Міністерство освіти і науки України
Харківський національний університет імені В. Н. Каразіна

ПРИХОВУВАННЯ ДАНИХ В АУДІОКОНТЕЙНЕРАХ

Методичні рекомендації
до лабораторних робіт з дисципліни «Стеганографія»

Харків – 2019

УДК 004.415.24 (075.8)
П 75

Рецензенти:

В. А. Краснобаєв – доктор технічних наук, професор кафедри електроніки і управляючих систем Харківського національного університету імені В. Н. Каразіна;

О. Г. Толстолузька – доктор технічних наук, старший науковий співробітник, професор кафедри теоретичної та прикладної системотехніки Харківського національного університету імені В. Н. Каразіна.

Затверджено до друку рішенням Науково-методичної ради Харківського національного університету імені В. Н. Каразіна (протокол № 1 від 30.10.2019 р.)

Приховування даних в аудіоконтейнерах : методичні рекомендації до лабораторних П 75 робіт з дисципліни «Стеганографія» / уклад. О. О. Кузнецов, М. О. Полуяненко, Т. Ю. Кузнецова. – Харків : ХНУ імені В. Н. Каразіна, 2019. – 52 с.

Методичні рекомендації розроблено для студентів факультету комп'ютерних наук за спеціальністю «Кібербезпека». Матеріали методичних рекомендацій мають допомогти студентам усвідомити специфіку безпеки інформаційних і комунікаційних систем та особливості професійної наукової діяльності у галузі захисту інформації. Передбачається, що в результаті навчання студенти оволодіють початковими навичками роботи з дисципліни «Стеганографія», сформують уявлення про використання методів та принципів приховування даних, набудуть практичних вмінь та навичок щодо розробки стеганографічних систем.

УДК 004.415.24 (075.8)

ЗМІСТ

1. МЕТА ТА ЗАВДАННЯ ЛАБОРАТОРНОЇ РОБОТИ

Мета роботи: закріпити теоретичні знання з теми «Стеганографічні методи приховування даних в аудіофайлі», придбати практичні вміння та навички з розробки стеганографічних систем, дослідити властивості стеганографічних методів, заснованих на низькорівневих властивостях слухової системи людини (ССЛ). Допускається застосування інших середовищ / мов програмування.

Лабораторна робота виконується в середовищі символьної математики MathCAD версії 12 або вище.

Завдання лабораторної роботи

Завдання 1. Реалізація алгоритмів вбудовування та вилучення повідомлень методом заміни найменш значущих біт даних:

– реалізувати в середовищі символьної математики MathCad (або іншому середовищі/мові програмування) алгоритми вбудовування і вилучення повідомлень у просторовій області аудіоконтейнерів за допомогою модифікації найменш значущого біта даних (методом LSB);

– застосовуючи розроблену програмну реалізацію виконати стеганографічне кодування інформаційного повідомлення, тобто сформувати заповнений контейнер (стеганограму);

– виконати порівняння «на слух» порожнього і заповненого контейнера і переконатися, що немає помітних похибок;

– виконати вилучення вбудованого повідомлення, переконатися в його автентичності;

– отримати заповнені контейнери від інших груп розробників і переконатися, що немає помітних похибок;

– вилучити повідомлення з отриманих стеганоконтейнерів інших груп розробників і переконатися в їх справжності.

Завдання 2. Експериментальні дослідження чутливості слухової системи людини до змін гучності аудіоконтейнерів:

– виконати вбудовування даних в різні за значущістю біти аудіоконтейнера, починаючи з найменш значущого;

– експериментально встановити, модифікація яких біт аудіоконтейнера не призводить до помітних похибок;

– розрахувати за отриманими емпіричними даними слуховий поріг чутливості до змін гучності аудіоконтейнерів.

Завдання 3. Реалізація алгоритмів вбудовування та вилучення повідомлень методом кодування початкових фаз:

– реалізувати в середовищі символьної математики MathCad (або іншому середовищі/мові програмування) алгоритми вбудовування і вилучення повідомлень в частотній області аудіоконтейнерів за допомогою кодування початкових фаз;

– застосовуючи розроблену програмну реалізацію, виконати вбудовування та витяг інформаційного повідомлення;

– виконати порівняння «на слух» порожнього і заповненого контейнера, зробити висновки.

Завдання 4. Реалізація алгоритмів вбудовування та вилучення повідомлень методом кодування ехо-сигналів:

– реалізувати в середовищі символьної математики MathCad (або іншому середовищі/мові програмування) алгоритми вбудовування і вилучення повідомлень у просторовій області аудіоконтейнерів за допомогою кодування ехо-сигналів;

– застосовуючи розроблену програмну реалізацію, виконати вбудовування та вилучення інформаційного повідомлення;

– виконати порівняння «на слух» порожнього і заповненого контейнера, зробити висновки.

2. ЛІТЕРАТУРА

Для підготовки до виконання лабораторної роботи рекомендуються сновні та додаткові джерела інформації [1–8].

Основні джерела інформації

1. И. П. Голямина. Звук // Физическая энциклопедия: в 5 т. / гл. ред. А. М. Прохоров. – М. : Советская энциклопедия (тт. 1–2); Большая Российская энциклопедия (тт. 3–5), 1988–1999.

2. W. Bender, D. Gruhl, N. Morimoto, A. Lu. Techniques for Data Hiding. IBM Systems Journal, 35 (3&4) pp. 313–336, 1996.

3. Конахович Г. Ф., Пузиренко О. Ю. Компьютерная стеганография. – К. : «МК-Пресс», 2006 – 288 с.

Додаткові джерела інформації

4. WAVE PCM soundfile format. 2003-01-20. Archived from the original on 2009-08-27. [Електронний ресурс]. – Режим доступу : https://web.archive.org/web/20090827003349/http://ccrma.stanford.edu/courses/422/projects/WaveFormat/.

5. Формат WAVE файлов. [Електронний ресурс]. – Режим доступу : http://alexei-s1.narod.ru/WAVE.htm.

6. Структура WAV файла. [Електронний ресурс]. – Режим доступу : (http://audiocoding.ru/article/2008/05/22/wav-file-structure.html.

7. Wave File Format – формат звукового файла WAV. [Електронний ресурс]. – Режим доступу : http://microsin.net/programming/pc/wav-format.html.

8. RIFF WAVE (WAV) file format. [Електронний ресурс]. – Режим доступу : http://www.textfiles.com/programming/FORMATS/riffform.txt.

3. МЕТОДИЧНІ ВКАЗІВКИ ЩОДО ОРГАНІЗАЦІЇ САМОСТІЙНОЇ РОБОТИ

1. Вивчити теоретичний матеріал лекції «Стеганографічні методи приховування даних в аудіофайлі».

2. Вивчити матеріал основного джерела інформації [3] з приховування даних в аудіоконтейнерах (ст. 196–230).

3. Вивчити матеріал додаткових джерел інформації [4–8]:

– WAVE PCM soundfile format [4];
– Формат WAVE файлів [5];
– Структура WAV файлу [6];
– Wave File Format – формат звукового файлу WAV [7];
– RIFF WAVE (WAV) file format [8].

4. Вивчити основні команди в середовищі символьної математики MathCAD щодо роботи з аудіофайлами [3, додаток В].

5. Підготувати відповіді на контрольні питання.

6. Підготувати бланк звіту з лабораторної роботи.

Допуск до виконання лабораторної роботи здійснюється за результатами письмового опитування.

4. ЗАГАЛЬНОТЕОРЕТИЧНІ ПОЛОЖЕННЯ ЗА ТЕМОЮ ЛАБОРАТОРНОЇ РОБОТИ

Властивості слухової системи людини, що використовуються в стеганографії

У цьому розділі подано короткі відомості з фізичних властивостей звуку та фізіологічних властивостей слухової системи людини (ССЛ), запозичені із основних джерел інформації [1, 2].

Звук – фізичне явище, що являє собою поширення у вигляді пружних хвиль у фізичному або газоподібному світі. У вузькому розумінні, під звуком розуміють саме ці коливання, що розглядаються у зв’язку з тим, як вони сприймаються органами чуття тварин і людини [1].

Як і будь-яка хвиля звук характеризується амплітудою. Звичайна людина здатна чути звукові коливання в діапазоні частот від 16–20 Гц до 15–20 кГц [1]. Звук нижче діапазону чутності людини називають *інфразвуком*; до 1 ГГц – *ультразвуком*, від 1 ГГц – *гіперзвуком*. Гучність звуку залежить від ефективного звукового тиску, частоти і форми коливань, а висота звуку – не тільки від частоти, а й від обсягу звукового тиску.

Методи вбудовування даних в аудіоконтейнери засновані на використанні різних низькорівневих властивостей ССЛ, що характеризують фізіологічні особливості і різні обмеження в сприйнятті звуку людиною.

Умовно виділяють такі *властивості*.

1. Слабка чутливість ССЛ до незначної зміни гучності аудіосигналу.

Ця властивість виражена у різних людей по-різному. Проте відомо [2], що ця властивість у більшості людей виражена слабко: ССЛ працює у надширокому динамічному діапазоні: слуховий апарат сприймає більш ніж мільярд до одного в діапазоні потужності і більш ніж тисяча до одного в частотному діапазоні. Крім цього високою є і чутливість до адитивного флуктуаційного (білого) шуму. Відхилення в звуковому файлі можуть бути виявлені аж до однієї десятимільйонної (на 70 дБ нижче за рівень зовнішніх шумів). Природно, ці значення залежать від властивості звукових контейнерів і від особливостей ССЛ конкретної людини.

2. Ефект маскування.

Цей ефект можна спостерігати на лекціях у разі тихого спілкування студентів між собою. Якщо частотний діапазон їхніх голосів збігається з діапазоном лектора, голоси студентів «маскуються» під гучний голос викладача й можуть залишатися непоміченими. Навпаки, якщо частотний діапазон студентських голосів відмінний від діапазону лектора (наприклад, голоси дівчат-студентів на фоні лектора-чоловіка), тоді «маскування» не вдається, всім слухачам заважають порушники навчальної дисципліни.

Цей ефект пояснюється такими фактами [2]: хоча ССЛ і має широкий динамічний діапазон, вона характеризується досить малим різницевим діапазоном. Як наслідок, голосні звуки сприяють маскуванню тихих звуків.

Крім того слід зазначити, що існують деякі види спотворень, викликаних навколишнім середовищем, які настільки звичні для слухача, що в більшості випадків ним ігноруються.

3. Частотна чутливість.

Природа наділила людину дивовижними органами чуття. Ми чуємо гуркіт струму в високовольтних проводах (з частотою 50 Гц) і писк комара (від 12000 до 16000 Гц). Однак найбільшу чутливість ССЛ має в «середніх» частотах: від 300 Гц до 3400 Гц (це частотний діапазон стандартного каналу тональної частоти, що використовується в телефонії). На низьких (менше 300 Гц) і високих (більше 3400 Гц) чутливість ССЛ знижується. Наприклад, ультразвук, за допомогою якого віддають команди дресирувальники дельфінів, ССЛ практично не сприймає.

4. Несприйнятливість ССЛ до зміни абсолютної фази аудіосигналу.

Якщо записати звук, що видає камертон, а потім відтворити його кілька разів у довільний момент часу, отримані звуки будуть сприйматися ССЛ однаково. Це виглядає для нас природним, проте один з параметрів (часова затримка) аудіосигналу був змінений.

Гармонійне коливання, що породжується камертоном, опишемо формулою

$$A(t,\varphi_0) = a\sin(wt+\varphi_0),$$

де a – амплітуда гармонічного (в цьому разі синусоїдального) коливання, $w = 2\pi f$ – 2кругова (циклічна) частота, f – частота гармонійного коливання, φ_0 – початкова фаза.

Зміна часового зсуву (що еквівалентно зміні початкової фази φ_0) не сприймається ССЛ. Для людини звукові хвилі $A(t,\varphi_0) = a\sin(wt+\varphi_0)$ і $A(t,\varphi'_0) = a\sin(wt+\varphi'_0)$ за $\varphi_0 \neq \varphi'_0$ звучать однаково.

На рис. 1 наведені міркування наочно продемонстровані для випадку гармонійного коливання з одиничною амплітудою, частотою 700 Гц і різними початковими фазами. Зрозуміло, що зміна початкової фази еквівалентна часовій затримці аудіосигналу, ССЛ зміну фази не сприймає.

У той же час слід зазначити, що відносну зміну початкової фази ССЛ сприймає. Якщо в процесі відтворення аудіосигналу стрибкоподібно змінити початкову фазу, то буде чутно клацання, тобто ССЛ сприйме таку зміну за спотворення аудіосигналу.

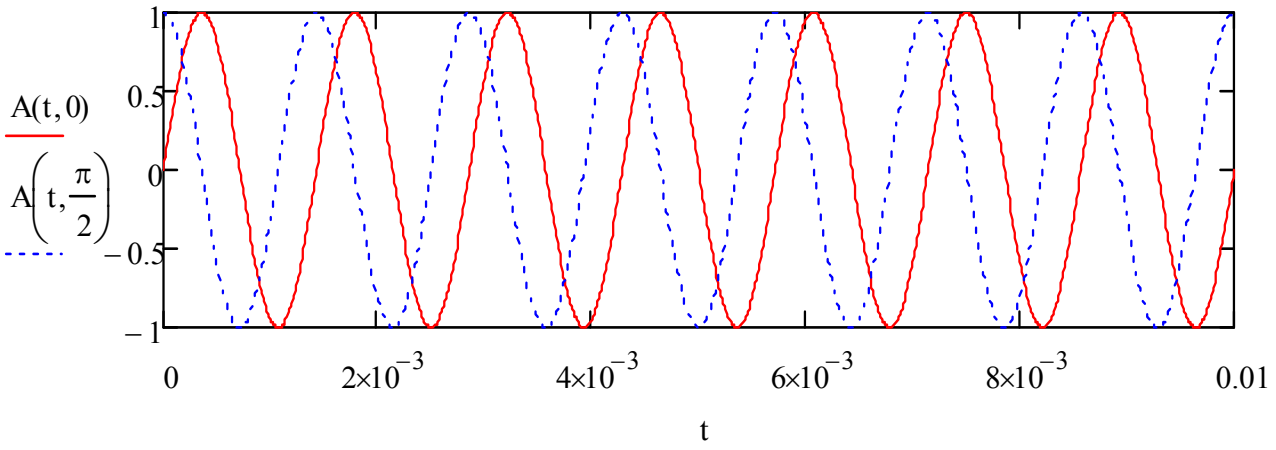


Рис. 1. Графік функції $A(t)$, *якщо* $a = 1$, $f = 700$ *на інтервалі 0 .. 0,01 для випадків* $\varphi_0 = 0$ *и* $\varphi'_0 = \dfrac{\pi}{2}$

5. Слабка чутливість ССЛ до незначної зміни ехо-сигналів.

Під відлунням (від нім. echo з лат. ēchō від грец. ἠχώ – відгомін) розуміють фізичне явище, що полягає в сприйнятті спостерігачем відбитої від перешкод хвилі (електромагнітної, звукової тощо).

Під звуковим відлунням розуміють відбитий звук.

Відлуння помічають, якщо чують безпосередній звук від джерела, коли в одній точці простору можна кілька разів почути звук з одного джерела, що прийшов по прямій колії і був відбитий (можливо кілька разів) від навколишніх предметів. Оскільки під час відображення звукова хвиля втрачає енергію, то звукова хвиля від більш сильного джерела звуку зможе

відбитися від поверхонь (наприклад, будинків, що стоять один навпроти одного, або стін), багато разів проходячи через одну точку, що може викликати багаторазове відлуння (таке відлуння можна спостерігати від грому).

Відлуння обумовлено тим, що хвилі можуть відбиватися твердими поверхнями. Це пов'язано з динамічною картиною розрідження і ущільнення повітря поблизу поверхні, що відбиває. Якщо джерело звуку розташоване неподалік від такої поверхні, оберненої до нього під прямим кутом або під кутом, близьким до прямого, то звук, відбившись від такої поверхні, як хвиля відбивається від берега, повертається до джерела.

Завдяки відлунню людина, що говорить може разом з іншими звуками чути свою власну мову, що ніби затрималася на деякий час. Якщо джерело звуку знаходиться на достатній відстані від поверхні, що відбиває, а крім джерела звуку поблизу немає ніяких додаткових звукових джерел, то відлуння стає найбільш виразним.

Відлуння стає помітним на слух, якщо інтервал між прямою і відбитою звуковою хвилею складає 50–60 мс, що відповідає 15–20 м, які звукова хвиля проходить від джерела і назад за нормальних умов.

До особливостей ССЛ належить слабка чутливість до незначної зміни відлуння. Наприклад, якщо відлуння сформовано хвилею, що відбита від перешкоди на відстані 0,1–1 м (затримка між звуком і відлунням близько 1 мс), таке відлуння практично не сприймається ССЛ, воно зливається з вихідною звуковою хвилею. Незначна зміна подібного відлуння теж не буде сприйнята ССЛ.

Формат аудіофайлів WAV

У цьому розділі подано короткі відомості формату WAV, запозичені із додаткових джерел інформації [4–8].

WAVE або **WAV** є короткою формою Wave Audio File Format (що рідше мало назву Аудіо для Windows). Цей формат є стандартом для зберігання цифрового аудіопотоку для комп'ютерних програм. Він є сферою застосування формату RIFF для зберігання аудіо в «ланцюжках».

Формат RIFF використовує порядок байтів little-endian (молодший байт йде першим). Для машин з форматом даних big-endian пропонувався формат RIFX, однак через істотно меншу в побутовому секторі популярність комп'ютерів з таким форматом даних, RIFX не поширився, нині формат RIFF відтворюється і на машинах з big-endian порядком байтів.

Основною концепцією RIFF-формату є так званий «chunk», тобто секція даних з заголовком і сигнатурою, що вказує на його вміст.

Формат секцій подано в таблиці 1.

Таблиця 1

Формат секції даних RIFF

Зсув	Тип поля	Ім'я поля	Коментар до поля
0 x 00	FOURCC	ckID	сигнатура
0 x 04	DWORD	ckSize	розмір даних
0 x 08	BYTE[]	ckData	дані

Якщо секція містить непарну кількість байтів, то після неї додається один байт. У такий спосіб всі секції завжди вирівняні на 2 байти, тобто загальний розмір секції завжди кратний 2.

Поле FOURCC (від англ. Four Character Code) – послідовність з чотирьох символів, що використовується для ідентифікації будь-яких даних. Відносно RIFF – це ckID (ідентифікатори секцій «chunk») і типи форм. Відносно відеопотоку поле FOURCC зазвичай використовується для ідентифікації кодека (наприклад, XVID, DIV3, MP43). Поле FOURCC займає 4 байти; це розмір 32-бітного числа, отже, іноді поле FOURCC записують у вигляді числа (уявлення числа в ASCII little-endian).

Заголовки WAV-файлу використовують стандартний формат RIFF. Перші 8 байт файлу – стандартний заголовок секції RIFF, що має ID секції «RIFF» і розмір секції, рівний розміру файлу мінус 8 байт, які використовуються для RIFF-заголовка. Перші 4 байти даних в секції RIFF визначають тип ресурсу, який можна знайти в секції. WAV-файли завжди використовують тип ресурсу WAVE. Після типу ресурсу (ID WAVE) розташовані ті секції звукового файлу, що визначають аудіосигнал.

Існує досить багато типів секцій, заданих для файлів WAV, але більшість WAV-файлів містять тільки дві з них – секцію формату **(«fmt»)** і секцію даних (**«data»**). Це саме ті секції, що необхідні для опису формату вибірки аудіо і для зберігання звукової інформації. Хоча офіційна специфікація не вимагає жорсткого порядку проходження секцій, провідним досвідом буде розміщення секції формату перед секцією даних. Багато програм очікують саме такий порядок секцій, і він найбільш розумний для передачі звукової інформації через повільні, послідовні джерела на зразок інтернет. Інакше якщо формат прийде після даних, то перед стартом відтворення необхідно запам'ятати всі звукові дані, і тільки після отримання формату запускати відтворення. Всі секції формату RIFF і відповідно секції WAVE зберігаються в форматі як в таблиці 2.

Таблиця 2

Значення полів секції RIFF в заголовках WAV-файлу

Зсув	Розмір	Опис	Значення
0 x 00	4	Chunk ID	RIFF (0 x 52494646)
0 x 04	4	Chunk Data Size	(file size) – 8
0 x 08	4	RIFF Type	WAVE (0 x 57415645)
0 x 10	WAVE chunks (секції WAV-файлу)		

Секція формату – «fmt»

Секція формату містить інформацію про те, як збережені звукові дані і як вони повинні відтворюватися. Інформація містить тип використовуваної компресії, кількість каналів, швидкість видачі вибірок (sample rate), кількість біт у вибірці (bits per sample) та інші атрибути.

Таблиця 3

Значення полів секції формату

Зсув	**Розмір**	**Опис**	**Значення**
0 x 00	4	Chunk ID	«fmt» (0 x 666D7420)
0 x 04	4	Chunk Data Size	16 + extra format bytes
0 x 08	2	Compression code	1– 65,535
0 x 0a	2	Number of channels	1– 65,535
0 x 0c	4	Sample rate	1 – 0 x FFFFFFFF
0 x 10	4	Average bytes per second	1 – 0 x FFFFFFFF
0 x 14	2	Block align	1 – 65,535
0 x 16	2	Significant bits per sample	2 – 65,535
0 x 18	2	Extra format bytes	0 – 65,535
0 x 1a	Додаткові дані формату (Extra format bytes)		

Ідентифікатор секції (Chunk ID) і обсяг даних (Data Size). Ідентифікатор секції завжди «fmt» (0 x 666D7420) і обсяг даних дорівнює розміру стандартного формату WAV (16 байт) плюс розмір всіх додаткових байт формату, що є необхідним для підтримки специфічних форматів звуку, якщо він не містить нестиснених даних PCM. Зверніть увагу, що ідентифікатор секції «fmt» закінчується символом пробілу (0 x 20).

Код стиснення (Compression Code). Номінація даних формату вказує на тип стиснення, що використовується для даних звуку. У таблиці 4 міститься список кодів стиснення, що використовуються на сьогодні.

Таблиця 4

Список кодів стиснення

Код	**Опис**
0 (0 x 0000)	Unknown
1 (0 x 0001)	PCM/uncompressed
2 (0 x 0002)	Microsoft ADPCM
6 (0 x 0006)	ITU G.711 a-law
7 (0 x 0007)	ITU G.711 Âμ-law
17 (0 x 0011)	IMA ADPCM
20 (0 x 0016)	ITU G.723 ADPCM (Yamaha)
49 (0 x 0031)	GSM 6.10
64 (0 x 0040)	ITU G.721 ADPCM
80 (0 x 0050)	MPEG
65,536 (0 x FFFF)	Experimental

Кількість каналів (Number of Channels). Кількість каналів вказує на число окремих аудіосигналів, закодованих в секції даних звуку (wave data chunk). Значення 1 означає моно сигнал, 2 означає стерео тощо.

Швидкість вибірок (Sample Rate). Кількість вибірок аудіосигналу, що припадають на секунду. На цю величину не впливає кількість каналів.

Середня кількість байт в секунду (Average Bytes Per Second).

Величина, що показує скільки байт за секунду даних має бути пропущена через цифроаналоговий перетворювач (D/A converter, DAC) під час відтворення файлу. Ця інформація корисна, щоб визначити – чи можуть дані надходити від джерела з потрібною швидкістю, щоб не відставати від відтворення. Ця величина просто обчислюється за формулою: AvgBytesPerSec = SampleRate * BlockAlign.

Вирівнювання блоку (Block Align). Кількість байт на одну вибірку. Ця величина може бути обчислена за формулою: BlockAlign = = SignificantBitsPerSample/8 * NumChannels.

Кількість використовуваних біт на вибірку (Significant Bits Per Sample). Величина вказує кількість біт, що формують кожну вибірку сигналу. Зазвичай ця величина 8, 16, 24 або 32. Якщо кількість біт не вирівняна за байтом (не ділиться без остачі на 8), кількість використовуваних байт на вибірку округляється вгору до мінімуму байт. Невикористані біти встановлюються в 0 й ігноруються. Такі формати (з кількістю біт на вибірку, що некратні 8) зустрічаються зрідка.

Додаткові дані формату (Extra Format Bytes). Величина вказує, скільки далі є додаткових даних, що описують формат. Її немає, якщо код стиснення 1 (uncompressed PCM file), але може мати будь-яку іншу величину для інших типів стиснення, що залежить від кількості необхідних для декодування даних. Якщо величина не вирівняна на слово (не ділиться без остачі на 2), повинен бути доданий додатковий байт в кінець даних, але величина має залишатися невирівняною.

Секція даних – «data»

Секція даних WAVE (Wave Data Chunk) містить дані цифрових вибірок аудіосигналу, які можна декодувати з використанням формату і методу компресії, зазначених в секції формату WAVE (Wave Format Chunk). Якщо код компресії 1 (нестислий PCM, Pulse Code Modulation), то дані подаються у вигляді неперетворених (raw) величин вибірок. Ця робота описує збережені стиснені дані PCM, не вдаючись утім у подробиці багатьох використовуваних форматів з компресією.

WAV-файли зазвичай містять тільки одну секцію даних (див. таблицю 5), але секцій може бути кілька, якщо вони містяться в секції списку WAVE (Wave List Chunk «wavl»).

Таблиця 5

Формат секції даних «data»

Зсув	Довжина	Тип	Опис	Значення
0 x 00	4	Char [4]	chunk ID	«data» (0 x 64617461)
0 x 04	4	dword	chunk size	залежить від кількості вибірок і компресії
0 x 08	дані вибірок (sample data)			

Аудіовибірки багатоканального цифрового аудіо зберігаються як дані, що чергуються (interlaced), які просто означають послідовні аудіовибірки декількох каналів (таких як стерео і канали оточення surround). Вибірки каналів збережені послідовно один за одним, перед тим як відбудеться перехід до наступного часу вибірки. Це зроблено з метою можливості послідовного програвання файлу навіть тоді, коли ще не весь файл прочитаний цілком. Це зручно, коли програється великий файл з диска (що не може бути розміщений цілком в пам'яті) або файл передається в послідовному потоці даних через мережеве з'єднання (наприклад, інтернет). Значення в діаграмі (таблиця 6) були б збережені в WAV-файлі в тому порядку, в якому вони перераховані в стовпці значень (від початку до кінця).

Таблиця 6

Вибірки стерео WAVE, що чергуються

Час	Канал	Значення
0	1 (лівий)	0 x 0053
	2 (правий)	0 x 0024
1	1 (лівий)	0 x 0057
	2 (правий)	0 x 0029
2	1 (лівий)	0 x 0063
	2 (правий)	0 x 003C

Єдиний момент, що стосується даних вибірок і може викликати певне збентеження – ситуація, коли вибірки представлені 8 бітами, вони визначені як значення без знаку (unsigned). Всі інші бітові розміри вказуються як величини зі знаком (signed). Наприклад, вибірка 16 біт може мати значення в діапазоні від –32768 до +32767, де середня точка (напруга сигналу дорівнює 0) відповідає значенню 0.

Як вже було зазначено раніше, всі секції RIFF (включно із секцією WAVE «data») повинні бути вирівняні за розміром на слово (2 байти). Якщо дані вибірок містяться в непарній кількості байт, в кінець даних вибірок необхідно додати вирівнюючий нульовий байт. У заголовку секції «data» розмір не повинен враховувати цей вирівнюючий байт.

Кодування найменш значущих біт (часова область)

У цьому розділі подано короткі відомості стосовно стеганографічних методів кодування найменш значущих біт в аудіоконтейнерах, запозичені із основних джерел інформації [2, 3].

Кодування молодших розрядів є найпростішим способом долучити конфіденційні дані до інших структур даних. Використовуючи звуковий сигнал, шляхом заміни найменш значущих біт (НЗБ) кожної точки здійснення вибірки, що подається двійковою послідовністю, можна приховати значний обсяг інформації.

Теоретично пропускна здатність стеганоканалу дорівнює 1 Кб/сек на 1 кГц в каналі без перешкод, а бітова швидкість передачі даних складе 8 Кб/сек в послідовності, що відцифрована з частотою 8 кГц, і 44 Кб/сек в послідовності з частотою дискретизації 44 кГц. Платою за високу пропускну здатність каналу є відчутний на слух низькочастотний шум. Чутність цього шуму безпосередньо залежить від вмісту сигналу-контейнера. Наприклад, шум глядачів під час ефіру спортивного змагання достатньо маскував би шум найменш значущихся біт, модифікованих кодуванням. Однак вказаний шум буде відчутним на слух під час використання в якості контейнера аудіозапису гри струнного квартету. Для компенсації внесених спотворень доцільним буде використання адаптивної атенюації даних.

Головний недолік методу кодування НЗБ, як і у випадку з графічним контейнером, – це його слабка стійкість до сторонніх впливів. Вбудована інформація може бути зруйнована через наявність шумів в каналі, в результаті передискретизації вибірки за винятком випадків, коли така інформація була побудована із долученням надмірності. Однак остання, забезпечуючи прийнятну стійкість до перешкод, призводить до зменшення швидкості передачі даних, найчастіше на один/два порядки. На практиці метод корисний тільки в замкнутих, повністю цифрових середовищах, які не потребують додаткового перетворення.

Для отримання інформації щодо WAV-файлу використовується вбудована функція MathCAD **GETWAVINFO** («файл»), де під аргументом «файл» мається на увазі текстовий рядок, що містить ім’я файлу (або ж повний шлях та ім’я файлу), що передбачається використовувати в якості контейнера.

Зазначена функція повертає чотирьохелементний вектор з інформацією про файл, що виступив її аргументом. Перший елемент вектора характеризує кількість каналів; другий – частоту дискретизації (в герцах); третій – кількість біт, якими кодується один відлік (визначальна кількість рівнів квантування); четвертий – середню кількість біт на секунду, яку повинен обробляти аудіопрогравач, щоб відтворювати цей файл в реальному часі.

Нижче наведемо два можливі варіанти використання цієї функції

$$\text{GETWAVINFO("C.wav")} = \begin{pmatrix} 2 \\ 22050 \\ 16 \\ 88200 \end{pmatrix}; \quad \textit{або} \quad \begin{pmatrix} N_K \\ f_д \\ Q \\ B \end{pmatrix} \equiv \text{GETWAVINFO("C.wav")}$$

$N_K = 2$ канали; $f_Д = 22050$ Гц; $Q = 16$ біт; $B = 88200$ біт/с.

Користуючись отриманою інформацією, знайдемо часовий вектор, що відповідає амплітудам аудіосигналу в окремому відліку дискретизації. Дані амплітуди можна обчислити за допомогою функції **READWAV** («файл»), що повертає масив, кожен стовпець якого є окремим каналом (так, наприклад, для моносигналу масив буде містити тільки 1 стовпець, для стерео – 2 тощо), а кожен рядок масиву відповідає моменту часу, що визначається номером відліку і частотою дискретизації сигналу. Окремий елемент масиву, залежно від кількості біт кодування Q, може набувати значення або від 0 до $2^8 - 1 = 255$ (при Q = 1 ... 8), або ж від $-2^{16} - 1 = -32768$ до $2^{16} - 1 - -1 = 32767$ (за Q = 9 ... 16).

Нехай **C := READWAV («C.wav»)**. Фрагмент імпортованого звуку (коди відліків від 1000-го по 1010-й в десятковому і двійковому вигляді) зображений на рис. 2.

i =	$C_{i,1}$ =	$C_{i,2}$ =	$C_{i,1}$ =	$C_{i,2}$ =
1000	5055	6154	1001110111111b	1100000001010b
1001	5213	6071	1010001011101b	1011110110111b
1002	4947	6125	1001101010011b	1011111101101b
1003	4131	5371	1000000100011b	111111111000b
1004	3131	4088	110000111011b	110011001010b
1005	2446	3274	100110001110b	100110110101b
1006	1842	2485	11100110010b	11001011111b
1007	1174	1631	10010010110b	1111011110b
1008	304	494	100110000b	100110000b
1009	–953	–1127	–1110110010b	–10001100111b
1010	–2466	–3080	–100110100010b	–110000001000b

Рис. 2. Фрагмент імпортованого аудіофайлу у вигляді масиву квантованих амплітуд

Володіючи інформацією про часові координати кожного з відліків і знаючи відповідні цим відлікам амплітуди (у квантованому вигляді), можна відтворити «осцилограму» імпортованого аудіофайлу (рис. 3).

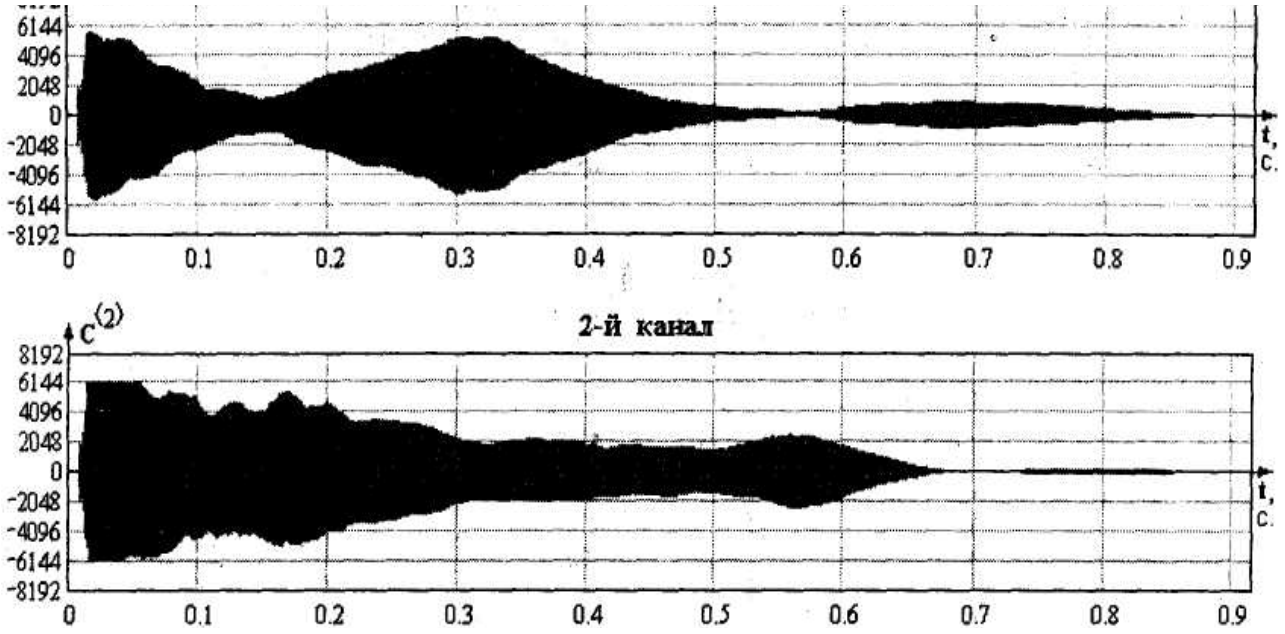


Рис. 3. Часові діаграми каналів сигналу «C.wav»

Методи кодування НЗБ засновано на вбудовувану інформаційних даних побітно в НЗБ окремих дискретних звітів. Очевидно, що за такого запису пропускна здатність буде визначатися кількістю вбудованих біт на один дискретний відлік. Водночас використовується перша низькорівнева властивість ССЛ.

Для того щоб записати масив S в файл скористаємося вбудованою функцією WRITEWAV («файл», частота дискретизації, кількість біт квантування)

$$\mathrm{WRITENWAV}(\text{"S\_LSB.wav"}, f_{\text{ä}}, Q) := S.$$

Можливі також модифікації методу НЗБ: метод псевдовипадкової перестановки, псевдовипадкового інтервалу, блокового вбудовування тощо.

Метод фазового кодування (частотна область)

У цьому розділі подано короткі відомості стосовно стеганографічного методу фазового кодування, запозичені із основних джерел інформації [2, 3].

Основна ідея методу фазового кодування полягає в заміні фази вихідного звукового сегмента на опорну фазу, характер зміни якої відображає собою дані, що необхідно приховати. Для того, щоб зберегти різницеву фазу між сегментами фази останніх відповідним чином узгоджуються. При цьому використовується низькорівнева властивість ССЛ, що полягає в несприйнятливості органами слуху людини абсолютної фази аудіосигналу.

Фазове кодування, коли воно може бути використано, є одним з найбільш ефективних методів за критерієм співвідношення сигнал/шум, що сприймається ССЛ. Істотна зміна співвідношення фаз між кожними частотними складовими призводить до значного розсіювання фази. Проте доти, доки модифікація фази досить мала, можна досягнути приховування, не відчутного на слух. Зрозуміло, модифікація вважається малою щодо конкретного спостерігача, оскільки фахівці зі спектрального аналізу здатні виявити ті зміни, які непрофесіоналу можуть здатися незначними.

Процедура фазового кодування полягає в такому:

1. Звукова послідовність $S[i], (1 \le i \le I)$ розбивається на серію N коротких сегментів (блоків) $S_n[i], (1 \le n \le I)$;

2. Для n-ого сегменту сигналу $S_n[i]$ застосовується K-точкове дискретне перетворення Фурьє (ДПФ), де K = I/N, і створюються масиви фаз $\varphi_n(\omega_k)$ і амплітуд $A_n(\omega_k)$ для $1 \le k \le K$;

3. Запам'ятовується різниця фаз між кожними сусідніми сегментами для $1 < n \le N$

$$\Delta\varphi_n(\omega_k) = \varphi_n(\omega_k) - \varphi_{n-1}(\omega_k); \quad \Delta\varphi_1(\omega_k) = 0;$$

4. Двійкова послідовність даних являє собою $\varphi_{data} = \pi/2$ або $\varphi_{data} = -\pi/2$, відображаючи, відповідно, «1» або «0»: $\varphi_1'(\omega_k) = \varphi_{data}$;

5. З урахуванням різниці фаз відтворюється новий масив фаз для n > 1

$$\left\| \begin{array}{l} \varphi_1'(\omega_k) = \varphi_{data} \\ \varphi_1'(\omega_k) = \varphi_1'(\omega_k) + \Delta\varphi_2(\omega_k) \\ \quad \ldots \\ \varphi_1'(\omega_k) = \varphi_{n-1}'(\omega_k) + \Delta\varphi_2(\omega_k) \\ \quad \ldots \\ \varphi_N'(\omega_k) = \varphi_{N-1}'(\omega_k) + \Delta\varphi_N(\omega_k) \end{array} \right\|;$$

6. Відновлення звукового сигналу здійснюється шляхом застосування операції зворотного ДПФ до вихідної матриці амплітуд і модифікованої матриці фаз;

Перед процесом розшифрування повинна бути проведена синхронізація послідовності. Стороні, що приймає повинні бути відомі довжина сегмента, точки ДПФ і інтервал даних. Значення основної фази першого сегмента визначається як «0» або «1», що являють собою закодовану двійкову послідовність.

Оскільки фаза $\varphi_1'(\omega_k)$ є модифікованою, відповідним чином будуть змінені і абсолютні фази наступних сегментів. Однак відносна різниця фаз

між кожними суміжними сегментами буде збережена. Саме до цієї відносної різниці в фазі і виявляється найбільша чутливість людського слуху.

Розкид фаз – спотворення, викликане порушенням кореляції фаз між кожною з частотних складових. Зменшення розкиду фаз обмежує швидкість передачі даних за фазового кодування. Однією з причин розкиду фаз можна назвати заміщення фази $\varphi_l'(\omega_k)$ двійковим кодом. Для зменшення спотворень значення модифікованої фази має бути близьким до її первинного значення. А для того, щоб знизити чутливість вбудованих даних до шуму, повинна збільшуватися різниця між структурами модифікованої фази. Для цього, наприклад, біту «0» може відповідати значення $-\pi/2$, а біту «1» – значення $+\pi/2$.

Ще одним джерелом спотворення є швидкість зміни модифікованої фази. Якщо спотворення застосовано до кожного елемента дискретизації ДПФ, це з великою ймовірністю зруйнує зв'язок між фазами сусідніх частотних складових, що в результаті призведе до накладення фонового биття. Шляхом більш повільної зміни фази і узгодження переходів між змінами фази досягається істотне зниження відчутних на слух спотворень.

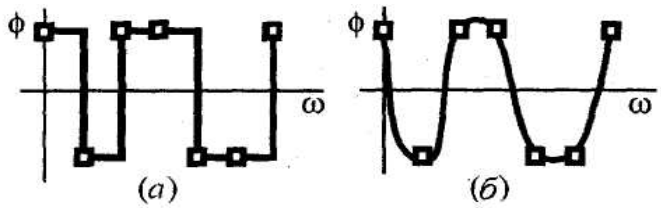


Рис. 4. Порівняння різких і згладжених переходів фази

На рис. 4 зображені різкі переходи фази порівняно зі згладженими. Круті фронти фазових переходів викликають значні спотворення контейнера (а); рівень спотворення зменшується, якщо фронти були попередньо згладжені (б).

Слід зазначити, що в обох випадках, зображених на рис. 4, інформаційні точки відповідають одним і тим самим значенням частоти. Така плавна зміна характеризується недоліком, що полягає у скороченні смуги пропускання, оскільки для того, щоб зробити можливим плавний перехід необхідно зарезервувати достатньо місця між кожною інформаційною точкою.

Приховування даних з використанням ехо-сигналу

У цьому розділі дано короткі відомості стосовно стеганографічного методу приховування даних з використанням ехо-сигналу, запозичені із основних джерел інформації [2, 3].

Цей метод означає вбудовування даних в аудіосигнал-контейнер шляхом долучення до ехо-сигналу [2, 3]. Дані приховуються зміною трьох параметрів ехо-сигналу: початкової амплітуди, швидкості загасання [(початкова амплітуда – загасання)/δ] і зсуву (рис. 5). При цьому використовується п'ята низькорівнева властивість ССЛ.

Коли зрушення (затримка) між первинним і ехо-сигналом зменшується, починаючи з деякого значення затримки, ССЛ стає нездатною виявити різницю між двома сигналами, а ехо-сигнал сприймається тільки як додатковий резонанс. Згадане значення важко визначити точно, оскільки воно залежить від якості первинного звукозапису, типу звуку, для якого формується ехо-сигнал, і в кінцевому підсумку – від слухача.

В цілому автори [2] дійшли висновку, що для більшості звуків і більшості слухачів змішування відбувається під час затримки, що відповідає приблизно одній мілісекунді.

Стеганокодер використовує два значення часу затримки: один для подання двійкового нуля («зсув» на рис. 5), а інше – для подання двійкової одиниці («зсув + δ »). Обидва значення часу затримки менші за той граничний час, за який ССЛ здатна розпізнати ехо-сигнал. Крім зменшення часу затримки для забезпечення невідчутності, також можна встановити рівні початкової амплітуди і часу загасання, які б не перевищували поріг чутливості ССЛ.

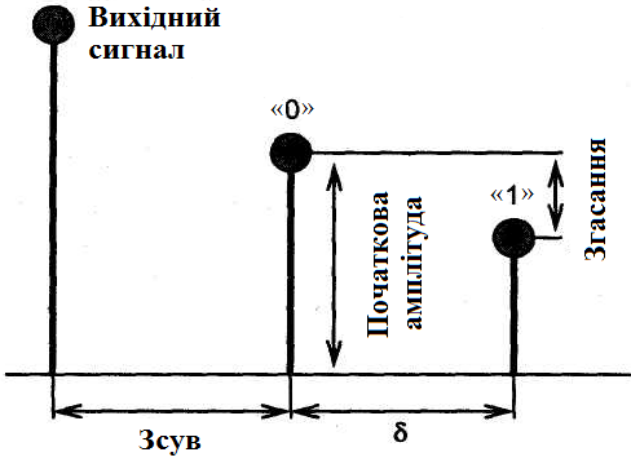


Рис. 5. Регульовані параметри ехо-сигналу

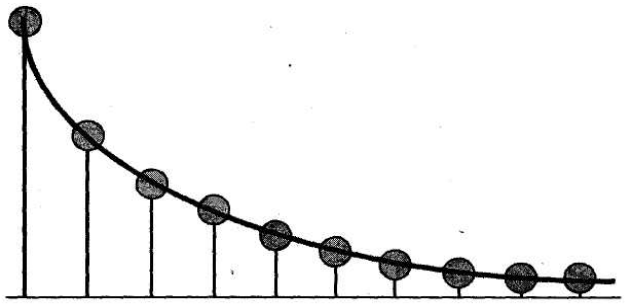


Рис. 6. Дискретний у часі експоненціал

Процес вбудовування даних можна подати у вигляді пристрою, що реалізує одну з двох можливих системних функцій. У часовій області системні функції – це дискретні в часі експоненти (рис. 6), відмінність між якими полягає лише в затримці між імпульсами.

Розглянемо приклад з двома імпульсами (один для копіювання вихідного сигналу, а інший – для формування ехо-сигналу). Очевидно, що збільшення кількості імпульсів призведе до зростання кількості ехо-сигналів.

На рис. 7 подалі представлені системні функції для кодування двійкових «1» і «0».

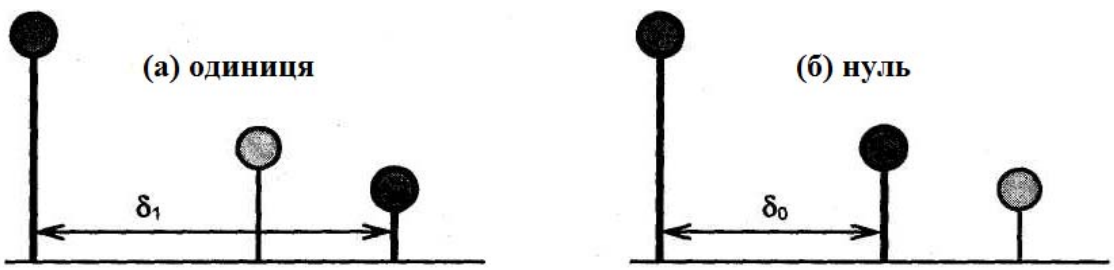


Рис. 7. Регульовані параметри ехо-сигналу

Затримка δ_{bit} між первинним і ехо-сигналом залежить від того, яке представлення або системна функція були застосовані. Представлення «1» створюється затримкою в δ_1 секунд, тоді як представлення «0» – затримкою в δ_0 секунд.

Для того щоб в вихідний сигнал закодувати більше одного біта, сигнал розкладається на менші сегменти. Кожен сегмент при цьому розглядається як окремий сигнал, і в нього можна вбудувати (шляхом ехо-відображення) один біт інформації. Результуючий закодований сигнал (що містить кілька біт) являє собою нове об'єднання всіх незалежно закодованих сегментів вихідного сигналу.

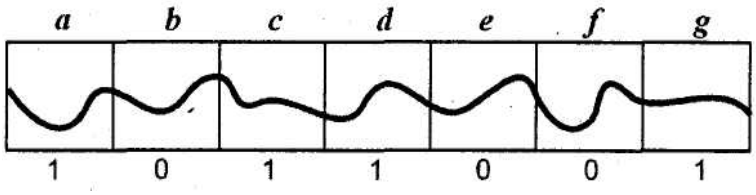


Рис. 8. Розбиття первинного сигналу на менші сегменти для вбудовування інформації, що являє собою послідовність двійкових даних

На рис. 8 зображений приклад, за якого сигнал був розділений на 7 рівних сегментів, позначених як *a, b, c, d, e, f* та *g*.

Нехай необхідно, щоб сегменти *a, c, d* та *g* містили «1». Отже, для кожного з них потрібно застосувати системну функцію представлення одиниці. Кожен сегмент індивідуально згортається з системної функції. Нулі, вміщені в сегменти *b, e* та *f,* кодуються аналогічно, використовуючи спосіб представлення нуля. Отримані після згортання з відповідною функцією результати повторно об'єднуються.

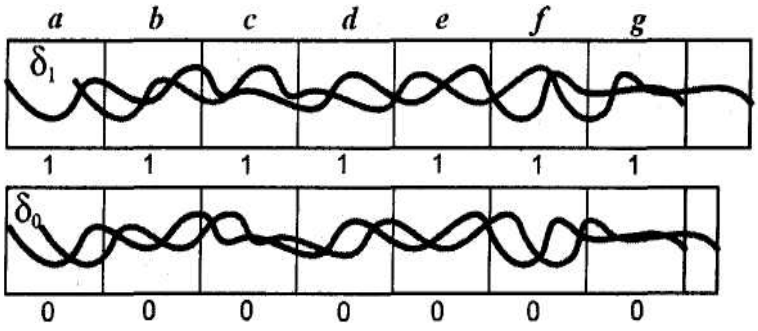


Рис. 9. Створення «одиничного» і «нульового» ехо-сигналів

Для досягнення мінімальної помітності повторного об'єднання, в [2, 3] попередньо пропонується створити окремі «одиничний» та «нульовий» ехо-сигнали, повторюючи первинний і використовуючи відповідні представлення «1» і «0». Отримані в результаті сигнали зображені на рис. 9.

«Одиничний» і «нульовий» ехо-сигнали містять лише одиниці й нулі відповідно. Для того щоб об'єднати ці два сигнали, також створюються два змішувальні сигнали (рис. 10), що являють собою послідовність двійкових даних. Їхній стан залежить від того, який біт необхідно приховати в тому чи іншому сегменті первинного сигналу.

«Одиничний» і «нульовий» сигнали, що змішуються, помножуються на відповідні їм ехо-сигнали. Інакше кажучи, останні масштабуються оди-

ницею або нулем протягом всього терміну дії сигналу в залежності від того, який біт передбачається помістити в будь-який з його окремих сегментів. Надалі обидва результати складаються між собою.

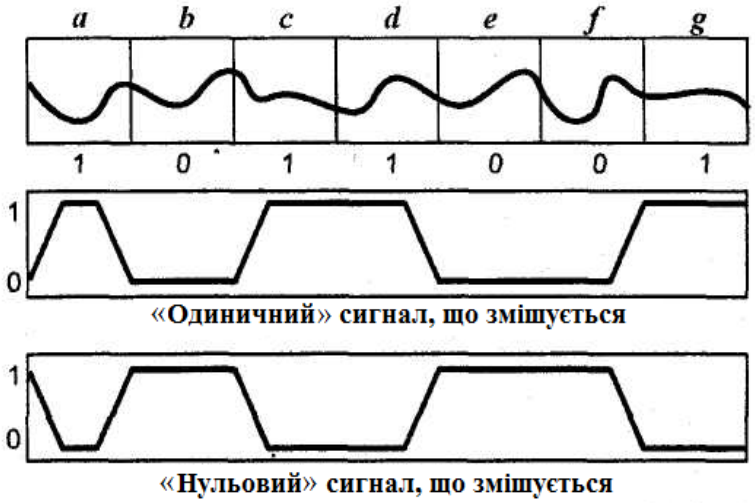


Рис. 10. Сигнали, що змішуються

Слід зауважити, що «нульові» сигнали, що змішуються, є інверсією «одиничного». Крім цього, фронти переходів кожного з сигналів є плавними. Сума обох змішувальних сигналів завжди дорівнює одиниці. Все це дозволяє отримати плавний перехід між сегментами, кодованими різними бітами, а також запобігає виникненню різких змін у звучанні результуючого (змішаного) сигналу. Блок-схему, що відображає повний процес вбудовування, показано на рис. 11.

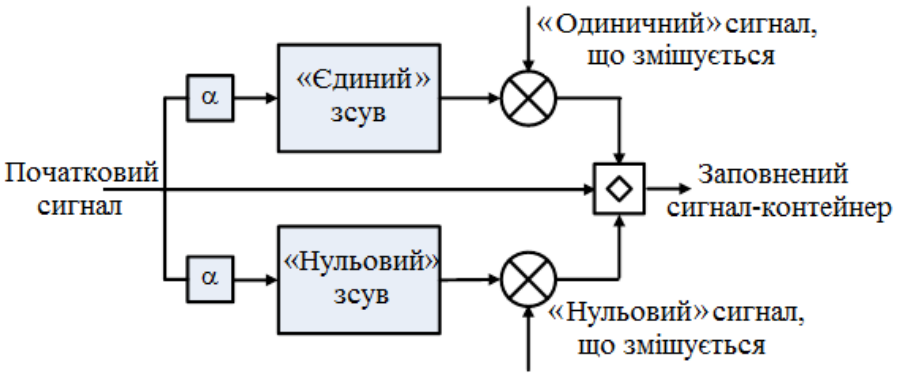


Рис. 11. Блок-схема вбудовування інформації методом ехо-сигналу

Вилучення вкладеної інформації є встановленням інтервалу між ехо-сигналами окремих сегментів. Для цього необхідно дослідити в двох позиціях амплітуду автокореляційної функції (АКФ) косинус-перетворення Фур'є (F) натурального логарифма спектра потужності (або так званого кепстра) кодованого сигналу [2, 3]

$$AC = F^{-1}\left\{\left[\ln_{compl.}\left(F(x)\right)\right]^2\right\}.$$

Розглянемо приклад процесу вилучення, наведений в [2]. Нехай отримано закодований сигнал, що являє собою таку послідовність імпульсів, де останні відокремлені один від одного певним інтервалом і характеризуються експоненціальним загасанням амплітуди. У всіх інших точках сигнал дорівнює нулю (рис. 12). Наступним кроком є пошук кепстра ехо-версії

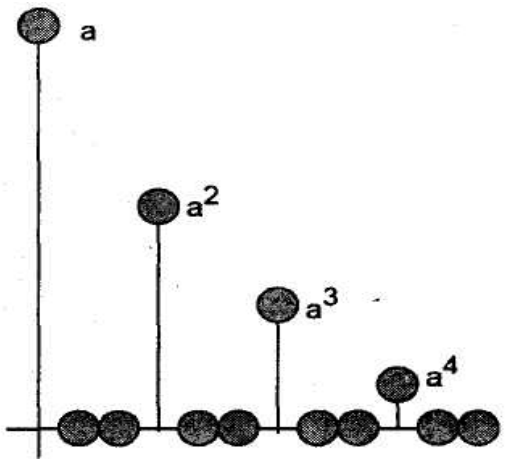


Рис. 12. Приклад сигналу
$x[n] = a^n \cdot u[n]$: $(0 < a < 1)$

сигналу. Результат обчислення кепстра робить інтервал між ехо-сигналом і первинним сигналом дещо більш вираженим.

На жаль, результат обчислення кепстра також дублює ехо-сигнал через кожні δ секунд. На рис. 13 це зображено як послідовність імпульсів на виході. До того ж, амплітуда імпульсів, що подають ехо-сигнал, є малою щодо первинного сигналу. Як наслідок їх важко виявити.

Рішення цієї проблеми полягає в обчисленні АКФ кепстра. За допомогою одноразового відображення сигналу з затримкою δ (рис. 14), отримуємо результат, зображений на рис. 15.

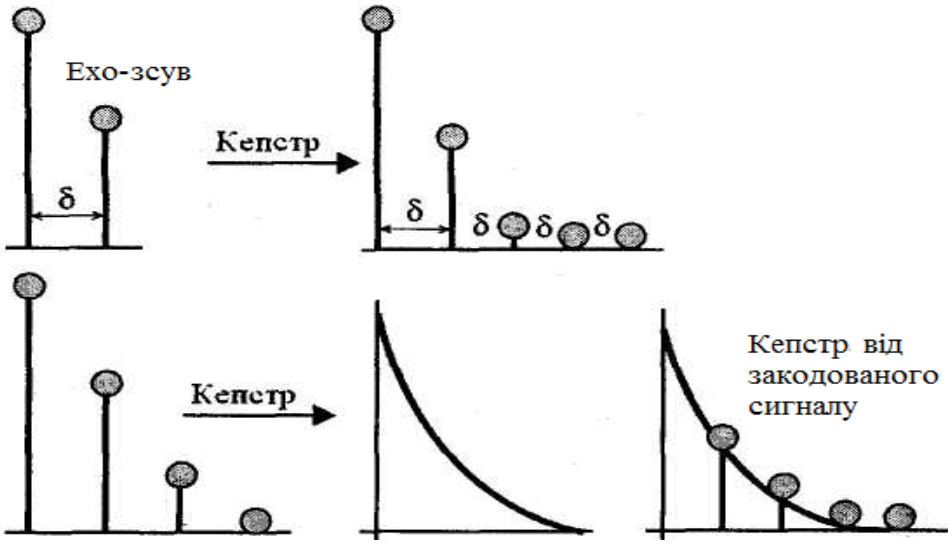


Рис. 13. Процес отримання кепстра від ехо-кодованого сигналу

Водночас значно посиленим є тільки перший імпульс, оскільки його «підтримують» наступні за ним імпульси. Отже, в позиції першого імпульсу ми отримуємо сплеск. Подібно першому імпульсу, сплеск повторюється через δ_1 або ж через δ_0 секунд після сплеску первинного сигналу. Залишкові складові імпульсів наближаються до нуля, що дозволяє ефективно боротися з шумами.

Наведемо критерій ухвалення рішення щодо того, який біт («1» або «0») приховано в часовій затримці сплеску АКФ щодо первинного сигналу. Згадаймо, що «1» кодувалася розміщенням ехо-сигналу через δ_1, а «0» – через δ_0 секунд після оригіналу. Аналогічно у разі вилучення біт є одиничним, якщо значення АКФ через δ_1 секунд більше, ніж через δ_0 секунд. В іншому випадку біт вважається нульовим.

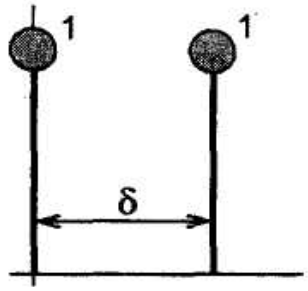


Рис. 14. Принцип відображення сигналу

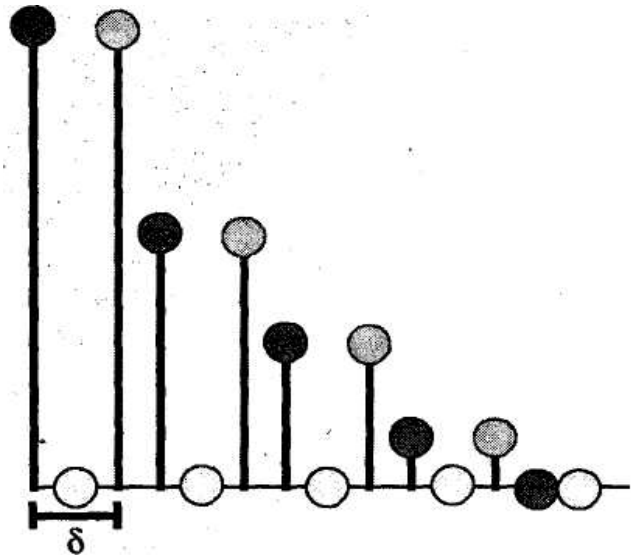


Рис. 15. Результат одноразового відображення сигналу

За твердженням авторів [2], за допомогою цього методу цілком можливо приховувати/вилучати інформацію у вигляді двійкового коду в/з потоку аудіоданих з мінімальною зміною первинного сигналу за пропускної здатності приблизно в 16 біт/с. Під мінімальною зміною мається на увазі той факт, що середньостатистична людина не буде здатна при цьому відчути істотну різницю між модифікованим і первинним сигналами.

Однак у [3] вказано на те, що розглянутий метод не є універсальним – для деяких аудіосигналів неможливо отримати досить високий коефіцієнт правильно розпізнаних під час добування біт навіть за браком перешкод в каналі зв'язку.

5. ПИТАННЯ ДЛЯ ПОТОЧНОГО КОНТРОЛЮ ПІДГОТОВЛЕНОСТІ СТУДЕНТІВ ДО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ

1. Основні властивості ССЛ, що використовуються в стеганографії.
2. Основні властивості аудіоконтейнерів, структура та основні властивості формату аудіофайлів WAV.
3. Методи НЗБ під час вбудовування повідомлень в аудіоконтейнери.
4. Метод фазового кодування.
5. Метод кодування ехо-сигналів.

6. КЕРІВНИЦТВО ДО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ

Завдання 1, 2

Завдання 1, 2 пропонуються для самостійного виконання.

Завдання 3
Реалізація алгоритмів вбудовування та вилучення повідомлень методом кодування початкових фаз

3.1. Завантажуємо вихідні дані: інформаційне повідомлення – текстовий документ (у форматі *.txt).

Виконуємо команду читання інформаційних даних текстового документа з заданого файлу у вигляді одновимірного масиву цілих чисел

«M:=READBIN(“[ім’я файлу].txt”, byte)».

Елементи масиву M_i знаходяться в інтервалі [0...255] і задають значення символів інформаційного повідомлення в кодуванні ASCII (кодування ASCII наведене в додатку). Координати елементів масиву M задають розташування окремих символів інформаційного повідомлення.

Наприклад, нехай в якості інформаційних даних виступає текстовий документ, що зберігається у файлі з ім’ям «1.txt». Тоді після виконання команди читання символів інформаційного повідомлення в кодуванні ASCII

«M:=READBIN(“1.txt”, byte)»

отримуємо:

M =

	0
0	200
1	237
2	242
3	229
4	240
5	229
6	241
7	32
8	...

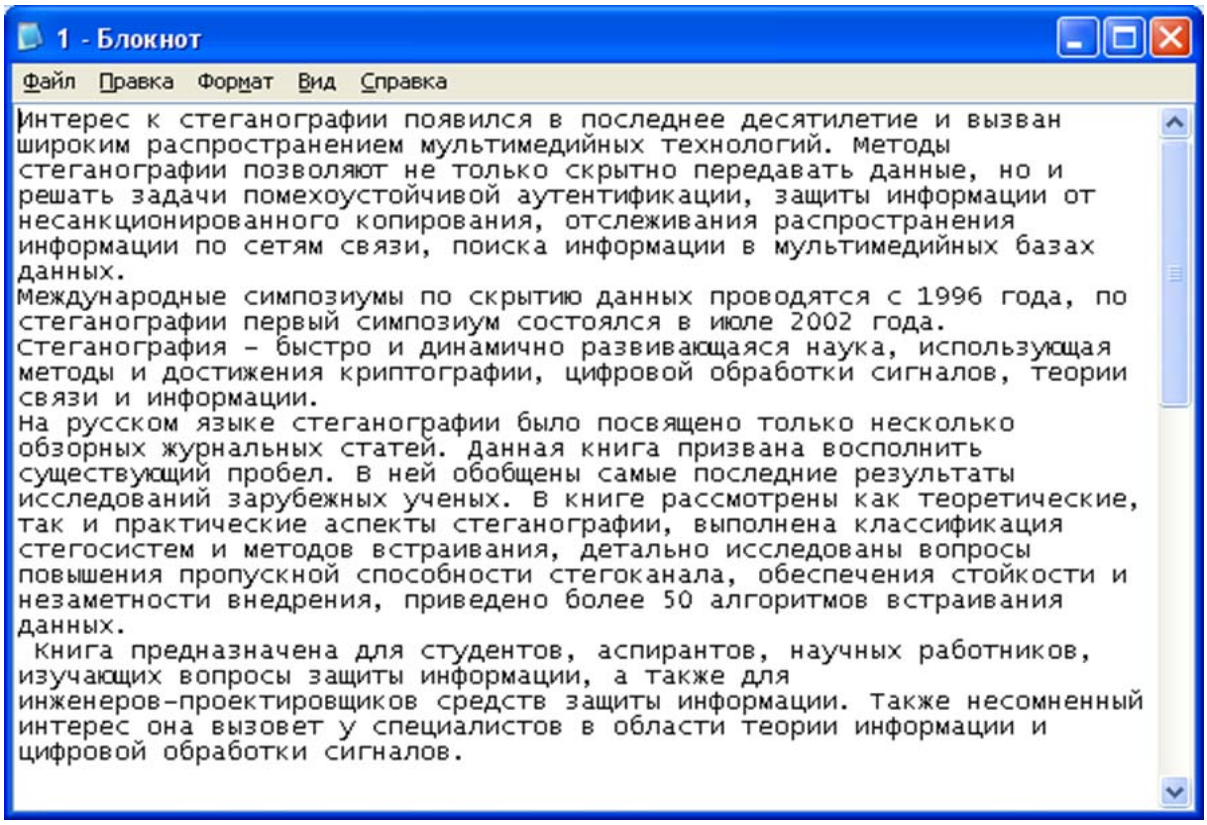
Значення нульового елемента масиву M дорівнює $M_0 = 200$, що відповідає символу «И» в кодуванні ASCII. Наступний елемент масиву $M_1 = 237$ відповідає символу «н» в кодуванні ASCII. Набір перших семи елементів масиву інформаційних даних

{200, 237, 242, 229, 240, 229, 241}

відповідає набору символів повідомлення

{И, н, т, е, р, е, с}

в кодуванні ASCII.



Для розглянутого прикладу в якості інформаційного повідомлення обрано перший абзац з анотації книги «Цифровая стеганография» авторів В. Г. Грибуніна, І. Н. Окова, І. В. Туринцева.

3.2. Перетворимо масив інформаційних даних.

Інформаційні дані в масиві M подані у вигляді набору цілих чисел з інтервалу [0...255], і задають значення символів інформаційного повідомлення в кодуванні ASCII. Для розглянутих стеганографічних методів вбудовування здійснюється побітово. Тобто інформаційні дані з масиву M слід попередньо підготувати, перетворивши їх на бітовий масив. Для цього в середовищі MathCAD використовуємо такі функції.

3.2.1. Функція перетворення вектора-стовпця з восьми біт на десятковий код

$$B\_D(x) := \sum_{i=0}^{7} \left(x_i \cdot 2^i \right).$$

Аргументом x функції B_D (x) є двійковий вектор-стовпець з восьми біт

$$x := \begin{pmatrix} x_0 & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \end{pmatrix}^T.$$

Значенням функції B_D (x) є ціле число в десятковому коді

$$B\_D(x) := x_0 \cdot 2^0 + x_1 \cdot 2^1 + x_2 \cdot 2^2 + x_3 \cdot 2^3 + x_4 \cdot 2^4 + x_5 \cdot 2^5 + x_6 \cdot 2^6 + x_7 \cdot 2^7.$$

Наприклад, нехай аргумент x функції B_D (x) заданий у такий спосіб

$$x := (1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1)^T.$$

Тоді значення функції B_D(x) дорівнює

$$B\_D(x) = 147.$$

3.2.2. Функція перетворення цілого числа в десятковому коді на двійковий вектор-стовпець з восьми біт

$$D\_B(x) := \left| \begin{array}{l} \text{for } i \in 0..7 \\ \quad \left| \begin{array}{l} V_i \leftarrow \text{mod}(x, 2) \\ x \leftarrow \text{floor}\left(\frac{x}{2}\right) \end{array} \right. \\ V \end{array} \right.$$

Аргументом x функції D_B (x) є ціле число в десятковому коді, значенням функції є двійковий вектор-стовпець з восьми біт.

Алгоритм обчислення значення функції D_B (x) є таких. На кожній ітерації (для кожного значення циклової змінної «i», обчислюються такі значення

$V_i \leftarrow \text{mod}(x, 2)$ – приведення десяткового числа x за модулем 2;

$x \leftarrow \text{floor}\left(\frac{x}{2}\right)$ – визначення цілої частини від ділення цілого числа x на два.

Тобто після кожної ітерації число x зменшується в два рази, а значення i-го біта, що повертається функцією двійкового вектора-стовпця, прирівнюється до результату приведення поточного значення x за модулем 2.

Наприклад, нехай аргумент функції D_B (x) дорівнює 27. Тоді значення функції D_B (x) обчислюється так

I = 0: V_0 = mod(27,2) = 1, x = floor(27/2) = 13;
I = 1: V_1 = mod(13,2) = 1, x = floor(13/2) = 6;
I = 2: V_2 = mod(6,2) = 0, x = floor(6/2) = 3;
I = 3: V_3 = mod(3,2) = 1, x = floor(3/2) = 1;
I = 4: V_4 = mod(1,2) = 1, x = floor(1/2) = 0;
I = 5: V_5 = mod(0,2) = 0, x = floor(0/2) = 0;
I = 6: V_6 = mod(0,2) = 0, x = floor(0/2) = 0;
I = 7: V_7 = mod(0,2) = 0, x = floor(0/2) = 0.

Отже, обчислене значення функції D_B(x) дорівнює

$$x = V = (1 \quad 1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0)^T.$$

3.2.3. Використовуючи функцію D_B (x), перетворимо масив M інформаційних цілих чисел на бітовий масив. Для цього скористаємося такою процедурою

$$
\begin{aligned}
M\_b := \Big\| & \text{for } i \in 0..\,\mathrm{rows}(M) - 1 \\
& \quad \Big\| V \leftarrow D\_B\left(M_i\right) \\
& \quad \Big\| \text{for } j \in 0..\,7 \\
& \quad \quad \Big\| M\_b_{i\cdot 8+j} \leftarrow V_j \,, \\
& M\_b
\end{aligned}
$$

Алгоритм формування бітового масиву інформаційних даних такий. Для кожного значення циклової змінної «i» (значення «i» пробігає по всіх індексах масиву M) виконується перетворення $D\_B(M_i)$ за розглянутим вище правилом. Тобто для всіх елементів масиву M формуються відповідні бітові вектори-стовпці. Всі елементи кожного вектора-стовпця перезаписуються до масиву M_b під відповідним індексом. Отже, кожен сформований біт записується до елемента M_b_{i8+j} масиву M_b, де j – циклова змінна, що пробігає всі індекси поточного (i-ого) вектора-стовпця.

Для прикладу розглянемо вихідний вектор-стовпець M, що містить цілі числа (коди) інформаційного повідомлення. Нехай M – масив цілих чисел з попереднього прикладу.

M =

	0
0	200
1	237
2	242
3	229
4	240
5	229
6	241
7	32
8	234
9	32
10	241
11	242
12	229
13	227
14	224
15	237
16	238
17	227
18	240
19	224
20	...

Виконання процедури перетворення масиву M в бітовий масив M_b відбувається так

I = 0: $V = D\_B(200) = (0\ 0\ 0\ 1\ 0\ 0\ 1\ 1)^T$,

J = 0: $M\_b_0 = 0$,
J = 1: $M\_b_1 = 0$,
J = 2: $M\_b_2 = 0$,
J = 3: $M\_b_3 = 1$,
J = 4: $M\_b_4 = 0$,
J = 5: $M\_b_5 = 0$,
J = 6: $M\_b_6 = 1$,
J = 7: $M\_b_7 = 1$;

I = 1: $V = D\_B(237) = (1\ 0\ 1\ 1\ 0\ 1\ 1\ 1)^T$,

J = 0: $M\_b_8 = 0$,
J = 1: $M\_b_9 = 0$,
J = 2: $M\_b_{10} = 0$,
J = 3: $M\_b_{11} = 1$,
J = 4: $M\_b_{12} = 0$,
J = 5: $M\_b_{13} = 0$,
J = 6: $M\_b_{14} = 1$,
J = 7: $M\_b_{15} = 1$;

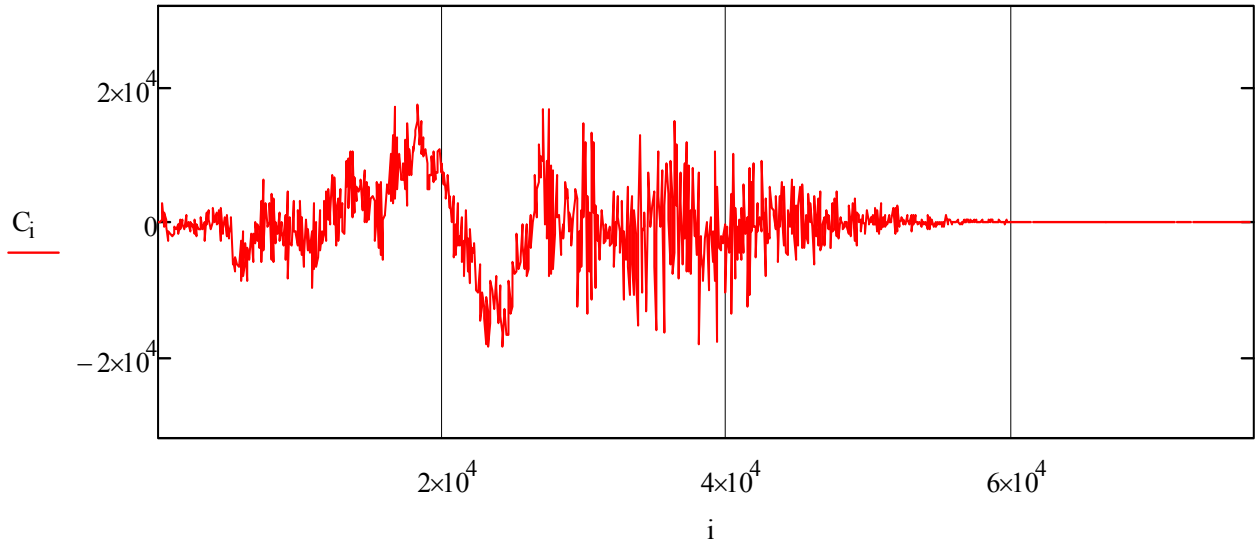
…

3.3. Завантажуємо аудіоконтейнер (в форматі *.wav). Для цього виконуємо такі команди

$$C := READWAV("111.wav")$$

$$W := GETWAVINFO("111.wav") \qquad W = \begin{pmatrix} 1 \\ 1.6 \times 10^4 \\ 16 \\ 3.2 \times 10^4 \end{pmatrix}$$

Перша команда завантажує масиви дискретних відліків з аудіофайлу «111.wav», друга – завантажує параметри цього аудіофайлу.



Значеннями змінної W є параметри аудіофайлу: кількість каналів, частота дискретизації, кількість бітів на один дискретний відлік, обмеження за характеристиками кодера.

Значеннями змінної C є дискретні відліки звукового файлу, наведені в графічному вигляді на рисунку.

3.4. Розіб'ємо контейнер на блоки (сегменти) рівної довжини. Для цього виконаємо команди

$$n := 1024 \qquad segments := ceil\left(\frac{part}{n}\right) - 2$$

$$Seg := \left| \begin{array}{l} \text{for } i \in 0..segments \\ \quad Seg_i \leftarrow submatrix[C, i \cdot n, (i+1) \cdot n - 1, 0, 0] \\ Seg \end{array} \right.$$

Отже, за довжини блоку 1024 отримали segments сегментів, які записані в масив «Seg».

3.5. Виконаємо над кожним сегментом n точкове дискретне перетворення Фур'є

$$\text{SegF} := \left| \begin{array}{l} \text{for } i \in 0..\text{segments} \\ \quad \left| \begin{array}{l} P \leftarrow \text{FFT}\left(\text{Seg}_i\right) \\ \text{SegF}_i \leftarrow P \end{array} \right. \\ \text{SegF} \end{array} \right.$$

В результаті перетворення отримаємо масив сегментів «SegF», що містять комплексні числа. Приклад для першого сегмента має вигляд:

$\text{SegF}_1 =$

	0
0	-317.551
1	-35.981-71.509i
2	-49.038-38.558i
3	-74.925-22.573i
4	-56.192-22.311i
5	-80.249-7.592i
6	-56.877-2.391i
7	-67.003-25.716i
8	-17.223-37.674i
9	674.181+106.513i
10	-243.145+98.23i
11	-179.021+35.377i
12	-154.546+47.73i
13	-251.452+127.724i
14	-36.77-201.079i
15	...

3.6. Знайдемо спектри амплітуд і фаз.

Для цього для кожного комплексного числа обчислимо модуль і аргумент

$$A := \left| \begin{array}{l} \text{for } i \in 0..\text{segments} \\ \quad A_i \leftarrow \overrightarrow{\left|\text{SegF}_i\right|} \\ A \end{array} \right. \qquad F := \left| \begin{array}{l} \text{for } i \in 0..\text{segments} \\ \quad F_i \leftarrow \overrightarrow{\arg\left(\text{SegF}_i\right)} \\ F \end{array} \right.$$

В результаті отримаємо:

$A_1 =$

	0
0	317.551
1	80.051
2	62.381
3	78.251
4	60.46
5	80.608
6	56.928
7	71.769
8	41.424
9	682.543
10	262.238
11	182.483
12	161.748
13	282.031
14	204.414
15	...

$F_1 =$

	0
0	3.142
1	-2.037
2	-2.475
3	-2.849
4	-2.764
5	-3.047
6	-3.1
7	-2.775
8	-2
9	0.157
10	2.758
11	2.946
12	2.842
13	2.672
14	-1.752
15	...

Перевіримо правильність виконаних операцій. Для цього знайдемо вихідні комплексні числа, використовуючи формулу Ейлера:

$\overrightarrow{\left(A_1 \cdot e^{i \cdot F_1}\right)} =$

	0
0	-317.551
1	-35.981-71.509i
2	-49.038-38.558i
3	-74.925-22.573i
4	-56.192-22.311i
5	-80.249-7.592i
6	-56.877-2.391i
7	-67.003-25.716i
8	-17.223-37.674i
9	674.181+106.513i
10	-243.145+98.23i
11	-179.021+35.377i
12	-154.546+47.73i
13	-251.452+127.724i
14	-36.77-201.079i
15	...

3.7. Обчислимо масив різниць фаз, як приклад наведено масив для першого блоку

$$dF := \left| \begin{array}{l} dF_0 \leftarrow 0 \\ \text{for } i \in 1 .. \text{segments} \\ \quad dF_i \leftarrow F_i - F_{i-1} \\ dF \end{array} \right.$$

$dF_1 =$

	0
0	3.142
1	-3.966
2	-2.376
3	-3.603
4	-3.203
5	-2.741
6	-2.262
7	-2.229
8	-0.566
9	0.422
10	2.129
11	2.205
12	2.337
13	3.027
14	-2.349
15	...

3.8. Реалізуємо кодування початкових фаз. Для цього запишемо до масиву початкових фаз значення за таким правилом:

$$Fdata := \left| \begin{array}{l} \text{for } i \in 0 .. \frac{n}{2} \\ \quad \left| \begin{array}{l} Fdata_i \leftarrow -\frac{\pi}{2} \text{ if } m_i = 1 \\ Fdata_i \leftarrow \frac{\pi}{2} \text{ if } m_i = 0 \end{array} \right. \\ Fdata \end{array} \right.$$

$Fdata =$

	0
0	-1.571
1	-1.571
2	1.571
3	1.571
4	1.571
5	-1.571
6	1.571
7	1.571
8	-1.571
9	1.571
10	-1.571
11	-1.571
12	1.571
13	-1.571
14	1.571
15	...

$m =$

	0
0	1
1	1
2	0
3	0
4	0
5	1
6	0
7	0
8	1
9	0
10	1
11	1
12	0
13	1
14	0
15	...

3.9. Розрахуємо нові значення початкових фаз. Для цього будемо використовувати масив інформаційних фаз (помістимо його на місце нульового блоку) і масив різниць фаз з п. 3.7.

$$FF := \left| \begin{array}{l} FF_0 \leftarrow Fdata \\ \text{for } i \in 1..segments \\ \quad FF_i \leftarrow FF_{i-1} + dF_i \\ FF \end{array} \right.$$

$FF_0 =$

	0
0	-1.571
1	-1.571
2	1.571
3	1.571
4	1.571
5	-1.571
6	1.571
7	1.571
8	-1.571
9	1.571
10	-1.571
11	-1.571
12	1.571
13	-1.571
14	1.571
15	...

$FF_1 =$

	0
0	1.571
1	-5.537
2	-0.806
3	-2.032
4	-1.632
5	-4.312
6	-0.691
7	-0.659
8	-2.136
9	1.993
10	0.558
11	0.634
12	3.908
13	1.456
14	-0.779
15	...

3.10. Для кожного отриманого блоку початкових фаз з використанням спектра амплітуд з п. 3.6 виконаємо зворотне перетворення Фур'є:

$$Segg := \left| \begin{array}{l} \text{for } i \in 0..segments \\ \quad \left| \begin{array}{l} P \leftarrow \overrightarrow{\left(A_i \cdot e^{i \cdot FF_i} \right)} \\ Segg_i \leftarrow IFFT(P) \end{array} \right. \\ Segg \end{array} \right.$$

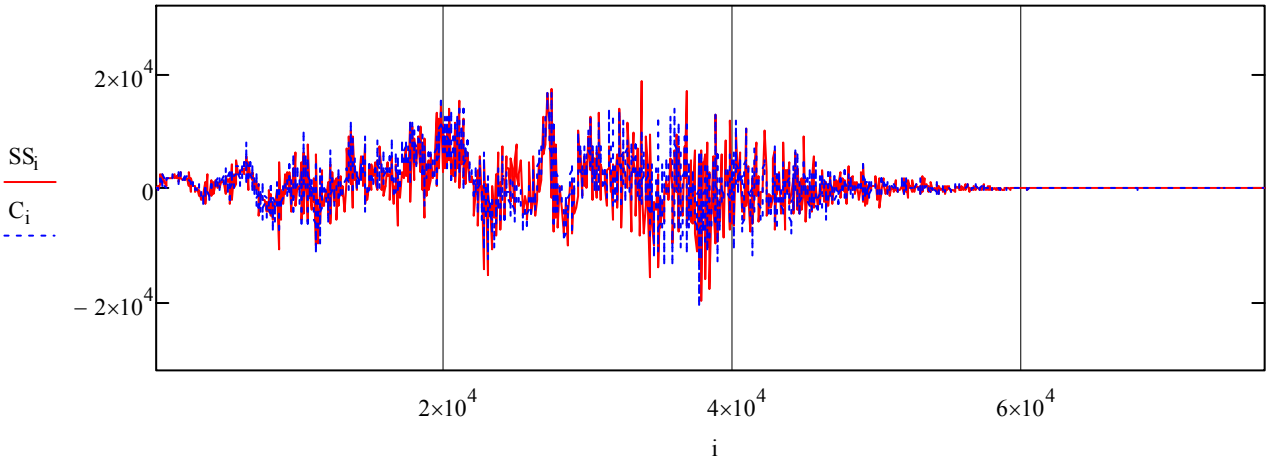
$Segg_1 =$

	0
0	$7.984 \cdot 10^3$
1	$7.307 \cdot 10^3$
2	$6.414 \cdot 10^3$
3	$5.31 \cdot 10^3$
4	$3.784 \cdot 10^3$
5	$2.149 \cdot 10^3$
6	617.737
7	-832.724
8	$-1.775 \cdot 10^3$
9	$-2.132 \cdot 10^3$
10	$-2.018 \cdot 10^3$
11	$-1.817 \cdot 10^3$
12	$-1.617 \cdot 10^3$
13	$-1.42 \cdot 10^3$
14	-907.981
15	...

Нові значення сигналу вже містять вбудовані біти даних за допомогою кодування абсолютних значень початкових фаз. Відносні зміни фаз модифіковані, вони залишилися такими, якими були до вбудовування, тобто відповідають п. 3.7.

3.11. Об’єднуємо отримані блоки в один масив і виводимо його в графічному вигляді

```
SS := | SS ← Segg_0
      | for i ∈ 1 .. segments
      |   SS ← stack(SS, Segg_i)
      | SS ← stack(SS, 0)
      | SS
```



Очевидно, що спотворення є, однак отриманий сигнал в цілому повторює форму вихідного аудіосигналу.

3.12. Запишемо отриманий масив в новий аудіофайл

WRITEWAV ("Stego.wav" , fd , Q) := SS

3.13. Реалізуємо вилучення вбудованих даних. Для цього завантажимо масив даних аудіоконтейнера з файлу і виділимо тільки нульовий блок даних:

C := READWAV("Stego.wav")

G := submatrix(C, 1, n, 0, 0)

G =

	0
0	-112
1	-202
2	-560
3	$-1.148 \cdot 10^3$
4	$-1.435 \cdot 10^3$
5	$-1.346 \cdot 10^3$
6	$-1.207 \cdot 10^3$
7	-591
8	130
9	798
10	$1.762 \cdot 10^3$
11	$3.119 \cdot 10^3$
12	$4.576 \cdot 10^3$
13	$5.898 \cdot 10^3$
14	$6.786 \cdot 10^3$
15	...

3.14. Виконаємо перетворення Фур'є для першого блоку і знайдемо масив початкових фаз:

$TG := FFT(G)$

$Fdata\_1 := arg(TG)$

TG =

	0
0	24.262
1	-5.177+1.994i
2	-11.808-7.784i
3	-32.784+2.088i
4	24.893-5.525i
5	42.397+12.397i
6	-91.661+51.198i
7	-2.41+10.012i
8	0.011+17.149i
9	$1.589 \cdot 10^{-3}$+47.742i
10	-0.013+57.856i
11	-0.016-19.669i
12	-0.033+34.409i
13	-0.018+45.387i
14	-0.024-15.244i
15	...

Fdata_1=

	0
0	0
1	2.774
2	-2.559
3	3.078
4	-0.218
5	0.284
6	2.632
7	1.807
8	1.57
9	1.571
10	1.571
11	-1.572
12	1.572
13	1.571
14	-1.572
15	...

3.15. Вилучимо вбудовані дані, порівняємо їх з попередньо вбудованим:

$m\_1 :=$ for $i \in 0..\text{rows}(Fdata\_1) - 1$

$m\_1_i \leftarrow 0$ if $Fdata\_1_i > 0$

$m\_1_i \leftarrow 1$ if $Fdata\_1_i \le 0$

m_1

m_1 =

	0
0	1
1	1
2	0
3	0
4	0
5	1
6	0
7	0
8	1
9	0
10	1
11	1
12	0
13	1
14	0
15	...

m =

	0
0	1
1	1
2	0
3	0
4	0
5	1
6	0
7	0
8	1
9	0
10	1
11	1
12	0
13	1
14	0
15	...

Дані вилучено вірно. Виконати порівняння «на слух» порожнього і заповненого контейнера та зробити висновки пропонується самостійно.

Завдання 4
Реалізація алгоритмів вбудовування та вилучення повідомлень методом кодування exo-сигналів

4.1. Завантажуємо вихідні дані як в п. 3.1 – 3.3.

4.2. Розбиваємо контейнер на чотири блоки

$$n := \text{rows}(C) - 1 \qquad k := 4 \qquad p := \text{round}\left(\frac{n+1}{k}\right)$$

$$C := \text{submatrix}(S, 0, n, 0, 0)$$

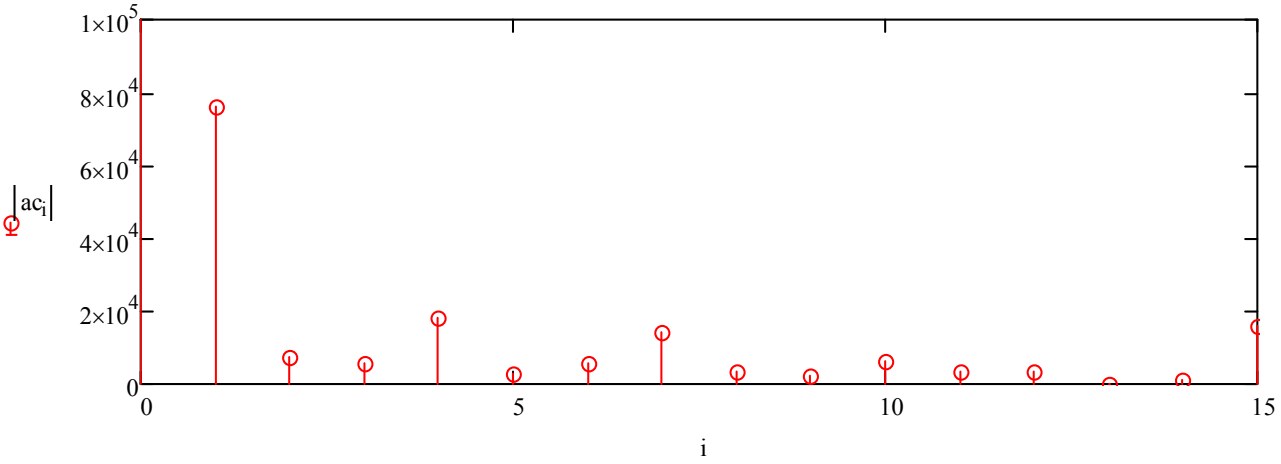
$$\text{SegC} := \left| \begin{array}{l} \text{for } i \in 0..k-1 \\ \quad \text{SegC}_i \leftarrow \text{submatrix}[S, ip, (i+1)\cdot p - 1, 0, 0] \\ \text{SegC} \end{array} \right.$$

4.3. Задаємо функцію обчислення коефіцієнта кепстра

$$AC(x) := \text{ICFFT}\left(\ln(\text{CFFT}(x))^2\right)$$

4.4. Обчислимо коефіцієнт кепстра для всього контейнера

$$ac := AC(C)$$



4.5. Виберемо дві точки для вбудовування інформаційних біт

$$\tau 1 := 7 \qquad p1 := 2$$
$$\tau 2 := 14 \qquad p2 := 2$$

У цьому разі ми вибрали затримку 7 і 14 інтервалів дискретизації і рівень загасання, що дорівнює 2.

4.6. Формуємо два штучні exo-сигнали

$$\text{Exo1} := \left| \begin{array}{l} \text{for } i \in 0..k-1 \\ \quad \left| \begin{array}{l} \text{for } j \in 0..\tau 1-1 \\ \quad \text{Exo1}_{i\cdot p+j} \leftarrow 0 \\ \text{for } j \in \tau 1..p-1 \\ \quad \text{Exo1}_{i\cdot p+j} \leftarrow \text{floor}\left(\frac{C_{i\cdot p+j-\tau 1}}{p1}\right) \end{array} \right. \\ \text{Exo1} \end{array} \right.$$

$$\text{Exo2} := \left| \begin{array}{l} \text{for } i \in 0..k-1 \\ \quad \left| \begin{array}{l} \text{for } j \in 0..\tau 2-1 \\ \quad \text{Exo2}_{i\cdot p+j} \leftarrow 0 \\ \text{for } j \in \tau 2..p-1 \\ \quad \text{Exo2}_{i\cdot p+j} \leftarrow \text{floor}\left(\frac{C_{i\cdot p+j-\tau 2}}{p1}\right) \end{array} \right. \\ \text{Exo2} \end{array} \right.$$

4.7. Формуємо новий аудіосигнал, до якого додаємо один з exo-сигналів (в залежності від значення вбудованих даних)

$$\text{C\_Sum} := \left| \begin{array}{l} \text{for } i \in 0..k-1 \\ \quad \text{for } j \in 0..p-1 \\ \quad \left| \begin{array}{l} \text{C\_Sum}_{i\cdot p+j} \leftarrow C_{i\cdot p+j} + \text{Exo1}_{i\cdot p+j} \quad \text{if } m_i = 1 \\ \text{C\_Sum}_{i\cdot p+j} \leftarrow C_{i\cdot p+j} + \text{Exo2}_{i\cdot p+j} \quad \text{if } m_i = 0 \end{array} \right. \\ \text{C\_Sum} \end{array} \right.$$

Для наочності наведемо власні значення вихідного сигналу, штучних відлунь, результату вбудовування та інформаційного повідомлення:

C =

	0
0	0
1	0
2	2
3	-2
4	2
5	3
6	4
7	4
8	2
9	4
10	2
11	4
12	2
13	4
14	4
15	…

Exo1 =

	0
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	1
10	-1
11	1
12	1
13	2
14	2
15	…

Exo2 =

	0
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0
10	0
11	0
12	0
13	0
14	0
15	…

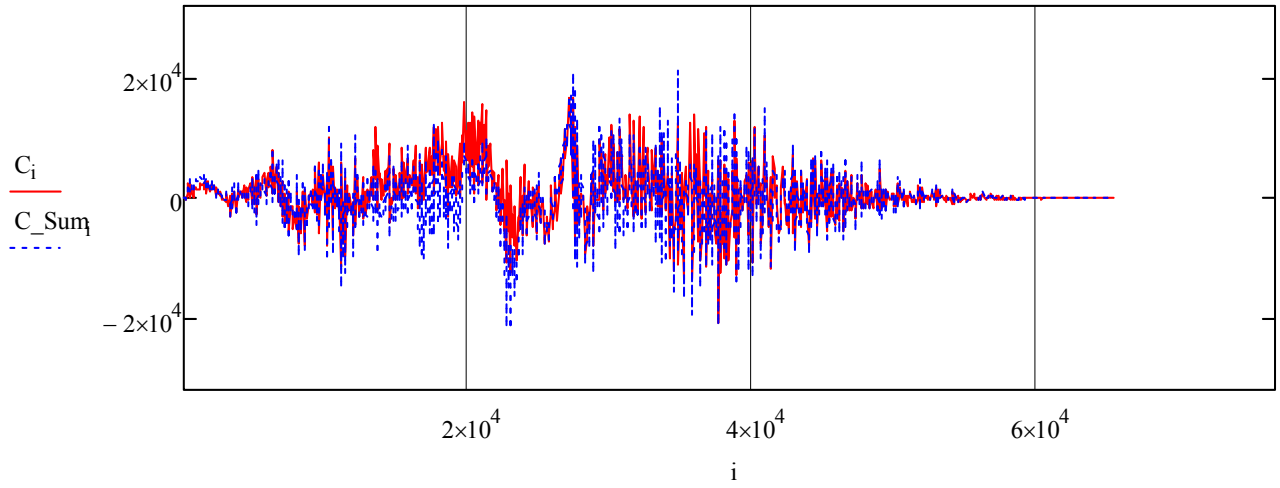
C_Sum=

	0
0	0
1	0
2	2
3	-2
4	2
5	3
6	4
7	4
8	2
9	5
10	1
11	5
12	3
13	6
14	6
15	…

m =

	0
0	1
1	1
2	0
3	0
4	0
5	1
6	0
7	0
8	1
9	0
10	1
11	1
12	0
13	1
14	0
15	…

4.8. Порівняємо вихідний і заповнений контейнери



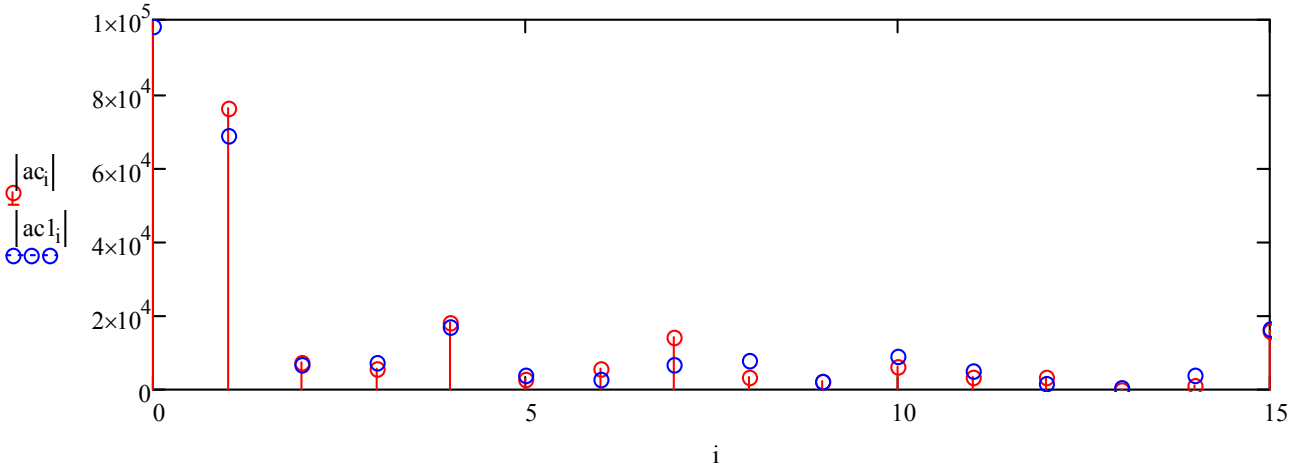
Очевидно, що спотворення є, але за формою сигнали відповідають один одному.

Зберігаємо отриманий масив у звуковий файл

$$WRITEWAV\ ("Stego\_Exo.wav"\ \ ,fd\ ,Q) := C\_Sum$$

4.9. Для вилучення інформаційного повідомлення завантажимо масив даних. Спершу обчислимо кепстр всього контейнера і порівняємо його з даними з п. 4.4.

$$C := READWAV("Stego\_Exo.wav") \qquad ac1 := AC(C)$$



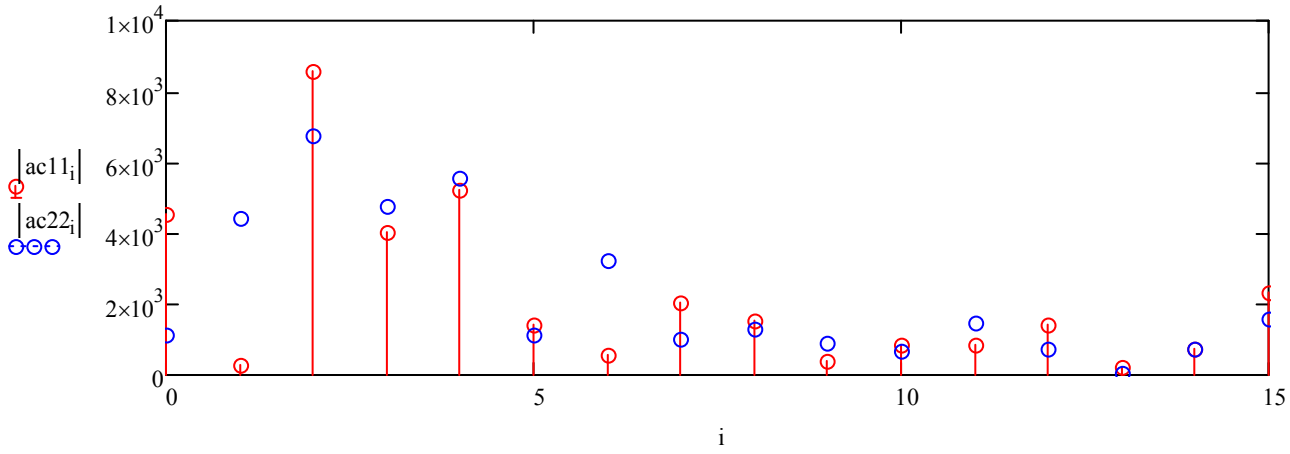
Очевидно, що за цими «усередненими» даними важко витягти чотири вбудованих біта.

4.10. Розіб'ємо заповнений контейнер на блоки і обчислимо коефіцієнт кепстра для кожного блоку

$$SegC\_Sum := \left| \begin{array}{l} \text{for}\ \ i \in 0..k-1 \\ \quad SegC\_Sum_i \leftarrow submatrix[C, i\cdot p, (i+1)\cdot p - 1, 0, 0] \\ SegC\_Sum \end{array} \right. \qquad ac22 := AC\left(SegC\_Sum_0\right)$$

$$ac11 := AC\left(SegC_0\right)$$

Для наочності наведемо малюнок з діаграмою коефіцієнтів кепстра нульових блоків аудіоконтейнерів до і після вбудовування:



Очевидно, що після вбудовування з'явився сплеск в межах першої точки вбудовування (шо відповідає «1»).

4.11. Задамо періодичну і аперіодичну функцію автокореляції

$$RP(S,e) := \frac{\sum_{i=0}^{rows(S)-1} \left(S_i \cdot S_{mod(i+e,rows(S))}\right)}{rows(S)} \qquad RA(S,e) := \frac{\sum_{i=0}^{rows(S)-1-e} \left(S_i \cdot S_{i+e}\right)}{rows(S)}$$

Під час вилучення повідомлення можемо використовувати будь-яку з них.

4.12. Вилучаємо повідомлення, порівнюючи між собою модуль коефіцієнтів функцій автокореляції в точках вбудовування. Для цього спочатку знайдемо кепстр для всіх блоків

$$ac := \left| \begin{array}{l} \text{for } i \in 0..rows(SegC\_Sum) - 1 \\ \quad ac_i \leftarrow AC\left(SegC\_Sum_i\right) \\ ac \end{array} \right.$$

Потім обчислимо коефіцієнти кореляції і порівняємо їх модулі

$$m2 := \left| \begin{array}{l} \text{for } i \in 0..rows(ac) - 1 \\ \quad \left| \begin{array}{l} a \leftarrow RA\left(ac_i, \tau 1\right) \\ b \leftarrow RA\left(ac_i, \tau 2\right) \\ m_i \leftarrow 0 \text{ if } |a| \le |b| \\ m_i \leftarrow 1 \text{ if } |a| > |b| \end{array} \right. \\ m \end{array} \right.$$

$$m2 = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

m =

	0
0	1
1	1
2	0
3	0
4	0
5	...

Очевидно, що порівняння вбудованого і вилученого повідомлення показує їх ідентичність. Однак це спостерігається не завжди. Підбір вдалих параметрів для надійного вилучення вбудованих даних пропонується виконати самостійно.

Виконати порівняння «на слух» порожнього і заповненого контейнера і зробити висновки пропонується самостійно.

7. ПРИКЛАД ОФОРМЛЕННЯ ЗВІТУ З ЛАБОРАТОРНОЇ РОБОТИ

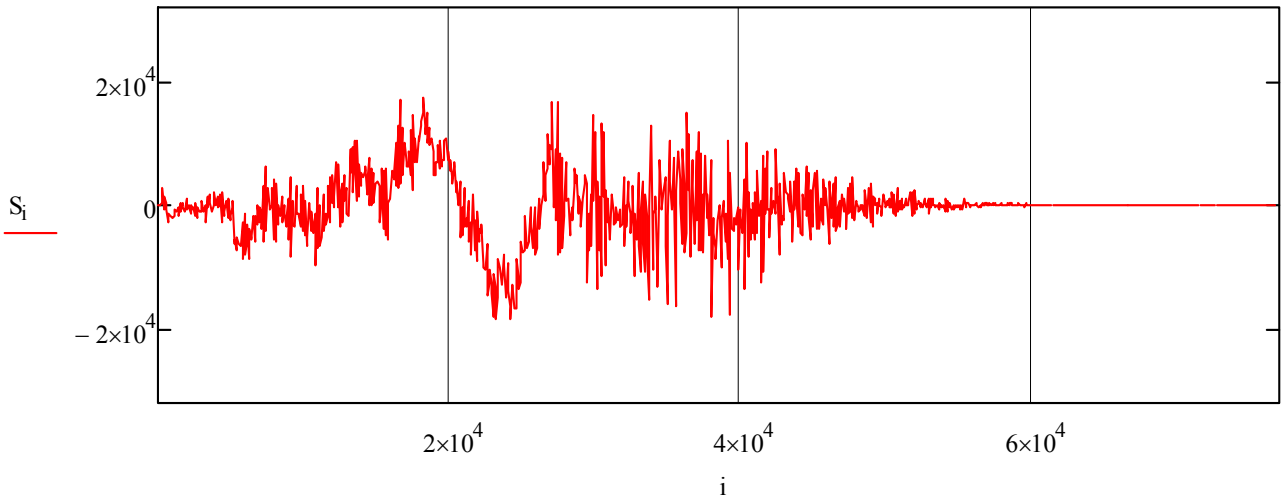
$C := READWAV("111.wav")$

$W := GETWAVINFO("111.wav")$ $\quad W = \begin{pmatrix} 1 \\ 1.6 \times 10^4 \\ 16 \\ 3.2 \times 10^4 \end{pmatrix}$

$M := READBIN("1.txt", "byte")$

$$D2B(x) := \left| \begin{array}{l} \text{for } i \in 0..7 \\ \quad V_i \leftarrow \text{mod}(x, 2) \\ \quad x \leftarrow \text{floor}\left(\frac{x}{2}\right) \\ V \end{array} \right. \qquad m := \left| \begin{array}{l} \text{for } i \in 0..64 \\ \quad b \leftarrow D2B(M_i) \\ \quad \text{for } j \in 0..7 \\ \quad\quad m_{8 \cdot i + j} \leftarrow b_j \\ m \end{array} \right.$$

$$S := \left| \begin{array}{l} \text{for } i \in 0..76800 \\ \quad S_i \leftarrow C_i \\ S \end{array} \right.$$



$$Seg := \left| \begin{array}{l} \text{for } i \in 0..74 \\ \quad Seg_i \leftarrow \text{submatrix}[S, i \cdot 1024, (i+1) \cdot 1024 - 1, 0, 0] \\ Seg \end{array} \right.$$

$$SegF := \left| \begin{array}{l} \text{for } i \in 0..74 \\ \quad P \leftarrow FFT\left(Seg_i\right) \\ \quad SegF_i \leftarrow P \\ SegF \end{array} \right.$$

$Seg_0 =$

	0
0	94
1	5
2	-4
3	3
4	1
5	-2
6	0
7	-2
8	-5
9	-8
10	-8
11	-11
12	-14
13	-10
14	-10
15	...

$S =$

	0
0	94
1	5
2	-4
3	3
4	1
5	-2
6	0
7	-2
8	-5
9	-8
10	-8
11	-11
12	-14
13	-10
14	-10
15	...

$$A := \left| \begin{array}{l} \text{for } i \in 0..74 \\ \quad A_i \leftarrow \overrightarrow{\left|SegF_i\right|} \\ A \end{array} \right.$$

$$F := \left| \begin{array}{l} \text{for } i \in 0..74 \\ \quad F_i \leftarrow \overrightarrow{\arg\left(SegF_i\right)} \\ F \end{array} \right.$$

$SegF_1 =$

	0
0	-36.263
1	5.996-35.013i
2	-22.457-12.154i
3	-17.277+62.537i
4	-67.994+33.644i
5	-154.52+16.917i
6	242.147-101.939i
7	74.498-2.968i
8	121.734-9.244i
9	-15.865+88.575i
10	7.213+32.235i
11	-4.543+29.932i
12	-14.607+4.999i
13	-12.015+3.231i
14	12.034+18.405i
15	...

$A_1 =$

	0
0	36.263
1	35.522
2	25.535
3	64.879
4	75.862
5	155.444
6	262.73
7	74.557
8	122.084
9	89.985
10	33.032
11	30.275
12	15.439
13	12.442
14	21.991
15	...

$F_1 =$

	0
0	3.142
1	-1.401
2	-2.646
3	1.84
4	2.682
5	3.033
6	-0.398
7	-0.04
8	-0.076
9	1.748
10	1.351
11	1.721
12	2.812
13	2.879
14	0.992
15	...

$\overrightarrow{\left(A_1 \cdot e^{i \cdot F_1}\right)} =$

	0
0	-36.263
1	5.996-35.013i
2	-22.457-12.154i
3	-17.277+62.537i
4	-67.994+33.644i
5	-154.52+16.917i
6	242.147-101.939i
7	74.498-2.968i
8	121.734-9.244i
9	-15.865+88.575i
10	7.213+32.235i
11	-4.543+29.932i
12	-14.607+4.999i
13	-12.015+3.231i
14	12.034+18.405i
15	...

$$dF := \left| \begin{array}{l} dF_0 \leftarrow 0 \\ \text{for } i \in 1..74 \\ \quad dF_i \leftarrow F_i - F_{i-1} \\ dF \end{array} \right.$$

$dF_{11} =$

	0
0	-3.142
1	2.921
2	-2.256
3	3.175
4	3.303
5	3.396
6	-2.738
7	-2.539
8	-4.989
9	1.839
10	-0.336
11	0.256
12	-2.421
13	-5.581
14	3.369
15	...

$$\text{Fdata} := \left| \begin{array}{l} \text{for } i \in 0..7 \\ \quad \text{Fdata}_i \leftarrow \left(F_0\right)_i \\ \text{for } i \in 8..15 \\ \quad \left| \begin{array}{l} \text{Fdata}_i \leftarrow -\frac{\pi}{2} \quad \text{if } m_{i-8} = 1 \\ \text{Fdata}_i \leftarrow \frac{\pi}{2} \quad \text{if } m_{i-8} = 0 \end{array} \right. \\ \text{for } i \in 16..512 \\ \quad \text{Fdata}_i \leftarrow \left(F_0\right)_i \\ \text{Fdata} \end{array} \right.$$

$m =$

	0
0	0
1	0
2	0
3	1
4	0
5	0
6	1
7	1
8	1
9	0
10	1
11	1
12	0
13	1
14	1
15	...

$\text{Fdata} =$

	0
0	0
1	2.774
2	-2.558
3	3.078
4	-0.218
5	0.284
6	2.632
7	1.808
8	1.571
9	1.571
10	1.571
11	-1.571
12	1.571
13	1.571
14	-1.571
15	...

$$\text{FF} := \left| \begin{array}{l} \text{FF}_0 \leftarrow \text{Fdata} \\ \text{for } i \in 1..74 \\ \quad \text{FF}_i \leftarrow \text{FF}_{i-1} + \text{dF}_i \\ \text{FF} \end{array} \right.$$

$F_0 =$

	0
0	0
1	2.774
2	-2.558
3	3.078
4	-0.218
5	0.284
6	2.632
7	1.808
8	-2.51
9	2.204
10	-2.013
11	1.672
12	-0.765
13	-2.65
14	-0.611
15	...

$$\text{Segg} := \left| \begin{array}{l} \text{for } i \in 0..74 \\ \quad \left| \begin{array}{l} P \leftarrow \overrightarrow{\left(A_i \cdot e^{i \cdot FF_i}\right)} \\ \text{Segg}_i \leftarrow \text{IFFT}(P) \end{array} \right. \\ \text{Segg} \end{array} \right.$$

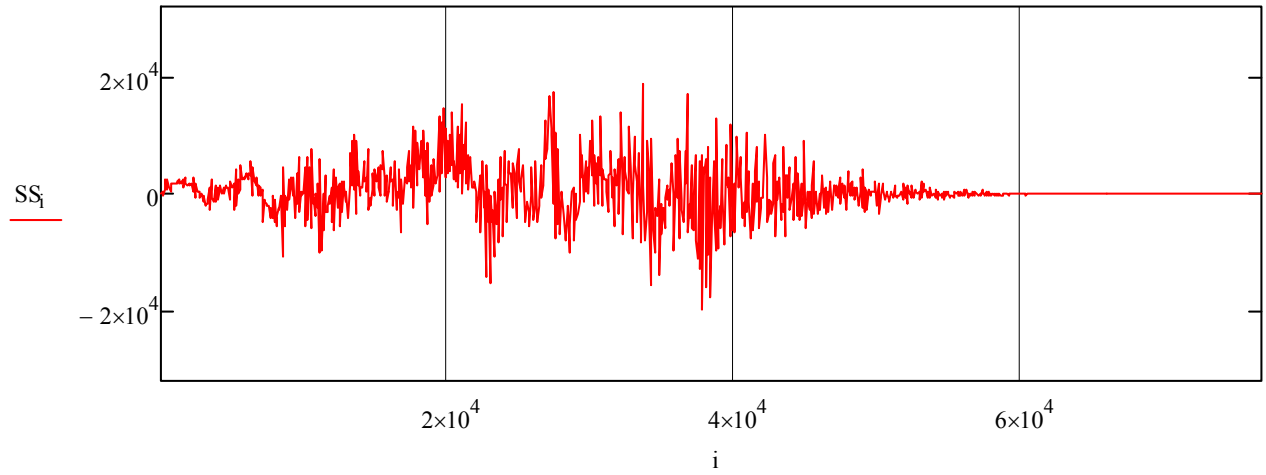
$\text{Segg}_1 =$

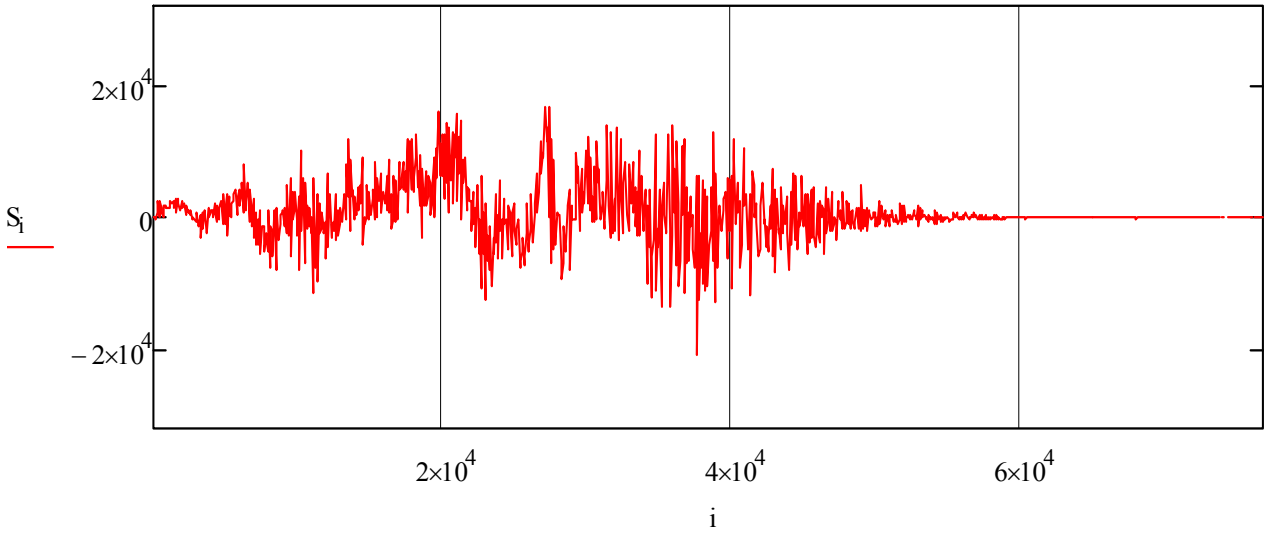
	0
0	$1.921 \cdot 10^3$
1	933.425
2	151.188
3	$-1.52 \cdot 10^3$
4	$-2.151 \cdot 10^3$
5	$-2.516 \cdot 10^3$
6	$-2.148 \cdot 10^3$
7	$-1.514 \cdot 10^3$
8	-665.748
9	$1.29 \cdot 10^3$
10	$2.16 \cdot 10^3$
11	$3.234 \cdot 10^3$
12	$3.329 \cdot 10^3$
13	$3.039 \cdot 10^3$
14	$1.555 \cdot 10^3$
15	...

$$\text{SS} := \left| \begin{array}{l} \text{SS} \leftarrow \text{Segg}_0 \\ \text{for } i \in 1..74 \\ \quad \text{SS} \leftarrow \text{stack}\left(\text{SS}, \text{Segg}_i\right) \\ \text{SS} \leftarrow \text{stack}(\text{SS}, 0) \\ \text{SS} \end{array} \right.$$

SS =

	0
0	246.983
1	127.756
2	88.09
3	64.129
4	31.02
5	-3.093
6	-32.063
7	-64.745
8	-97.996
9	-130.678
10	-159.652
11	-190.785
12	-220.95
13	-243.023
14	-267.887
15	...





WRITEWAV ("Stego.wav" , fd , Q) := SS

C := READWAV ("Stego.wav")

G := submatrix(C, 0, 1023, 0, 0)

TG := FFT(G)

Fdata_1 := arg(TG)

$$m\_1 := \left| \begin{array}{l} \text{for } i \in 8..15 \\ \quad \left| \begin{array}{l} m\_1_{i-8} \leftarrow 0 \text{ if } Fdata\_1_{1} > 0 \\ m\_1_{i-8} \leftarrow 1 \text{ if } Fdata\_1_{1} \le 0 \end{array} \right. \\ m\_1 \end{array} \right.$$

$G =$

	0
0	246
1	127
2	88
3	64
4	31
5	-3
6	-32
7	-64
8	-97
9	-130
10	-159
11	-190
12	-220
13	-243
14	-267
15	...

$TG =$

	0
0	24.262
1	-5.177+1.994i
2	-11.808-7.784i
3	-32.784+2.088i
4	24.893-5.525i
5	42.397+12.397i
6	-91.661+51.198i
7	-2.41+10.012i
8	0.011+17.149i
9	$1.589 \cdot 10^{-3}$+47.742i
10	-0.013+57.856i
11	-0.016-19.669i
12	-0.033+34.409i
13	-0.018+45.387i
14	-0.024-15.244i
15	...

$$m\_1 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

$Fdata\_1 =$

	0
0	0
1	2.774
2	-2.559
3	3.078
4	-0.218
5	0.284
6	2.632
7	1.807
8	1.57
9	1.571
10	1.571
11	-1.572
12	1.572
13	1.571
14	-1.572
15	...

$FF_0 =$

	0
0	0
1	2.774
2	-2.558
3	3.078
4	-0.218
5	0.284
6	2.632
7	1.808
8	1.571
9	1.571
10	1.571
11	-1.571
12	1.571
13	1.571
14	-1.571
15	...

Метод кодування exo-сигналів

$$C := \text{submatrix}(S, 0, 65535, 0, 0)$$

Розбиваємо контейнер на чотири блоки

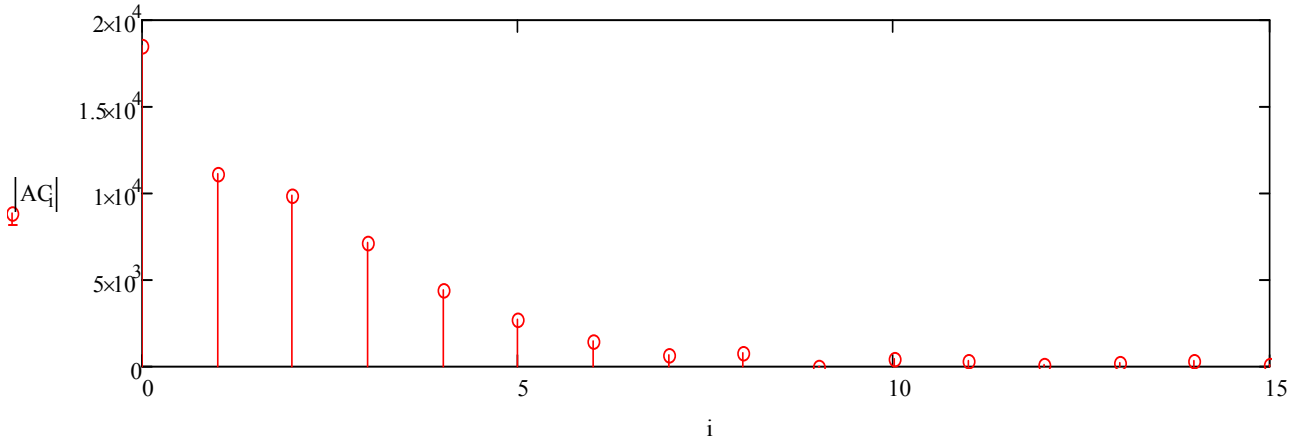
$$\text{SegC} := \left| \begin{array}{l} \text{for } i \in 0..3 \\ \quad \text{SegC}_i \leftarrow \text{submatrix}[S, i \cdot 16384, (i+1) \cdot 16384 - 1, 0, 0] \\ \text{SegC} \end{array} \right.$$

Обчислюємо коефіцієнт кепстра для всього контейнера

$$F := \text{CFFT}(C)$$

$$L := \log\left(\vec{F}\right)^2$$

$$AC := \text{ICFFT}(L)$$



Вибираємо дві точки для вбудовування і загасання exo-сигналів

$$\tau 1 := 7 \qquad p1 := 2$$

$$\tau 2 := 14 \qquad p2 := 2$$

$$\text{Exo1} := \left| \begin{array}{l} \text{for } i \in 0..3 \\ \quad \left| \begin{array}{l} \text{for } j \in 0..\tau 1 - 1 \\ \quad \text{Exo1}_{i\cdot 16384 + j} \leftarrow 0 \\ \text{for } j \in \tau 1..16383 \\ \quad \text{Exo1}_{i\cdot 16384 + j} \leftarrow \text{floor}\left(\frac{C_{i\cdot 16384 + j - \tau 1}}{p1}\right) \end{array} \right. \\ \text{Exo1} \end{array} \right.$$

$$\text{Exo2} := \left| \begin{array}{l} \text{for } i \in 0..3 \\ \quad \left| \begin{array}{l} \text{for } j \in 0..\tau 2 - 1 \\ \quad \text{Exo2}_{i\cdot 16384 + j} \leftarrow 0 \\ \text{for } j \in \tau 2..16383 \\ \quad \text{Exo2}_{i\cdot 16384 + j} \leftarrow \text{floor}\left(\frac{C_{i\cdot 16384 + j - \tau 2}}{p1}\right) \end{array} \right. \\ \text{Exo2} \end{array} \right.$$

$$\text{C\_Sum} := \left| \begin{array}{l} \text{for } i \in 0..3 \\ \quad \text{for } j \in 0..16383 \\ \quad \left| \begin{array}{l} \text{C\_Sum}_{i\cdot 16384 + j} \leftarrow C_{i\cdot 16384 + j} + \text{Exo1}_{i\cdot 16384 + j} \text{ if } m_i = 1 \\ \text{C\_Sum}_{i\cdot 16384 + j} \leftarrow C_{i\cdot 16384 + j} + \text{Exo2}_{i\cdot 16384 + j} \text{ if } m_i = 0 \end{array} \right. \\ \text{C\_Sum} \end{array} \right.$$

AC =

	0
0	$-1.851\cdot 10^4$
1	$1.119\cdot 10^4$
2	$9.856\cdot 10^3$
3	$7.192\cdot 10^3$
4	$4.42\cdot 10^3$
5	$2.754\cdot 10^3$
6	$1.449\cdot 10^3$
7	675.633
8	747.638
9	33.539
10	-472.149
11	-323.131
12	76.357
13	-282.871
14	-316.488
15	...

C =

	0
0	94
1	5
2	-4
3	3
4	1
5	-2
6	0
7	-2
8	-5
9	-8
10	-8
11	-11
12	-14
13	-10
14	-10
15	...

Exo1 =

	0
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	47
8	2
9	-2
10	1
11	0
12	-1
13	0
14	-1
15	...

Exo2 =

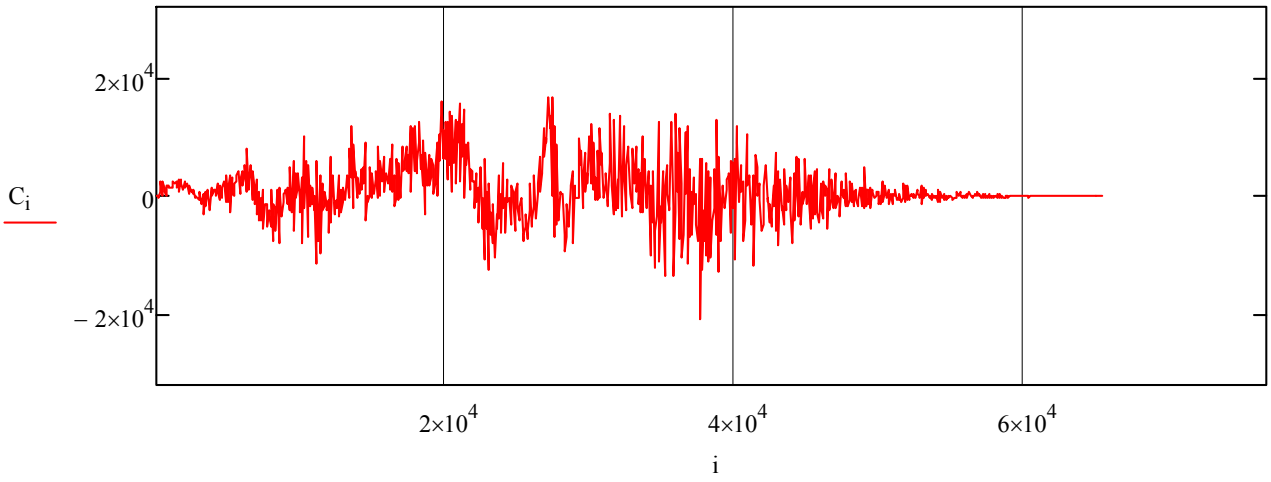
	0
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0
10	0
11	0
12	0
13	0
14	47
15	...

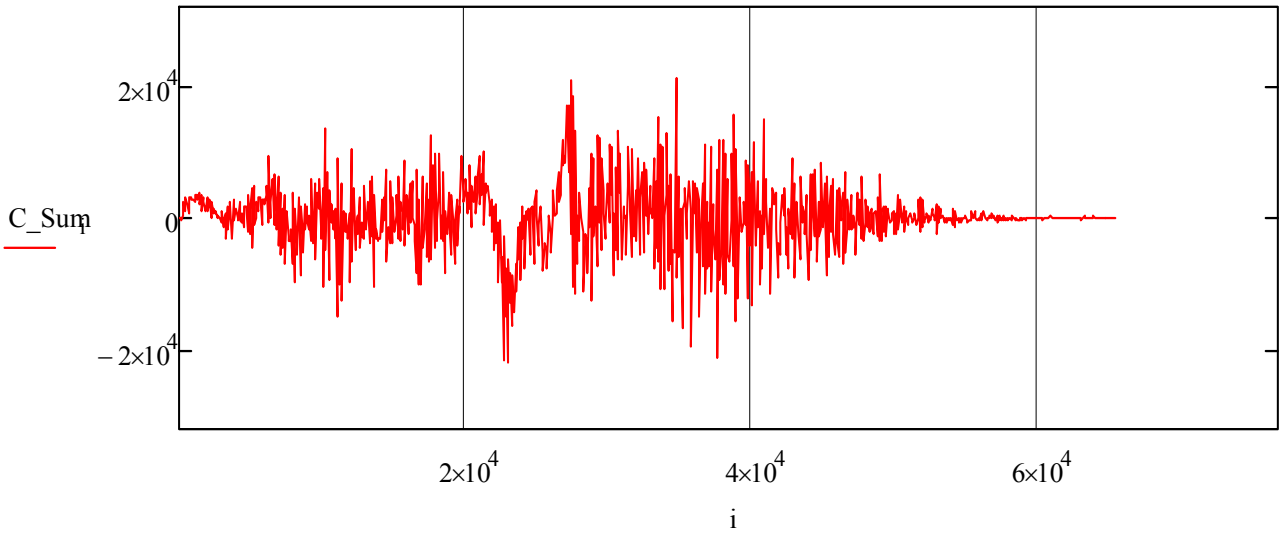
C_Sum =

	0
0	94
1	5
2	-4
3	3
4	1
5	-2
6	0
7	-2
8	-5
9	-8
10	-8
11	-11
12	-14
13	-10
14	37
15	...

m =

	0
0	0
1	0
2	0
3	1
4	0
5	0
6	1
7	1
8	1
9	0
10	1
11	1
12	0
13	1
14	1
15	...





Записуємо звуковий файл

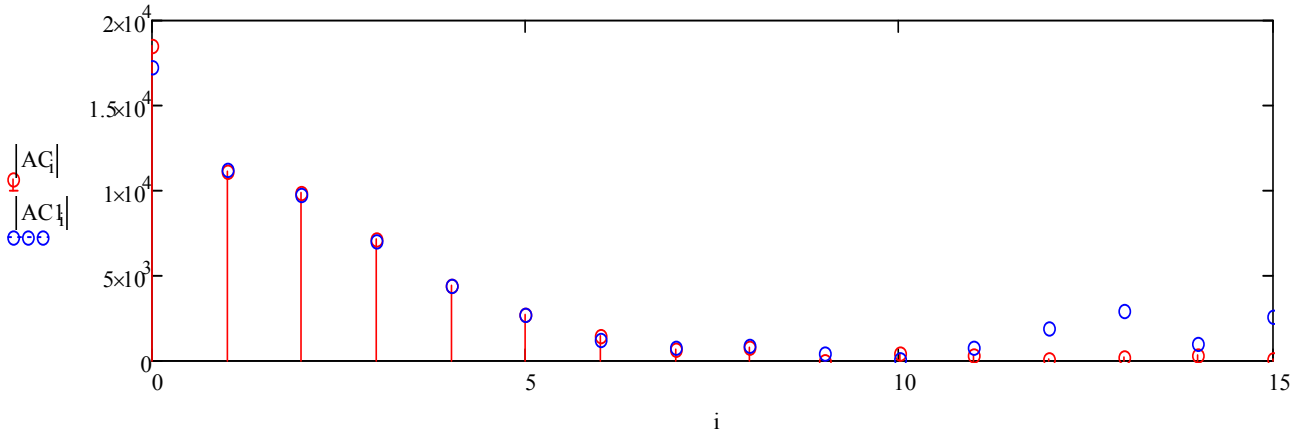
$$WRITEWAV("Stego\_Exo.wav"\ ,fd\ ,Q)\ :=\ C\_Sum$$

Обчислюємо коефіцієнт кепстра для кожного підблоку і витягаємо інформаційні біти

$$F1\ :=\ CFFT(C\_Sum)$$

$$L1\ :=\ \log\left(\overrightarrow{F1}\right)^2$$

$$AC1\ :=\ ICFFT(L1)$$

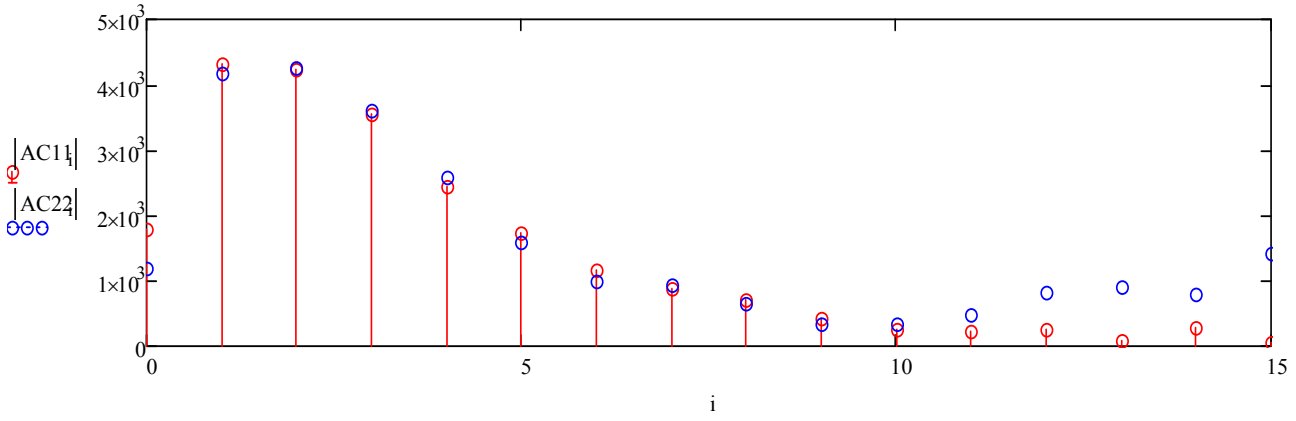


$$\text{SegC\_Sum} := \left| \begin{array}{l} \text{for } i \in 0..3 \\ \quad \text{SegC\_Sum}_i \leftarrow \text{submatrix}[\text{C\_Sum}, i\cdot 16384, (i+1)\cdot 16384 - 1, 0, 0] \\ \text{SegC\_Sum} \end{array} \right.$$

$$F11 := \text{CFFT}\left(\text{SegC}_0\right) \qquad F22 := \text{CFFT}\left(\text{SegC\_Sum}_0\right)$$

$$L11 := \log\left(\overrightarrow{F11}\right)^2 \qquad L22 := \log\left(\overrightarrow{F22}\right)^2$$

$$AC11 := \text{ICFFT}(L11)$$



Задамо функцію автокореляції (періодичну і аперіодичну)

$$RP(S,e) := \frac{\sum_{i=0}^{\text{rows}(S)-1} \left(S_i \cdot S_{\text{mod}(i+e,\text{rows}(S))}\right)}{\text{rows}(S)} \qquad RA(S,e) := \frac{\sum_{i=0}^{\text{rows}(S)-1-e} \left(S_i \cdot S_{i+e}\right)}{\text{rows}(S)}$$

Обчислимо коефіцієнт кепстра для кожного блоку

$$ac := \left| \begin{array}{l} \text{for } i \in 0..\text{rows}(\text{SegC\_Sum}) - 1 \\ \quad ac_i \leftarrow AC\left(\text{SegC\_Sum}_i\right) \\ ac \end{array} \right. \qquad mt := \left| \begin{array}{l} \text{for } i \in 0..20 \\ \quad mt_i \leftarrow \left|RP\left(ac_0, i\right)\right| \\ mt \end{array} \right.$$

Обчислимо коефіцієнт автокореляції для кожного блоку і винесемо вбудовані біти

$$m2 := \left| \begin{array}{l} \text{for } i \in 0..\text{rows}(ac) - 1 \\ \quad \left| \begin{array}{l} a \leftarrow RP\left(ac_i, \tau 1\right) \\ b \leftarrow RP\left(ac_i, \tau 2\right) \\ m_i \leftarrow 0 \text{ if } |a| \le |b| \\ m_i \leftarrow 1 \text{ if } |a| > |b| \end{array} \right. \\ m \end{array} \right.$$

$$m2 = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} \qquad m =$$

	0
0	1
1	1
2	0
3	0
4	0
5	...

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. И. П. Голямина. Звук // Физическая энциклопедия: в 5 т. / гл. ред. А. М. Прохоров. – М. Советская энциклопедия (тт. 1–2); Большая Российская энциклопедия (тт. 3–5), 1988–1999.
2. W. Bender, D. Gruhl, N. Morimoto, A. Lu. Techniques for Data Hiding. IBM Systems Journal, 35 (3&4) pp. 313–336, 1996.
3. Конахович Г. Ф., Пузиренко О. Ю. Компьютерная стеганография. – К. «МК-Пресс», 2006 – 288 с.
4. WAVE PCM soundfile format. 2003-01-20. Archived from the original on 2009-08-27. [Електронний ресурс]. – Режим доступу : https://web.archive.org/web/20090827003349/http://ccrma.stanford.edu/courses/422/projects/WaveFormat/.
5. Формат WAVE файлов. [Електронний ресурс]. – Режим доступу : http://alexei-s1.narod.ru/WAVE.htm.
6. Структура WAV файла. [Електронний ресурс]. – Режим доступу : http://audiocoding.ru/article/2008/05/22/wav-file-structure.html.
7. Wave File Format – формат звукового файла WAV. [Електронний ресурс]. – Режим доступу : http://microsin.net/programming/pc/wav-format.html
8. RIFF WAVE (WAV) file format. [Електронний ресурс]. – Режим доступу : http://www.textfiles.com/programming/FORMATS/riffform.txt.

Навчальне видання

Кузнецов Олександр Олександрович
Полуяненко Микола Олександрович
Кузнецова Тетяна Юріївна

ПРИХОВУВАННЯ ДАНИХ В АУДІОКОНТЕЙНЕРАХ

Методичні рекомендації
до лабораторних робіт з дисципліни «Стеганографія»

Коректор *Б. О. Хільська*
Комп'ютерне верстання *В. В. Савінкова*
Макет обкладинки *І. М. Дончик*

Формат 60 х 84/16. Ум. друк. арк. 3,14. Наклад 50 пр. Зам № 140/19.

Видавець і виготовлювач
Харківський національний університет імені В. Н. Каразіна,
61022, м. Харків, майдан Свободи, 4.
Свідоцтво суб'єкта видавничої справи ДК № 3367 від 13.01.2009

Видавництво ХНУ імені В. Н. Каразіна
Тел. 705-24-32