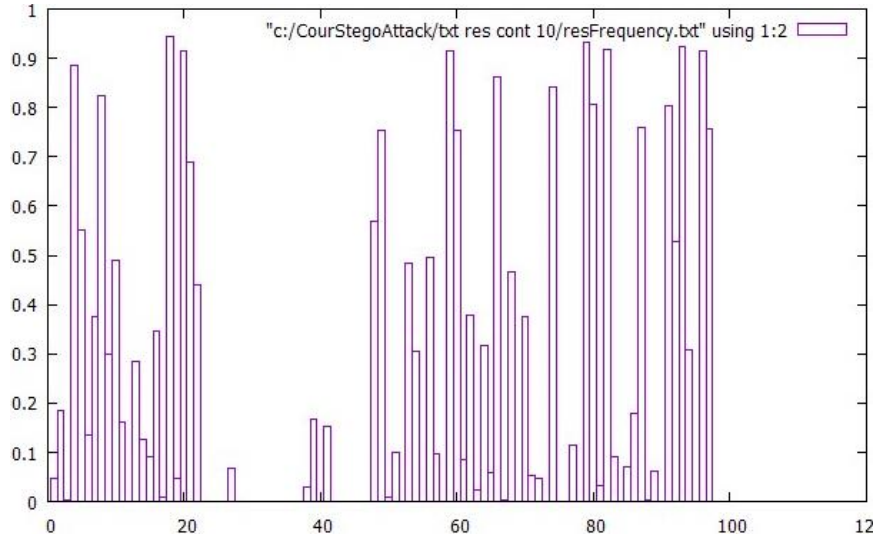


Рис. 4.9. Перевірка на випадковість послідовності LSB 100 аудіо файлів, заповнених на 10% максимальної можливості.

З рис. 4.9. ви можете побачити, що з 10% смугою послідовності LSB для файлів з відносною кількістю нульових байтів менше 0,0348 (значення для 22 файлів) є випадковими.

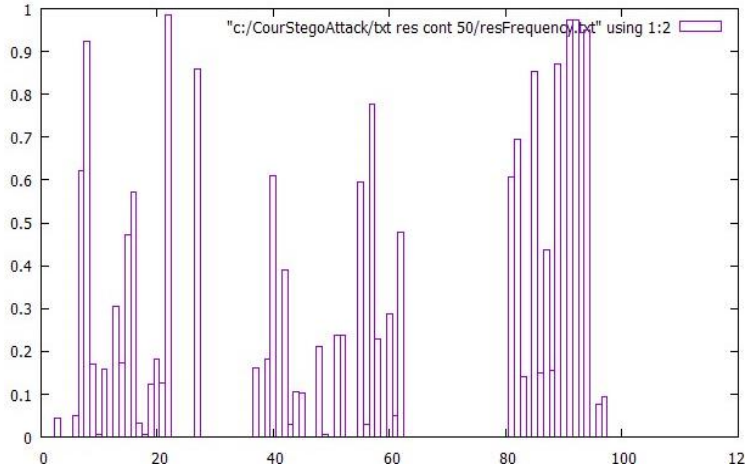


Рис. 4.10. Перевірка на випадковість послідовності LSB 100 аудіо файлів, заповнених на 50% максимальної можливості.

З рис. 4.10. можна побачити, що на п’ятдесят відсотків вражаючі послідовності LSB у файлах з відносною кількістю нульових байтів менше 0,0348 (значення для 22-го файлу) є переважно випадковими.

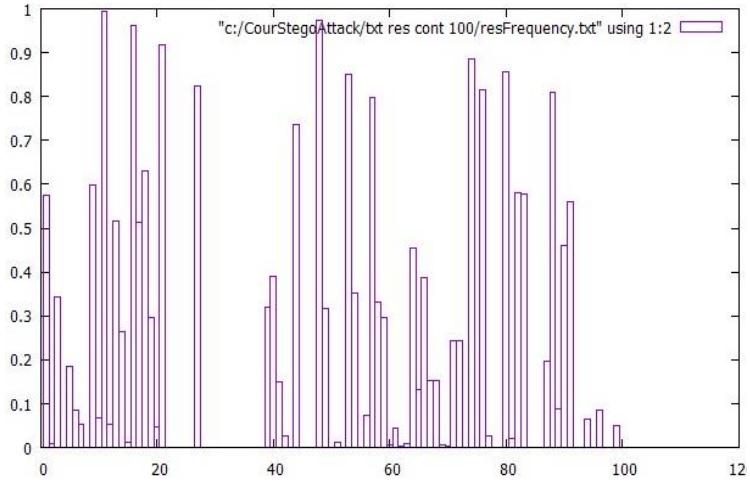


Рис. 4.11. Перевірка на випадковість послідовності LSB 100 аудіо файлів, заповнених на 100% максимальної можливості

З рис. 4.11 ви можете побачити, що при 100% перепаді послідовності LSB у файлах з відносною кількістю нульових байтів менше 0,0348 (значення для 22 файлів) є випадковими.

Тестова база даних із 100 аудіофайлів була перевірена на вкладеність за допомогою цього методу. На графіках нижче показано значення параметра Δ. Значення цього параметра порівнювали для порожніх і заповнених файлів (за допомогою розробленого програмного забезпечення Stegora WaveHide) на 100%. Результати показано на графіках нижче. По осі абсцис відкладені графіки досліджуваних номерів файлів, по осі ординат – отримане значення параметра Δ.

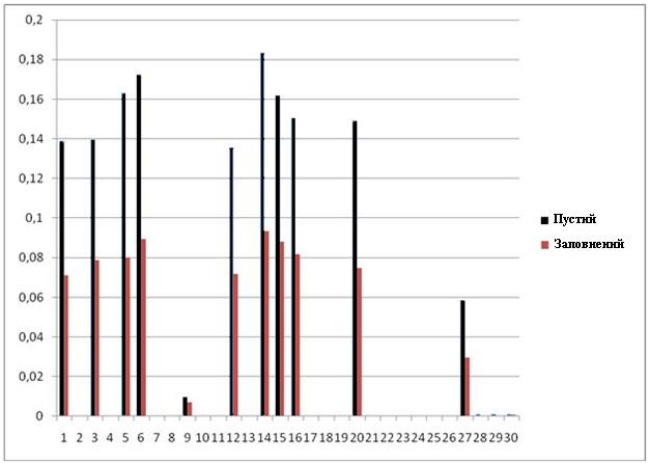


Рис. 4.12. Тестування файлів з 1 до 30

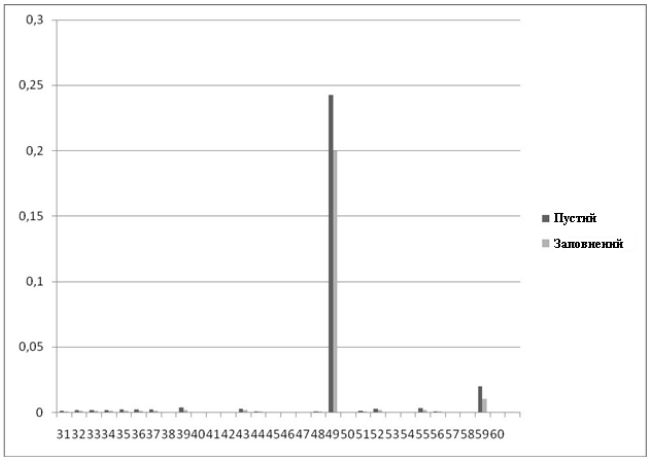


Рис. 4.13. Тестування файлів з 31 до 60

Як видно з результатів тестування розробленого програмного забезпечення (див. рис. 4.12 - 4.15), заповнена ємність стискається в середньому в два рази гірше. Таким чином, для визначення наявності вкладень в аудіофайлах можна використовувати метод стеганалізу, заснований на алгоритмах стиснення.

Однак, оскільки параметр Δ різний для різних типів файлів, для ефективного використання цього методу необхідно класифікувати файли та визначити власне граничне значення для кожного класу Δ.

4.6. Визначення вкладень в аудіо файл

Практичне застосування методики виявлення вкладених аудіофайлів складається з наступних кроків:

1. Береться файл, що перевіряється
2. Визначається відносне число нульових байт
3. Якщо це число менше, ніж 0,0038 (підібрано емпірично), то файл перевіряється частотним тестом NIST.

Якщо тест показує, що порядок молодших бітів у файлі є випадковим (тобто результат перевірки частоти перевищує 0,01), то це з високою достовірністю вказує на те, що файл має вкладення стего.

Програма реалізує цю техніку таким чином.

1. Натиснувши кнопку Аналіз, спочатку виберіть файл для аналізу, а потім файл, у якому буде збережено послідовність LSB вибраного файлу. Стандартний текстовий редактор Memo1 відображатиме відносне число нуль b. Відкриється набір статистичних тестів NIST. Виберіть файл із послідовністю молодших бітів аналізованого файлу, формат зчитування: текст, кількість послідовностей: 1, встановіть довжину послідовності відповідно до розміру файлу та натисніть кнопку «Тестувати».
2. Натискання кнопки Показати результати відкриває текстовий документ з результатами тестування та їх назвами.

Результати тестування бази даних із 108 різних аудіофайлів у форматі хвилі показали, що метод на основі алгоритмів стиснення [8] потребує подальшого вдосконалення класифікації аудіофайлів. З великою часткою

впевненості можна говорити про можливість використання цього методу для визначення вбудовування в аудіофайли, що належать до одного класу.

Результати тестування цієї бази даних, але методом частотного аналізу, показали наступне. Метод працює дуже ефективно для файлів з відносною кількістю нульових байтів менше 0,038. Файли цього класу характеризуються великим інформаційним навантаженням. До цього класу відносяться шуми, записи музичних інструментів з великим шумом (великий шум пов'язаний з використанням неякісних АЦП).

Тому, щоб ефективно ідентифікувати інвестиції в стеки, необхідно вдосконалити обидва методи та, можливо, їх комплексне використання.

4.7. Висновок

Було виявлено, що запропонована техніка працює дуже ефективно (вказано до 10% вкладеності), але лише на аудіофайлах з невеликою відносною кількістю нульових байтів.

ВИСНОВКИ

Дослідження в галузі стеганографії, тобто приховування інформації в різних контейнерах з метою приховування факту передачі, а зокрема цифрової стеганографії, є дуже перспективним напрямком захисту інформації, оскільки в сучасному світі, у світі В інформаційних технологіях завдання передачі секретної інформації прирівнюється до таємного спілкування, тобто приховування факту передачі повідомлення. Тому необхідно продовжувати дослідження в цій галузі з метою пошуку нових ефективних методів або вдосконалення існуючих.

У статті розглядаються способи вбудовування інформації в звукові файли. Після аналізу відомих методів стало зрозуміло, що метод ехо-сигналу є найбільш перспективним, але він потребує вдосконалення з точки зору пропускної здатності та ймовірності правильного вилучення вбудованих бітів інформації.

Для оцінки ефективності враховувалися такі параметри, як складність реалізації, необхідна обчислювальна потужність, особливі вимоги до контейнерів - звукових файлів, тип ключової інформації та складність її визначення зловмисником, вплив спроб знищити приховану інформацію про безпеку контейнера. Ці критерії оцінки дозволяють повною мірою оцінити ефективність того чи іншого методу.

Вибір способу вбудовування інформації в аудіофайли за допомогою ехо-перетворення обумовлений тим, що цей спосіб підходить для захисту аудіофайлів цифровими водяними знаками. Стійкість до амплітудно-частотних атак дозволяє обійти інші методи, нестійкі до цих атак.

Проведені дослідження з вивчення впливу часу затримки відлуння-сигналів та їх амплітуди на ефективність стеганографічного захисту інформації не претендують на ідеальність і правду, оскільки оцінюються однією особою. Тим не менш, застосування отриманих рекомендацій, безумовно, допоможе досягти певної високої ефективності методу вбудовування інформації в звукові файли за допомогою перетворення ехо-сигналу.

Реалізуючи метод стеганографічного захисту інформації шляхом перетворення ехо-сигналів, стало зрозуміло, що основна складність полягає в реалізації максимально ефективного алгоритму вилучення вбудованих бітів. Деякі дослідження в цьому напрямку показали, що досягти найбільшої ефективності можна лише при індивідуальному підході до кожного контейнера, змінюючи при цьому час затримки накладеного ехо-сигналу.

Також виявилося, що розробляючи цей продукт, можна створити новий сегмент на ринку програмного забезпечення, т.к. цей продукт буде унікальним і затребуваним у зв’язку з тим, що в умовах комп’ютеризації суспільства актуальним стало питання підтвердження авторства на об’єкти інтелектуальної власності, зокрема звукові файли.

Програма розгорнута та готова до використання. Можливість покращувати його та випускати додаткові версії є хорошою основою для комерційного використання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ДСТУ 3008-95. Документація. Звіти у сфері науки та техніки. Структура та правила оформлення. Київ, - 2016.
2. Єлізаренко А. О. Елементи радіоприймальних пристроїв. Конспект лекцій. Харків, - 2022.
3. D. Kahn, The Codebreakers: The Story of Secret Writing, Scribner, 2020. - 473 с.
4. Конахович, Г.Ф. Комп'ютерна стеганографія. Теорія та практика. / Г.Ф. Конахович, А.Ю. Пузиренко // Київ, МК-Прес, 2016 – 288с.
5. Максименко, С.Д. Загальна психологія: Навчальний посібник. / С.Д. Максименко, В.О. Соловєєнко - К.: МАУП, 2020. - 256с.
6. Massey J. L. An Introduction to Modern Cryptology. – Zürich: ETH Zurich, 2018. - №5. - С. 24-42.
7. Zhong, X., Huang, P.-C., Mastorakis, S., & Shih, F. Y. An automated and robust image watermarking scheme based on deep neural networks, 2020.
8. Ferdowsi, A., & Saad, W. Deep learning-based dynamic watermarking for secure signal authentication in the Internet of Things, 2017.
9. Arnold, M. MP3 robust audio watermarking/ M. Arnold, S. Kanka // International Watermarking Workshop. - 2019. – 1275 с.
10. Attacks on digital watermarks: classification, estimation-based attacks, and benchmarks / S. Voloshynovkiy, S. Pereira, T. Pun and others // IEEE Communications Magazine. 2010. Vol. 39. № 8. P.118-126.
11. Bas, P.A geometrical and frequential water-marking scheme using similarities / P. Bas, J.-M. Chassery, F. Davoine // In SPIE Conference on Security and Watermarking of Multimedia Contents. 2019. №3657. P. 264-272.
12. Bassia, P. Robust audio watermarking in the time domain / P. Bassia, I. Pitas, // Department of Informatics, University of Tressaloniki. – 956с.

13. Bender, W. Techniques for data hiding / W. Bender, B. Gruhl, N. Morimoto // IBM systems journal. - 2016. - Vol, 35. № 3. – 3c.
14. Boney, L. Digital watermarks for audio signals / L. Boney, A.H. Tewfic, A.K. Hamdy // Department of Electrical engineering, University of Minnesota.
15. Chae J. A robust embedded data from wave-let coefficients/J.Chae,D. Mukherjee, B. Manjunath // Proceedings of SPIE, Electronic Imaging, Storage and Retrieval for Image and Video Database. 2018. Vol. 3312. P. 308-317.
16. Chen, B. An Information-Theoretic Approach to the Design of Robust Digital Watermarking Systems / B. Chen, G.W. Wornell // Proceeding Int. Conf. on Acoustics, Speech and Signal Processing. 1999.
17. Chu, C.-J. Luminance channel modulated watermarking of digital images / C.-J. Chu, A.W. Wiltz // Proceedings of the SPIE Wavelet Applications Conference. 2019. P. 437-445.
18. Cox, I.J. Secure Spread Spectrum Watermarking for Multimedia / I.J. Cox, J. Killian, F.T. Leighton // IEEE Trans. Image Proc. 1997. Vol.6. № 12. P. 1673-1687.
19. Langelaar, G. Copy Protection for Multimedia Data based on Labeling Techniques / G. Langelaar, J. van der Lubbe, J. Biemond // 17th Symposium on Information Theory in the Benelux. 2016. – 389c.
20. Swanson, M.D. Multimedia Data-Embedding and Watermarking Strategies / M.D. Swanson, M. Kobayahi, A.H. Tewfik // Proceeding of IEEE. 2018. Vol. 86. №. 6. P. 1064-1087.
21. Wolgang, R.B. Perceptual Watermarking for Digital Images and Video / R.B. Wolgang, C.I. Podilchuk, E.J. Delp // Proceeding IEEE, Special Issue on Identification and Protection of Multimedia Information. 2019. Vol. 87. №. 7. P. 1088-1126.

22. Wong, P.W. A Public Key Watermark for Image Verification and Authentication / P.W. Wong // Proc. Int. Conf. Im. Proc. 2018. Vol. I. P. 455-459.

ДОДАТКИ

ДОДАТОК А. Вихідний код розробленого програмного забезпечення

А.1 Частина файлу SoundWatermarkDlg.h

```
// SoundWatermarkDlg.h : header file
//
#pragma once
#include <string>
// CSoundWatermarkDlg dialog
class CSoundWatermarkDlg : public CDialog
{
// Construction
public:
CSoundWatermarkDlg(CWnd* pParent = NULL);// standard constructor
// Dialog Data
enum { IDD = IDD_SOUNDWATERMARK_DIALOG };
protected:
virtual void DoDataExchange(CDataExchange*  pDX);// DDX/DDV support
// Implementation
protected:
HICON m_hIcon;
CEdit wavFilePathEdit;
CEdit watermarkFilePathEdit;
// Generated message map functions
virtual BOOL OnInitDialog();
afx_msg void OnPaint();
afx_msg HCURSOR OnQueryDragIcon();
DECLARE_MESSAGE_MAP()
public:
afx_msg void OnAbout();
afx_msg void OnSettings();
afx_msg void OnDropFiles(HDROP hDropInfo);
afx_msg void OnClickedBrowseWav();
afx_msg void OnClickedBrowseWatermark();
afx_msg void OnClickedAddWatermark();
private:
afx_msg bool textFieldsIsEmpty();
std::string getPathToWavFile();
std::string getPathToWatermarkFile();};
```

А.2 Частина файлу SoundWatermarkDlg.cpp

```
#include "SoundWatermarkDlg.h"
#include "SCoder/containers/wavcontainer.h"
#include "SCoder/coders/echocoder.h"
#include "SCoder/keys/echokey.h"
#define WAV_FILE_EXTENSION ".wav"
void CSoundWatermarkDlg::OnPaint()
{
if (IsIconic())
{
CPaintDC dc(this); // device context for painting
SendMessage(WM_ICONERASEBKGND, reinterpret_cast<WPARAM>(dc.GetSafeHdc()), 0);
// Center icon in client rectangle
int cxIcon = GetSystemMetrics(SM_CXICON);
int cyIcon = GetSystemMetrics(SM_CYICON);
CRect rect;
GetClientRect(&rect);
int x = (rect.Width() - cxIcon + 1) / 2;
int y = (rect.Height() - cyIcon + 1) / 2;
// Draw the icon
dc.DrawIcon(x, y, m_hIcon);
}
else
{
CDialog::OnPaint();
}
}
//////////////////////////////////////////////////////////////////
void CSoundWatermarkDlg::OnDropFiles(HDROP hDropInfo)
{
// get filename
char *fileName = new char[MAX_FNAME_LEN];
int fileNameLength = 0;
if(!(fileNameLength = DragQueryFile(hDropInfo, 0, fileName, MAX_FNAME_LEN)))
{
MessageBox("Could not retrieve file name. Try another file");
return;
```

}

```
}
// get mouse position
POINT point;
GetCursorPos(&point);
ScreenToClient(&point);
// set text to edit control relative to mouse position
 CWnd *pointedWindow = ChildWindowFromPoint(point);
int windowID = pointedWindow->GetDlgCtrlID();
if (windowID == IDC_WAV_FILE_PATH_EDIT)
{
// check file extension
if (strcmp(strlwr(fileName+fileNameLength-4),  WAV_FILE_EXTENSION))
{
MessageBox("File extension should be '.wav'");
return;
}
pointedWindow->SetWindowText(fileName);
}
else if (windowID == IDC_WATERMARK_FILE_PATH_EDIT)
{
pointedWindow->SetWindowText(fileName);
}
CDialog::OnDropFiles(hDropInfo);
}
//////////////////////////////////////////////////////////////////////
void CSoundWatermarkDlg::OnClickedBrowseWav()
{
CFileDialog fd(true, 0, 0, 4|2, "Wave (*.wav)|*.wav|");
if (fd.DoModal()==IDOK)
wavFilePathEdit.SetWindowText(fd.m_ofn.lpstrFile);
}
void CSoundWatermarkDlg::OnClickedBrowseWatermark()
{
CFileDialog fd(true);
if (fd.DoModal()==IDOK)
watermarkFilePathEdit.SetWindowText(fd.m_ofn.lpstrFile);
}
```

```
//////////////////////////////////////////////////////////////////
void CSoundWatermarkDlg::OnClickedAddWatermark()
{
// check text fields
if (textFieldsIsEmpty())
{
MessageBox("Specify files first");
return;
}
// try to open wav file
std::string pathToWavFile = getPathToWavFile();
WAVContainer wavContainer;
if (!wavContainer.Open(pathToWavFile))
{
MessageBox("Wrong WAV file (should be PCM 16 or 32 bits)");
return;
}
EchoCoder coder;
// get watermark from file
std::string pathToWatermark = getPathToWatermarkFile();
std::string watermarkFile;
std::string line;
std::ifstream watermarkIfStream(pathToWatermark.c_str());
while(std::getline(watermarkIfStream,line))
watermarkFile += line;
// check if message will fit container
int containerCapacity = (BITS_PER_SECOND * ((float)wavContainer.Size()
(float)wavContainer.SampleRate())) / 8;
if (watermarkFile.size() > containerCapacity)
{
MessageBox("Container is too small to inject choosen watermark");
return;
}
// generate key from watermark file
EchoKey key("key", Key::STRING);
// inject watermark
coder.SetMessage(&wavContainer, watermarkFile.c_str(), &key);
```

```
wavContainer.Save(pathToWavFile);
}
////////////////////////////////////////////////////////////////
afx_msg bool CSoundWatermarkDlg::textFieldsIsEmpty()
{
std::string pathToWavFile = getPathToWavFile();
char pathToWatermarkFile[MAX_FNAME_LEN];
watermarkFilePathEdit.GetWindowText(pathToWatermarkFile, MAX_FNAME_LEN);
if (pathToWavFile.length() == 0 || strlen(pathToWatermarkFile) == 0)
{
return true;
}
return false;
}
////////////////////////////////////////////////////////////////
std::string CSoundWatermarkDlg::getPathToWavFile()
{
char pathToWavFile[MAX_FNAME_LEN];
wavFilePathEdit.GetWindowText(pathToWavFile, MAX_FNAME_LEN);
return std::string(pathToWavFile);
}
////////////////////////////////////////////////////////////////
std::string CSoundWatermarkDlg::getPathToWatermarkFile()
{
char pathToWatermarkFile[MAX_FNAME_LEN];
watermarkFilePathEdit.GetWindowText(pathToWatermarkFile, MAX_FNAME_LEN);
return std::string(pathToWatermarkFile);
}
```

A.3 Часть файла echocoder.cpp

```
////////////////////////////////////////////////////////////////
void EchoCoder::SetMessage( WAVContainer* _container, const std::string& _message,
   const Key* _key )
{
 // Mustbe a echo key
 if ( _key->IsEchoKey() )
 {
m_FrameSize = _container->SampleRate() / BITS_PER_SECOND;
```

```
CalculateKey(_container);
 // Hide message using echo algorithm
 LSBSoundCoder::SetMessage(_container, _message);
 }
}
//////////////////////////////////////////////////////////////////
std::string EchoCoder::GetMessage( const WAVContainer* _container, const Key* _key )
{
 // Mustbe a echo key
 if ( _key->IsEchoKey() )
 {
m_FrameSize = _container->SampleRate()  / BITS_PER_SECOND;
CalculateKey(_container);
 // Hide message using echo algorithm
 return LSBSoundCoder::GetMessage(_container);
 }
 // Not a echo key
 return "";
}
//////////////////////////////////////////////////////////////////
void EchoCoder::CalculateKey(const  WAVContainer* _container)
{
int frameRate = _container->SampleRate();
m_EchoFirstOffset = frameRate/750;
m_EchoSecondOffset = frameRate/1500;
}
//////////////////////////////////////////////////////////////////
bool EchoCoder::SetBit(  bool _bit )
{
int sizeOfShift = (_bit ? m_EchoFirstOffset : m_EchoSecondOffset);
short *samplesForShift = new short[sizeOfShift];
ZeroMemory(samplesForShift, sizeOfShift * sizeof(short));
for (int i=0; i<m_FrameSize; i++)
{
if (!JumpToNextSample())
{
// Bit has not been written
```