МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

на тему:

«Інформаційна аудиостеганографічна технологія»

Завідувач кафедрою **Довбиш А.С.**

Керівник роботи **Шелехов І.В.**

Студент гр. ІН-м.02 **Устик Д.С.**

СУМИ 2021

Сумський державний університет
(назва вузу)

Факультет *ЕлІТ* Кафедра *Комп'ютерних наук*
Спеціальність *122 «Комп'ютерні науки»*

Затверджую:
зав.кафедрою _____
_____
"_____" _____ 20___р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТОВІ

Устику Дмитру Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) *Інформаційна аудиостеганографічна технологія*

затверджую наказом по інституту від "____" _____ 20 ___ р. № _____

2. Термін здачі студентом закінченого проекту (роботи) _____

3. Вхідні данні до проекту (роботи) _____
_____
_____
_____
_____

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) *1) Аналіз проблеми та постановка задачі 2) Методи цифрової аудіостеганографії 3) Інформаційне та програмне забезпечення*

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

6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання _____

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів дипломного проекту (роботи)	Термін виконання проекту (роботи)	Примітка
1.	*Аналіз проблеми та постановка задачі*		
2.	*Методи цифрової аудіостеганографії*		
3.	*Інформаційне та програмне забезпечення*		
4.	*Оформлення пояснювальної записки до кваліфікаційної магістерської роботи*		

Студент – дипломник _____
(підпис)

Керівник проекту _____
(підпис)

РЕФЕРАТ

Записка: 50 стор., 7 рис., 11 табл., 1 додаток, 30 джерел.

Об’єкт дослідження — слабоформалізований процес стеганографії аудіосигналів.

Мета роботи — розробка вдосконаленого методу аудіостеганографії у форматі Bi-Last Significant Bit (Bi-LSB) MP3, що дозволить вирішити як шифрування секретного повідомлення, так і гарантувати його цілісність.

Методи дослідження — методи подання та обробки аудиоданих, методи кодування аудіосигналів, методи цифрової стеганографії.

Результати — сформовано вхідний математичний опис аудіостенографічної системи, розроблено математичну модель функціонування аудіостенографічної системи, розроблено методи оцінки якості функціонування аудіостенографічної системи, розроблено і програмно реалізовано алгоритм шифрування секретного повідомлення аудіостенографічною системою, розроблено і програмно реалізовано алгоритм забезпечення цілісності секретного повідомлення аудіостенографічною системою, виконано тестування аудіостенографічної системи.

АУДІОСТЕНОГРАФІЧНА СИСТЕМА, СЕКРЕТНЕ ПОВІДОМЛЕННЯ, АУДІОКОДЕК, КВАНТУВАННЯ АНАЛОГОВОГО СИГНАЛУ, РЯД ФУР’Є, MP3, НАДІЙНІСТЬ, ЄМНІСТЬ, СКРИТНІСТЬ

ЗМІСТ

ВСТУП

Інформація поширюється по всьому світу через Інтернет у цифровій формі [22]. Існують проблеми та проблеми щодо безпеки інформації, що передається від відправника до одержувача. Головне питання – захист цифрових даних від будь-яких форм вторгнення, проникнення та крадіжки. Головною проблемою є розробка рішення для захисту інформації та забезпечення її безпеки під час передачі. Три складові інформаційної безпеки: конфіденційність, цілісність і доступність [23]. Конфіденційність забезпечує збереження інформації від будь-якого несанкціонованого доступу. Це можна зробити за допомогою методів приховування інформації, а саме криптографії та стеганографії [24].

Криптографія передбачає акт шифрування та дешифрування цифрових даних. Основними недоліками таких методів є те, що, навіть якщо повідомлення було зашифроване, воно все ще існує. Стеганографія зосереджується на приховуванні будь-яких цифрових даних на нешкідливому цифровому носії, слово стеганографія походить від старогрецького слова, що означає закритий лист.

Криптографія передбачає акт шифрування та дешифрування цифрових даних. Основними недоліками таких методів є те, що, навіть якщо повідомлення було зашифровано, воно все ще існує. Стеганографія зосереджується на приховуванні будь-яких цифрових даних на нешкідливому цифровому носії, слово стеганографія походить від старогрецького слова, що означає закритий лист [10].

Стеганографія використовувалася для приховування таємних повідомлень у давні часи [25]. Його використовував Гістіей, тиран Мілету, який у 499 році до нашої ери татуював скальпи своїх рабів із прихованим повідомленням із наказом своїм людям напасти на перса. Повідомлення стало прихованим, коли волосся рабів відросло. На думку дослідників, стеганографія може бути описана як дослідження засобів приховування

вторинної інформації в первинній інформації без впливу на розмір інформації або причину будь-яких форм спотворення, які можуть бути сприйняті.

Первинна інформація, відома як носій або хост, була вбудована у вторинну інформацію, яка зазвичай прихована і може бути у формі файлу або повідомлення. Носій із вбудованою інформацією називається стегосигналом, файлом, бітовим потоком або послідовністю [11].

Стеганографія є одним із двох методів, що використовуються для прихованого спілкування. Однак водяні знаки є другою технікою, яка може вставляти водяний знак в обкладинку хоста, щоб зберегти авторські права для хостів. Стеганографія зазвичай встановлює безпеку даних «точка-точка» [26]. Потужність стеганографічної техніки, що забезпечує збереження даних на носії від атак або змін, слабка під час передачі, зберігання або перетворення формату [10].

Процес вбудовування інформації в основний носій у техніці стеганографії та водяних знаків зазвичай здійснюється прозоро [27]. Таким чином, з точки зору видимих і невидимих водяних знаків, процес має забезпечити надійність, щоб будь-які навмисні атаки не скомпрометували, не видаляли або не викликали знищення інформації в позначених носіях, водночас зберігаючи якість сигналу [28].

1 ОБРОБКА ТА КОДУВАННЯ АУДІОСИГНАЛІВ

1.1 Методи подання аудіосигналів

Аналоговим сигналом s є кінцева функція $s(t)$ з дійсним значенням неперервної змінної t (так званим часом), визначена для будь-якого значення часу на інтервалі $-\infty < t < +\infty$. Цифровий сигнал s є обмеженою дискретною послідовністю s_n з єдиним індексом n (так званим дискретним часом), визначеним для будь-якого значення часу $n = -\infty ... + \infty$.

Тобто сигнал – це функція від часу, де моменти часу беруться з неперервного простору у випадку аналогових сигналів і з дискретного простору у випадку цифрових сигналів. При цьому значення, які може приймати аналоговий сигнал, є нескінченними (незважаючи на розподіл), оскільки вони формуються на інтервалах дійсних значень. У випадку цифрових сигналів значення, які аналізуються в поточний момент, є дискретними, оскільки діапазони, на яких визначаються сигнали є кінцевими. Такий процес квантифікації при перетворенні аналогового сигналу в цифровий спрошує подальшу обробку аудіосигналів.

Зазвичай сигнал представлений однією або кількома синусоїдальними функціями, яка отримує в якості параметрів кутову частоту і момент, в який сигнал має бути представлений, і множиться на реальну константу. Це рівняння має вигляд:

$$s(t) = A \cdot sin(\omega t)$$

Зазвичай сигнал розкладається на кілька синусоїдних функцій, кожна з яких має різну частоту. Ось чому сигнал залежить тільки від параметра часу.

Отже, ці сигнали є не більше ніж хвилями, які по черзі можуть складатися з кількох хвиль різної частоти. Частота хвилі вказує на кількість циклів, які вона завершує за секунду. Частота є оберненою до періоду, тобто

час, необхідний хвилі для завершення циклу. Залежно від частоти хвилі людина здатна сприймати їх за допомогою слухової або зорової системи, або

не сприймати їх взагалі. Частоти чутного для людини коливаються приблизно від 20 Гц до 22 КГц.

Оцифрування аналогових сигналів

Нехай дано реальний сигнал у безперервній області, який необхідно перетворити в сигнал дискретної області також з дискретними значеннями. Процес, за допомогою якого обираються, які моменти безперервної часової області будуть подані в новій дискретній часовій області, називається вибіркою. При цьому використовується фундаментальна теорема Найквіста і Шеннона, яка стверджує наступне:

Враховуючи функцію $s(t)$ з частотами не більшими за B Гц, її можна повністю відновити, якщо частота дискретизації більша або дорівнює $2B$ Гц.

Ця теорема говорить, що, використовуючи відповідну частоту дискретизації, не буде втрачено жодної інформації з аналогового сигналу, принаймні, якщо використати безперервний діапазон реальних значень. Проблема полягає в тому, що діапазон значень, який буде використовуватися, є дискретним, а не безперервним, тому необхідно створити таке відображення, щоб призначити кожному безперервному значенню сигналу, який тепер дискретизовано в часі, єдине дискретне значення. Цей процес відомий як квантування. Він може бути реалізований декількома можливими альтернативними способами. Наприклад, метод ІКМ (імпульсно-кодова модуляція) складається з попередньо визначеної довжини кроку квантування, тобто відстані між кожними двома сусідніми парами значень, які сигнал може прийняти в поточний момент. Коли довжина цього кроку квантування фіксована, вважається, що використовується лінійна або рівномірна стратегія. Якщо використовується крок квантування змінної довжини, то квантування вважається нелінійним або нерівномірним. Крім того, також можна подавати в кожен момент t замість значення сигналу, тобто $s(t)$, різницю між значенням у момент сигналу в момент t і безпосередньо попередній момент $t–1$. Такий метод квантування використовує підхід ДІКМ (диференціальної імпульсно-кодової модуляції). Помилка, що виникає внаслідок квантування цього

сигналу, відома як помилка квантування. На рис. 1.1 та 1.2 подано приклади рівномірного та неоднорідного квантування та ІКМ та ДІКМ відповідно.

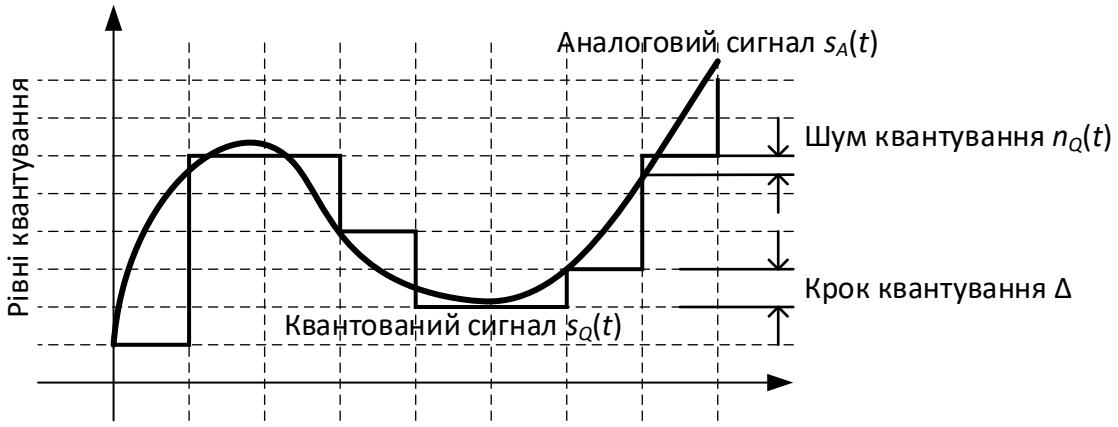

Рисунок 1.1 – Приклад рівномірного квантування аналогового сигналу

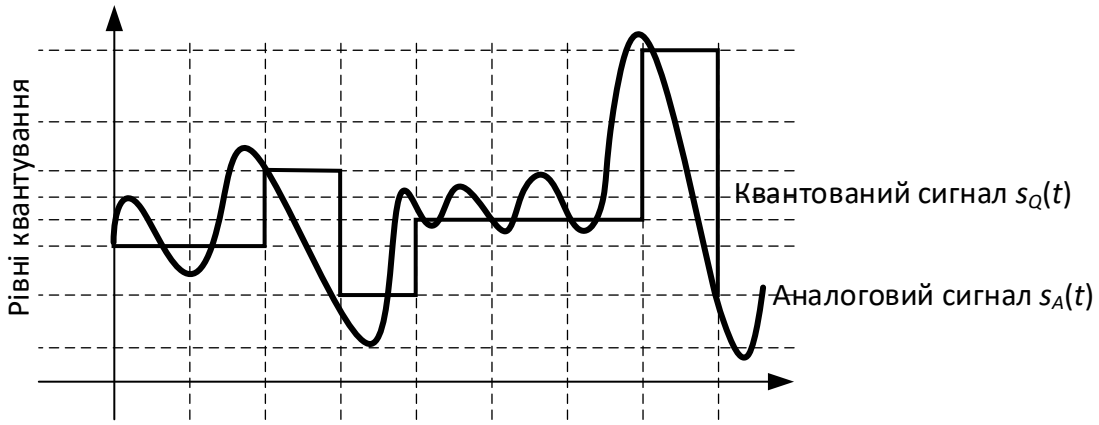

Рисунок 1.2 – Приклад нерівномірного квантування аналогового сигналу

Область часу і область частоти: ряд Фур'є

У першій половині 19 століття, Жозеф Фур'є, який вивчав проблему поширення тепла в твердих тілах, прийшов до висновку, що деякі рішення його досліджень мають вигляд $f(t)$ $g(x)$, де $g(x)$ були синусоїдальними функціями. Вивчаючи ці функції, Фур'є заявив, що найбільш загальну функцію $g(x)$ можна описати як лінійну комбінацію синусоїдальних функцій виду:

$$g(x) = \sum_{k=0}^{\infty} \left(a_k \sin(kx) + b_k \cos(kx)\right) \qquad (1.1)$$

Ця функція відома як ряд Фур'є. При цьому сигнал (аналоговий або цифровий) розглядався як функція від часу, але з рівнянням (1.1) сигнал можна розглядати з іншої точки зору - сигнал (який, зрештою, є не більше ніж функція) можна визначити як підсумок лінійної адитивної комбінації кожної синусоїдальної функції різних частот. Іншими словами: сигнал $s(x)$ веде себе як синус і косинус частоти ω_0 і амплітуди a_0 і b_0 відповідно, плюс синус і косинус частоти ω_1 та амплітуди a_1 та b_1 відповідно. У випадку акустичних сигналів, до яких чутлива людина, тобто сигналів з частотами в діапазоні від 20 Гц до 22 кГц, основною задачею при квантуванні буде пошук спектральних складових в цьому інтервалі. В результаті сигнал буде подано у вигляді з сигналів окремих частот, які його складають [1].

1.2 Методи кодування аудіосигналів

Звуковий сигнал є поданням звуку. При цьому для цифрових сигналів таке подання використовує послідовность двійкових чисел. Аудіосигнали мають частоти в діапазоні звукових частот приблизно від 20 до 20 000 Гц, що відповідає нижній і верхній межі людського слуху.

Для подання аудиосигналу використовуються кодеки. Аудіокодек на програмному рівні є спеціалізованою комп'ютерною програмою, кодеком, який стискає (виконує компресію) або розтискає (виконує декомпресію) цифрових звукових даних відповідно до файлового звукового формату.

Різні аудіоформати можуть бути розділені на 3 категорії:

1 Нестиснені аудіоформати
2 Аудіоформати із втратами при стисненні
3 Аудіоформати зі стисненням без втрат

Нестиснені аудіоформати є найбільш точним і справжнім відображенням оригінальних звукових хвиль. Звук із нестиснених файлів дуже схожий на те, як він був записаний і призначений для прослуховування. Оскільки файли не зазнають жодного стиснення, не відбувається втрати

інформації, що виливається в високоякісний звук, але великий розмір вихідного файлу. Далі наведені приклади нестиснених аудіоформатів.

Pulse-Code Modulation (PCM)

Це один із широко використовуваних форматів, коли мова йде про нестиснені аудіофайли. Здебільшого тому, що процес перетворення аудіосигналу (присутнього у формі сигналу) в цифровий потік здійснюється без особливих змін. І взагалі без стиснення. В результаті формат пропонує точне відображення оригінального аналогового звуку. Саме з цієї причини PCM став найбільш часто використовуваним аудіоформатом на компакт-дисках і DVD.

Waveform Audio File Format (WAV)

Формат WAV був розроблений Microsoft і IBM ще в 1991 році. WAV, у двох словах, схожий на контейнер для різних аудіоформатів, через що іноді може містити стиснені аудіоформати. Однак це рідко буває, і в більшості випадків файли не стиснуті і представлені в аудіоформаті PCM.

Audio Interchange File Format (AIFF)

Подібно до WAV, який розроблено Microsoft і IBM і в основному призначений для використання в Windows, формат AIFF є власним форматом, розробленим Apple для своєї лінійки Mac. Подібно до WAV для Windows, AIFF — це контейнер для Apple, який може зберігати різні види аудіоформатів.

Стиснення із втратами - це стиснення, яке призводить до втрати даних під час процесу стиснення. А оскільки для зменшення розміру аудіофайлу потрібне стиснення, компроміс щодо якості є прийнятним у більшості випадків використання. Залежно від того, як виконується стиснення, воно може виявитися або хорошим стисненням — з невеликою втратою даних — або поганим стисненням, яке повністю погіршує якість аудіо та вносить ефекти, які змінюють вихідний звук. Оскільки менший розмір файлу також супроводжує втрату даних і точності звуку, стиснення з втратами не є кращим

вибором у професійних налаштуваннях, які вимагають високоякісного звуку. Приклади форматів:

MPEG-1 Audio Layer 3 (MP3)

MP3 є відносно старішим форматом, ніж більшість інших аудіоформатів, які зазвичай використовуються сьогодні. Коли він був випущений, MP3 замінив файли MIDI і WAV, які широко використовувалися в той час. Пізніше був прийнятий як стандартний формат, що використовується в аудіо компакт-дисках і різних музичних порталах.

Однією з головних причин широкого поширення MP3 є менший розмір файлу, який вдається досягти формату за допомогою алгоритму стиснення даних із втратами, який позбавляє від деяких даних (звук, який зазвичай виходить за межі слухових можливостей людей). А колись, оскільки більшість пристроїв не мали такого великого обсягу пам’яті, як сьогодні, розмір аудіофайлу був головним стримуючим фактором у виборі аудіоформату. Сьогодні більшість пристроїв на ринку мають підтримку аудіофайлів у форматі MP3, що говорить про його популярність та широке поширення.

Advanced Audio Coding (AAC)

Розроблений як наступник формату MP3, AAC запропонував кращу якість звуку, ніж MP3, і незабаром став одним із популярних аудіоформатів. Однак, незважаючи на свою популярність, він все ще не міг перемогти MP3, який на той час повністю домінував в аудіоіндустрії. Хоча, якщо порівнювати з MP3, AAC вирізняється своїм вдосконаленим алгоритмом стиснення, який згодом допоміг йому виділитися серед інших алгоритмів стиснення кращою та покращеною якістю звуку. Незважаючи на популярність MP3, формат AAC все ще активно використовується на різних платформах, таких як YouTube, iDevices, PlayStation, Nintendo і навіть Android.

OGG (Vorbis)

На відміну від більшості інших аудіоформатів, OGG не є абревіатурою. У нього є власний набір переваг перед іншими аудіоформатами. Загалом, OGG — це більше мультимедійний контейнер, який може містити різні типи

форматів стиснення, серед яких найпомітнішим є Vorbis. Ось чому ці аудіофайли іноді називають OGG Vorbis. Протягом багатьох років Vorbis повільно набував популярності завдяки двом відмінним характеристикам — однією з них є відкритий вихідний код, а також кращою якістю звуку, ніж більшість форматів стиснення звуку з втратами.

Windows Media Audio (WMA)

WMA було введено для вирішення деяких проблем з алгоритмом стиснення, що використовується в MP3, який Microsoft вдалося добре виправити і представити кращий аудіоформат, який вирішував багато проблем з якістю стиснення. Однак, оскільки це був власний формат, він не знайшов великої ваги в аудіоіндустрії, і, незважаючи на те, що було виправлено деякі з помітних проблем, які мали MP3, він все ще не знайшов свого місця на різних пристроях і платформах, які зараз використовуються.

На відміну від стиснення з втратами, яке втрачає деякі дані під час процесу стиснення і, в свою чергу, змінює оригінальну якість звуку, стиснення без втрат, з іншого боку, зменшує розмір файлу аудіо, зберігаючи якість аудіо без втрати даних. А оскільки втрата даних під час стиснення не відбувається, якість звуку не погіршується, що робить ці формати кращим вибором для використання в професійному середовищі. Приклади форматів:

Free Lossless Audio Codec (FLAC)

FLAC – це відкритий формат, який швидко став одним із найпопулярніших форматів для високоякісного аудіо. Певним чином він дуже схожий на MP3, але без втрат, тому, на відміну від MP3, він не зазнає втрати даних під час стиснення і пропонує фактичне представлення оригінального аудіо, схоже на якість на рівні CD у меншому розмірі файлу. Протягом багатьох років різні платформи та пристрої почали підтримувати формат FLAC, завдяки чому став альтернативою дуже популярному формату MP3.

Apple Lossless Audio Codec (ALAC)

Подібно до FLAC, формат ALAC також пропонує оригінальну якість звуку, але в порівняно меншому розмірі файлу. Незважаючи на схожість з

FLAC, ALAC не має такої ж популярності та діапазону підтримуваних платформ і пристроїв, оскільки він розроблений для роботи лише з продуктами Apple. І навпаки, користувачі пристроїв Apple не можуть використовувати формат FLAC.

Windows Media Audio (WMA) – Lossless

WMA – lossless дуже схожий на звичайний WMA – Lossy, який також є власним аудіоформатом, що належить Microsoft. Ці два формати відрізняються за якістю звуку, яку вони пропонують, при цьому WMA без втрат пропонує вищу якість завдяки ефективному стисненню.

1.3 Структура MP3 файлу

Одним із кодеків, що використовуються для стиснення аудіо в цифрову форму, є MP3; він намагається зайняти мінімально можливий простір, і в той же час підтримує якомога кращу якість аудіо. Цей метод в цій області є одним з найкращих досягнень

«Група експертів з рухомих зображень (MPEG)» створила MP3; вона була створена в січні 1988 року з метою створення стандартів, застосовних на міжнародному рівні, для кодованого відображення аудіо, рухомих зображень та їх комбінації. Функціонування групи здійснюється під спільним керівництвом «Електротехнічної комісії (IEC) та «Міжнародної організації зі стандартизації (ISO)». У світі відео спочатку було розроблено стандарт, який дозволяє відтворювати аудіо-відеоматеріал із пристрою, здатний передавати 1,5 мільйона біт/секунду; простими словами, звичайний CD-ROM.

1992 рік знаменує собою період, коли MP3 був випущений як частина моделі MPEG. Термін «MP3» використовувався для позначення рівня 3 режиму стиснення MPEG-1.

Layer-3 являє собою найскладніший мод, який був оптимізований для забезпечення найвищої якості при низькій швидкості передачі даних (близько 128 кбіт/с).

Маючи властиві можливості, які зменшують розмір файлу без компромісу в якості чутності, він став дуже успішним. MP3 спричинив революцію в Інтернеті, став першим аудіоформатом, завдяки якому процес обміну аудіофайлами в Інтернеті став можливим. Раніше для завантаження високоякісних файлів потрібно було багато годин. З MP3 час завантаження було значно скорочено без будь-яких помітних змін у якості «звуку» аудіо. Алгоритм MPEG досягає стиснення без втрат, хоча стиснення MP3 вважається стисненням із втратами, оскільки після стиснення деякі дані не можуть бути відновлені. Після певного тесту було зроблено висновок, що досвідчені слухачі не змогли розрізнити оригінальні кліпи навіть при використанні коефіцієнта стиснення шість до одного.

Якщо проаналізувати, чому MP3 і жодна інша технологія стиснення стали основним інструментом доставки аудіо в Інтернет, випливають наступні факти:

- Відкритий стандарт:

Це рівень MPEG-1 3. Специфікації доступні для всіх, хто зацікавлений у впровадженні стандартів, оскільки жодна компанія не володіє стандартом.

- Наявність кодерів і декодерів:

Це викликано попитом для професійного використання; багато MP3-декодерів і кодерів легко доступні для цілеспрямованого використання. Це прискорює та спрощує впровадження технології MP3.

- Допоміжні технології:

Комп’ютерні звуки широко поширені, і комп’ютери стали достатньо швидкими, щоб виконувати такі функції, як програмне кодування аудіо та декодування. Швидкий доступ до Інтернету для підприємств, університетів і поширення CD-аудіо-записувачів і компакт-дисків зробили великий внесок у легкість розповсюдження музики у форматі MP3 через комп’ютери. Коротше кажучи, MPEG-1/2/3 рівень 3 – це правильна технологія, яка доступна в потрібний час.[4]

Композиція файлів MP3 являє собою короткі кадрі даних, які доповнені заголовками. Теги метаданих також можуть міститися в MP3. Теги бувають двох типів: ID3v1, який є старішим форматом, і подається в кінці файлу. Тег завжди має довжину 128 байт і має сім полів; вони вказують ім'я виконавця, альбом, назву пісні, жанр та інші характеристики... Через відсутність гнучкості та статичний розмір тип тегу повільно замінюється стандартом ID3v2, який є більш досконалим [3].

Тег ID3v2 є більш гнучкими і є попередньо завантаженим у файл. Існують майже гнучкі структури, схожі на структуру самих файлів MP3. Теги ID3v2 складаються з власних кадрів; кадри зберігають різні біти інформації. Він складається зі стандартних рядків символів, таких як ім’я виконавця, назва пісні або більш розширена інформація про те, як було виконано кодування файлу. Теги ID3v2 корисні для надання підказок декодеру. Наприклад, теги ID3v2 зберігають криві вирівнювання. Теги ID3v2 не мають встановлених обмежень, тому теоретично вони можуть зростати нескінченно. Стеганографіст повинен бути готовий мати справу з тегами інформації, присутніми після або перед потоком аудіоданих, оскільки немає чітких вимог. Однак логічно припустити, що теги ID3v1 в майбутньому будуть зустрічатися рідше.

Теги ID3v2 біль привабливі для вбудовування інформації через їх можливість розширення. Однак немає гарантії, що вони будуть присутні в кожному файлі MP3. Тому найкращим підходом є вбудовування даних у кадри даних. Перед обговоренням методології стенографії було б доцільно детальніше розглянути кадр даних.

Розмір кадру не є очевидним, тому необхідно мати можливість ідентифікувати початок і кінець кадру. Перед кожним кадром є заголовок кадру. Усі заголовки ідентичні за змістом і структурою. Таким чином, ідентифікація заголовка MP3 - це лише питання відповідності шаблону.

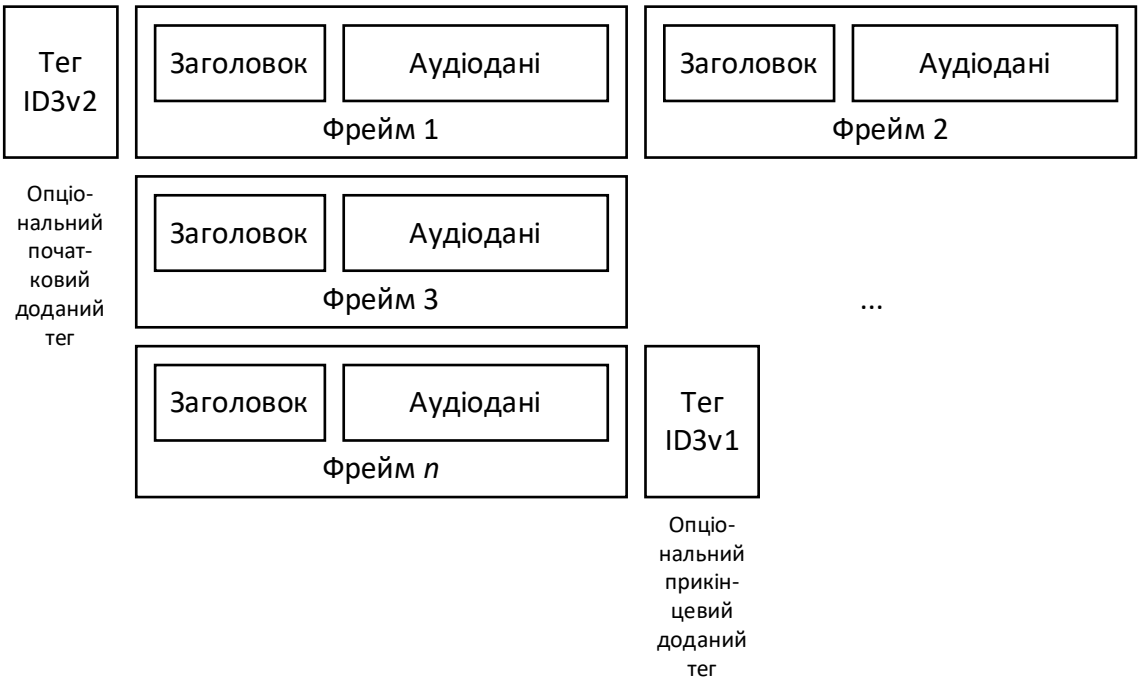

Рисунок 1.3 – Модель MP3-файлу

Блок синхронізації (Sync) — це 12-бітовий блок у кожному заголовку, і саме вони починаються, як показано на рис. 1.4. Блок синхронізації являє собою рядок, роль якого полягає в тому, щоб допомогти декодеру знайти заголовок. Отже, для ідентифікації кадру потрібно просто виявити 12 послідовних бітів, ініціалізованих як 1.

Рисунок 1.4 – Структура MP3-файлу

Однак шаблон не обов'язково прив’язаний до заголовка. Дійсно, шаблон можна легко ідентифікувати в будь-якому блоці даних, який довший. Є ще кілька перевірок, які виконуються для перевірки 4-байтового блоку даних заголовка:

- Поле Швидкість передачі даних не має значення 0000 або 1111.
- Поле Частота не дорівнює 11.
- Поле «Рівень» не має значення 00.

4-байтовий блок, який не порушує вищевказаних умов і починається з Sync, ймовірно, є заголовком [7].

Таблиця 1.1 – Короткий опис полів MP3-файлу

Поле	Застосування	Довжина
A	Кадр «Sync»	12
B	Версія Аудио	2
C	Рівень MPEG (I, II, III)	2
D	Захист (контрольна сума)	1
E	Індекс бітрейту	4
F	Частота (44.1kHz)	2
G	Проміжний біт (включ. або виключ., компенсатор незаповнених кадрів)	1
H	Приватний біт (включ. або виключ., дозволяє використовувати конкретні тригери)	1
I	Режим каналу (подвійний канал, змішане стерео, одиночний канал, стерео)	2
J	Розширення режиму	2
K	Авторське право	1
L	Справжність (Оригінал/Копія)	1
M	Виділення	2

1.4 Постановка задачі

Проведений аналітичний огляд доводить, що аудіостеганографія є ефективним методом захисту даних і надсилання їх через Інтернет. Метою роботи є розробка вдосконаленого методу аудіостеганографії у форматі Bi-Last Significant Bit (Bi-LSB) MP3, що дозволить вирішити задачу як шифрування секретного повідомлення, так і гарантувати його цілісність. Для досягнення поставленої мети необхідно виконати такі завдання:

1) Сформувати вхідний математичний опис аудіостенографічної системи.

2) Розробити математичну модель функціонування аудіостенографічної системи.

3) Розробити методи оцінки якості функціонування аудіостенографічної системи.

4) Розробити і програмно реалізувати алгоритм шифрування секретного повідомлення аудіостенографічною системою.

5) Розробити і програмно реалізувати алгоритм гарантування цілісності секретного повідомлення аудіостенографічною системою.

6) Виконати тестування аудіостенографічної системи.

2 МЕТОДИ ЦИФРОВОЇ АУДІОСТЕГАНОГРАФІЇ

2.1 Основні принципи та визначення

Стеганографія — старовинне мистецтво, яке відродилося в останні роки; це мистецтво приховує припущення, що відбувається спілкування [8]. Існування двох каналів зв'язку має бути приховано від можливого зловмисника [9]. Стеганографія в основному бере частину інформації, а потім приховує інформацію в іншому комп'ютерному файлі (звукозаписи, зображення та тексти), що містить незначні або невикористані ділянки даних. Пізніше ці файли можна транспортувати або надсилати, не підозрюючи, що є насправді всередині них [10].

Саме стеганографія в сучасному суспільстві відіграє все більш важливу роль, оскільки вимоги щодо захисту цифрових авторських прав і прихованих комунікацій продовжують зростати [11]. Стеганографія займається методами забезпечення того, щоб секретне повідомлення було вбудовано (яке може бути серійним номером, прихованим повідомленням або знаком авторського права) у файл (наприклад, аудіозапис, відеофільм, або навіть комп'ютерний код). Третій стороні важко видалити або виявити вбудований матеріал, коли об'єкт в який було вбудовано повідомлення, має вбудований матеріал. Це називається стего-об'єктом. Наприклад, ми можемо вставити текст у зображення або позначку в текст, тощо.

У досконалій стего-системі зображення стего не можна відрізнити від оригінального файлу. Щоб уникнути випадкового повторного використання стего-об'єкту, одержувач і відправник повинні знищити всі такі об'єкти, які вони вже використовували для передачі інформації [12].

Основна модель системи стеганосистеми вказана на схемі (рис. 2.1) нижче [13].

Рисунок 2.1 – Модель стеганосистеми

Кожен метод приховування даних складається з:

- Алгоритму вбудовування;
- Алгоритму вилучення;

Використання алгоритму вбудовування полягає в тому, щоб приховати секретні повідомлення в носії, де ключ захищає процес вбудовування, щоб лише ті, хто передає секретне ключове слово, могли мати доступ до прихованого повідомлення.

Застосування алгоритму вилучення здійснюється до носія прихованого секретного повідомлення.

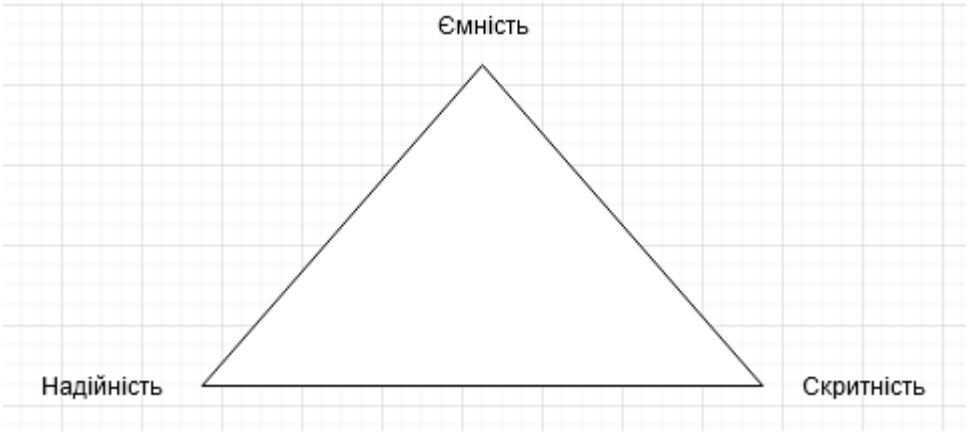

Рисунок 2.2 – Властивості стеганографії

Найважливішими властивостями схем приховування даних є ємність, надійність і скритність. На рис. 2.2 представлений трикутник властивостей стеганографії:

Надійність

Вбудовані дані повинні мати імунітет до модифікації, починаючи від навмисної спроби видалення до будь-яких непередбачуваних маніпуляцій [14]. Прикладами є нелінійні та лінійні фільтри (збільшення різкості, розмиття, середня фільтрація), перефарбовування, слабке стиснення, повторна дискретизація, масштабування, додавання шуму, обертання, кадрування, друк/сканування/копіювання, аналого-цифрове та цифрово-аналогове перетворення.

Ємність

Ємність — це кількість інформації, яку можна приховати відносно розміру вихідного файлу [15]. Існують компроміси між ступенем погіршення сигналу та кількістю вбудованих даних. Метод приховування даних може працювати як з високою стійкістю до модифікації, так і з високим ступенем вбудованих даних, але не одночасно. Інструмент розрядної площини охоплює методи, які застосовують вставку LSB(Last significant bit), а також маніпулювання шумом. Ці підходи є найбільш поширеними в стеганографії і їх легко застосувати в аудіо та зображенні. З незначним впливом на носій можна приховати дивовижну кількість інформації [10]. Однак ці підходи вразливі до невеликих змін, що виникають внаслідок стиснення з втратами або обробки зображень [12].

Скритність

Ця властивість потрібна для забезпечення прихованого зв'язку. Наприклад, якщо метод стеганографії використовує шумовий компонент цифрових зображень для вбудовування секретного повідомлення, він повинен робити це, не вносячи статистично значущих змін у шум на носії. Концепція скритності по суті прив'язана до статистичної моделі джерела зображення.

Якщо зловмисник має більш детальну модель джерела, він може виявити наявність прихованого повідомлення, зверніть увагу, що здатність виявити наявність прихованого повідомлення не означає можливість прочитати повідомлення [16].

2.2 Класифікація стеганографічних методів

Стеганографію можна умовно розділити на три типи:

1. Чиста стеганографія

Коли системі не потрібно змінювати секретну інформацію, це називається «чиста стеганографія», наприклад стего-ключ. Процес можна представити наступним чином: відображення E: $C \times M \rightarrow C$, де C представляє всі можливі вихідні файли, а M — усі можливі повідомлення. Хоча формула D: $C \rightarrow M$ використовується для вилучення будь-якого секретного повідомлення з файлу, звичайно, $|C| \geq |M|$ є необхідною умовою. Алгоритми тут не є загальнодоступними, лише відправники або одержувачі можуть отримати доступ до процесу вилучення та вбудовування [10].

2. Стеганографія з використанням приватного (секретного) ключа

Цей тип необхідний для зміни секретного ключа в ході спілкування. Повідомлення вбудовується у файл за допомогою секретного ключа (стего-ключа). Люди, які знають секретний ключ, можуть отримати оригінальне повідомлення всередині файлу та прочитати його. У той час як у чистій стеганографії секретний ключ більш схильний до перехоплення, у приватній стеганографії існує секретний канал і через нього обмінюються стего-ключами. У разі перехоплення приватних стеганографічних повідомлень тільки люди, які знають ключі, можуть побачити повідомлення та витягнути його [17].

3. Стеганографія з використанням публічного ключа

Цей тип покладається на засоби публічних ключів в системах які оперують відкритими та секретними ключами для захисту каналів зв’язку. На стороні відправника повідомлення кодується відкритим ключем, але тільки

секретний ключ, який математично відповідає цьому відкритому ключу, може розшифрувати повідомлення.

Переваги стеганографії з відкритим ключем очевидні в більш ефективній реалізації криптографії з відкритим ключем, вона має різні рівні безпеки, що не дозволяє іншим сторонам приєднатися до спілкування не підозрюючи про використання стеганографії, також вони повинні знати, як зламати використовуваний алгоритм [17].

Носії інформації

Ці носії можуть відноситись до різних видів даних: тексту, дисків, аудіо, зображень, звуку, мережевого трафіку або інших цифрових форм передачі даних [12]. Нижче наведені методи приховування даних:

- Приховування в тексті

Можливо, знадобиться приховати частину або весь текст, написаний в документах. Веб-браузери ігнорують деяке форматування, наприклад додаткові розриви рядків, пробіли та табуляції, тому HTML-файли використовуються для передачі даних. Ці додаткові розриви рядків, пробіли тощо не можна виявити без доступу до джерела веб-сторінки. Можна використовувати різні способи, щоб приховати текстову інформацію, наприклад кодування зі зсувом рядків, кодування зі зсувом слів [12].

- Приховування в зображенні

Зображення є об’єктом таємної комунікації. Для властивостей ЗСЛ (зорової системи людини) слід враховувати багато деталей приховування зображень, наприклад ефекти яскравості, маскування країв і контраст. Більш ніж одна частина з 30 для випадкових шаблонів може демонструвати зміни, тому ЗСЛ не має високої чутливості до змін яскравості. Іншою важливою властивістю для ЗСЛ є низька чутливість до малих просторових частот, наприклад зміни яскравості зображення [14].

- Приховування у відео

Здебільшого приховування у відео використовує прийоми, які використовуються для приховування звуку та зображення, оскільки відео

сформоване із зображень і звуків. Оскільки відео складається з рухомих зображень зі звуками, це є перевагою, так як будь-яке невелике спотворення не буде помічено користувачем через такий безперервний обсяг даних [18].

- Приховування в аудіо

Приховування в аудіо є особливо складним через його великий діапазон частот, який становить понад 1000 до 1, а також потужність понад 1000 000 до 1. Аудіосигнали також чутливі до випадкових шумів. Шум можна виявити, якщо він знаходиться в діапазоні від одного до мільйона в звуковому файлі [14]. Під час приховування в аудио користувач повинен скористатися перевагами слабкості слухової системи людини, але також повинен пам’ятати про її високу чутливість [19].

«Чарівний трикутник» — це назва сили та «нечутності» алгоритму, який використовується для приховування; це три парадоксальні вимоги для приховування в звуці. Але використання слухової системи людини є більш надійним способом, гучні звуки можуть перекривати тихі через обмежений диференціальний діапазон ССЛ. ССЛ сприймає лише відносну фазу, але не сприймає абсолютну фазу. Нарешті, існують деякі спотворення навколишнього середовища, настільки поширені, що в більшості випадків їх ігнорує слухач. Ці недоліки можуть бути використані методами приховування даних. Одним із недоліків, які можна було б усунути в процесі приховування даних, це ігнорування шумів навколишнього середовища.

Щоб зрозуміти методи приховування інформації в аудіофайлі, спочатку слід зрозуміти передачу аудіосигналу. Крім того, необхідно розуміти цифрове подання аудіо та середовище передачі. Вміст цифрового представлення аудіофайлу це:

1. Квантування вибірки
2. Частота дискретизації

Квантування — це лінійний 16-розрядний або 8-розрядний логарифм. Діапазон частоти дискретизації становить від 8 кГц до 44,1 кГц (якість компакт-дисків. Теорема Найквіста показує, що «максимальна

використовувана частота звуку обмежена частотою дискретизації/2». Це важливо в техніках приховування [20]).

Далі будуть наведені безпосередньо методи аудиостеганографії.

Зміна амплітуди

Популярний метод зміни амплітуди називається вставкою LSB, цей метод може бути реалізований в стеганографії або водяних знаках [10]. Під час вставки методом LSB можуть виникнути помилки під час оцифрування аудіосигналів.

Очевидно, що в цьому методі дані кодуються в LBS аудіоданих. Наприклад, візьмемо 16-розрядний вибірковий файл, приховування даних може використовувати найменші чотири значущих біти. Тим не менш, з прихованими даними може статися спотворення, а також дані можуть бути виявлені, в прихованому стані. Приховані дані можуть бути виявлені за схожість або шум. Зміни в LSB можуть викликати чутний шум [19]. Під час копіювання, стиснення або аналого-цифрового, цифрово-аналогово перетворення можливий ризик спотворення даних [20].

Фазове кодування

Іншим методом приховування аудіоданих є фазове кодування, воно відносить фазу звукового сигналу до опорної фази, яка представляє дані. Усі фази коригуються відповідно до необхідної відносної фази.

Фазове кодування ефективне для кодування сигналів з метою регулювання рівня шуму. Коли фази між частотними сегментами змінюються, з’являється очевидна дисперсія. У будь-якому випадку, можуть бути виявлені неприйнятні модифікації, якщо це пов’язано з невеликими змінами фази. У цьому випадку може бути виконано нечутне кодування. Зазвичай, слухачі не можуть помітити зміни в аудіо, коли фазове кодування виконується з плавними фазовими зсувами, це робить фазове кодування одним з найефективніших методів з точки зору рівня шуму [20]. Фазове кодування має дуже низький рівень місткості даних. Власне, це і є основним недоліком цього методу [21].

Кодування з розширеним спектром

Іншим методом приховування інформації в аудіосигналах є кодування з розширеним спектром. Збереження потужності та пропускної здатності є необхідністю передачі аудіоданих, тому канал зв'язку прагне зробити частотну область аудіосигналів якомога вузькою. В [14] обговорюється метод, який називається «Direct Sequence Spread Spectrum» (DSSS). У цьому методі чіп використовується для розмноження звукового сигналу з метою його поширення, послідовність більшої довжини модулюється на певному рівні. Частоту дискретизації можна вибрати як швидкість мікросхеми, оскільки сигнали хоста вже є дискретними сигналами. Однією з проблем у використанні DSSS є те, як правильно визначити початок і кінець квантів фазової синхронизації мікросхеми.

Метод кодування «Frequency Hopped Spread Spectrum» (FHSS) також використовує розширений спектр, у цій техніці частота несучого сигналу швидко переміщається з певної частоти на іншу [10]. Новий аудіосигнал розбивається на невеликі частини, кожна частина передається спеціальною частотою, пов’язаною з нею.

Використання кодування з розширеним спектром може використовуватись через його стійкість до змін. Важко змінити вбудовані дані, не зробивши видимих змін даних носія.

Приховування даних у відлунні

У цьому методі кодування аудіосигнал розглядається як сигнал хоста для вбудовування даних шляхом надання відлуння. Приховування даних здійснюється шляхом налаштування трьох параметрів відлуння: швидкості загасання, початкової амплітуди та зміщення (затримки). Коли зсув між новим сигналом і відлунням зменшується, два сигнали об’єднуються. У певний момент людина не може помітити ніякої різниці між цими двома сигналами. Ехо тут вводиться як додатковий резонанс [14].

Дані можуть бути приховані в аудіофайлі, використовуючи різні часові затримки, які відокремлюють новий і ехо-сигнали один від одного. Новий

сигнал може бути розділений на кілька невеликих частин, щоб дозволити вбудовувати більше одного біта, кожна частина цього сигналу може бути відтворена до певного біта [10].

Ця техніка особливо добре працює для звукових файлів, які не містять додаткових недоліків, таких як тиша, або коли є втрати в кодуванні або деякі шуми в сигналі [19].

2.3 Методи оцінки якості стеганосистеми

AWGN - Additive White Gaussian Noise

Щоб оцінити продуктивність розробленого методу, AWGN (Additive White Gaussian Noise) з різними значеннями дисперсії може додаватися до стего-файлу перед вилученням секретного повідомлення. AWGN — це базова модель шуму, яка використовується в теорії інформації для імітації впливу багатьох випадкових процесів, що відбуваються в природі. Модифікатори позначають конкретні характеристики:

- Адитивний (Additive), оскільки він додається до будь-якого шуму, який може бути притаманним інформаційній системі;
- Білий(White) посилається на ідею, що він має рівномірну потужність у смузі частот для інформаційної системи. Це аналогія з білим кольором, який має рівномірне випромінювання на всіх частотах у видимому спектрі.
- Гаусів (Gaussian), оскільки він має нормальний розподіл у часовій області із середнім значенням часової області, рівним нулю.

Хеш-функції

Це типовий метод цілісності, який широко використовується в різних протоколах і програмах. Він приводить рядки різної довжини у відповідні рядки постійного розміру[29]. Він відіграє важливу роль у сучасній криптографії. На практиці хеш-функції залежать від того, щоб прийняти повідомлення як вхідні дані, а потім продукують результат, пов’язаний з хешами цього повідомлення. Основна ідея щодо хеш-функцій полягає в тому, що хеші діють як компактне відображення, відоме як відбитки, дайджести або

цифрові відбитки вхідного рядка. Ці функції широко використовуються для забезпечення цілісності даних у поєднанні з моделями цифрового підпису, оскільки зазвичай спочатку хешуються повідомлення, а потім підписуються хеші замість вихідного повідомлення. Як правило, хеш-функції повинні мати наступні властивості: простота обчислення та стиснення [30].

Одним з основних класів хеш-функцій є коди аутентифікації повідомлень (MAC – Message authentication code). Він дозволяє розпізнавати повідомлення за допомогою симетричних методів. Ця техніка потребує два окремих вхідних наборів даних: повідомлення та секретного ключа щоб отримати вихідні дані із постійним розміром з метою забезпечення цілісності даних і симетричної аутентифікації даних та розпізнавання в моделях із симетричним ключем. Іншим типом хеш-функцій є коди виявлення модифікацій (MDC – Modification detection code), які представляють хеші повідомлень. Основна мета MDC — спростити, у поєднанні з подальшими механізмами, цілісність даних для різних додатків. Ці коди є підкласами хеш-функцій без ключа. Також їх можна класифікувати на два класи: односторонні хеш-функції (OWHF – One-way hash functions), та хеш-функції, стійкі до зіткнень (CRHF – Collision resistant hash functions) [30].

Іншим типом хеш-функцій є Modification Detection 5 (MD5), який пропонує хеш-значення розміром 16 байт. Ця функція широко використовується в криптографічних програмах і перевірці цілісності даних [30].

Контрольна сума

Контрольна сума є одним з основних методів, які використовуються для перевірки цілісності. Її можна розрахувати для даних диска, а також її можна стабільно зберігати. Сума залежить від порівняння збережених обчислених значень з новими для кожної прочитаної порції даних. Цей метод в основному створюється з використанням методу хеш-функції. Метод контрольної суми допомагає виявити зміни в цілісності. Однак цей метод не міг запропонувати жодної допомоги у відновленні даних через невідповідність між збереженими

та обчисленими значеннями контрольних сум. Збережені контрольні суми можуть бути пошкоджені або змінені. Інша причина проблеми відновлення з використанням методу контрольної суми полягає в тому, що вона в основному обчислюється з використанням односторонньої хеш-функції, де дані потім не можуть бути відновлені, щоб запропонувати вихідні значення контрольної суми [29].

Контрольні суми широко використовуються для зменшення дублікатів в об’єктах даних, оскільки ці повторювані об’єкти мають однакове значення контрольної суми. Ці об'єкти можна розпізнати на основі використання раціонально стійкої до зіткнень моделі контрольної суми на основі порівняння контрольних сум цих об'єктів. Інше використання контрольних сум - це індексація даних. Хоча контрольні суми, стійкі до зіткнень, більші, ніж традиційні числа цілого типу, вони можуть допомогти у пропонуванні подвійної функціональності з мінімальними витратами. Контрольні суми також можна використовувати для іменування дескрипторів, що забезпечує простий метод отримання відповідних контрольних сум до блоку даних, де це, у свою чергу, допомагає підвищити ефективність перевірки цілісності [29].

2.4 Математична модель стеганосистеми

Метод Bi-LSB залежить від заміни другого правого найменш важливого біта в вихідному повідомленні на частину секретного повідомлення для підвищення безпеки.

Алгоритм включає три основні етапи: попередня обробка, вбудовування/вилучення і перевірка повідомлень. Попередня обробка застосовується для підвищення безпеки повідомлень, які мають бути приховані у файлі MP3. Етап вбудовування та вилучення включає розробку запропонованого алгоритму для файлів MP3 для вирішення поточної проблеми безпеки традиційної техніки LSB, тоді як етап перевірки повідомлень застосовується для розпізнавання ефективності прихованих повідомлень у файлах MP3.

Щоб оцінити ефективність представленого методу Bi-LSB, додається AWGN з різними значеннями дисперсії до stego-файлу перед вилученням секретного повідомлення, де потім обчислюються значення PSNR і порівнюються з тими, які були отримані перед додаванням цього шуму. Наступний рисунок описує представлений алгоритм:

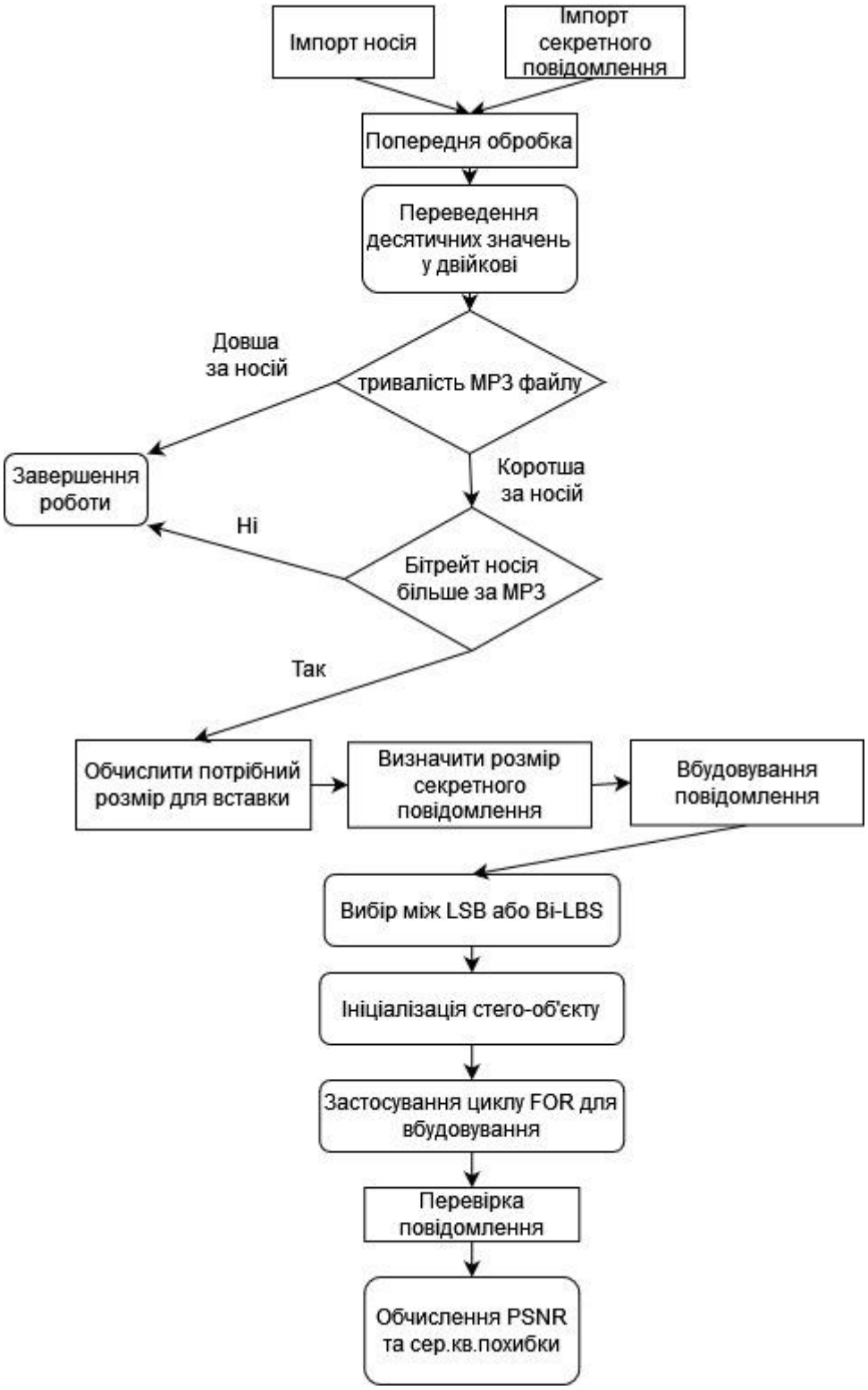

Рисунок 2.3 - Алгоритм стегосистеми

Етап попередньої обробки

Вихідне повідомлення (C), і повідомлення (M) є файлами MP3. У цій роботі секретне повідомлення перевіряється та попередньо обробляється на основі реалізації трьох методів: контрольна сума, хеш-функція і порівняння частоти для створення кодової книги. Отримані попередньо оброблені файли C і M потім використовуються в подальшому етапі.

Етап вставки і вилучення

Етап включає два основних процеси: вбудовування та вилучення. У процесі вбудовування обидва попередньо оброблені файли C і M обробляються, щоб представити стего-об’єкт (SO – Stego object). Після цього метод Bi-LSB використовується для вбудовування секретного повідомлення в носій. У процесі вилучення SO обробляється для вилучення файлу M.

Етап перевірки повідомлення

Традиційні методи стеганографії приховують секретне повідомлення, але не перевіряють, чи були внесені зміни до вмісту повідомлення під час комунікації. В алгоритм додано крок перевірки вмісту, щоб переконатися, що вилучене повідомлення точно таке ж, як оригінальне секретне повідомлення. Це робиться за допомогою трьох функцій перевірки вмісту (контрольна сума, хеш-функція (MD5), частота).

Ці методи створюють кодову книгу, яка потім використовується для порівняння з вихідним секретним повідомленням, щоб перевірити, чи це повідомлення було атаковано.

3 ІНФОРМАЦІЙНЕ І ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Формування вхідних даних

При формування файлів-носіїв аудіоданих в роботі використані файли MP3, оскільки вони характеризуються високим стиснення даних. У роботі розглядаються 5 MP3 файлів, що містять музику різних жанрів: блюз, класика, джаз, поп та R&B. При цьому частота дискретизації становить 48 кГц, що відповідає значенню параметра Bit rate 320 кбіт/с. Таблиця 3.1. ілюструє набір даних MP3, що використовувався в цій роботі.

Таблиця 3.1 – Характеристики вхідних даних

Жанр	Час	Розмір, МБ	Частота дискретизації, кГц	Bit rate, кбіт/с
Класика	2:54	6.67	48	320
Джаз	3:12	7.34	48	320
R&B	3:51	8.81	48	320
Поп	4:00	9.16	48	320
Блюз	4:41	10.7	48	320

При формування файлів-носіїв секретних повідомлень в роботі також використано файли MP3, що містить музику жанрів: блюз, джаз. При цьому частота дискретизації становить 22.05 кГц, що відповідає значенню параметра Bit rate 96 кбіт/с, та 44.1 кГц, що відповідає значенню параметра Bit rate 128 кбіт/с. Таблиця 3.2. ілюструє характеристику секретного повідомлення, що використовувалось в цій роботі.

Таблиця 3.2 – Характеристика секретного повідомлення

Жанр	Час	Розмір, МБ	Частота дискретизації, кГц	Bit rate, кбіт/с
Блюз	2:54	3.22	22,05	96

3.2 Короткий опис програмної реалізації стегосистеми

Етап попередньої обробки

На цьому етапі користувачеві пропонується вибрати аудіофайл-носій для імпорту в програму, який повинен бути в тому самому каталозі, що і програма. Метод «mp3read» використовується для імпорту носія в робочу область програми для створення наступних змінних:

- Cover: вектор, що містить усі кадри аудіофайлу носія (від -1 до +1);
- FS : частота дискретизації;
- NBITS: кількість бітів (зазвичай 8 біт);
- OPTS: структура, що містить інформацію про аудіофайл.

Підготовка файла-носія

На цьому кроці файл-носій конвертується з десяткового представлення в двійкове, методи mp3read і mp3write використовуються для читання файла-носія.

Підготовка таємного повідомлення

Після підготовки носія секретне повідомлення готується до вбудовування у носій. На цьому кроці застосовуються три методи, у кожному з яких кодова книга створюється на етапі попередньої обробки, значення кодової книги зберігаються для порівняння пізніше. На цьому кроці користувачеві пропонується вибрати аудіофайл секретного повідомлення для імпорту в систему за допомогою функцій “uigetfile” і “mp3read”; цей файл повинен знаходитись в одному каталозі з програмою. Це, у свою чергу, створює такі змінні:

- Message: вектор, що містить усі кадри аудіофайлу секретного повідомлення (від -1 до +1);
- FS_Message : частота дискредитації секретного повідомлення;
- NBITS_Message : кількість бітів секретного повідомлення (зазвичай 8);
- OPTS_Message : структура, що містить інформацію секретного повідомлення.

Після цього, значення секретного повідомлення потім перетворюються на додатні за допомогою функції «abs», які потім перетворюються з десяткових значень у двійкові. Після цього застосовуються дві перевірки. Перша перевірка – для того, щоб перевірити, чи довжина секретного повідомлення менша за носій, чи ні, для того, щоб розпізнати, чи є аудіофайл моно чи стерео за допомогою структури OPTS, зокрема змінної «OPTS_Message.fmt.mpgPad». Якщо довжина більша, генерується попередження і обчислення припиняються. У випадку моно створюється один вектор-стовпчик, якщо стерео — двовимірна матриця, тобто лівий і правий канали.

Друга перевірка застосовується для перевірки бітрейту секретного повідомлення і носія. Бітрейт носія має бути більшим за бітрейт секретного повідомлення. Якщо умова не виконується, то генерується попередження і робота припиняється. Після цього користувача запитують, чи хоче він вставити весь аудіофайл чи лише його частину. Після цього секретне повідомлення перетворюється з десяткового вигляду в двійкове.

Етап вставки повідомлення

На цьому етапі користувачеві пропонується перевірити бажаний тип техніки LSB - традиційну або покращену. Після цього стего-об'єкт ініціалізується вектором носія. Потім генерується випадкове значення, разом з випадковим значенням файла-носія. Секретне повідомлення в двійковому вигляді записується в один рядок для легшого пошуку найменш важливого біта та ініціалізації двох лічильників: «k» і «w». Потім створюється цикл «FOR» для вбудовування повідомлення, яке починається з раніше згенерованого випадкового значення із використанням кроку «mod(i,100)» для вбудовування секретного повідомлення в носій.

Для традиційної техніки в кожній із ітерацій циклу обраний байт змінюється за допомогою наступної логіки:

- Якщо використовується 4-LSB, то біти з 2-го по 5-й замінюються першими доступними 4 бітами секретного повідомлення;

- Якщо використовується 2-LSB, то біти з 2-го по 3-й замінюються першими доступними 2 бітами секретного повідомлення;
- Якщо використовується 1-LSB, лише 2-й біт замінюється першим бітом, доступним у секретному повідомленні.

Для покращеної техніки Bi-LSB у кожній із ітерацій циклу біти приховуються таким чином:

- Приховування 1 біта в першому байті;
- Приховування 2 бітів у другому байті;
- Приховування 2 бітів у третьому байті;
- Приховування 1 біта у четвертому байті

Після цього змінений байт знову перетворюється з двійкового в десятковий для створення стего-об’єкта. У кожній із ітерацій лічильник «k» оновлюється залежно від обраного алгоритму LSB.

Перевірка (оцінка спотворення) повідомлення

На цьому етапі обчислюються відношення пікового сигналу до шуму (PSNR) і середньоквадратична помилка (MSE). Обидва показники представляють дві метрики помилок, які використовуються для порівняння якості. PSNR і MSE можна розрахувати відповідно до таких формул:

$$MSE = \frac{1}{N} \cdot \sum_{i=1}^{N} \big(X(i) - Y(i)\big)^2 \tag{3.1}$$

$$PSNR = 10log_{10} \frac{MAX^2}{MSE} \tag{3.2}$$

де X – носій, Y – стего-об’єкт, N – розмір носія, MAX – максимальна зміна до файлу-носія. Для порівняння двох алгоритмів аудіостеганографії або оцінки впливу шуму на якість одного і того ж самого алгоритму в роботі застосовується відношення:

$$Deg = 1 - \frac{PSNR_i}{PSNR_j} \tag{3.3}$$

де $PSNR_i$ та $PSNR_j$ - відношення пікового сигналу до шуму при різних умовах застосування аудіостеганографії.

Оцінка цілісності повідомлення

На цьому етапі застосовуються різні методи перевірки цілісності. При цьому зберігається кодова книга відповідно до застосованого методу. Метод контрольної суми спрощує перевірку за рахунок даних в кінці отриманого повідомлення. Хеш-функція являє собою представлення даних різного розміру даними фіксованого розміру (хеш-суми, хеші).

Програмна реалізація

Програмна реалізація цих етапів виконувалась в середовищі для наукових і інженерних розрахунків MATLAB у взгляді m-функцій та m-сценаріїв (Табл.3.3)

Таблиця 3.3 – Основні m-функції та m-сценарії

Назва	Вхідні параметри	Вихідні параметри	Опис
mp3read	FILE – ім'я файлу N – час зчитування аудіосигналу з файлу, MONO - мітка «моно/стерео» DOWNSAMP – коефіцієнт зниження дискретизації, DELAY – відтермінування при формуванні вихідного масиві	Y – вихідний масив FS – частота дискретизації NBITS - розрядність OPTS – масив інших характеристик структури mp3 файлу	Функція зчитує MP3-файл і перетворює аудіосигнал на вектор Y
mysystem	cmd – системна команда	w – лог повідомлень про помилку	Функція запуску системних команд і логування повідомлень про помилку
tokenize	s – вхідний рядок t - розмежувач	a – масив рядків	Функція, що перетворює рядок символів, що розділені розмежувачем, на масив рядків
mp3write	D – вхідний масив SR - частота дискретизації NBITS - розрядність FILE - ім'я файлу OPTIONS - масив інших характеристик структури mp3 файлу	-	Функція записує MP3-файл з вхідного вектору D
DataHash	Data - вхідний масив Opt – масив характеристик даних вхідного масиву	Hash – контрольна сума	Функція для обчислення контрольної суми масивів
ConvertObject	DataObj – вхідний масив даних	DataBin – бінеаризований масив даних	Функція для конвертування масивів в бінарну форму

Продовження табл. 3.3

Назва	Вхідні параметри	Вихідні параметри	Опис
ConvertBinObject	DataBin – вхідний бінеаризований масив даних	DataObj –масив даних	Функція для конвертування бінарних масивів в цілочисельні
blsb_enc	DataBinIn – бінеаризований масив аудіоданих з файлу-носія DataBin – бінеаризований масив аудіоданих секретного повідомлення pass – пароль	DataBinOut – бінеаризований масив аудіоданих з вбудованим секретним повідомленням	Функція вбудовування секретного повідомлення в носій за модифікованим алгоритмом
blsb_dec	DataBinIn – бінеаризований масив аудіоданих з вбудованим секретним повідомленням pass – пароль	DataBin – бінеаризований масив аудіоданих секретного повідомлення	Функція вилучення секретного повідомлення за модифікованим алгоритмом
lsb_enc	DataBinIn – бінеаризований масив аудіоданих з файлу-носія DataBin – бінеаризований масив аудіоданих секретного повідомлення pass – пароль	DataBinOut – бінеаризований масив аудіоданих з вбудованим секретним повідомленням	Функція вбудовування секретного повідомлення в носій за стандартним алгоритмом
lsb_dec	DataBinIn – бінеаризований масив аудіоданих з вбудованим секретним повідомленням pass – пароль	DataBin – бінеаризований масив аудіоданих секретного повідомлення	Функція вилучення секретного повідомлення за стандартним алгоритмом

Програмний код наведено в Додатку.

3.3 Аналіз результатів

Далі наведені результати роботи програми з використанням традиційні техніки LSB і вдосконалену. Секретне повідомлення у жанрі «Блюз» було вбудоване у файли-носії, після чого було розраховано значення PSNR, відповідно до методу вбудовування. Результат наведено в таблиці 3.4.

Таблиця 3.4 – Відношення пікового сигналу до шуму

Жанр носія	Час	Розмір, МБ	Жанр прихованого повідомлення	4-LSB	2-LSB	1-LSB	BLSB
Блюз	04:41	10.7	Блюз	44.3862	63.4260	76.2214	95.2852
Класика	03:12	8.67	Блюз	35.5372	45.8897	57.2895	75.8143
Джаз	03:12	8.34	Блюз	47.3317	69.7706	83.1974	102.7913
Поп	04:00	9.16	Блюз	38.7945	50.9703	57.9350	73.4542
R&B	03:51	8.81	Блюз	46.7376	68.5931	81.9648	101.4737

Можна побачити, що значення PSNR запропонованого методу кращі за такі, отримані з використанням інших методів LSB. Порівняння значень PSNR між методом BLSB та традиційними методами після вбудовування повідомлення жанру «Блюз» у носії наведено в таблиці 3.5.

Таблиця 3.5 – Порівняння значень PSNR залежно від методу

Жанр носія	Покращення PSNR між BLSB та 4-LSB (%)	Покращення PSNR між BLSB та 2-LSB (%)	Покращення PSNR між BLSB та 1-LSB (%)
Блюз	53,4	33,4	20,0
Класика	53,1	39,4	24,4
Джаз	53,9	32,1	19,0
Поп	47,1	30,6	21,1
R&B	53,9	32,4	19,2

В таблиці 3.6 наведені значення PSNR без додавання AWGN та з його додаванням до стего-об’єкту.

Таблиця 3.6 – Порівняння PSNR для методу BLSB з додаванням AWGN

Жанр носія	Жанр прихованого повідомлення	PSNR для методу BLSB	Значення AWGN	PSNR з урахуванням AWGN	Розбіжність (%)
Блюз	Блюз	95.2852	0.1	92.5227	2.89
Класика	Блюз	75.8143	0.1	70.5173	6.98
Джаз	Блюз	102.7913	0.1	98.6331	4.04
Поп	Блюз	73.4542	0.1	70.5211	3.99
R&B	Блюз	101.4737	0.1	97.5227	3.89

Можна зробити висновок, що спостерігається очевидна деградація значення PSNR після додавання AWGN.

Алгоритм виконує перевірку цілісності повідомлення шляхом порівняння його характеристик перед вбудовуванням та після вилучення. Це виконується за допомогою двох прийомів: порівняння контрольної суми та хеш-функції. В таблицях 3.6 і 3.7 наведено результати перевірки збігу характеристик повідомлення без додавання AWGN та з його додаванням.

Таблиця 3.7 – Результат перевірки повідомлення без додавання AWGN

Жанр носія	Жанр прихованого повідомлення	Збіг контрольної суми (%)	Збіг хеш-функції (%)
Блюз	Блюз	100	100
Класика	Блюз	100	100
Джаз	Блюз	100	100
Поп	Блюз	100	100
R&B	Блюз	100	100

Таблиця 3.8 – Результат перевірки повідомлення з додаванням AWGN

Жанр носія	Жанр прихованого повідомлення	Збіг контрольної суми (%)	Збіг хеш-функції (%)
Блюз	Блюз	20.9695	86.2071
Класика	Блюз	21.0825	91.4787
Джаз	Блюз	20.9695	86.2071
Поп	Блюз	20.9869	90.6557
R&B	Блюз	20.9925	81.8492

Як видно з таблиць, після додавання AWGN, секретне повідомлення втрачає свою початкову якість.

Для порівняння між методами BLSB та 1-LSB, в таблиці 3.9 наведено отримані значення PSNR після додавання AWGN, а також в таблиці 3.10 наведено результат обробки повідомлення методами перевірки цілісності після додавання AWGN.

Таблиця 3.9 – Порівняння PSNR для методу 1-LSB з додаванням AWGN

Жанр носія	Жанр прихованого повідомлення	PSNR для методу 1-LSB	Значення AWGN	PSNR з урахуванням AWGN
Блюз	Блюз	76.2214	0.1	65.1109
Класика	Блюз	57.2895	0.1	48.0195
Джаз	Блюз	83.1974	0.1	71.9531
Поп	Блюз	57.9350	0.1	47.1057
R&B	Блюз	81.9648	0.1	69.9001

Таблиця 3.10 – Результат перевірки повідомлення з додаванням AWGN методом 1-LSB

Жанр носія	Жанр прихованого повідомлення	Збіг контрольної суми (%)	Збіг хеш-функції (%)
Блюз	Блюз	18.50	83.59
Класика	Блюз	20.45	87.43
Джаз	Блюз	17.51	82.19
Поп	Блюз	18.11	88.95
R&B	Блюз	18.27	79.47

Згідно з результатами, представленими в таблицях, метод BLSB є більш ефективним в порівнянні з іншими методами.

ВИСНОВКИ

В роботі було проведено розробку вдосконаленого методу аудіостеганографії у форматі Bi-Last Significant Bit (Bi-LSB) MP3, що дозволяє виконувати як шифрування секретного повідомлення, так і гарантувати його цілісність. При цьому було виконано такі завдання:

1) Сформовано вхідний математичний опис аудіостенографічної системи.

2) Розроблено математичну модель функціонування аудіостенографічної системи.

3) Розроблено методи оцінки якості функціонування аудіостенографічної системи.

4) Розроблено і програмно реалізовано алгоритм шифрування секретного повідомлення аудіостенографічною системою.

5) Розроблено і програмно реалізовано алгоритм гарантування цілісності секретного повідомлення аудіостенографічною системою.

6) Виконано тестування аудіостенографічної системи.

СПИСОК ЛІТЕРАТУРИ

1. Stein J. Y. Digital Signal Processing: a Computer Science Perspective. Wiley-Interscience, New York, 2000. – 856 p.
2. Spanias A., Painter T., Atti V. Audio Signal Processing and Coding. Wiley-Interscience, New Jersey, 2006. – 541 p.
3. Supurovic P. MPEG Audio Compression Basics. – http://www.chested. chalmers.se\~kf96svgu
4. Brandenburg K. MP3 and AAC Explained. – Proceeding of AES. 17th International Conference on High Quality Audio Coding. – p. 24-35.
5. Pan D. A Tutorial on MPEG / Audio Compression. IEEE Multimedia, 1995. – p. 60 – 74.
6. Maciak L., Ponniah M., Sharma. R. MP3 Steganography, 2008.– 846 p.
7. ID3. The Private Life of MP3 Frames, 2014. – www.id3.org/MP3frame.html
8. Provos N. Probabilistic Method for Improving Information Hiding. CITI Technical Report, 2001. –120 p.
9. Kivanc M. Information Hiding Codes and their Applications to Images and Audio, PhD Thesis, University of Illinois at Urbana-Champaign, 2002. – 163 p.
10. Katzenbeisser S., Petitcolas F. Information Hiding Techniques for Steganography and Digital Watermarking. Artech House Inc., 2000. – 210 p.
11. Matthews C. Behind The Music: Principles of Audio Steganography, 2003. – 84 p.
12. Johnson N.F., Duricn Z., Jajodia S. Information Hiding: Steganography and Watermarking Attack and Counter measurements. Kluwer Academic Publishers, USA, 2001. – 137 p.
13. Cacciaguerra S., Ferretti S. Data Hiding: Steganography and Copyright Marking. Department of Computer Science, University of Bologna, Italy.– http://www.cs.unibo.it/~scacciag/home-file/teach/datahiding.pdf
14. Bender W., Gruhl D., Morimoto N., Lu A. Techniques for Data Hiding, IBM System Journal, 1996. – vol. 35. – p. 187-203.

15. Lin T., Delp J. A Review of Data Hiding in Digital Images. – http://www.ece.purdue.edu/~ace
16. Fridrich J. Applications of Data Hiding in Digital Images. – http://www.ssie.Binghamton.edu/~jirif.
17. Dunbar B. Steganographic Techniques and their use in an Open Systems Environment, SANS Institute, 2002. – 263 p.
18. Karen R. Steganography and Steganalysis. – www.krenn.n1/univ/cry/ steg/article.pdf.
19. Sellars D. An Introduction to Steganography", University of Camberge, 2003. – 74 p.
20. Yang Y. Digital Watermarking Technology", Faculty of Computer Science, Dalhousie University. – http://www.cs.dal.ca/~yyang/6505/6605.pdf.
21. Polpitiya A.D., Khan W.J. Information Hiding in Audio Files with Encryption, Data Security Project, Washington University, 2001. – 52 p.
22. Fricker, R., Schonlau, M. Advantages and Disadvantages of Internet Research Surveys: Evidence from the Literature. Field Methods, 2002. – №14(4). – p. 347-367.
23. Feruza, Y., Kim, T. IT Security Review: Privacy, Protection, Access Control, Assurance and System Security. International Journal of Multimedia and Ubiquitous Engineering, 2007.– № 2(2).– p. 17- 32.
24. Lenti J. Steganographic Methods. Department Of Control Engineering and Information Technology, Budapest University. Periodica Poltechnica Ser. El. Eng., 2000.– № 44. – p. 249–258.
25. Rahim, L. B. A., Bhattacharjee, S., Aziz, I. B. An Audio Steganography Technique to Maximize Data Hiding Capacity along with Least Modification of Host. In Proceedings of the First International Conference on Advanced Data and Information Engineering (DaEng- 2013) Springer Singapore. – p. 277-289

26. Mandala, J., Kotagiri, S., Kapala, K. Watermarking Scheme for Color Images. International Journal of Emerging Trends & Technology in Computer Science (IJETTCS), 2014.– № 2 (5). – p. 179-182.
27. Manimegalai, P., Gomathi, K. S., Ponniselvi, D., Santha, M. The Image Steganography and Steganalysis Based On Peak-Shaped Technique for MP3 Audio and Video. International Journal of Computer Science and Mobile Computing, 2014. – № 3 (1). – p. 300-308.
28. Scagliola, M., Pérez, F., Guccione, P. An Extended Analysis of Discrete Fourier Transform - Rational Dither Modulation For Non-White Hosts, 1st IEEE International Workshop On Information Forensics and Security, 2009. – p.146-150.
29. Sivathanu, G. Wright, C. P and Zadok, E. Ensuring Data Integrity in Storage: Techniques and Applications. a report submitted to Stony Brook University, 2005.– 85 p.
30. Menezes, A., van Oorschot, P. and Vanstone, S. Hash Functions and Data Integrity, Handbook of Applied Cryptography. By CRC Press, 1996.– 372 p.

ДОДАТОК

```
function [Y,FS,NBITS,OPTS] = mp3read(FILE,N,MONO,DOWNSAMP,DELAY)
path = fileparts(which('mp3read'));
tmpdir = 'E:\sounds\tmp\';
if isempty(tmpdir) || exist(tmpdir,'file')==0 tmpdir = '/tmp';
end
if exist(tmpdir,'file')==0 tmpdir = '';
end
mpg123 = 'E:\sounds\mpg123.exe'; mp3info = 'E:\sounds\mp3info.exe'; ame = 'E:\sounds\lame.exe';
ext = lower(computer); if ispc
ext = 'exe'; rmcmd = 'del';
end
mpg123 = fullfile(path,['mpg123.',ext]); mp3info = fullfile(path,['mp3info.',ext]);
if length(FILE) > 6 && (strcmp(lower(FILE(1:7)),'http://') == 1 ...
|| strcmp(lower(FILE(1:6)),'ftp://'))
else
OVERNET = 0;
end
N = 0;
end
if ischar(N)
FMT = lower(N);
N = 0;
end
if length(N) == 1
end
if nargin < 3 forcemono = 0;
else
FMT = lower(MONO);
MONO = 0;
end
forcemono = (MONO ~= 0); end
if nargin < 4 downsamp = 1;
else
downsamp = DOWNSAMP; end
if downsamp ~= 1 && downsamp ~= 2 && downsamp ~= 4 error('DOWNSAMP can only be 1, 2, or 4');
end
if strcmp(FMT,'native') == 0 && strcmp(FMT,'double') == 0 && ... strcmp(FMT,'size') == 0
error(['FMT must be ''native'' or ''double'' (or ''size''), not ''',FMT,'''']);
end
if isempty(ext)
FILE = [FILE, '.mp3'];
end
if ~OVERNET
cmd = ['"',mp3info, '" -r m -p "%Q %u %b %r %v * %C %e %E %L %O %o %p" "', FILE,'"'];
nums = str2num(w(1:(starpos - 2))); strs = tokenize(w((starpos+2):end));
SR = nums(1); nframes = nums(2);
nchans = 2 - strcmp(strs{6}, 'mono'); layer = length(strs{4});
bitrate = nums(4)*1000; mpgv = nums(5);
smpspfrm = 384;
elseif SR < 32000 && layer ==3 smpspfrm = 576;
if mpgv == 1
error('SR < 32000 but mpeg version = 1'); end
else
smpspfrm = 1152; end
OPTS.fmt.mpgBitrate = bitrate; OPTS.fmt.mpgVersion = mpgv;
OPTS.fmt.mpgLayer = strs{4}; OPTS.fmt.mpgOriginal = strs{5}; OPTS.fmt.mpgChanmode = strs{6};
OPTS.fmt.mpgPad = strs{7};
OPTS.fmt.mpgSampsPerFrame = smpspfrm;
else
SR = 44100;
nframes = 0; end
if SR == 16000 && downsamp == 4
error('mpg123 will not downsample 16 kHz files by 4 (only 2)'); end
if MPG123059
mpg123delay44kHz = 2257;
```

```
kHz sampling
if SR == 16000
rawdelay = mpg123delay16kHz; else
rawdelay = mpg123delay44kHz;
delay = round(rawdelay/downsamp); else
end else
delay = DELAY; end
if downsamp == 1 downsampstr = '';
else
downsampstr = [' -',num2str(downsamp)]; end
FS = SR/downsamp;
if forcemono == 1 nchans = 1; chansstr = ' -m';
else
chansstr = ''; end
if strcmp(FMT,'size') == 1
Y = [floor(smpspfrm*nframes/downsamp)-delay, nchans]; else
tmpfile = fullfile(tmpdir, ['tmp',num2str(round(1000*rand(1))),'.wav']);
skipx = 0;
skipblks = 0; skipstr = ''; sttfrm = N(1)-1;
if sttfrm > 0
skipblks = floor(sttfrm*downsamp/smpspfrm); skipx = sttfrm - (skipblks*smpspfrm/downsamp);
skipstr = [' -k ', num2str(skipblks)];
end
skipx = skipx + delay;
lenstr = ''; endfrm = -1;
decblk = 0;
if length(N) > 1 endfrm = N(2);
if endfrm > sttfrm
decblk = ceil((endfrm+delay)*downsamp/smpspfrm) - skipblks + 10;
end
cmd=['"',mpg123,'"', downsampstr, chansstr, skipstr, lenstr, ...
        ' -q -w "', tmpfile,'" "',FILE,'"'];
Y = wavread(tmpfile);
if decblk > 0 && length(Y) < decblk*smpspfrm/downsamp
returned ',num2str(length(Y))]); end
mysystem([rmcmd,' "', tmpfile,'"']);
if skipx+endfrm-sttfrm > length(Y) endfrm = length(Y)+sttfrm-skipx;
end
if endfrm > sttfrm
Y = Y(skipx+(1:(endfrm-sttfrm)),:); elseif skipx > 0
Y = Y((skipx+1):end,:); end
Y = int16((2^15)*Y);
end end
function w = mysystem(cmd)
if s ~= 0
error(['unable to execute ',cmd,' (',w,')']); end
w = w((1+max([0,findstr(w,10)])):end);

function a = tokenize(s,t)
a = []; p = 1;
n = 1;
l = length(s);
nss = findstr([s(p:end),t],t); for ns = nss
p = p+1; else
if p <= l
a{n} = s(p:(ns-1)); n = n+1;
p = ns+1; end
end end

function mp3write(D,SR,NBITS,FILE,OPTIONS)
[path] = fileparts(which('mp3write'));
if isempty(tmpdir) || exist(tmpdir,'file')==0 tmpdir = '/tmp';
end
if exist(tmpdir,'file')==0 tmpdir = '';
end
if ispc
ext = 'exe'; rmcmd = 'del';
end
```

```
lame = fullfile(path,['lame.',ext]);
OPTIONS = FILE;
end
FILE = NBITS;
NBITS = 16;
end
if mynargin < 5
OPTIONS = '--quiet -h';
[nr, nc] = size(D); if nc < nr
D = D';
[nr, nc] = size(D); end
if isempty(ext)
FILE = [FILE, '.mp3'];
end
nchan = nr; nfrm = nc;
if nchan == 1 monostring = ' -m m';
else
monostring = ''; end
lameopts = [' ', OPTIONS, monostring, ' '];
if length(which('popenw')) > 0
cmd = ['"',lame,'"', lameopts, '-r -s ',num2str(SR),' - "',FILE,'"'];
p = popenw(cmd); if p < 0
error(['Error running popen(',cmd,')']); end
nrem = nfrm; base = 0;
while nrem > 0
thistime = min(nrem, blksiz);
done = popenw(p,32767*D(:,base+(1:thistime)),'int16be');
nrem = nrem - thistime;
base = base + thistime;
else
disp('Warning: popenw not available, writing temporary file');
tmpfile = fullfile(tmpdir,['tmp',num2str(round(1000*rand(1))),'.wav']);
wavwrite(D',SR,tmpfile);
cmd = ['"',lame,'"', lameopts, '"',tmpfile, '" "', FILE, '"'];
mysystem(cmd);
mysystem([rmcmd, ' "', tmpfile,'"']); end

function w = mysystem(cmd)
if s ~= 0
error(['unable to execute ',cmd,' (',w,')']); end
w = w((1+max([0,findstr(w,10)])):end);

function Hash = DataHash(Data, Opt)
if isempty(usetypecastx)
if ~usejava('jvm')
Error_L('NoJava', 'Java is required.'); end
usetypecastx = ~isempty(which('typecastx'));
end
Method = 'MD5'; OutFormat = 'hex'; isFile  = false;
isBin   = false;
nArg = nargin; if nArg == 2
if isa(Opt, 'struct') == 0
end
if isfield(Opt, 'Method') Method = upper(Opt.Method);
end
if isfield(Opt, 'Format') OutFormat = Opt.Format;
end
if strcmpi(Opt.Input, 'File') isFile = true;
if ischar(Data) == 0
Error_L('CannotOpen', 'lst input FileName must be a string'); end
elseif strncmpi(Opt.Input, 'bin', 3)
if (isnumeric(Data) || ischar(Data) || islogical(Data)) == 0 || ...
        issparse(Data)
```

```
Error_L('BadDataType', ...
'1st input must be numeric, CHAR or LOGICAL for binary
input.');
end end
end
elseif nArg ~= 1
end
try
Engine = java.security.MessageDigest.getInstance(Method); catch
Error_L('BadInput2', 'Invalid algorithm: [%s].', Method); end
if isFile
Found = FileExist(Data); if ~Found
Error_L('FileNotFound', 'File not found: %s.', Data); end
FID = fopen(Data, 'r'); if FID < 0
Error_L('CannotOpen', 'Cannot open file: %s.', Data); end
[Data, Count] = fread(FID, Chunk, '*uint8'); Engine.update(Data);
while Count == Chunk
[Data, Count] = fread(FID, Chunk, '*uint8'); Engine.update(Data);
end fclose(FID);
if usetypecastx
Hash = typecastx(Engine.digest, 'uint8'); else
Hash = typecast(Engine.digest, 'uint8'); end
elseif isBin
if isempty(Data)
 if usetypecastx
 Hash = typecastx(Engine.digest, 'uint8'); else
 Hash = typecast(Engine.digest, 'uint8'); end
 elseif usetypecastx
 Engine.update(typecastx(Data(:), 'uint8')); else
 Engine.update(typecastx(real(Data(:)), 'uint8'));
 Engine.update(typecastx(imag(Data(:)),  'uint8'));
 end
 Hash = typecastx(Engine.digest, 'uint8');
 else
 if isreal(Data) Engine.update(typecast(Data(:), 'uint8'));
 else
 Engine.update(typecast(real(Data(:)), 'uint8'));
 Engine.update(typecast(imag(Data(:)),  'uint8'));
 end
 elseif islogical(Data)
 elseif ischar(Data)
 Hash = typecast(Engine.digest, 'uint8'); end
 elseif usetypecastx
 Hash    = typecastx(Engine.digest, 'uint8');
 else
 Hash    = typecast(Engine.digest, 'uint8'); end
 switch OutFormat case 'hex'
 Hash = sprintf('%.2x', double(Hash)); case 'HEX'
 Hash = sprintf('%.2X', double(Hash)); case 'double'
 Hash = double(reshape(Hash, 1, [])); case 'uint8'
 Hash = reshape(Hash, 1, []); case 'base64'
 Hash = fBase64_enc(double(Hash)); otherwise
 Error_L('BadOutFormat', ...
 '[Opt.Format] must be: HEX, hex, uint8, double, base64.');
 end
 function Engine = CoreHash_(Data, Engine)
 typecastx([ndims(Data), size(Data)], 'uint8')]);
 if issparse(Data)
 [i1, i2, Data] = find(Data); Engine.update(typecastx(i1, 'uint8'));
 Engine.update(typecastx(i2, 'uint8'));
 end
 if isstruct(Data)
 F   = sort(fieldnames(Data));
 for iS = 1:numel(Data)
 Engine = CoreHash_(Data(iS).(F{iField}), Engine); end
 end
```

```
elseif iscell(Data)
Engine = CoreHash_(Data{iS}, Engine); end
elseif isnumeric(Data) || islogical(Data) ||
    ischar(Data) if isempty(Data) == 0
if isreal(Data)
else
Engine.update(typecastx(real(Data(:)), 'uint8'));
Engine.update(typecastx(imag(Data(:)),  'uint8'));
end end
elseif isa(Data, 'function_handle')
Engine = CoreHash_(ConvertFuncHandle(Data), Engine);
elseif (isobject(Data) || isjava(Data)) && ismethod(Data, 'hashCode')
    Engine = CoreHash_(Data.hashCode, Engine);
else
Engine = CoreHash_(ConvertObject(Data), Engine); catch
warning(['JSimon:', mfilename, ':BadDataType'], ...
    ['Type of variable not considered: ', class(Data)]);
end end

function Engine = CoreHash(Data, Engine)
Engine.update([uint8(class(Data)),  ...
typecast([ndims(Data), size(Data)], 'uint8')]);
if isstruct(Data)
F   = sort(fieldnames(Data));
for iS = 1:numel(Data)
Engine = CoreHash(Data(iS).(F{iField}), Engine); end
end
elseif iscell(Data)
Engine = CoreHash(Data{iS}, Engine); end
elseif isempty(Data) elseif isnumeric(Data)
if isreal(Data) Engine.update(typecast(Data(:), 'uint8'));
else
Engine.update(typecast(real(Data(:)), 'uint8'));
Engine.update(typecast(imag(Data(:)),  'uint8'));
end
elseif islogical(Data)
elseif ischar(Data)
elseif isa(Data, 'function_handle')
Engine = CoreHash(ConvertFuncHandle(Data), Engine);
elseif (isobject(Data) || isjava(Data)) && ismethod(Data, 'hashCode')
    Engine = CoreHash(Data.hashCode, Engine);
else
try
Engine = CoreHash(ConvertObject(Data), Engine); catch
warning(['JSimon:', mfilename, ':BadDataType'], ...
    ['Type of variable not considered: ', class(Data)]);
end end

function DataBin = ConvertObject(DataObj)
DataBin = uint8(DataObj);

function Out = fBase64_enc(In)
Pool = [65:90, 97:122, 48:57, 43, 47];
v8  = [128; 64; 32; 16; 8; 4; 2; 1];
v6  = [32, 16, 8, 4, 2, 1];
In = reshape(In, 1, []);
X   = rem(floor(In(ones(8, 1), :) ./ v8(:, ones(length(In), 1))), 2);
Y   = reshape([X(:); zeros(6 - rem(numel(X), 6), 1)], 6, []);
Out = char(Pool(1 + v6 * Y));
function Ex = FileExist(FileName)
dirFile = dir(FileName); if length(dirFile) == 1
Ex = ~(dirFile.isdir); else
Ex = false; end

function Error_L(ID, varargin)
error(['JSimon:', mfilename, ':', ID], ['*** %s: ', varargin{1}], ...
    mfilename, varargin{2:nargin - 1});
```