ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

АКФ– автокореляційна функція

АС – автоматизована система

БЧХ – код Боуза-Чоудхурі-Хокенгема

ВЧ – високочастотний (сигнал)

ГПВП – генератор псевдовипадкової перестановки

ГПВФ – генератор псевдовипадкової функції

ДКП – дискретне косинусне перетворення

ДПФ– дискретне перетворення Фур'є

ЗСЛ – зорова система людини

ІКМ – імпульсно-кодова модуляція

КВЗ – канал відкритого зв'язку

КПЗ – канал прихованого зв'язку

КППД – канал передачі приховуваних даних

КС **–** комп'ютерна стеганографія

ЛРЗЗЗ – лінійний регістр зсуву із зворотнім зв'язком

МСЕ– Міжнародний союз електрозв'язку

НЗБ – найменший значущий біт

НЧ – низькочастотний (сигнал)

ПВП – псевдовипадкова послідовність

ПВЧ – псевдовипадкове число

ПЗ – пропускна здатність

ПКЛ– перетворення Карунена-Лоева

ППЗ – прихована пропускна здатність

РС – розширення спектру

РСПП – розширення спектру сигналу прямою послідовністю

ССЛ – слухова система людини

ЦВЗ – цифровий водяний знак

ЦОС– цифрова обробка сигналів

ЦС – цифрова стеганографія

ШПФ – швидке перетворення Фур'є

ASCII – американський стандартний код для обміну інформацією (*American Standard Code for Information Interchange*). Набір з 128 кодів символів для машинного подання прописних і рядкових літер латинського алфавіту, чисел, розділових знаків і спеціальних символів, кожному з яких відповідає конкретне 7-бітове двійкове число. Перші 32 символи даного коду є керуючими (такими, як символи "переведення рядка", "повернення каретки") і слугують для керування друком і передачею даних. Вони не можуть бути виведені у текстовому вигляді. 8-й біт при передачі даних може використовуватися для контролю парності або для розширеного набору символів ASCII (*extended ASCII*), що включає літери різних мов і графічні символи

BMP – бітове (растрове) відображення графічного об'єкта (*BitMaP*), використовується для представлення зображень. Стандартний формат графічних файлів, який передбачає 4, 8 і 24 біти квантування на один піксель

CR – службовий ASCII-код, який позначає операцію повернення курсору (каретки) – переведення його до лівого краю листа при виведенні тексту на символьний пристрій (*Carriage Return*)

DCT – дискретне косинусне перетворення (*Discrete Cosine Transform*), математичне перетворення, використовуване в алгоритмах компресії зображень, наприклад, у JPEG

FDCT – пряме дискретне косинусне перетворення (*Forward Discrete Cosine Transform*)

GIF – формат обміну графічними даними (*Graphics Interchange Format*), розроблений інформаційною службою CompuServe у 1987 р. для ефективного пересилання графіки (GIF87a). Широко використовується для зберігання простих растрових зображень, що містять великі поля одного кольору

IDCT – зворотне дискретне косинусне перетворення (*Inverse Discrete Cosine Transform*)

JPEG – стандарт на компресії із втратами повноколірних нерухомих відеозображень на основі алгоритму дискретного косинусного перетворення з коефіцієнтом компресії даних більше 25:1. Розроблений групою експертів з машинної обробки фотографічних зображень (*Joint Photographic Experts Group*). Блоки зображення 8×8 пікселів, що не перетинаються обробляються з використанням цілочисельної арифметики

LF – службовий ASCII-код, який викликає переведення курсору на екрані до тієї ж самої колонки на один рядок нижче (*Line Feed*)

LFSR – лінійний регістр зсуву із зворотнім зв’язком (*Linear Feedback Shift Register*)

LSB – молодший значущий біт (розряд) двійкового числа (*Least Significant Bit*)

MSB – старший значущий біт (розряд) двійкового числа (*Most Significant Bit*)

PCM – імпульсно-кодова модуляція (*Pulse Code Modulation*)

RGB – основна палітра – “червоний, зелений, синій” (*Red-Green-Blue*), що використовується в програмуванні та комп'ютерній графіці

SS – розширення спектру сигналів (*Spread-Spectrum*)

VPN – віртуальна приватна мережа (*Virtual Private Network*), підмережа корпоративної мережі, яка забезпечує безпечне входження в неї віддалених користувачів. Використовується для безпечного пересилання Інтернетом конфіденційних даних за рахунок інкапсуляції (тунелювання) IP-пакетів всередині інших пакетів, які потім маршрутизуються

XOR – виключаюче АБО (нееквівалентність, додавання за модулем 2) (*eXclusive OR*) – бінарна логічна операція, результат якої є істинним лише тоді, коли значення операндів не співпадають.

ВСТУП

Інформація є однією з найцінніших речей сучасного життя. Отримання доступу до неї з появою глобальних комп'ютерних мереж стало надзвичайно простим. Але легкість і швидкість такого доступу зробили значними й загрози безпеці даних за відсутності заходів щодо їх захисту, зокрема – загрози неавторизованого доступу до інформації.

Задача надійного захисту авторських прав, прав інтелектуальної власності або конфіденційних даних (які здебільшого мають цифровий формат) від несанкціонованого доступу є однією з найстаріших і не вирішених на сьогодні проблем. У зв'язку з інтенсивним розвитком і поширенням технологій, які дозволяють за допомогою комп'ютера інтегрувати, обробляти і синхронно відтворювати різноманітні типи сигналів (так звані мультимедійні технології), питання захисту інформації, представленої у цифровому вигляді, є надзвичайно актуальними. Переваги представлення і передачі даних у цифровому вигляді (легкість відновлення, висока потенційна завадостійкість, перспективи використання універсальних апаратних і програмних рішень) можуть бути перекреслені з легкістю, з якою можливі їх викрадення і модифікація. Тому в усьому світі назрілим є питання розробки методів (заходів) захисту інформації організаційного, методологічного і технічного характеру, серед них – методи криптографії і стеганографії.

Криптографічний (з грецької *χρυπτός* – таємний, *γράφω* – пишу) захист інформації (система зміни останньої з метою зробити її незрозумілою для непосвячених, приховання *змісту* повідомлень за рахунок їх шифрування) не знімає зазначену вище проблему повністю, оскільки наявність шифрованого повідомлення сама по собі привертає увагу і зловмисник, заволодівши захищеним криптографічно файлом, відразу розуміє про розміщення в ньому секретної інформації, і всю сумарну міць своєї комп'ютерної мережі переводить на дешифрування даних.

Приховування ж самого *факту існування* секретних даних при їх передачі, зберіганні чи обробці є задачею ***стеганографії*** (з грецької *στεγανός* – прихований) – науки, яка вивчає способи і методи приховання конфіденційних відомостей. Задача видобування інформації при цьому відступає на другий план і розв'язується в більшості випадків стандартними криптографічними методами. Інакше кажучи, під приховуванням існування розуміється не лише унеможливлення виявлення в перехопленому повідомленні наявності іншого (прихованого) повідомлення, але й взагалі зробити неможливим викликання підозр на цей рахунок, оскільки в останньому випадку проблема інформаційної безпеки повертається до стійкості криптографічного коду. Таким чином, займаючи свою нішу в забезпеченні безпеки, стеганографія не замінює, а доповнює криптографію [1]. Стеганографування здійснюється найрізноманітнішими способами. Загальною ж рисою цих способів є те, що приховуване повідомлення вбудовується в деякий не приваблюючий увагу об'єкт, який згодом відкрито транспортується (пересилається) адресату.

Історично напрямок стеганографічного приховування інформації був першим [2], але згодом у багато чому був витіснений криптографією. Інтерес до стеганографії відродився в останнє десятиріччя і був викликаний широким розповсюдженням технологій мультимедіа (що є цілком закономірним, з огляду на зазначені вище проблеми, пов'язані із захистом інформації). Не менш важливою стала поява нових типів каналів передачі інформації, що в сукупності з першим фактором надало нового імпульсу розвиткові та удосконаленню стеганографії, сприяло виникненню нових стеганографічних методів, в основу яких було закладено особливості представлення інформації в комп'ютерних файлах, обчислювальних мережах і т.д. Це, в свою чергу, дає можливість казати про становлення нового напрямку в галузі захисту інформації – ***комп'ютерної стеганографії*** (КС) [3-5,19].

З 1996 р. проводяться міжнародні симпозіуми з проблем приховання даних (*Information Workshop on Information Hiding*). Перша конференція, присвячена стеганографії, відбулася у липні 2002 р. На сьогодні стеганографія є наукою, яка швидко і динамічно розвивається, використовуючи при цьому методи і досягнення криптографії, цифрової обробки сигналів, теорії зв'язку та інформації.

Методи стеганографії дозволяють не лише приховано передавати дані (так звана *класична стеганографія*), але й успішно вирішувати задачі завадостійкої аутентифікації, захисту інформації від несанкціонованого копіювання, відстеження поширення інформації мережами зв'язку, пошуку інформації в мультимедійних базах даних тощо. Ці обставини дозволяють в

межах традиційно існуючих інформаційних потоків чи інформаційного середовища вирішувати деякі важливі питання захисту інформації низки прикладних галузей.

Існують два ключових напрямки використання КС: *пов'язаний* з цифровою обробкою сигналів (ЦОС) і *не пов'язаний*. У першому випадку секретні повідомлення вбудовуються у цифрові дані, які, як правило, мають аналогову природу (мова, зображення, аудіо- і відеозаписи) [1,3-5]. У другому — конфіденційна інформація розміщується в заголовках файлів чи пакетів даних. Але цей напрямок не знайшов широкого застосування через відносну легкість розкриття і/або знищення прихованої інформації. Переважна більшість поточних досліджень в галузі стеганографії так або інакше пов'язана саме з ЦОС. Що дозволяє говорити про ***цифрову стеганографію*** (ЦС) [5,19].

Можна виділити щонайменше дві причини популярності в наш час досліджень в галузі стеганографії: обмеження на використання криптографічних засобів в низці країн світу і поява проблеми захисту прав власності на інформацію, представлену у цифровому вигляді.

Перша причина спричинила велику кількість досліджень у дусі класичної стеганографії (тобто приховання власне факту передачі), друга – не менш чисельні роботи в галузі так званих *цифрових водяних знаків* (ЦВЗ) – спеціальних міток, приховано вбудовуваних у зображення (або інші цифрові дані) з метою можливості контролювання його використання.

Приховування інформації лише завдяки факту невідомості зловмиснику методу або методів, закладених до основи приховання, є на сьогоднішній день малоефективним. Ще у 1883 р. фламандський криптограф *A. Kerckhoffs* писав про те, що система захисту інформації повинна виконувати функції, на неї покладені, навіть за повної інформованості супротивника про її структуру і алгоритми функціонування [6]. Вся секретність системи захисту повідомлень, що передаються, повинна міститися в ключі, тобто в попередньо (як правило) розділеному між адресатами фрагменті інформації. Незважаючи на те, що цей принцип відомий вже більше 100 років, досі зустрічаються розробки, що ними зневажають. Очевидно, що вони не можуть застосовуватися з серйозною метою.

В основі багатьох підходів до вирішення задач стеганографії лежить загальна з криптографією методична база, яку заклав ще всередині минулого століття *C.E. Shannon* [45,60]. Але й дотепер теоретичні основи стеганографії залишаються практично неопрацьованими.

Беручі до уваги вищесказане, можна зробити висновок, що на сьогодні існує актуальна науково-технічна проблема удосконалення алгоритмів і методів проведення стеганографічного приховування конфіденційних даних або захисту авторських прав на певну інформацію. На сьогодні не бракує стеганографічних програм як початкового, так і професійного рівня (*S-Tools, Steganos Security Suite, bmpPacker* та ін.). Але захищеність їх коду (особливо це стосується програм професійного рівня) не дозволяє простежити методи, закладені в основу алгоритмів їх дії. Викладені ж на Internet- ресурсах численні тексти програм мало чим зараджують через їх низьку інформативність. Компіляція пропонованих текстів має своїм результатом виконувальну програму, алгоритм дії якої неможливо простежити, оскільки остання видає вже готовий результат – заповнений стеганоконтейнер, і практично не існує можливості заздалегідь встановити достатність рівня прихованості конфіденційної інформації у цьому контейнері. Отже, є цілком очевидною є нестача саме *програм навчального рівня*, які б наочно демонстрували весь процес стеганографічного перетворення крок за кроком, що можна було б використати в учбовому процесі при підготовці фахівців в галузі захисту інформації.

Стан порушеного питання в області стеганографії характеризується наступними основними досягненнями. Питання стеганографічного приховування секретних відомостей, включаючи побудову ефективних алгоритмів приховування, свого часу розглядали в своїх роботах *G.J. Simmons, J. Fridrich, R.J. Anderson, W. Bender, N. Morimoto, C. Cachin, I. Pitas* та ін. [7-9,13-16]. Результати досліджень стеганографічних алгоритмів на стійкість приводять в своїх працях *J. Fridrich, R. Popa, N.F. Johnson, S. Voloshynovskiy* [9,17,18,20,40,41]. Також необхідно відзначити праці авторів *B. Pfitzmann, B. Schneier* і *S. Craver* з питань узгодження термінології та формування основних стеганографічних протоколів [10-12].

Тривалий час у вітчизняній літературі і літературі країн СНД стеганографії було присвячено лише декілька оглядових журнальних статей [1,4,22-24,48]. Крім того, заслуговує на увагу робота [5] авторів *Грибунін В.Г., Оков І.М., Туринцев І.В.* В останній час відсутність

наукової літератури зазначеної тематики вітчизняних авторів певною мірою ліквідовано такими фахівцями як *В.О. Хорошко, О.Д. Азаров, М.Є. Шелест* та ін. [3], заслугою яких є чи не перша спроба системного викладення стеганографічних методів, узагальнення найостанніших результатів досліджень в галузі комп'ютерної стеганографії.

Метою даної роботи є викладення теоретичних і, що не менш важливо, практичних основ комп'ютерної стеганографії. Розглянуто особливості і перспективи використання сучасної системи символьної математики MathCAD v.12 в цілях стеганографічного захисту інформації.

Проведено аналіз спеціалізованих літературних джерел та ресурсів мережі Internet щодо перспективних напрямків, за якими можливе використання стеганографії як інструменту захисту інформації в автоматизованих системах обробки даних.

Шляхом дослідження відомих публікацій вітчизняних і закордонних авторів здійснено системне викладення проблем надійності і стійкості довільної стеганографічної системи по відношенню до видів здійснюваних на неї атак, а також оцінки пропускної здатності каналу прихованого обміну даними, яким, по суті, є стеганосистема. Наведено результати існуючих інформаційно-теоретичних досліджень проблеми інформаційного приховання у випадку активної протидії порушника.

Здійснено системне викладення відомих стеганографічних методів, спрямованих на приховання конфіденційних даних у комп'ютерних файлах графічного, звукового і текстового форматів.

Наведено приклади програмних комплексів для демонстрації принципів, закладених в основу методів стеганографічного приховування інформації в просторовій (часовій) або частотній областях використовуваного контейнера (нерухомого зображення, аудіосигналу або текстового документу).

Використання під час комп'ютерного моделювання універсальної математичної системи MathCAD v.12 дозволяє використати потужні засоби реалізації чисельних методів розрахунку і математичного моделювання в поєднанні з можливістю виконання операцій символьної математики [25,26]. Сторони, що здійснюють прихований обмін даними, практично позбавляються необхідності у програмуванні власне розв'язку задач, на них лише покладається коректний опис алгоритму розв'язку на вхідній мові MathCAD, яка є мовою дуже високого рівня. Зазначене є суттєвою перевагою у порівнянні з існуючими на сьогодні програмами, написаними за допомогою таких низькорівневих мов як C/C++, Basic та візуальних інтерфейсів на їх основі. Останні, хоча і відрізняються достатньо високим рівнем гнучкості з точки зору можливостей реалізації тих або інших методів стеганографії, проте характеризуються незрівнянно тривалим внесенням змін до вже написаної і скомпільованої програми. Час, затрачуваний на внесення модифікацій, стає особливо важливим у випадку багатоетапних досліджень, що мають місце при використанні програми в навчальному процесі.

Завдяки своїй наочності та можливості швидкого проведення модифікацій програмних модулів, розроблені комплекси відповідають вимогам, що ставляться до програм, використовуваним у навчальних цілях. Підхід поєднання теоретичного викладення матеріалу з демонстрацією його практичного використання дозволяє позбавитися абстрактності формулювань, прийнятої у спеціалізованій і довідковій літературі з інформаційної безпеки, і сприяє розвитку у студентів здорового інтересу до практичних аспектів вирішення науково-технічних задач із захисту інформації. Книга може використовуватися в якості короткого довідникового посібника з питань комп'ютерної стеганографії при використанні сучасних комп'ютерно-математичних систем.

Для викладачів і студентів ВНЗ і університетів, що навчаються з напрямку підготовки "Інформаційна безпека", а також для спеціалістів, які працюють в галузі захисту інформації і зацікавлені в ефективному використанні можливостей сучасних обчислювальних систем.

1. МІСЦЕ СТЕГАНОГРАФІЧНИХ СИСТЕМ В ГАЛУЗІ ІНФОРМАЦІЙНОЇ БЕЗПЕКИ

1.1. Основні джерела і наслідки атак на інформацію, що обробляється в автоматизованих системах

Рівень розвитку і безпека інформаційного простору, які є системоутворюючими факторами в усіх сферах національної безпеки, активно впливають на стан політичної, економічної, оборонної та інших складових національної безпеки України.

Інформаційна безпека, захист якої відповідно до ст.17 Конституції України, поряд із суверенітетом, територіальною цілісністю та економічною безпекою, є найважливішою функцією держави, досягається шляхом розробки сучасного законодавства, впровадження сучасних безпечних інформаційних технологій, побудовою функціонально повної національної інфраструктури, формуванням і розвитком інформаційних відносин тощо.

Закон України "Про захист інформації в інформаційно-телекомунікаційних системах" [33] дає наступне визначення терміну *"захист інформації"* в інформаційних, телекомунікаційних та інформаційно-телекомунікаційних системах – це діяльність, спрямована на запобігання несанкціонованим діям щодо інформації в зазначених системах. Окремим видом захисту інформації є *технічний захист*, спрямований на забезпечення за допомогою інженерно-технічних заходів та/або програмних і технічних засобів унеможливлення витоку, знищення та блокування (унеможливлення доступу) інформації, порушення цілісності (зміни вмісту) та режиму доступу до інформації. *Доступом до інформації* є отримання користувачем (фізичною або юридичною особою) можливості обробляти інформацію в системі. Дії, що провадяться з порушенням порядку доступу до інформації, установленого відповідно до законодавства, є *несанкціонованими*.

Неоднозначним, на думку фахівців, є трактування поняття "атака на інформацію", оскільки остання, особливо в електронному вигляді, може бути представлена сотнями різноманітних видів. Інформацією можна вважати і окремий файл, і базу даних, і лише один запис в ній, і цілий програмний комплекс. На сьогодні всі ці об'єкти піддаються або можуть бути піддані атакам з боку деякої особи (групи осіб) – *порушника*.

При зберіганні, підтримці та наданні доступу до будь-якого інформаційного об'єкта його власник, або уповноважена ним особа, накладає (явно або самоочевидно) набір правил по роботі з нею. Навмисне їх порушення і класифікується як *атака на інформацію* [34-37].

З масовою комп'ютеризацією усіх сфер діяльності людини обсяг інформації, яка зберігається в електронному вигляді, зріс у тисячі разів. Це, в свою чергу, значно підвищило ризик *витоку інформації*, в результаті чого остання стає відомою (доступною) суб'єктам, що не мають права доступу до неї. З появою комп'ютерних мереж навіть відсутність фізичного доступу до комп'ютера перестала бути гарантією збереженості інформації.

Розглянемо можливі наслідки атак на інформацію, у першу чергу – *економічні втрати:*

а) розкриття комерційної інформації може призвести до значних прямих збитків;

б) відомості про викрадення великого об'єму інформації суттєво впливає на репутацію організації, опосередковано призводячи до втрат в обсягах торгових операцій;

в) якщо викрадення залишилося непоміченим, конкуренти можуть скористатися цим з метою повного розорення організації, нав'язуючи тій фіктивні або збиткові угоди;

г) до збитків може призвести і проста підміна інформації як на етапі її передачі, так і на етапі зберігання всередині організації.

Цілком очевидно, що атаки на інформацію можуть принести й величезну *моральну шкоду* з огляду на можливу конфіденційність інформації.

За даними дослідницької групи *CNews Analytics*, яка спеціалізується на дослідженнях ринків інформаційних технологій і телекомунікацій, глобальний економічний збиток від комп'ютерних злочинів, що так або інакше пов'язані з атакою на інформацію, у 2002 р. склав близько 49,2 млрд.$, а в 2003 р. лише за серпень – 8,23 млрд.$ (16160 атак). У 2004 р., вважають аналітики

цієї ж групи, сумарні економічні втрати склали від 300 до 400 млрд.$ [38]. З огляду на загальну тенденцію можна зробити висновок про зростання кількості комп'ютерних злочинів і в Україні.

Дослідницький центр *DataPro Research Corp.* [39], наводить таку картину розподілу основних причин втрати чи пошкодження інформації в комп'ютерних мережах: 52% – ненавмисні дії персоналу, 10% – навмисні дії персоналу, 10% – відмова обладнання, 15% – пожежі, 10% – затоплення. Що стосується умисних дій, то за даними того ж дослідження, головний мотив комп'ютерних злочинів – викрадення грошей з електронних рахунків (44%), далі йдуть викрадення секретної інформації (16%), пошкодження програмного забезпечення (16%), фальсифікація інформації (12%), замовлення послуг за чужий рахунок (10%). Серед тих, хто був виконавцем зазначених дій, – поточний кадровий склад установ (81%), сторонні особи (13%), колишні працівники цих самих установ (6%).

Зрозуміло, що в такій ситуації особлива увага повинна приділятися створенню інформаційних систем, захищених від різноманітних загроз. Але при цьому в ході проектування і створення таких систем виникає ціла низка проблем, основними з яких є: складність інтеграції певних функцій безпеки в елементи архітектури системи; складність формування вичерпного набору необхідних і достатніх вимог безпеки; відсутність загальноприйнятих методів проектування систем безпеки.

Особливостями проектування архітектури системи забезпечення безпеки інформації, як, мабуть, і інших складних багатофакторних систем, є множинність параметрів, які треба враховувати і які важко зафіксувати, постійне зростання кількості загроз інформаційній безпеці. При цьому більшість існуючих рішень, наприклад таких як екранування і VPN, враховують лише частину проблем забезпечення безпеки інформації, а використовувані методи в основному зводяться до застосування визначеного переліку продуктів.

1.2. Категорії інформаційної безпеки з позицій захисту автоматизованих систем від несанкціонованого доступу

Інформація з точки зору інформаційної безпеки характеризується наступними категоріями:

конфіденційність – гарантія того, що конкретна інформація є доступною лише тому колу осіб, для якого вона призначена; порушення цієї категорії є *викраденням* або *розкриттям інформації*;

цілісність – гарантія того, що на даний момент інформація існує в її початковому вигляді, тобто при її зберіганні чи передаванні не було зроблено несанкціонованих змін; порушення даної категорії зветься *фальсифікацією повідомлення*;

автентичність – гарантія того, що джерелом інформації є саме той суб'єкт, який заявлений як її автор; порушення цієї категорії зветься *фальсифікацією автора повідомлення*;

апельованість – гарантія того, що в разі необхідності можна бути довести, що автором повідомлення є саме заявлений суб'єкт і ніхто інший; відмінність даної категорії від попередньої в тому, що при підміні автора, хтось інший намагається заявити, що він є автором повідомлення, а при порушенні апельованості – сам автор намагається уникнути відповідальності за видане ним повідомлення.

Стосовно інформаційних систем використовуються інші категорії:

надійність – гарантія того, що система вестиме себе заплановано як в нормальному, так і у позаштатному режимах;

точність – гарантія точного і повного виконання всіх команд;

контроль доступу – гарантія того, що різні групи осіб мають різний доступ до інформаційних об'єктів, і ці обмеження доступу постійно виконуються;

контрольованість – гарантія того, що у будь-який момент може бути виконана повноцінна перевірка будь-якого компоненту програмного комплексу;

контроль ідентифікації – гарантія того, що клієнт (адресат) є саме тим, за кого себе видає;

стійкість до навмисних збоїв – гарантія того, що при навмисному внесенні помилок в межах заздалегідь обговорених норм система вестиме себе прогнозовано.

1.3. Можливі варіанти захисту інформації в автоматизованих системах

Серед можливих методів (заходів) захисту конфіденційної інформації найпоширенішим на сьогодні є метод ***криптографічного захисту***, під яким розуміється приховання змісту повідомлення за рахунок його *шифрування* (*кодування*) за певним алгоритмом, що має на меті зробити повідомлення незрозумілим для непосвячених в цей алгоритм або у зміст ключа, який використовувався при шифруванні. Але зазначений метод захисту є неефективним щонайменше з двох причин.

По-перше, зашифрована за допомогою більш-менш стійкої криптосистеми інформація є недоступною (протягом часу, що визначається стійкістю криптосистеми) для ознайомлення без знання алгоритму і ключа. Наслідком цього стало те, що силові структури деяких країн застосовують адміністративні санкції проти так званої "стійкої криптографії", обмежуючи використання криптографічних засобів приватними і юридичними особами без відповідної ліцензії.

По-друге, слід звернути увагу на те, що криптографічний захист захищає лише *зміст* конфіденційної інформації. При цьому сама лише наявність шифрованої інформації здатна привернути увагу потенційного зловмисника, який, заволодівши криптографічно захищеним файлом і здогадуючись про розміщення в ньому секретних даних, за певної мотивації здатен перевести всю сумарну міць підконтрольної йому комп'ютерної мережі на дешифрування цих даних. У цьому випадку проблема інформаційної безпеки повертається до стійкості криптографічного коду.

На противагу вищезазначеному, ***стеганографічний захист*** забезпечує приховання самого *факту існування* конфіденційних відомостей при їх передачі, зберіганні чи обробці. Під приховуванням факту існування розуміється не тільки унеможливлення виявлення в перехопленому повідомленні наявності іншого (прихованого) повідомлення, але й взагалі зробити неможливим викликання на цей рахунок будь-яких підозр. Задача неможливості неавторизованого видобування інформації при цьому відступає на другий план і розв'язується у більшості випадків додатковим використанням стандартних криптографічних методів. Загальною рисою стеганографічних методів є те, що приховуване повідомлення вбудовується в деякий не приваблюючий увагу об'єкт (контейнер), який згодом відкрито транспортується (пересилається) адресату.

1.4. Проблеми і задачі роботи

Завданням пропонованої роботи є розробка програмного комплексу для демонстрації принципів, закладених в основу поширених на сьогодні методів стеганографічного приховування з можливістю обчислення основних показників спотворення контейнера при вбудовуванні до нього приховуваних даних.

Дана задача вирішується попереднім опрацюванням наступних питань:

– розгляд особливостей побудови стеганографічних систем та основних типів атак на зазначені системи;

– аналіз сучасних досліджень і публікацій з приводу існуючих методів стеганографії та заходів підвищення їх стійкості до стеганоаналізу і пропускної здатності;

– формулювання практичних рекомендацій стосовно вбудовування даних.

2. ОСОБЛИВОСТІ ПОБУДОВИ СТЕГАНОГРАФІЧНИХ СИСТЕМ

2.1. Предмет, термінологія і області застосування стеганографії

В процесі дослідження стеганографії, стає очевидним, що вона, по суті, не є чимось новим. Задачі захисту інформації від неавторизованого доступу тим або іншим способом вирішувалися протягом всієї історії людства [2].

В останнє десятиріччя, завдяки масовому поширенню мультимедійних технологій і засобів телекомунікацій розвиток стеганографії вийшов на принципово новий етап, який фахівці називають *комп'ютерною стеганографією* (КС). Серед основних галузей використання КС: приховання (шляхом вбудовування) повідомлень у цифрових даних, які, як правило, мають аналогову природу (мова, зображення, аудіо- чи відеозаписи). Також в якості контейнерів (або так званих "носіїв") можливе використання текстових файлів або виконувальних файлів програм [1,3-5,19,20]. Так, наприклад, найменш значимі біти цифрового зображення або аудіофайлу можуть бути замінені даними з текстового файлу таким чином, що сторонній незалежний спостерігач не виявить жодної втрати в якості зображення чи звуку [3,9,14,21-23,27]. Отже, скажімо, зображення, викладене на певному Internet-ресурсі для загального користування, потенційно може таємно містити важливу для певних кіл інформацію і при цьому не викликати жодних підозр широкого загалу. Публікації деяких світових ЗМІ після вересневих терактів у США навіть вказували на зазначену техніку приховування як можливий засіб зв'язку між членами терористичних організацій, які планували атаки на знак протесту проти впливу Заходу на світовий устрій [28-30].

Незважаючи на численні відкриті публікації та щорічні конференції, тривалий час стеганографія не мала усталеної термінології. З середини 80-х р.р. минулого століття для опису моделі стеганографічної системи (скорочено – *стегосистеми* або, що на думку автора цієї роботи є більш правильним, *стеганосистеми*, оскільки приставка "стего" у перекладі з латини означає "дах" або "черепиця") використовувалася так звана "проблема ув'язнених", яку запропонував у 1983 р. *G.J. Simmons* [7]. Основні поняття стеганографії були узгоджені у 1996 р. на 1-й Міжнародній конференції з приховування даних – Information Workshop on Information Hiding'96 [10]. Тим не менш навіть таке першоутворююче поняття як "стеганографія" різними спеціалістами трактується неоднаково. Наприклад, деякі фахівці розуміють під стеганографією лише приховану передачу інформації. Інші ж відносять до неї такі додатки як, наприклад, метеорний радіозв'язок, радіозв'язок із псевдовипадковим перестроюванням частоти, широкосмуговий радіозв'язок [31,32]. В роботі [5] приводиться наступне визначення *цифрової стеганографії:* "...наука про непомітне і надійне приховання одних бітових послідовностей в інших, що мають аналогову природу". Згадуванням про аналогову природу цифрових даних підкреслюється факт вбудовування інформації в оцифровані безперервні сигнали. Таким чином, у порівнянні з ЦС *комп'ютерна стеганографія* має більш широкий зміст, оскільки в її межах розглядаються питання введення даних в заголовки IP-пакетів, до текстових повідомлень та файлів інших форматів.

Слово "непомітне" в наведеному вище визначенні цифрової стеганографії розуміє під собою обов'язкове включення людини в систему стеганографічної передачі даних. Тобто людина розглядається як специфічний приймач даних, який пред'являє до системи передачі достатньо важко формалізовані вимоги [5].

Таким чином, *стеганографічна система* або, скорочено, *стеганосистема* – це сукупність засобів і методів, які використовуються з метою формування прихованого (непомітного) каналу передачі інформації [10,22]. Іншими словами, це система, яка виконує задачу вбудовування і виокремлення повідомлень з іншої інформації. Причому процес приховування даних, подібно до процесу компресії (ущільнення), є відмінним від операції шифрування. Його метою є не обмежувати чи регламентувати доступ до сигналу(файлу)-контейнера, а в значній мірі гарантувати, що вбудовані дані залишаться непошкодженими (немодифікованими) і такими, що підлягають відновленню [14].

При побудові стеганосистеми повинні враховуватися наступні положення [3,5,11]:

– стеганосистема повинна мати прийнятну обчислювальну складність реалізації (під обчислювальною складністю розуміється кількість кроків або арифметико-логічних операцій, необхідних для розв'язання обчислювальної проблеми, у даному випадку – процесу вбудовування/видобування приховуваної інформації до/з сигналу контейнера);
– повинна забезпечуватися необхідна пропускна здатність (що є особливо актуальним для стеганосистем прихованої передачі даних);
– методи приховування повинні забезпечувати автентичність і цілісність секретної інформації для авторизованої особи;
– потенційний порушник має повне уявлення про стеганосистему і деталі її реалізації. Єдине, що йому невідоме, – це ключ, за допомогою якого тільки його власник може встановити факт наявності та зміст прихованого повідомлення;
– якщо факт існування прихованого повідомлення стає відомим порушнику, це не повинне дозволити останньому його видобути доти, доки ключ зберігається в таємниці;
– порушник повинен бути позбавлений будь-яких технічних та інших переваг в розпізнаванні або ж, принаймні, розкритті змісту секретних повідомлень.

На думку авторів [5], стеганографія включає в себе наступні напрямки:
– вбудовування інформації з метою її прихованої передачі;
– вбудовування цифрових водяних знаків (ЦВЗ);
– вбудовування ідентифікаційних номерів;
– вбудовування заголовків.

Аналіз інших літературних джерел, зокрема [3,14,16,19-21], та ресурсів мережі *Internet* дозволяє зробити висновок, що на сьогодні стеганосистеми активно застосовуються для розв'язання таких ключових завдань:
– захист конфіденційної інформації від несанкціонованого доступу;
– захист авторського права на інтелектуальну власність;
– подолання систем моніторингу і управління мережними ресурсами;
– "камуфлювання" програмного забезпечення;
– створення прихованих від законного користувача каналів витоку інформації.

Область **захисту конфіденційної інформації від несанкціонованого доступу** є найбільш ефективною при вирішенні проблеми захисту секретної інформації. Наприклад, лише одна секунда оцифрованого стереозвуку з частотою дискретизації 44.1 кГц та рівнем квантування 8 біт за рахунок заміни найменш значимих молодших розрядів чисел, що характеризують відліки рівнів звукового сигналу, на біти приховуваного повідомлення дозволяє приховати близько 11 кБ інформації (при об'ємі аудіофайлу ~ 88.2 кБ). При цьому зміна результуючого рівня аудіосигналу, який відповідає модифікованому відліку, є меншою за 0,8%, що при прослуховуванні не виявляється переважною більшістю людей. Якщо ж звук є 16-бітним, то зміна рівнів взагалі стає меншою за 0,005%.

Іншою важливою задачею стеганографії є **захист авторського права** від так званого "піратства". На комп'ютерні графічні зображення наноситься спеціальна мітка, яка залишається невидимою для людини, але розпізнається спеціалізованим програмним забезпеченням (ПЗ). Даний напрямок призначений не лише для обробки зображень, але й для файлів з аудіо- чи відеоінформацією і повинен забезпечити захист інтелектуальної власності.

Стеганографічні методи, спрямовані на протидію **системам моніторингу і управління мережними ресурсами** промислового шпигунства, дозволяють протидіяти спробам контролю над інформаційним простором при проходженні інформації через сервери управління локальних і глобальних обчислювальних мереж.

Ще однієї областю використання стеганосистем є **камуфлювання ПЗ**. У тих випадках, коли використання ПЗ незареєстрованними користувачами є небажаним, воно може бути закамуфльовано під стандартні універсальні програмні продукти (наприклад, текстові редактори) або приховане у файлах мультимедіа (наприклад, у звуковому супроводі відео).

Як би не відрізнялися напрямки використання стеганографії, вимоги, що при цьому ставляться, у багато чому залишаються незмінними (див. вище). Але існують і відхилення від загальноприйнятих правил. Так, наприклад, відмінність постановки задачі прихованої передачі даних від постановки задачі вбудовування ЦВЗ полягає у тому, що в першому випадку порушник

повинен виявити приховане повідомлення, тоді як у другому випадку його існування не приховується. Більше того, порушник на законних підставах може мати пристрій виявлення ЦВЗ (наприклад, у складі DVD-програвача).
Потенційно можливі сфери використання стеганографії зображено на рис.2.1.

Рис.2.1. Потенційні області застосування стеганографії

2.2. Проблема стійкості стеганографічних систем

Кожне із зазначених вище завдань вимагає певного співвідношення між *стійкістю* вбудованого повідомлення до зовнішніх впливів і *розміром* вбудовуваного повідомлення.
Для більшості сучасних методів, що використовуються для приховання повідомлень у файлах цифрового формату, має місце наступна залежність надійності системи від об'єму вбудовуваних даних (рис.2.2), з якої видно, що збільшення останнього істотно знижує надійність системи.
Таким чином, існує перспектива прийняття оптимального рішення при виборі між кількістю (об'ємом) прихованих даних і ступенем стійкості (прихованості) до можливої модифікації чи аналізу сигнала-контейнера. Шляхом обмеження ступеня погіршення якостей контейнеру, які здатна сприймати людина, при стеганографічній його обробці можна досягти або високого рівня (обсягу) вбудовуваних даних, або високої стійкості до модифікацій (аналізу), але в жодному разі не обох цих показників одночасно, оскільки зростання одного з них неминуче призводить до зменшення іншого. Не дивлячись на те, що дане ствердження математично може бути продемонстроване лише для деяких методів стеганографії (наприклад, для приховування шляхом розширення спектру), очевидно, що воно є справедливим і для інших методів приховування даних [14]. За будь-якого методу, завдяки надлишковості інформації є можливість підвищити ступінь надійності приховання, жертвуючи при цьому пропускною здатністю (об'ємом приховуваних даних). Об'єм вбудованих даних і ступінь модифікації контейнера змінюються від методу до методу. Також є очевидним і той факт, що в залежності від цілей, для яких використовується приховання даних, різними є і вимоги стосовно рівня стійкості системи до модифікації контейнера. Як наслідок, для різних цілей є оптимальним й різні методи стеганографії.

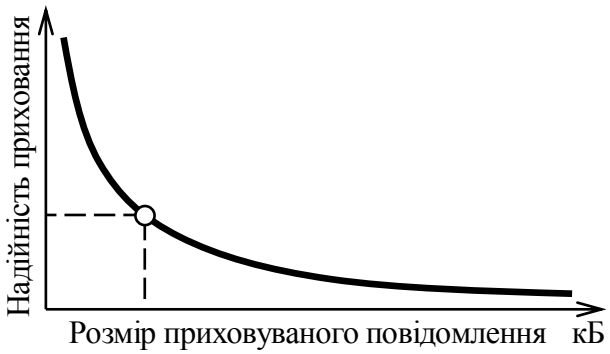


Рис.2.2. Взаємозв'язок між стійкістю стеганосистеми та об'ємом приховуваного повідомлення при незмінному розмірі файлу - контейнера

Розглянемо процес проведення *стеганоаналізу* – оцінки перехопленого контейнера на предмет наявності в ньому прихованого повідомлення. Приховання інформації всередині електронного носія вимагає зміни (перебудови) властивостей останнього, що в тій або іншій мірі призводить до погіршення його характеристик або до набуття цими характеристиками невластивих їм значень. Ці характеристики можуть виконати роль "підписів", що сигналізують про існування вбудованого повідомлення, і, таким чином, основна ідея стеганографії – приховання факту існування секретної інформації – не буде виконаною.
Стеганоаналіз на предмет наявності прихованої інформації може набувати різних форм: *виявлення наявності* (детектування), *видобування* і, зрештою, *видалення* чи *руйнування* прихованих даних [40]. Крім того, порушник може поверх вже існуючої прихованої інформації вбудувати певну дезінформацію. Більш детально атаки на стеганосистеми і протидія їм розглянуті у розділі 3.

2.3. Структурна схема і математична модель типової стеганосистеми

В загальному випадку стеганосистема може бути розглянута як система зв'язку [5]. Узагальнена структурна схема стеганосистеми зображена на рис.2.3.

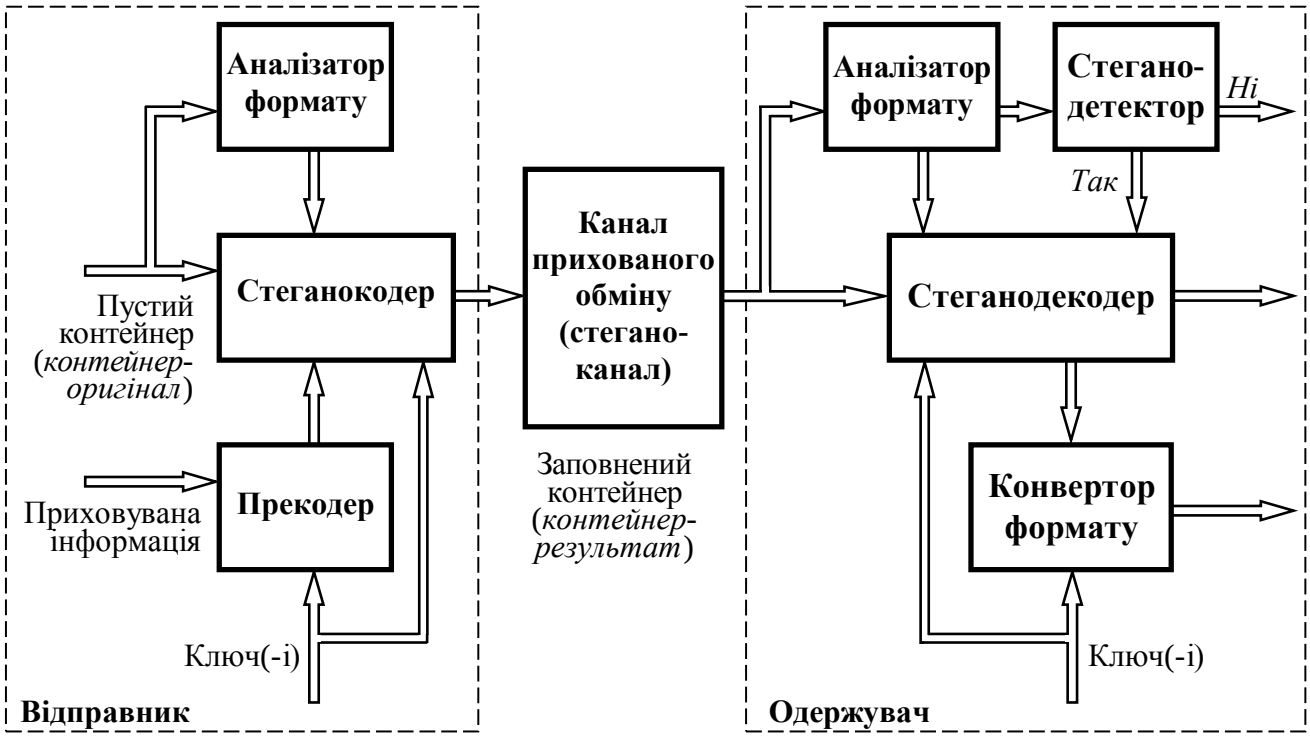


Рис.2.3. Структурна схема стеганосистеми як системи зв'язку

Основними стеганографічними поняттями є *повідомлення* і *контейнер*. Повідомленням $\boldsymbol{m} \in \boldsymbol{M}$ є секретна інформація, наявність якої необхідно приховати, $\boldsymbol{M} = \{\boldsymbol{m}_1, \boldsymbol{m}_2, ..., \boldsymbol{m}_n\}$ – множина всіх повідомлень. Контейнером $\boldsymbol{c} \in \boldsymbol{C}$ зветься несекретна інформація, яку можна використати для приховання повідомлення, $\boldsymbol{C} = \{\boldsymbol{c}_1, \boldsymbol{c}_2, ..., \boldsymbol{c}_q\}$ – множина всіх контейнерів, причому $\boldsymbol{q} >> \boldsymbol{n}$. В якості повідомлення і контейнера можуть виступати як звичайний текст, так і файли мультимедійного формату. *Пустий контейнер* (або так званий *контейнер-оригінал*) – це контейнер $\boldsymbol{c}$, який не містить прихованої інформації. *Заповнений контейнер* (*контейнер-результат*) – контейнер $\boldsymbol{c}$, що містить приховане повідомлення $\boldsymbol{m}$ ($\boldsymbol{c}_{\boldsymbol{m}}$). Однією з вимог, яка при цьому ставиться: контейнер-результат не повинен бути візуально відмінним від контейнера-оригіналу. Виокремлюють два основних типи контейнерів: *потоковий* і *фіксований*.
Потоковий контейнер являє собою послідовність біт, що безперервно змінюється. Повідомлення пакується до нього в реальному масштабі часу, так що в кодері невідомо заздалегідь, чи вистачить розмірів контейнера для передачі всього повідомлення. В один контейнер великого розміру може

бути вбудовано декілька повідомлень. Інтервали між вбудовуваними бітами визначаються генератором ПВП з рівномірним розподілом інтервалів між відліками. Основна проблема полягає у здійсненні синхронізації, визначенні початку і кінця послідовності. Якщо в даних контейнера існують біти синхронізації, заголовки пакетів і т.д., то приховувана інформація може йти відразу після них. Важкість забезпечення синхронізації є перевагою з точки зору забезпечення прихованості передачі. Нажаль, на сьогодні практично відсутні роб**о**ти, присвячені розробці стеганосистем з потоковим контейнером. Як приклад перспективної реалізації потокового контейнера можна навести стеганоприставку до звичайного телефону. При цьому під прикриттям пересічної, незначущої телефонної розмови можна передавати іншу розмову, дані і т.п. Не знаючи секретного ключа не можна не лише довідатися про зміст прихованої передачі, але й про сам факт її існування.

У *фіксованого контейнера* розміри і характеристики є заздалегідь відомими. Це дозволяє здійснювати вкладення даних оптимальним (в певному смислі) чином. Надалі розглядатимуться переважно фіксовані контейнери (у подальшому – контейнери).

Контейнер може бути обраним, випадковим або нав'язаним. *Обраний* контейнер залежить від вбудовуваного повідомлення, а в граничному випадку є його функцією. Такий тип контейнера більше характерний саме для стеганографії. *Нав'язаний* контейнер з'являється, коли особа, що надає контейнер, підозрює про можливе приховане переписування і бажає запобігти йому. На практиці ж частіше за все мають справу з *випадковим* контейнером [5].

Приховування інформації, яка здебільшого має великий об'єм, висуває суттєві вимоги до контейнера, розмір якого повинен щонайменше в декілька разів перевищувати розмір вбудовуваних даних. Зрозуміло, що для підвищення прихованості зазначене співвідношення повинне бути якомога більшим.

Перед тим, як здійснити вкладення повідомлення в контейнер, його необхідно перетворити до певного зручного для пакування виду. Крім того, перед запаковуванням до контейнеру, для підвищення захищеності секретної інформації останню можна зашифрувати достатньо стійким криптографічним кодом [14]. У багатьох випадках також є бажаною стійкість отриманого стеганоповідомлення до спотворювань (у тому числі й зловмисних) [5]. У процесі передачі звук, зображення чи будь-яка інша інформація, яка використовується в якості контейнера, може зазнавати різних трансформацій (у тому числі з використанням алгоритмів із втратою даних): зміни об'єму, перетворення на інший формат тощо. Тому для збереження цілісності вбудовуваного повідомлення може стати необхідним використання коду з виправленням помилок (завадостійке кодування). Початкову обробку приховуваної інформації виконує зображений на рис.2.3 *прекодер*. В якості однієї з найважливіших попередніх обробок повідомлення (а також і контейнера) можна назвати обчислення його узагальненого перетворення Фур'є. Це дозволяє здійснити вбудовування даних в спектральній області, що значно підвищує їх стійкість до спотворень. Слід зазначити, що для підвищення секретності вбудовування, попередня обробка досить часто виконується з використанням *ключа*.

Пакування повідомлення до контейнера (з урахуванням формату даних, що представляють контейнер) відбувається за допомогою *стеганокодера*. Вкладення відбувається, наприклад, шляхом модифікації наймолодших значущих біт контейнера. Взагалі, саме алгоритм (стратегія) внесення елементів повідомлення до контейнера визначає *методи* стеганографії, які в свою чергу поділяються на певні *групи*, наприклад, в залежності від того, файл якого формату було обрано в якості контейнера.

У більшості стеганосистем для пакування і видобування повідомлення використовується *ключ*, який зумовлює секретний алгоритм, що визначає порядок занесення повідомлення до контейнера. За аналогією з криптографією, тип ключа обумовлює існування двох типів стеганосистем: *з секретним ключем* (використовується один ключ, який визначається до початку обміну стеганограмою або передається захищеним каналом) і *з відкритим ключем* (для пакування і розпакування повідомлення застосовуються різні ключі, які різняться таким чином, що за допомогою обчислень неможливо вивести один ключ з іншого. Тому один з ключів (відкритий) може вільно передаватися незахищеним каналом).

В якості секретного алгоритму може бути використаний *генератор псевдовипадкової послідовності* (ПВП) біт. Якісний генератор ПВП, орієнтований на використання в системах захисту інформації повинен задовольняти певним вимогам [42]. Серед яких:
– *криптографічна стійкість* (відсутність у порушника можливості передбачити наступний біт на основі відомих йому попередніх з імовірністю, що відрізняється від $1/2$. На практиці криптографічна стійкість оцінюється статистичними методами. Національним Інститутом Стандартів і Технологій США (НІСТ) розроблено Керівництво з проведення статистичних випробувань генераторів ПВП, орієнтованих на використання в задачах криптографічного захисту інформації [43]);
– *добрі статистичні властивості* (ПВП за своїми статистичними властивостями не повинна істотно відрізнятися від істинно випадкової послідовності);
– *великий період формованої послідовності*;
– *ефективна апаратна / програмна реалізація*.
Статистично (криптографічно) безпечний генератор ПВП повинен задовольняти наступним вимогам:
– жоден статистичний тест не виявляє у ПВП жодних закономірностей, іншими словами не відрізняє цю послідовність від істинно випадкової;
– при ініціалізації випадковими значеннями генератор породжує статистично незалежні псевдовипадкові послідовності.
В якості основи генератора може використовуватися, наприклад, лінійний рекурентний регістр. Тоді адресатам для забезпечення зв'язку повинне повідомлятися початкове заповнення цього регістру. Числа, породжувані генератором ПВП, можуть визначати позиції модифікованих відліків у випадку *фіксованого* контейнера або інтервали між ними у випадку *потокового* контейнера. Слід зауважити, що метод випадкового обрання величини інтервалу між вбудовуваними бітами не є достатньо ефективним з двох причин. По-перше, приховані дані повинні бути розподілені по всьому контейнеру. Тому рівномірний розподіл довжин інтервалів (від найменшого до найбільшого) може бути досягнутий лише наближено, оскільки повинна існувати впевненість у тому, що все повідомлення вбудоване, тобто вмістилося у контейнер. По-друге, довжини інтервалів між відліками шуму (у багатьох моделях сигнал-контейнер розглядається як адитивний шум [44]) розподілені не за рівномірним, а за експонентним законом. Генератор ПВП з експонентно розподіленими інтервалами є складним в реалізації.
Приховувана інформація заноситься у відповідності з ключем до тих бітів, модифікація яких не призводить до істотних спотворень контейнера. Ці біти утворюють так званий *стеганошлях*. Під "істотним" розуміється спотворення, що веде до зростання імовірності виявлення факту наявності прихованого повідомлення після проведення стеганоаналізу.
Стеганографічний канал – канал передачі контейнера-результату (взагалі ж, існування каналу, як, власне кажучи, і одержувача – найбільш узагальнюючий випадок, оскільки заповнений контейнер може, наприклад, зберігатися у "відправника", який поставив собі за мету обмежити неавторизований доступ до певної інформації. У даному випадку відправник виступає і в ролі одержувача). Під час перебування у стеганографічному каналі контейнер, що містить приховане повідомлення, може піддаватися навмисним *атакам* або випадковим *завадам*, детальний опис яких приведено в розділі 3.
В *стеганодетекторі* здійснюється визначення наявності в контейнері (можливо вже зміненому) прихованих даних. Ця зміна може бути обумовлена впливом помилок в каналі зв'язку, операцій обробки сигналу, навмисних атак порушників. Як вже зазначалося вище, в багатьох моделях стеганосистем сигнал-контейнер розглядається як адитивний шум. Тоді задача виявлення і виділення стеганоповідомлення є класичною для теорії зв'язку. Але такий підхід не враховує двох факторів: невипадкового характеру контейнера і вимог по збереженню його якості. Ці моменти не зустрічаються у відомій теорії виявлення і виділення сигналів на фоні адитивного шуму. Очевидно, що їх врахування дозволить побудувати більш ефективні стеганосистеми.
Розрізнюють стеганодетектори, призначені лише для виявлення факту наявності вбудованого повідомлення, і пристрої, призначені для виділення цього повідомлення з контейнеру – *стеганодекодери*.

Отже, в стеганосистемі відбувається об'єднання двох типів інформації таким чином, щоб вони по-різному сприймалися принципово різними детекторами. В якості одного з детекторів виступає система виділення прихованого повідомлення, в якості іншого – людина. Алгоритм вбудовування повідомлення в найпростішому випадку складається з двох основних етапів: *1)* вбудовування в стеганокодері секретного повідомлення до контейнера-оригіналу; *2)* виявлення (виділення) в стеганодетекторі прихованого зашифрованого повідомлення з контейнера-результату. Виходячи з цього, розглянемо *математичну модель стеганосистеми*.

Процес тривіального *стеганографічного перетворення* описується залежностями:

$$\boldsymbol{E}:\ \boldsymbol{C} \times \boldsymbol{M} \to \boldsymbol{S}; \tag{2.1}$$

$$\boldsymbol{D}:\ \boldsymbol{S} \to \boldsymbol{M}, \tag{2.2}$$

де $\boldsymbol{S} = \{(\boldsymbol{c}_1, \boldsymbol{m}_1), (\boldsymbol{c}_2, \boldsymbol{m}_2), \ldots, (\boldsymbol{c}_n, \boldsymbol{m}_n), \ldots, (\boldsymbol{c}_q, \boldsymbol{m}_q)\} = \{\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_q\}$ – множина контейнерів-результатів (стеганограм).

Залежність (2.1) описує процес приховування інформації, залежність (2.2) – видобування прихованої інформації. Необхідною умовою при цьому є відсутність "перетинання" [3], тобто, якщо $\boldsymbol{m}_a \neq \boldsymbol{m}_b$, причому $\boldsymbol{m}_a, \boldsymbol{m}_b \in \boldsymbol{M}$, а $(\boldsymbol{c}_a, \boldsymbol{m}_a), (\boldsymbol{c}_b, \boldsymbol{m}_b) \in \boldsymbol{S}$, то $\boldsymbol{E}(\boldsymbol{c}_a, \boldsymbol{m}_a) \cap \boldsymbol{E}(\boldsymbol{c}_b, \boldsymbol{m}_b) = \varnothing$. Крім того, необхідно, щоб потужність множини $|\boldsymbol{C}| \geq |\boldsymbol{M}|$. При цьому обидва адресати (відправник і одержувач) повинні знати алгоритм прямого ($\boldsymbol{E}$) та оберненого ($\boldsymbol{D}$) стеганографічного перетворення.

Отже, в загальному випадку *стеганосистема* – сукупність $\boldsymbol{\Sigma} = (\boldsymbol{C}, \boldsymbol{M}, \boldsymbol{S}, \boldsymbol{E}, \boldsymbol{D})$ контейнерів (оригіналів і результатів), повідомлень і перетворень, що їх пов'язують.

Для більшості стеганосистем множина контейнерів $\boldsymbol{C}$ обирається таким чином, щоб в результаті стеганографічного перетворення (2.1) заповнений контейнер і контейнер-оригінал були подібними, що формально може бути оцінене за допомогою функції подібності [3].

Означення 2.1. Нехай $\boldsymbol{C}$ – непорожня множина, тоді функція $\boldsymbol{sim}(\boldsymbol{C}) \to (-\infty, 1]$ є *функцією подібності* на множині $\boldsymbol{C}$, якщо для будь-яких $\boldsymbol{x}, \boldsymbol{y} \in \boldsymbol{C}$ справедливо, що $\boldsymbol{sim}(\boldsymbol{x}, \boldsymbol{y}) = 1$ у випадку $\boldsymbol{x} = \boldsymbol{y}$ і $\boldsymbol{sim}(\boldsymbol{x}, \boldsymbol{y}) < 1$ при $\boldsymbol{x} \neq \boldsymbol{y}$.

Стеганосистема може вважатися *надійною*, якщо $\boldsymbol{sim}[\boldsymbol{c}, \boldsymbol{E}(\boldsymbol{c}, \boldsymbol{m})] \approx 1$ для всіх $\boldsymbol{m} \in \boldsymbol{M}$ і $\boldsymbol{c} \in \boldsymbol{C}$. Причому в якості контейнера $\boldsymbol{c}$ повинен обиратися такий, що раніше не використовувався. Крім того, неавторизована особа не повинна мати доступ до набору контейнерів, що використовуються для секретного зв'язку.

Обрання визначеного контейнера $\boldsymbol{c}$ з набору можливих контейнерів $\boldsymbol{C}$ може здійснюватися довільно (так званий *сурогатний метод* вибору контейнера) або шляхом обрання найбільш придатного, який менше за інших зміниться під час стеганоперетворення (*селективний метод*). В останньому випадку контейнер обирається відповідно до правила:

$$\boldsymbol{c} = \max_{\boldsymbol{x} \in \boldsymbol{C}} \boldsymbol{sim}\big[\boldsymbol{x}, \boldsymbol{E}(\boldsymbol{x}, \boldsymbol{m})\big]. \tag{2.3}$$

Також слід зазначити, що функції прямого ($\boldsymbol{E}$) і зворотного ($\boldsymbol{D}$) стеганографічного перетворення у загальному випадку можуть бути довільними (але, звичайно, відповідними одна одній), однак на практиці вимоги до стійкості прихованої інформації накладають на зазначені функції певні обмеження. Так, у переважній більшості випадків, $\boldsymbol{E}(\boldsymbol{c}, \boldsymbol{m}) \approx \boldsymbol{E}(\boldsymbol{c} + \boldsymbol{\delta}, \boldsymbol{m})$, або $\boldsymbol{D}[\boldsymbol{E}(\boldsymbol{c}, \boldsymbol{m})] \approx \boldsymbol{D}[\boldsymbol{E}(\boldsymbol{c} + \boldsymbol{\delta}, \boldsymbol{m})] = \boldsymbol{m}$, тобто незначно модифікований контейнер (на величину $\boldsymbol{\delta}$) не призводить до зміни прихованої в ньому інформації [5].

2.4. Протоколи стеганографічних систем

Важливе значення для досягнення цілей стеганографії мають *протоколи*. Під протоколом розуміється "порядок дій, що вживаються двома чи більше сторонами, призначений для вирішення певної задачі" [11]. Можна розробити винятково ефективний алгоритм приховання інформації, але через його неправильне застосування не досягти своєї мети. І протокол, і алгоритм є певною послідовністю дій. Відмінність між ними полягає в тому, що до протоколу повинні бути обов'язково залучені двоє або більше сторін. При цьому припускається, що учасники приймають на себе зобов'язання дотримуватись протоколу. Так само, як і алгоритм,

протокол складається з кроків. На кожному кроці протоколу виконуються певні дії, які можуть полягати, наприклад, у проведенні деяких обчислень, або у здійсненні якихось дій.

Як вже зазначалося в попередньому підрозділі, в стеганографії розрізнюють системи із секретним ключем і системи з відкритим ключем. У перших використовується один ключ, який повинен бути заздалегідь відомий авторизованим абонентам до початку прихованого обміну секретними повідомленнями (або ж пересланий захищеним каналом під час зазначеного обміну). У системах з відкритим ключем для вбудовування і видобування прихованої інформації використовуються різні, не вивідні один з одного ключі – відкритий і секретний.

З огляду на різноманіття стеганографічних систем, доцільно звести їх до наступних чотирьох типів [3]: *безключові стеганосистеми*, *системи із секретним ключем*, *системи з відкритим ключем* та *змішані стеганосистеми*.

2.4.1. Безключові стеганосистеми

Для функціонування безключових стеганосистем крім алгоритму стеганографічного перетворення відсутня необхідність в жодних додаткових даних на зразок стеганоключа.

Означення 2.2. Сукупність $\boldsymbol{\Sigma} = (\boldsymbol{C}, \boldsymbol{M}, \boldsymbol{S}, \boldsymbol{E}, \boldsymbol{D})$, де $\boldsymbol{C}$ – множина контейнерів-оригіналів; $\boldsymbol{M}$ – множина секретних повідомлень, причому $|\boldsymbol{M}| \le |\boldsymbol{C}|$; $\boldsymbol{S}$ – множина контейнерів-результатів, причому $\boldsymbol{sim}(\boldsymbol{C}, \boldsymbol{S}) \to 1$; $\boldsymbol{E}: \boldsymbol{C} \times \boldsymbol{M} \to \boldsymbol{S}$ та $\boldsymbol{D}: \boldsymbol{S} \to \boldsymbol{M}$ – відповідно функції прямого (вбудовування) і оберненого (видобування) стеганоперетворення, причому $\boldsymbol{D}[\boldsymbol{E}(\boldsymbol{c}, \boldsymbol{m})] = \boldsymbol{m}$ для будь-яких $\boldsymbol{m} \in \boldsymbol{M}$ і $\boldsymbol{c} \in \boldsymbol{C}$, зветься *безключовою стеганографічною системою*.

Таким чином, безпека безключових стеганосистем базується лише на таємності використовуваних стеганографічних перетворень $\boldsymbol{E}$ і $\boldsymbol{D}$. Це суперечить визначальному принципові, що встановив *A. Kerckhoffs* для систем захисту інформації [6], оскільки стійкість системи залежатиме лише від ступеню поінформованості порушника щодо функцій $\boldsymbol{E}$ і $\boldsymbol{D}$.

Для підвищення безпеки безключових систем, перед початком процесу стеганографічного приховування попередньо виконується криптографічне шифрування приховуваної інформації. Цілком очевидно, що такий підхід збільшує захищеність усього процесу зв'язку, оскільки ускладнює виявлення прихованого повідомлення. Однак "сильні" стеганосистеми, як правило, здатні виконувати функції, що на них покладені, без попереднього криптографічного захисту вбудовуваного повідомлення.

2.4.2. Стеганосистеми із секретним ключем

За принципом Керхгофса, безпека системи повинна ґрунтуватися на певному фрагменті секретній інформації – ключі, який (як правило, попередньо) розділяється між авторизованими особами. Відправник, вбудовуючи секретне повідомлення в обраний контейнер $\boldsymbol{c}$, використовує стеганоключ $\boldsymbol{k}$. Якщо даний ключ є відомим одержувачеві, то він зможе видобути з контейнера приховане повідомлення. Без знання ключа будь-яка стороння особа цього зробити не зможе.

Означення 2.3. *Стеганосистемою із секретним ключем* називають сукупність $\boldsymbol{\Sigma} = (\boldsymbol{C}, \boldsymbol{M}, \boldsymbol{K}, \boldsymbol{S^K}, \boldsymbol{E}, \boldsymbol{D})$, де $\boldsymbol{C}$ – множина контейнерів-оригіналів; $\boldsymbol{M}$ – множина секретних повідомлень, причому $|\boldsymbol{M}| \le |\boldsymbol{C}|$; $\boldsymbol{S^K}$ – множина контейнерів-результатів, причому $\boldsymbol{sim}(\boldsymbol{C}, \boldsymbol{S^K}) \to 1$; $\boldsymbol{K}$ – множина секретних стеганоключів; $\boldsymbol{E}: \boldsymbol{C} \times \boldsymbol{M} \times \boldsymbol{K} \to \boldsymbol{S^K}$ та $\boldsymbol{D}: \boldsymbol{S^K} \times \boldsymbol{K} \to \boldsymbol{M}$ – функції прямого і оберненого стеганоперетворення з властивістю $\boldsymbol{D}[\boldsymbol{E}(\boldsymbol{c}, \boldsymbol{m}, \boldsymbol{k}), \boldsymbol{k}] = \boldsymbol{m}$ для будь-яких $\boldsymbol{m} \in \boldsymbol{M}$, $\boldsymbol{c} \in \boldsymbol{C}$ і $\boldsymbol{k} \in \boldsymbol{K}$.

Даний тип стеганосистем припускає наявність безпечного (захищеного) каналу обміну стеганоключами.

Іноді ключ $\boldsymbol{k}$ обчислюють за допомогою секретної хеш-функції (*hash function*), використовуючи деякі характерні риси контейнера. Якщо стеганоперетворення $\boldsymbol{E}$ не змінює в остаточній стеганограмі обрані особливості контейнера, то одержувач також зможе обчислити стеганоключ (хоча й у цьому випадку захист залежатиме від таємності хеш-функції, і, таким чином, знову порушується принцип Керхгофса). Очевидно, що для досягнення адекватного рівня захисту, таку особливість у контейнері необхідно вибирати досить уважно.

У деяких алгоритмах під час видобування прихованої інформації додатково потрібні відомості про первинний контейнер або деякі інші дані, які відсутні у стеганограмі. Такі системи становлять обмежений інтерес, оскільки вони вимагають передавання початкового вигляду контейнера, що еквівалентно традиційній задачі ключового обміну. Подібні алгоритми можуть бути відзначені як окремий випадок стеганосистем із секретним ключем, у яких $\boldsymbol{K} = \boldsymbol{C}$ або $\boldsymbol{K} = \boldsymbol{C} \times \boldsymbol{K'}$, де $\boldsymbol{K'}$ – множина додаткового набору секретних ключів.

2.4.3. Стеганосистеми з відкритим ключем

Стеганографія з відкритим ключем спирається на досягнення криптографії останніх 30 років. Стеганографічні системи з відкритим ключем не мають потреби в додатковому каналі ключового обміну. Для їхнього функціонування необхідно мати два стеганоключі: один *секретний*, який необхідно зберігати в таємниці, а другий – *відкритий*, який може зберігатися в доступному для всіх місці. При цьому відкритий ключ використовується для вбудовування повідомлення, а секретний – для її видобування.

Означення 2.4. *Стеганосистемою з відкритим ключем* називають сукупність $\boldsymbol{\Sigma} = (\boldsymbol{C}, \boldsymbol{M}, \boldsymbol{K}, \boldsymbol{S^K}, \boldsymbol{E}, \boldsymbol{D})$, де $\boldsymbol{C}$ – множина контейнерів-оригіналів; $\boldsymbol{M}$ – множина секретних повідомлень, $|\boldsymbol{M}| \le |\boldsymbol{C}|$; $\boldsymbol{S^K}$ – множина контейнерів-результатів, $\boldsymbol{sim}(\boldsymbol{C}, \boldsymbol{S^K}) \to 1$; $\boldsymbol{K} = (\boldsymbol{k}_\text{B}, \boldsymbol{k}_\text{C})$ – множина пар стеганоключів (відкритий ключ $\boldsymbol{k}_\text{B}$ використовується для приховування інформації, а секретний ключ $\boldsymbol{k}_\text{C}$ – для її видобування); $\boldsymbol{E}\text{: } \boldsymbol{C} \times \boldsymbol{M} \times \boldsymbol{k}_\text{B} \to \boldsymbol{S^K}$ та $\boldsymbol{D}\text{: } \boldsymbol{S^K} \times \boldsymbol{k}_\text{C} \to \boldsymbol{M}$ – функції прямого і оберненого стеганоперетворення з властивістю $\boldsymbol{D}[\boldsymbol{E}(\boldsymbol{c}, \boldsymbol{m}, \boldsymbol{k}_\text{B}), \boldsymbol{k}_\text{C}] = \boldsymbol{m}$ для будь-яких $\boldsymbol{m} \in \boldsymbol{M}$, $\boldsymbol{c} \in \boldsymbol{C}$ та $\boldsymbol{k}_\text{B}, \boldsymbol{k}_\text{C} \in \boldsymbol{K}$.

Слід зауважити, що стеганоключ не шифрує дані, а приховує місце їх вбудовування у контейнері. Приховані дані можуть бути додатково зашифровані класичними методами, але це питання не стосується безпосередньо стеганографії.

Стеганосистеми з відкритими ключами використовують той факт, що функція видобування прихованих даних $\boldsymbol{D}$ може бути застосована до будь-якого контейнера не залежно від того, знаходиться в ньому приховане повідомлення чи ні ($\boldsymbol{c_i}$ або $\boldsymbol{s_i}$). Якщо приховане повідомлення відсутнє, на виході одержуватиметься певна випадкова послідовність. Якщо ця послідовність статистично не відрізняється від шифртексту криптосистеми з відкритим ключем, тоді в безпечній стеганосистемі можна приховувати отриманий у такий спосіб шифртекст, а не відкритий [3].

2.4.4. Змішані стеганосистеми

На практиці перевага віддається безключовим стеганосистемам, хоча останні можуть бути відразу розкриті у випадку, якщо порушник дізнається про метод стеганоперетворення, що був при цьому застосований. У зв'язку з цим у безключових системах часто використовують особливості криптографічних систем з відкритим і/або секретним ключем [3,12].

З огляду на велику різноманітність форматів, які можуть мати приховувані повідомлення і контейнери (текст, звук або відео, що у свою чергу також поділяються на відповідні підформати), доцільним бачиться попереднє перетворення приховуваного повідомлення на зручний для вбудовування і оптимальний з точки зору рівня прихованості в заданому контейнері формат [5]: $\boldsymbol{U}\text{: } \boldsymbol{C} \times \boldsymbol{M} \times \boldsymbol{K} \to \boldsymbol{W}$, $\boldsymbol{w} = \boldsymbol{U}(\boldsymbol{c}, \boldsymbol{m}, \boldsymbol{k})$. Тобто вважається за доцільне врахуванння як особливості вбудовуваного повідомлення, так і особливості контейнера, до якого його планується ввести. Довільність функції $\boldsymbol{U}$ обмежується вимогами стійкості до різного роду впливів на отриманий контейнер-результат. Крім того, функція $\boldsymbol{U}$ є складеною:

$$\boldsymbol{U} = \boldsymbol{T} \circ \boldsymbol{G}, \tag{2.4}$$

де $\boldsymbol{G}\text{: } \boldsymbol{M} \times \boldsymbol{K} \to \boldsymbol{Z}$; $\boldsymbol{T}\text{: } \boldsymbol{C} \times \boldsymbol{Z} \to \boldsymbol{W}$.

Функція $\boldsymbol{G}$ може бути реалізована, наприклад, за допомогою криптографічного безпечного генератора ПВП з $\boldsymbol{K}$ в якості початкового значення. Для підвищення стійкості прихованого повідомлення можуть застосовуватися завадостійкі коди, наприклад, коди Хемінга, БЧХ, Голея, згорткові коди [65]. Оператор $\boldsymbol{T}$ модифікує кодові слова $\boldsymbol{Z}$ з урахуванням формату контейнера, в результаті чого одержується оптимальне для вбудовування повідомлення. Функція $\boldsymbol{T}$ повинна

бути обрана таким чином, щоб контейнер-оригінал $\boldsymbol{C}$, контейнер-результат $\boldsymbol{S}$ і модифікований у передбачених межах контейнер-результат $\tilde{\boldsymbol{S}}$ породжували одне й те саме оптимальне для вбудовування повідомлення:

$$\boldsymbol{T}: \boldsymbol{C} \times \boldsymbol{Z} = \boldsymbol{T}: \boldsymbol{S} \times \boldsymbol{Z} = \boldsymbol{T}: \tilde{\boldsymbol{S}} \times \boldsymbol{Z} = \rightarrow \boldsymbol{W}. \qquad (2.5)$$

Процес вбудовування повідомлення $\boldsymbol{W}$ до контейнера-оригіналу $\boldsymbol{C}$ при цьому можна описати як суперпозицію сигналів:

$$\boldsymbol{E}: \boldsymbol{C} \times \boldsymbol{V} \times \boldsymbol{W} \rightarrow \boldsymbol{S}; \quad \boldsymbol{s}(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{c}(\boldsymbol{x}, \boldsymbol{y}) * \boldsymbol{v}(\boldsymbol{x}, \boldsymbol{y})\, \boldsymbol{w}(\boldsymbol{x}, \boldsymbol{y})\, \boldsymbol{p}(\boldsymbol{x}, \boldsymbol{y}), \qquad (2.6)$$

де $\boldsymbol{v}(\boldsymbol{x}, \boldsymbol{y})$ – маска вбудовування повідомлень, яка враховує характеристики зорової системи середньостатистичної людини і слугує для зменшення помітності цих повідомлень; $\boldsymbol{p}(\boldsymbol{x}, \boldsymbol{y})$ – проектуюча функція, що залежить від ключа; знак «$*$» позначує оператор суперпозиції, що в загальному випадку включає до себе, окрім додавання, обмеження рівня і квантування.

Проектуюча функція здійснює "розподіл" оптимізованого повідомлення по всій області контейнера. Її використання може розглядатися як реалізація рознесення конфіденційної інформації паралельними каналами. Крім того, дана функція має певну просторову структуру і кореляційні властивості, що використовуються для протидії, наприклад, геометричним атакам (див. розділ 3).

Ще один можливий опис процесу вбудовування наведено в [5] з посиланням на [41]. Представимо стеганографічну систему як систему зв'язку з передачею додаткової інформації (рис.2.4). В даній моделі кодер і декодер мають доступ, окрім ключа, ще й до інформації про канал (тобто про контейнер і про можливі атаки). У залежності від положення перемикачів $\boldsymbol{A}$ і $\boldsymbol{Б}$ виділяють чотири класи стеганосистем (при цьому вважається, що ключ завжди відомий кодеру і декодеру).

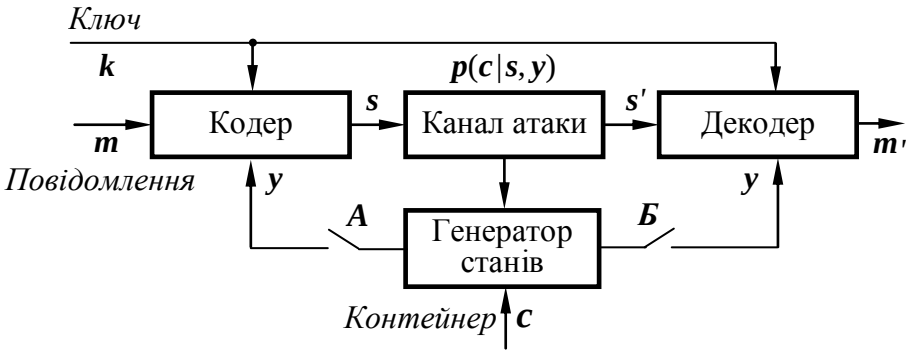


Рис.2.4. Представлення стеганосистеми як системи зв'язку з передачею додаткової інформації

I клас: Додаткова інформація відсутня (перемикачі розімкнуті) – так звані "класичні" стеганосистеми. В ранніх роботах зі стеганографії вважалося, що інформація про канал є недоступною кодеку. Виявлення прихованої інформації здійснювалося шляхом обчислення коефіцієнта кореляції між прийнятим контейнером і обчисленим за ключем повідомленням Якщо коефіцієнт перевищував деякий поріг, приймалося рішення щодо присутності вбудованих даних. Але відомо, що кореляційний приймач є оптимальним лише у випадку адитивної гаусівської завади. При інших атаках (наприклад, геометричних спотвореннях) дані стеганосистеми давали незадовільні результати.

II клас: Інформація про канал є відомою тільки кодеру (ключ $\boldsymbol{A}$ замкнутий, $\boldsymbol{Б}$ розімкнутий). Така конструкція привернула до себе увагу завдяки роботі [47]. Особливістю схеми є те, що, будучи "сліпою", вона має ту ж теоретичну пропускну здатність, що й схема з наявністю контейнера-оригінала в декодері. До недоліків стеганосистем класу II можна віднести високу складність кодера (необхідність побудови кодової книги для кожного контейнера), а також відсутність адаптації схеми до можливих атак. В останній час запропоновано низку практичних підходів, які усувають ці недоліки. Зокрема, для зниження складності кодера пропонується використовувати структуровані кодові книги, а декодер розраховувати на випадок найгіршої атаки.

III клас: Додаткова інформація є відомою тільки декодеру (перемикач ***А*** розімкнутий, ***Б*** замкнутий). Декодер будується з урахуванням можливих атак. У результаті одержуються стійкі до геометричних атак стеганосистеми. Одним з методів досягнення цієї мети є використання так званого опорного вбудованого повідомлення (аналог пілот-сигналу в радіозв'язку). Опорне повідомлення – невелика кількість біт, вбудовуваних в інваріантні до перетворювань коефіцієнти сигналу. Наприклад, можна виконати вбудовування в амплітудні коефіцієнти перетворення Фур'є, які є інваріантними до афінних (геометричних) перетворювань. Тоді опорне повідомлення вкаже, яке перетворення виконав над контейнером порушник. Іншим призначенням пілотного повідомлення є боротьба із завмираннями, за аналогією з радіозв'язком. Завмираннями в даному контексті можна вважати зміну значень відліків сигналу при вбудовуванні даних, атаках, додаванні негаусівського шуму тощо. У радіозв'язку для боротьби із завмираннями використовується метод рознесеного прийому (по частоті, у часі, просторі, за кодом). У стеганографії ж використовується рознесення вбудованих повідомлень у просторі контейнера. Пілотне повідомлення генерується в декодері на основі ключа.
IV клас: Додаткова інформація є відомою як у кодері, так і в декодері (обидва перемикачі замкнуті). Як відзначено у [46], всі перспективні стеганосистеми повинні будуватися саме за цим принципом. Оптимальність такої схеми досягається шляхом оптимального узгодження кодера з сигналом-контейнером, а також адаптивним управлінням декодером в умовах спостереження каналу атак.

2.5. Висновки

В даному розділі шляхом аналізу спеціалізованих літературних джерел та ресурсів мережі Internet наведено узагальнене визначення поняття стеганографічної системи, визначено існуючі та перспективні напрямки, за якими можливе використання стеганографії як інструменту захисту інформації в автоматизованих системах, окреслено проблему співвідношення між стійкістю стеганосистеми та об'ємом приховуваного за її допомогою повідомлення, розкрито сутність таких основних понять стеганографії як повідомлення, контейнер-оригінал, контейнер-результат, стеганоключ, стеганоканал та ін. Це дозволило безпосередньо перейти до побудови структурної схеми стеганосистеми, яку проведено з позицій теорії зв'язку, виконати систематизований огляд відомих протоколів стеганосистем.
Отримані результати дозволяють зробити висновок про недоцільність використання безключових стеганосистем, безпека яких базується лише на таємності використовуваних стеганографічних перетворень. Наведено переваги використання відкритого ключа порівняно із секретним ключем. До основного переліку стеганографічних протоколів запропоновано включити протокол, що розуміє під собою додаткове попереднє перетворення приховуваної інформації до оптимального формату, виходячи з особливостей формату носія, який планується використати в якості контейнера.

3. ПРИНЦИПИ СТЕГАНОГРАФІЧНОГО АНАЛІЗУ

3.1. Вступні положення

Основною метою стеганоаналізу є моделювання стеганографічних систем та їхнє дослідження для отримання якісних і кількісних оцінок надійності використовуваного стеганоперетворення, а також побудова методів виявлення приховуваної в контейнері інформації, її модифікації або руйнування.

Термінологія стеганоаналізу є аналогічною термінології криптоаналізу, однак присутні й деякі істотні розбіжності. Криптоаналіз застосовується з метою *дешифрування змісту* криптограм, а стеганоаналіз – для *виявлення наявності* прихованої інформації.

За рівнем забезпечення таємності стеганосистеми поділяються на теоретично стійкі, практично стійкі і нестійкі системи [3].

Теоретично стійка (абсолютно надійна) стеганосистема здійснює приховування інформації лише в тих фрагментах контейнера, значення елементів яких не перевищують рівень шумів або помилок квантування, і при цьому теоретично доведено, що неможливо створити стеганоаналітичний метод виявлення прихованої інформації (рис.3.1, *а*).

Практично стійка стеганосистема проводить таку модифікацію фрагментів контейнера, зміни яких можуть бути виявлені, але відомо, що на даний момент необхідні стеганоаналітичні методи в порушника відсутні або поки що не розроблені (рис.3.1, *б*).

Нестійка стеганосистема приховує інформацію таким чином, що існуючі стеганоаналітичні засоби дозволяють її виявити (рис.3.1, *в*). У цьому випадку стеганографічний аналіз допомагає знайти уразливі місця стеганографічного перетворення і провести його удосконалення таким чином, щоб усі зміни, внесені до контейнеру, знову виявилися б в області теоретичної або, принаймні, практичної нерозрізненості (рис.3.1, *а, б*).

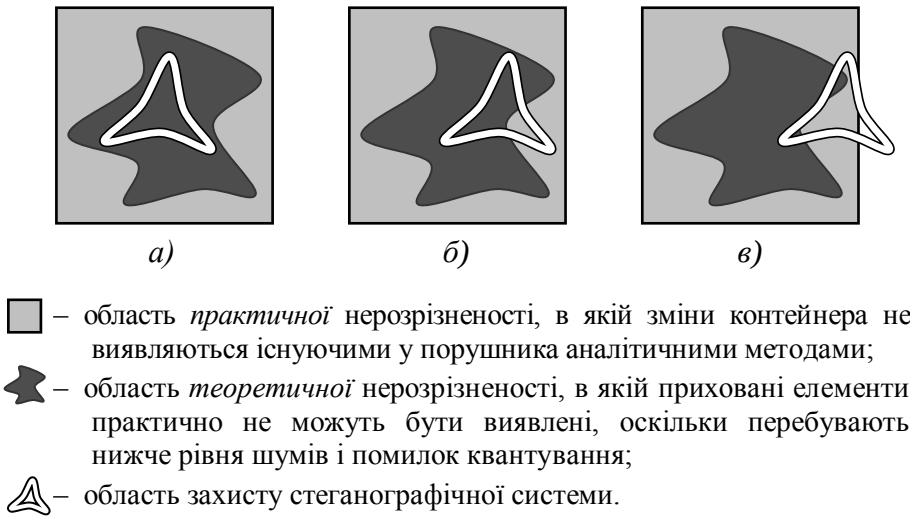


Рис.3.1. Співвідношення методів стеганозахисту і стеганоаналізу

Порушник може бути пасивним, активним і зловмисним. В залежності від цього він може створювати різні загрози. *Пасивний порушник* може лише виявити факт наявності стеганоканалу і (можливо) дізнаватися про зміст повідомлення. Чи буде здатним він прочитати повідомлення після його виявлення залежить від стійкості системи шифрування, і це питання, як правило, не розглядається в стеганографії. Діапазон дій *активного порушника* є значно ширшим. Приховане повідомлення може бути ним видалене або ж зруйноване. Дії *зловмисного порушника* найбільш небезпечні. Він здатний не лише зруйнувати, але й створювати фальшиві стеганограми (дезінформація) [40].

Для здійснення тієї чи іншої загрози порушник застосовує атаки.

3.2. Види атак на стеганографічну систему

Стеганосистема вважається *зламаною*, якщо порушникові вдалося, принаймні, довести існування прихованого повідомлення в перехопленому контейнері. Передбачається, що порушник здатний проводити будь-які види атак і має необмежені обчислювальні можливості. Якщо йому не вдається підтвердити гіпотезу про те, що в контейнері приховано секретне повідомлення, то стеганографічна система вважається *стійкою*.

У більшості випадків виділяють декілька етапів зламу стеганографічної системи:

– виявлення факту присутності прихованої інформації;

– видобування прихованого повідомлення;

– видозміна (модифікація) прихованої інформації;

– заборона на здійснення будь-якого пересилання інформації, у тому числі і прихованої [40].

Перші два етапи відносяться до пасивних атак на стеганосистему, а останні – до активних (або зловмисних) атак. Виділяють такі види атак на стеганосистеми (за аналогією з криптоаналізом) [3,5]:

– *атака на основі відомого заповненого контейнера*. У цьому випадку порушник має у своєму розпорядженні один або декілька заповнених контейнерів (в останньому випадку передбачається, що вбудовування прихованої інформації здійснювалося тим самим способом). Завдання порушника може складатися у виявленні факту наявності стеганоканалу (основне завдання), а також у видобуванні даних чи визначенні ключа. Знаючи ключ, порушник матиме можливість аналізу інших стеганоповідомлень;

– *атака на основі відомого вбудованого повідомлення*. Цей тип атаки більшою мірою характерний для систем захисту інтелектуальної власності, коли в якості ЦВЗ, наприклад, використовується відомий логотип фірми. Завданням аналізу є одержання ключа. Якщо відповідний прихованому повідомленню заповнений контейнер невідомий, то завдання є вкрай важко розв’язуваним;

– *атака на основі обраного прихованого повідомлення*. У цьому випадку порушник може пропонувати для передачі свої повідомлення й аналізувати отримувані при цьому контейнери-результати;

– *адаптивна атака на основі обраного прихованого повідомлення*. Ця атака є окремим випадком попередньої. При цьому порушник має можливість обирати повідомлення для нав’язування їх адаптивно, в залежності від результатів аналізу попередніх контейнерів-результатів.

– *атака на основі обраного заповненого контейнера*. Цей тип атаки є більше характерним для систем ЦВЗ. Стеганоаналітик має детектор заповнених контейнерів у вигляді “чорного ящика” і декілька таких контейнерів. Аналізуючи продетектовані приховані повідомлення, порушник намагається розкрити ключ.

Крім того, в порушника може існувати можливість застосувати ще три атаки, які не мають прямих аналогій у криптоаналізі:

– *атака на основі відомого порожнього контейнера*. Якщо останній є відомим порушнику, то шляхом порівняння його з підозрюваним на присутність прихованих даних контейнером, той завжди може встановити факт наявності стеганоканалу. Незважаючи на тривіальність цього випадку, у ряді робіт приводиться його інформаційно-теоретичне обґрунтування. Набагато цікавішим бачиться сценарій, коли контейнер відомий приблизно, з деякою похибкою (як це може мати місце при додаванні до нього шуму). У цьому випадку існує можливість побудови стійкої стеганосистеми [5];

– *атака на основі обраного порожнього контейнера*. У цьому випадку порушник здатний змусити користуватися запропонованим ним контейнером. Останній, наприклад, може мати більші однорідні області (однотонні зображення), і тоді буде важко забезпечити таємність вбудовування;

– *атака на основі відомої математичної моделі контейнера або його частини*. При цьому атакуючий намагається визначити відмінність підозрілого повідомлення від відомої йому моделі. Наприклад, можна припустити, що біти всередині певної частини зображення є корельованими. Тоді відсутність такої кореляції може служити сигналом про наявне приховане повідомлення. Завдання того, хто вбудовує повідомлення, полягає у тому, щоб не порушити статистики контейнера. Відправник і той, хто атакує, можуть мати у своєму розпорядженні різні моделі

сигналів, тоді в інформаційно-приховуючому протиборстві, переможе той, хто має ефективнішу (оптимальнішу) модель.
Основна мета атаки на стеганографічну систему аналогічна атакам на криптосистему з тією лише різницею, що різко зростає значимість активних (зловмисних) атак. Будь-який контейнер може бути замінений з метою видалення або руйнування прихованого повідомлення, незалежно від того, існує воно в контейнері чи ні. Виявлення існування прихованих даних зберігає час на етапі їхнього видалення, тому що буде потрібно обробляти тільки ті контейнери, які містять приховану інформацію. Навіть за найкращих умов для атаки, задача видобування прихованого повідомлення з контейнера може виявитися дуже складною. Однозначно стверджувати про факт існування прихованої інформації можна лише після її виділення в явному вигляді. Іноді метою стеганографічного аналізу є не алгоритм взагалі, а пошук, наприклад, конкретного стеганоключа, що використовується для вибору бітів контейнера в стеганоперетворенні.

3.3. Основні етапи практичного стеганоаналізу

Фактично будь-яке стеганографічне перетворення базується на двох визначальних принципах [3]:
– в якості носія прихованої інформації (контейнера) обирається об’єкт, структура якого припускає можливість певного спотворення власної інформації контейнера, зберігаючи при цьому його функціональність;
– рівень внесених до структури контейнера спотворень повинен бути нижчим за рівень чутливості засобів розпізнавання (у тому числі й до розпізнавання органами відчуття людини).
В якості стеганоконтейнерів, як вже зазначалося вище, можуть використовуватися майже всі відомі носії інформації, застосовувані у сучасних мережах передавання даних. При цьому методи приховування інформації орієнтуються в основному на внутрішню структуру контейнера, яка може являти собою символьні або бітові дані, коефіцієнти перетворення Фур’є, широкосмугове кодування, коефіцієнти ущільнення тощо. Приховування даних у медіасередовищі вимагає дотримання певних умов при внесенні змін, що має на меті усунення прояву слідів застосування операцій стеганоперетворення. Наприклад, у випадку зображень зазначені зміни можуть за певних дій з боку порушника (як навмисних, так і випадкового характеру) ставати видимими для людського ока і, отже, явно вказувати на використання стеганографічних засобів. Очевидно, що сліди, залишені останніми, можуть істотно допомогти виявити існування прихованого повідомлення, таким чином, компрометуючи стеганосистему в цілому.
Однією з головних *задач стеганоаналізу* є дослідження можливих слідів застосування стеганографічних засобів і розробка методів, які б дозволяли виявити факти їхнього використання [3]. Застосування конкретного стеганографічного перетворення вимагає від стеганоаналітика індивідуального підходу до його дослідження.
Дослідження повідомлень, прихованих одним з множини існуючих стеганографічних методів, або, більш точно, підозрілих у цьому відношенні, є досить трудомістким процесом.
Для успішного проведення стеганоаналізу є необхідним, але в жодному разі не достатнім:
– мати для аналізу стеганозасіб, за допомогою якого здійснюється приховування повідомлення;
– мати можливість відновлювати застосовувані в системі стеганографічний і, можливо, криптографічний алгоритми, проводити їхній експертний аналіз і розробляти алгоритм визначення ключів;
– мати змогу використовувати для проведення стеганоаналізу обчислювальний ресурс необхідної потужності;
– підтримувати на належному рівні теоретичні і практичні знання в галузі комп’ютерної стеганографії.
Можна виділити наступні декілька напрямків практичного розвитку стеганографічного аналізу [3]:
1) Розробка імовірнісно-статистичних методів розпізнавання, застосування елементів штучного інтелекту для отримання оцінок надійності стеганографічних перетворень, а також при створенні детекторів (фільтрів) для аналізу інформаційних потоків з метою виявлення і перекриття прихованих каналів зв’язку. У такому випадку перевірка наявності прихованої інформації

зводиться до певної оцінки з використанням статистичних критеріїв (послідовної кореляції, ентропії зображення, дисперсії молодшого біта тощо). Розроблювані з цією метою засоби повинні не лише забезпечувати низький рівень похибки під час розпізнавання прихованих повідомлень (особливо в тих випадках, коли використовуються попередні шифрування), але й бути універсальними, тобто повинна існувати можливість детектування повідомлень вбудованих різними стеганографічними методами.
2) Аналіз конкретних програмних стеганографічних засобів з метою відновлення алгоритмів і оптимальної розробки методу їхнього дослідження. Основна складність в даному випадку полягає у великій трудомісткості, обумовленій необхідністю індивідуального підходу до кожного конкретного алгоритму, який реалізує метод приховування інформації, а також значним об'ємом обчислень, необхідних для відновлення стеганоключів.
3) Розробка технології активних і зловмисних атак для внесення невідновлюваних спотворень у передбачувану стеганограму з метою спровокувати її повторне передавання в іншому контейнері, що підтвердило б факт використання стеганозасобів.

3.4. Оцінювання якості стеганосистеми

Створення й експлуатація надійного стеганографічного засобу передбачає наявність визначеного інструментарію для його контролю й оцінювання [3]. Кількісне оцінювання стійкості стеганографічної системи захисту до зовнішніх впливів являє собою досить складну задачу, яка зазвичай на практиці реалізується методами *системного аналізу*, *математичного моделювання* або *експериментального дослідження*.
Як правило, професійно розроблена стеганосистема забезпечує трирівневу модель захисту інформації, що вирішує дві основні задачі: *приховування* самого факту наявності інформації, що захищається (перший рівень захисту) і *блокування* несанкціонованого доступу до інформації, яке здійснюється шляхом обрання відповідного методу приховування інформації (другий рівень захисту). Зрештою, необхідно брати до уваги й можливість існування третього рівня – попереднього криптографічного захисту (*шифрування*) приховуваної інформації.
На рис.3.2 представлена можлива структура процесу моделювання й оцінювання стійкості стеганосистеми [3,48]. Як видно, надійність і час стійкості стеганосистеми у випадку проведення аналізу й випробувань визначаються обчислювальними можливостями комплексу.
Оцінювання якості основної характеристики стеганосистеми – *рівня прихованості* – забезпечується шляхом проведення аналітичних досліджень (стеганоаналізу) і натурних випробувань [3]. Для оцінювання якості стеганографічного приховування часто застосовують відомі методи з інших галузей, у першу чергу – криптоаналізу. Оскільки абонент-одержувач може відновлювати приховану інформацію з прийнятого повідомлення, то цілком очевидно, що існує деякий механізм її видобування. Якщо порушник, висуваючи гіпотези про можливе стеганографічне перетворення, має деякий інструмент для їхньої перевірки, то він має шанси на підтвердження факту існування прихованої інформації, здійснення пошуку механізму видобування секретного повідомлення і, зрештою, розкриття змісту повідомлення. Тому, у першу чергу, для детектування стеганограм можливе застосування різновидів описаних вище атак на стеганосистему і значну частину методів криптоаналізу.

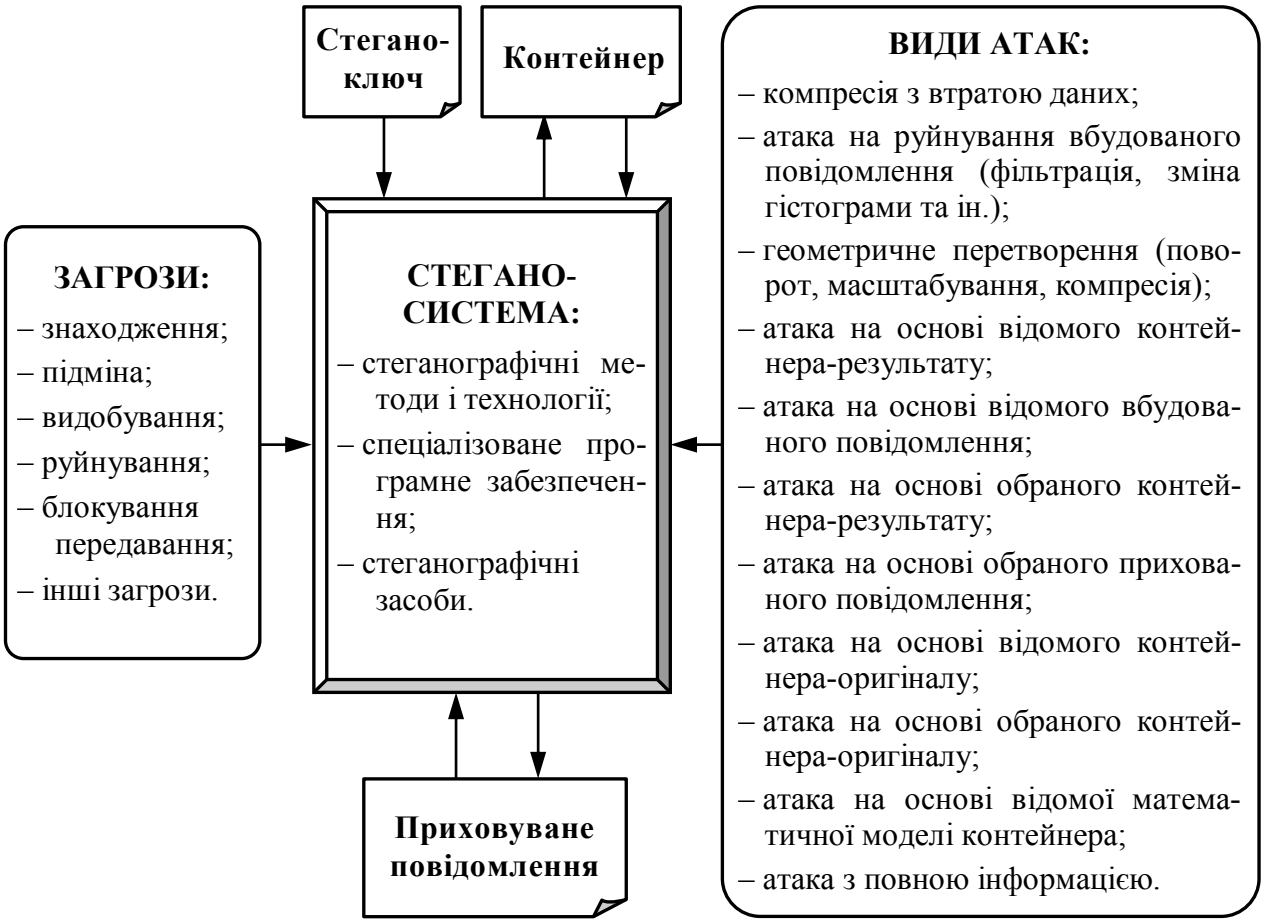


Рис.3.2. Модель аналізу загроз й оцінювання стійкості стеганосистем

Достатньо ефективними у деяких випадках є методи оцінювання рівня прихованості стеганозасобів на основі аналізу їхніх *статистичних характеристик*. Статистична теорія дає кількісні критерії випадковості, що дозволяє створювати детектори, які виявлятимуть статистичні розбіжності між послідовностями. У випадку наявності необхідного об'єму аналізованої послідовності з достатньо високою імовірністю можна робити висновки про основні характеристики послідовності, виділеної для аналізу з контейнера. На початковому етапі аналізу рекомендується скористатися традиційними *статистичними* (χ^2, тести на заборонені символи, на довжину циклу тощо), *емпіричними* (перевірки частот, серій, інтервалів, перестановок; перевірки на монотонність, "покер-тестом", тестом "збирача купонів") або *спектральними* тестами [3]. В подальшому доцільно використовувати більш гнучкі методи, які іноді спеціально розроблюються під конкретну задачу.

Для порівняльного оцінювання якості стеганографічних засобів розроблюють різні *показники*, що дають кількісні оцінки. Найбільша їхня кількість розроблена для стеганометодів, які працюють із зображеннями та відео (методів ЦВЗ). Зазвичай такі показники оперують із зображенням на рівні пікселів, хоча після належної адаптації вони застосовні й до інших способів опису зображення, а також до аудіоданих. Найбільш популярним показником при аналізі рівня спотворювань, які вносяться у контейнер під час приховування в ньому інформації, є взяте з радіотехніки співвідношення *сигнал/шум*, яке обчислюється у децибелах.

У табл.3.1 наведено низку показників, використовуваних під час оцінювання спотворень, що вносяться стеганоперетвореннями до зображення [49-52].

Більшість показників спотворення або критеріїв якості, що використовуються при візуальній обробці інформації, належать до групи *різницевих показників спотворення*. Дані показники базуються на відмінності між контейнером-оригіналом (неспотворений сигнал) і контейнером-результатом (спотворений сигнал).

До другої групи входять показники, засновані на кореляції між оригінальним і спотвореним сигналами (так звані *кореляційні показники спотворення*).

Таблиця 3.1.

Найпоширеніші показники візуального спотворення, основані на аналізі піксельної структури контейнера

Різницеві показники спотворення	
Максимальна різниця *(Maximum Difference)*	$$MD = \max_{x,y} \left\| C_{x,y} - S_{x,y} \right\|. \quad (3.1)$$
Середня абсолютна різниця *(Average Absolute Difference)*	$$AD = \frac{1}{XY} \cdot \sum_{x,y} \left\| C_{x,y} - S_{x,y} \right\|. \quad (3.2)$$
Нормована середня абсолютна різниця *(Normalized Average Absolute Difference)*	$$NAD = \sum_{x,y} \left\| C_{x,y} - S_{x,y} \right\| \Big/ \sum_{x,y} \left\| C_{x,y} \right\|. \quad (3.3)$$
Середньоквадратична помилка *(Mean Square Error)*	$$MSE = \frac{1}{XY} \cdot \sum_{x,y} \left(C_{x,y} - S_{x,y} \right)^2. \quad (3.4)$$
Нормована середньоквадратична помилка *(Normalized Mean Square Error)*	$$NMSE = \sum_{x,y} \left(C_{x,y} - S_{x,y} \right)^2 \Big/ \sum_{x,y} \left(C_{x,y} \right)^2. \quad (3.5)$$
L^p-норма *(L^p-norm)*	$$L^p = \left(\frac{1}{XY} \cdot \sum_{x,y} \left\| C_{x,y} - S_{x,y} \right\|^p \right)^{1/p}. \quad (3.6)$$
Лапласова середньоквадратична помилка *(Laplacian Mean Square Error)*[*]	$$LMSE = \sum_{x,y} \left(\nabla^2 C_{x,y} - \nabla^2 S_{x,y} \right)^2 \Big/ \sum_{x,y} \left(\nabla^2 C_{x,y} \right)^2. \quad (3.7)$$
Відношення сигнал/шум *(Signal to Noise Ratio)*	$$SNR = \sum_{x,y} \left(C_{x,y} \right)^2 \Big/ \sum_{x,y} \left(C_{x,y} - S_{x,y} \right)^2. \quad (3.8)$$
Максимальне відношення сигнал/шум *(Peak Signal to Noise Ratio)*	$$PSNR = XY \cdot \max_{x,y} \left(C_{x,y} \right)^2 \Big/ \sum_{x,y} \left(C_{x,y} - S_{x,y} \right)^2. \quad (3.9)$$
Якість зображення *(Image Fidelity)*	$$IF = 1 - \sum_{x,y} \left(C_{x,y} - S_{x,y} \right)^2 \Big/ \sum_{x,y} \left(C_{x,y} \right)^2. \quad (3.10)$$
Кореляційні показники спотворення	
Нормована взаємна кореляція *(Normalized Cross-Correlation)*	$$NC = \sum_{x,y} C_{x,y} \cdot S_{x,y} \Big/ \sum_{x,y} \left(C_{x,y} \right)^2. \quad (3.11)$$
Якість кореляції *(Correlation Quality)*	$$CQ = \sum_{x,y} C_{x,y} \cdot S_{x,y} \Big/ \sum_{x,y} C_{x,y}. \quad (3.12)$$

[*] $\nabla^2 \tilde{N}_{x,y} = \tilde{N}_{x+1,y} + \tilde{N}_{x-1,y} + \tilde{N}_{x,y+1} + \tilde{N}_{x,y-1} - 4 \cdot \tilde{N}_{x,y}$.

Інші показники	
Структурний зміст *(Structural Content)*	$$SC = \sum_{x,y}\left(C_{x,y}\right)^2 \Big/ \sum_{x,y}\left(S_{x,y}\right)^2. \quad (3.13)$$
Загальне сигма-відношення сигнал/шум *(Global Sigma Signal to Noise Ratio)*	$$GSSNR = \sum_{b}\sigma_b^2 \Big/ \sum_{b}\left(\sigma_b - \tilde{\sigma}_b\right)^2, \quad (3.14)$$ де $\sigma_b = \sqrt{\frac{1}{n}\cdot\sum_{\text{áëîê } b}\left(\tilde{N}_{x,y}\right)^2 - \left(\frac{1}{n}\cdot\sum_{\text{áëîê } b}\tilde{N}_{x,y}\right)^2}$.
Сигма-відношення сигнал/шум *(Sigma Signal to Noise Ratio)*	$$SSNR' = \frac{1}{n}\cdot\sum_{b} SSNR_b, \quad (3.15)$$ де $SSNR_b = 10\cdot\lg\left[\frac{\sigma_b^2}{\left(\sigma_b - \tilde{\sigma}_b\right)^2}\right]$.
Нормоване відношення сигма/помилка *(Normalized Sigma to Error Ratio)*	$$NSER = \frac{1}{\max\left(SER\right)}\cdot\sum_{b} SER_b, \quad (3.16)$$ де $SER_b = \sigma_b^2 \Big/ \left[\frac{1}{n}\cdot\sum_{\text{блок}\,b}\left(C_{x,y} - S_{x,y}\right)^2\right]$.
Подібність гістограм *(Histogram Similarity)*	$$HS = \sum_{c=0}^{255}\left\lvert f_C(c) - f_S(c)\right\rvert, \quad (3.17)$$ де $f_C(c)$ – відносна частота градації кольору c у зображенні з 256 рівнями кольорів.

У наведених співвідношеннях через $C_{x,\,y}$ позначається піксель порожнього контейнера з координатами (x, y), а через $S_{x,y}$ – відповідний піксель заповненого контейнера. У параметрах ***GSSNR*** , ***SSNR*** та ***SER*** аналізоване зображення попередньо розбивається на N блоків по n пікселів розміром $X \times Y$, де X і Y – відповідно, кількість рядків і стовпців у блоці (наприклад, блок 8×8 пікселів). Більш детальний опис показників можна отримати, зокрема, з [52].

Розглянуті вище показники базуються на аналізі окремих елементів сигналу (у даному випадку – пікселів зображення). Слабкі місця таких показників є відомими вже протягом тривалого часу (наприклад, відсутність корельованості різницевих показників спотворення із зором людини). В останнє десятиріччя все більше досліджень спрямовані на винайдення такого показника спотворення, який би був адаптований до людської зорової чи слухової системи шляхом врахування різноманітних впливів [53-56]. Розглянемо показник спотворення, який запропонували свого часу *J.E. Farrell* та *C.J. van den Branden Lambrecht* [55].

Ступінь сприйманої людиною якості оперує чутливістю до контрасту та явищем маскування системою візуалізації людини і базується на багатоканальній моделі людського просторового зору.

Обчислення даного показника вимагає наступних дій: проведення великокрокової сегментації зображення; розкладання помилки кодування і первинного зображення на перцепційні (такі що відносяться до сприйняття органами відчуттів) компоненти, використовуючи гребінчасті фільтри; обчислення порогу виявлення для кожного пікселя, використовуючи первинне зображення як маску; розподілення фільтрованої помилки за допомогою порогу прийняття

рішення, об'єднання по всім колірним каналам. Одиниця вимірювання показника визначається як *одиниця перевищення порогу*, що розуміє під собою *тільки значиму (помітну) відмінність* (*Just Noticeable Difference – JND*). Загальний показник, *приховане максимальне відношення сигнал/шум* (*Masked Peak Signal to Noise Ratio – MPSNR*):

$$\boldsymbol{MPSNR} = 10 \cdot \mathbf{lg}\left(\frac{255^2}{\boldsymbol{\varepsilon}^2} \right), \tag{3.18}$$

де $\boldsymbol{\varepsilon}$ – обчислене спотворення. Оскільки даний показник якості не відповідає смислу, який закладено у поняття *децибел*, його називають *візуальним* або *зоровим децибелом* (ВдБ).

У більшості випадків більш корисною є нормалізована оцінка якості. У [49] пропонується використовувати оцінку якості $\boldsymbol{Q}$ у відповідності до рекомендацій сектора радіозв'язку МСЕ – ITU-R Rec. 500:

$$\boldsymbol{Q} = \frac{5}{1 + \boldsymbol{N} \cdot \boldsymbol{\varepsilon}}, \tag{3.19}$$

де $\boldsymbol{\varepsilon}$ – обчислене спотворення; $\boldsymbol{N}$ – нормувальний коефіцієнт, який зазвичай обирається таким, щоб характеристика спотворення відображувала відповідну якісну оцінку. У табл.3.2 наведено оцінки і відповідні зорове сприйняття і якість.

Така оцінка має декілька переваг, зокрема, відсутність руйнування неспотворених зображень.

Таблиця 3.2.

ITU-R Rec. 500. Оцінки якості за шкалою від 1 до 5

Оцінка	*Спотворення*	*Якість*
5	Непомітне	Відмінна
4	Помітне, не подразнююче	Добра
3	Несуттєво подразнююче	Задовільна
2	Подразнююче	Незадовільна
1	Надзвичайно подразнююче	Вкрай незадовільна

3.5. Абсолютно надійна стеганосистема

У [3,15] наведене формальне теоретико-інформаційне визначення стійкості стеганосистеми стосовно пасивних атак. Головна ідея базується на випадковості обрання контейнера $\boldsymbol{c}$ з множини $\boldsymbol{C}$ з імовірністю $\boldsymbol{P_C}$.

Вбудовування до контейнеру секретного повідомлення можна описати як функцію, визначену на множині $\boldsymbol{C}$. Нехай $\boldsymbol{P_S}$ – імовірність формування стеганограми $\boldsymbol{E}(\boldsymbol{c}, \boldsymbol{m}, \boldsymbol{k})$ на множині $\boldsymbol{S}$ всіх можливих стеганограм, отриманих за допомогою стеганосистеми. Якщо контейнер $\boldsymbol{c}$ ніколи не використовується для отримання стеганограми, то $\boldsymbol{P_S}(\boldsymbol{c}) = 0$. Для обчислення імовірності $\boldsymbol{P_S}$ необхідно врахувати розподіл імовірностей на множині ключів $\boldsymbol{K}$ та множині повідомлень $\boldsymbol{M}$.

Визначимо на множині $\boldsymbol{Q}$ таке співвідношення для *відносної ентропії*, за допомогою якого можна виміряти неефективність прийняття невірної гіпотези про розподіл $\boldsymbol{P}_1$ у випадку істинного розподілу $\boldsymbol{P}_0$:

$$\boldsymbol{D}\left(\boldsymbol{P}_0 \| \boldsymbol{P}_1\right) = \sum_{\boldsymbol{q} \in \boldsymbol{Q}} \boldsymbol{P}_0(\boldsymbol{q}) \cdot \mathbf{log}_2\left(\frac{\boldsymbol{P}_0(\boldsymbol{q})}{\boldsymbol{P}_1(\boldsymbol{q})} \right), \tag{3.20}$$

де вираз $\mathbf{log}_2(\bullet)$ є логарифмічним відношенням правдоподібності.

Відносна ентропія між двома розподілами завжди є невід'ємною і дорівнює 0 лише у випадку тотожності даних розподілів. Таким чином, для стеганоперетворення можна отримати деяку оцінку. Наведемо визначення надійності стеганосистеми в термінах відносної ентропії.

Означення 3.1. Нехай Σ – стеганографічна система; P_S – розподіл імовірностей передачі каналом зв'язку стеганограм; P_C – розподіл імовірностей передачі каналом зв'язку пустих контейнерів. Система Σ називається ρ-*надійною* до пасивних атак, якщо $D(P_C \| P_S) \le \rho$, і є *абсолютно надійною*, якщо $\rho = 0$.

Як вже зазначалося, співвідношення $D(P_C \| P_S)$ дорівнює нулеві тільки у тому випадку, коли обидва розподіли імовірностей є рівними один одному. Отже, стеганосистема Σ є *теоретично абсолютно надійною*, якщо процес вбудовування секретного повідомлення до контейнеру не змінює розподіл P_C. Абсолютно безпечна система може бути створена, наприклад, на основі одноразової гамми [3].

На підставі сказаного, формулюється наступна теорема.

Теорема 3.1. Існує абсолютно надійна стеганосистема.

ДОВЕДЕННЯ. У [15] проведено конструктивне доведення даного твердження. Нехай контейнер C являє собою рівномірно розподілену n-бітову послідовність для деякого додатного n. Відправник за допомогою генератора ключа одержує рівномірно розподілений n-бітовий ключ K. Вважається, що функція вбудовування полягає у побітовому складанні за модулем 2 (операція XOR) n-бітового секретного повідомлення (в ролі якого в даному випадку виступає власне контейнер C) з ключем K: $S = C \oplus K$. Одержувач декодує отриману послідовність повторним застосуванням операції XOR : $C = S \oplus K$. Є цілком очевидним, що результуюча стеганограма S також являтиме собою рівномірно розподілену n-бітову послідовність. Отже, $P_C \sim P_S$, звідки $D(P_C \| P_S) = 0$. ТЕОРЕМУ ДОВЕДЕНО.

3.6. Стійкість стеганосистем до пасивних атак

Пасивний порушник намагається знайти відповідь на питання – містить перехоплений ним контейнер приховану інформацію чи ні. Для цього йому необхідно провести оцінювання неоднорідності деяких параметрів контейнера, виявити в ньому "підозрілі" ділянки, завищене зашумлення та інші сліди присутності прихованих повідомлень. Ця задача може бути формалізована у вигляді проблеми *перевірки статистичних гіпотез* [5]. З цією метою вводять тестову функцію стеганодетектора, яка в залежності від типу останнього може видавати дворозрядні (у більш складному випадку – μ-розрядні) рішення про наявність/відсутність вбудованого повідомлення:

$$D : \tilde{N} \to \{0;1\}, \tag{3.21}$$

$$D(C) = \begin{cases} 1, \text{ÿêùo êoímeéíep ìicmumü npuxoâaíe noâiäoìëeí íÿ;} \\ 0, \text{ÿêùo npuxoâaíe noâiäoìëeí íÿ â êoímeéíepi âiäñymío.} \end{cases}$$

За допомогою даної функції порушник здатен оцінювати повідомлення, перехоплені ним в несекретному каналі. В якості детектора прихованих повідомлень зазвичай використовують кореляційний приймач, зображений на рис.3.3.

Нехай в наслідок приховання у зображенні-контейнері повідомлення, в деякої частини пікселів значення яскравості було збільшено на 1, а в інших – лишилося незмінним, або ж було зменшене на 1. Тоді $s = c + m$, де $m = E(c, k)$. Корелятор детектора обчислює величину

$$s \cdot m = (c + m) \cdot m = c \cdot m + m \cdot m.$$

Оскільки m приймає значення ±1, то $\sum c \cdot m \to 0$, а $m \cdot m$ завжди буде додатнім. Тому $s \cdot m$ буде дуже

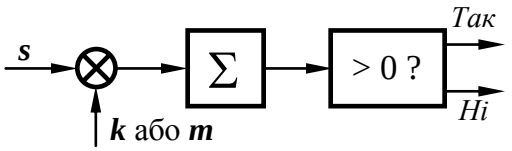


Рис.3.3. Кореляційний детектор прихованих повідомлень

близьким до $\boldsymbol{m}\cdot\boldsymbol{m}$. Тоді можна скористатися відомостями з теорії зв'язку і записати імовірність помилкового виявлення наявності прихованого повідомлення, як додаткову (комплементарну) функцію помилок від кореня квадратного з відношення $\boldsymbol{m}\cdot\boldsymbol{m}$ ("енергії сигналу") до дисперсії значень пікселів яскравості ("енергія шуму") [5].

В детекторі можливе виникнення двох типів помилок. Існує імовірність того, що при аналізі контейнера детектор не виявить наявного в ньому прихованого повідомлення (β-імовірність пропуску цілі або так звана *помилка 2-го роду*), а також імовірність помилкового виявлення прихованого повідомлення в пустому контейнері (α-імовірність помилкової тривоги або так звана *помилка 1-го роду*). Зниження однієї імовірності призводить до зростання іншої. На практиці намагаються максимізувати для пасивного атакуючого помилку 2-го роду. Ідеальна ж стеганосистема повинна забезпечувати помилку 2-го роду $\beta = 1$. Нижче буде показано, що всі абсолютно надійні стеганосистеми мають цю властивість (за умови, що атакуючий робить помилку 1-го роду з імовірністю $\alpha = 0$).

Для ρ-надійних стеганографічних систем імовірності α і β пов'язані між собою відповідно до поданої нижче теореми [15].

Теорема 3.2. Нехай Σ – стеганографічна система, яка є ρ-надійною проти пасивних атак. Тоді імовірність β того, що порушник не виявить приховане повідомлення, й імовірність α того, що він помилково виявить неіснуюче приховане повідомлення, задовольняють співвідношення $\boldsymbol{d}(\alpha, \beta) \le \rho$, де $\boldsymbol{d}(\alpha, \beta)$ – відносна двійкова ентропія, що визначається як

$$\boldsymbol{d}(\alpha,\beta) = \alpha\cdot\mathbf{log}_2\frac{\alpha}{1-\beta}+(1-\alpha)\cdot\mathbf{log}_2\frac{1-\alpha}{\beta}. \qquad (3.22)$$

Зокрема, якщо $\alpha = 0$, то $\beta \ge 2^{-\rho}$.

ДОВЕДЕННЯ. У випадку, коли контейнери *не містять* прихованого повідомлення, то їх розподіл імовірностей відповідає $\boldsymbol{P_C}$. Розглянемо випадкову величину $\boldsymbol{f}(\boldsymbol{C})$ і обчислимо її імовірність $\pi_{\boldsymbol{C}}$. Якщо $\boldsymbol{f}(\boldsymbol{C}) = 1$ (контейнер приймається за стеганограму), порушник робить помилку 1-го роду. Таким чином, $\pi_{\boldsymbol{C}}(1) = \alpha$ і $\pi_{\boldsymbol{C}}(0) = 1 - \alpha$.

Якщо контейнер *містить* приховане повідомлення, то розподіл імовірності відповідає $\boldsymbol{P_S}$. Обчислимо імовірність $\pi_{\boldsymbol{S}}$ для $\boldsymbol{f}(\boldsymbol{S})$. У випадку, коли $\boldsymbol{f}(\boldsymbol{S}) = 0$ (стеганограма приймається за порожній контейнер), порушник робить помилку 2-го роду. Отже, $\pi_{\boldsymbol{S}}(0) = \beta$ і $\pi_{\boldsymbol{S}}(1) = 1 - \beta$. Відносна ентропія $\boldsymbol{D}\left(\pi_{\boldsymbol{C}}\middle\|\pi_{\boldsymbol{S}}\right)$, відповідно до (3.20) може бути виражена в такий спосіб:

$$\boldsymbol{D}\left(\pi_{\tilde{N}}\middle\|\pi_{\boldsymbol{S}}\right) = \sum_{\boldsymbol{q}\in\{0;1\}}\pi_{\tilde{N}}(\boldsymbol{q})\cdot\mathbf{log}_2\frac{\pi_{\tilde{N}}(\boldsymbol{q})}{\pi_{\boldsymbol{S}}(\boldsymbol{q})} =$$

$$= \alpha\cdot\mathbf{log}_2\frac{\alpha}{1-\beta}+(1-\alpha)\cdot\mathbf{log}_2\frac{1-\alpha}{\beta} = \boldsymbol{d}(\alpha,\beta).$$

Відзначимо таку властивість функції відносної ентропії: детермінована обробка не може збільшити відносну ентропію між двома розподілами. Нехай $\boldsymbol{Q}_0$ і $\boldsymbol{Q}_1$ – дві випадкові величини, визначені на множині спостережень $\boldsymbol{Q}$ з відповідними розподілами імовірностей $\boldsymbol{P}_{\boldsymbol{Q}_0}$ і $\boldsymbol{P}_{\boldsymbol{Q}_1}$, а $\boldsymbol{f}$ – довільна функція відображення, що перетворює множину спостережень $\boldsymbol{Q}$ на множину спостережень $\boldsymbol{T}$ ($\boldsymbol{f}: \boldsymbol{Q} \to \boldsymbol{T}$). Тоді є справедливим наступний вираз:

$$\boldsymbol{D}\left(\boldsymbol{P}_{\boldsymbol{T}_0}\middle\|\boldsymbol{P}_{\boldsymbol{T}_1}\right) \le \boldsymbol{D}\left(\boldsymbol{P}_{\boldsymbol{Q}_0}\middle\|\boldsymbol{P}_{\boldsymbol{Q}_1}\right),$$

де через $\boldsymbol{T}_0$ і $\boldsymbol{T}_1$ ($\boldsymbol{T}_0, \boldsymbol{T}_1 \in \boldsymbol{T}$) відповідно позначені випадкові величини $\boldsymbol{f}(\boldsymbol{Q}_0)$ і $\boldsymbol{f}(\boldsymbol{Q}_1)$. Таким чином, $\boldsymbol{d}(\alpha,\beta) = \boldsymbol{D}\left(\pi_{\boldsymbol{C}}\middle\|\pi_{\boldsymbol{S}}\right) \le \boldsymbol{D}\left(\boldsymbol{P}_{\boldsymbol{C}}\middle\|\boldsymbol{P}_{\boldsymbol{S}}\right) \le \rho$. Враховуючи, що $\lim\limits_{\alpha\to 0}\mathbf{log}_2[\alpha/(1-\beta)] = 0$, отримуємо: $\boldsymbol{d}(0,\beta) = \mathbf{log}_2(1/\beta)$. Отже, якщо $\alpha = 0$, то $\beta \ge 2^{-\rho}$. ТЕОРЕМУ ДОВЕДЕНО.

Отже, для ρ-надійної стеганосистеми з $\alpha = 0$, необхідно забезпечити, щоб $\rho \to 0$. При цьому імовірність $\beta \to 1$, що еквівалентне неможливості виявлення порушником прихованого у контейнері повідомлення.

3.7. Активні і зловмисні атаки

Навіть за умови неможливості виділення і читання прихованого повідомлення, факт наявності останнього можна відносно легко виявити, ще легшою є операція знищення даного повідомлення. Наприклад, якщо повідомлення приховане у файлі BMP-формату методом заміни палітри, то вплив на цей файл випадковою зміною кольорів у палітрі зробить повідомлення таким, що не видобувається, іншими словами – знищить його.

Під час проектування та дослідження стеганографічних систем особлива увага повинна бути приділена вивченню впливу на них активних і зловмисних атак [3]. *Активні атаки* здатні змінити контейнер під час зв'язку: порушник може перехопити стеганограму, що була надіслана від передавальної сторони до приймальної, змінити її (шляхом певної обробки контейнера) і відправити результат приймальній стороні. У даному випадку передбачається, що при активній атаці неможливо повністю замінити контейнер і його семантику, а можна лише провести незначні заміни таким чином, щоб оригінал і змінений контейнер залишалися візуально і семантично подібними. *Зловмисними атаками* є такі, за яких існує можливість підміни повідомлення іншим, тобто організується фальсифікований стеганографічний обмін під ім'ям одного з партнерів зв'язку.

Стеганографічні системи є надзвичайно чутливими до модифікацій контейнера (наприклад, для зображення – це згладжування і фільтрація, для звуку – фільтрація). Так, просте ущільнення із втратами може призвести до повної втрати інформації, оскільки при цьому вилучаються незначущі компоненти сигналу і, тим самим, знищується секретна інформація, яка була в них прихована. Під час активних атак, коли немає можливості витягти приховану інформацію або довести її існування, її можна знищити простим додаванням у стеганограму випадкових завад. У разі цифрових зображень атакуючий може застосувати поширені методи обробки зображень, у тому числі змінити його формат.

У сучасних комп'ютерних системах реалізуються стеганографічні перетворення з високою надлишковістю, які є стійкими до трансформації контейнера (обертання, масштабування, друкування з наступним скануванням тощо). Тому однією з важливих вимог до прикладної стеганосистеми є забезпечення стійкості до випадкових або навмисних атак.

3.8. Стійкість стеганографічної системи до активних атак

Виходячи з розглянутих вище особливостей атак на стеганосистеми, можна зробити висновок, що протидія статистичному стеганоаналізу повинна полягати в побудові математичних моделей сигналів-контейнерів, пошуку на їх основі "дозволених" для модифікації областей і вбудовуванню в них приховуваної інформації, чия статистика не відрізняється від статистики контейнера. Така нерозрізненість визначає стійкість стеганосистеми

Як і для криптографічних систем захисту інформації, безпека стеганосистем описується й оцінюється їх *стійкістю* (стеганографічною стійкістю або скорочено стеганостійкістю). Але визначення стійкості і зламу даних систем є різними. В криптографії система захисту інформації є стійкою, якщо володіючи перехопленою криптограмою, порушник не здатний видобути інформацію, що в ній міститься. Неформально визначимо, що стеганосистема є стійкою, якщо порушник, спостерігаючи за інформаційний обміном між відправником і одержувачем, не здатен виявити, що під прикриттям контейнерів передаються приховувані повідомлення, і тим більше дізнатися про зміст останніх. У більш широкому смислі, під стійкістю стеганосистем розуміється їх здатність приховувати від кваліфікованого порушника факт прихованої передачі даних, здатність протистояти спробам порушника зруйнувати, спотворити, видалити приховано передавані повідомлення, а також спроможність підтвердити чи спростувати автентичність приховано передаваної інформації.

Стеганографічна система є стійкою до *активних атак*, якщо приховувана за її допомогою інформація не може бути змінена без *значних* змін контейнера, внаслідок яких останній втратить свою функціональність [3].

Означення 3.2. Нехай Σ – стеганографічна система і φ – клас відображень $C \to S$. Тоді система Σ буде φ-стійкою, якщо у випадку стеганосистем із секретним ключем для всіх $f \in \varphi$ є справедливим

$$D\{f[E(c, m, k), k]\} = D[E(c, m, k), k] = m, \tag{3.23}$$

а у випадку безключових стеганосистем, незалежно від вибору $m \in M$, $c \in C$ і $k \in K$:

$$D\{f[E(c, m)]\} = D[E(c, m)] = m. \tag{3.24}$$

Очевидно, що існує і зворотній взаємозв'язок між *надійністю* стеганосистеми та її *стійкістю:* чим більш стійкою до модифікацій контейнера буде стеганосистема, тим вона буде менш надійною, оскільки стійкість може бути досягнута завадостійким кодуванням, що може призвести до суттєвого спотворення контейнера і, можливо, до зміни розподілу імовірності P_S.

Багато стеганосистем є стійкими лише до певного класу відображень (ущільнення із втратами, геометричні перетворення, фільтрація, переквантування відліків, адитивний білий шум, перетворення ЦАП→АЦП тощо). Ідеальна стеганосистема повинна бути стійкою до всіх відображень типу "збереження λ-подібності", тобто відображенням $f: C \to S$ із властивістю $sim[c, f(c)] \to \lambda$ і $\lambda \approx 1$. Але такі системи є складними в проектуванні і, через необхідність додаткового застосування завадостійкого кодування, мають занадто низьку пропускну здатність. З іншого боку, система є "λ-слабкою", якщо для кожного контейнера існує таке відображення "збереження λ-подібності", що прихована інформація буде невідновлюваною з точки зору співвідношень (3.23) або (3.24).

У загальному випадку, існує два підходи до створення стійких стеганосистем [3]:

1) передбачаючи можливі атаки на стеганограми з боку порушників, стеганографічне перетворення відразу проектується стійким до знищення прихованих даних певним класом модифікацій;

2) реалізуються перетворення, які мають властивість оберненості до можливих модифікацій з метою відновлення початкового вигляду стеганограми. Так у [14] запропоновано метод "афінного кодування" для протидії афінним перетворенням зображення. При цьому передбачається оцінка параметрів перетворень, вимірювання змін форми, розмірів і напрямків деяких кодованих образів.

Стійкі алгоритми повинні приховувати дані в найбільш суттєвих фрагментах контейнера, оскільки інформація, яка кодується в шумовому компоненті, може бути вилучена без зусиль. Наприклад, відомо [57], що стеганографічні перетворення, які працюють з частотною областю контейнера, в переважній більшості випадків є більш стійкими до модифікацій, порівняно з алгоритмами, які працюють у просторовій (для зображення) або часовій (для звуку) областях. Використовуючи ці властивості, можна створити стійкі стеганосистеми, які, наприклад, зберігатимуть приховувану інформацію в коефіцієнтах дискретного косинусного перетворення (ДКП) зображення.

3.9. Свідомо відкритий стеганографічний канал

Вище було розглянуто моделі стеганосистем, при яких використовується секретний ключ (*private-key*), який розділяється між визначеними учасниками стеганографічного обміну. Подібні моделі є обмеженими для пасивного порушника, який, тим не менш, здатний виявити факт передачі прихованого повідомлення і за певних обставин дізнатися про його зміст. Інша справа, якщо порушник є активним або зловмисним. Тоді він не лише може вносити завади в стеганоканал, але й навіть повністю імітувати відправника. Оскільки у більшості випадків апріорна інформація про відправника в одержувача є відсутньою, він не здатний відрізнити фальсифікацію. Тому, здійснення прихованої передачі даних за наявності активного порушника є набагато складнішою проблемою, порівняно з фактом присутності пасивного порушника.

В роботах [5,12] представлено протокол, який дозволяє вирішити цю задачу. Даний протокол заснований на введенні до розгляду каналу з винятково малою пропускною здатністю – свідомо відкритого (*supraliminal*) каналу. Такий канал утворюється за рахунок вбудовування приховуваних даних у найбільш важливі ознаки контейнера, спотворення яких призведе до

повної деградації останнього. Іншими словами, інформація вбудовується в контейнер таким чином, що її видно, але неможливо змінити без істотних змін характерних властивостей контейнера.

Очевидна й сфера використання зазначеного протоколу – концепція відкритого стеганоканалу використовується переважно для вбудовування ЦВЗ. Розглянемо принцип його використання.

Припустимо, при активній атаці порушникові вдається внести лише незначні зміни в контейнер, що пересилається. Отже, деяка інформація, яка специфічна для конкретного контейнера, збережеться, оскільки її не можна видалити без істотної зміни семантики контейнера. Справа у тому, що порушник у багатьох випадках не може вносити значні завади в стеганоканал, настільки щоб передавана інформація була повністю спотворена. Не може через причини не технічного характеру, а з юридичних або інших міркувань. Тому, якщо секретне повідомлення вбудувати в істотні фрагменти контейнера, то його можна передавати між абонентами з високим ступенем цілісності навіть за наявності активних перешкод.

Вбудовування інформації до найбільш важливих елементів контейнера – основний принцип застосування ЦВЗ. Відмінність свідомо створюваного відкритого каналу полягає в тому, що для вбудовування і видобування даних не потрібний секретний ключ. Місце розміщення приховуваних біт є загальновідомим, але їх неможливо видалити без помітного руйнування контейнера. Крім того, ЦВЗ може не нести в собі жодної осмисленої інформації, наприклад, бути функцією самого зображення. У випадку ж свідомого каналу, навпаки, контейнер може бути функцією приховуваного короткого повідомлення.

Припустимо, що кожному контейнерові відповідає деякий шаблон, у якому формально описані всі характерні особливості контейнера [3]. Нехай $\boldsymbol{S}$ – множина всіх шаблонів і $\boldsymbol{f}: \boldsymbol{S} \to \{0; 1\}^{\boldsymbol{N}}$ – функція шаблонів. Для того, щоб передати бітовий рядок даних $\boldsymbol{m} \in \{0; 1\}^{\boldsymbol{N}}$, передавальна сторона обирає з множини $\boldsymbol{S}$ деякий шаблон $\boldsymbol{s} \in \boldsymbol{f}^{-1}(\boldsymbol{m})$ і надсилає відкритим каналом контейнер, якому відповідає цей шаблон. Пасивний порушник може підозрювати про існування прихованого обміну в шумовому компоненті контейнера і з метою руйнування секретного повідомлення може дещо змінити стеганограму. Однак при цьому він не в змозі змінити шаблон контейнера. В свою чергу приймальна сторона може відновити шаблон $\boldsymbol{s}$ із прийнятої стеганограми (можливо, модифікованої пасивним порушником) і видобути дані $\boldsymbol{m}$ за допомогою функції $\boldsymbol{f}$.

Використовувати відкритий стеганоканал для пересилання бітового рядка $\boldsymbol{m}$ з явним змістом недоцільно, оскільки пасивний порушник за допомогою відкритої функції $\boldsymbol{f}$ може легко відновити вбудовану інформацію. Однак, якщо повідомлення $\boldsymbol{m}$ є випадковим секретним ключем або виглядає як випадковий шифртекст (тобто попередньо захищене криптографічно), то у порушника не буде підстав для підозри і доказів (якщо він не здатний зламати криптосистему), що передана інформація є чимось більш істотним, ніж випадковий бітовий рядок.

Для практичної реалізації даного протоколу необхідно створити низку умов. Очевидно, що основні труднощі полягають у формуванні контейнера, тоді як робота по видобуванню даних може бути легко автоматизована. Для практичного ж застосування свідомо відкритого каналу повинні бути автоматизовані обидві операції. По-перше, повинна існувати можливість створення для будь-якого шаблона такого контейнеру, невеликі зміни якого при активних атаках не призводитимуть до зміни прихованих даних. При цьому у пасивного порушника не повинно існувати можливості шляхом маніпуляцій зі стеганограмою змінити шаблон $\boldsymbol{s}$ і привести його до такого вигляду $\boldsymbol{s}'$, що $\boldsymbol{f}(\boldsymbol{s}) \neq \boldsymbol{f}(\boldsymbol{s}')$. По-друге, повинна існувати можливість формування шаблону для кожного одержаного контейнера. Крім того, функція $\boldsymbol{f}$ має бути загальнодоступною, а $\boldsymbol{f}$ і $\boldsymbol{f}^{-1}$ – обчислюваними. Видобування повідомлення з пустого контейнера повинне повертати випадковий рядок даних. Отже, єдина відмінність між заповненим і порожнім контейнерами полягає у тому, що рядок $\boldsymbol{f}(\boldsymbol{s})$ має певне значення.

Описана схема може бути також застосована для прихованого обміну ключами. Протокол обміну наступний [12].

Передавальна сторона генерує пару відкритого ($\boldsymbol{E}$) і секретного ($\boldsymbol{D}$) ключів; обчислює представницький опис (шаблон) контейнеру, що відповідає ключу $\boldsymbol{E}$: $\boldsymbol{s_E} \in \boldsymbol{f}^{-1}(\boldsymbol{E})$; генерує контейнер, який відповідає $\boldsymbol{s_E}$, і надсилає його приймальній стороні.

Приймальна сторона видобуває з прийнятого контейнера відкритий ключ $\boldsymbol{E}$ передавальної сторони: $\boldsymbol{E} \in \boldsymbol{f}(\boldsymbol{s_E})$; генерує свій секретний ключ $\boldsymbol{K}$; шифрує його за допомогою відкритого ключа

$\boldsymbol{E}$; знаходить відповідний отриманій послідовності $\boldsymbol{K_E}$ опис (шаблон) контейнера: $\boldsymbol{s_{KE} \in f^{-1}(K_E)}$; генерує контейнер, який відповідає $\boldsymbol{s_{KE}}$, і надсилає його передавальній стороні.

Передавальна сторона відновлює зашифрований секретний ключ: $\boldsymbol{K_E \in f(s_{KE})}$ і розшифровує його, використовуючи свій секретний ключ $\boldsymbol{D}$.

Тепер сторони можуть обмінюватися повідомленнями, вбудовуваними до контейнеру з використанням секретного ключа $\boldsymbol{K}$. Порушник в результаті перехоплення каналу може отримати відкритий ключ передавальної сторони і зашифрований цим ключем секретний ключ приймальної сторони. Значення останнього без знання секретного ключа передавальної сторони залишатиметься для нього невідомим.

Слід зазначити, що головна проблема схеми відкритого стеганографічного каналу полягає в ефективній реалізації функції $\boldsymbol{f}$. Крім того, такий канал не підходить для прихованої передачі повідомлень великого обсягу, оскільки він має низьку пропускну здатність і є відкритим для порушника.

L. von Ahn та *N.J. Hopper* [58] формалізували стеганографію з відкритим ключем у випадку пасивного порушника, а також створили обмежену модель за наявності порушника, який здійснює активну атаку на стеганоканал. На їх погляд, необхідно забезпечувати стійкість проти атак конкретних (заздалегідь визначених) порушників, у тих випадках, однак, коли одержувач повинен бути упевнений в автентичності відправника. Це, на думку авторів [59], є обмеженням моделі порівняно із принципами, закладеними у процес обміну з відкритим ключем. У своїй роботі вони пропонують комплексну теоретичну модель протоколу стеганографічного обміну з відкритим ключем в разі активних атак, причому особи, які беруть у цьому участь, апріорі не потребують розділення секретної інформації, а порушник може впливати на канал і встановлювати так звану *адаптивну до контейнерів атаку*. Такий вид атаки бачиться найбільш узагальнюючим проти стеганосистем, протоколи яких побудовані з використанням відкритості ключа. Це дозволяє порушнику надсилати приймальній стороні довільну послідовність адаптивно обраних вбудованих у контейнер повідомлень і вивчати інтерпретацію кожного з повідомлень; тобто, якщо одержувач розглядає повідомлення як пустий контейнер або як стеганограму і видобуває вбудоване повідомлення в останньому випадку. Описана у [59] модель побудована на припущенні, що стеганосистема з відкритим ключем за своєю сутністю є криптографічною системою з відкритим ключем з додатковою необхідною умовою, що результат її роботи (стеганограма) повинна відповідати розподілові використаного при цьому контейнера.

3.10. Висновки

У даному розділі шляхом дослідження відомих публікацій вітчизняних і закордонних авторів здійснено системне викладення питань надійності і стійкості довільної стеганографічної системи по відношенню до видів здійснюваних на неї атак. Останні було виділено у відповідності до класифікації типів порушників на пасивних, активних і зловмисних.

Проведення аналогії між стегано- і криптоаналізом дозволило виділити як спільні, так і характерні лише для стеганосистем види атак (атака на основі відомого порожнього контейнера, атака на основі обраного порожнього контейнера, атака на основі відомої математичної моделі контейнера або його частини).

Було здійснено огляд низки публікацій, присвячених розглядові показників, використовуваних з метою оцінювання спотворень, що вносяться стеганоперетвореннями у піксельну структуру контейнера.

4. ПРОПУСКНА ЗДАТНІСТЬ КАНАЛІВ ПЕРЕДАЧІ ПРИХОВУВАНИХ ДАНИХ

4.1. Поняття пропускної здатності

Для розроблюваних або досліджуваних стеганографічних систем важливо визначити, наскільки великою може бути *пропускна здатність* створюваних при цьому каналів передачі приховуваних даних (КППД) і як вона залежатиме від інших характеристик стеганосистем та умов їх використання. Під пропускною здатністю каналів передачі приховуваних даних або просто *прихованою пропускною здатністю* (ППЗ) розуміють максимальну кількість інформації, яка може бути вбудована до одного елементу (наприклад, пікселя чи відліку) контейнера. Обов'язковою умовою при цьому є безпомилковість передачі приховуваних даних одержувачеві, а також їх захищеність від таких атак порушника, як спроби виявлення факту наявності каналу прихованого зв'язку, одержання змісту приховуваних повідомлень, навмисне введення сфальсифікованих даних або ж руйнування вбудованої до контейнера інформації [5].

Канал прихованого зв'язку (КПЗ) утворюється всередині каналу відкритого зв'язку (КВЗ), для якого *C.E. Shannon* у своїх роботах з теорії інформації визначив пропускну здатність [45,60,70]. Пропускна здатність КВЗ визначається як кількість інформації, яку потенційно можна передати без помилок за одне використання каналу. При цьому не висувається жодних вимог до захищеності від атак організованого порушника. Тому цілком логічним буде припущення, що прихована пропускна здатність КПЗ, в якому за одне використання каналу передається один елемент контейнера із вкладеною у нього прихованою інформацією, у жодному випадку не може бути більшою за пропускну здатність КВЗ.

На сьогодні намітилися різні, іноді діаметрально протилежні підходи до визначення кількості інформації, яка підлягає захистові від різноманітних атак порушника власне за допомогою стеганографічних методів. Дані розходження, як зазначається у [5], зумовлені відмінністю у цілях захисту інформації, видами порушника, його можливостями та реалізованими ним атаками на стеганосистеми, видом використовуваних контейнерів і приховуваних повідомлень і багатьма іншими факторами. У тій самій роботі пропонується здійснити оцінювання величину ПЗ КППД методами теорії інформації для різних стеганосистем. Теоретико-інформаційні методи дозволяють одержати строгі оцінки кількості приховуваної інформації, які цілком правомірно можуть бути використані як теоретично досяжні граничні швидкості передачі приховуваної інформації для стеганосистем, не зважаючи на принципи, що закладені до основи їхньої побудови.

Пропонується розглянути *два основних підходи до оцінки пропускної здатності КППД*.

Перший з них, розвинутий у робо*т*ах [61,62], орієнтований на стеганографічні системи, в яких повідомлення, що підлягають захистові, повинні бути безпомилково передані в умовах активної протидії порушника. Даний підхід описує сценарій приховання так званих безнадлишкових повідомлень у даних контейнера, і, що найголовніше, дозволяє врахувати той факт, що крім спотворювань структури контейнера при вбудовуванні до нього приховуваних даних, можливі його навмисні спотворювання з боку порушника, крім того, існує ще й імовірність спотворень випадкового характеру, викликаних ненавмисними завадами каналу зв'язку.

Порушник крім *пасивних* дій аналізу може використати й *активні* дії (атакуючий порушник). Метою атакуючого порушника є руйнування приховуваної інформації. Така постановка завдання інформаційного приховання є характерною, наприклад, для систем ЦВЗ.

У [5] сформульовано завдання інформаційного приховання як завдання безпомилкової передачі приховуваної інформації при впливі випадкових і навмисних завад та визначено максимальну швидкість безпомилкової передачі при різних стратегіях дій відправника й атакуючого. Пропонований підхід визначає *теоретично досяжну* швидкість достовірної передачі приховуваних повідомлень, хоча в явному виді й не оцінює захищеність останніх від виявлення факту їх існування. Але для низки стеганосистем не потрібно приховувати факт використання стеганографічного захисту: власник авторських або майнових прав на медіаконтейнер,

що захищений ЦВЗ, як правило, відкрито повідомляє про застосування даної системи захисту. У розглянутому підході досліджуються умови, при яких приховувана інформація гарантовано передається в умовах довільних спроб порушника щодо її руйнування.

Знання параметрів стеганосистеми і можливих стратегій дій передавальної сторони не повинне дозволити порушникові оптимізувати руйнуючий вплив й оцінити його ефективність. Особливістю таких стеганосистем є, по-перше, те, що руйнуючий вплив відбувається лише в момент передачі прихованих даних і повинен виконуватися в режимі реального часу. По-друге, існує апріорна непоінформованість законного одержувача щодо приховано передаваної йому інформації. По-третє, порушник у переважній більшості випадків не здатен достовірно оцінити ефективність своїх дій.

Інша ситуація виникає при намаганні активного порушника зруйнувати ЦВЗ, з метою привласнити собі контейнер (права на нього). Порушник може як завгодно довго здійснювати руйнуючий вплив, вибираючи таку оптимальну стратегію, за якої, зруйнувавши ЦВЗ, він збереже необхідну йому якість контейнера. При цьому порушник заздалегідь знає про існування прихованої інформації, і використовуючи загальновідомий детектор (див. рис.3.3), здатний оцінити ефективність своїх атак на ЦВЗ.

Другий підхід, що пропонується, наприклад, авторами робіт [19,44,46], дає оцінки прихованої пропускної здатності безпосередньо в процесі вбудовування приховуваних повідомлень у надлишкові дані контейнера. Такий підхід враховує, що контейнери формуються реальними надлишковими джерелами з істотною пам'яттю, такими як джерела зображень або аудіосигналів. У цьому випадку оцінки ПЗ залежать від характеристик замаскованості прихованого каналу. Такий підхід є орієнтованим на стеганосистеми, в яких реалізується прихована передача апріорно невідомої одержувачеві інформації, причому пасивний порушник намагається в процесі спостереження за каналом відкритого зв'язку виявити факт наявності КПЗ і, в разі встановлення цього факту, прагне розкрити зміст прихованого повідомлення у перехопленому контейнері.

Відома велика кількість робіт із синтезу стеганосистем, автори яких пропонують різні способи вбудовування даних у надлишкові за своєю природою контейнери [21,24,63,64]. При цьому кількість інформації, вбудованість якої залишатиметься непомітною, оцінюється за допомогою додатково введених критеріїв рівня прихованості (див. попередній розділ). Існуючі на сьогодні оцінки ППЗ таких стеганоканалів, однак, не враховують можливі випадкові й навмисні спотворення контейнерів при їх передачі каналом зв'язку.

4.2. Інформаційне приховання при активній протидії порушника

У рамках першого підходу до оцінки ППЗ, розглянемо загальне формулювання завдання інформаційного приховання у випадку активної протидії з боку порушника. Основні результати цього підходу були отримані в роботі [61] і застосовані у [5]. Наведемо деякі з них.

4.2.1. Формулювання завдання інформаційного приховання при активній протидії порушника

Розглянемо узагальнену структурну схему стеганографічної системи передачі прихованих повідомлень, представлену на рис.4.1.

У даній схемі приховувані повідомлення $\boldsymbol{m}$ рівномірно розподілені у множині повідомлень $\boldsymbol{M}$ і повинні бути безпомилково передані декодеру. Передавальна сторона подає *порожній контейнер* $\boldsymbol{c}^N$ (який являє собою послідовність з $\boldsymbol{N}$ незалежно й ідентично розподілених відліків у відповідності з роз-

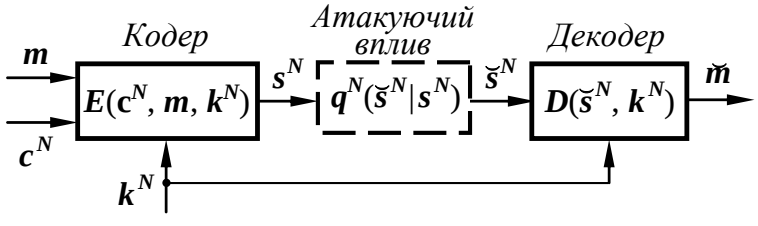


Рис.4.1. Узагальнена структурна схема стеганосистеми при активній протидії порушника

поділом контейнера $\boldsymbol{p}(\boldsymbol{c})$), *секретний ключ* $\boldsymbol{k}^N$ (кожен символ $\boldsymbol{k_i}$ якого незалежно і рівноімовірно розподілений за функцією $\boldsymbol{p}(\boldsymbol{k})$), і *повідомлення* $\boldsymbol{m}$ на вхід кодера. Останній формує *стеганограму* $\boldsymbol{s}^N$, яка передається одержувачеві незахищеним каналом зв'язку.

Стеганограма $\boldsymbol{s}^N$ перехоплюється й обробляється порушником з метою руйнування або видалення повідомлення $\boldsymbol{m}$. Спотворену порушником стеганограму позначимо $\breve{\boldsymbol{s}}^N$, а атакуючий вплив позначимо умовною функцією розподілу $\boldsymbol{q}^N\left(\breve{\boldsymbol{s}}^N \middle| \boldsymbol{s}^N\right)$. Ця обробка включає, як окремий випадок, формування спотвореної стеганограми у вигляді $\breve{\boldsymbol{s}}^N = \boldsymbol{q}^N\left(\boldsymbol{s}^N\right)$, де $\boldsymbol{q}^N$ – детерміноване відображення.

Основне припущення – порушник знає розподіли всіх змінних у стеганосистемі та, власне, опис стеганосистеми, але не знає використовуваного секретного ключа (принцип Керхгофса для систем захисту інформації).

Нехай контейнер $\boldsymbol{c}$, стеганограма $\boldsymbol{s}$ і модифікована порушником стеганограма $\breve{\boldsymbol{s}}$ належать одній множині $\boldsymbol{C}$ ($\boldsymbol{c}, \boldsymbol{s}, \breve{\boldsymbol{s}} \in \boldsymbol{C}$). Декодер одержувача обчислює оцінку $\breve{\boldsymbol{m}}$ первинного прихованого повідомлення $\boldsymbol{m}$. Якщо $\boldsymbol{m} \neq \breve{\boldsymbol{m}}$, то атакуючий зумів зруйнувати інформацію, яка захищалася стеганографічною системою.

Формально визначимо внесені спотворювання в стратегіях передавальної сторони і порушника. Це завершує математичний опис стеганосистеми і дозволяє визначити швидкість безпомилкової передачі для схеми, представленої на рис.4.1.

Означення 4.1. Функція спотворення, що вноситься відправником повідомлення, являє собою невід'ємну функцію $\boldsymbol{d}_1$: $\boldsymbol{C} \times \boldsymbol{S} \to \Re_+$.

Означення 4.2. Функція спотворення, що вноситься атакуючою стороною, являє собою невід'ємну функцію $\boldsymbol{d}_2$: $\boldsymbol{S} \times \breve{\boldsymbol{S}} \to \Re_+$.

Функція спотворення $\boldsymbol{d}_1$ є обмеженою: $\boldsymbol{d}_{1,\max} = \max\limits_{(\boldsymbol{c},\boldsymbol{s}) \in \tilde{\boldsymbol{N}} \times \boldsymbol{S}} \boldsymbol{d}_1(\boldsymbol{c},\boldsymbol{s}) < \infty$. Крім того, дана міра спотворення є симетричною: $\boldsymbol{d}_1(\boldsymbol{c}, \boldsymbol{s}) = \boldsymbol{d}_1(\boldsymbol{s}, \boldsymbol{c})$ для всіх $\boldsymbol{c}, \boldsymbol{s} \in \boldsymbol{C} = \boldsymbol{S}$. Виконання рівності $\boldsymbol{d}_1(\boldsymbol{c}, \boldsymbol{s}) = 0$ означає збіг: $\boldsymbol{c} = \boldsymbol{s}$. Якщо $\boldsymbol{d}_1(\boldsymbol{c}, \boldsymbol{s}) = 1$, то контейнер-результат не відповідає контейнеру-оригіналу.

Функції спотворення $\boldsymbol{d_i}$, $\boldsymbol{i} \in \{1; 2\}$ поширюються на спотворення символьних послідовностей з довжиною блоків $\boldsymbol{N}$:

$$\boldsymbol{d}_{\boldsymbol{i}}^{\boldsymbol{N}}(\boldsymbol{x}^N, \boldsymbol{y}^N) = \boldsymbol{N}^{-1} \cdot \sum_{\boldsymbol{j}=1}^{\boldsymbol{N}} \boldsymbol{d_i}(\boldsymbol{x_j}, \boldsymbol{y_j}).$$

Назвемо спотворення контейнера $\boldsymbol{c}$, викликане вбудовуванням у нього приховуваного повідомлення $\boldsymbol{m}$, *спотвореннам, викликаним кодуванням*, а спотворення, викликане атакуючими діями порушника – *спотвореннам, викликаним атакуючим впливом*.

Означення 4.3. Стеганосистема з довжиною блоку $\boldsymbol{N}$, що призводить до спотворення, викликаного *кодуванням*, і рівень якого не перевищує $\boldsymbol{A}_1$, є сукупністю множин приховуваних повідомлень $\boldsymbol{M}$ з кількістю елементів (потужністю) $|\boldsymbol{M}|$, контейнерів $\boldsymbol{C}$, стеганограм $\boldsymbol{S} \sim \breve{\boldsymbol{S}} \sim \boldsymbol{C}$ і ключів $\boldsymbol{K}$, а також визначених на них функцій кодування $\boldsymbol{E}$ і декодування $\boldsymbol{D}$. Причому $\boldsymbol{E}$ – відображення контейнера $\boldsymbol{c}^N$, повідомлення $\boldsymbol{m}$ і ключа $\boldsymbol{k}^N$ у стеганограму: $\boldsymbol{E} : \boldsymbol{C} \times \boldsymbol{M} \times \boldsymbol{K} \to \boldsymbol{C}$, $\boldsymbol{s}^N = \boldsymbol{E}(\boldsymbol{c}^N, \boldsymbol{m}, \boldsymbol{k}^N)$. Це відображення є обмеженим величиною середнього спотворення $\boldsymbol{A}_1$, викликаного кодуванням:

$$\sum_{\boldsymbol{c}^N \in \tilde{\boldsymbol{N}}} \sum_{\boldsymbol{m} \in \boldsymbol{M}} \sum_{\boldsymbol{k}^N \in \boldsymbol{K}} |\boldsymbol{M}|^{-1} \cdot \boldsymbol{p}\left(\boldsymbol{c}^N, \boldsymbol{k}^N\right) \cdot \boldsymbol{d}_1^N\left[\boldsymbol{c}^N, \boldsymbol{E}\left(\boldsymbol{c}^N, \boldsymbol{m}, \boldsymbol{k}^N\right)\right] \leq \boldsymbol{A}_1. \qquad (4.1)$$

Відображення $\boldsymbol{D} : \boldsymbol{C} \times \boldsymbol{K} \to \breve{\boldsymbol{M}}$ є декодувальним відображенням прийнятої стеганопослідовності $\breve{\boldsymbol{s}}^N$ і ключа $\boldsymbol{k}^N$ у декодоване повідомлення $\breve{\boldsymbol{m}} = \boldsymbol{D}(\breve{\boldsymbol{s}}^N, \boldsymbol{k}^N)$.

Таким чином, величина $\boldsymbol{A}_1$ характеризує ступінь спотвореності контейнера, який є максимально припустимим при вбудовуванні до контейнера приховуваного повідомлення. Не дивлячись на те, що дане визначення формально описує стеганосистеми *блокового* типу, на практиці воно може бути розширене і на стеганосистеми *потокового* типу, у яких вікно обробки описується ковзним блоком довжиною $\boldsymbol{N}$. У цьому випадку параметр $\boldsymbol{N}$ стеганосистеми може

бути названий довжиною кодового обмеження стеганосистеми (за аналогією з безперервними кодами).

У більшості випадків спотворення A_1 є малим, оскільки апріорно приймається, що результат вбудовування до контейнера повідомлення повинний бути непомітним для сторонньої особи (у тому числі й порушника). У стеганосистемах, в яких контейнер являє собою корисний для одержувача інформаційний сигнал і якість якого необхідно зберегти, величина A_1 обмежується. У системах ЦВЗ вимога мінімізації A_1 формулюється як вимога прозорості водяного знака, що засвідчує приналежність контейнера [5].

Крім того, визначення обмеження спотворення (4.1) містить усереднення по відношенню до розподілу $p(c^N, k^N)$ і по відношенню до рівномірного розподілу повідомлень. Такий вибір зроблено для зручності, оскільки це дозволяє використовувати класичні положення теорії Шеннона [60].

Розподіл $p(c^N, k^N)$ та обрання відображення E визначають конкретний вигляд розподілу $p(s^N)$ множини формованих стеганограм.

Означення 4.4. *Атакуючий вплив* (без пам'яті), що приводить до спотворення A_2, описується умовною функцією розподілу $q^N(\breve{s}^N \mid s^N)$ з множини S до множини $\breve{S}$, такою що

$$\sum_{s^N \in S} \sum_{\breve{s}^N \in \breve{S}} d_2^N\left(s^N, \breve{s}^N\right) \cdot q^N\left(\breve{s}^N \middle| s^N\right) \cdot p\left(s^N\right) \le A_2 . \qquad (4.2)$$

За визначенням, A_2 є максимальною величиною спотворення стеганограми, яке було викликане навмисними діями порушника. Фізичний зміст обмеження величини A_2 полягає в наступному. У системах ЦВЗ порушник, намагаючись видалити водяний знак із завіреного контейнера, змушений сам зменшувати величину A_2, щоб істотно не спотворити цінний для нього контейнер. В інших стеганосистемах величина A_2 обмежується наявним в атакуючого енергетичним потенціалом встановлення завад, виникаючими завадами для інших каналів зв'язку при використанні спільного ресурсу та з інших причин.

Логічним є припущення, що для реальних стеганосистем зазвичай виконується співвідношення $A_2 \ge A_1$.

Відповідно до означення 4.4, атакуючий вплив описується й обмежується усередненими спотвореннями між множинами S і $\breve{S}$. В інших випадках, якщо атакуючий знає опис функції E, то атакуючий вплив описується і обмежується усередненим спотворенням між множинами C і $\breve{S}$:

$$\sum_{\substack{c^N, m, \\ k^N, \breve{s}^N}} d^N\left(c^N, \breve{s}^N\right) \cdot q^N\left[\breve{s}^N \middle| E\left(c^N, m, k^N\right)\right] \cdot p\left(c^N, k^N\right) \le A_2 . \qquad (4.3)$$

Визначення A_2 відповідно до виразу (4.3) припускає, що порушникові відомі точні імовірнісні характеристики контейнерів. Як буде показано далі, ця обставина істотно ускладнює завдання забезпечення захищеності приховуваної інформації, тому в стійких стеганосистемах використаються різні методи приховання від порушника характеристик використовуваних контейнерів. Наприклад, такі методи включають використання для вбудовування підмножини контейнерів з імовірнісними характеристиками, що відрізняються від характеристик усієї множини відомих порушникові контейнерів або рандомізована компресія сигналу контейнера перед вбудовуванням до нього приховуваного повідомлення [15]. Тому обчислення спотворення A_2 у відповідності до означення 4.4 є більше універсальним, оскільки порушник завжди має можливість вивчати імовірнісні характеристики спостережуваних стеганограм.

Маючи опис стеганосистеми і атакуючого впливу $q^N(\breve{s}^N \mid s^N)$, можна описати змагання (гру) між передавальною і атакуючою сторонами.

Означення 4.5. *Інформаційно-приховуюче змагання*, що призводить до спотворювань (A_1, A_2), описується взаємодією використовуваної стеганосистеми, що спричиняє спотворення кодування A_1, і атакуючого впливу, що викликає спотворення A_2.

Швидкість передачі приховуваних повідомлень по стеганоканалу визначається у вигляді $\boldsymbol{R} = \boldsymbol{N}^{-1}\cdot\log|\boldsymbol{M}|$. При цьому швидкість передачі $\boldsymbol{R}$ виражається через середню кількість біт приховуваного повідомлення, які безпомилково передаються (переносяться) одним символом (пікселем, відліком) стеганопослідовності $\boldsymbol{s}^{\boldsymbol{N}}$. Це визначення є співзвучним "класичному" визначенню швидкості передачі звичайних повідомлень каналом відкритого зв'язку, яка виражається в середній кількості безпомилково переданих біт за одне використання каналу [60,65,70].

Імовірність руйнування приховуваного повідомлення (середню імовірність помилки) в стеганопослідовності довжиною $\boldsymbol{N}$ визначають як

$$\boldsymbol{P}_{\text{br.}}^{\boldsymbol{N}} = |\boldsymbol{M}|^{-1}\cdot\sum_{\boldsymbol{m}\in\boldsymbol{M}}\boldsymbol{P}\left[\boldsymbol{D}\left(\tilde{\boldsymbol{S}},\boldsymbol{K}\right)\neq\boldsymbol{m}\middle|\boldsymbol{M}=\boldsymbol{m}\right], \tag{4.4}$$

де приховувані повідомлення $\boldsymbol{m}$ рівноімовірно обираються серед множини $\boldsymbol{M}$. Імовірність $\boldsymbol{P}_{\text{br.}}^{\boldsymbol{N}}$ є середньою імовірністю того, що атакуючий успішно спотворить приховано передаване повідомлення, усередненою на множині всіх повідомлень. Атакуючий зазнає успіху в інформаційному змаганні, якщо декодоване під час прийому повідомлення не збігається із вбудованим у контейнер приховуваним повідомленням, або ж декодер не здатен прийняти однозначне рішення.

Теоретично досяжну швидкість безпомилкової передачі приховуваних повідомлень і приховану пропускну здатність при спотвореннях не більше за $(\boldsymbol{A}_1, \boldsymbol{A}_2)$, пропонується визначити в такий спосіб.

Означення 4.6. Швидкість $\boldsymbol{R}$ безпомилкової передачі приховуваних повідомлень є досяжною для спотворень не більше за $(\boldsymbol{A}_1, \boldsymbol{A}_2)$, якщо існує стеганосистема з довжиною блоку $\boldsymbol{N}$, яка спричиняє спотворення кодування не більше $\boldsymbol{A}_1$ на швидкості $\boldsymbol{R}_{\boldsymbol{N}} > \boldsymbol{R}$, така що $\boldsymbol{P}_{\text{br.}}^{\boldsymbol{N}} \to 0$ при $\boldsymbol{N} \to \infty$ за будь-яких атак порушника, що призводять до спотворень не більше $\boldsymbol{A}_2$.

Означення 4.7. Прихована пропускна здатність $\boldsymbol{B}(\boldsymbol{A}_1, \boldsymbol{A}_2)$ є *супремумом* (верхньою межею) всіх досяжних швидкостей безпомилкової передачі приховуваних повідомлень при спотвореннях не більше за $(\boldsymbol{A}_1, \boldsymbol{A}_2)$.

Таким чином, ППЗ є *верхньою межею* швидкості безпомилкової передачі приховуваних даних, за якої спотворення контейнера, викликані вбудовуванням у нього зазначених повідомлень$(\boldsymbol{A}_1)$ і діями порушника по руйнуванню цих повідомлень$(\boldsymbol{A}_2)$, не перевищують заданих величин.

Як і ПЗ каналів передачі відкритих повідомлень, ПЗ каналів передачі приховуваних повідомлень визначається в ідеалізованих умовах, при яких затримка кодування / декодування є нескінченою (тобто $\boldsymbol{N} \to \infty$), статистика контейнерів, приховуваних повідомлень, стеганограм і ключів є точно відомою, складність побудови стеганосистеми є необмеженою.

Є цілком очевидним, що така пропускна здатність каналу прихованого зв'язку має зміст *теоретичної межі*, яка вказує області, в яких існують і, відповідно, не існують стеганосистеми при заданих величинах спотворювань. Відомо, що швидкості реальних систем передачі відкритих повідомлень можуть лише наближатися до величини ПЗ відкритих каналів, причому по мірі наближення до неї обчислювальна складність реалізації систем передачі зростає спочатку приблизно за *лінійною*, потім за *квадратичною* і далі за *експонентною* залежністю від довжини блоку кодування $\boldsymbol{N}$ [60]. Цілком імовірно, аналогічні залежності зростання складності є справедливими і для стеганосистем у міру наближення швидкості передачі приховуваних даних до величини ППЗ. Це припущення підтверджується наявним досвідом побудови стеганосистем. Відомо, що спроби збільшити швидкість передачі приховуваних даних спричиняють істотне ускладнення методів приховання інформації [63,66].

4.2.2. Приховуюче перетворення

Для повного представлення стеганосистеми й умов її функціонування наведемо формальний опис *приховуючого перетворення*, виконуваного при вбудовуванні інформації до контейнера, і *атакуючого впливу*, здійснюваного порушником для протидії прихованій передачі.

Для цього розглянемо допоміжну випадкову послідовність $\boldsymbol{u}$, визначену на множині $\boldsymbol{U}$. Фізично послідовність $\boldsymbol{u}$ описує результат перетворення приховуваного повідомлення $\boldsymbol{m}$ з метою його адаптації до вбудовування в контейнер заданого формату. Слід зауважити, що в той час як у стеганосистемі контейнери, ключі й стеганограми являють собою послідовності однакової довжини $\boldsymbol{N}$, довжина приховуваних повідомлень, їхній алфавіт та імовірнісний розподіл у переважній більшості випадків не збігаються з відповідними характеристиками зазначених послідовностей.

Визначимо допоміжну множину $\boldsymbol{O} = \{(\boldsymbol{c}, \boldsymbol{k}) \in \boldsymbol{C} \times \boldsymbol{K}: \boldsymbol{p}(\boldsymbol{c}, \boldsymbol{k}) > 0\}$. Тоді потужність множини $\boldsymbol{U}$ повинна задовольняти умові: $|\boldsymbol{U}| \le |\boldsymbol{S}| \cdot |\boldsymbol{O}| + 1$.

У загальному вигляді приховуюче перетворення, що використовується відправником для вбудовування приховуваного повідомлення в контейнер, визначається наступним чином.

Означення 4.8. Приховуюче перетворення, що викликає спотворення кодування $\boldsymbol{A}_1$, описується умовною функцією розподілу $\tilde{\boldsymbol{q}}(\boldsymbol{s},\boldsymbol{u}|\boldsymbol{c},\boldsymbol{k})$ відображення з множини $\boldsymbol{C} \times \boldsymbol{K}$ у множину $\boldsymbol{C} \times \boldsymbol{U}$, такою, що виконується умова

$$\sum_{\boldsymbol{c},\boldsymbol{s},\boldsymbol{k},\boldsymbol{u}} \boldsymbol{d}_1(\boldsymbol{c},\boldsymbol{s}) \cdot \tilde{\boldsymbol{q}}(\boldsymbol{s},\boldsymbol{u}|\boldsymbol{c},\boldsymbol{k}) \cdot \boldsymbol{p}(\boldsymbol{c},\boldsymbol{k}) \le \boldsymbol{A}_1. \tag{4.5}$$

Розширення приховуючого перетворення без пам'яті довжиною $\boldsymbol{N}$ описується наступною умовною функцією:

$$\tilde{\boldsymbol{q}}^{\boldsymbol{N}}\left(\boldsymbol{s}^{\boldsymbol{N}},\boldsymbol{u}^{\boldsymbol{N}}\,\middle|\,\boldsymbol{c}^{\boldsymbol{N}},\boldsymbol{k}^{\boldsymbol{N}}\right) = \prod_{\boldsymbol{i}=1}^{\boldsymbol{N}} \tilde{\boldsymbol{q}}\left(\boldsymbol{s}_{\boldsymbol{i}},\boldsymbol{u}_{\boldsymbol{i}}\,|\,\boldsymbol{c}_{\boldsymbol{i}},\boldsymbol{k}_{\boldsymbol{i}}\right). \tag{4.6}$$

Для успішного приховання інформації від кваліфікованого порушника доцільно використовувати не одне, а множину приховуючих перетворень повідомлень.

Означення 4.9. Узагальнене приховуюче перетворення, що спричиняє спотворення кодування не більше величини $\boldsymbol{A}_1$, складається з множини $\tilde{\Psi}$ усіх приховуючих перетворень, що задовольняють умові (4.5).

Узагальнене приховуюче перетворення описує всі можливі варіанти дій відправника при вбудовуванні повідомлень $\boldsymbol{m}$ у контейнер таким чином, щоб величина спотворення кодування не перевищувала припустиму $\boldsymbol{A}_1$. Слід зазначити, що в стеганографії важливо, щоб у приховуючого інформацію існувала множина можливих варіантів, серед яких він рівноімовірно й непередбачувано для порушника обирає конкретний варіант приховання повідомлення, яке потребує захисту.

Для аналізу стеганосистеми зручно записати функцію $\tilde{\boldsymbol{q}}$ у формі добутку функцій розподілу:

$$\tilde{\boldsymbol{q}}(\boldsymbol{s},\boldsymbol{u}|\boldsymbol{c},\boldsymbol{k}) = \boldsymbol{p}(\boldsymbol{s}|\boldsymbol{c},\boldsymbol{u},\boldsymbol{k}) \cdot \boldsymbol{p}(\boldsymbol{u}|\boldsymbol{c},\boldsymbol{k}), \tag{4.7}$$

де $\boldsymbol{p}(\boldsymbol{s}|\boldsymbol{c},\boldsymbol{u},\boldsymbol{k})$ відноситься до "основного" приховуючого перетворення, а $\boldsymbol{p}(\boldsymbol{u}|\boldsymbol{c},\boldsymbol{k})$ – до "допоміжного" приховуючого перетворення.

4.2.3. Атакуючий вплив

Наведемо формальний опис дії порушника стосовно перетворення перехопленої стеганограми $\boldsymbol{s}$ на спотворену стеганограму $\breve{\boldsymbol{s}}$ з метою руйнування прихованої інформації, яка в ній міститься.

Означення 4.10. Атакуючий вплив, що викликає спотворення $\boldsymbol{A}_2$, описується умовною функцією розподілу $\boldsymbol{q}(\breve{\boldsymbol{s}}|\boldsymbol{s})$ відображення з множини $\boldsymbol{S}$ у множину $\breve{\boldsymbol{S}}$, такою, що виконується умова

$$\sum_{\boldsymbol{s},\breve{\boldsymbol{s}}} \boldsymbol{d}_2(\boldsymbol{s},\breve{\boldsymbol{s}}) \cdot \boldsymbol{q}(\breve{\boldsymbol{s}}|\boldsymbol{s}) \cdot \boldsymbol{p}(\boldsymbol{s}) \le \boldsymbol{A}_2. \tag{4.8}$$

Розширення атакуючого впливу без пам'яті довжиною $\boldsymbol{N}$ описується умовною функцією виду

$$q^N\left(\breve{s}^N \middle| s^N\right) = \prod_{i=1}^{N} q\left(\breve{s}_i \middle| s_i\right). \tag{4.9}$$

Означення 4.11. Узагальнений атакуючий вплив, що викликає спотворення не більше величини A_2, складається з множини ψ всіх атакуючих впливів, які задовольняють умові (4.8).

Аналогічно набору варіантів дій передавальної сторони, в атакуючого також є свій набір атакуючих впливів (множина ψ). Порушник, перехопивши стеганограму, намагається обрати такий атакуючий вплив з множини ψ, який би максимізував імовірність руйнування прихованої в ній інформації.

4.3. Прихована пропускна здатність каналу при активній протидії порушника

4.3.1. Основна теорема інформаційного приховання при активній протидії порушника

Дослідимо приховану пропускну здатність у випадку активної протидії порушника, який прагне зруйнувати приховано передавану інформацію. Інформаційно-приховуюче змагання між передавальною і атакуючою сторонами зручно описати методами *теорії ігор* [67,68,97]. В теорії ігор зазвичай мова йде про ігри двох сторін (осіб); вважається, що ігри є обмеженими, тобто обидва гравці в обох випадках можуть робити лише певну кількість кроків і гра завершується після обмеженої кількості кроків. Звідси випливає обмеженість кількості стратегій обох гравців. Під поняттям *стратегії* розуміється така система правил, за допомогою якої задаються дія (дії) гравця у певній ситуації. Метою теорії ігор є знаходження кращої стратегії окремих гравців.

Ціна гри дорівнює величині ППЗ, для максимізації якої відправник інформації оптимально будує приховуюче перетворення. Для мінімізації ППЗ атакуючий синтезує оптимальний атакуючий вплив. Величина ППЗ може бути отримана послідовним з'єднанням приховуючого перетворення і атакуючого впливу. Для можливості оцінки величини прихованої ПЗ для стеганосистеми із двійковим алфавітом, дослідимо теоретико-ігрові аспекти проблеми приховання інформації стеганосистемами.

Розглянемо *основну теорему інформаційного приховання при активній протидії порушника* [61]. Для будь-яких стеганосистем довільної складності і будь-яких атак без пам'яті дана теорема обмежує зверху швидкість безпомилкової передачі для приховуючого інформацію за умови, що атакуючий знає опис приховуючого перетворення, а одержувач знає описи як приховуючого перетворення, так і атакуючого впливу. Дана умова насправді не є важкоздійсненною, як це може здатися на перший погляд. Навіть якщо стратегії дій відправника інформації та атакуючого є невідомими, але стаціонарними, то можна стверджувати, що як атакуючий, так і одержувач потенційно здатні їх визначити, обробивши достатньо великий обсяг статистичного матеріалу. Це припущення є цілком реалістичним, хоча й не завжди може бути досягнуте на практиці з огляду на високу обчислювальну складність [5].

Попередньо розглянемо два твердження, що встановлюють області існування стеганосистем, потенційно здатних безпомилково передавати приховувану інформацію при заданому атакуючому впливі [5].

Надалі позначатимемо через $H(x)$ – ентропію змінної x, через $I(x; y)$ – повну кількість інформації між x та y, а через $I(x; y \mid z)$ – умовну повну кількість інформації між x та y, зумовлених z [68].

Твердження 4.1. Зафіксуємо атакуючий вплив $q(\breve{s}|s)$ і оберемо приховуюче перетворення $\tilde{q}(s,u|c,k)$, яке максимізує кількість інформації виду

$$J(\tilde{q}, q) = I(u; \breve{s} \mid k) - I(u; c \mid k) \tag{4.10}$$

над $\tilde{\psi}$. Для будь-якого як завгодно малого значення $\xi > 0$ і достатньо великого значення N існує стеганосистема з довжиною блоку N, що забезпечує імовірність руйнування приховуваних повідомлень $P_{\text{br.}}^{N} < \xi$ для множини приховуваних повідомлень потужністю

$$|M| < 2^{N\cdot\left[I(u;\breve{s}\,|\,k)-I(u;c\,|\,k)-\xi\right]}.$$

Твердження 4.2. Нехай стеганосистема з довжиною блоку N здатна безпомилково передавати приховувані повідомлення із швидкістю $R = N^{-1}\cdot\log|M|$ біт/елемент контейнера при атакуючому впливі $q(\breve{s}\,|\,s)$. Якщо для будь-якого $\xi > 0$ стеганосистема забезпечує імовірність руйнування приховуваних повідомлень $P_{\text{br.}}^{N} < \xi$ при $N \to \infty$, то існує кінцевий алфавіт U і таке приховуюче перетворення $\tilde{q}(s,u\,|\,c,k)$, що виконується нерівність

$$R \le I(u;\breve{s}\,|\,k) - I(u;c\,|\,k).$$

Теорема 4.1. Нехай атакуючому відомий опис узагальненого приховуючого перетворення $\tilde{\psi}$, а одержувачеві відомий опис узагальненого приховуючого перетворення $\tilde{\psi}$ і узагальненого атакуючого впливу ψ. Для будь-якого інформаційно-приховуючого змагання, що призводить до спотворювань не більше за (A_1, A_2), швидкість передачі R приховуваних повідомлень є досяжною тоді і тільки тоді, коли $R < \underline{B}$.

Величина $\underline{B}$ визначається як

$$\underline{B} = \max_{\tilde{q}(s,u|c,k)\in\tilde{\psi}} \min_{q(\breve{s}|s)\in\psi} J(\tilde{q},q), \tag{4.11}$$

де u – випадкова змінна над довільним кінцевим алфавітом U, змінні $(u, c, k) \to s \to \breve{s}$ утворюють марківський ланцюг, який являє собою окрему форму наступного марківського ланцюга: $u \to (c, s) \to \breve{s}$, характеристики якого розглядаються у [69]. Кількість інформації $J(\tilde{q},q)$ визначається виразом (4.10).

Таким чином, теорема 4.3 визначає величину *нижньої межі* прихованої ПЗ в умовах, коли всі учасники інформаційного змагання знають стратегії дій один одного. Треба зауважити, що в даній теоремі визначається величина ППЗ стеганоканалу, про існування якого атакуючому відомо. Дана ППЗ дорівнює середній кількості біт інформації на один елемент контейнера, яку порушник не може зруйнувати, обираючи будь-яку стратегію протидії з наявної множини ψ при спотворенні контейнера не більше величини A_2.

Доведення цієї теореми зводиться до наступного. Зафіксуємо атакуючий вплив $q \in \psi$. У твердженні 4.1 доводиться, що всі швидкості безпомилкової передачі приховуваних повідомлень, менші за $\max_{\tilde{q}\in\tilde{\psi}} J(\tilde{q},q)$, є досяжними. Твердження 4.2 містить зворотний результат, тобто достовірна передача вище цієї швидкості неможлива. Оскільки атакуючий обізнаний з розподілом $\tilde{q}$, він здатен обрати такий розподіл q, який мінімізуватиме швидкість передачі.

Далі показано, що у важливому спеціальному випадку, коли $k = c$ (тобто секретним ключем стеганосистеми є опис використовуваного контейнера, а сам контейнер відомий одержувачеві), немає втрати в оптимальності при обмеженні кодера стеганосистеми видом, представленим на рис.4.1.

Наслідок. У випадку $k = c$ вибір значення змінної u є оптимальним, тоді і тільки тоді, якщо стеганограма s може бути записана у формі $s = E(c, u)$, де відображення $E(c, \bullet)$ є оборотним для всіх значень c. Зокрема, вибір $u = s$ є оптимальним. Прихована ПЗ у цьому випадку визначається наступним чином:

$$B = \max_{p(s|c)} \min_{q(\breve{s}|s)} I(s;\breve{s}|c) = \min_{q(\breve{s}|s)} \min_{p(s|c)} I(s;\breve{s}|c). \tag{4.12}$$

Це випливає з того, що коли $k = c$, вираз (4.10) може бути записане у вигляді

$$J(\tilde{q},q) = I(u;\breve{s}\,|\,c) = I(u;\breve{s}\,|\,c) - I(u;c\,|\,c) = I(u,c;\ \breve{s}\,|\,c) \le I(s;\breve{s}\,|\,c). \tag{4.13}$$

Отже, цілком логічно, що величина прихованої ПЗ дорівнює взаємній інформації між стеганограмою $\boldsymbol{s}$ і спотвореною стеганограмою $\breve{\boldsymbol{s}}$ за умови, що відправникові й одержувачеві приховуваної інформації є відомим порожній контейнер $\boldsymbol{c}$.

Для практичних систем захисту інформації, якщо секретним ключем стеганосистеми є опис використовуваного контейнера, виникають дві проблеми. По-перше, одержувач повинен знати контейнер-оригінал, що обмежує можливу область застосування таких стеганосистем. По-друге, відправник і одержувач приховуваних повідомлень повинні використовувати секретну ключову інформацію дуже великого обсягу, що є незручним на практиці.

4.3.2. Властивості прихованої пропускної здатності стеганоканалу

Розглянемо властивості ППЗ, наведені у [5]. Прихована пропускна здатність є функцією аргументів $\boldsymbol{A}_1$ і $\boldsymbol{A}_2$, що зручно виразити у вигляді $\boldsymbol{B}(\boldsymbol{A}_1, \boldsymbol{A}_2)$, і характеризується наступними властивостями:

1) Величина $\boldsymbol{B}(\boldsymbol{A}_1, \boldsymbol{A}_2)$ монотонно зростає при збільшенні рівня спотворення, викликаного кодуванням ($\boldsymbol{A}_1$) і монотонно зменшується при зростанні спотворення, спричиненого атакуючим впливом ($\boldsymbol{A}_2$).

2) Функція $\boldsymbol{B}(\boldsymbol{A}_1, \boldsymbol{A}_2)$ опукла за аргументом $\boldsymbol{A}_2$.

3) Величина $\boldsymbol{B}(\boldsymbol{A}_1, \boldsymbol{A}_2)$ обмежена зверху ентропією спотвореної стеганограми $\breve{\boldsymbol{s}}$ і ентропією контейнера $\boldsymbol{c}$:

$$\boldsymbol{B}(\boldsymbol{A}_1, \boldsymbol{A}_2) \le \max_{\tilde{\boldsymbol{q}} \in \tilde{\boldsymbol{\Psi}}} \min_{\boldsymbol{q} \in \boldsymbol{\Psi}} \boldsymbol{H}(\breve{\boldsymbol{s}}) \le \boldsymbol{H}(\boldsymbol{s}) \le \boldsymbol{H}(\boldsymbol{c}) \le \log |\boldsymbol{C}| .$$

Дана властивість є очевидною, оскільки ППЗ не може бути більше ентропії спотвореної стеганограми $\breve{\boldsymbol{s}}$. У свою чергу, в силу можливої втрати інформації через атакуючий вплив, величина $\boldsymbol{H}(\breve{\boldsymbol{s}})$ не може бути більше ентропії стеганограми $\boldsymbol{s}$, а $\boldsymbol{H}(\boldsymbol{s})$ через можливу втрату інформації при вбудовуванні приховуваних повідомлень не може перевищувати ентропію $\boldsymbol{H}(\boldsymbol{c})$ порожнього контейнера $\boldsymbol{c}$. З теорії інформації відомо, що ентропія джерела не може перевищувати логарифм від потужності його алфавіту [70]. Оскільки найчастіше використовуються контейнери у вигляді істотно надлишкових зображень або аудіосигналів, то для таких контейнерів виконується нерівність $\boldsymbol{H}(\boldsymbol{c}) << \log |\boldsymbol{C}|$, що істотно зменшує можливе значення ППЗ. Отже, в стеганосистемі чим ближчими є характеристики дискретних контейнерів до бернуллівського розподілу (або неперервних контейнерів до гаусівського), тим більша величина ППЗ може бути досягнута.

4) Величина $\boldsymbol{B}(0, \boldsymbol{A}_2) = 0$ для будь-яких значень атакуючого спотворення $\boldsymbol{A}_2$, оскільки $\boldsymbol{A}_1 = 0$ означає, що $\boldsymbol{s} = \boldsymbol{c}$, тобто контейнер-оригінал повністю збігається зі стеганограмою (жодної приховуваної інформації не передається).

5) Якщо є припустимим достатньо велике атакуюче спотворення $\boldsymbol{A}_2$, то для будь-якого значення спотворення $\boldsymbol{A}_1$ може бути побудована атака порушника, в якій стеганопослідовність $\breve{\boldsymbol{s}}^N$ формується незалежно від $\boldsymbol{s}^N$. Отже, у послідовності $\breve{\boldsymbol{s}}^N$ усунуті всі сліди прихованого повідомлення і ППЗ дорівнює нулю для будь-яких значень спотворення кодування $\boldsymbol{A}_1$. Таким чином, якщо атакуючий має можливість подавляти канал передачі приховуваних повідомлень необмежено потужною завадою, то він гарантовано зруйнує повідомлення, що були передані. Але у багатьох практичних випадках інформаційного приховання в порушника відсутній такий енергетичний потенціал, або ж в разі його наявності ним неможливо скористатися в повній мірі.

4.3.3. Коментарі отриманих результатів

Наведемо висновки з теореми 4.1 і коментарі властивості прихованої пропускної здатності [5].

1) Теорема 4.1 визначає, що встановлення теоретичної можливості прихованої безпомилкової передачі інформації й теоретичної можливості протидії цьому зводиться до обчислення величини ППЗ $\boldsymbol{B}$ за відомих стратегій сторін і порівняння її з необхідною швидкістю передачі приховуваної інформації $\boldsymbol{R}$. Якщо ППЗ виявиться менше необхідної швидкості, то навіть

теоретично не існує способу передачі приховуваних повідомлень без спотворень і задача атакуючого щодо руйнування довільних стеганосистем гарантовано вирішуватиметься.

Оптимальна атака порушника полягає у внесенні такого спотворення $\boldsymbol{A}_2$, за якого величина ППЗ є меншою за необхідну швидкість передачі приховуваних повідомлень. Оптимальна стратегія приховуючого інформацію зводиться до обрання такого кодування і такої величини викликаного ним спотворення $\boldsymbol{A}_1$, при яких з урахуванням спотворення $\boldsymbol{A}_2$ необхідна швидкість безпомилкової передачі не перевищуватиме ППЗ. Це означає, що теоретично існує такий спосіб безпомилкової передачі. Однак теоретична можливість ще не означає, що передавальна сторона буде здатною реалізувати її на практиці. Наприклад, розроблювач стеганосистеми може не знати оптимальних принципів її побудови (вони ще не відкриті), або через обмеженість в обчислювальних ресурсах він не може собі дозволити оптимальну обробку, або вимоги до своєчасності доставки приховуваних повідомлень обмежують довжину $\boldsymbol{N}$ блоку кодування тощо.

Таким чином, успіх будь-якої із змагаючихся сторін в остаточному підсумку визначатиметься співвідношенням між швидкістю передачі $\boldsymbol{R}$ і величинами спотворення $\boldsymbol{A}_1$ і $\boldsymbol{A}_2$ контейнера, у якому приховується інформація.

Розглянута теорема інформаційного приховання при активній протидії порушника нагадує фундаментальну теорему *C. Shannon'a*, у якій визначається, що існує спосіб безпомилкової передачі повідомлень по каналу із завадами, якщо швидкість передачі менше пропускної здатності каналу, і неможлива достовірна передача із швидкістю, більшою за пропускну здатність каналу. *C. Shannon* також показав, що існують залежності між відношенням потужності корисного сигналу до потужності завад у каналі зв'язку та величиною швидкості безпомилкової передачі повідомлень цим каналом. Аналогічно цьому, в інформаційно-приховуючому змаганні існують подібні залежності між відношенням величини спотворення кодування $\boldsymbol{A}_1$ до величини спотворення атакуючого впливу $\boldsymbol{A}_2$ та величиною швидкості безпомилкової передачі приховуваних повідомлень стеганоканалом. Але при зовнішній подібності, у задач відкритої та прихованої передачі є істотні відмінності. Відкритий зв'язок здійснюється в умовах впливів випадкових перешкод каналу зв'язку, а передача приховуваної інформації повинна бути забезпечена навіть за умови оптимізованої навмисної протидії активного порушника.

2) Розглянемо зв'язок задачі інформаційного приховання із задачею захисту інформації від перехоплення в каналі, який прослуховується. У 1975 р. американський вчений *A.D. Wyner* запропонував метод захисту інформації від читання порушником, що заклав основу *теорії кодового зашумлення* [5,71-73]. Відправник дискретних повідомлень здійснює їх випадкове надлишкове кодування і передає перетворені повідомлення одержувачеві основним каналом зв'язку. Порушник спостерігає їх у підслуховуючому каналі, який є відведенням від основного каналу. Випадкове кодування побудовано таким чином, що якщо в підслуховуючому каналі є помилки, то при декодуванні вони розмножуються й надійно спотворюють захищувану інформацію. Метод кодового зашумлення призначений для систем передачі, у яких основний канал є безпомилковим. Наприклад, основний канал утворений на основі волоконно-оптичної лінії, а порушник намагається вести розвідку по каналах побічного електромагнітного випромінювання і наведень, в яких у силу їхньої природи існує велика кількість завад. Відзначимо, що порушник знає опис системи кодового зашумлення, яка не використовує секретної ключової інформації (спосіб захисту некриптографічний). Підслуховуючий канал характеризується секретною ПЗ, яка являє собою максимальну швидкість безпомилкової передачі основним каналом за умови, що невизначеність для перехоплювача є максимальною (невизначеність захищуваних повідомлень дорівнює ентропії цих повідомлень). Однак, якщо підслуховуючий канал є менш зашумленим, порівняно з основним каналом, то секретна ПЗ дорівнює нулю.

У завданні інформаційного приховання атакуючий здатний на більше, ніж звичайний перехоплювач у підслуховуючому каналі, оскільки він після перехоплення захищуваного повідомлення навмисно спотворює основний канал. Тому основний канал передачі не менш зашумлений, ніж підслуховуючий канал. Отже, у завданні інформаційного приховання з активним порушником секретна ПЗ дорівнює нулю.

3) Обрання змінної $\boldsymbol{u}$ незалежно від контейнера $\boldsymbol{c}$, як це робиться в системі ЦВЗ [5], є в загальному випадку не оптимальним. Аналіз виразу (4.10) показує, що швидкості безпомилкової передачі в цьому випадку обмежені зверху величиною $\boldsymbol{I(u;\ \breve{s} \mid k)}$.

4) Нехай виконується умова $\boldsymbol{A_2 \geq A_1}$. Якщо атакуючому відомий опис контейнера $\boldsymbol{c^N}$, то оптимальна атака полягає лише у формуванні спотвореної стеганограми у вигляді $\breve{\boldsymbol{s}}^{\boldsymbol{N}} = \boldsymbol{c^N}$. У цьому випадку вихідний сигнал після атаки не містить жодних слідів повідомлення і прихована ПЗ дорівнює нулю. На практиці це може означати наступне. Якщо порушникові відомий оригінал захищуваної від піратського копіювання інформації, то жодні стеганосистеми не захистять авторські чи майнові права виробників цієї інформації.

Розглянемо потенційно сильну атаку, у якій атакуючий прагне сконструювати досить близьку до оригіналу оцінку контейнера $\boldsymbol{c^N}$. Якщо атакуючий здатний синтезувати спотворену стеганограму $\breve{\boldsymbol{s}}$ таку, що $\boldsymbol{H(\breve{s} \mid c) < \xi}$, то ППЗ обмежуватиметься зверху величиною

$$\boldsymbol{I(u;\ \breve{s} \mid k) - I(u; c \mid k) = I(u; c, \breve{s} \mid k) - I(u; c \mid \breve{s}, k) - I(u; c, \breve{s} \mid k) -} \tag{4.14}$$
$$\boldsymbol{I(u;\ \breve{s} \mid c, k) \leq I(u; \breve{s} \mid c, k) \leq H(\breve{s} \mid c, k) \leq H(\breve{s} \mid c) < \xi,}$$

для будь-якого $\boldsymbol{u}$. Отже, величина прихованої ПЗ стеганоканалу $\boldsymbol{B(A_1, A_2) < \xi}$.

Таким чином, якщо порушник здатен сформувати досить точну оцінку контейнера (іншими словами, виконується нерівність $\boldsymbol{H(\breve{s} \mid c) < \xi}$, де величина $\boldsymbol{\xi}$ досить мала), то величина ППЗ обмежена цією малою величиною. На практиці це означає, що маючи заповнений контейнер (стеганограму), порушник може спробувати відтворити з нього з деякою припустимою похибкою контейнер-оригінал, з якого вилучене приховуване повідомлення (це є особливо актуальним в галузі захисту за допомогою ЦВЗ мультимедійної інформації).

4.4. Двійкова стеганосистема передачі приховуваних повідомлень

Визначимо величину прихованої ПЗ стеганосистеми, у якої алфавіт приховуваних повідомлень, контейнерів, ключів і стеганограм є двійковим: $\boldsymbol{m = c = k = s} = \{0; 1\}$ [5,61]. Нехай контейнер $\boldsymbol{c}$ формується джерелом Бернуллі з параметром $\boldsymbol{p}$ = 0.5 (тобто двійкові символи послідовності контейнера є рівноімовірними і незалежними один від одного). Функція спотворення $\boldsymbol{d_1 = d_2}$ описується відстанню Хемінга: $\boldsymbol{d(x, y)}$= 0, якщо $\boldsymbol{x = y}$ і $\boldsymbol{d(x, y)}$ = 1, якщо $\boldsymbol{x \neq y}$. Опис контейнера є секретним ключем стеганосистеми ($\boldsymbol{k = c}$) і є відомим одержувачеві. Нехай стеганограми формуються у вигляді $\boldsymbol{s = c \oplus z}$, де операція «⊕» є підсумовуванням за модулем 2. Очевидно, що змінна $\boldsymbol{z}$ матиме бернуллівський розподіл і відображатиме приховуване повідомлення $\boldsymbol{m}$ із спотворенням $\boldsymbol{A_1}$. Спотворення $\boldsymbol{A_1}$ означає, що кожен символ двійкової послідовності $\boldsymbol{z}$ відрізняється від відповідного символу двійкової послідовності $\boldsymbol{m}$ з імовірністю $\boldsymbol{A_1}$. Перетворення повідомлення $\boldsymbol{m}$ на послідовність $\boldsymbol{z}$ виконується передавальною стороною з використанням кодера із спотворенням $\boldsymbol{A_1}$. Порушник обробляє стеганограму накладенням на неї двійкової шумової послідовності $\boldsymbol{a}$, в якій одиничний символ породжується з імовірністю $\boldsymbol{A_2}$. Одержувач підсумовує спотворену стеганограму $\breve{\boldsymbol{s}}$ з двійковою послідовністю $\boldsymbol{c}$ за модулем 2, і з отриманої в такий спосіб двійкової послідовності $\breve{\boldsymbol{z}}$ декодує прийняте приховуване повідомлення $\breve{\boldsymbol{m}}$.

Особливістю такої стеганосистеми є те, що приховуване повідомлення в ній при вбудовуванні спотворюється з імовірністю спотворення $\boldsymbol{A_1}$ і це спотворення дорівнює спотворенню кодування стеганограми. Описана стеганосистема зображена на рис.4.2.

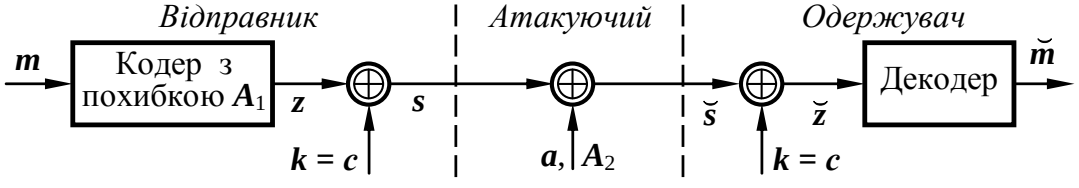


Рис.4.2. Структурна схема двійкової стеганосистеми

Твердження 4.3. Для двійкової стеганосистеми при рівнях спотворень A_1, $A_2 \le 0.5$, прихована пропускна здатність визначається як

$$B = \underline{H}(A_1 * A_2) - \underline{H}(A_2), \tag{4.15}$$

де $\underline{H}(t) = -t \cdot \log(t) - (1-t)\cdot\log(1-t)$; $A_1 * A_2 = A_1\cdot(1-A_2) + A_2\cdot(1-A_1)$.

Для даної стеганосистеми змінну u можна формувати як $u = s$ або $u = z$, причому обидва варіанти можуть бути оптимальними, оскільки в якості операції вбудовування використовується операція підсумовування за модулем 2 [5]. Оптимальна атака порушника визначається у вигляді $\breve{s} = s \oplus a$, де a є випадковою двійковою послідовністю, розподіленою за бернуллівським законом з імовірністю появи одиничного символу – A_2.

При рівнях спотворень $A_1 \ge 0.5$ і $A_2 < 0.5$ ППЗ дорівнює $B = 1 - \underline{H}(A_2)$. Якщо $A_2 \ge 0.5$, ППЗ дорівнює нулю.

Необхідно зауважити, що при $A_1 = 0.5$ ППЗ не дорівнює нулю незалежно від значення $A_2 < 0.5$. Це пояснюється тим, що при перетворенні приховуваного повідомлення m у послідовність z спотворення не є рівноімовірним: особа, що приховує інформацію, може обрати такий розподіл помилок A_1, при якому мінімізуватиметься зміна повідомлення m. Для $A_2 = 0.5$ ППЗ дорівнюватиме нулю при будь-яких значеннях A_1. Неважко помітити, що при у цьому випадку вихід $\breve{s}$ каналу зв'язку не залежить від його входу s, що означає розрив каналу зв'язку. А якщо при обриві каналу зв'язку не можлива передача інформації по відкритому каналу зв'язку, то тим більше не можлива й передача прихованим каналом, який утворено на основі відкритого.

Застосуємо наслідок теореми 4.1 для аналізу двійкової стеганосистеми. Нехай $z = c \oplus s$, $a = s \oplus \breve{s}$. Платіжна функція має вигляд $I(s; \breve{s} \mid c)$. Приймемо, що $A_1, A_2 \le 0.5$.

Крок 1. Зафіксуємо $q(\breve{s} \mid s)$. Для всіх $q \in \psi$, одержимо

$$I(s; \breve{s} \mid c) \overset{(a)}{=} H(\breve{s} \mid c) - H(\breve{s} \mid s, c) \overset{(b)}{=}$$
$$\overset{(b)}{=} H(\breve{s} \mid c) - H(\breve{s} \mid s) = H(\breve{s} \oplus c \mid c) - H(a) = H(z \oplus a \mid c) - \underline{H}(A_2) \overset{(c)}{\le}$$
$$\overset{(c)}{\le} H(z \oplus a) - \underline{H}(A_2) \overset{(d)}{\le}$$
$$\overset{(d)}{\le} \underline{H}(A_1 * A_2) - \underline{H}(A_2),$$

де рівність (a) справедлива відповідно до визначення умовної взаємної інформації; (b) виконується завдяки тому, що $c \to s \to \breve{s}$ є марківським ланцюгом; нерівність (c) справедлива, оскільки умова зменшує ентропію. Рівність у (c) досягається тоді і тільки тоді, коли $z \oplus a$, отже, z є незалежною від c. Нерівність (d) справедлива, оскільки z і a є незалежними (в силу того, що $z \to s \to a$ формує марківський ланцюг і $P[z = 1] \le A_1$. Рівність у (d) досягається, якщо змінна z має бернуллівський розподіл з дисперсією A_1. Розподіл $p(s \mid c)$ задовольняє обом нерівностям з рівністю і тому максимізує значення $I(s; \breve{s} \mid c)$.

Крок 2. Зафіксуємо $p(s \mid c)$. Мінімізуватимемо $I(s; \breve{s} \mid c)$ над $q(\breve{s} \mid s)$. При визначеному раніше розподілі $p(s \mid c)$, z і s є незалежними. Оскільки $z \to s \to a$ формує марківський ланцюг, z і a є також незалежними. Маємо

$$I(s; \breve{s} \mid c) = I(s \oplus c;\ \breve{s} \oplus c \mid c) = I(z; z \oplus a \mid c) = H(z) - H(z \mid z \oplus a, c) \overset{(e)}{\ge}$$
$$\overset{(e)}{\ge} H(z) - H(z \mid z \oplus a) = I(z; z \oplus a) \overset{(f)}{\ge} \underline{H}(A_1 * A_2) - \underline{H}(A_2),$$

де нерівність (e) справедлива, оскільки умова зменшує ентропію; нерівність (f) справедлива тому, що z і a є незалежними і $P[a = 1] \le A_2$ (що стає рівністю, коли a – змінна з бернуллівським розподілом з імовірністю одиничного символу A_2).

Розглянута двійкова стеганосистема схожа на систему шифрування однократної підстановки (шифр гаммування з нескінченною рівноімовірною незалежною шифруючою гаммою). При незалежній і рівноімовірній послідовності c виконується рівність $H(z) = H(z \mid s)$, що означає, що дана система задовольняє вимозі щодо ідеальних криптосистем [60], отже, перехоплення й аналіз криптограми s не дає атакуючому жодної інформації стосовно захищуваного повідомлення z. Однак ця двійкова система задовольняє також вимозі щодо ідеальних стеганосистем: розподіли

$\boldsymbol{p}(\boldsymbol{c})$ і $\boldsymbol{p}(\boldsymbol{s})$ ідентичні, тому для порушника неможливо визначити, чи належать перехоплені ним дані до розподілу $\boldsymbol{p}(\boldsymbol{c}^N)$ порожніх контейнерів або ж до розподілу $\boldsymbol{p}(\boldsymbol{s}^N)$ стеганограми із вбудованим повідомленням [15]. Але при цьому зауважується, що в розглянутій стеганосистемі передбачається опис контейнерів і, відповідно, стеганограм бернуллівським розподілом, а це зазвичай не є характерним для реальних систем приховання інформації [5].

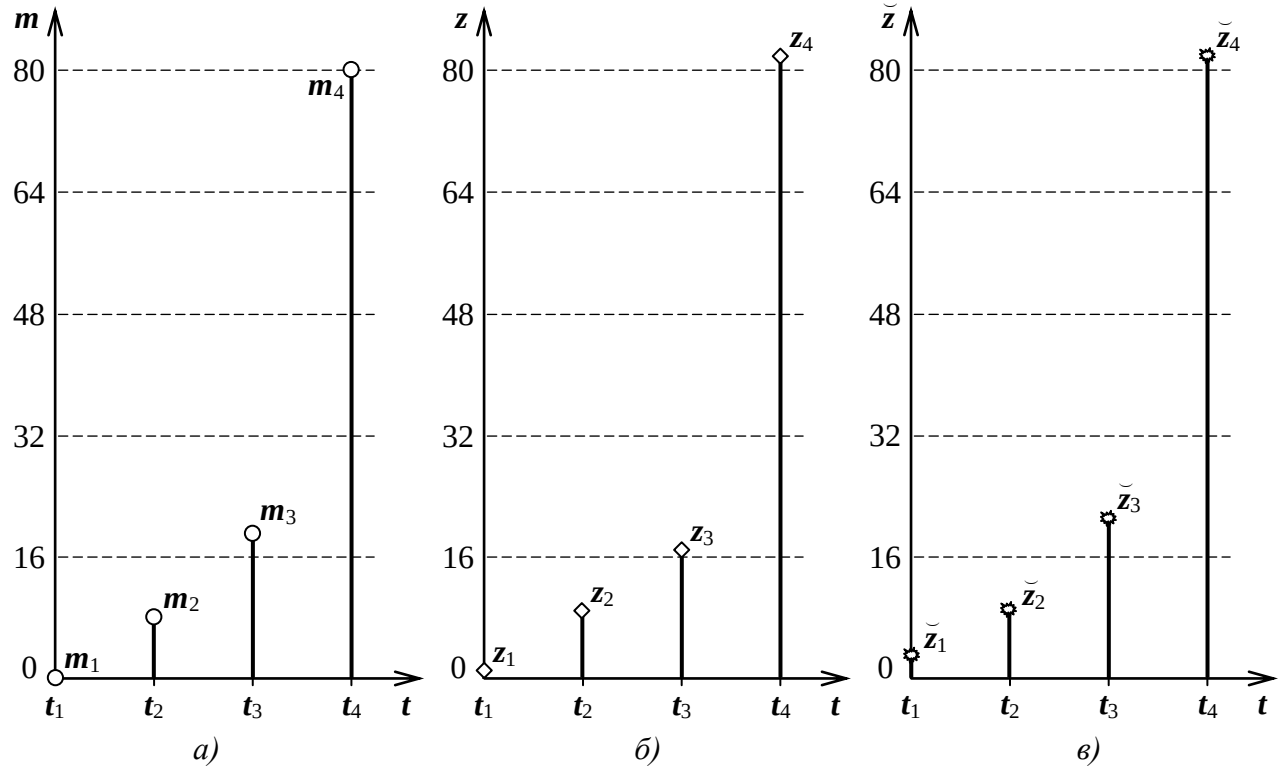
Розглянемо приклад двійкової стеганосистеми з вибором $\boldsymbol{u} = \boldsymbol{z}$ [5]. Нехай існує необхідність в прихованій передачі повідомлення $\boldsymbol{m}$, яке являє собою оцифрований мовний сигнал з кількістю рівнів квантування – 8. У загальному вигляді приховуване повідомлення може бути представлене у вигляді $\boldsymbol{m} = \{\boldsymbol{m}_1, \boldsymbol{m}_2, \boldsymbol{m}_3, \boldsymbol{m}_4, ...\}$. Нехай перші декілька відліків повідомлення в моменти часу дискретизації $\boldsymbol{t}_1, \boldsymbol{t}_2, \boldsymbol{t}_3, \boldsymbol{t}_4, ...$ приймають десяткові значення $\boldsymbol{m}_1 = 0$, $\boldsymbol{m}_2 = 8$, $\boldsymbol{m}_3 = 19$, $\boldsymbol{m}_4 = 80$ (рис. 4.3, *а*). У двійковій формі приховуване повідомлення запишемо як

$$\boldsymbol{m}_1 = 0000\ 0000_2, \quad \boldsymbol{m}_2 = 0000\ 1000_2, \quad \boldsymbol{m}_3 = 0001\ 0011_2, \quad \boldsymbol{m}_4 = 0101\ 0000_2, \ldots .$$

Перетворимо двійкову послідовність $\boldsymbol{m}$ у двійкову послідовність $\boldsymbol{z}$ з похибкою $\boldsymbol{A}_1$. У двійковій стеганосистемі похибка кодування $\boldsymbol{A}_1$ обчислюється за метрикою Хемінга. Нехай $\boldsymbol{A}_1 = 1/8$. Отже, для формування послідовності $\boldsymbol{z} = \{\boldsymbol{z}_1, \boldsymbol{z}_2, \boldsymbol{z}_3, \boldsymbol{z}_4, ...\}$ особа, що приховує інформацію, спотворює восьму частину бітів послідовності $\boldsymbol{m}$. Для зменшення спотворення приховуваного повідомлення йому доцільно спотворювати тільки молодші біти двійкової послідовності $\boldsymbol{m}$. Нехай приховуючий інформацію обрав послідовність $\boldsymbol{z}$ наступного вигляду:

$$\boldsymbol{z}_1 = 0000\ 0001_2, \quad \boldsymbol{z}_2 = 0000\ 1001_2, \quad \boldsymbol{z}_3 = 0001\ 0001_2, \quad \boldsymbol{z}_4 = 0101\ 0010_2, \ldots .$$

У десятковому вигляді послідовність $\boldsymbol{z}$ зображена на рис.4.3, *б*.



а) *б)* *в)*

Рис.4.3. Приклад двійкової стеганосистеми із спотвореннями $\boldsymbol{A}_1 = 1/8$ і $\boldsymbol{A}_2 = 1/16$

Припустимо, що за допомогою генератора ПВЧ було сформовано секретний ключ $\boldsymbol{k} = \{\boldsymbol{k}_1, \boldsymbol{k}_2, \boldsymbol{k}_3, \boldsymbol{k}_4, ...\}$:

$$\boldsymbol{k}_1 = 0100\ 1101_2, \quad \boldsymbol{k}_2 = 0111\ 0010_2, \quad \boldsymbol{k}_3 = 0101\ 0101_2, \quad \boldsymbol{k}_4 = 0101\ 1001_2, \ldots .$$

Відправник за правилом $\boldsymbol{s} = \boldsymbol{k} \oplus \boldsymbol{z}$ формує стеганограму $\boldsymbol{s} = \{\boldsymbol{s}_1, \boldsymbol{s}_2, \boldsymbol{s}_3, \boldsymbol{s}_4, ...\}$:

$$\boldsymbol{s}_1 = 0100\ 1100_2, \quad \boldsymbol{s}_2 = 0111\ 1011_2, \quad \boldsymbol{s}_3 = 0100\ 0100_2, \quad \boldsymbol{s}_4 = 0000\ 1011_2, \ldots .$$

Нехай спотворення $\mathbf{A}_2 = 1/16$. Порушник випадковим чином формує двійкову послідовність $\boldsymbol{a} = \{\boldsymbol{a}_1, \boldsymbol{a}_2, \boldsymbol{a}_3, \boldsymbol{a}_4, ...\}$, в якій імовірність появи одиничних символів складає $\boldsymbol{A}_2$:

$$\boldsymbol{a}_1 = 0000\ 0010_2, \quad \boldsymbol{a}_2 = 0000\ 0000_2, \quad \boldsymbol{a}_3 = 0000\ 0100_2, \quad \boldsymbol{a}_4 = 0000\ 0000_2, \ldots .$$

Атакуючий вплив являє собою додавання за модулем 2 стеганограми $\boldsymbol{s}$ і шумової послідовності $\boldsymbol{a}$. Отже, спотворена стеганограма $\breve{\boldsymbol{s}} = \{\breve{\boldsymbol{s}}_1, \breve{\boldsymbol{s}}_2, \breve{\boldsymbol{s}}_3, \breve{\boldsymbol{s}}_4, ...\}$ матиме наступний вигляд:

$$\breve{\boldsymbol{s}}_1 = 0100\ 1110_2, \quad \breve{\boldsymbol{s}}_2 = 0111\ 1011_2, \quad \breve{\boldsymbol{s}}_3 = 0100\ 0000_2, \quad \breve{\boldsymbol{s}}_4 = 0000\ 1011_2, \ldots .$$

Одержувач для формування прийнятого повідомлення $\breve{\boldsymbol{z}}$ додає за модулем 2 послідовність $\breve{\boldsymbol{s}}$ до послідовності ключа $\boldsymbol{k}$:

$$\breve{\boldsymbol{z}}_1 = 0000\ 0011_2, \quad \breve{\boldsymbol{z}}_2 = 0000\ 1001_2, \quad \breve{\boldsymbol{z}}_3 = 0001\ 0101_2, \quad \breve{\boldsymbol{z}}_4 = 0101\ 0010_2, \ldots .$$

У декодері одержувач з послідовності відновлює повідомлення $\boldsymbol{m}$. У найпростішому випадку $\breve{\boldsymbol{m}} = \breve{\boldsymbol{z}}$. Вигляд послідовності $\breve{\boldsymbol{m}}$ зображено на рис.4.3, *в*.

Якщо приховуване повідомлення $\boldsymbol{m}$ являє собою мовний сигнал, то при зазначених величинах спотворень $\boldsymbol{A}_1$ і $\boldsymbol{A}_2$ ступінь наближеності $\breve{\boldsymbol{m}}$ до $\boldsymbol{m}$, тобто якість забезпечуваного прихованого телефонного зв'язку, для низки телекомунікаційних завдань може бути оцінений як задовільний.

4.5. Висновки

У даному розділі введене до розгляду одне з ключових понять теорії передачі інформації і, зокрема, стеганографічних систем (як каналів прихованого обміну даними) – пропускна здатність каналу передачі. В ході опрацювання закордонних і вітчизняних літературних джерел виділено два основних підходи до оцінки ПЗ стеганосистем:

– підхід, орієнтований на стеганосистеми, в яких приховані повідомлення повинні бути безпомилково передані адресатові в умовах активної протидії порушника. При цьому враховується, що крім спотворювань контейнера при вбудовуванні в нього конфіденційних даних, імовірні його навмисні спотворення з боку активного порушника і/або спотворення, викликані випадковими завадами в каналі зв'язку;

– підхід, орієнтований на стеганосистеми, в яких реалізується прихована передача апріорно невідомої одержувачеві інформації, причому пасивний порушник намагається в процесі спостереження виявити факт наявності каналу прихованого зв'язку і, в разі успіху, прагне розкрити зміст прихованих даних.

В рамках першого підходу наведено основні завдання інформаційного приховання в разі активної протидії порушника; описано приховуюче перетворення, виконуване при вбудовуванні інформації в контейнер, і атакуючий вплив, здійснюваний порушником для протидії прихованій передачі; наведена основна теорема інформаційного приховання при активній протидії порушника; наведене визначення величини прихованої ПЗ двійкової стеганосистеми.

На основі цього розглянуто основні властивості прихованої пропускної здатності стеганоканалу, здійснено коментарі отриманих результатів, що дозволило закласти обґрунтовану теоретичну базу для розробки систем стеганографічного приховання конфіденційної інформації.

5. СТЕГАНОГРАФІЧНІ МЕТОДИ ПРИХОВУВАННЯ ДАНИХ І ЇХ РЕАЛІЗАЦІЯ У СИСТЕМІ MathCAD

5.1. Вступні положення

У даному розділі роботи розглядаються стеганографічні методи приховування даних для різних типів інформаційного середовища в якості стеганоконтейнерів. При цьому значна увага приділена проблемі практичної реалізації методів, що розглядаються, з використанням сучасних засобів обчислювальної техніки і програмного забезпечення. У відповідності до [3], під час розгляду методів позначатимемо літерою $\boldsymbol{C}$ контейнер, що являє собою послідовність елементів $\boldsymbol{c_i}$ довжиною $\boldsymbol{l_C}$. У випадку використання в якості контейнера файлу цифрового звуку це буде кількість відліків в одиницю часу, стосовно файлу цифрового зображення – послідовність, отримана шляхом векторизації зображення (тобто шляхом розгортання масиву всіх пікселів зображення у вектор)[1)].

Для двійкових масивів контейнерів значення $\boldsymbol{c_i}$ можуть приймати значення «0» або «1»; для квантового зображення або звуку (якщо кількість біт, якими кодується один відлік, дорівнює 8) – змінюватись в діапазоні від «0» до «255» (2^8 = 256 градацій); для звуку, відліки якого кодуються 16-ма бітами, – від «–32768» до «32767» (2^{16} = 65536 градацій); для текстів $\boldsymbol{c_i}$ – це символ кодової таблиці, код якого може приймати значення від «0» до «255».

Аналогічно позначатимемо літерою $\boldsymbol{S}$ заповнений контейнер (*стеганограму*) – послідовність елементів $\boldsymbol{s_j}$ довжиною $\boldsymbol{l_S}$, а літерою $\boldsymbol{M}$ – повідомлення довжиною $\boldsymbol{l_M}$, яке підлягає приховуванню. За відсутності спеціального обумовлення, вважатимемо, що $\boldsymbol{m_n} \in \{0; 1\}$.

5.2. Класифікація методів приховування даних

Переважна більшість методів комп'ютерної стеганографії (КС) базується на двох ключових принципах [3]:

1) файли, які не вимагають абсолютної точності (наприклад, файли з зображенням, звуковою інформацією тощо), можуть бути видозмінені (звичайно, до певного ступеня) без втрати своєї функціональності;

2) органи чуттів людини не здатні надійно розрізняти незначні зміни у модифікованих таким чином файлах та/або відсутній спеціальний інструментарій, який був би спроможним виконати дану задачу.

В основі базових підходів до реалізації методів КС в рамках того або іншого інформаційного середовища лежить виділення малозначимих фрагментів цього середовища і заміна існуючої в них інформації на інформацію, яку необхідно приховати. Оскільки в КС розглядаються середовища, підтримувані сучасними засобами обчислювальної техніки і комп'ютерних мереж, то все інформаційне середовище, зрештою, може бути представлене у цифровому вигляді [3]. Таким чином, незначущі для кадру інформаційного середовища фрагменти відповідно до того або іншого алгоритму чи методики замінюються (заміщуються) на фрагменти приховуваної інформації. Під кадром інформаційного середовища в даному випадку розуміється певна його частина, виділена за характерними ознаками. Такими ознаками зазвичай є семантичні характеристики виділюваної частини інформаційного середовища. Наприклад, кадром може бути обране деяке окреме зображення, звуковий файл, Web-сторінка та ін.

Для існуючих методів комп'ютерної стеганографії вводять наступну класифікацію (рис.5.1) [3,5].

[1)] В подальшому буде показана можливість проведення векторизації й цифрового звуку, якщо той має два і більше каналів.

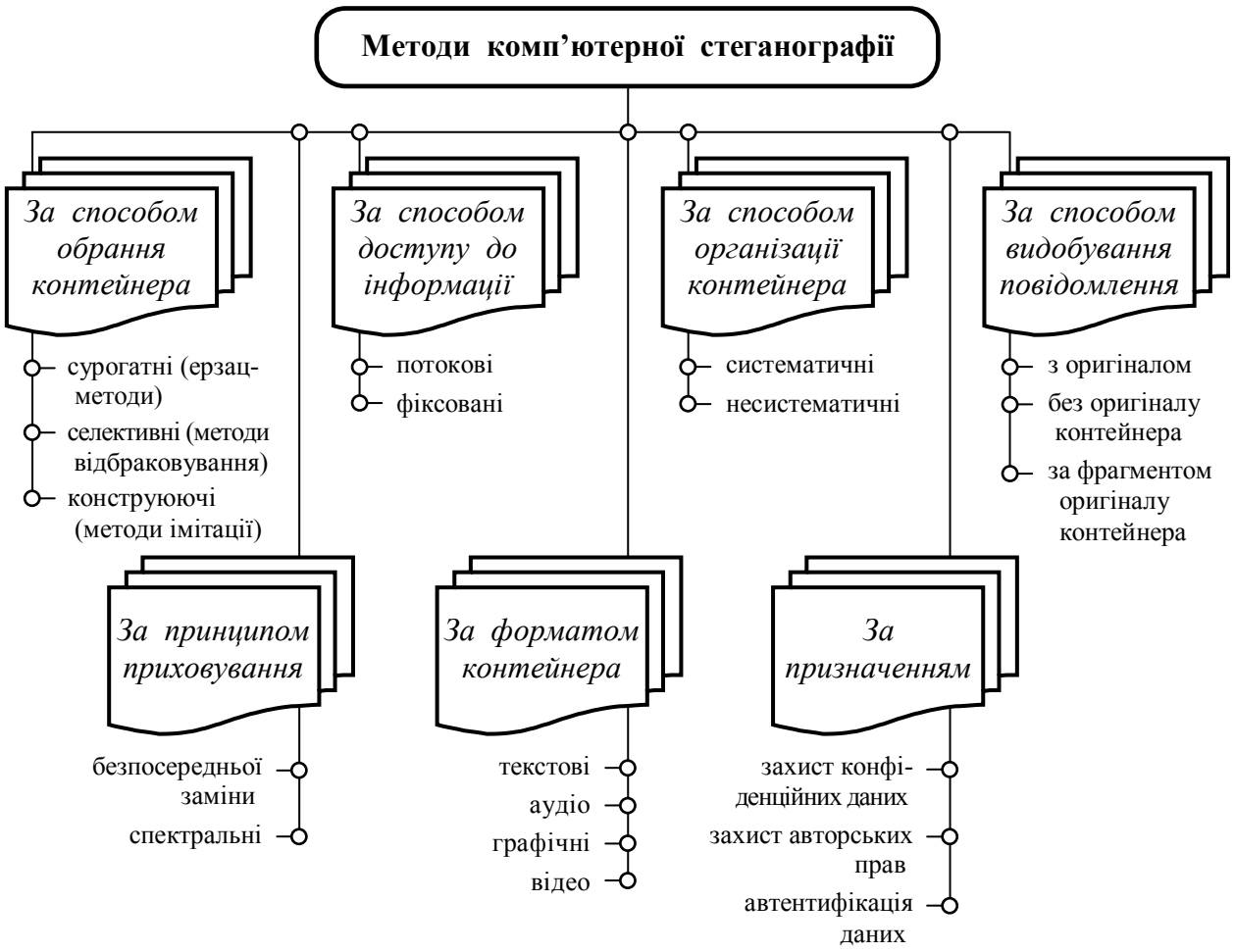


Рис.5.1. Класифікація методів комп'ютерної стеганографії

Як уже зазначалося у розділі 2, **за способом обрання контейнера** розрізнюють сурогатні (або так звані ерзац-методи), селективні та конструюючі методи стеганографії [3].

В *сурогатних* (*безальтернативних*) методах стеганографії повністю відсутня можливість вибору контейнера і для приховування повідомлення обирається перший контейнер, що трапився, – ерзац-контейнер, – який у більшості випадків не є оптимальним для приховування повідомлення заданого формату.

У *селективних* методах КС передбачається, що приховане повідомлення повинно відтворювати спеціальні статистичні характеристики шуму контейнера. Для цього генерують велику кількість альтернативних контейнерів, з наступним обранням (шляхом відбраковування) найоптимальнішого з них для конкретного повідомлення. Окремим випадком такого підходу є обчислення деякої хеш-функції для кожного контейнера. При цьому для приховання повідомлення обирається той контейнер, хеш-функція якого збігається зі значенням хеш-функції повідомлення (тобто стеганограмою є обраний контейнер).

У *конструюючих* методах стеганографії контейнер генерується самою стеганосистемою. При цьому існують декілька варіантів реалізації. Так, наприклад, шум контейнера може імітуватися приховуваним повідомленням. Це реалізується за допомогою процедур, які не лише кодують приховане повідомлення під шум, але й зберігають модель початкового шуму. У граничному випадку за моделлю шуму може будуватися ціле повідомлення. Прикладом може слугувати метод, реалізований у програмі *MandelSteg* [74], яка в якості контейнера генерує фрактал Мандельброта (*Mandelbrot fractal*), або ж апарат функцій імітації [67].

За способом доступу до приховуваної інформації розрізняють методи для *потокових* (*безперервних*) контейнерів і методи для фіксованих (*обмеженої довжини*) контейнерів (докладніше див. підрозділ 2.3).

За способом організації контейнери, подібно завадозахищеним кодам, можуть бути *систематичними* і *несистематичними* [3]. У перших можна вказати конкретні місця стеганограми, де знаходяться інформаційні біти власне контейнера, а де шумові біти, призначені для приховування інформації (як, наприклад, у широко поширеному методі найменшого значущого біту). У випадку несистематичної організації контейнера такий поділ не можливий. У цьому разі для виділення прихованої інформації необхідно обробляти вміст усієї стеганограми.

За використовуваним принципом приховування методи комп'ютерної стеганографії поділяють на два основних класи: методи *безпосередньої заміни* і *спектральні* методи. Якщо перші, використовуючи надлишковість інформаційного середовища в просторовій (для зображення) або часовій (для звуку) області, полягають в заміні малозначимої частини контейнера бітами секретного повідомлення, то другі для приховування даних використовують спектральне представлення елементів середовища, куди вбудовуються приховувані дані (наприклад, до різних коефіцієнтів масивів дикретно-косинусних перетворень, перетворень Фур'є, Карунена-Лоева, Адамара, Хаара тощо) [5].

Основним напрямком комп'ютерної стеганографії є використання властивостей саме надлишковості контейнера-оригінала. Але при цьому треба зважати на те, що в результаті приховування інформації відбувається спотворення деяких статистичних властивостей контейнера або ж порушення його структури. Це необхідно враховувати для зменшення демаскувальних ознак.

В особливу групу можна також виділити методи, що **використовують спеціальні властивості форматів представлення файлів** [3]:

– зарезервовані для розширення поля файлів, які зазвичай заповнюються нулями і зазвичай не враховуються програмою;

– спеціальне форматування даних (зсування слів, речень, абзаців або обирання визначених позицій символів);

– використання незадіяних ділянок на магнітних та оптичних носіях;

– видалення файлових заголовків-ідентифікаторів тощо.

В основному, для таких методів характерні низький ступінь скритності, низька пропускна здатність і слабка продуктивність.

За призначенням розрізнюють стеганометоди власне для *прихованого передавання* (або *прихованого збереження*) даних і методи для приховування даних у цифрових об'єктах з метою захисту авторських прав на них.

За типами контейнера виділяють стеганографічні методи із контейнерами у вигляді тексту, аудіофайлу, зображення та відео.

Надалі пропонується розглянути докладніше стеганографічні методи приховання даних у нерухомих зображеннях, в аудіосигналах та текстових файлах.

5.3. Приховування даних у нерухомих зображеннях

Більшість досліджень присвячена використанню в якості стеганоконтейнерів саме зображень. Це обумовлено наступними причинами:

– існуванням практичної необхідності захисту цифрових фотографій, картин, відео від протизаконного тиражування й розповсюдження;

– відносно великим обсягом цифрового представлення зображень, що дозволяє вбудовувати ЦВЗ великого обсягу або ж підвищувати стійкість цього вбудовування;

– заздалегідь відомим (фіксованим) розміром контейнера, відсутністю обмежень, що накладаються вимогами приховання у реальному часі;

– наявністю в більшості реальних зображень текстурних областей, що мають шумову структуру і найкращим чином підходять для вбудовування інформації;

– слабкою чутливістю людського ока до незначних змін кольорів зображення, його яскравості, контрастності, вмісту в ньому шуму, спотворень поблизу контурів;

– зрештою, добре розробленими останнім часом методами цифрової обробки зображень.

Але, як зазначається в [5], остання причина викликає й значні труднощі в забезпеченні стійкості ЦВЗ: чим більш досконалими стають методи компресії, тим менше лишається

можливостей для вбудовування сторонньої інформації. Розвиток теорії й практики алгоритмів компресії зображень призвів до зміни уявлень про техніку вбудовування ЦВЗ. Якщо спочатку пропонувалося вбудовувати інформацію в незначущі біти для зменшення візуальної помітності, то сучасний підхід, навпаки, полягає у вбудовуванні ЦВЗ до найбільш істотних областей зображень, руйнування яких призводитиме до повної деградації самого зображення. Тому цілком зрозумілим є необхідність врахування стеганоалгоритмами не лише алгоритмів компресії зображень, але й властивостей зорової системи людини (ЗСЛ).

5.3.1. Основні властивості ЗСЛ, які необхідно враховувати при побудові стеганоалгоритмів

Властивості ЗСЛ можна поділити на дві групи: *низькорівневі* (“фізіологічні”) і *високорівневі* (“психофізіологічні”) [75,76]. Майже до середини 90-х р.р. дослідники брали до уваги, головним чином, низькорівневі властивості зору. В останні роки позначилася тенденція побудови стеганоалгоритмів з урахуванням й високорівневих характеристик ЗСЛ.

Виділяють три найважливіші низькорівневих властивості, які впливають на помітність стороннього шуму в зображенні: *чутливість до зміни яскравості (контрасту) зображення*, *частотна чутливість* й *ефект маскування*.

На рис.5.2 зображено залежність мінімального контрасту $\Delta I / I$ від яскравості I.

Як видно, для середнього діапазону зміни яскравості, контраст приблизно постійний, тоді як для малих і великих яскравостей значення порогу нерозрізненості (ΔI) зростає. Встановлено, що $\Delta I \approx (0.01 \div 0.03) \cdot I$ для середніх значень яскравості.

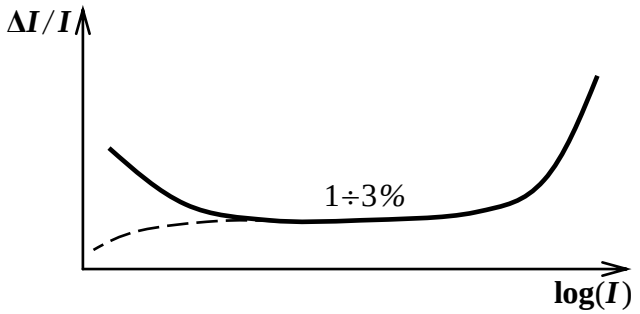


Рис.5.2. Чутливість до зміни контрасту і поріг нерозрізненості ΔI

Крім того, у [5] відзначено, що результати новітніх досліджень суперечать “класичній” теорії і показують, що при малих значеннях яскравості поріг нерозрізненості зменшується, тобто ЗСЛ є більш чутливою до шуму в цьому діапазоні [5].

Частотна чутливість ЗСЛ проявляється в тому, що людина є набагато більше сприйнятливою до низькочастотного (НЧ), чим до високочастотного (ВЧ) шуму. Це пов’язане з нерівномірністю амплітудно-частотної характеристики ЗСЛ.

Складові елементи ЗСЛ розщеплюють відеосигнал, що надходить, на окремі складові, кожна з яких збуджує нервові закінчення ока через низку підканалів. Виділювані оком складові мають різні *просторові* й *частотні* характеристики, а також різну просторову орієнтацію (горизонтальну, вертикальну, діагональну) [77]. У випадку одночасного впливу на око двох складових з подібними між собою характеристиками збуджуються одні й ті самі підканали. Це призводить до *ефекту маскування*, який полягає у збільшенні порогу виявлення зорового сигналу в присутності іншого сигналу, що має аналогічні характеристики. Тому, адитивний шум набагато помітніше на НЧ (однотонних) ділянках зображення, порівняно з ВЧ ділянками, тобто в останньому випадку спостерігається маскування. Найбільш сильно даний ефект проявляється, коли обидва сигнали мають однакову орієнтацію і місце розташування [5].

Частотна чутливість тісно пов’язана з яскравісною. Відомий також і вираз для визначення порогу маскування на основі відомої яскравісної чутливості, що дозволяє знайти метрику спотворення зображення, яка б враховувала властивості ЗСЛ. Такого типу математичні моделі добре розроблені для випадку квантування коефіцієнтів дискретного косинусного перетворення зображення, оскільки саме воно застосовується в стандарті *JPEG*.

Ефект маскування у *просторовій області* може бути пояснений шляхом побудови стохастичних моделей зображення. При цьому зображення представляється у вигляді марківського випадкового поля, розподіл імовірностей якого описується, наприклад, узагальненим гаусівським законом.

У [5] пропонується наступна узагальнена схема вбудовування даних у зображення:

1) виконати фільтрацію зображення за допомогою орієнтованих смугових фільтрів. При цьому одержується розподіл енергії за частотно-просторовими компонентами;

2) обчислити поріг маскування на основі знання локальної величини енергії;

3) масштабувати значення енергії впроваджуваної інформації у кожному компоненті таким чином, щоб воно було менше за поріг маскування.

Багато алгоритмів вбудовування даних так чи інакше використовують цю схему.

Високорівневі властивості ЗСЛ поки що рідко враховуються при побудові стеганоалгоритмів [5]. Їхньою відмінністю від низькорівневих є те, що ці властивості проявляються "вторинно", обробивши первинну інформацію від ЗСЛ, мозок видає команди на "підстроювання" зорової системи під зображення. Основні з цих властивостей наступні:

1) *чутливість до контрасту* – висококонтрастні ділянки зображення і перепади яскравості звертають на себе більш значну увагу;

2) *чутливість до розміру* – більші ділянки зображення є "помітнішими" порівняно із меншими за розміром. Причому існує поріг насичення, коли подальше збільшення розміру не відіграє ролі;

3) *чутливість до форми* – довгі й тонкі об'єкти викликають більшу увагу, чим закруглені й однорідні;

4) *чутливість до кольорів* – деякі кольори (наприклад, червоний) є "помітнішим" за інші. Цей ефект підсилюється, якщо фон заднього плану відрізняється від кольорів фігур на ньому;

5) *чутливість до місця розташування* – людина схильна у першу чергу розглядати центр зображення. Також уважніше розглядаються фігури переднього плану, ніж заднього;

6) *чутливість до зовнішніх подразників* – рух очей спостерігачів залежить від конкретної обстановки, від отриманих їм перед переглядом або під час нього інструкцій, додаткової інформації.

Останнім часом створена достатня кількість методів приховування даних в цифрових зображеннях, що дозволяє провести їхню класифікацію і виділити наступні узагальнюючі групи [3]:

– методи заміни в просторовій (часовій) області;
– методи приховування в частотній області зображення;
– широкосмужні методи;
– статистичні (стохастичні) методи;
– методи спотворення;
– структурні методи.

Далі розглядаються особливості, які характерні для кожної з виділених груп. Паралельно наводяться програмні модулі у системі MathCAD, які дозволяють реалізувати той або інший метод, а також проміжні і кінцеві результати відповідних стеганографічних перетворень. До кожного модуля наводяться стислі пояснення щодо його функціонування і використаних функцій системи MathCAD. При цьому найбільш вичерпні пояснення даються до перших методів, що розглядаються. В подальшому значення функцій, зміст яких було розкрито раніше, не пояснюються. Повну уяву про можливості і правила використання (синтаксис) типових об'єктів мови MathCAD можна отримати, наприклад, з [25,26]. Крім того у даній роботі в *додатках A, B, C, D* подана стисла інформація стосовно вбудованих операторів, функцій і директив, а також системних змінних та програмних операторів системи MathCAD.

5.3.2. Приховування даних у просторовій області зображення

Алгоритми, що описуються в даному підрозділі, вбудовують приховувані дані в області первинного зображення. Їх перевагою є те, що для вбудовування нема необхідності виконувати обчислювально складні і тривалі лінійні перетворення зображень.

Кольорове зображення $\boldsymbol{C}$ представлятимемо через дискретну функцію, яка визначає вектор кольору $\boldsymbol{c}(\boldsymbol{x}, \boldsymbol{y})$ для кожного пікселя зображення $(\boldsymbol{x}, \boldsymbol{y})$, де значення кольору задає трикомпонентний вектор у колірному просторі. Найбільш розповсюдженим способом передавання кольору є модель *RGB*, у якій основні кольори – червоний, зелений і синій, а будь-який інший колір може бути представлений у вигляді зваженої суми основних кольорів. Вектор кольору $\boldsymbol{c}(\boldsymbol{x}, \boldsymbol{y})$ у RGB-просторі представляє інтенсивність основних кольорів. Повідомлення вбудовується за рахунок маніпуляцій колірними складовими $\{\boldsymbol{R}(\boldsymbol{x}, \boldsymbol{y}), \boldsymbol{G}(\boldsymbol{x}, \boldsymbol{y}), \boldsymbol{B}(\boldsymbol{x}, \boldsymbol{y})\}$ або безпосередньо яскравістю $\boldsymbol{\lambda}(\boldsymbol{x}, \boldsymbol{y}) \in \{1, 2, ..., \boldsymbol{L_C}\}$.

Загальний принцип даних методів полягає в заміні надлишкової, малозначимої частини зображення бітами секретного повідомлення. Для видобування повідомлення необхідно знати алгоритм, за яким розміщувалася по контейнеру прихована інформація.

5.3.2.1. Метод заміни найменшого значущого біта

Метод заміни *найменшого значущого біта* (НЗБ, *LSB – Least Significant Bit*) є найпоширенішим серед методів заміни в просторовій області [3,5,9,14,19,20].

Молодший значущий біт зображення несе в собі менше всього інформації. Відомо, що людина у більшості випадків не здатна помітити зміну в цьому біті. Фактично, НЗБ є шумом. Тому його можна використовувати для вбудовування інформації, шляхом заміни найменш значущих бітів пікселів зображення бітами секретного повідомлення При цьому, для зображення в градаціях сірого (кожен піксель зображення кодується одним байтом) обсяг вбудовуваних даних може складати $1/8$ від загального об'єму контейнера. Наприклад, у зображення розміром 512×512 можні вбудувати ~ 32 кБ інформації. Якщо модифікувати два молодших біта (що також практично непомітно), то дану пропускну здатність можна збільшити ще вдвічі.

Популярність даного методу обумовлена його простотою і тим, що він дозволяє приховувати у відносно невеликих файлах досить великі об'єми інформації (пропускна здатність створюваного прихованого каналу зв'язку складає від 12,5 до 30%). Метод зазвичай працює з растровими зображеннями, які представлені у форматі без компресії (наприклад, *GIF* і *BMP*) [3] [2)]. Метод НЗБ має низьку стеганографічну стійкість до атак пасивного та активного порушників. Основним його недоліком є сильна чутливість до найменших спотворень контейнера. Для послаблення цієї чутливості часто додатково застосовують завадостійке кодування.

Перед імпортом зображення-контейнера в документ MathCAD його необхідно підготувати у відповідному редакторі і записати у вигляді файлу в поточний (для формованого документу MathCAD) каталог роботи (слід зауважити, що для уникнення можливих проблем з підтримкою кирилиці бажано, щоб адреса розміщення файлу на диску, як, власне, і ім'я файлу, складалися з латинських символів). MathCAD підтримує формати *BMP, JPEG, GIF, PCX* і *TGA.* Як було зазначено вище, формати *BMP* і *GIF*, дозволяють зберігати зображення практично без втрати їх якості і тому є більш придатними в ролі носіїв інформації.

Розглянемо структуру *BMP*-файлу: він містить точкове (растрове) зображення і складається з трьох основних розділів: заголовку файлу, заголовку растру і растрових даних. *Заголовок файлу* містить інформацію про файл (його тип, об'єм і т.п.). До *заголовку растру* винесена інформація про ширину і висоту зображення, кількість бітів на піксель, розмір растру, глибину кольору, коефіцієнт компресії тощо. Нас передовсім цікавитимуть *растрові дані* – інформація про колір кожного пікселя зображення. Колір пікселя визначається сполученням трьох основних колірних складових – червоного, зеленого і синього (скорочено – RGB), кожній з яких відповідає своє значення інтенсивності, яке може змінюватися від 0 до 255. Отже, за кожен з колірних каналів відповідає 8 бітів (1 байт), а глибина кольору зображення в цілому – 24 біти (3 байти).

[2)] Взагалі, формати *BMP* і *GIF* використовують алгоритми компресії, але останні є найпростішими, що дозволяє зберігати зображення практично без втрати його якості.

1) *Імпорт графічного файлу* виконується операцією *Picture* з позиції *Insert* головного меню програми. У модулі, що при цьому з'явився, необхідно заповнити шаблон даних у лівому нижньому куті, для чого в подвійних лапках ввести ім'я файлу (або ж, за необхідності, – повний шлях його розміщення на диску) і натиснути клавішу *Enter*.

Приклад кольорового зображення, відтвореного за допомогою операції *Picture* приведений на рис.5.3. Дане зображення має розмір 128×128 пікселів, глибина кольору – 24 біти. Для можливості обробки зображення, необхідно перевести колірні характеристики кожного його пікселя у числову матрицю. Для виконання цієї операції застосовується функція **READRGB**("ім'я_файлу"), що повертає масив, з трьох підмасивів, які, у свою чергу, несуть інформацію про розклад кольорового зображення на колірні компоненти *R, G* і *B :*

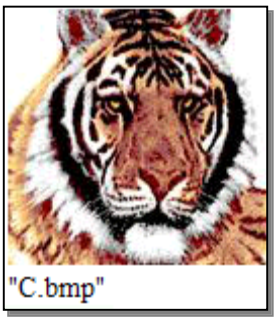
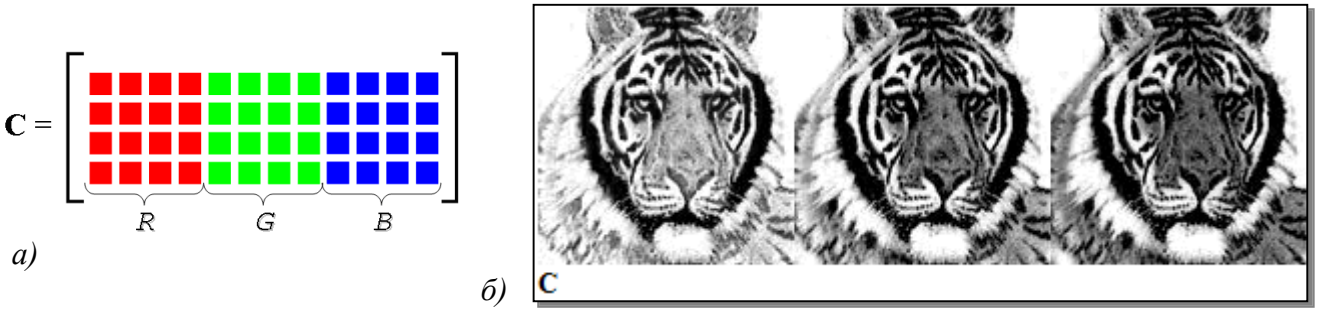


Рис.5.3. Зображення-контейнер

C := **READRGB**("C.bmp").

При цьому три колірних компоненти розміщуються одна за одною у спільному масиві **C** (рис.5.4,*а*). На рис.5.4,*б* приведена графічна інтерпретація масиву **C** у вигляді зображення з градаціями сірого. Образ ліворуч характеризує інтенсивність *червоного* у кожному пікселі зображення "C.bmp", середній – інтенсивність *зеленого*, а той, що праворуч, – інтенсивність *синього*.



а) *б)*

Рис.5.4. Графічна інтерпретація масиву колірних компонентів контейнера-оригіналу

Для відокремлення колірних складових можна використати вбудовані функції виділення відповідних колірних компонентів, кожна з яких повертає масив, який відповідає певному колірному компоненту графічного файлу:

R := **READ_RED**("C.bmp"); **G** := **READ_GREEN**("C.bmp"); **B** := **READ_BLUE**("C.bmp").

2) В якості повідомлення, яке треба приховати, використаємо, наприклад, перші вісім абзаців зі вступу до даної роботи. Текст повідомлення збережемо у файлі "M.txt" поточного для формованого документу MathCAD каталогу роботи у такому форматі: тип файлу – *звичайний текст (*.txt)*; кодування – *кирилиця (Windows)* [3]. Імпорт текстового повідомлення можна виконати за допомогою функції **READBIN**("ім'я_файлу", "тип_формату_даних"). У даному разі дані представлені як 8-бітове беззнакове ціле число (байт): **M** := **READBIN**("M.txt", "byte").

Результатом обчислення даного виразу є матриця-стовпець (вектор), кожен елемент якої відповідає розширеному ASCII-коду відповідного символу (літери) повідомлення, яке імпортується. У десятковому вигляді коди символів можуть приймати значення від 0 до 255; у двійковому вигляді для цього є достатнім використання 8 бітів на один символ – так зване однобайтове кодування, на що, зрештою, вказує параметр "byte" як аргумент функції READBIN (див. додатки *В* та *Е*).

---

[3] Існує можливість проведення приховання файлів будь-якого формату. Єдина умова, що при цьому повинна виконуватися – обрання файлу-контейнеру належного об'єму (наприклад, орієнтовне співвідношення між об'ємами файлу-контейнера і файлу-повідомлення для методу НЗБ – 8:1).

Фрагмент імпортованого повідомлення, а саме коди перших 16 символів (включно з пробілами) у десятковому і двійковому видах, зображено на рис.5.5 [4]. Необхідно зауважити, що за умовчанням нижня границя індексації масивів дорівнює 0. В даній роботі індексація починається з 1 (якщо не буде оговорено інше), що, зокрема, можна встановити записом **ORIGIN** := 1 на початку документу або ввести 1 у позицію *Array Origin* на вкладці *Built-In Variables* діалогового вікна *Worksheet Options* з меню *Tools* системи MathCAD.

Перевірку імпортування файлу повідомлення (якщо в якості повідомлення використовується звичайний текст) можна виконати записом **vec2str**(**M**). Зазначена функція повертає рядок символів, які відповідають вектору **M** ASCII-кодів [5].

M =

	1
1	178
2	237
3	244
4	238
5	240
6	236
7	224
8	246
9	179
10	255
11	32
12	186
13	32
14	238
15	228
16	237

M =

	1
1	10110010b
2	11101101b
3	11110100b
4	11101110b
5	11110000b
6	11101100b
7	11100000b
8	11110110b
9	10110011b
10	11111111b
11	100000b
12	10111010b
13	100000b
14	11101110b
15	11100100b
16	11101101b

Рис.5.5. Фрагмент повідомлення, яке підлягає прихованню

3) Перед приховуванням текстового файлу "M.txt" у контейнері "C.bmp", його можна захистити *криптографічним кодуванням*. Для наочності і стислості викладення використаємо модифікований *код Віженера*. Взагалі ж для стійкості стеганоповідомлення до можливих спотворень рекомендується застосовувати код з виправленням помилок.

Алфавіт джерела повідомлення задаємо у вигляді ASCII-кодів: **i** := 1..256; $\mathbf{A}_{\mathbf{i}}$:= **i** – 1 (для більшої захищеності, елементи вектору **A** можна переставити за певним законом, що відповідним чином враховувати при розшифровуванні).

Об’єм алфавіту джерела: **Na** := **rows**(**A**), де **rows**(**A**) – функція, яка повертає кількість рядків масиву **A**. У даному разі: **Na** = 256 символів.

З символів алфавіту задаємо секретний ключ, наприклад: **K** := "@J|eKc-l98O". Кількість символів у ключі (за відповідною функцією): **Nk** := **strlen**(**K**), **Nk** = 11 символів.

Об’єм повідомлення, яке підлягає кодуванню: **Nm** := **rows**(**M**), **Nm** = 5390 символи. (включно із скритими службовими ASCII-символами “переводу рядка” і “повернення каретки”, див. додаток *E*).

---

[4] Для зміни формату відображуваних даних у MathCAD необхідно виконати наступне: вибрати операцію *Result* з меню *Format* (або двічі натиснути ліву кнопку маніпулятора безпосередньо на виведених даних) для появи діалогового вікна *Result Format*; на вкладці *Display Options* у випадаючому меню *Radix* обрати необхідну основу системи числення: *decimal* (десяткова), *binary* (двійкова), *octal* (восьмерична) або *hexadecimal* (шістнадцятирична).

[5] Можлива ситуація некоректного відображення кириличних символів в одержуваному рядку. Цю проблему можна вирішити використанням кодової сторінки 1251 замість 1250 і 1252, що можна задати заміною параметрів "1250" і "1252" на параметр "1251" у реєстрі ОС Windows:
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Nls\CodePage
STRING 1250, значення "c_1251.nls" (якщо було "c_1250.nls")
STRING 1252, значення "c_1251.nls" (якщо було "c_1252.nls")
За можливі проблеми, що виникли після редагування реєстру автор роботи відповідальності не несе. Для більш безпечного редагування реєстру краще скористатися спеціалізованою програмою, наприклад, XP Tweaker RE. Крім того, некоректність відображення кириличних символів у системі MathCAD жодним чином не впливає на результати стеганографічних перетворень, оскільки алгоритми оперують з двійковими відображеннями ASCII-кодів, а видобуте з контейнера повідомлення можна без проблем прочитати в будь-якому з поширених текстових редакторів.

Розширюємо ключ на довжину повідомлення (**Nm**), використовуючи програмний модуль (М.1). У даному модулі функція **str2vec**(**K**) перетворює рядок символів **K** на вектор їх ASCII-кодів. Програмний оператор **for** організовує цикл зміни **i** (змінна циклу) із заданою кількістю повторів (у даному випадку – від 1 до **Nm**). Функція **mod**(**i**,**Nk**) повертає залишок від ділення **i** на **Nk**.

```
K' := | K ← str2vec(K)
      | for i ∈ 1 .. Nm
      |   | r ← mod(i , Nk)
      |   | K'_i ← K_r    if r > 0
      |   | K'_i ← K_Nk   if r = 0
      | K'
```
(М.1)

Проводимо *кодування повідомлення*, використовуючи модуль (М.2).

```
M_cod := | for j ∈ 1 .. Nm
         |   | for i ∈ 1 .. Na
         |   |   | m ← i   if M_j = A_i
         |   |   | n ← i   if K'_j = A_i
         |   | r ← mod(m + n , Na)
         |   | M_cod_j ← A_r    if r > 0
         |   | M_cod_j ← A_Na   if r = 0
         | M_cod
```
(М.2)

4) Для того, щоб при розпакуванні контейнеру з отриманої множини символів можна було чітко визначити початок і кінець саме прихованого повідомлення, доцільно ввести відповідні *секретні мітки*, які б обмежували цей корисний зміст.

Для унеможливлення прийняття за мітки символів випадкового утворення перші повинні складатися з достатньої кількості символів. Крім того, для зниження імовірності виявлення міток при проведенні стеганоаналізу бажано, щоб коди цих символів були достатньо рознесені на ASCII-вісі (наприклад, використовувати поруч з латинськими символами символів кирилиці та службових символів – так звана *транслітерація*; застосування псевдовипадкових послідовностей кодів символів тощо). Нехай мітки мають наступний вигляд: μ_s := "n0ч@m0к"; μ_e := "KiHeu,6".

Обмежуючі мітки додаємо до тексту закодованого повідомлення, для чого використовуємо функцію **stack**(**A**,**B**,...), яка дозволяє об'єднувати записані через кому масиви. Об'єднання відбувається шляхом "насадження" матриці **A** на матрицю **B**; отриманої таким чином матриці – на наступну матрицю (якщо така є) і т.д. Зрозуміло, що початкові матриці повинні мати однакову кількість стовпців, тому необхідно перетворити мітки з рядків на вектори ASCII-кодів. Отже, **sMe** := **stack**(**str2vec**(μ_s), **M_cod**, **str2vec**(μ_e)).

Загальна кількість символів в приховуваному повідомленні: **rows**(**sMe**) = 5404 симв. Кількість НЗБ контейнера, яка для цього потрібна (8 біт/символ): 8·**rows**(**sMe**) = 43232 біт. Загальна кількість НЗБ контейнера: **rows**(**C**)·**cols**(**C**) = 3·128·128 = 49152 > 43232 біт. Таким чином, файл зображення має достатній об'єм для того, щоб приховати повідомлення.

5) Для подальших обчислень буде необхідним переведення десяткового числа (яким за умовчуванням кодується кожен символ) у формат двійкового. Також знадобиться і зворотне перетворення. Оскільки дані функції в MathCAD є відсутніми (існує, як це було показано вище, лише можливість перетворення формату відповіді, що для наших цілей не є прийнятним), пропонується використати наступні модулі, зміст яких є цілком очевидним.

Перетворення двійкового числа **x**, яке задане матрицею-стовпцем, причому 1-й елемент матриці – наймолодший розряд числа, на десяткове виконується за допомогою модуля (М.3).

$$\mathbf{B2D}(\mathbf{x}) := \sum_{i=1}^{8} x_i \cdot 2^{i-1} \qquad \text{(М.3)}$$

Перетворення десяткового числа на двійкове реалізується модулем (М.4). Причому функція **mod**(**x**,2) повертає залишок від ділення **x** на 2 («0», якщо **x** парне, і «1», якщо **x** непарне). Функція **floor**(...) повертає найбільше ціле число, яке менше або дорівнює дійсному значенню аргументу.

```
D2B(x) := | for i ∈ 1 .. 8
          |   | V_i ← mod(x , 2)
          |   | x ← floor(x / 2)
          | V
```
(М.4)

6) Для більшої зручності і наочності подальших дій, розгорнемо матрицю **C** у вектор, тимчасово змінивши порядок колірних матриць з **R**-**G**-**B** на **B**-**G**-**R**, що підвищить захищеність прихованої інформації (у даному випадку можна використати більш надійні, але й складніші, алгоритми). У нашому випадку застосуємо модуль (М.5), в якому функція **augment**(**A**,**B**,...) об'єднує матриці **A**, **B**, ..., що мають однакову кількість рядків (об'єднання проводиться стовпець до стовпця, матриці повинні мати однакову кількість рядків). Операція $\mathbf{C'}^{\langle i \rangle}$ дозволяє обирати **i**-й стовпець з матриці **C'**, кожен з яких згодом додається до результуючого вектора **Cv**. Функція **cols**(**C'**) повертає кількість стовпців масиву **C'**.

$$\begin{aligned}&\mathbf{Cv} := \mathbf{C'} \leftarrow \mathbf{augment}(\mathbf{B}, \mathbf{G}, \mathbf{R})\\ &\quad \mathbf{Cv} \leftarrow \mathbf{C'}^{\langle 1 \rangle}\\ &\quad \mathbf{for}\ \ i \in 2 .. \mathbf{cols}(\mathbf{C'})\\ &\quad\quad \mathbf{Cv} \leftarrow \mathbf{stack}\left(\mathbf{Cv}, \mathbf{C'}^{\langle i \rangle}\right)\end{aligned} \qquad \text{(М.5)}$$

7) На основі вектора **Cv** формуємо новий вектор, що вже міститиме приховане закодоване повідомлення (модуль (М.6)).

Кожен символ закодованого повідомлення (операція циклу **for** $\mu \in 1..\mathbf{rows}(\mathbf{sMe})$) переводиться у двійковий формат (змінна **b**), кожен з 8 розрядів якого записується замість НЗБ числа, яке відповідає інтенсивності того чи іншого кольору відповідного пікселя (останнє також попередньо переводиться у двійковий формат (змінна **P**)).

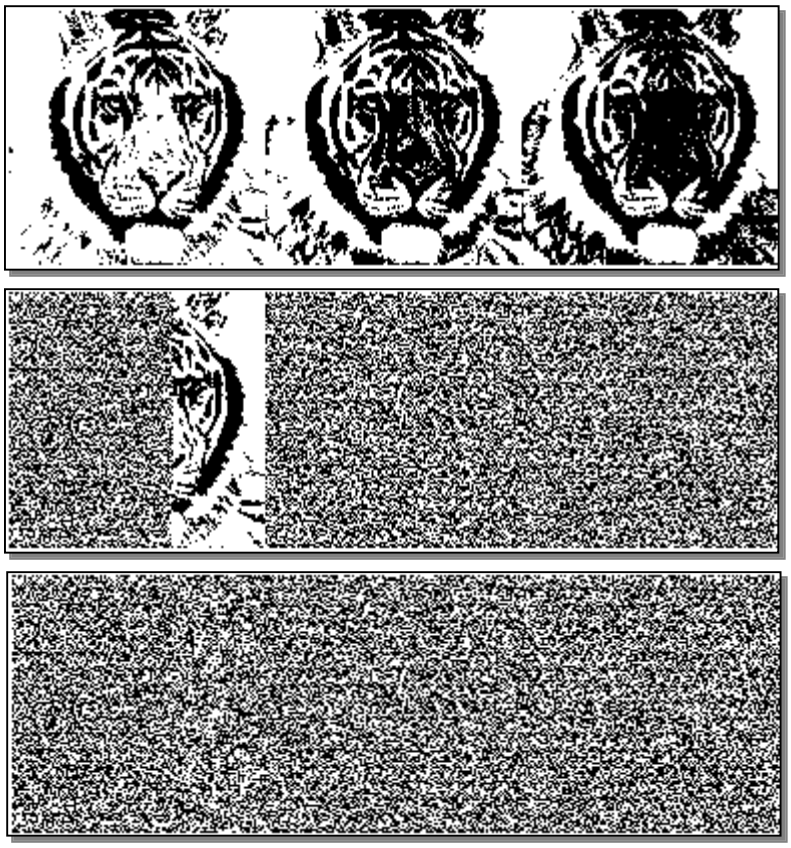
Після проведеної зміни модифіковане двійкове число **P** переводиться у формат десяткового і записується до відповідної позиції вектора **Sv**.

Після обробки останнього символу повідомлення **sMe** проводиться модифікація елементів масиву **Cv**, які ще не зазнали змін. Молодшим бітам кожного з таких елементів присвоюються значення 0 або 1 (у даному випадку – за рівномірним законом розподілу (функція **round**(...) повертає округлене до найближчого цілого значення аргументу), хоча більш правильно було б провести дослідження закону розподілу значень вже модифікованих молодших бітів і відповідним чином змінювати ті, що лишилися (дана процедура є темою окремого дослідження і в рамках даної роботи не приводиться).

$$\begin{aligned}&\mathbf{Sv} := \mathbf{for}\ \ \mu \in 1 .. \mathbf{rows}(\mathbf{sMe})\\ &\quad \mathbf{b} \leftarrow \mathbf{D2B}\left(\mathbf{sMe}_{\mu}\right)\\ &\quad \mathbf{for}\ \ i \in 1 .. 8\\ &\quad\quad \mathbf{P} \leftarrow \mathbf{D2B}\left[\mathbf{Cv}_{i+8\cdot(\mu-1)}\right]\\ &\quad\quad * \ \mathbf{P}_1 \leftarrow \mathbf{b}_i\\ &\quad\quad\quad \mathbf{Sv}_{i+8\cdot(\mu-1)} \leftarrow \mathbf{B2D}(\mathbf{P})\\ &\mathbf{for}\ \ j \in \mathbf{rows}(\mathbf{Sv}) + 1 .. \mathbf{rows}(\mathbf{Cv})\\ &\quad \mathbf{P} \leftarrow \mathbf{D2B}\left(\mathbf{Cv}_j\right)\\ &\quad * \ \mathbf{P}_1 \leftarrow \mathbf{round}(\mathbf{rnd}(1))\\ &\quad\quad \mathbf{Sv}_j \leftarrow \mathbf{B2D}(\mathbf{P})\\ &\mathbf{Sv}\end{aligned} \qquad \text{(М.6)}$$

Це унеможливлює згодом виявити факт модифікованості зображення. В іншому випадку, проаналізувавши зображення, побудоване з одних лише НЗБ контейнеру, порушник у більшості випадків (якщо символів повідомлення "не вистачило" на весь контейнер) виявить границю введення даних і при певних зусиллях зможе видобути приховану інформацію. Звичайно, цю інформацію ще необхідно розшифрувати, але факт її наявності вже буде розкрито і питання захисту повернеться до криптографічної стійкості використаного кодування. На рис.5.6 в якості прикладу наведено графічні інтерпретації масивів колірних компонентів, відтворені лише за НЗБ (0/1) контейнера-оригіналу *(а)*, контейнера-результату без модифікації *(б)* і з модифікацією *(в)* надлишкових бітів. Як видно, за відсутності "дописування" при неповному заповненні контейнера чітко простежується границя введення повідомлення (рис.5.6,*б*) – слід нагадати, що підмасиви **R** і **B** були поміняні місцями, тому останньою модифікувалася матриця **R**, яка, зрештою, і виявилася заповненою не до кінця. "Дописування" рівноімовірними 0 і 1 дещо виправляє ситуацію, хоча при додатковому стеганоаналізі на закон розподілу значень НЗБ невідповідність відразу буде виявлена, що ще раз каже про необхідність попереднього аналізу розподілу значень вже модифікованих бітів або ж хоча б продублювати частину повідомлення для заповнення всього контейнеру. Також очевидна відмінність між рисунками *а* і *в*. Тому бажаною є унікальність зображення, яке планується

використати в якості контейнера. Тут треба зауважити, що всі графічні контейнери умовно поділяються на "чисті" і "зашумлені" [95]. У перших простежується зв'язок між молодшим й іншими сімома бітами колірних компонентів, а також залежність між наймолодшими бітами. Пакування повідомлення в "чисте" зображення руйнує існуючі залежності, що, як було показано вище, легко виявляється. Якщо ж зображення вже початково зашумлене (скановане зображення, цифрова фотографія тощо), то визначення стороннього вкладення стає на порядок важчим, хоча й можливим при використанні теорії імовірностей і математичної статистики.



а)

б)

в)

Рис.5.6. Масиви колірних компонентів, відтворені за НЗБ

Також можна зазначити, що для більшої прихованості біти повідомлення слід вносити не послідовно, а лише до кожного 2-го чи навіть 3-го пікселя, або ж підпорядкувати внесення певному, відомому лише авторизованим особам, закону. Відповідні модифікації легко здійснити шляхом внесення відповідних незначних змін до модуля (М.6).

8) Отриманий за допомогою модуля (М.6) вектор **Sv** згортаємо в матрицю **S**, що має розмірність первинної матриці **C** (модулі (М.7) і (М.8)).

$$\begin{aligned} &\mathbf{S'} := \mathbf{for}\ \ i \in 1, 2 .. \mathbf{cols(C)} \\ &\qquad \mathbf{S'}^{\langle i \rangle} \leftarrow \mathbf{submatrix}[\mathbf{Sv}, (i-1)\cdot\mathbf{rows(C)}+1, i\cdot\mathbf{rows(C)}, 1, 1] \end{aligned} \tag{М.7}$$

Функція **submatrix**(**A**, **x**, **X**, **y**, **Y**) повертає частину матриці **A**, яка складається з елементів, спільних для рядків від **x** до **X** та стовпців від **y** до **Y** включно.

9) Користуючись цією ж функцією, виокремлюємо з масиву **S'** колірні матриці та розставляємо їх по своїх місцях (**R↔B**), одержуючи контейнер-результат **S** (М.8).

На рис.5.7 показана графічна інтерпретація масиву **S** у вигляді зображення з градаціями сірого та відтворене за колірними складовими зображення-контейнер з прихованим повідомленням.

$$\begin{aligned} \mathbf{B_M} &:= \mathbf{submatrix}\left(\mathbf{S'}, 1, \mathbf{rows(C)}, 1, \frac{\mathbf{cols(C)}}{3}\right) \\ \mathbf{G_M} &:= \mathbf{submatrix}\left(\mathbf{S'}, 1, \mathbf{rows(C)}, \frac{\mathbf{cols(C)}}{3}+1, 2\cdot\frac{\mathbf{cols(C)}}{3}\right) \\ \mathbf{R_M} &:= \mathbf{submatrix}\left(\mathbf{S'}, 1, \mathbf{rows(C)}, 2\cdot\frac{\mathbf{cols(C)}}{3}+1, \mathbf{cols(C)}\right) \\ \mathbf{S} &:= \mathbf{augment}(\mathbf{R_M}, \mathbf{G_M}, \mathbf{B_M}) \end{aligned} \tag{М.8}$$

Порівнюючи рис.5.7 з рис.5.3 і 5.4 можна зробити висновок про відсутність помітних візуальних відхилень.

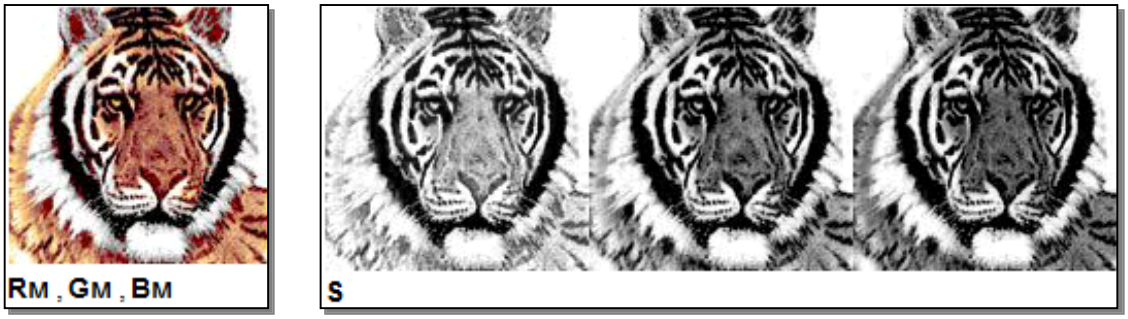


Рис.5.7. Контейнер-результат та його інтерпретація у вигляді масиву колірних компонентів

Залишається лише записати масив **S** у файл: **WRITERGB**("S_LSB.bmp") := **S**. Цілком очевидно, що об'єм отриманого файлу відповідатиме об'єму файлу зображення-оригіналу.

10) Для дослідження впливу на ступінь прихованості того, до якого з розрядів числа, яке характеризує ту або іншу властивість пікселя (у нашому випадку – інтенсивність певного кольору), заноситиметься секретна інформація, у модулі (М.6) у помічених зірочкою рядках слід замість індексу "1" внести індекс, який відповідає розряду, що модифікуватиметься. На рис.5.8 зображено результат, отриманий при внесенні даних до 8-го (найстаршого) біта числа, яке відповідає інтенсивності кольорів.

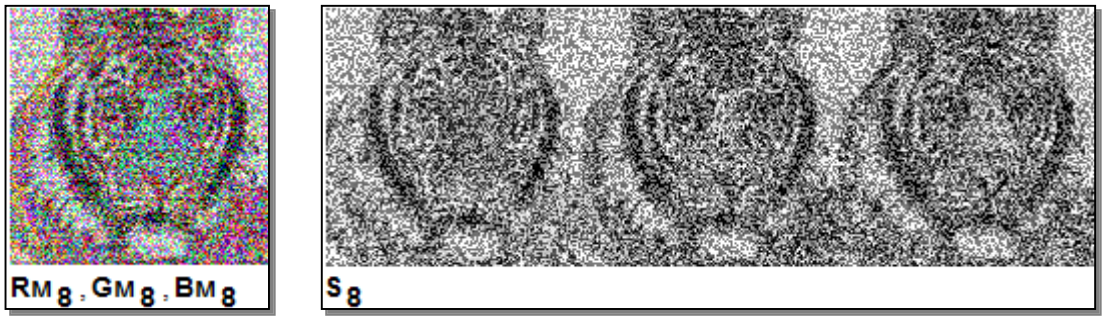


Рис.5.8. Кольорове зображення та масив колірних складових в разі внесення приховуваних даних до 8-го біту інтенсивностей кольорів.

Встановлено, що візуально не є помітним, якщо в якості "носіїв" використовувати не лише наймолодший, але й наступний за ним біт кожного із зазначених вище чисел (особливо, якщо в зображенні відсутні великі однотонні ділянки). Отже, в якості одного з можливих ступенів захисту бачиться використання змінюваного за певним законом почергового запису до цих двох бітів. Або ж, жертвуючи прихованістю, можна вдвічі збільшити ємність контейнера.

11) Розглянемо процес *розпакування прихованого повідомлення*. Попередньо знаючи, що повідомлення було поміщене до масиву колірних компонентів, виділяємо відповідні кожному кольору підмасиви, переводячи значення колірних характеристик кожного пікселя зображення, що містить у собі скрите закодоване повідомлення, у числові матриці [6]:

R* := **READ_RED**("S_LSB.bmp"); **G*** := **READ_GREEN**(•); **B*** := **READ_BLUE**(•).

З отриманих матриць, відповідним чином змінюючи їх порядок, подібно до (М.5) формуємо вектор **Sv*** (модуль (М.9)). Належним чином обробляючи кожні вісім елементів отриманого вектора, розпаковуємо приховане повідомлення, використовуючи модуль (М.10).

$$\begin{aligned} \mathbf{Sv^*} := \;& \mathbf{S'} \leftarrow \mathbf{augment}(\mathbf{B^*}, \mathbf{G^*}, \mathbf{R^*}) \\ & \mathbf{Sv^*} \leftarrow \mathbf{S'}^{\langle 1 \rangle} \\ & \mathbf{for}\ \ i \in 2 .. \mathbf{cols}(\mathbf{S'}) \\ & \quad \mathbf{Sv^*} \leftarrow \mathbf{stack}\left(\mathbf{Sv^*}, \mathbf{S'}^{\langle i \rangle}\right) \end{aligned} \qquad \text{(М.9)}$$

Слід зазначити, що оскільки наперед невідомо, яку частину вектора **Sv*** займає саме

[6] Тут і надалі символом ***** позначатимуться параметри, що відносяться до приймальної сторони. Також за умовчанням вважатиметься, що приймальна сторона обізнана з алгоритмом вбудовування.

корисна інформація, до уваги беруться всі його елементи. Значення кожного елементу формованого при цьому вектора **Mf*** являють собою коди символів "квазіповідомлення", які обчислюються у зворотному до (М.6) порядку: кожен молодший розряд вісімки перетворених у двійковий формат елементів вектора **Sv*** формує двійкове число коду символу, формат якого згодом перетворюється на десятковий. Отримане число присвоюється **μ**-му елементу вектора **Mf*.**

```
Mf* := | for μ ∈ 1 .. rows(Sv*)/8
       |   for i ∈ 1 .. 8
       |     P ← D2B[Sv*_(i+8·(μ−1))]
       |     b_i ← P_1
       |   Mf*_μ ← B2D(b)
       |   Mf*_μ ← Mf*_μ + 32.5  if  Mf*_μ < 32
       | Mf*
```
(М.10)

12) У зв'язку з неможливістю оброблення рядковими функціями 12-ї версії MathCAD символів, ASCII-коди яких мають значення від 0 до 31 включно (за винятком службових символів *LF* (код 10) і *CR* (код 13)), додатково вноситься заміна значень 0, 1, 2, ..., 31 елементів вектора **Mf*** відповідним додаванням до кожного з них коефіцієнту 32.5 (коефіцієнт є дробом для того, щоб у подальшому було можливим відрізнити "справжні" значення елементів масиву від тих, що перед цим мали неформатні значення). Таку заміну, звичайно, відповідним чином необхідно врахувати у подальшому, для чого запам'ятаємо номери рядків вектору **Mf***, елементи яких мають дробові значення (модуль (М.11)).

При цьому, до 1-го стовпця формованого масиву **N** заносяться номери елементів значення яких дорівнюють 32.5 (колишні нулі), до 2-го стовпця – номери елементів із значенням 33.5 (колишні одиниці) і т.д.

```
N := | for s ∈ 0, 1 .. 31
     |   i ← 1
     |   for μ ∈ 1 .. rows(Mf*)
     |     if Mf*_μ = s + 32.5
     |       N_(i, s+1) ← μ
     |       i ← i + 1
     | N
```
(М.11)

13) Знаючи, що текст корисної інформації є обмеженим мітками **μ*s** := "n0ч@m0к"; **μ*e** := "KiHeu,6", виокремлюємо його з видобутого квазіповідомлення, використовуючи модуль (М.12).

Вектор ASCII-кодів **Mf***, попередньо перетворений за допомогою функції **vec2str**(...) на відповідний йому рядок символів, послідовно проходиться у пошуку стартової та кінцевої міток. Така операція виконується шляхом порівняння виокремленої частини рядку даних з відповідними мітками, які повинні бути відомими одержувачеві.

Виокремлення проводиться за допомогою функції **substr**(**Mf***, **μ**, **β**), яка повертає підрядок довжиною у **β** символів з рядка **Mf*** починаючи з символу **μ** (треба зауважити, що в даному випадку 1-й символ рядку має номер, який відповідає значенню вбудованої змінної **ORIGIN**) [7]. У нашому випадку, оскільки кожна з міток складається з 7 символів, **β** = 7. Послідовним збільшенням **μ**, відбувається просування вздовж рядка даних **Mf***. При виконанні зазначених у модулі умов (оператор **if**), коефіцієнтам **s** і **e** присвоюються відповідні значення номерів початкової і кінцевої позицій корисної інформації в рядку даних **Mf***. Такі додаткові умови, як **s** = 0; **e** = 0; **s** ≠ 0; **e** ≠ 0 введені для пришвидшення пошуку.

Зворотне перетворення рядку символів **Mf*** на вектор їх ASCII-кодів дозволяє в цьому ж модулі провести безпосереднє виокремлення прихованої інформації та відновити елементи, значення яких були примусово змінені на дробові.

---

[7] У меню *Tools / Worksheet Options* на вкладці *Calculation* слід помітити *Use ORIGIN for string indexing* (дана можливість є відсутньою у попередніх версіях MathCAD, в яких 1-й символ рядку завжди

мав індекс 0. Для коректної роботи модуля (М.12) у цих версіях необхідно виконати зазначені зміни μ на $\mu - 1$ та s на $s - 1$).

```
M_cod* := | s ← 0
          | e ← 0
          | β_s ← strlen(μ*_s)
          | β_e ← strlen(μ*_e)
          | Mf* ← vec2str(Mf*)
          | for μ ∈ 1 .. strlen(Mf*)
          |     | s ← μ + β_s   if substr(Mf*, μ, β_s) = μ_s ∧ s = 0
          |     | e ← μ − 1     if substr(Mf*, μ, β_e) = μ_e ∧ e = 0
          |     | break if s ≠ 0 ∧ e ≠ 0
          | Mf* ← substr(Mf*, s, e − β_s)
          | M_cod* ← str2vec(Mf*)
          | for n ∈ 1 .. cols(N)
          |     for i ∈ 1 .. rows(N)
          |         | break if N_{i,n} = 0
          |         | M_cod*_{N_{i,n} − β_s} ← n − 1   if 0 < N_{i,n} ≤ rows(M_cod*) + β_s
          | M_cod*
```

(М.12)

14) Розпаковане повідомлення необхідно *декодувати*. Неодмінні початкові умови при цьому – авторизованій стороні є відомими:

– алфавіт джерела повідомлення (**i** := 1..256; **A***$_i$:= **i** – 1) об’ємом **Na*** := **rows**(**A***), **Na*** = 256 символів;

– секретний ключ **K*** := "@J|eKc-I98O" з **Nk*** := **strlen**(**K***), **Nk*** = 11 символів;

– об’єм повідомлення, що підлягає декодуванню: **Nm*** := **rows**(**M_cod***), **Nm*** = 5390 симв.

За допомогою модуля (М.13) секретний ключ **K*** розширюється на довжину **Nm*** повідомлення **M_cod*** (аналогічно тому, як це робилося при кодуванні повідомлення – М.1).

Декодування секретного повідомлення проводиться за допомогою модуля (М.14).

```
K** := | K* ← str2vec(K*)
       | for i ∈ 1 .. Nm*
       |     | r ← mod(i, Nk*)
       |     | K**_i ← K*_r     if r > 0
       |     | K**_i ← K*_Nk*   if r = 0
       | K**
```

(М.13)

```
M* := | for j ∈ 1 .. Nm*
      |     | for i ∈ 1 .. Na*
      |     |     | m ← i   if M_cod*_j = A*_i
      |     |     | n ← i   if K**_j = A*_i
      |     | r ← mod(Na* + m − n, Na*)
      |     | M*_j ← A*_r     if r > 0
      |     | M*_j ← A*_Na*   if r = 0
      | M*
```

(М.14)

Декодоване повідомлення записуємо у файл:

WRITEBIN("M_dec.txt","byte",0) := **M***.

15) Проведемо обчислення показників візуального спотворення, наведених у розділі 3 (формули (3.1)–(3.17)). Отримані результати зведемо до табл.5.1.

5.3.2.2. Метод псевдовипадкового інтервалу

У вищерозглянутому найпростішому випадку проводиться заміна НЗБ усіх послідовно розташованих пікселів зображення. Інший підхід – *метод випадкового інтервалу* [78], полягає у випадковому розподілі бітів секретного повідомлення по контейнеру, у результаті чого відстань між двома вбудованими бітами визначається псевдовипадково. Ця методика особливо ефективна у випадку, коли бітова довжина секретного повідомлення є істотно меншою за кількість пікселів зображення. Розглянемо найпростіший випадок даного методу, коли інтервал між двома

послідовними вбудовуваннями бітів повідомлення є функцією координат попередньо модифікованого пікселя.

1) Нехай повідомлення, яке необхідно приховати: **M** := "© Пузиренко О.Ю., 2005 р.". У якості контейнера **C** використаємо підмасив **B** синьої колірної компоненти зображення рис.5.3.

2) Внесемо мітки, які визначатимуть границі корисного повідомлення у контейнері. На відміну від попереднього метода, стартова мітка визначатиме порядковий номер елементу контейнера, починаючи з якого в останній заноситимуться дані. Нехай $\mu_s := 154$. Мітка μ_e сигналізуватиме про завершення корисної частини серед видобутих символів, μ_e := "KiHeu,6".

3) Приймемо, що для внесення бітів повідомлення до контейнеру із змінним кроком, величина останнього обумовлюється кількістю одиниць у двійковому значенні номеру елементу контейнера, який модифікувався попередньо. Для підрахунку величини кроку (інтервалу) скористаємося модулем (М.16), який підсумовує кількість символів матриці-стовпця **x**, значення яких дорівнює "1". Коефіцієнт **K** у даному випадку виступає у ролі найпростішого ключа, який може приймати будь-які цілі значення (у тому числі й від'ємні, але у цьому випадку стартова мітка повинна мати значення, близьке до найбільшого значення індексу елементів контейнера). Також при обранні **K** слід брати до уваги загальну кількість біт, що необхідна для утаєння повідомлення, а також наявну кількість елементів масиву контейнера. Нехай **K** := 9.

$$\mathbf{step}(x) := K \cdot \sum_{i=1}^{\mathbf{rows}(x)} x_i \qquad \text{(М.16)}$$

Обмежуючу мітку μ_e за допомогою рядкової функції **concat**(...), яка об'єднує рядки, що є її аргументами, додамо до тексту повідомлення, яке підлягає приховуванню. Результат об'єднання перетворимо на вектор ASCII-кодів: **Me** := **str2vec**(**concat**(**M**, μ_e)).

Загальна кількість символів в одержаному повідомленні: **rows**(**Me**) = 32 симв. Кількість НЗБ контейнера, яка для цього потрібна (8 біт/символ): 8·**rows**(**Me**) = 256 біт.

4) Розгорнемо масив **B** у вектор (М.17), на основі якого формуємо новий вектор, який міститиме приховане повідомлення (М.18). Кожен символ повідомлення **Me** (операція циклу **for** $\mu \in 1..$**rows**(**Me**)) переводиться у двійковий формат (змінна **b**), кожен розряд якого записується замість наймолодшого біту числа **P**, яке відповідає значенню інтенсивності синього кольору певного пікселя. При цьому елементи масиву **Cv** проходяться не послідовно, а із змінним кроком, величина якого обумовлюється функцією **step**(...) [8]. Стартовий елемент задається міткою μ_s. Після проведеної зміни модифіковане двійкове число **P** переводиться у формат десяткового і записується у відповідну позицію вектора **Sv**, який на початку модуля був прийнятий рівним вектору **Cv**.

```
Cv := | Cv ← B^⟨1⟩
      | for i ∈ 2.. cols(B)
      |   Cv ← stack(Cv, B^⟨i⟩)
```
(М.17)

```
Sv := | Sv ← Cv
      | z ← μ_s
      | for μ ∈ 1.. rows(Me)
      |   | b ← D2B(Me_μ)
      |   | for i ∈ 1.. 8
      |   |   | z ← z + step(D2B(z))
      |   |   | P ← D2B(Cv_z)
      |   |   | P_1 ← b_i
      |   |   | Sv_z ← B2D(P)
      | Sv
```
(М.18)

5) Зворотне згортання вектора **Sv** до масиву, який має розмірність контейнера, здійснюється з використанням модуля (М.7) з тією лише відмінністю, що аргументом функцій розмірності масиву (**rows** і **cols**) є масив **B**. Щоб оцінити ступінь "розсіяння" бітів приховуваного повідомлення по масиву контейнера в якості прикладу наведемо результат присвоєння пікселю, до якого планувалося ввести біт повідомлення, нульового значення інтенсивності (чорний колір) при попередньому загальному висвітленні зображення (рис.5.10).

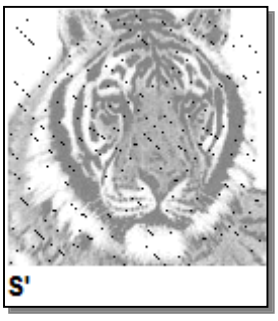


Рис.5.10. Розсіяння бітів повідомлення по масиву контейнера

6) Результуюче кольорове зображення визначатиметься масивом об'єднання колірних масивів: **S** := **augment**(**R**, **G**, **S'**).

[8] При обчисленні функції **D2B(z)** необхідно у (М.4) граничне значення змінної циклу **i** змінити з 8 на таке, що дозволить переводити у двійковий формат найстарший індекс елементів вектору **Cv**.

7) При видобуванні прихованого повідомлення повинні бути відомими параметри **μ*s, μ*e, K*** і, зрештою, масив **B***, який повинен містити приховані дані.

Розгортання масиву **B*** у вектор **Sv*** відбувається за аналогічним до (М.17) модулем. Видобування повідомлення з вектору **Sv*** проводиться за допомогою модуля (М.19) у порядку, зворотному операції вбудовування.

З одержаного вектора **Mf*** шляхом порівняння з міткою **μ*e** виділеного фрагменту видобувається корисне повідомлення **M*** (М.20).

```
Mf* := | z ← μ*_s
       | for μ ∈ 1 .. rows(Sv*)
       |     for i ∈ 1 .. 8
       |         z ← z + step(D2B(z))
       |         break if z > rows(Sv*)
       |         P ← D2B(Sv*_z)
       |         b_i ← P_1
       |     Mf*_μ ← B2D(b)
       |     μ ← μ + 1
       | Mf*
```
(М.19)

```
M* := | μ*_e ← str2vec(μ*_e)
      | β_e ← rows(μ*_e)
      | for μ ∈ 1 .. rows(Mf*) − β_e
      |     M* ← submatrix(Mf*, 1, μ − 1, 1, 1) if submatrix(Mf*, μ, μ + β_e − 1, 1, 1) = μ*_e
      | vec2str(M*)
```
(М.20)

M* = "© Пузиренко О.Ю., 2005 р.".

Результати обчислення візуального спотворення зведені до табл.5.1.

5.3.2.3. Метод псевдовипадкової перестановки

Недоліком методу псевдовипадкового інтервалу є те, що біти повідомлення у контейнері розміщені у тій самій послідовності, що й в самому повідомлені, і лише інтервал між ними змінюється псевдовипадково. Тому для контейнерів фіксованого розміру більш оптимальним є використання методу *псевдовипадкової перестановки (обрання)* [79]. Його сутність полягає у тому, що генератор ПВЧ створює послідовність індексів $j_1, j_2, ..., j_{l_M}$ і зберігає k-й біт повідомлення в пікселі з індексом j_k. Нехай N – загальна кількість біт (наймолодших) у наявному контейнері; P^N – перестановка чисел $\{1, 2, ..., N\}$. Тоді, якщо ми маємо для приховання конфіденційне повідомлення довжиною n біт, можна просто вбудувати ці біти замість бітів контейнера $P^N(1), P^N(2), ..., P^N(n)$. Функція перестановки повинна бути псевдовипадковою, тобто вона повинна забезпечувати обрання бітів контейнера приблизно випадковим чином. Отже, секретні біти будуть рівномірно поширені на всьому бітовому просторі контейнера. Але, у цьому випадку індекс певного біту контейнера може з'явитися в послідовності більше одного разу, іншими словами, може відбутися "перетинання" – спотворення вже вбудованого біта. Якщо кількість бітів повідомлення набагато менша за розмір зображення, то імовірність перетинання є незначною, і спотворені біти можуть бути відновлені за допомогою коригувальних кодів. Імовірність, принаймні, одного перетинання оцінюється як:

$$p \approx 1 - \exp\left[-\frac{l_M \cdot (l_M - 1)}{2 \cdot l_C}\right], \quad l_M << l_C . \tag{5.1}$$

При збільшенні l_M і $l_C = const$ дана імовірність прямує до одиниці.

Для запобігання перетинань можна запам'ятовувати всі індекси використаних елементів j_i і перед модифікацією нового пікселя проводити перевірку його на повторюваність. Також

можна застосовувати генератори ПВЧ без повторюваності чисел. Останній випадок розглянемо більш докладно.

Для наших цілей функція перестановки також залежить від секретного ключа $\boldsymbol{K}$. При цьому генератор псевдовипадкової перестановки $\boldsymbol{P}^{\boldsymbol{N}}$ є функцією, яка для кожного значення $\boldsymbol{K}$ виробляє різні псевдовипадкові перестановки чисел $\{1, 2, ..., \boldsymbol{N}\}$. Позначимо через $\boldsymbol{P}_{\boldsymbol{K}}^{\boldsymbol{N}}$ генератор перестановок з відповідним ключем $\boldsymbol{K}$. Якщо перестановка $\boldsymbol{P}_{\boldsymbol{K}}^{\boldsymbol{N}}$ є захищеною по обчисленню (тобто розкриття вимагає невиправдано великих витрат обчислювальних ресурсів порушника), то можливість розкриття змісту або припущення самого лише виду перестановок без володіння інформацією про секретний ключ $\boldsymbol{K}$ практично дорівнює нулеві.

Секретний генератор псевдовипадкової перестановки (ГПВП) може бути ефективно реалізований на основі генератора псевдовипадкової функції (ГПВФ) [98], який, як і ГПВП, виробляє різні функції, що не піддаються прогнозуванню, при кожному окремому значенню ключа; але множина значень функції не повинна дорівнювати області її визначення. ГПВФ легко реалізується з секретної хеш-функції $\boldsymbol{H}$ шляхом об'єднання аргументу $\boldsymbol{i}$ з секретним ключем $\boldsymbol{K}$ та взяття від результуючого бітового рядка функції $\boldsymbol{H}$:

$$\boldsymbol{f}_{\boldsymbol{K}}(\boldsymbol{i}) = \boldsymbol{H}(\boldsymbol{K} \circ \boldsymbol{i}), \tag{5.2}$$

де $\boldsymbol{K} \circ \boldsymbol{i}$ – об'єднання (конкатенація) бітових рядків $\boldsymbol{K}$ та $\boldsymbol{i}$; $\boldsymbol{f}_{\boldsymbol{K}}(\boldsymbol{i})$ – результуюча псевдовипадкова функція від $\boldsymbol{i}$, яка залежить від параметру $\boldsymbol{K}$.

Генератор *Luby* та *Reckoff*'а побудований наступним чином. Запис виду $\boldsymbol{a} \oplus \boldsymbol{b}$ розуміє під собою побітове додавання за модулем 2 аргументу $\boldsymbol{a}$ до аргументу $\boldsymbol{b}$, причому результат додавання має ту саму розмірність, що й $\boldsymbol{a}$. Нехай $\boldsymbol{i}$ – рядок двійкових даних довжиною $2 \cdot \boldsymbol{l}$. Розділимо $\boldsymbol{i}$ на дві частини – $\boldsymbol{x}$ та $\boldsymbol{y}$ довжиною $\boldsymbol{l}$ кожна, а ключ $\boldsymbol{K}$ – на чотири частини: $\boldsymbol{K}_1$, $\boldsymbol{K}_2$, $\boldsymbol{K}_3$, $\boldsymbol{K}_4$. Тоді

$$\boldsymbol{y} = \boldsymbol{y} \oplus \boldsymbol{f}_{\boldsymbol{K}_1}(\boldsymbol{x}) = \boldsymbol{y} \oplus \boldsymbol{H}(\boldsymbol{K}_1 \circ \boldsymbol{x});$$

$$\boldsymbol{x} = \boldsymbol{x} \oplus \boldsymbol{f}_{\boldsymbol{K}_2}(\boldsymbol{y}) = \boldsymbol{x} \oplus \boldsymbol{H}(\boldsymbol{K}_2 \circ \boldsymbol{y});$$

$$\boldsymbol{y} = \boldsymbol{y} \oplus \boldsymbol{f}_{\boldsymbol{K}_3}(\boldsymbol{x}) = \boldsymbol{y} \oplus \boldsymbol{H}(\boldsymbol{K}_3 \circ \boldsymbol{x});$$

$$\boldsymbol{x} = \boldsymbol{x} \oplus \boldsymbol{f}_{\boldsymbol{K}_4}(\boldsymbol{y}) = \boldsymbol{x} \oplus \boldsymbol{H}(\boldsymbol{K}_4 \circ \boldsymbol{y});$$

повернення $\boldsymbol{y} \circ \boldsymbol{x}$.

Для кожного значення ключа $\boldsymbol{K}$ алгоритм повертає псевдовипадкову перестановку з чисел $\{1, ..., 2^{2 \cdot \boldsymbol{l}}\}$. *Luby* та *Reckoff* показали, що перестановка є настільки ж секретною, наскільки й генератор ПВФ. Вони також навели простий алгоритм перестановки з $\{1, ..., 2^{2 \cdot \boldsymbol{l}+1}\}$. Якщо значення функції $\boldsymbol{f}_{\boldsymbol{K}}$ являють собою достатньо довгі бітові послідовності, той самий ефект можна отримати, прийнявши, що $\boldsymbol{y}$ – перші $\boldsymbol{l}$ біт рядка $\boldsymbol{i}$, а $\boldsymbol{x}$ – останні $\boldsymbol{l} + 1$ біт.

Вищенаведена конструкція дозволяє отримати перестановку $\boldsymbol{P}_{\boldsymbol{K}}^{2^{\boldsymbol{k}}}$ з $\{1, ..., 2^{\boldsymbol{k}}\}$ для довільного $\boldsymbol{k}$. Проте, коли кількість біт контейнера становить $\boldsymbol{N}$ існує потреба перестановки $\boldsymbol{P}_{\boldsymbol{K}}^{\boldsymbol{N}}$ з $\{1, ..., \boldsymbol{N}\}$. Перевагою методу, запропонованого у [79] є те, що існує можливість обмежитися лише наявними для $\boldsymbol{P}_{\boldsymbol{K}}^{\boldsymbol{N}}$ аргументами. Нехай $\boldsymbol{k} = \lceil \log_2(\boldsymbol{N}) \rceil$ (квадратні дужки означають округлення до найменшого цілого, що більше або дорівнює аргументу). Тоді $2^{\boldsymbol{k}-1} < \boldsymbol{N} \le 2^{\boldsymbol{k}}$. При цьому підраховуються значення $\boldsymbol{P}_{\boldsymbol{K}}^{2^{\boldsymbol{k}}}(1), \boldsymbol{P}_{\boldsymbol{K}}^{2^{\boldsymbol{k}}}(2), ...$ і видаляються з послідовності будь-які числа, що перевищують $\boldsymbol{N}$ й одержують таким чином значення $\boldsymbol{P}_{\boldsymbol{K}}^{\boldsymbol{N}}(1), \boldsymbol{P}_{\boldsymbol{K}}^{\boldsymbol{N}}(2), ...$. Це є можливим, коли функція перестановки обчислена для зростаючих значень аргументів починаючи з одиниці. Таким чином, генератор ПВП $\boldsymbol{P}^{\boldsymbol{N}}$ для довільного $\boldsymbol{N}$ може бути побудований на основі алгоритму *Luby* та *Reckoff*'а.

Але, коли $\boldsymbol{N}$ є складеним (як у випадку зображення), існує більш зручний спосіб побудови ГПВП. Наведений нижче алгоритм оснований на блочному кодуванні з довільним розміром блоку [79,99]. Кількість біт контейнера повинна становити собою складене число з двох співмножників приблизно однакового порядку, тобто $\boldsymbol{N} = \boldsymbol{X} \cdot \boldsymbol{Y}$ для деяких $\boldsymbol{X}$ та $\boldsymbol{Y}$. У випадку, коли

дані приховуються у НЗБ пікселів цифрового зображення, параметри X та Y є розмірами даного зображення. Для одержання координат i-го пікселя зображення для приховання біту повідомлення ($i \in \{1, ..., N\}$) необхідно виконати наступні обчислення:

$$x = \mathbf{div}(i, Y) + 1; \tag{5.3а}$$

$$y = \mathbf{mod}(i, Y) + 1; \tag{5.3б}$$

$$x = \mathbf{mod}(x + f_{K_1}(y), X) + 1; \tag{5.3в}$$

$$y = \mathbf{mod}(y + f_{K_2}(x), Y) + 1; \tag{5.3г}$$

$$x = \mathbf{mod}(x + f_{K_3}(y), X) + 1; \tag{5.3д}$$

$$i = (x, y) \text{ або } i = (x-1)\cdot Y + y, \tag{5.3е}$$

де $\mathbf{div}(i, X)$ і $\mathbf{mod}(i, X)$ – функції, що повертають, відповідно, ціле і залишок від ділення i на X. Другий варіант формули (5.3е) застосовний у випадку, якщо масив зображення попередньо було розгорнуто у вектор (по рядкам). Додавання одиниці необхідне при індексації елементів масиву зображення з 1.

Перші два раунди алгоритму ((5.3а) – (5.3г)) необхідні для того, щоб "розсіяти" біти приховуваного повідомлення серед найменш значущих бітів контейнера. При цьому 1-й раунд надає випадкового характеру x-координатам пікселя-контейнера, а 2-й – y-координатам. 3-й раунд необхідний для запобігання атаці на відкритий (незашифрований) текст. У випадку лише двох раундів нехай $i = (b-1)\cdot Y + a$, а $P_K^N(i)$ – значення перестановки. Якщо криптоаналітик здатний припустити значення a і може одержати пару "відкритий текст – кодований текст" ($i' = (z-1)\cdot Y + a$, $P_K^N(i')$) для деякого z, то він здатний встановити b. Навіть при тому, що автори [79] вважають, що пропонований ними алгоритм буде достатньо стійким з трьома раундами, вони визнають, що у деяких випадках може знадобитися 4 або більше раундів, що ще більше підвищить стійкість алгоритму до зламу.

Промоделюємо даний метод у програмі MathCAD.

1) Повідомлення, яке треба приховати: **M** := "© Пузиренко О.Ю., 2005 р.". Контейнер **C** – підмасив **B** синьої колірної компоненти зображення рис.5.3. При цьому кількість біт у повідомленні: L_M := **strlen**(**M**), L_M = 200 біт; геометричні розміри контейнера: **X** := **rows**(**C**), **X** = 128 пікс.; **Y** := **cols**(**C**), **Y** = 128 пікс.; **N** := **X**·**Y**, **N** = 16384.

2) Для формування ключа використаємо модуль (М.21), який дозволяє на основі первинного ключа $K_o \geq 2$ сформувати вектор, що міститиме $\Re$ пар ключів (кожна пара ключів використовуватиметься у відповідному раунді обчислення координат **x** та **y**).

```
K := | for s ∈ 1 .. 2·ℜ
     |   K_s ← K_o                                          if s = 1
     |   K_s ← str2num(substr(num2str((K_{s-1})^2), 0, 3))  if s > 1
     |   K_s ← str2num(substr(num2str(K_s), 0, 2))          if K_s > 255
     | K
```

(М.21)

У даному модулі почергово використовується три функції: **num2str**(**d**) – перетворення аргументу-числа **d** на відповідний рядок **A**; **substr**(**A**, 0, 3) – виділення фрагменту рядка **A**, що містить 3 перші символи рядка; **str2num**(**a**) – зворотне перетворення **a**-фрагменту на число, яке й присвоюється **s**-му елементові масиву **K**. У випадку K_S > 255 за допомогою аналогічної комбінації функцій число скорочується на один символ.

Наведемо приклад обчислення пар ключів при K_o = 125 і $\Re$ = 6 (рис.5.12).

$K^T =$

1	2	3	4	5	6	7	8	9	10	11	12
125	156	243	59	34	115	132	174	30	90	81	65

Рис.5.12. Приклад обчислення ключів **K**

3) Вбудовування бітів повідомлення до псевдовипадкових пікселів контейнера виконаємо за модулем (М.22), який реалізує алгоритм (5.3). На початку модуля масиву **S** прирівнюється вихідний масив **C**. Також проводиться конвертування повідомлення з рядкового формату на вектор двійкових даних $\mathbf{M}_{\mathbf{vec\_bin}}$. При обчисленні координат **x** та **y** використовується операція векторизації, яка дозволяє поелементно додавати по модулю 2 двійкові вектори **K** та **y** (або **x**). При цьому розмірність зазначених векторів повинна бути однаковою, для чого використовується функція **submatrix**(...).

```
S := | S ← C
     | M_vec ← str2vec(M)
     | M_vec_bin ← D2B(M_vec_1)
     | for j ∈ 2.. rows(M_vec)
     |   M_vec_bin ← stack(M_vec_bin, D2B(M_vec_j))
     | for i ∈ 1.. L_M
     |   x ← floor(i / Y) + 1
     |   y ← mod(i, Y) + 1
     |   for s ∈ 1.. ℜ
     |     x ← mod(x + B2D((D2B(K_{2·s−1}) ⊕ submatrix(D2B(y), 1, 8, 1, 1)), X) + 1
     |     y ← mod(y + B2D((D2B(K_{2·s}) ⊕ submatrix(D2B(x), 1, 8, 1, 1)), Y) + 1
     |   P ← D2B(C_{x,y})
     |   P_1 ← M_vec_bin_i
     |   S_{x,y} ← B2D(P)
     | S
```

(М.22)

Оцінимо ступінь “розсіяння” бітів повідомлення по масиву контейнера при різній кількості раундів $\Re$ обчислення координат **x** та **y** (рис.5.11). Формування рисунку проводиться аналогічно до формування рис.5.10. Як видно, прийнятний рівень розсіяння досягається вже при $\Re = 4$.

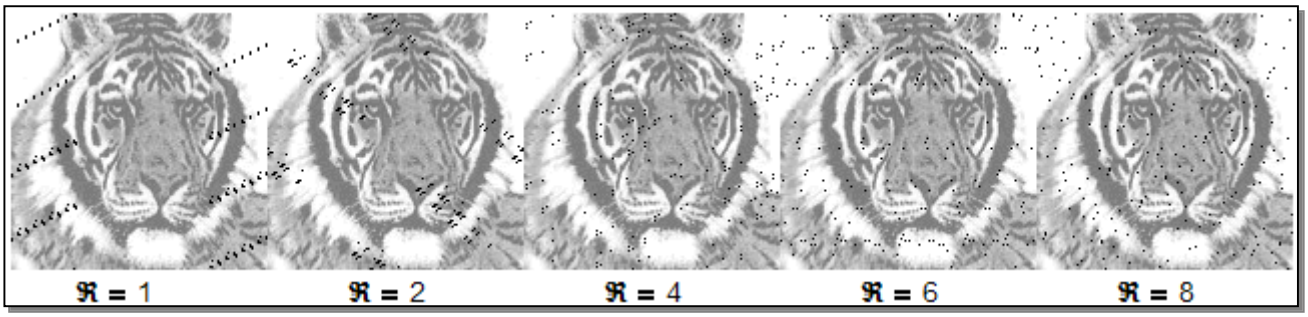


Рис.5.11. Розсіяння бітів повідомлення по масиву контейнера при зміні параметру $\Re$

4) На приймальній стороні повинні бути відомими первинний ключ $\mathbf{K_o}$*, масив колірності, до якого проводилося вбудовування (**S***). З останнього одержуються значення **X***, **Y***, **N***. Модуль, призначений для видобування прихованого повідомлення, наведено нижче. Зазначимо, що у даному методі і в подальшому при вбудовуванні повідомлення не проводилося попереднє додавання стартової і кінцевої міток до останнього. Відповідно, при видобуванні рядок символів крім корисного змісту міститиме ще й довільний набір символів, що входять до кодування ASCII. Відсутність етапів виділення корисного повідомлення з усієї множини символів у наступних

модулях жодним чином не говорить про неможливість здійснення зазначеного. За бажанням, наведені модулі можна легко адаптувати для можливості аналізу попередньо вбудованих міток (за аналогією з вищерозглянутими методами).

```
M* := | for i ∈ 1 .. N*/8
      |   x* ← floor(i / Y*) + 1
      |   y* ← mod(i , Y*) + 1
      |   for s ∈ 1 .. ℜ
      |     x* ← mod(x* + B2D((D2B(K*_(2·s−1)) ⊕ submatrix(D2B(y*) , 1 , 8 , 1 , 1))⃗ , X*) + 1
      |     y* ← mod(y* + B2D((D2B(K*_(2·s)) ⊕ submatrix(D2B(x*) , 1 , 8 , 1 , 1))⃗ , Y*) + 1
      |   P* ← D2B(S*_(x*, y*))
      |   M* vec_bin_i ← P*_1
      | for j ∈ 1 .. rows(M* vec_bin)/8
      |   M* vec_j ← B2D(submatrix(M* vec_bin , 8·j − 7 , 8·j , 1 , 1))
      | vec2str(M* vec)
```

(M.23)

M* = "© Пузиренко О.Ю., 2005 р.□G'P†e2□*UM‹jSьZЧжYФ□...

Результати обчислення візуального спотворення зведені до табл.5.1.

5.3.2.4. Метод блокового приховування

Метод блокованого приховування є ще одним підходом до реалізації методу заміни і полягає в наступному [3]. Зображення-оригінал розбивається на l_M неперетинних блоків, Δ_i ($1 \le i \le l_M$) довільної конфігурації, для кожного з яких обчислюється біт парності $b(\Delta_i)$:

$$b(\Delta_i) = \sum_{j \in \Delta_i}^{\mathrm{mod}\,2} LSB(C_j). \qquad (5.4)$$

У кожному блоці проводиться приховування одного секретного біту M_i. Якщо біт парності $b(\Delta_i) \neq M_i$, то відбувається інвертування одного з НЗБ блоку Δ_i, у результаті чого $b(\Delta_i) = M_i$. Обрання блоку може здійснюватись псевдовипадково з використанням стеганоключа. Хоча цей метод має таку ж стійкість до спотворень, як і всі попередні, він має низку переваг. По-перше, існує можливість модифікувати значення такого пікселя в блоці, зміна якого призведе до мінімальної зміни статистики контейнера. По-друге, вплив наслідків вбудовування секретних даних у контейнер можна зменшити за рахунок збільшення розміру блоку.

Наведемо приклад програми у MathCAD, яка дозволяє здійснити стеганографічний захист текстового повідомлення методом блокового приховування.

1) Початкові вихідні дані відповідають прийнятим при моделюванні попереднього методу.

2) Поділ масиву контейнера на блоку здійснюватимемо наступним чином: якщо кількість біт у повідомленні (L_M) не перевищує кількості стовпців масиву **C** (**Y**), то один блок відповідає окремому стовпцю масиву **C**. Якщо $L_M > Y$, то один блок дорівнює $1/\chi$ від окремого стовпця масиву, де $\chi = \mathrm{ceil}(L_M \div Y)$. Значення χ повинне бути відомим одержувачеві.

Даний алгоритм вбудовування реалізований у модулі (М.24). Початок модуля є аналогічним модулю (М.22). Лічильник σ дозволяє виокремлювати відповідну співвідношенню χ

частину від загальної довжини стовпця масиву – визначаються індекси рядків, починаючи з якого (**r1**) і по який (**r2**) виділяється фрагмент **Δ** **y**-го стовпця. Для кожного блоку **Δ** проводиться обчислення біту парності **b**. Якщо **b** не дорівнює поточному значенню біту повідомлення, з блоку **Δ** випадковим чином обирається індекс **n** пікселя, інтенсивність кольору якого збільшується або зменшується на одиницю, в залежності від того, парним чи непарним є первинне його значення. За допомогою функції **putregion**(**S**, **Δ**, **r1**, **y**) здійснюється вбудовування модифікованого масиву **Δ** до загального масиву **S**, починаючи з рядка **r1** і стовпця **y** в бік найстарших індексів рядків і стовпців відповідно.

На рис.5.12 зображено пікселі контейнера, інтенсивність кольору яких зазнала змін (для наочності встановлено чорний колір даних пікселів). Очевидним є зниження кількості модифікованих пікселів порівняно з результатом застосування двох попередніх методів (див. рис.5.10 і 5.11).

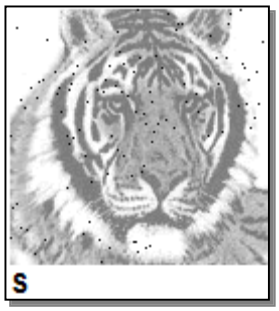


Рис.5.12. Розсіяння бітів повідомлення по масиву контейнера

```
S := | S ← C
     | M_vec ← str2vec(M)
     | M_vec_bin ← D2B(M_vec_1)
     | for j ∈ 2 .. rows(M_vec)
     |   M_vec_bin ← stack(M_vec_bin, D2B(M_vec_j))
     | for σ ∈ 1 .. χ
     |   r1 ← (σ − 1)·floor(X ÷ χ) + 1
     |   r2 ← σ·floor(X ÷ χ)
     |   for y ∈ 1 .. Y
     |     break if y + (σ − 1)·Y > L_M
     |     Δ ← submatrix(S, r1, r2, y, y)
     |     b ← 0
     |     for x ∈ 1 .. rows(Δ)
     |       P ← D2B(Δ_x)
     |       LSB ← P_1
     |       b ← b ⊕ LSB
     |     if b ≠ M_vec_bin_{y+(σ−1)·Y}
     |       n ← ceil(rnd(rows(Δ)))
     |       Δ_n ← Δ_n + 1 if mod(Δ_n, 2) = 0
     |       Δ_n ← Δ_n − 1 otherwise
     |       S ← putregion(S, Δ, r1, y)
     | S
```

(M.24)

3) Модуль видобування:

```
M* := | for σ ∈ 1 .. χ*
      |   r1 ← (σ − 1)·floor(X* ÷ χ*) + 1
      |   r2 ← σ·floor(X* ÷ χ*)
      |   for y ∈ 1 .. Y*
      |     Δ ← submatrix(S*, r1, r2, y, y)
      |     b ← 0
      |     for x ∈ 1 .. rows(Δ)
      |       P ← D2B(Δ_x)
      |       LSB ← P_1
      |       b ← b ⊕ LSB
      |     M*_vec_bin_{y+(σ−1)·Y} ← b
      | for j ∈ 1 .. rows(M*_vec_bin) ÷ 8
      |   M*_vec_j ← B2D(submatrix(M*_vec_bin, 8·j − 7, 8·j, 1, 1))
      | vec2str(M*_vec)
```

(M.25)

M* = "© Пузиренко О.Ю., 2005 р.'v€>t□Ц"

Результати обчислення візуального спотворення зведені до табл.5.1.

5.3.2.5. Методи заміни палітри

Для приховування даних можна також скористатися палітрою кольорів, присутніх у форматі зображення [80]. Палітра з N кольорів визначається як список пар індексів (i, Λ_i), який визначає відповідність між індексом i та його вектором колірності Λ_i (так звана таблиця кольорів). Кожному пікселю зображення ставиться у відповідність певний індекс у таблиці. Оскільки порядок кольорів у палітрі не є важливим для відтворення загального зображення, конфіденційна інформація може бути прихована шляхом перестановки кольорів у палітрі. Існує $N!$ різних способів перестановки N-кольорової палітри, що є цілком достатнім для приховування невеликого повідомлення. Але методи приховування, в основі яких лежить порядок формування палітри, також є нестійкими: будь-яка атака, пов'язана зі зміною палітри, знищує вбудоване повідомлення.

Найчастіше сусідні кольори в палітрі не обов'язково є схожими, тому деякі стеганометоди перед приховуванням даних проводять упорядкування палітри таким чином, що суміжні кольори стають подібними. Наприклад, значення кольору може бути упорядковане за відстанню d у RGB-просторі, де $d = \sqrt{R^2 + G^2 + B^2}$ [3]. Оскільки ЗСЛ є більш чутливою до змін яскравості кольору, то доцільно сортувати вміст палітри саме за значеннями яскравості сигналу. Після сортування палітри можна змінювати НЗБ індексів кольору без особливого спотворення зображення.

Деякі стеганометоди [81] передбачають зменшення загальної кількості значень кольорів (до $N/2$) шляхом "розмивання" зображення. При цьому елементи палітри дублюються таким чином, щоб значення кольорів для них розрізнялося несуттєво. У підсумку кожне значення кольору розмитого зображення відповідає двом елементам палітри, які обираються у відповідності до біта приховуваного повідомлення.

Можна запропонувати наступний варіант реалізації методу заміни палітри.

1) Початкові дані стандартні.

2) Таблицю кольорів одержимо, використовуючи підмасив інтенсивності червоного **R** (М.26). Секретність таблиці визначатиме алгоритм її формування на основі масиву **R**. Упорядкуємо таблицю **T** за інтенсивністю кольору за допомогою функції **csort**(**T**, **c**), яка дозволяє переставити рядки масиву **T** таким чином, щоб відсортованим виявився стовпець **c.**

На рис.5.13 наведено фрагменти оригінальної та відсортованої колірних таблиць.

T =

	1	2
1	1	255
2	2	251
3	3	250
4	4	249
5	5	247
6	6	244
⋮	⋮	⋮
251	251	230
252	252	75
253	253	196
254	254	193
255	255	55
256	256	2

$\mathbf{T}_{sort}$ =

	1	2
1	99	0
2	49	1
3	256	2
4	98	3
5	139	4
6	174	5
⋮	⋮	⋮
251	3	250
252	2	251
253	74	252
254	108	253
255	109	254
256	1	255

```
T := i ← 1
     for x ∈ 1..X
       for y ∈ 1..Y
         break if i = 257
         Λ ← R[x,y]
         if i = 1
           T[i,1] ← i
           T[i,2] ← Λ
           i ← i + 1
         if i > 1
           δ ← 0
           for j ∈ 1..i − 1
             δ ← 1 if T[j,2] = Λ
           if δ = 0
             T[i,1] ← i
             T[i,2] ← Λ
             i ← i + 1
     T
```

(М.26)

$\mathbf{T}_{sort} := \mathbf{csort}(\mathbf{T}, 2)$

Рис.5.13. Оригінальна (**T**) та відсортована
(**T_{sort}**) колірні таблиці

На рис.5.14 наведено графічну інтерпретацію колірних таблиць.

Рис.5.14. Графічне відображення колірних таблиць **T** і **T_{sort}**

3) Модуль вбудовування повідомлення до контейнера (М.27) реалізує наступний алгоритм. Формування бітового вектора з символьного рядка аналогічне наведеному у (М.22). З масиву контейнера **C** шляхом перебирання індексів рядків (**x**) і стовпців (**y**) змінній **pix** присвоюються значення інтенсивностей кольору відповідних пікселів контейнера. Внутрішнім циклом $\mathbf{i} \in 1..256$ проводиться пошук відповідного значення інтенсивності у сортованій колірній таблиці **T_{sort}**. У випадку знаходження, змінній **n** присвоюється значення індексу, який відповідає даній інтенсивності у таблиці **T** (1-й стовпець **T_{sort}**). Змінній $\mathfrak{I}$ – значення індексу, який відповідає даній інтенсивності у таблиці **T_{sort}**. Якщо НЗБ індексу **n** не дорівнює поточному біту приховуваного повідомлення, відбувається пошук найближчого індексу, НЗБ якого дорівнює бітові повідомлення. Пошук ведеться вниз (**L**) і вгору (**H**) від індексу $\mathfrak{I}$. Попереднє присвоєння змінним λ_L та λ_H значення $-/+1000$ гарантує неможливість дублювання попередніх значень λ, якщо просування вниз або вгору від індексу $\mathfrak{I}$ не призвело до виконання

```
S := | Ⓘ -- див. (М.22)
     | μ ← 1
     | for x ∈ 1 .. X
     |   for y ∈ 1 .. Y
     |     | break if μ > L_M
     |     | pix ← C_{x,y}
     |     | for i ∈ 1 .. 256
     |     |   if T_sort_{i,2} = pix
     |     |     | n ← T_sort_{i,1}
     |     |     | ℑ ← i
     |     |     | break
     |     | if mod(n,2) ≠ M_vec_bin_μ
     |     |   | for σ ∈ 1 .. 255
     |     |   |   | λ_L ← −1000
     |     |   |   | if mod(T_sort_{ℑ−σ,1},2) = M_vec_bin_μ if ℑ − σ ≥ 1
     |     |   |   |   | λ_L ← T_sort_{ℑ−σ,2}
     |     |   |   |   | break
     |     |   | for σ ∈ 1 .. 255
     |     |   |   | λ_H ← 1000
     |     |   |   | if mod(T_sort_{ℑ+σ,1},2) = M_vec_bin_μ if ℑ + σ ≤ 256
     |     |   |   |   | λ_H ← T_sort_{ℑ+σ,2}
     |     |   |   |   | break
     |     |   | S_{x,y} ← λ_L if pix − λ_L ≤ λ_H − pix
     |     |   | S_{x,y} ← λ_H if pix − λ_L > λ_H − pix
     |     | μ ← μ + 1
     | S
```
(М.27)

поставленої умови (останнє є можливим при знаходженні індексу $\Im$ надто близько до нижньої або верхньої межі відсортованої колірної таблиці). Після знаходження значень λ_L і λ_H пікселю контейнера **S** присвоюється те з них, яке по колірній вісі знаходиться найближче до інтенсивності відповідного пікселя контейнера **C** (**pix**). Після вбудовування останнього біта повідомлення зовнішній цикл переривається – контейнер заповнений.

4) При видобуванні повідомлення на основі масиву **R*** необхідно сформувати таблиці кольорів **T*** і $\mathbf{T^*_{sort}}$. Модулі, що реалізують дану операцію, ідентичні (М.26).

5) Модуль видобування (М.28) для інтенсивності кожного пікселя масиву **S*** проводить пошук відповідної інтенсивності у колірній таблиці. При знаходженні, μ-му елементу бітового повідомлення **M*** присвоюється значення НЗБ індексу, що відповідає даній інтенсивності у

```
M* := | μ ← 1
      | for x ∈ 1 .. X*
      |   for y ∈ 1 .. Y*
      |     | pix ← S*_{x,y}
      |     | for i ∈ 1 .. 256
      |     |   if T*_sort_{i,2} = pix
      |     |     | M*_μ ← mod(T*_sort_{i,1},2)
      |     |     | μ ← μ + 1
      | for j ∈ 1 .. rows(M*) ÷ 8
      |   M*_vec_j ← B2D(submatrix(M*, 8·j − 7, 8·j, 1, 1))
      | vec2str(M*_vec)
```
(М.28)

невідсортованій таблиці. Одержаний бітовий вектор в кінці модуля переводиться на рядок символів.

Результати обчислення візуального спотворення зведені до табл.5.1.

5.3.2.6. Метод квантування зображення

До методів приховання в просторовій області можна також віднести *метод квантування зображення* [3,82], заснований на міжпіксельній залежності, яку можна описати деякою функцією **Θ**. У найпростішому випадку можна обчислити різницю ε_i між суміжними пікселями c_i та c_{i+1} (або c_{i-1} та c_i) і задати її як параметр функції Θ: $\Delta_i = \Theta(c_i - c_{i+1})$, де Δ_i – дискретна апроксимація різниці сигналів $c_i - c_{i+1}$. Оскільки Δ_i є цілим числом, а реальна різниця $c_i - c_{i+1}$ – дійсним, то виникають помилки квантування $\delta_i = \Delta_i - \varepsilon_i$. Для сильно корельованих сигналів ця помилка близька до нуля: $\delta_i \approx 0$. У даному методі приховування інформації проводиться шляхом корегування різницевого сигналу Δ_i. Стеганоключ являє собою таблицю, яка кожному можливому значенню Δ_i ставить у відповідність визначений біт, наприклад:

Δ_i	–4	–3	–2	–1	0	1	2	3	4
b_i	1	0	1	1	0	0	1	0	1

Для приховування i-го біта повідомлення обчислюється різниця Δ_i. Якщо при цьому b_i не відповідає секретному бітові, який необхідно приховати, то значення Δ_i замінюється найближчим Δ_j, для якого така умова виконується. При цьому відповідним чином корегуються значення інтенсивностей пікселів, між якими обраховувалася різниця Δ_i. Видобування секретного повідомлення проводиться відповідно до значення b^*_i, що відповідає різниці Δ^*_i.

Наведемо приклад програми, що реалізує метод квантування зображення.

1) Вихідні дані є стандартними.

2) Стеганоключ обчислимо за модулями (М.29) і (М.30). При цьому модуль (М.29) повертає всі можливі різниці сигналів (від –255 до +255), а модуль (М.30) – значення бітів, які відповідають цим різницям.

```
Δ := | for i ∈ 1 .. 511
     |   Δ_i ← i − 256
     | Δ
```
(М.29)

```
b := | i ← 1
     | for y ∈ 1 .. Y
     |   c ← R^⟨y⟩
     |   b_i ← 0
     |   for x ∈ 1 .. X
     |     b_i ← b_i ⊕ mod(c_x, 2)
     |     b_i ← b_i ∨ 1  if mod(x, 3) = 0
     |   i ← i + 1
     | b ← stack(b, b, b, b)
```
(М.30)

Значення b_i у даному разі обчислюються на основі масиву червоної колірної складаової. При цьому для кожного стовпця масиву **R** обчислюється сума по модулю 2 елементів стовпця з булевим додаванням до результату одиниці при кожному третьому елементі. Наприкінці модуля одержаний вектор **b** розширюється на довжину вектора **Δ**. Таким чином елементи масиву **b** носять псевдовипадковий характер. Фрагменти сформованого стеганоключа наведено на рис.5.15.

3) Проведемо розгортання масиву контейнера **C** (масив синьої колірної складової) у вектор, використовуючи модуль (М.17). Задаємося стартовим індексом елемента отриманого вектора, починаючи з якого проводитиметься вбудовування бітів повідомлення (наприклад, $\mu_S := 105$).

Для підрахунку величини кроку (псевдовипадкового інтервалу) використаємо модуль (М.16). Нехай при цьому **K** := 8.

4) Алгоритм вбудовування реалізує модуль (М.31). Формування вектора двійкових даних з рядка символів аналогічне наведеному у (М.22) (при цьому **S** ← **C** слід замінити на **Sv** ← **Cv**).

Δ=

	1
1	-255
2	-254
3	-253
4	-252
⋮	⋮
254	-2
255	-1
256	0
257	1
258	2
⋮	⋮
508	252
509	253
510	254
511	255

b=

	1
1	1
2	0
3	1
4	0
⋮	⋮
254	0
255	1
256	0
257	1
258	0
⋮	⋮
508	0
509	0
510	0
511	1

Рис.5.15. Фрагменти стеганоключа

```
Sv:= | Ⓘ -- див. (М.22)
     | z ← μ_S
     | for μ ∈ 1.. L_M
     |   z ← z + step(D2B(z))
     |   Δ' ← Cv_z − Cv_(z−1)
     |   for i ∈ 1.. rows(Δ)
     |     if Δ' = Δ_i
     |       ℑ ← i
     |       break
     |   if b_ℑ ≠ M_(vec_bin_μ)
     |     for σ ∈ 1.. rows(Δ) − 1
     |       ℑ_L ← −1000
     |       if b_(ℑ−σ) = M_(vec_bin_μ)   if ℑ − σ ≥ 1
     |         ℑ_L ← ℑ − σ
     |         break
     |     for σ ∈ 1.. rows(Δ) − 1
     |       ℑ_H ← 1000
     |       if b_(ℑ+σ) = M_(vec_bin_μ)   if ℑ + σ ≤ rows(Δ)
     |         ℑ_H ← ℑ + σ
     |         break
     |     if ℑ − ℑ_L < ℑ_H − ℑ
     |       Sv_z ← Sv_(z−1) + Δ_(ℑ_L)   if 0 ≤ Sv_(z−1) + Δ_(ℑ_L) ≤ 255
     |       Sv_(z−1) ← Sv_z − Δ_(ℑ_L)   otherwise
     |     if ℑ − ℑ_L ≥ ℑ_H − ℑ
     |       Sv_z ← Sv_(z−1) + Δ_(ℑ_H)   if 0 ≤ Sv_(z−1) + Δ_(ℑ_H) ≤ 255
     |       Sv_(z−1) ← Sv_z − Δ_(ℑ_H)   otherwise
Sv
```

(М.31)

Для кожного **μ**-го біта повідомлення проводиться обчислення індексу **z** елементу вектора контейнера **Cv**. Обчислюється різниця **Δ'** між суміжними пікселями $\mathbf{Cv_z}$ і $\mathbf{Cv_{z-1}}$. Внутрішнім циклом **i** ∈ 1..**rows**(**Δ**) проводиться пошук відповідного значення різниці у векторі **Δ**. У випадку знаходження, змінній $\mathfrak{I}$ присвоюється значення індексу **i**, який відповідає даній різниці у **Δ**. Якщо значення $\mathbf{b}_{\mathfrak{I}}$ не дорівнює поточному біту приховуваного повідомлення, відбувається пошук найближчого індексу, при якому $\mathbf{b_i}$ дорівнює бітові повідомлення. Пошук проводиться вниз (**L**) і вгору (**H**) від індексу $\mathfrak{I}$. Попереднє присвоєння змінним $\mathfrak{I}_\mathbf{L}$ та $\mathfrak{I}_\mathbf{H}$ значення ±1000 забезпечує неможливість дублювання попередніх значень $\mathfrak{I}$, якщо рух вниз або вгору від $\mathfrak{I}$ не призвело до виконання поставленої умови (останнє є можливим при знаходженні індексу $\mathfrak{I}$ надто близько до нижньої або верхньої межі вектора **b**). Після знаходження значень $\mathfrak{I}_\mathbf{L}$ і $\mathfrak{I}_\mathbf{H}$ обирається той з них, який є найближчим до початкового значення $\mathfrak{I}$. Інтенсивність пікселя контейнера $\mathbf{Sv_z}$ дорівнює збільшеній на величину $\Delta_{\mathfrak{I}_{\mathbf{L(H)}}}$ інтенсивності суміжного пікселя $\mathbf{Sv_{z-1}}$. Якщо дане збільшення призводить до виходу значення інтенсивності кольору за межі [0; 255], то, навпаки, інтенсивності суміжного пікселя $\mathbf{Sv_{z-1}}$ присвоюється значення інтенсивності пікселя $\mathbf{Sv_z}$, зменшеній на величину $\Delta_{\mathfrak{I}_{\mathbf{L(H)}}}$. Після вбудовування останнього біта повідомлення зовнішній цикл переривається.

Проводимо зворотне згортання вектора у матрицю, що має розмірність первинного масиву **C** (М.7). Одержуємо масив **S**.

5) При видобуванні повідомлення попередньо формується стеганоключ – вектори **Δ*** і **b***. Програмні модулі при цьому є ідентичними (М.29) і (М.30). Масив контейнера розгортається у вектор **Sv*** (подібно до (М.17)).

6) Модуль видобування (М.32) обчислює різницю інтенсивностей суміжних пікселів Sv^*_z та Sv^*_{z-1} і виконує пошук відповідної різниці у кодовій таблиці **Δ***. Значення біту $b^*_{\mathfrak{I}}$, яке відповідає даній різниці, присвоюється поточному елементу вектора **M***. Наприкінці модуля вектор двійкових даних перетворюється на символьний рядок.

```
M* := | z ← μ*_5
      | for μ ∈ 1.. rows(Sv*)
      |   | z ← z + step(D2B(z))
      |   | Δ' ← Sv*_z − Sv*_{z−1}   if z ≤ rows(Sv*)
      |   | break if z > rows(Sv*)
      |   | for i ∈ 1.. rows(Δ*)
      |   |   if Δ' = Δ*_i
      |   |   |  ℑ ← i
      |   |   |  break
      |   | M*_μ ← b*_ℑ
      | for j ∈ 1.. rows(M*) ÷ 8
      |   M*_vec_j ← B2D(submatrix(M*, 8·j − 7, 8·j, 1, 1))
      | vec2str(M*_vec)
```
(М.32)

Одержані при обчисленні візуального спотворення результати зведено до табл.5.1.

5.3.2.7. Метод Куттера-Джордана-Боссена

M. Kutter, F. Jordan та *F. Bossen* [83] запропонували алгоритм вбудовування до каналу синього кольору зображення, яке має RGB-кодування, оскільки до синіх кольорів ЗСЛ є найменш чутливою. Розглянемо алгоритм передачі одного біта секретної інформації. Нехай $\boldsymbol{M_i}$ – біт, що підлягає вбудовуванню, $\boldsymbol{C} = \{\boldsymbol{R}, \boldsymbol{G}, \boldsymbol{B}\}$ – зображення-контейнер, $\boldsymbol{p} = (\boldsymbol{x}, \boldsymbol{y})$ – псевдовипадковий піксель контейнеру, до якого виконуватиметься вбудовування.

Секретний біт $\boldsymbol{M_i}$ вбудовується у канал синього кольору шляхом модифікації яскравості $\lambda_{x,y} = 0.29890 \cdot \boldsymbol{R}_{x,y} + 0.58662 \cdot \boldsymbol{G}_{x,y} + 0.11448 \cdot \boldsymbol{B}_{x,y}$:

$$\boldsymbol{B}'_{x,y} = \begin{cases} \boldsymbol{B}_{x,y} - \upsilon \cdot \lambda_{x,y}, & \text{при } \boldsymbol{m_i} = 0; \\ \boldsymbol{B}_{x,y} + \upsilon \cdot \lambda_{x,y}, & \text{при } \boldsymbol{m_i} = 1. \end{cases} = \boldsymbol{B}_{x,y} + (2 \cdot \boldsymbol{m_i} - 1) \cdot \upsilon \cdot \lambda_{x,y}, \tag{5.5}$$

де υ – константа, що визначає енергію вбудовуваного сигналу. Її величина залежить від призначення стеганосистеми. Чим більшою є υ, тим вище стійкість вбудованої інформації до спотворень, але й тим сильніше її помітність.

Видобування біта одержувачем здійснюється без наявності в нього первинного зображення, тобто "наосліп". Для цього виконується передбачення значення первинного, немодифікованого пікселя на основі значень сусідніх пікселів. Для одержання оцінки пікселя запропоновано використовувати значення декількох пікселів, розміщених у тому ж стовпці і у тому ж рядку масиву контейнера. Автори використовували "хрест" пікселів розміром 7×7. Оцінка $\hat{\boldsymbol{B}}^*_{x,y}$ отримується у вигляді

$$\hat{\boldsymbol{B}}^*_{x,y} = \frac{1}{4 \cdot \sigma} \cdot \left[\sum_{i=-\sigma}^{+\sigma} \boldsymbol{B}^*_{x+i,y} + \sum_{j=-\sigma}^{+\sigma} \boldsymbol{B}^*_{x,y+j} - 2 \cdot \boldsymbol{B}^*_{x,y} \right], \tag{5.6}$$

де σ – кількість пікселів зверху (знизу, ліворуч, праворуч) від оцінюваного пікселя ($\sigma = 3$).

При видобуванні вбудованого біта обчислюється різниця δ між поточним ($\boldsymbol{B}^*_{x,y}$) і прогнозованим ($\hat{\boldsymbol{B}}^*_{x,y}$) значеннями інтенсивності пікселя $\boldsymbol{p} = (\boldsymbol{x}, \boldsymbol{y})$:

$$\delta = \boldsymbol{B}^*_{x,y} - \hat{\boldsymbol{B}}^*_{x,y}\,. \tag{5.7}$$

Знак δ означатиме біт, який було вбудовано: якщо $\delta < 0$, то $\boldsymbol{M_i} = 0$; якщо $\delta > 0$, то $\boldsymbol{M_i} = 1$.

Функції вбудовування і видобування у даному методі не є симетричними. Тобто функція видобування не становить собою зворотну функцію вбудовування. Хоча, як зазначають автори, правильне розпізнання біту повідомлення у випадку застосування описаних вище процедур, є високоімовірним, але не стовідсотково вірним. Для зменшення імовірності помилок видобування було запропоновано в процесі вбудовування кожен біт повторювати декілька разів (багатократне вбудовування). Оскільки при цьому кожен біт було повторено τ разів, то одержується τ оцінок одного біта повідомлення. Секретний біт отримується після усереднення різниці між реальним і оціненим значеннями інтенсивності пікселя в отриманому одержувачем контейнері:

$$\delta = \tau^{-1} \cdot \sum_{i=1}^{\tau} \left[\boldsymbol{B}^*_{x,y} - \hat{\boldsymbol{B}}^*_{x,y} \right]. \tag{5.8}$$

Як і у попередньому випадку, знак усередненої різниці визначатиме значення вбудованого біта. У роботі [83] показано, що алгоритм є стійким до багатьох з відомих атак: НЧ фільтрації зображення, його компресії відповідно до алгоритму JPEG, обрізанню країв.

Пропонується наступна реалізація даного методу.

1) Первинні дані стандартні.

2) Масив яскравості одержується використанням функції **READBMP**("ім'я_файлу"), що повертає масив, який представляє зображення BMP-формату у яскравісному форматі (градаціях сірого): λ := **READBMP**("C.bmp").

3) Алгоритм вбудовування реалізується програмним модулем (М.33). При цьому обчислення псевдовипадкових координат пікселя, до якого відбуватиметься вбудовування біту повідомлення виконується за алгоритмом, описаним у підпункті 5.3.2.3 для методу псевдовипадкової перестановки. Попередньо задається параметр υ, який визначає енергію вбудовуваного сигналу, а також кількість повторювань вбудовування τ одного і того ж самого біту. Встановлено, що результат вбудовування є візуально непомітним при значеннях $\upsilon < 0{,}05$. Але у цьому випадку для зменшення помилок при видобуванні доводиться значно підвищувати кількість приховувань τ поточного біту ($\tau > 35$), що також негативно відбивається на статистиці зображення. Оптимальними, на наш погляд, є значення $\upsilon \approx 0.15$ і $\tau < 20$, але, знову ж таки, все залежить від характеристик зображення, яке було обране в якості контейнера.

```
B' := | (I) -- див. (М.22), замість S ← C використати B' ← B
      | for i ∈ 1 .. τ·L_M
      |   x ← floor(i / Y) + 1
      |   y ← mod(i, Y) + 1
      |   for s ∈ 1 .. ℜ
      |     x ← mod(x + B2D((D2B(K_{2·s−1}) ⊕ submatrix(D2B(y), 1, 8, 1, 1)), X) + 1
      |     y ← mod(y + B2D((D2B(K_{2·s}) ⊕ submatrix(D2B(x), 1, 8, 1, 1)), Y) + 1
      |   j ← ceil(i / τ)
      |   B'_{x,y} ← B_{x,y} + (2·M vec_bin_j − 1)·υ·λ_{x,y}
      |   B'_{x,y} ← 255 if B'_{x,y} > 255
      |   B'_{x,y} ← 0 if B'_{x,y} < 0
      | B'
```

(М.33)

Обчислення індексу елементу двійкового вектора повідомлення за формулою **ceil(i/τ)** дозволяє один і той самий біт приховати рівно **τ** разів, після чого вбудовуватиметься наступний біт повідомлення і т.д.

Після проведення модифікації інтенсивності пікселя, визначеного координатами (**x**, **y**), проводиться корегування значення результуючої інтенсивності. Інакше, при початковому значенні інтенсивності кольору пікселя, наприклад, 255, внесення біту повідомлення «1» призведе не до зростання значення інтенсивності, а навпаки, – до зменшення інтенсивності в бік темних відтінків (всі числа, що більші за 255 при формуванні зображення автоматично В іншому випадку, при занадто низьких початкових значеннях інтенсивності, зокрема 0, внесення біту «0» може призвести до одержання від'ємного значення, яке розумітиметься як значення інтенсивності, близької до 255.

4) Перед видобуванням повідомлення необхідні бути відомими: параметри контейнера, первинний ключ K_0*, кількість циклів обчислення координат (**x**, **y**) **ℜ***, кількість дублюючих вбудовувань одного біта **τ***, а також розмірність хреста **σ** (кількість пікселів зверху (знизу, ліворуч, праворуч) від оцінюваного пікселя) – рис.5.16.

Модуль видобування прихованого повідомлення (М.34) містить у собі блок обчислення псевдовипадкових координат (**x**, **y**), ідентичний відповідному у модулі вбудовування (М.33). Далі йдуть блоки виконання умов, які у сукупності дозволяють врахувати проблемні випадки, коли оцінюваний піксель знаходиться занадто близько до краю(-їв) зображення щоб побудувати повноцінний хрест з оточуючих пікселів (див. рис.5.16).

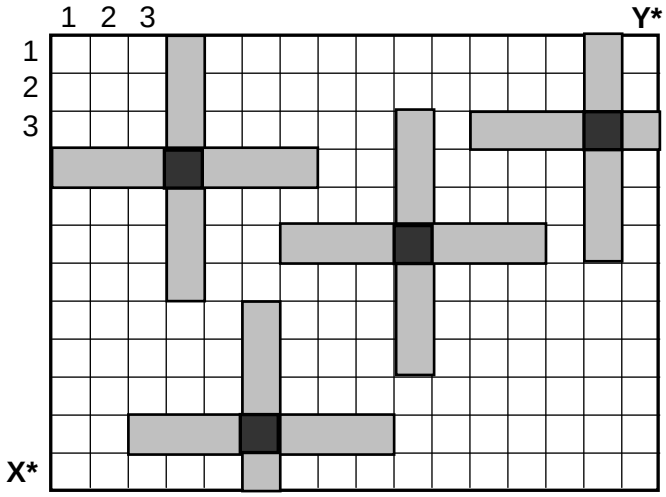


Рис.5.16. Приклади оцінюваних пікселів і оцінювальних конфігурацій ("хрестів")

$$
\begin{aligned}
&\mathbf{M^*} := \left|\begin{array}{l}
\mathbf{t} \leftarrow 1 \\
\textbf{for}\ \ \mathbf{i} \in 1 .. \mathbf{X^*} \cdot \mathbf{Y^*} \\
\quad \mathbf{x} \leftarrow \mathbf{floor}(\mathbf{i} \div \mathbf{Y^*}) + 1 \\
\quad \mathbf{y} \leftarrow \mathbf{mod}(\mathbf{i}, \mathbf{Y^*}) + 1 \\
\quad \textbf{for}\ \ \mathbf{s} \in 1 .. \mathfrak{R}^* \\
\quad\quad \mathbf{x} \leftarrow \mathbf{mod}\left(\mathbf{x} + \mathbf{B2D}\left(\overrightarrow{\left(\mathbf{D2B}\left(\mathbf{K^*}_{2 \cdot \mathbf{s}-1}\right) \oplus \mathbf{submatrix}(\mathbf{D2B}(\mathbf{y}), 1, 8, 1, 1)\right)}\right), \mathbf{X^*}\right) + 1 \\
\quad\quad \mathbf{y} \leftarrow \mathbf{mod}\left(\mathbf{y} + \mathbf{B2D}\left(\overrightarrow{\left(\mathbf{D2B}\left(\mathbf{K^*}_{2 \cdot \mathbf{s}}\right) \oplus \mathbf{submatrix}(\mathbf{D2B}(\mathbf{x}), 1, 8, 1, 1)\right)}\right), \mathbf{Y^*}\right) + 1 \\
\quad \textbf{if}\ \ \sigma < \mathbf{x} \le \mathbf{X^*} - \sigma \\
\quad\quad \sigma 1 \leftarrow -\sigma \\
\quad\quad \sigma 2 \leftarrow \sigma \\
\quad \textbf{if}\ \ \mathbf{x} \le \sigma \\
\quad\quad \sigma 1 \leftarrow 1 - \mathbf{x} \\
\quad\quad \sigma 2 \leftarrow \sigma \\
\quad \textbf{if}\ \ \mathbf{x} > \mathbf{X^*} - \sigma \\
\quad\quad \sigma 1 \leftarrow -\sigma \\
\quad\quad \sigma 2 \leftarrow \mathbf{X^*} - \mathbf{x} \\
\quad \textbf{if}\ \ \sigma < \mathbf{y} \le \mathbf{Y^*} - \sigma \\
\quad\quad \sigma 3 \leftarrow -\sigma \\
\quad\quad \sigma 4 \leftarrow \sigma \\
\quad \textbf{if}\ \ \mathbf{y} \le \sigma \\
\quad\quad \sigma 3 \leftarrow 1 - \mathbf{y} \\
\quad\quad \sigma 4 \leftarrow \sigma \\
\quad \textbf{if}\ \ \mathbf{y} > \mathbf{Y^*} - \sigma \\
\quad\quad \sigma 3 \leftarrow -\sigma \\
\quad\quad \sigma 4 \leftarrow \mathbf{Y^*} - \mathbf{y} \\
\quad \mathbf{B^*\hat{}}_{\mathbf{t}} \leftarrow \dfrac{\sum_{\mathbf{a}=\sigma 1}^{\sigma 2} \mathbf{B^*}_{\mathbf{x}+\mathbf{a}, \mathbf{y}} + \sum_{\mathbf{a}=\sigma 3}^{\sigma 4} \mathbf{B^*}_{\mathbf{x}, \mathbf{y}+\mathbf{a}} - 2 \cdot \mathbf{B^*}_{\mathbf{x}, \mathbf{y}}}{\sigma 2 - \sigma 1 + \sigma 4 - \sigma 3} \\
\quad \mathbf{B^*}_{\mathbf{t}} \leftarrow \mathbf{B^*}_{\mathbf{x}, \mathbf{y}} \\
\quad \mathbf{t} \leftarrow \mathbf{t} + 1 \\
\quad \textbf{if}\ \ \mathbf{t} > \tau^* \\
\quad\quad \delta \leftarrow \dfrac{1}{\tau^*} \cdot \sum_{\tau=1}^{\tau^*} \left(\mathbf{B^*}_{\tau} - \mathbf{B^*\hat{}}_{\tau}\right) \\
\quad\quad \mathbf{j} \leftarrow \mathbf{ceil}(\mathbf{i} \div \tau^*) \\
\quad\quad \mathbf{M^*}_{\mathbf{vec\_bin}_{\mathbf{j}}} \leftarrow 0 \quad \textbf{if}\ \ \delta \le 0 \\
\quad\quad \mathbf{M^*}_{\mathbf{vec\_bin}_{\mathbf{j}}} \leftarrow 1 \quad \textbf{if}\ \ \delta > 0 \\
\quad\quad \mathbf{t} \leftarrow 1 \\
\textbf{for}\ \ \mathbf{j} \in 1 .. \mathbf{rows}\left(\mathbf{M^*}_{\mathbf{vec\_bin}}\right) \div 8 \\
\quad \mathbf{M^*}_{\mathbf{vec}_{\mathbf{j}}} \leftarrow \mathbf{B2D}\left(\mathbf{submatrix}\left(\mathbf{M^*}_{\mathbf{vec\_bin}}, 8 \cdot \mathbf{j} - 7, 8 \cdot \mathbf{j}, 1, 1\right)\right) \\
\mathbf{vec2str}\left(\mathbf{M^*}_{\mathbf{vec}}\right)
\end{array}\right. \qquad \text{(M.34)}
\end{aligned}
$$

У наведеному модулі попередньо проводиться генерування псевдовипадкових індексів (**x**, **y**), які визначатимуть елемент масиву **B***, навколо якого робитиметься оцінка прилеглих пікселів. За результатами генерування даних індексів обчислюється кількість пікселів зверху і знизу, ліворуч і праворуч від оцінюваного. У подальшому проводиться обчислення оцінки первинного значення оцінюваного пікселя (формула (5.6)). Одержаний результат заноситься до **t**-го елементу **B*^**. Оцінюване значення пікселя $\mathbf{B^*}_{\mathbf{x}, \mathbf{y}}$ зберігається у буферному масиві $\mathbf{B^*}_{\mathbf{t}}$ (велика бета). Якщо $\mathbf{t} \le \tau^*$, продовжується накопичення оцінок пікселів, до яких було вбудовано

один і той самий біт повідомлення. У випадку **t** > **τ***, узагальнюються підсумки попереднього оцінювання: обчислюється усереднена різниця δ між первинними і оціненими значеннями інтенсивностей пікселів, що виступили контейнерами для одного біту вбудовуваних даних. В залежності від знаку одержаної різниці **j**-му елементу вектора двійкових даних (індекс елемента визначається за результатом обчислення функції **ceil**(**i**/**τ***)) присвоюється значення 0 або 1. Змінна **t** скидається в 1. Починається збір оцінок значень інтенсивностей пікселів, до яких було вбудовано наступний біт повідомлення. Процес повторюється до тих пір, доки не буде проаналізовано всі елементи графічного масиву.

Результати обчисленні візуального спотворення контейнера зведено до табл.5.1.

5.3.2.8. Метод Дармстедтера-Делейгла-Квісквотера-Макка

Нетрадиційний блоковий метод вбудовування у просторову область контейнера запропонували *V. Darmstaedter, J.-F. Delaigle, J.J. Quisquater* та *B. Macq* [100]. Розроблений ними метод дозволяє досягти компромісу між стійкістю стеганосистеми до спотворювань і якістю вбудовування і, зрештою, обчислювальною складністю. Метод базується на елементарному перцептуальному (відчуттєвому) сприйнятті і дозволяє пристосовувати вбудовування відносно поточного вмісту блоків.

Перед вбудовуванням, конфіденційна інформація перетворюється на вектор двійкових даних. Кожен біт вбудовується до окремого блоку. У розгляданому авторами варіанті розмірність блоків становила 8×8 пікселів. Головною причиною такого вибору була домірність з блоками, що використовуються при JPEG-компресії, і, таким чином, дія компресії однаково поширюватиметься на кожен з вбудованих бітів. Крім того, при цьому інформацію вбудовується з надмірністю, що підвищує загальну стійкість стеганосистеми.

Процес **вбудовування** бітів повідомлення здійснюється в загальному випадку у чотири етапи:

1) Поділ масиву зображення-контейнера на блоки 8×8 пікселів.

2) Класифікація пікселів окремого блоку на *зони* з приблизно однорідними значеннями яскравості;

3) Поділ кожної зони на *категорії* у відповідності до індивідуальної (псевдовипадкової) маски;

4) Вбудовування біту в залежності від співвідношення між середніми значеннями категорій кожної зони шляхом модифікації значень яскравості кожної категорії в кожній зоні.

Розглянемо останні три етапи докладніше.

Класифікація на зони. Мета полягає у тому, щоб розбити пікселі всередині блока на групи, які б мали приблизно однакову яскравість. Така класифікація бере до уваги особливості блока, що викликають інтерес з точки зору невидимості та стійкості. При класифікації автори виділяють три типи контрасту:

– *різко виражений контраст* (рис.5.17*а*), коли можна розрізнити дві зони, розділені стрибком яскравості;

– *поступовий контраст* (рис.5.17*б*), коли дві однорідні зони розділені поступовою зміною яскравості;

– *шумовий (нечіткий) контраст* (рис.5.17*в*) з яскравістю, розподіленою на зразок випадкового шуму (у граничному випадку шумовий контраст вироджується в однотонове зображення – контраст відсутній, всі пікселі блоку мають однакову яскравість).

Відсортовані за зростанням значення яскравостей пікселів блоку можна представити зростаючою функцією $\boldsymbol{F}(\boldsymbol{i})$, де $\boldsymbol{F}(1)$ являє собою найменше значення яскравості серед усіх наявних у даному блоці, а $\boldsymbol{F}(\boldsymbol{N}^2)$ – найбільше серед наявних у блоці значень яскравості ($\boldsymbol{N}$ – розмірність квадратного блоку). Тип контрасту блоку визначає крутизна функції $\boldsymbol{F}(\boldsymbol{i})$, яку позначимо через $\boldsymbol{S}(\boldsymbol{i})$. Нехай $\boldsymbol{S}_{\max}$ – максимальна крутизна функції $\boldsymbol{F}$, при $\boldsymbol{i} = \alpha$. Якщо $\boldsymbol{S}_{\max}$ нижче за заданий поріг $\boldsymbol{T}_1$, вважається, що блок має шумовий контраст. Якщо $\boldsymbol{S}_{\max}$ вище за поріг $\boldsymbol{T}_1$, блок має або поступовий, або різко виражений контраст. У цьому випадку додатково визначають

параметри β_+ та β_- – індекси в найближчому околі точки α, відповідно вище і нижче за неї, які задовольняють нерівності

$$S(\alpha)-S(\beta_+)>T_2 \quad \text{і} \quad S(\alpha)-S(\beta_-)>T_2, \tag{5.9}$$

де T_2 – ще одне задане значення порогу.

Якщо контраст різко виражений, то $\beta_+ \approx \alpha$ і $\beta_- \approx \alpha$. Якщо контраст поступовий, то інтервал $[\beta_+, \beta_-]$ являє собою *перехідну зону* поступового контрасту.

Класифікація пікселів $p(x, y)$ на дві зони визначається наступними правилами:

1) Для *поступового* та *різко вираженого* контрастів:

– якщо $p(x, y) \le F(\beta_-)$, піксель $p(x, y)$ належить до *зони* ***1***;

– якщо $p(x, y) \ge F(\beta_+)$, піксель $p(x, y)$ належить до *зони* ***2***;

– якщо $F(\beta_-) < p(x, y) < F(\beta_+)$, піксель $p(x, y)$ належить до *перехідної зони*.

2) Для *шумового* контрасту пікселі поділяються на дві зони однакової розмірності:

– якщо $p(x, y) < F(N^2/2)$, піксель $p(x, y)$ належить до *зони* ***1***;

– якщо $p(x, y) > F(N^2/2)$, піксель $p(x, y)$ належить до *зони* ***2***.

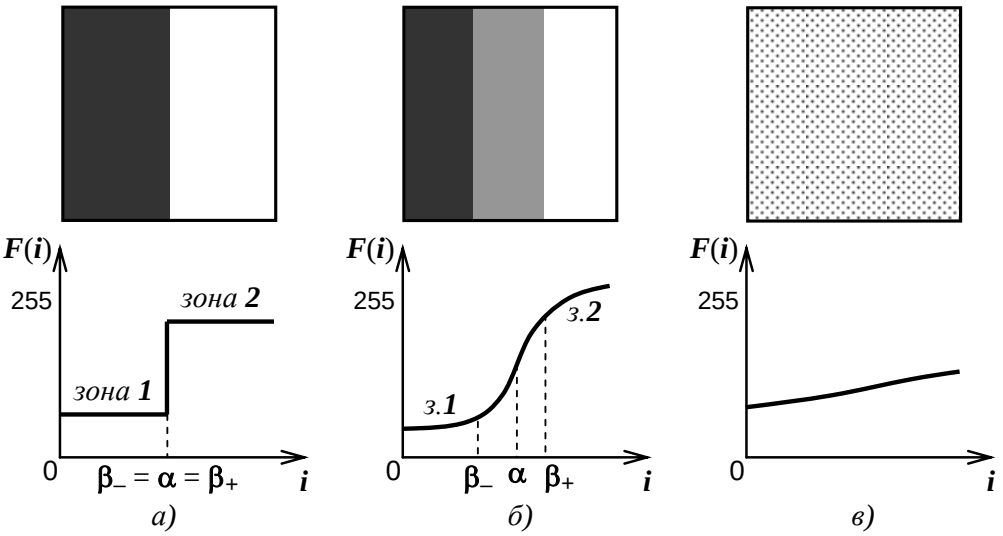


Рис.5.17. Класифікація на зони: різко виражений контраст (*а*), поступовий контраст (*б*), шумовий контраст (*в*).

У блоках 1-го і 2-го типів зони з різною яскравістю не обов'язково повинні розташовуватися впритул одна до одної, не обов'язково повинні містити рівну кількість пікселів. Більше того, деякі пікселі взагалі можуть не належати жодній з цих зон. В блоках 3-го типу класифікація більш складна.

Поділ зон на категорії. Після поділу на зони необхідно передбачити вбудовування біту шляхом модифікації певних характеристик зон. Нажаль, як зазначають автори, результат безпосереднього впливу на зони або недостатньо стійкий, або не задовольняє з по результатам візуальних спотворень вихідного зображення.

Для пошуку оптимального для вбудовування пікселя зони поділяються на дві категорії (***A*** і ***Z***). Для сортування пікселів по цим категоріям на блоки накладаються маски, причому бажаною є індивідуальність масок для кожного блоку. Призначення маскок полягає у забезпеченні секретності вбудовування. Прикладж масок для двох зон наведено на рис.5.18*а,б*.

A	A	A	A	Z	Z	Z	Z
A	A	A	A	Z	Z	Z	Z
A	A	A	A	Z	Z	Z	Z
A	A	A	A	Z	Z	Z	Z
Z	Z	Z	Z	A	A	A	A
Z	Z	Z	Z	A	A	A	A
Z	Z	Z	Z	A	A	A	A
Z	Z	Z	Z	A	A	A	A

а)

A	A	Z	Z	A	A	Z	Z
A	A	Z	Z	A	A	Z	Z
Z	Z	A	A	Z	Z	A	A
Z	Z	A	A	Z	Z	A	A
A	A	Z	Z	A	A	Z	Z
A	A	Z	Z	A	A	Z	Z
Z	Z	A	A	Z	Z	A	A
Z	Z	A	A	Z	Z	A	A

б)

Рис.5.18. Приклади використовуваних масок розмірами 4×4 (*а*) і 2×2 (*б*)

Автори рекомендують використовувати більш складні комбінації і змінювати маску при переході до приховання кожного наступного біта повідомлення. Категорія, до якої буде віднесено той або інший піксель, залежатиме від двох факторів: просторового розташування пікселя в масиві блоку та номера зони, до якої його було віднесено. Важливо відзначити, що алгоритм формування масок повинен триматися у секреті, оскільки знання конфігураці останніх суттєво знижує стійкість стеганосистеми вцілому.

Правила вбудовування бітів. За результатами виконання перших трьох етапів одеражно 4 різних групи пікселів у визначених блоках: в залежності від зони (*1* або *2*) та категорій (***A*** і ***Z***). Слід зауважити, що існує ще й п'ята група пікселів – ті, що не ввійшли до жодної із зон, але вони не беруть участі в подальшому аналізі. Для зазначених 4-х підмножин можуть бути обчислені 6 параметрів: чотири середніх значення яскравостей λ_{1A}, λ_{1Z}, λ_{2A} і λ_{2Z} для груп, що містять відповідно n_{1A}, n_{1Z}, n_{2A} і n_{2Z} пікселів; два середніх значення яскравості відповідних зон: Λ_1 і Λ_2. Середні значення яскравостей однакових зон об'єднуються разом. Таким чином, один біт повідомлення вбудовується до кожної із зон. Це підвищує стійкість системи і дозволяє вбудовувати біт без надмірної зміни блоку.

Вбудовування біту ***b*** до блоку виконується у відповідності до зв'язків між категоріями середніх значень яскравості. Правило вбудовування наступне:

$$\begin{cases} \lambda^*_{1Z} - \lambda^*_{1A} = E; \\ \lambda^*_{2Z} - \lambda^*_{2A} = E, \end{cases} \text{при } b = 0; \qquad (5.10\ a)$$

$$\begin{cases} \lambda^*_{1A} - \lambda^*_{1Z} = E; \\ \lambda^*_{2A} - \lambda^*_{2Z} = E, \end{cases} \text{при } b = 1, \qquad (5.10\ б)$$

де λ^*_{1A}, λ^*_{1Z}, λ^*_{2A} і λ^*_{2Z} – середні значення яскравості, необхідні для приховання біту ***b***; ***E*** – рівень вбудовування, тобто необхідна різниця між зазначеними середніми значеннями.

Для того, щоб зробити результат вбудовування як можна більш непомітним, повинні бути збережені низькі частоти (до яких є найбільш чутливою ЗСЛ). Збереження середніх значень інтенсивностей кожної зони забезпечується виконанням наступних умов:

$$\frac{n_{1A} \cdot \lambda^*_{1A} + n_{1Z} \cdot \lambda^*_{1Z}}{n_{1A} + n_{1B}} = \Lambda_1; \qquad (5.11\ a)$$

$$\frac{n_{2A} \cdot \lambda^*_{2A} + n_{2Z} \cdot \lambda^*_{2Z}}{n_{2A} + n_{2B}} = \Lambda_2. \qquad (5.11\ б)$$

Формули (5.10) і (5.11) дозволяють визначити значення λ^*_{1A}, λ^*_{1Z}, λ^*_{2A} і λ^*_{2Z}. Яскравість пікселів кожної зони повинна бути адаптована для збереження значень Λ_1 і Λ_2. При цьому вважається, що зміна яскравості всіх пікселів. які належать до однієї зони, є однаковою.

Нехай Δ_{1A}, Δ_{1Z}, Δ_{2A} і Δ_{2Z} – зміна яскравості. Тоді маємо:

$$\Delta_{ij} = \lambda^*_{ij} - \lambda_{ij}, \qquad (5.12)$$

де $i = \{1, 2\}$; $j = \{A, Z\}$.

Видобування вбудованої інформації з контейнера вимагає наявності інформації про розмірність блоків, на які розбивається зображення, а також про конфігурацію масок, що використовувалися при вбудовуванні. Процес видобування складається з наступних етапів:

1) розбиття зображення на блоки розмірністю $N \times N$;

2) класифікація пікселів окремого блоку на зони;

3) поділ кожної зони на категорії;

4) співставлення середніх значень яскравості для визначення значення вбудованого біту даних.

Етапи 1 і 2 є ідентичним до алгоритму вбудовування. Етап 3 пропонується розглянути докладніше.

Нехай Σ_1 і Σ_2 – значення, одержані шляхом порівняння середніх значень яскравості:

$$\Sigma_1 = \lambda_{1A} - \lambda_{1B}; \qquad (5.13\ a)$$

$$\Sigma_2 = \lambda_{2A} - \lambda_{2B}. \qquad (5.13\ б)$$

Знак обчислених Σ_1 і Σ_2 дозволяє зробити припущення щодо істинного значення прихованого біту. Крім того, абсолютні значення Σ_1 і Σ_2 несуть інформацію про рівень достовірності такого припущення. Можливі 3 випадки:

Випадок 1: $\Sigma_1 \cdot \Sigma_2 > 0$. При цьому $b^* = 1$, якщо $\Sigma_1 > 0$, і $b^* = 0$, якщо $\Sigma_2 < 0$. Рівень достовірності: *дуже високий*, якщо $|\Sigma_1|$ і $|\Sigma_2| > 2$; *дуже високий*, якщо $|\Sigma_1|$ або $|\Sigma_2| > 2.5$; *низький*, якщо $|\Sigma_1|$ і $|\Sigma_2| < 0.7$; *високий* в усіх інших випадках.

Випадок 2: $\Sigma_1 \cdot \Sigma_2 < 0$. Додатково обчислюється наступний параметр:

$$\Sigma' = \Sigma_1 \cdot (n_{1A} + n_{1Z}) + \Sigma_2 \cdot (n_{2A} + n_{2Z}).$$

При цьому $b^* = 1$, якщо $\Sigma' > 0$, і $b^* = 0$, якщо $\Sigma' < 0$. Рівень достовірності при цьому є *низьким*.

Випадок 3: $\Sigma_1 \cdot \Sigma_2 \approx 0$. Проводиться обчислення $\Sigma' = \max(|\Sigma_1|, |\Sigma_2|)$. При цьому $b^* = 1$, якщо $\Sigma' > 0$, і $b^* = 0$, якщо $\Sigma' < 0$. Рівень достовірності при цьому є *низьким*. Якщо $\Sigma' = 0$, значення біту невизначене.

Для підвищення завадостійкості автори пропонують використовувати код БЧХ.

Розглянемо реалізацію описаного вище методу у програмному середовищі MathCAD.

1) Виділимо масиви колірних компонентів зображення:

R := **READ_RED**("C.bmp"); **G** := **READ_GREEN**("C.bmp"); **B** := **READ_BLUE**("C.bmp").

Аналогічно попереднім методам, вбудовування проводитимемо до масиву **B** синьої колірної складової.

2) Визначаємо розмірність масиву контейнера **X** та **Y**, а також задаємо розмірність сегментів (блоків), на які він розбиватиметься: **X** := **rows**(**B**), **X** = 128; **Y** := **cols**(**B**), **Y** = 128; **N** := 8. При цьому загальна кількість сегментів N_S := $X \cdot Y / N^2$, N_S = 256 сегментів.

3) За допомогою програмного модуля (М.35) проводимо розбиття загального масиву **B** на окремі блоки. При цьому останні виокремлюються від з масиву за допомогою функції **submatrix** і зберігаються як елементи вектора **S**. На початковому етапі, стовпці, що обмежують виокремлюваний з масиву сегмент, мають індекси **c1** ← 1, **c2** ← **N**. Поступово нарощуючи значення індексів рядків **r1** і **r2**, повністю проходиться ділянка, обмежена стовпцями **c1** і **c2**. У випадку, коли індекс **r2** співпадає з індексом останнього рядка, індекси стовпців збільшуються на ширину виокремлюваних сегментів **N** і т.д. Зазначимо, що даний модуль не розрахований на поділ зображення на сегменти, розмірність яких не є кратною розмірності зображення. Останній випадок можна врахувати введенням до тіла програмного модуля відповідних обмежуючих умов. У даному випадку, в результаті обчислення програмного модуля повертається вектор **S** з 256 елементів, кожен з яких являє собою матрицю 8×8.

```
S := | c1 ← 1
     | c2 ← N
     | for b ∈ 1 .. N_S
     |     r1 ← mod[N·(b − 1) + 1, X]
     |     r2 ← r1 + N − 1
     |     S_b ← submatrix(B, r1, r2, c1, c2)
     |     if r2 = X
     |         c1 ← c1 + N
     |         c2 ← c2 + N
     | S
```
(М.35)

Кожен сегмент **S** призначено для приховування одного біта секретного повідомлення **M**. Тому бажано перевірити достатність кількості сегментів для цієї операції. Повідомлення є ідентичним використовуваному при розгляді попередніх методів; його довжина L_M = 200 біт. Таким чином, кількість сегментів є цілком достатньою.

4) Проводимо класифікацію пікселів кожного з блоків на зони *1* і *2* (М.36). До уваги береться лише кількість сегментів, достатня для вбудовування даного повідомлення **M**. На початку циклу зміни **s** змінній **Б** присвоюється значення **s**-го сегмента **S**. Одержаний масив **Б**

розгортається у вектор **f**, який у відсортованому вигляді присвоюється змінній **F**. Таким чином вектор **F** має N^2 елементів, що йдуть по зростанню.

Для існування можливості прийняття однакових рішень як на передавальній стороні при вбудовуванні, так і на приймальній при видобуванні стосовно точки, при якій функція **F** має найбільшу крутизну (див. рис.5.17), останню бажано згладити, використовуючи в якості вузлових точок лише деяку частину (**r**) від загальної їх кількості (N^2). Зрозуміло, що абсциса першої вузлової точки повинна дорівнювати одиниці ($\chi_1 \leftarrow 1$), останньої – $\chi_r \leftarrow N^2$. Абсциси проміжних вузлових точок формуються за виразом, зміст якого є очевидним. До масиву ϕ заносяться відповідні ординати вузлових точок.

Користуючись вбудованою функцією **lspline** формуємо вектор кубічних сплайн-коефіцієнтів других похідних при наближенні до опорних точок (при цьому граничні точки лінійні). Для проведення сплайн-апроксимації використовуємо функцію **interp**(**spline**, χ, ϕ, ω), яка для кожної шуканої абсциси ω обчислює значення апроксимованої функції.

З основ диференціального обчислення відомо [101], що кут нахилу дотичної в заданій точці до графіку функції дорівнює арктангенсу від похідної цієї

```
Zone := | for s ∈ 1 .. L_M
        |   Б ← S_s
        |   f ← Б^⟨1⟩
        |   for k ∈ 2 .. N
        |     f ← stack(f, Б^⟨k⟩)
        |   F ← sort(f)
        |   r ← 10
        |   χ_1 ← 1
        |   χ_r ← N^2
        |   for x ∈ 2 .. r − 1
        |     χ_x ← (x − 1)·round[N^2 ÷ (r − 1)]
        |   for x ∈ 1 .. rows(χ)
        |     φ_x ← F_(χ_x)
        |   spline ← lspline(χ, φ)
        |   S_max ← 0
        |   α ← 0
        |   for ω ∈ 1 .. N^2
        |     if d/dω interp(spline, χ, φ, ω) > S_max
        |       | S_max ← d/dω interp(spline, χ, φ, ω)
        |       | α ← ω
        |   α ← N^2/2  if α = 0
        |   α ← 2  if α = 1
        |   α ← (N^2 − 1)  if α = N^2
        |   for i ∈ 1 .. N                    if S_max < T_1
        |     for j ∈ 1 .. N
        |       Zone'_(i,j) ← mod(j + i, 2) + 1
        |   if S_max ≥ T_1
        |     for x ∈ α .. 1
        |       | β_ ← α  if x = 1
        |       | if (F_α − F_x) > T_2
        |       |   | β_ ← x
        |       |   | break
        |     for x ∈ α .. N^2
        |       | β_+ ← α  if x = N^2
        |       | if (F_x − F_α) > T_2
        |       |   | β_+ ← x
        |       |   | break
        |     for i ∈ 1 .. N
        |       for j ∈ 1 .. N
        |         | Zone'_(i,j) ← 1  if Б_(i,j) ≤ F_β_
        |         | Zone'_(i,j) ← 2  if Б_(i,j) ≥ F_β_+
        |         | Zone'_(i,i) ← "--" if F_β_ < Б_(i,j) < F_β_+
        |   Zone_s ← Zone'
        | Zone
```

(M.36)

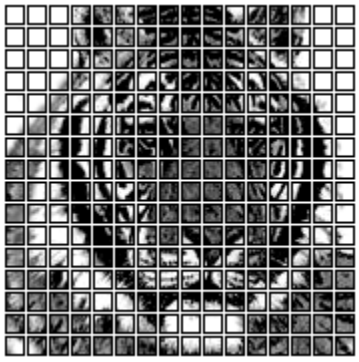
функції при даному значенні аргументу. Отже,

абсцису точки α максимальної крутизни сплайн-апроксимованої функції ϕ можна знайти за максимумом першої похідної $(d/d\omega)$ даної функції серед усіх можливих значень ω. У випадку знайдення максимального значення крутизни, змінній α присвоюється відповідне значення ω. Якщо для будь-якого ω не знайдено похідної, більшої за $\mathbf{S_{max}}$ (усі пікселі сегмента мають однакову інтенсивність), змінній α присвоюється значення $\mathbf{N}^2/2$. Якщо точка максимальної крутизни є граничною (тобто дорівнює 1 або $\mathbf{N}^2$), то вона зсувається на один крок праворуч або ліворуч відповідно (за відсутності такої поправки будуть відсутніми пікселі, що належать, відповідно, до *1*-ї або *2*-ї зони).

У тому випадку, якщо одержане значення крутизни буде меншим за порогове ($\mathbf{T}_1$), пікселі діляться між зонами *1* і *2* порівну у шаховому порядку. Якщо $\mathbf{S_{max}} \geq \mathbf{T}_1$, проводиться пошук абсцис β_- та β_+. При значення інтенсивності пікселя $\mathbf{Б}_{i,j} \leq \mathbf{F}_{\beta_-}$ останній відноситься до зони *1*; при $\mathbf{Б}_{i,j} \geq \mathbf{F}_{\beta_+}$ – до зони 2; при $\mathbf{F}_{\beta_-} < \mathbf{Б}_{i,j} < \mathbf{F}_{\beta_+}$ – до перехідної зони ("--").

В результаті обчислення модуля одержується вектор **Zone**, кожен елемент якого являє собою матрицю ідентичної розмірності із відповідним сегментом зображення ($\mathbf{N}\times\mathbf{N}$). У свою чергу кожен елемент такої матриці може приймати три значення: 1, 2 або "--", що визначає, до якої з можливих зон належить піксель сегмента **S** з ідентичними координатами.

Наведемо результати поділу зображення на сегменти та класифікації пікселів по зонам для деяких з цих сегментів (рис.5.19). Нумерація сегментів наступна: у першому стовпці – з 1-го по 16-й, у другому – з 17-го по 32-й і т.д. Значення порогів: $\mathbf{T}_1 := 6$, $\mathbf{T}_2 := 3$.



$\mathbf{Zone}_3 =$

1	2	1	2	1	2	1	2
2	1	2	1	2	1	2	1
1	2	1	2	1	2	1	2
2	1	2	1	2	1	2	1
1	2	1	2	1	2	1	2
2	1	2	1	2	1	2	1
1	2	1	2	1	2	1	2
2	1	2	1	2	1	2	1

$\mathbf{Zone}_7 =$

2	2	2	"--"	1	1	1	1
2	2	2	1	1	1	1	1
2	2	2	1	1	1	1	1
2	2	1	1	1	1	1	1
"--"	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1

$\mathbf{Zone}_{40} =$

2	2	2	2	"--"	1	1	1
2	2	2	2	1	1	1	1
2	2	2	2	1	1	1	1
2	2	2	2	1	1	1	1
2	2	2	2	1	1	1	1
2	2	2	2	2	1	1	1
2	2	2	2	2	2	1	1
2	2	2	2	2	2	1	1

а) *б)* *в)* *г)*

Рис.5.19. Схема поділу зображення на сегменти (*а*) і приклади класифікації пікселів 3-го (*б*), 7-го (*в*) і 40-го (*г*) сегментів по зонам.

5) Проводимо поділ кожної зони на категорії "A" і "Z" у відповідності до індивідуальної маски. Простий програмний модуль генерування псевдовипадкових масок зображено нижче.

```
μ := | for i ∈ 1 .. N_S
     |   for j ∈ 1 .. N^2
     |     μ'_j ← 0
     |   for j ∈ 1 .. N^2 ÷ 2
     |     μ'_(mod(i + j·K_0, N^2) + 1) ← "A"
     |   for j ∈ 1 .. N^2
     |     μ'_j ← "Z" if μ'_j = 0
     |   μ_i ← submatrix(μ', 1, N, 1, 1)
     |   for c ∈ 2 .. N
     |     μ_i ← augment[μ_i, submatrix[μ', c·N − (N − 1), c·N, 1, 1]]
     | μ
```

(M.37)

Для кожного з N_S сегментів зображення генерується масив μ розмірністю $N \times N$. На початку такого генерування створюється вектор μ' нульових елементів розмірністю N^2. Половині з цих елементів присвоюється символьне значення "A", причому присвоєння відбувається з псевдовипадковим кроком, який визначається заданим значенням параметру K_o і поточними значенням змінних циклу **i** та **j** (для того, щоб не вийти за межі розмірності вектора, використовується функція повернення залишку від ділення). Всім іншим елементам присвоюється значення "Z". Після заповнення, вектор μ' згортається до матриці μ. Приклади деяких масок, одержаних для $K_o := 123$, зображені на рис.5.20.

$\mu 1 =$

"A"	"Z"	"Z"	"A"	"Z"	"A"	"A"	"Z"
"Z"	"A"	"Z"	"A"	"A"	"Z"	"A"	"Z"
"Z"	"A"	"Z"	"Z"	"A"	"Z"	"A"	"A"
"Z"	"Z"	"A"	"Z"	"A"	"A"	"Z"	"A"
"A"	"Z"	"A"	"Z"	"Z"	"A"	"Z"	"A"
"A"	"Z"	"Z"	"A"	"Z"	"A"	"A"	"Z"
"Z"	"A"	"Z"	"A"	"A"	"Z"	"A"	"Z"
"Z"	"A"	"Z"	"Z"	"A"	"Z"	"A"	"A"

$\mu 2 =$

"A"	"Z"	"A"	"Z"	"Z"	"A"	"Z"	"A"
"A"	"Z"	"Z"	"A"	"Z"	"A"	"A"	"Z"
"Z"	"A"	"Z"	"A"	"A"	"Z"	"A"	"Z"
"Z"	"A"	"Z"	"Z"	"A"	"Z"	"A"	"A"
"Z"	"Z"	"A"	"Z"	"A"	"A"	"Z"	"A"
"A"	"Z"	"A"	"Z"	"Z"	"A"	"Z"	"A"
"A"	"Z"	"Z"	"A"	"Z"	"A"	"A"	"Z"
"Z"	"A"	"Z"	"A"	"A"	"Z"	"A"	"Z"

$\mu 3 =$

"Z"	"Z"	"A"	"Z"	"A"	"A"	"Z"	"A"
"A"	"Z"	"A"	"Z"	"Z"	"A"	"Z"	"A"
"A"	"Z"	"Z"	"A"	"Z"	"A"	"A"	"Z"
"Z"	"A"	"Z"	"A"	"A"	"Z"	"A"	"Z"
"Z"	"A"	"Z"	"Z"	"A"	"Z"	"A"	"A"
"Z"	"Z"	"A"	"Z"	"A"	"A"	"Z"	"A"
"A"	"Z"	"A"	"Z"	"Z"	"A"	"Z"	"A"
"A"	"Z"	"Z"	"A"	"Z"	"A"	"A"	"Z"

Рис.5.20. Приклади масок розбиття по категоріям для сегментів 1-3.

6) Модуль вбудовування бітів (М.38) побудований на основі виразів (5.10)-(5.12). На початку модуля створюється вектор двійкових даних на основі приховуваного повідомлення **M**. Циклом зміни **s** проводиться вбудовування окремого біта $b \leftarrow M_{vec\_bin_S}$ до блоку $Б \leftarrow S_S$ з урахуванням відповідного масиву зон $Z \leftarrow Zone_S$ та маски $м \leftarrow \mu_S$.

Розглянемо процедуру вбудовування більш детально. На основі даних стосовно належності пікселів **s**-го блоку до зони 1 або 2, а також до категорії "A" або "Z", проводиться підрахунок кількості пікселів, що задовольняють одній з можливих комбінацій зон і категорій (1A, 1Z, 2A, 2Z). Результати заносяться до відповідного елементу масиву **n** (рис.5.21), який на початку підрахунку мав нульове значення. Аналогічно проводиться підрахунок загальної яскравості $\Sigma\lambda$ пікселів, що належать тій або іншій парі зон і категорій (конфігурація заповнюваного при цьому масиву відповідає попередньому випадку – рис.5.21). За одержаними результатами обчислюється масив λ середніх значень яскравості кожної з чотирьох груп пікселів. Для скорочення програмних рядків дане обчислення ведеться з використанням операції векторизації – одночасного проведення деякої скалярної математичної операції (у нашому випадку – ділення) над усіма елементами масиву(-ів), поміченими знаком векторизації «$\rightarrow$». Звичайно, така паралельність обчислення відноситься не до самих обчислень, а лише до їх алгоритмічного запису. Тому кардинального зменшення часу виконання операції чекати при цьому не доводиться.

	1	2	
1	**1A**	**1Z**	*зони*
2	**2A**	**2Z**	
	категорії		

Рис.5.21. Конфігурація масивів **n** та $\Sigma\lambda$.

Далі для кожної зони проводиться розрахунок середніх значень яскравості Λ (формули (5.11)), які необхідно зберегти і після проведення вбудовування біту до сегменту зображення. Для розв'язку системи рівнянь (5.10), (5.11) з двома невідомими (λ^*_{1A} і λ^*_{1Z} або λ^*_{2A} і λ^*_{2Z}) використано вбудовану функцію MathCAD розв'язку лінійної системи з n рівнянь при n невідомих – **lsolve**(H,V), де H – квадратна несингулярна матриця; V – вектор, що має ту саму кількість рядків, що й матриця H.

Для зручності пояснення, перепишемо дані системи у відповідності до прийнятих у програмному модулі (М.38) позначень:

– при **b =** 1:

$$\left.\begin{aligned} \mathbf{n}_{\mathbf{x},1}\cdot\lambda_{\mathbf{x},1}+\mathbf{n}_{\mathbf{x},2}\cdot\lambda_{\mathbf{x},2}&=\Lambda_{\mathbf{x}}\cdot\left(\mathbf{n}_{\mathbf{x},1}+\mathbf{n}_{\mathbf{x},2}\right),\\ \lambda_{\mathbf{x},1}-\lambda_{\mathbf{x},2}&=\mathbf{E};\end{aligned}\right\}\ \Rightarrow\ H=\begin{pmatrix}\mathbf{n}_{\mathbf{x},1} & \mathbf{n}_{\mathbf{x},2}\\ 1 & -1\end{pmatrix},\ V=\begin{pmatrix}\Lambda_{\mathbf{x}}\cdot\left(\mathbf{n}_{\mathbf{x},1}+\mathbf{n}_{\mathbf{x},2}\right)\\ \mathbf{E}\end{pmatrix};$$

– при **b =** 0:

$$\left.\begin{aligned} \mathbf{n}_{\mathbf{x},1}\cdot\lambda_{\mathbf{x},1}+\mathbf{n}_{\mathbf{x},2}\cdot\lambda_{\mathbf{x},2}&=\Lambda_{\mathbf{x}}\cdot\left(\mathbf{n}_{\mathbf{x},1}+\mathbf{n}_{\mathbf{x},2}\right),\\ \lambda_{\mathbf{x},2}-\lambda_{\mathbf{x},1}&=\mathbf{E};\end{aligned}\right\}\ \Rightarrow\ H=\begin{pmatrix}\mathbf{n}_{\mathbf{x},1} & \mathbf{n}_{\mathbf{x},2}\\ -1 & 1\end{pmatrix},\ V=\begin{pmatrix}\Lambda_{\mathbf{x}}\cdot\left(\mathbf{n}_{\mathbf{x},1}+\mathbf{n}_{\mathbf{x},2}\right)\\ \mathbf{E}\end{pmatrix},$$

де **x** – номер зони.

$\Delta :=$ ⓘ -- див. (М.22), крім $\mathbf{S}\leftarrow\mathbf{C}$

for $\mathbf{s}\in 1..\mathbf{L}_{\mathbf{M}}$

$\mathbf{b}\leftarrow \mathbf{M}^{\mathbf{vec\_bin}_{\mathbf{s}}}$

$\mathbf{Б}\leftarrow \mathbf{S}_{\mathbf{s}}$

$\mathbf{Z}\leftarrow \mathbf{Zone}_{\mathbf{s}}$

$\mathbf{M}\leftarrow \mu_{\mathbf{s}}$

$\mathbf{n}\leftarrow\begin{pmatrix}0 & 0\\ 0 & 0\end{pmatrix}$

for $\mathbf{i}\in 1..\mathbf{N}$

for $\mathbf{j}\in 1..\mathbf{N}$

$$\left.\begin{aligned} \mathbf{n}_{1,1}&\leftarrow \mathbf{n}_{1,1}+1 \quad \text{if} \quad \mathbf{Z}_{\mathbf{i},\mathbf{j}}=1\wedge \mathbf{M}_{\mathbf{i},\mathbf{j}}=\text{"A"}\\ \mathbf{n}_{1,2}&\leftarrow \mathbf{n}_{1,2}+1 \quad \text{if} \quad \mathbf{Z}_{\mathbf{i},\mathbf{j}}=1\wedge \mathbf{M}_{\mathbf{i},\mathbf{j}}=\text{"Z"}\\ \mathbf{n}_{2,1}&\leftarrow \mathbf{n}_{2,1}+1 \quad \text{if} \quad \mathbf{Z}_{\mathbf{i},\mathbf{j}}=2\wedge \mathbf{M}_{\mathbf{i},\mathbf{j}}=\text{"A"}\\ \mathbf{n}_{2,2}&\leftarrow \mathbf{n}_{2,2}+1 \quad \text{if} \quad \mathbf{Z}_{\mathbf{i},\mathbf{j}}=2\wedge \mathbf{M}_{\mathbf{i},\mathbf{j}}=\text{"Z"}\end{aligned}\right\}\ ⓐ$$

$\Sigma\lambda\leftarrow\begin{pmatrix}0 & 0\\ 0 & 0\end{pmatrix}$

for $\mathbf{i}\in 1..\mathbf{N}$

for $\mathbf{j}\in 1..\mathbf{N}$

$$\left.\begin{aligned} \Sigma\lambda_{1,1}&\leftarrow \Sigma\lambda_{1,1}+\mathbf{Б}_{\mathbf{i},\mathbf{j}} \quad \text{if} \quad \mathbf{Z}_{\mathbf{i},\mathbf{j}}=1\wedge \mathbf{M}_{\mathbf{i},\mathbf{j}}=\text{"A"}\\ \Sigma\lambda_{1,2}&\leftarrow \Sigma\lambda_{1,2}+\mathbf{Б}_{\mathbf{i},\mathbf{j}} \quad \text{if} \quad \mathbf{Z}_{\mathbf{i},\mathbf{j}}=1\wedge \mathbf{M}_{\mathbf{i},\mathbf{j}}=\text{"Z"}\\ \Sigma\lambda_{2,1}&\leftarrow \Sigma\lambda_{2,1}+\mathbf{Б}_{\mathbf{i},\mathbf{j}} \quad \text{if} \quad \mathbf{Z}_{\mathbf{i},\mathbf{j}}=2\wedge \mathbf{M}_{\mathbf{i},\mathbf{j}}=\text{"A"}\\ \Sigma\lambda_{2,2}&\leftarrow \Sigma\lambda_{2,2}+\mathbf{Б}_{\mathbf{i},\mathbf{j}} \quad \text{if} \quad \mathbf{Z}_{\mathbf{i},\mathbf{j}}=2\wedge \mathbf{M}_{\mathbf{i},\mathbf{j}}=\text{"Z"}\end{aligned}\right\}\ ⓑ$$

$\overrightarrow{\left(\frac{\Sigma\lambda}{\mathbf{n}}\right)}$

$\lambda\leftarrow\left(\frac{\Sigma\lambda}{\mathbf{n}}\right)$

for $\mathbf{x}\in 1..2$

$$\Lambda_{\mathbf{x}}\leftarrow\frac{\lambda_{\mathbf{x},1}\cdot\mathbf{n}_{\mathbf{x},1}+\lambda_{\mathbf{x},2}\cdot\mathbf{n}_{\mathbf{x},2}}{\mathbf{n}_{\mathbf{x},1}+\mathbf{n}_{\mathbf{x},2}}$$

$$\lambda'_{\mathbf{x}}\leftarrow \mathbf{lsolve}\left[\begin{pmatrix}\mathbf{n}_{\mathbf{x},1} & \mathbf{n}_{\mathbf{x},2}\\ 1 & -1\end{pmatrix},\begin{bmatrix}\Lambda_{\mathbf{x}}\cdot\left(\mathbf{n}_{\mathbf{x},1}+\mathbf{n}_{\mathbf{x},2}\right)\\ \mathbf{E}\end{bmatrix}\right] \quad \text{if} \quad \mathbf{b}=1$$

$$\lambda'_{\mathbf{x}}\leftarrow \mathbf{lsolve}\left[\begin{pmatrix}\mathbf{n}_{\mathbf{x},1} & \mathbf{n}_{\mathbf{x},2}\\ -1 & 1\end{pmatrix},\begin{bmatrix}\Lambda_{\mathbf{x}}\cdot\left(\mathbf{n}_{\mathbf{x},1}+\mathbf{n}_{\mathbf{x},2}\right)\\ \mathbf{E}\end{bmatrix}\right] \quad \text{if} \quad \mathbf{b}=0$$

$\lambda^*\leftarrow \mathbf{augment}\left(\lambda'_1,\lambda'_2\right)^{\mathbf{T}}$

$\Delta_{\mathbf{S}}\leftarrow\lambda^*-\lambda$

Δ

(М.38)

При будь-якому значенні біта **b**, функція **lsolve** обчислюється двічі (для кожної із зон). Результатом обчислення функції **lsolve** є вектор з двох елементів, значення яких відповідають шуканим невідомим. Одержані два вектори об'єднуємо стовпець до стовпця, а результат об'єднання транспонуємо (щоб перейти до конфігурації, зображеної на рис.5.21).

На завершальному етапі обчислюється різниця (5.12). Результат обчислення (який, звичайно, також має конфігурацію, що відповідає рис.5.12) присвоюється **s**-му елементу вектора **Δ**.

7) Проводимо модифікацію блоків зображення, використовуючи програмний модуль (М.39). При цьому, якщо номер блоку **s** перевищує загальну кількість біт у повідомленні (**L_M**), модифікація блоку не проводиться.

```
SM := | for s ∈ 1 .. N_S
      |   Б ← S_s
      |   if s ≤ L_M
      |     Б' ← Б
      |     ∇ ← Δ_s
      |     Z ← Zone_s
      |     м ← μ_s
      |     for i ∈ 1 .. N
      |       for j ∈ 1 .. N
      |         Б'_{i,j} ← Б_{i,j} + ∇_{1,1}  if Z_{i,j} = 1 ∧ м_{i,j} = "A"
      |         Б'_{i,j} ← Б_{i,j} + ∇_{1,2}  if Z_{i,j} = 1 ∧ м_{i,j} = "Z"
      |         Б'_{i,j} ← Б_{i,j} + ∇_{2,1}  if Z_{i,j} = 2 ∧ м_{i,j} = "A"
      |         Б'_{i,j} ← Б_{i,j} + ∇_{2,2}  if Z_{i,j} = 2 ∧ м_{i,j} = "Z"
      |   Б' ← Б  if s > L_M
      |   SM_s ← Б'
      | SM
```

(М.39)

8) Повертаємо блоки на відповідні їм місця у зображенні, користуючись модулем (М.40). На першому етапі перші **X/N** блоків (у нашому випадку – 16) об'єднуються в один спільний масив шляхом об'єднання останнього рядка попереднього блоку з першим рядком наступного (функція **stack**). В подальшому за допомогою циклу подібна операція повторюється над кожною наступною групою з **X/N** блоків. Згруповані таким чином блоки паралельно об'єднуються між собою стовпець до стовпця (функція **augment**).

```
BM := | BM ← SM_1
      | for b ∈ 2 .. X/N
      |   BM ← stack(BM, SM_b)
      | BM' ← 0
      | for b ∈ X/N + 1 .. N_S
      |   BM' ← SM_b  if BM' = 0
      |   BM' ← stack(BM', SM_b)  otherwise
      |   if mod(b, X/N) = 0
      |     BM ← augment(BM, BM')
      |     BM' ← 0
      | BM
```

(М.40)

Відтворене на основі масиву **BM** зображення у більшості випадків буде занадто спотвореним (рис.5.22) через вже згаданий вище вихід значень інтенсивностей пікселів за межі [0; 255]. Тому даний масив необхідно додатково пронормувати, для чого, наприклад, можна використати програмний модуль (М.41).

$$\mathbf{BM}_{norm} := \left| \begin{array}{l} \mathbf{BM}_{min} \leftarrow \min(\mathbf{BM}) \\ \mathbf{BM}_{max} \leftarrow \max(\mathbf{BM}) \\ \text{for } x \in 1..X \\ \quad \text{for } y \in 1..Y \\ \quad\quad \mathbf{BM}_{norm_{x,y}} \leftarrow \text{round}\left(\dfrac{\mathbf{BM}_{x,y} + |\mathbf{BM}_{min}|}{\mathbf{BM}_{max} + |\mathbf{BM}_{min}|} \cdot 255 \right) \\ \mathbf{BM}_{norm} \end{array} \right. \qquad \text{(M.41)}$$

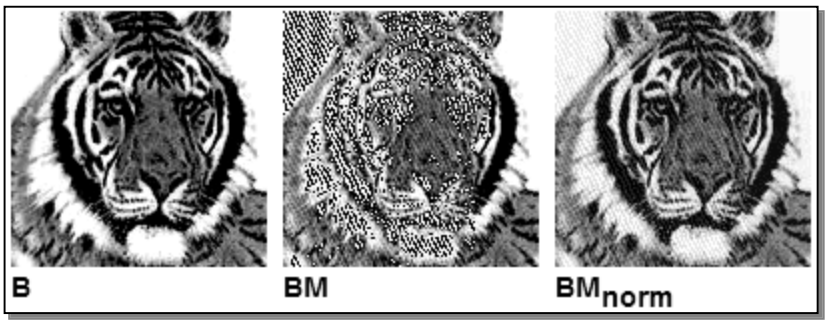


Рис.5.22. Порівняння первинного (**B**), модифікованого (**BM**) (при значенні **E** = 35) і додатково нормованого (**BMnorm**) зображень

9) Перейдемо до видобування прихованого повідомлення. Прийняті позначення: **BM*** – заповнений контейнер; **X*** та **Y*** – відповідно, розміри контейнера по вертикалі (кількість рядків пікселів) та горизонталі (кількість стовпців); **N*** – розмірність виокремлюваного блоку контейнера; **N*S** – загальна кількість блоків; **T*1**, **T*2** – значення 1-го і 2-го порогів порівняння; **μ*** – масив секретних масок.

Програмні модулі поділу зображення на блоки (**SM***), класифікації пікселів блоку на зони (**Zone***) є аналогічними, відповідно, (M.35) і (M.36).

На відміну від операції вбудовування, коли обчислення кількості пікселів **n**, що задовольняють одній з можливих комбінацій зон і категорій, а також середніх значень яскравості **λ** кожної з чотирьох груп пікселів велося безпосередньо у програмному модулі (див. (M.38)), при видобуванні зазначені масиви необхідно сформувати окремо (M.42), (M.43).

$$\mathbf{n}^* := \left| \begin{array}{l} \text{for } s \in 1..N^*_S \\ \quad Z \leftarrow \mathbf{Zone}^*_s \\ \quad M \leftarrow \mu^*_s \\ \quad \mathbf{n} \leftarrow \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \\ \quad \textcircled{а} \text{ -- див. (M.38)} \\ \quad \mathbf{n}^*_s \leftarrow \mathbf{n} \\ \mathbf{n}^* \end{array} \right. \qquad \text{(M.42)}$$

$$\lambda^* := \left| \begin{array}{l} \text{for } s \in 1..N^*_S \\ \quad Б \leftarrow \mathbf{SM}^*_s \\ \quad Z \leftarrow \mathbf{Zone}^*_s \\ \quad M \leftarrow \mu^*_s \\ \quad \mathbf{n} \leftarrow \mathbf{n}^*_s \\ \quad \Sigma\lambda \leftarrow \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \\ \quad \textcircled{б} \text{ -- див. (M.38)} \\ \quad \lambda^*_s \leftarrow \overrightarrow{\left(\dfrac{\Sigma\lambda}{\mathbf{n}} \right)} \\ \lambda^* \end{array} \right. \qquad \text{(M.43)}$$

Порівняння середніх значень яскравості для кожної з зон (формула (5.13)) проводимо, використовуючи модуль (M.44).

Безпосередньо модуль видобування – (M.45), в якому реалізовано перевірку умов, викладених у теоретичному описі даного методу. Параметр **ε** обирається достатньо близьким до нуля (~ 0.05).

$$\Sigma := \left| \begin{array}{l} \text{for } s \in 1..N^*_S \\ \quad \text{for } x \in 1..2 \\ \quad\quad E_x \leftarrow (\lambda^*_s)_{x,1} - (\lambda^*_s)_{x,2} \\ \quad \Sigma_s \leftarrow E \\ \Sigma \end{array} \right. \qquad \text{(M.44)}$$

```
M* vec_bin := | for s ∈ 1 .. N* S
              |   E ← Σ_S
              |   n ← n*_s
              |   if E_1·E_2 > ε
              |     M*_s ← 1  if E_1 > 0
              |     M*_s ← 0  if E_1 < 0
              |   if E_1·E_2 < ε
              |     E' ← E_1·(n_{1,1} + n_{1,2}) + E_2·(n_{2,1} + n_{2,2})
              |     M*_s ← 1  if E' > 0
              |     M*_s ← 0  if E' < 0
              |   if −ε ≤ E_1·E_2 ≤ ε
              |     E" ← max(|E_1|, |E_2|)
              |     M*_s ← 1  if E" > 0
              |     M*_s ← 0  if E" < 0
              |     M*_s ← "0/1"  if E" = 0
              | M*
```
(М.45)

Вектор двійкових даних, що повертається модулем (М.45), перетворюється на рядок символів аналогічно тому, як це робилося при моделюванні попередніх методів.

До табл.5.1 зведено результати, одержані при обчисленні візуального спотворення .

5.3.2.9. Інші методи приховування даних у просторовій області

Нетрадиційним є алгоритм, заснований на *копіюванні блоків* з випадково обраної текстурної області до іншої, яка має подібні статистичні характеристики [14]. Це призводить до появи в зображенні повністю ідентичних блоків. Ці блоки можуть бути виявлені в такий спосіб:

– аналіз функції автокореляції стеганозображення і знаходження її піків;

– зсув зображення відповідно до цих піків і віднімання зображення з його зсунутої копії;

– різниця в місцях розміщення копійованих блоків повинна бути близькою до нуля. Тому можна обрати деякий поріг і значення, менші за цей поріг по абсолютній величині, вважати шуканими блоками.

Оскільки копії блоків ідентичні, вони змінюються однаково при перетвореннях усього зображення. Якщо зробити розмір блоків досить великим, алгоритм буде стійким до більшості з негеометричних спотворень. У проведених авторами експериментах була підтверджена стійкість алгоритму до фільтрації, компресія, обертання зображення [14].

Основним недоліком алгоритму бачиться виняткова складність знаходження достатньої кількості областей, блоки з яких можуть бути замінені без помітного погіршення якості зображення. Крім того, у даному алгоритмі як контейнер можуть використовуватися лише досить текстурні зображення.

Алгоритм, запропонований у [102], дозволяє вбудовувати інформацію до блоків 8×8 зображення-контейнера. На початку алгоритму створюється маска $\boldsymbol{\mu}(x, y)$, розмірність якої відповідає розмірності масиву контейнера, а елементами є псевдовипадково розподілені 0 і 1: $\boldsymbol{\mu}(x, y) \in \{0; 1\}$. Кожен блок $\boldsymbol{Б}$ в залежності від значення елементів маски ділиться на два підмасиви $\boldsymbol{Б}_1$ і $\boldsymbol{Б}_2$. Для кожного підмасиву обчислюються середні значення яскравості – λ_1 і λ_2. Біт приховуваного повідомлення вбудовується наступним чином:

$$s(x, y) = \begin{cases} 1, \text{ при } \lambda_1 - \lambda_2 > E; \\ 0, \text{ при } \lambda_1 - \lambda_2 < -E, \end{cases} \tag{5.14}$$

де E – деякий поріг (необхідна різниця між зазначеними середніми значеннями яскравості).

У тих випадках, коли умова (5.14) не виконується, відповідним чином модифікують значення яскравості пікселів одного з підмасивів ($\boldsymbol{Б}_1$ або $\boldsymbol{Б}_2$). Для видобування біту прихованого повідомлення проводиться обчислення відповідних середніх значень яскравості підмасивів – λ^*_1 і λ^*_2. Різниця між ними дозволяє визначити значення прихованого біту:

$$\boldsymbol{b_i} = \begin{cases} 1, \text{ при } \lambda^*_1 - \lambda^*_2 > 0; \\ 0, \text{ при } \lambda^*_1 - \lambda^*_2 < 0; \\ \boldsymbol{?} \text{ при } \lambda^*_1 - \lambda^*_2 = 0. \end{cases} \qquad (5.15)$$

5.3.3. Приховування даних у частотній області зображення

Як вже було зазначено вище, стеганографічні методи приховування даних у *просторовій* області зображення є нестійкими до переважної більшості відомих видів спотворювань. Так, наприклад, застосування операції компресії із втратами (стосовно зображення, це може бути JPEG-компресія) призводить до часткового або, що є більш імовірним, повного знищення вбудованої до контейнера інформації. Більш стійкими до різноманітних спотворювань, у тому числі й компресії, є методи, які використовують для приховування даних не просторову область контейнера, а *частотну*.

Існує декілька способів представлення зображення в частотній області. При цьому використовується та або інша декомпозиція зображення, яке використовується в якості контейнера. Наприклад, існують методи на основі використання *дискретного косинусного перетворення* (ДКП), *дискретного перетворення Фур'є* (ДПФ), *вейвлет-перетворення, перетворення Карунена-Лоева* та деякі інші. Подібні перетворення можуть застосовуватися або до окремих частин зображення, або ж до зображення в цілому.

Найбільшого поширення серед усіх ортогональних перетворень, у стеганографії зазнали вейвлет-перетворення і ДКП [5], що певною мірою пояснюється значним поширенням їх застосування під час компресії зображень. Крім того, є доцільним застосовувати для приховування даних саме те перетворення зображення, якому останнє піддаватиметься згодом при можливій компресії. Так, відомо, що алгоритм ДКП є базовим у стандарті JPEG, а вейвлет-перетворення – у стандарті JPEG2000. Стеганоалгоритм може бути досить стійким до подальшої компресії зображення, лише якщо він враховуватиме особливості алгоритму перспективного стиснення. При цьому, звичайно, стеганоалгоритм, до основи якого закладено вейвлет-перетворення, зовсім не обов'язково буде стійким до дискретнокосинусного алгоритму стиснення, і навпаки. Автори [5] зазначають, що ще більш значні труднощі виникають при обранні методу стеганоперетворення під час приховання даних у потоковому відео. Причиною цього є те, що алгоритми компресії відеоінформації на додачу до компресії нерухомого кадру, однією з свої складових мають кодер векторів компенсування руху (цілком очевидно, що у нерухомому зображенні таке компенсування є відсутнім за непотрібністю). Для того, щоб бути в достатній мірі стійким, стеганоалгоритм повинен враховувати даний фактор.

На сьогодні відкритим залишається питання існування стійкого стеганоперетворення, яке було б незалежним від застосовуваного в подальшому алгоритму компресії. Автори [19,44] з позицій теорії інформації провели розгляд різних ортонормальних перетворень, таких як ДПФ, ДКП, перетворення Хартлі, субсмугове перетворення та ін.

На сьогоднішній день відомо багато моделей для оцінки пропускної здатності каналу передачі прихованих даних. Приведемо нижче модель, представлену у роботах [5,84].

Нехай $\boldsymbol{C}$ – первинне зображення (контейнер-оригінал), $\boldsymbol{M}$ – повідомлення, що підлягає приховуванню. Тоді модифіковане зображення (стеганоконтейнер) $\boldsymbol{S} = \boldsymbol{C} + \boldsymbol{M}$. Модифіковане зображення $\boldsymbol{S}$ візуально нерозрізненне з первинним і може бути піддано у стеганоканалі компресії із втратами: $\boldsymbol{S}^{\nabla} = \Theta(\boldsymbol{S})$, де $\Theta(\bullet)$ – оператор компресії. Завдання одержувача – видобути з одержаного контейнера $\boldsymbol{S}^{\nabla}$ вбудовані на попередньому етапі біти даних $\boldsymbol{M_i}$. Основне, що нас цікавитиме при цьому, – відповідь на питання: яку кількість біт можна ефективно вбудувати до зображення і згодом видобути з нього за умови задовільно низької імовірності помилок на

Таблиця 5.1.

Показники візуального спотворення у випадку приховування даних в просторовій області зображення

Назва показника спотворення	*Оригінал*	*Методи приховування у просторовій області*							
		НЗБ	*ПВ інтервал*	*ПВП*	*Блокового кодування*	*Заміни палітри*	*Квантування*	*Куттера-Джордона*	*Дармстедтера-Делейгла*
Макс. абсолютна різниця, ***MD***	0	1	1	1	1	3	3	38	54
Середня абсолютна різниця, ***AD***	0	0.494	$7.690\cdot10^{-3}$	$5.920\cdot10^{-3}$	$6.165\cdot10^{-3}$	$9.827\cdot10^{-3}$	$7.141\cdot10^{-3}$	4.588	17.704
Норм. середня абс. різниця, ***NAD***	0	$3.823\cdot10^{-3}$	$5.956\cdot10^{-5}$	$4.585\cdot10^{-5}$	$4.774\cdot10^{-5}$	$7.611\cdot10^{-5}$	$5.535\cdot10^{-5}$	0.050	0.137
Середньоквадр. помилка, ***MSE***	0	0.494	$7.690\cdot10^{-3}$	$5.920\cdot10^{-3}$	$6.165\cdot10^{-3}$	0.017	$9.460\cdot10^{-3}$	235.708	456.887
Нормована середньоквадр. помилка, ***NMSE***	0	$2.010\cdot10^{-5}$	$3.132\cdot10^{-7}$	$2.411\cdot10^{-7}$	$2.510\cdot10^{-7}$	$7.084\cdot10^{-7}$	$3.853\cdot10^{-7}$	$9.599\cdot10^{-3}$	0.019
$\boldsymbol{L^p}$-норма, $\boldsymbol{p} = 2$	0	0.703	0.088	0.077	0.079	0.132	0.097	11.301	21.375
Лапласова середньоквадр. помилка, ***LMSE***	0	$9.815\cdot10^{-4}$	$1.560\cdot10^{-5}$	$1.263\cdot10^{-5}$	$1.253\cdot10^{-5}$	$1.990\cdot10^{-5}$	$1.855\cdot10^{-5}$	0.240	0.420
Відношення с/ш, ***SNR***	∞	$4.975\cdot10^{4}$	$3.193\cdot10^{6}$	$4.148\cdot10^{6}$	$3.983\cdot10^{6}$	$1.412\cdot10^{6}$	$2.596\cdot10^{6}$	192.271	53.746
Макс. відношення с/ш, ***PSNR***	∞	$1.317\cdot10^{5}$	$8.455\cdot10^{6}$	$1.044\cdot10^{7}$	$1.055\cdot10^{7}$	$3.738\cdot10^{6}$	$6.873\cdot10^{6}$	509.139	142.322
Якість зображення, ***IF***	1	0.999980	≈1	≈1	≈1	0.999999	≈1	0.994799	0.981394
Норм. взаємна кореляція, ***NC***	1	0.999439	0.999992	0.999998	0.999988	0.999942	1.000001	0.988343	0.942705
Якість кореляції, ***CQ***	190.182	190.076	190.181	190.182	190.180	190.172	190.183	187.966	179.286
Структурний зміст, ***SC***	1	1.001103	1.000016	1.000004	1.000025	1.000114	0.999999	1.018447	1.106175
Загальне сигма-відношення с/ш, ***GSSNR***	∞	$1.298\cdot10^{5}$	$9.751\cdot10^{6}$	$8.026\cdot10^{6}$	$5.306\cdot10^{6}$	$5.797\cdot10^{5}$	$3.648\cdot10^{6}$	187.522	31.555
Сигма-відношення с/ш, ***SSNR***	∞	142.5	62	42.4	39.7	7.6	19.5	41.8	57.3
Нормоване відношення с/помилка, ***NSER***	256	60	155	175	179	241	214	111	83.5
Подібність гістограм, ***HS***	0	3918	176	138	154	184	150	2068	10372

останньому етапі. Іншими словами, яка пропускна здатність каналу передачі прихованих даних за умови наявності у каналі зв'язку певного алгоритму компресії?

Блок-схема такого стеганоканалу представлена на рис.5.23.

Повідомлення $\boldsymbol{M}$ передається по каналу, який має два джерела "шуму": $\boldsymbol{C}$ – зображення-контейнер і "шум" $\boldsymbol{\Theta}$, що виникає в результаті операцій компресії/декомпресії. $\boldsymbol{S}^{\nabla}$, $\boldsymbol{M}^{*}$ – можливо спотворені стеганоконтейнер і, як результат, – повідомлення.

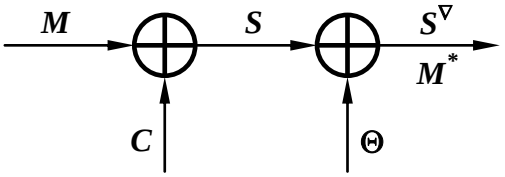


Рис.5.23. Блок-схема стеганоканалу з атакою у вигляді компресії

Структурна схема стеганосистеми приведена на рис.5.24.

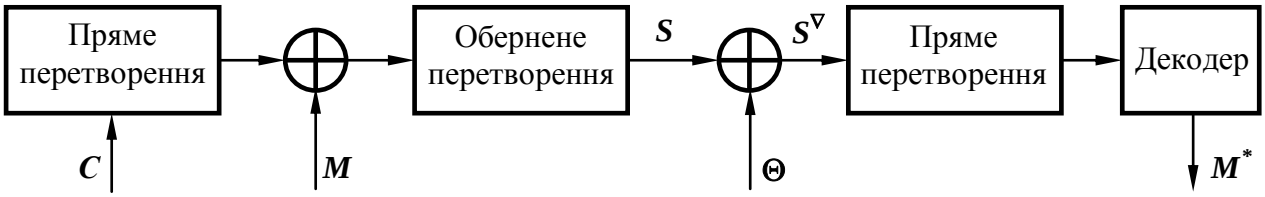


Рис.5.24. Структурна схема стеганосистеми за наявності в стеганоканалі атаки компресії

Зображення розкладається на $\boldsymbol{D}$ субсмуг (пряме перетворення), до кожної з яких вбудовується приховувана інформація. Після оберненого перетворення отримується модифіковане зображення $\boldsymbol{S}$. Після компресії/декомпресії $\boldsymbol{\Theta}$ в каналі зв'язку одержується зображення $\boldsymbol{S}^{\nabla}$, яке на приймальній стороні знову піддається прямому перетворенню і з кожної із $\boldsymbol{D}$ субсмуг незалежно видобувається приховане повідомлення $\boldsymbol{M}^{*}$.

Як зазначається у [5], реальні зображення не становлять собою випадкові процеси з рівномірно розподіленими значеннями величин. Відомо, і це використовується в алгоритмах компресії, що більша частина енергії зображень зосереджена в низькочастотній (НЧ) частині спектру. Звідси й виникає потреба у здійсненні декомпозиції зображення на субсмуги, до яких додається стеганоповідомлення. НЧ субсмуги містять переважну частину енергії зображення і, таким чином, носять шумовий характер. Високочастотні (ВЧ) субсмуги найбільшим чином піддаються впливові з боку різноманітних алгоритмів обробки, як наприклад компресія або НЧ-фільтрація. Отже, для вбудовування повідомлення найоптимальнішими є середньочастотні субсмуги спектра зображення. Типовий розподіл шуму зображення і обробки за спектром частоти зображено на рис.5.25 [44,85].

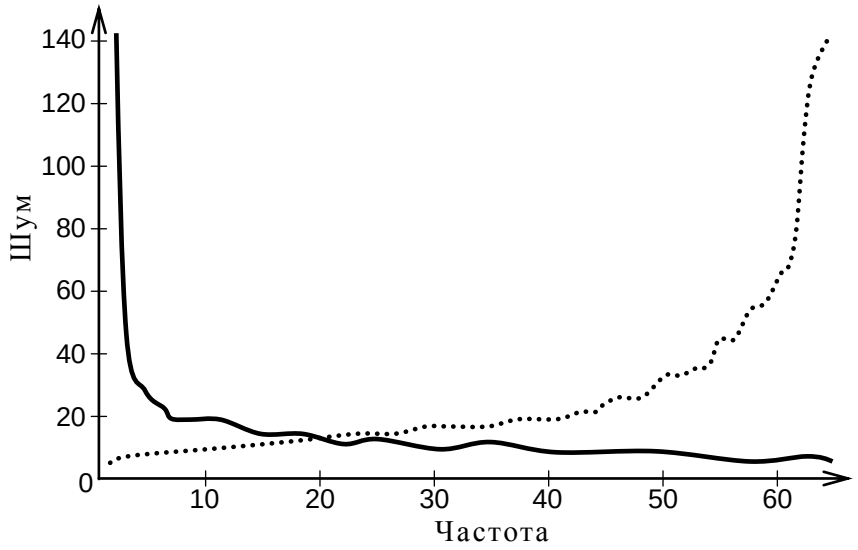


Рис.5.25. Залежність шуму зображення (суцільна лінія) і шуму обробки (пунктирна лінія) від частоти

Стеганографічний канал можна розкласти на низку незалежних підканалів. Такий поділ здійснюється за рахунок виконання прямого і оберненого перетворення. У кожному з D підканалів існує по два джерела шуму. Нехай $\sigma^2_{\tilde{N},\Theta_j}$, при j = 1, ..., D – дисперсія коефіцієнтів перетворення (шуму зображення) у кожному з підканалів. Тоді вираз для пропускної здатності каналу стеганосистеми набуде вигляду:

$$B = \frac{X \cdot Y}{2 \cdot D} \cdot \sum_{j=1}^{D} \log_2 \left(1 + \frac{\nu_j^2}{\sigma^2_{C_j} + \sigma^2_{\Theta_j}} \right) \text{ біт}, \qquad (5.16.)$$

де ν_j – візуальний поріг для j-ї субсмуги (ν_j^2 – максимально припустима енергія стеганоповідомлення, виходячи з вимог збереження візуальної якості зображення); X та Y – піксельний розмір зображення-контейнера.

Обрання значення візуального порогу ґрунтується на урахуванні властивостей ЗСЛ. Відомо, що шум у ВЧ областях зображення більш прийнятний, чим у НЧ областях. Можна ввести деякі зважувальні коефіцієнти: $\nu_j^2 = \kappa \cdot \sigma^{2\alpha}_{\tilde{N},\Theta_j}$, де $0 \le \alpha \le 1$, а $\kappa << \sigma_{\tilde{N},\Theta_j}$ для будь-якого j є константою. Випадок, коли α = 0, відповідає рівномірному розподілу стеганограми по всім субсмугам. Випадок α = 1 відповідає розподілу стеганограми відповідно до дисперсій субсмуг. Після деяких спрощень можна одержати вираз для пропускної здатності каналу передачі прихованих даних:

$$B = \frac{X \cdot Y}{2 \cdot D} \cdot \sum_{j=1}^{D} \log_2 \left(1 + \frac{\kappa_1 \cdot \sigma^{2\alpha}_{\tilde{N}_j}}{\sigma^2_{C_j}} \right) \approx \frac{X \cdot Y}{2 \cdot D} \cdot \log_2 \left(1 + \sum_{j=1}^{D} \frac{\kappa_1}{\sigma^{2(1-\alpha)}_{C_j}} \right). \qquad (5.17)$$

У виразі (5.17) знак наближення є справедливим, оскільки $\frac{\kappa_1 \cdot \sigma^{2\alpha}_{\tilde{N}_j}}{\sigma^2_{C_j}} << 1$ для будь-якого значення j. Як видно, при α = 1 декомпозиція жодним чином не впливатиме на пропускну здатність стеганоканалу. При $\alpha < 1$ пропускна здатність зростатиме за рахунок того, що в області з низькою дисперсією (високочастотній області) додається відносно більше енергії стеганосигналу.

У [85] було проведено численні експерименти, які дозволили авторам дати певні рекомендації стосовно обрання виду перетворення для стеганографії. Відомо, що перетворення можна впорядкувати за досяжними виграшами від алгоритму кодування (рис.5.26). Під виграшем від кодування розуміється ступінь перерозподілу дисперсій коефіцієнтів перетворення.

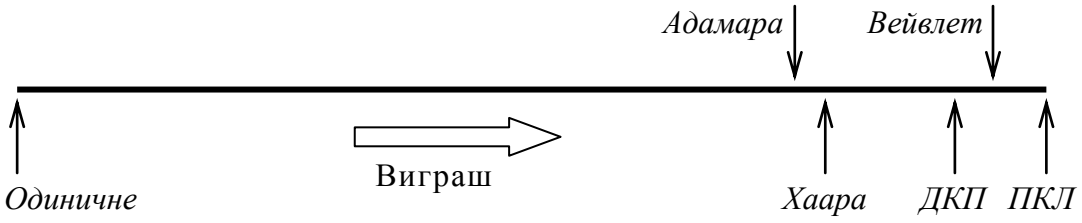


Рис.5.26. Види перетворень, упорядковані за досяжним виграшем від алгоритму кодування

Найбільший виграш дає перетворення Карунена-Лоева (ПКЛ), найменший – розкладання за базисом одиничного імпульсу (тобто відсутність перетворення). Перетворення, що мають високі значення виграшу від кодування, такі як ДКП, вейвлет-перетворення, характеризуються різко нерівномірним розподілом дисперсій коефіцієнтів субсмуг. Високочастотні субсмуги не підходять для вбудовування через великий шум обробки, а низькочастотні – через високий шум зображення (див. рис.5.25). Тому, як зазначається у [85], доводиться обмежуватися

середньочастотними смугами, у яких шум зображення приблизно дорівнює шуму обробки. Оскільки таких смуг небагато, то пропускна здатність стеганоканалу є порівняно малою. У випадку застосування перетворення з більш низьким виграшем від кодування, наприклад, перетворення Адамара або Фур'є, існує більше блоків, у яких шум зображення приблизно дорівнює шуму обробки, а, отже, є вищою і пропускна здатність. Висновок, до якого прийшли автори зазначеної роботи, є досить несподіваним: для підвищення пропускної здатності стеганографічного каналу доцільно застосовувати перетворення з меншими виграшами від кодування, які погано підходять для компресії сигналів.

Ефективність застосування вейвлет-перетворення і ДКП для компресії зображень пояснюється тим, що вони добре моделюють процес обробки зображення в ЗСЛ, відокремлюючи значимі деталі від незначущих. Отже, дані перетворення більш доречно застосовувати у випадку активного порушника, оскільки модифікація значимих коефіцієнтів може привести до неприйнятного спотворення зображення. При застосуванні перетворення з низькими значеннями виграшу від кодування існує значна небезпека порушення вбудованих даних, через те що коефіцієнти перетворення є менш стійкими до модифікацій. Проте, існує більша гнучкість у виборі перетворення, і якщо останнє є невідомим порушникові (хоча це й суперечить принципові Керхгофса), то модифікація стеганограми буде суттєво ускладненою.

Під час цифрової обробки зображення часто застосовується двомірна версія *дискретного косинусного перетворення*:

$$\Omega(\upsilon,\nu)=\frac{\zeta(\upsilon)\cdot\zeta(\nu)}{\sqrt{2\cdot N}}\cdot\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}C(x,y)\cdot\cos\left[\frac{\pi\cdot\upsilon\cdot(2\cdot x+1)}{2\cdot N}\right]\cdot\cos\left[\frac{\pi\cdot\nu\cdot(2\cdot y+1)}{2\cdot N}\right]; \qquad (5.18\ a)$$

$$S(x,y)=\frac{1}{\sqrt{2\cdot N}}\cdot\sum_{\upsilon=0}^{N-1}\sum_{\nu=0}^{N-1}\zeta(\upsilon)\cdot\zeta(\nu)\cdot\Omega(\upsilon,\nu)\cdot\cos\left[\frac{\pi\cdot\upsilon\cdot(2\cdot x+1)}{2\cdot N}\right]\cdot\cos\left[\frac{\pi\cdot\nu\cdot(2\cdot y+1)}{2\cdot N}\right], \qquad (5.18,\ б)$$

де $C(x,y)$ і $S(x,y)$ – відповідно, елементи первинного і відтвореного за коефіцієнтами ДКП зображення розмірністю $N\times N$; x, y – просторові координати пікселів зображення; $\Omega(\upsilon,\nu)$ – масив коефіцієнтів ДКП; υ, ν – координати у частотній області; $\zeta(\upsilon)=1/\sqrt{2}$, якщо υ = 0, і $\zeta(\upsilon)=1$, якщо $\upsilon > 0$.

Розглянемо існуючі методи, що базуються на алгоритмі ДКП.

5.3.3.1. Метод відносної заміни величин коефіцієнтів ДКП (метод Коха і Жао)

Один з найпоширеніших на сьогодні методів приховування секретної інформації в частотній області зображення полягає у *відносній зміні величин коефіцієнтів ДКП*, який свого часу описали *E. Koch* і *J. Zhao* [86,87]. Для цього первинне зображення розбивається на боки розміром 8×8 пікселів. ДКП застосовується до кожного блоку (формула (5.18 *а*)), в результаті чого одержуються матриці 8×8 коефіцієнтів ДКП, які зазвичай позначаються $\Omega_b(\upsilon, \nu)$, де b – номер блоку C, а (υ, ν) – позиція коефіцієнта у цьому блоці. Кожен блок при цьому призначений для приховування одного біта даних.

Було описано дві реалізації алгоритму: псевдовипадково можуть обиратися два або три коефіцієнти ДКП. Розглянемо перший варіант. Під час організації секретного каналу абоненти повинні попередньо домовитися про конкретні два коефіцієнти ДКП з кожного блоку, які будуть використовуватися для приховування даних: (υ_1, ν_1) та (υ_2, ν_2). Зазначені коефіцієнти повинні відповідати косинус-функціям із середніми частотами, що забезпечить прихованість інформації у суттєвих для ЗСЛ областях сигналу, до того ж інформація не буде знищуватися при JPEG-компресії.

Процес приховування починається з випадкового обрання блоку C_b зображення, призначеного для кодування b-го біта повідомлення. Вбудовування інформації здійснюється таким чином: для передачі біта «0» прагнуть, щоб різниця абсолютних значень коефіцієнтів ДКП була більшою за деяку додатну величину, а для передачі біта «1» ця різниця робиться меншою за деяку від'ємну величину:

$$\begin{cases} \left|\Omega_b(\upsilon_1,\nu_1)\right| - \left|\Omega_b(\upsilon_2,\nu_2)\right| > P, \text{ при } m_b = 0; \\ \left|\Omega_b(\upsilon_1,\nu_1)\right| - \left|\Omega_b(\upsilon_2,\nu_2)\right| < -P, \text{ при } m_b = 1. \end{cases} \tag{5.19}$$

Таким чином, первинне зображення спотворюється за рахунок внесення змін до коефіцієнтів ДКП, якщо їхня відносна величина не відповідає приховуваному бітові. Чим більше P, тим більш стійкою до компресії є стеганосистема, але якість зображення при цьому значно погіршується. Після відповідного корегування коефіцієнтів, які повинні задовольняти нерівності (5.19), проводиться обернене ДКП.

Для видобування даних в декодері виконується та сама процедура обрання коефіцієнтів, а рішення про переданий біт приймається у відповідності до правила:

$$\begin{cases} m_b^* = 0, \text{ при } \left|\Omega_b^*(\upsilon_1,\nu_1)\right| > \left|\Omega_b^*(\upsilon_2,\nu_2)\right|; \\ m_b^* = 1, \text{ при } \left|\Omega_b^*(\upsilon_1,\nu_1)\right| < \left|\Omega_b^*(\upsilon_2,\nu_2)\right|. \end{cases} \tag{5.20}$$

1) Виділяємо масиви колірних компонентів зображення:

R := **READ_RED**("C.bmp"); **G** := **READ_GREEN**("C.bmp"); **B** := **READ_BLUE**("C.bmp").

2) У зв'язку з низькою чутливістю ЗСЛ до каналу синього кольору і можливим за певних обставин досить значним спотворенням контейнера при вбудовуванні, секретне повідомлення вбудовуватимемо до масиву **B**. Нехай, як і у попередніх випадках, повідомлення має вигляд **M** := "© Пузиренко О.Ю., 2005 р.".

3) Визначимо розмірність масиву **B** та задамо розмірність сегментів (блоків), на які він розбиватиметься: кількість рядків **X** := **rows**(**B**), **X** = 128; кількість стовпців **Y** := **cols**(**B**), **Y** = 128; розмірність сегментів **N** := 8 пікселів. Загальна кількість сегментів, на яку розбивається зображення: $N_C := X \cdot Y / N^2$, $N_C = 256$ сегментів.

Розбивку масиву **B** на сегменти **C** проводимо за допомогою модуля (М.46).

```
C := | c1 ← 1
     | c2 ← N
     | for b ∈ 1 .. N_C
     |   | r1 ← mod[N·(b − 1) + 1 , X]
     |   | r2 ← r1 + N − 1
     |   | C_b ← submatrix(B , r1 , r2 , c1 , c2)
     |   | if r2 = X
     |   |   | c1 ← c1 + N
     |   |   | c2 ← c2 + N
     | C
```
(М.46)

4) Кожен сегмент **C** призначено для приховування одного біта конфіденційного повідомлення **M**. Тому попередньо необхідно перевірити достатність кількості сегментів для цієї операції.

Кількість біт у повідомленні, яке приховується (один символ повідомлення кодується вісімкою бітів): $L_M := 8 \cdot \text{strlen}(M) = 200$ біт $< N_C = 256$. Отже, об'єм повідомлення має прийнятний для вбудовування обсяг.

5) Застосуємо до кожного з сегментів пряме дискретне косинусне перетворення – див. вираз (5.18 *а*). Програмний модуль прямого ДКП (М.47) складається з двох частин: перша (*а*) визначає значення коефіцієнтів ζ для поточного значення аргументу χ, друга (*б*) проводить обчислення спектральних коефіцієнтів ДКП для кожного сегменту C_b, при цьому одержуються відповідна матриця розмірністю $N \times N$. Коефіцієнт у лівому верхньому куті матриці Ω_b (нагадаємо, що в нашому випадку нумерація елементів масивів починається з одиниці) – $(\Omega_b)_{1,1}$ містить інформацію про яскравість всього сегмента (його зазвичай називають *DC-коефіцієнтом*). Інші коефіцієнти називаються *AC-коефіцієнтами*. Причому, коефіцієнти НЧ компонентів розташовані ближче до лівого верхнього кута, а ВЧ компонентів – ближче до правого нижнього кута (рис.5.27).

$$\zeta(\chi) := \begin{cases} \frac{1}{\sqrt{2}} & \text{if } \chi = 0 \\ 1 & \text{if } \chi > 0 \end{cases} \tag{М.47 \textit{а}}$$

$$\Omega := \left| \begin{array}{l} \text{for } b \in 1..N_C \\ \quad \text{for } \upsilon \in 0..N-1 \\ \quad\quad \text{for } \nu \in 0..N-1 \\ \quad\quad\quad \Omega'_{\upsilon+1,\nu+1} \leftarrow \dfrac{\zeta(\upsilon)\cdot\zeta(\nu)}{\sqrt{2\cdot N}} \times \\ \quad\quad\quad\quad \times \sum_{x=0}^{N-1}\sum_{y=0}^{N-1}\left[(C_b)_{x+1,y+1}\cdot\cos\left[\dfrac{\pi\cdot\upsilon\cdot(2\cdot x+1)}{2\cdot N}\right]\cdot\cos\left[\dfrac{\pi\cdot\nu\cdot(2\cdot y+1)}{2\cdot N}\right]\right] \\ \quad \Omega_b \leftarrow \Omega' \\ \Omega \end{array} \right. \qquad \text{(М.47 } б\text{)}$$

Як зазначалося у попередньому розділі, НЧ компоненти містять переважну частину енергії зображення і, отже, носять шумовий характер. ВЧ компоненти більше піддаються впливові з боку різних алгоритмів обробки (див. рис.5.25). Таким чином, для вбудовування повідомлення більш оптимальними є середньочастотні компоненти спектру зображення.

6) Задаємо позиції двох коефіцієнтів ДКП в масиві Ω_b, які використовуватимуться при вбудовуванні і видобуванні повідомлення до/з контейнера. Ці два коефіцієнти повинні відповідати косинус-функціям із середніми частотами, що забезпечить прихованість інформації в суттєвих областях сигналу, і зробить її стійкішою до JPEG-компресії. Для створення більш захищеної від зламу стеганосистеми зазначена пара коефіцієнтів ДКП повинна обиратися псевдовипадково, звичайно, з множини середньочастотних коефіцієнтів.

Нехай перший коефіцієнт визначається координатами ($\upsilon_1 := 4$; $\nu_1 := 5$), а другий – ($\upsilon_2 := 5$; $\nu_2 := 4$).

Також встановлюємо значення порогу, з яким порівнюватимуться різниці модулів коефіцієнтів. Нехай $P := 25$.

	1	2	3	4	5	6	7	8
1	–603	203	11	45	–30	–14	–14	–7
2	–108	–93	10	49	27	6	8	2
3	–42	–20	–6	16	17	9	3	3
4	56	69	7	–25	–10	–5	–2	–2
5	–33	–21	17	8	3	–4	–5	–3
6	–16	–14	8	2	–4	–2	1	1
7	0	–5	–6	–1	2	3	1	1
8	9	5	–6	–9	0	3	3	2

– НЧ компоненти;

– СЧ компоненти;

– ВЧ компоненти.

Рис.5.27. Приклад масиву Ω коефіцієнтів ДКП

7) Вбудовування інформації проводитимемо у відповідності до рекомендацій, даних перед виразом (5.19): для передачі біта «0» необхідно, щоб різниця модулів коефіцієнтів ДКП була більшою за величину **P**, а для передачі біта «1» ця різниця повинна бути меншою за **–P**. Даний принцип покладено в основу модуля (М.48). На початку модуля масиву Ω_M присвоюється значення масиву Ω. Рядок символів **M** перетворюється на вектор їх ASCII-кодів. У циклі зміни μ (перебирання кодів символів) формат коду кожного символу перетворюється з десяткового на двійковий. Кожен з вісімки отриманих при цьому бітів приховується у окремому сегменті зображення шляхом модифікації значень коефіцієнтів ДКП відповідного сегменту. Для зменшення помітності вбудовування, замість односторонньої зміни величини одного з двох коефіцієнтів ДКП для задоволення вимог (5.19), можна змінювати їх одночасно у протилежних напрямках: наприклад, при вбудовуванні «0» – збільшувати модуль коефіцієнту з координатами (υ_1, ν_1) і одночасно зменшувати модуль коефіцієнту з координатами (υ_2, ν_2).

$$
\begin{aligned}
\Omega_M := \;& \Omega_M \leftarrow \Omega \\
& M \leftarrow \mathrm{str2vec}(M) \\
& b \leftarrow 1 \\
& \text{for } \mu \in 1..\mathrm{rows}(M) \\
& \quad m \leftarrow \mathrm{D2B}\left(M_\mu\right) \\
& \quad \text{for } i \in 1..8 \\
& \quad\quad \Omega' \leftarrow \Omega_{i+N\cdot(\mu-1)} \\
& \quad\quad \omega_1 \leftarrow \left|\Omega'_{\upsilon_1,\nu_1}\right| \\
& \quad\quad \omega_2 \leftarrow \left|\Omega'_{\upsilon_2,\nu_2}\right| \\
& \quad\quad z_1 \leftarrow 1 \;\text{ if }\; \Omega'_{\upsilon_1,\nu_1} \ge 0 \\
& \quad\quad z_1 \leftarrow -1 \;\text{ if }\; \Omega'_{\upsilon_1,\nu_1} < 0 \\
& \quad\quad z_2 \leftarrow 1 \;\text{ if }\; \Omega'_{\upsilon_2,\nu_2} \ge 0 \\
& \quad\quad z_2 \leftarrow -1 \;\text{ if }\; \Omega'_{\upsilon_2,\nu_2} < 0 \\
& \quad\quad \omega_1 \leftarrow P + \omega_2 + 1 \;\text{ if }\; \omega_1 - \omega_2 \le P \;\text{ if }\; m_i = 0 \\
& \quad\quad \omega_2 \leftarrow P + \omega_1 + 1 \;\text{ if }\; \omega_1 - \omega_2 \ge -P \;\text{ if }\; m_i = 1 \\
& \quad\quad \Omega'_{\upsilon_1,\nu_1} \leftarrow z_1\cdot\omega_1 \\
& \quad\quad \Omega'_{\upsilon_2,\nu_2} \leftarrow z_2\cdot\omega_2 \\
& \quad\quad \Omega_{M_b} \leftarrow \Omega' \\
& \quad\quad b \leftarrow b + 1 \\
& \Omega_M
\end{aligned}
\tag{M.48}
$$

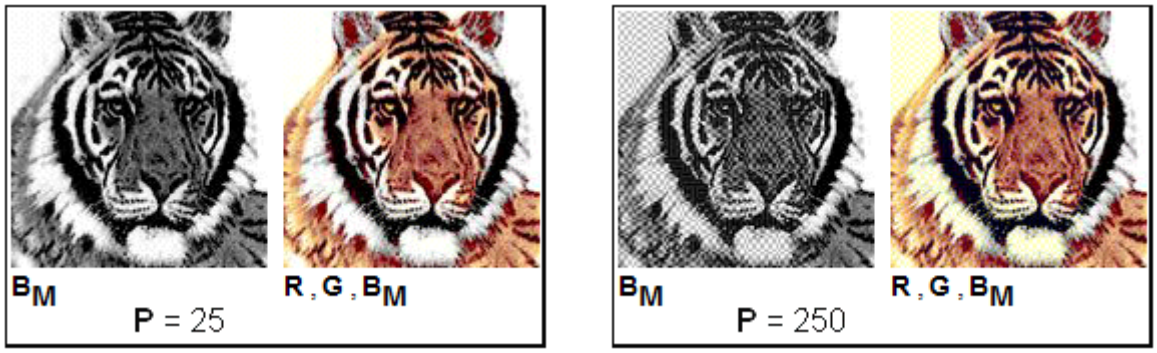
8) Виконуємо обернене ДКП (М.49) відповідно до формули (5.18 *б*).

$$
\begin{aligned}
C_M := \;& \text{for } b \in 1..N_C \\
& \quad \text{for } x \in 0..N-1 \\
& \quad\quad \text{for } y \in 0..N-1 \\
& \quad\quad\quad C'_{M_{x+1,y+1}} \leftarrow \frac{1}{\sqrt{2\cdot N}}\cdot\sum_{\upsilon=0}^{N-1}\sum_{\nu=0}^{N-1}\left[\zeta(\upsilon)\cdot\zeta(\nu)\cdot\left(\Omega_{M_b}\right)_{\upsilon+1,\nu+1}\times\right. \\
& \quad\quad\quad\quad \left.\times\cos\left[\frac{\pi\cdot\upsilon\cdot(2\cdot x+1)}{2\cdot N}\right]\cdot\cos\left[\frac{\pi\cdot\nu\cdot(2\cdot y+1)}{2\cdot N}\right]\right] \\
& \quad C_{M_b} \leftarrow C'_M \\
& C_M
\end{aligned}
\tag{M.49}
$$

9) Проводимо збирання сегментів до масиву, використовуючи програмний модуль (М.50). Наприкінці модуля проводиться нормування значень інтенсивностей пікселів зображення, оскільки модифікування коефіцієнтів ДКП у деяких випадках може призвести до виходу вказаних значень за припустимі межі [0, 255].

Відтворюємо масив відновленого зображення графічно (рис.5.28 *а*). Також на рис.5.28 зображено контейнер-результат, відновлений за колірними складовими **R-G-B$_M$**. Крім того наведено вигляд, якого набувають масив синього кольору і контейнер-результат у випадку встановлення значення порогу **P** := 250 за незмінних значень (υ_1; ν_1) і (υ_2; ν_2) (рис.5.28 *б*).

$$
\begin{aligned}
&B_M := \Big| \; B_M \leftarrow C_{M_1} \\
&\quad \text{for } b \in 2 .. X \div N \\
&\quad\quad B_M \leftarrow \text{stack}\left(B_M , C_{M_b}\right) \\
&\quad B'_M \leftarrow 0 \\
&\quad \text{for } b \in X \div N + 1 .. N_C \\
&\quad\quad \Big| \; B'_M \leftarrow C_{M_b} \quad \text{if } B'_M = 0 \\
&\quad\quad \Big| \; B'_M \leftarrow \text{stack}\left(B'_M , C_{M_b}\right) \quad \text{otherwise} \\
&\quad\quad \Big| \; \text{if } \text{mod}(b , X \div N) = 0 \\
&\quad\quad\quad \Big| \; B_M \leftarrow \text{augment}\left(B_M , B'_M\right) \\
&\quad\quad\quad \Big| \; B'_M \leftarrow 0 \\
&\quad B_M \leftarrow \frac{B_M + \left|\min\left(B_M\right)\right|}{\max\left(B_M + \left|\min\left(B_M\right)\right|\right)} \cdot 255
\end{aligned}
\qquad \text{(M.50)}
$$



а) *б)*

Рис.5.28. Результати вбудовування секретного повідомлення **M** до контейнера шляхом модифікації коефіцієнтів ДКП сегментів масиву каналу синього при різних значеннях порогу **P**

10) Розглянемо процес *видобування повідомлення* на приймальній стороні.

Одержувачеві повинні бути відомими: алгоритм приховання, колірний масив-контейнер (**B***) і його розмірність (**X*** та **Y***), розмірність сегментів (**N***) та матричні координати коефіцієнтів косинусних функцій, які використовувалися для приховання [(υ^*_1; ν^*_1) та (υ^*_2; ν^*_2)].

Проводиться обчислення загальної кількості сегментів у зображенні-контейнері: $N^*_C := X^* \cdot Y^* / N^{*2}$.

11) Розбивка масиву **B*** на сегменти виконується за модулем, аналогічним за будовою модулю (М.46). В результаті одержується масив **C***, кожен елемент якого становить собою матрицю розмірністю **N***×**N***.

12) До кожного сегмента застосовується пряме ДКП (див. модуль (М.47)), при цьому отримується масив **Ω*** коефіцієнтів ДКП кожного окремого сегмента **C***.

13) Модулем (М.51) здійснюється видобування прихованої інформації. До основи модуля покладено перевірку співвідношень (5.20), за результатами якої біту повідомлення, що видобувається, присвоюється значення 0 або 1. Після видобування кожного восьмого біту одержаний вектор (**m***) перетворюється з масиву двійкових даних на відповідне десяткове число, яке заноситься до визначеного змінною **j** елементу вектора **M***. Результат видобування в наприкінці модуля трансформується з вектору ASCII-кодів на символьний рядок.

14) Результати обчислення показників візуального спотворення при **P** = 0.5 і **P** = 25 зведені до табл.5.4.

```
M* := | i ← 1
      | j ← 1
      | for μ ∈ 1 .. N*C
      |   Ω' ← Ω*μ
      |   ω1 ← |Ω'υ*1, ν*1|
      |   ω2 ← |Ω'υ*2, ν*2|
      |   m*i ← 0  if ω1 > ω2
      |   m*i ← 1  if ω1 < ω2
      |   i ← i + 1  if rows(m*) < 8
      |   if rows(m*) = 8
      |     | i ← 1
      |     | M*j ← B2D(m*)
      |     | m* ← 0
      |     | j ← j + 1
      | vec2str(M*)
```
(M.51)

5.3.3.2. Метод Бенгама-Мемона-Ео-Юнг

D. Benham, N. Memon, B.-L. Yeo, та *Minerva Yeung* [103] запропонували оптимізовану версію попереднього методу. Причому оптимізація була проведена ними за двома напрямками: по-перше, було запропоновано для вбудовування використовувати не всі блоки, а лише найбільш придатні для цього, по-друге, у частотній області блоку для вбудовування обираються не два, а три коефіцієнти ДКП, що, як буде показано нижче, суттєво зменшує спотворення контейнера. Розглянемо зазначені удосконалення більш докладно.

Придатними для вбудовування інформації вважаються блоки зображення, які одночасно задовольняють наступним двом вимогам: 1) блоки не повинні мати різких переходів яскравості; 2) не повинні бути занадто монотонними. Блоки, що не відповідають першій вимозі, характеризуються наявністю декількох занадто великих значень низькочастотних коефіцієнтів ДКП, порівнянних за своєю величиною з DC-коефіцієнтом. Для блоків, що не задовольняють другій вимозі, є характерною рівність нулеві більшості високочастотних коефіцієнтів. Вказані особливості являються критерієм відбраковування непридатних блоків.

Зазначені вимоги відбраковування врахуємо використанням двох порогових коефіцієнтів: $\boldsymbol{P}_{\mathbf{L}}$ (для першої вимоги) і $\boldsymbol{P}_{\mathbf{H}}$ (для другої вимоги), перевищення ($\boldsymbol{P}_{\mathbf{L}}$) або недосягнення ($\boldsymbol{P}_{\mathbf{H}}$) яких вказуватиме, що блок не придатний для модифікацій у частотній області.

Вбудовування біту повідомлення до блоку здійснюється наступним чином. Обираються (для більшої стійкості стеганосистеми – псевдовипадково) три коефіцієнти ДКП блоку з середньочастотної області з координатами (υ_1, ν_1), (υ_2, ν_2) та (υ_3, ν_3). Якщо необхідно провести вбудовування «0», ці коефіцієнти змінюються таким чином (звичайно, якщо це необхідно), щоб третій коефіцієнт став менше кожного з перших двох; якщо потрібно приховати «1», він робиться більшим за інші:

$$\begin{cases} \left.\begin{aligned} &\left|\boldsymbol{\Omega}_{\boldsymbol{b}}(\upsilon_3,\nu_3)\right| < \left|\boldsymbol{\Omega}_{\boldsymbol{b}}(\upsilon_1,\nu_1)\right|; \\ &\left|\boldsymbol{\Omega}_{\boldsymbol{b}}(\upsilon_3,\nu_3)\right| < \left|\boldsymbol{\Omega}_{\boldsymbol{b}}(\upsilon_2,\nu_2)\right|. \end{aligned}\right\}, \text{ при } \boldsymbol{m}_{\boldsymbol{b}} = 0; \\ \left.\begin{aligned} &\left|\boldsymbol{\Omega}_{\boldsymbol{b}}(\upsilon_3,\nu_3)\right| > \left|\boldsymbol{\Omega}_{\boldsymbol{b}}(\upsilon_1,\nu_1)\right|; \\ &\left|\boldsymbol{\Omega}_{\boldsymbol{b}}(\upsilon_3,\nu_3)\right| > \left|\boldsymbol{\Omega}_{\boldsymbol{b}}(\upsilon_2,\nu_2)\right|. \end{aligned}\right\}, \text{ при } \boldsymbol{m}_{\boldsymbol{b}} = 1. \end{cases} \tag{5.21}$$

Як і у попередньому методі, для встановлення достатності розрізнення зазначених коефіцієнтів ДКП, введемо до виразу (5.21) значення порогу розрізнення $\boldsymbol{P}$:

$$\begin{cases} \left|\boldsymbol{\Omega}_{\boldsymbol{b}}(\upsilon_3,\nu_3)\right| < \min\left(\left|\boldsymbol{\Omega}_{\boldsymbol{b}}(\upsilon_1,\nu_1)\right|, \left|\boldsymbol{\Omega}_{\boldsymbol{b}}(\upsilon_2,\nu_2)\right|\right) - \boldsymbol{P}, \text{ при } \boldsymbol{m}_{\boldsymbol{b}} = 0; \\ \left|\boldsymbol{\Omega}_{\boldsymbol{b}}(\upsilon_3,\nu_3)\right| > \max\left(\left|\boldsymbol{\Omega}_{\boldsymbol{b}}(\upsilon_1,\nu_1)\right|, \left|\boldsymbol{\Omega}_{\boldsymbol{b}}(\upsilon_2,\nu_2)\right|\right) + \boldsymbol{P}, \text{ при } \boldsymbol{m}_{\boldsymbol{b}} = 1. \end{cases} \tag{5.22}$$

У випадку, якщо така модифікація призводить до занадто великої деградації зображення, коефіцієнти не змінюють, тобто блок не використовується в якості контейнера.

Використання трьох коефіцієнтів замість двох і, що найголовніше, відмова від модифікації у випадку неприйнятних спотворювань зображення зменшує похибки, що вносяться повідомленням. Одержувач завжди може визначити блоки, до яких не проводилося вбудовування, повторивши аналіз, аналогічний виконаному на передавальній стороні.

Наведемо можливий варіант реалізації даного методу у програмі MathCAD.

1) Вихідні дані, програмні модулі розбиття масиву контейнера на блоки та проведення прямого ДКП аналогічні використаним при моделюванні попереднього методу ((М.46), (М.47)).

2) Задаємо координати трьох коефіцієнтів ДКП у масиві Ω_b, які будемо використовувати для вбудовування і видобування бітів повідомлення до/з контейнера. Наприклад, ($\upsilon_1 := 7; \nu_1 := 3$), ($\upsilon_2 := 5; \nu_2 := 5$), ($\upsilon_3 := 3; \nu_3 := 7$).

Встановлюємо значення порогів відбраковування блоків. Нехай $P_L := 2600$, $P_H := 40$. Також задаємо поріг розрізнення $P := 50$.

3) Вбудовування повідомлення **M** до блоків контейнера виконуємо, використовуючи програмний модуль (М.52). На початку обчислення масиву Ω_M присвоюється значення немодифікованого масиву коефіцієнтів ДКП для всіх блоків зображення – Ω. Після перетворення формату повідомлення з рядка символів на вектор двійкових даних розмірністю $L_M \times 1$ відбувається безпосередньо модифікація оптимальних для вбудовування блоків. Змінна-лічильник **w** перед початком циклу перебирання елементів вектора M_{vec_bin} приймає нульове значення. На початку циклу здійснюється перевірка умови невиходу змінної **w** за межі загальної кількості сегментів N_C. Якщо ж $w > N_C$, цикл переривається після присвоєння змінній Ω_M рядка символів, який інформує відправника про кількість вбудованих біт (при поточних значеннях порогів P_L і P_H) і рекомендує розширити границі (тобто збільшити значення P_L і/або зменшити P_H), що призведе до зростання кількості блоків, які відповідатимуть поставленим вимогам. Звичайно, такий крок спричинить появу у списку блоків, обраних для вбудовування, не зовсім оптимальних для цієї операції. Тому в ході домовленості між сторонами прихованого обміну щодо алгоритму псевдовипадкового вибору трьох пар координат коефіцієнтів ДКП та значень порогів відбраковування обов'язково повинна бути присутньою фаза перевірки обраного в якості контейнера зображення (набору зображень) на достатність його пропускної здатності. Іншими словами, після встановлення певних значень порогів P_L і P_H, обчислюється кількість блоків, визнаних придатними для вбудовування, та виконується оцінка візуального спотворення контейнера. За одержаними результатами приймається рішення про достатність обраних значень порогів або ж про необхідність їх зміни.

Встановлені вище параметри дозволяють визнати придатними для вбудовування 201 блок у обраному нами контейнері. Наведемо результати обчислення кількості придатних блоків $N_{C_{opt.}}$ в залежності від параметрів P_L і P_H (рис.5.29).

Повернемося до модуля (М.52). Циклом зміни параметру **c** здійснюється вибір матриці коефіцієнтів ДКП для **c**-го блоку. Для обраної матриці підраховується сума модулів коефіцієнтів, що відповідають низькочастотній (Σ_{LF}) і високочастотній (Σ_{HF}) ділянкам масиву Ω' (див. рис.5.27). До уваги не береться лише DC-коефіцієнт (умова ($\upsilon + \nu > 2$)). Якщо значення одержаних сум задовольняють поставленим вимогам змінній **w** присвоюється номер блоку, визнаного придатним для вбудовування, цикл зміни **c** переривається.

Проводиться зчитування коефіцієнтів масиву Ω' (на якому відбулася зупинка), які відповідають координатам ($\upsilon_1; \nu_1$), ($\upsilon_2; \nu_2$), ($\upsilon_3; \nu_3$). Якщо третій коефіцієнт не відповідає вимогам (5.22), проводиться модифікація двох коефіцієнтів з трьох – власне третього коефіцієнту і коефіцієнту, який найбільше відрізняється від третього. Заміна відбувається таким чином, щоб різниця між даними коефіцієнтами дорівнювала порогу розрізнення **P**. Можлива й одностороння модифікація лише третього коефіцієнту (додатково зображено на рисунку для модуля (М.52)), але у цьому випадку порушення структури зображення будуть більш значними.

Модифіковані значення коефіцієнтів замінюють відповідні за координатами значення у масиві Ω', який, у свою чергу, присвоюється **w**-му елементу результуючого масиву Ω_M.

$\Omega_M :=$ | $\Omega_M \leftarrow \Omega$

Ⓘ -- див. (М.22), крім $S \leftarrow C$

$w \leftarrow 0$

for $j \in 1 .. L_M$

$w \leftarrow w + 1$

if $w > N_C$

$\Omega_M \leftarrow$ **concat**("Вбудовано", **num2str**$(j-1)$, " біт. ", "Розширьте границі")

break

for $c \in w .. N_C$

$\Omega' \leftarrow \Omega_c$

$\Sigma_{LF} \leftarrow 0$

$v\$ \leftarrow N - 1$

for $\upsilon \in 1 .. N-1$

for $\nu \in 1 .. v\$$

$\Sigma_{LF} \leftarrow \Sigma_{LF} + |\Omega'_{\upsilon,\nu}|$ **if** $(\upsilon + \nu) > 2$

$v\$ \leftarrow v\$ - 1$ **if** $\nu = v\$$

$\Sigma_{HF} \leftarrow 0$

$v\$ \leftarrow N$

for $\upsilon \in 3 .. N$

for $\nu \in v\$ .. N$

$\Sigma_{HF} \leftarrow \Sigma_{HF} + |\Omega'_{\upsilon,\nu}|$

$v\$ \leftarrow v\$ - 1$ **if** $\nu = v\$$

if $\Sigma_{LF} < P_L \wedge \Sigma_{HF} > P_H$

$w \leftarrow c$

break

$\omega_1 \leftarrow \Omega'_{\upsilon_1,\nu_1}$

$\omega_2 \leftarrow \Omega'_{\upsilon_2,\nu_2}$

$\omega_3 \leftarrow \Omega'_{\upsilon_3,\nu_3}$

if $\omega_3 > \omega_1 \vee \omega_3 > \omega_2$ **if** $M_{vec\_bin_j} = 0$

$\omega_{min} \leftarrow \min\left(\begin{pmatrix}\omega_1\\ \omega_2\end{pmatrix}\right)$

$\omega_3 \leftarrow \omega_{min} - \frac{P}{2}$

$\omega_1 \leftarrow \omega_1 + \frac{P}{2}$ **if** $\omega_1 = \omega_{min}$ $\Big)$ $\underline{\omega_3 \leftarrow \omega_{min} - P}$

$\omega_2 \leftarrow \omega_2 + \frac{P}{2}$ **if** $\omega_2 = \omega_{min}$

if $\omega_3 < \omega_1 \vee \omega_3 < \omega_2$ **if** $M_{vec\_bin_j} = 1$

$\omega_{max} \leftarrow \max\left(\begin{pmatrix}\omega_1\\ \omega_2\end{pmatrix}\right)$

$\omega_3 \leftarrow \omega_{max} + \frac{P}{2}$

$\omega_1 \leftarrow \omega_1 - \frac{P}{2}$ **if** $\omega_1 = \omega_{max}$ $\Big)$ $\underline{\omega_3 \leftarrow \omega_{max} + P}$

$\omega_2 \leftarrow \omega_2 - \frac{P}{2}$ **if** $\omega_2 = \omega_{max}$

$\Omega'_{\upsilon_1,\nu_1} \leftarrow \omega_1$

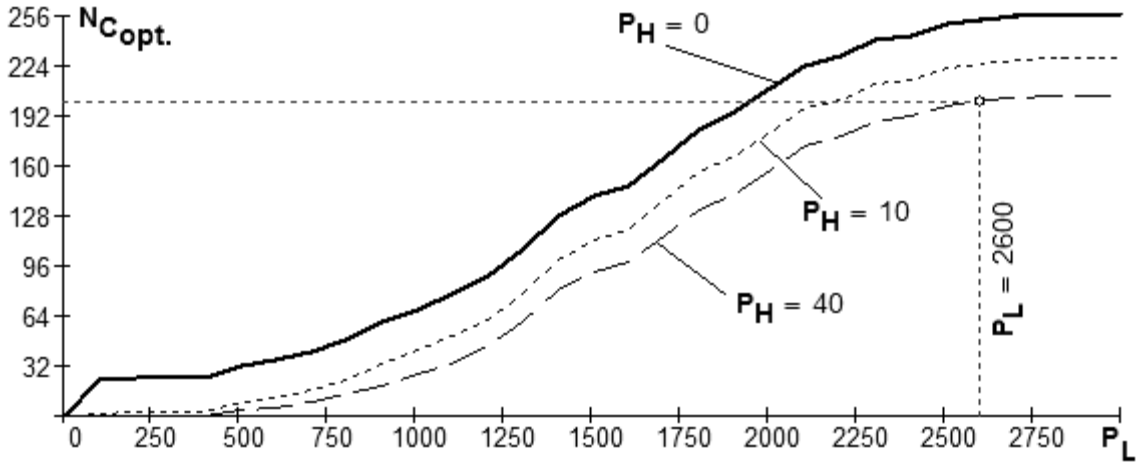
$\Omega'_{\upsilon_2,\nu_2} \leftarrow \omega_2$

$\Omega'_{\upsilon_3,\nu_3} \leftarrow \omega_3$

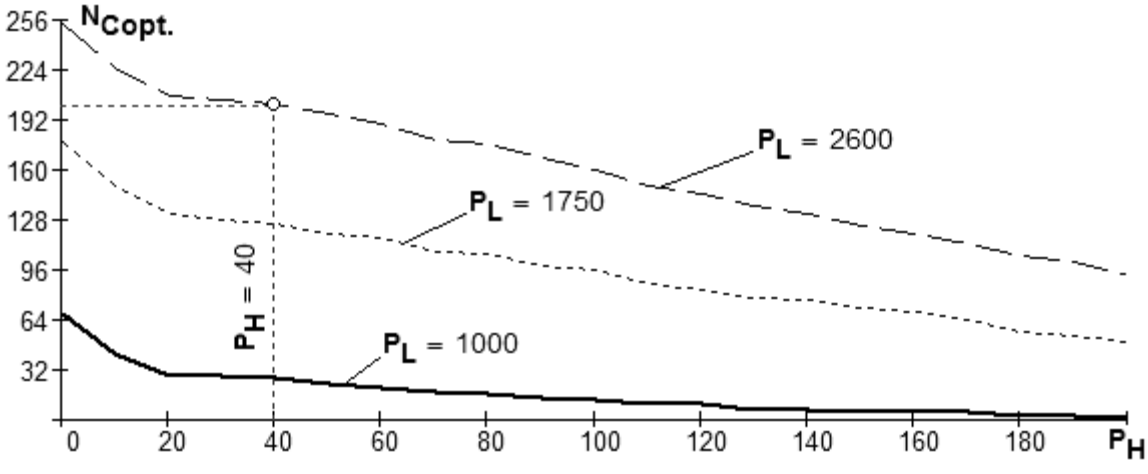
$\Omega_{M_w} \leftarrow \Omega'$

Ω_M

(М.52)



а)



б)

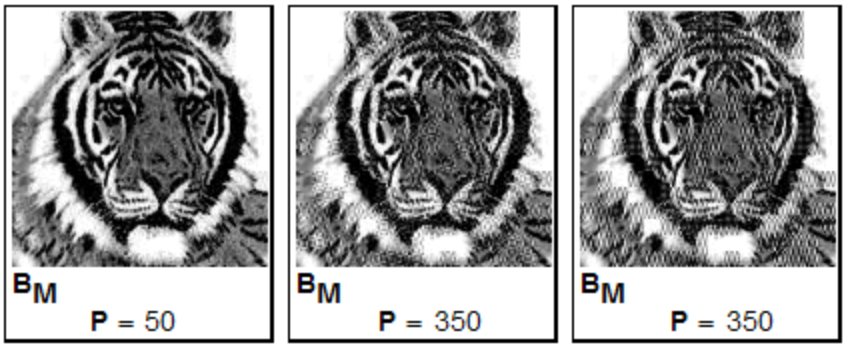
Рис.5.29. Залежність кількості придатних для вбудовування блоків від значень порогів P_L і P_H: P_L = *var*, P_H = *const* (*а*); P_L = *const*, P_H = *var* (*б*)

4) Використовуючи модуль (М.53), проводимо обернене дискретно-косинусне перетворення.

$$
\begin{aligned}
&C_M := \Big| \ \textbf{for}\ \ b \in 1..N_C \\
&\quad \Big| \ \textbf{for}\ \ x \in 0..N-1 \\
&\quad\quad \textbf{for}\ \ y \in 0..N-1 \\
&\quad\quad\quad C'M_{x+1,y+1} \leftarrow \left| \frac{1}{\sqrt{2\cdot N}} \cdot \sum_{\upsilon=0}^{N-1} \sum_{\nu=0}^{N-1} \left[\zeta(\upsilon)\cdot\zeta(\nu)\cdot\left(\Omega M_b\right)_{\upsilon+1,\nu+1} \times \right.\right. \\
&\quad\quad\quad\quad \left.\left. \times \cos\left[\frac{\pi\cdot\upsilon\cdot(2\cdot x+1)}{2\cdot N}\right]\cdot\cos\left[\frac{\pi\cdot\nu\cdot(2\cdot y+1)}{2\cdot N}\right]\right] \right| \\
&\quad\quad C_{M_b} \leftarrow \frac{C'_M}{\max(C'_M)}\cdot 255 \quad \textbf{if}\ \ \max(C'_M) > 255 \\
&\quad\quad C_{M_b} \leftarrow C'_M \quad \textbf{otherwise} \\
&\quad C_M
\end{aligned}
\qquad \text{(М.53)}
$$

При цьому, як і у попередньо розглянутому методі, є можливим вихід значень інтенсивностей пікселів за припустимі межі. Нормування, аналогічне використаному у (М.50) недоцільне через те, що нормуватиметься весь масив контейнера і при видобуванні виникне висока імовірність помилкового відбраковування (або, навпаки, обрання) блоків, в яких передбачається вбудованим біт інформації. Тому пропонується провести нормування безпосередньо результатів зворотного ДКП окремих блоків, а саме таких, у яких максимальне абсолютне значення будь-якого елементу перевищує 255. В іншому випадку елементам блоку контейнера повертається лише їх абсолютне значення.

5) Збирання блоків до спільного зображення виконуємо програмним модулем (М.50), але вже без нормування значень елементів контейнера на завершальному етапі. Графічне представлення відновленого масиву синьої складової при вказаних вище початкових даних наведено на рис.5.30 *а*. На рис.5.30 *б* представлено вигляд, якого набуває даний масив у випадку встановлення значення порогу розрізнення $\mathbf{P} := 350$ за незмінних інших значень (модифікуються один з перших двох коефіцієнтів і третій). На рис.5.30 *в* зображено результат односторонньої модифікації третього коефіцієнту ДКП (див. (М.52)) за незмінних інших даних.



а) *б)* *в)*

Рис.5.30. Результати вбудовування повідомлення **M** до масиву каналу синього при обранні оптимальних для вбудовування блоків та різних значеннях порогу **P**

Порівнюючи рис.5.30 з рис.5.28 можна зробити очевидний висновок про перевагу методу Бенгама та ін. над методом Коха і Жао, оскільки перший дозволяє ефективно відібрати саме ті блоки зображення, вбудовування до яких буде якнайменш помітним.

6) Процес *видобування повідомлення* повторює аналіз, виконаний під час його вбудовування. Попередньо повинні бути відомими: алгоритм приховання, масив-контейнер (**B***) і його розмірність (**X***, **Y***), розмірність сегментів (**N***) та координати коефіцієнтів ДКП, які використовувалися під час приховування (або алгоритм їх одержання): (υ^*_1; ν^*_1), (υ^*_2; ν^*_2) і (υ^*_3; ν^*_3). Проводиться обчислення загальної кількості сегментів у зображенні-контейнері ($\mathbf{N}^*_\mathbf{C}$).

7) Розбивка масиву **B*** на сегменти $\mathbf{C}^*_\mathbf{b}$ виконується програмним модулем (М.46).

8) До кожного сегмента застосовується пряме ДКП (М.47), результатом чого є масив **Ω*** коефіцієнтів ДКП сегментів $\mathbf{C}^*_\mathbf{b}$.

9) Видобування прихованої інформації здійснюється програмним модулем (М.53), до основи якого покладено перевірку блоку на придатність до вбудовування, на підставі чого робиться висновок про те, чи було обрано цей блок передавальною стороною в якості контейнера для біту повідомлення. При позитивному рішенні виконується перевірка співвідношень (5.21), за результатами якої формується вектор двійкових даних, що в подальшому перетворюється на рядок символів (аналогічно тому, як це здійснювалося у (М.51)).

10) Результати обчислення показників візуального спотворення при **P** = 1 і **P** = 35 зведені до табл.5.4.

$$
\begin{aligned}
\mathbf{M^*} := \;& \mathbf{i} \leftarrow 1 \\
& \mathbf{j} \leftarrow 1 \\
& \mathbf{w} \leftarrow 0 \\
& \textbf{for}\ \ \mu \in 1 .. \mathbf{N^*_C} \\
& \quad \mathbf{w} \leftarrow \mathbf{w} + 1 \\
& \quad \textbf{for}\ \ \mathbf{c} \in \mathbf{w} .. \mathbf{N^*_C} \\
& \quad\quad \textbf{break}\ \ \textbf{if}\ \ \mathbf{w} > \mathbf{N_C} \\
& \quad\quad \Omega' \leftarrow \Omega^*_{\mathbf{c}} \\
& \quad\quad \Sigma_{\mathbf{LF}} \leftarrow 0 \\
& \quad\quad \mathbf{v\$} \leftarrow \mathbf{N^*} - 1 \\
& \quad\quad \textbf{for}\ \ \upsilon \in 1 .. \mathbf{N^*} - 1 \\
& \quad\quad\quad \textbf{for}\ \ \nu \in 1 .. \mathbf{v\$} \\
& \quad\quad\quad\quad \Sigma_{\mathbf{LF}} \leftarrow \Sigma_{\mathbf{LF}} + \left|\Omega'_{\upsilon,\nu}\right| \ \ \textbf{if}\ \ (\upsilon + \nu) > 2 \\
& \quad\quad\quad\quad \mathbf{v\$} \leftarrow \mathbf{v\$} - 1 \ \ \textbf{if}\ \ \nu = \mathbf{v\$} \\
& \quad\quad \Sigma_{\mathbf{HF}} \leftarrow 0 \\
& \quad\quad \mathbf{v\$} \leftarrow \mathbf{N^*} \\
& \quad\quad \textbf{for}\ \ \upsilon \in 3 .. \mathbf{N^*} \\
& \quad\quad\quad \textbf{for}\ \ \nu \in \mathbf{v\$} .. \mathbf{N^*} \\
& \quad\quad\quad\quad \Sigma_{\mathbf{HF}} \leftarrow \Sigma_{\mathbf{HF}} + \left|\Omega'_{\upsilon,\nu}\right| \\
& \quad\quad\quad\quad \mathbf{v\$} \leftarrow \mathbf{v\$} - 1 \ \ \textbf{if}\ \ \nu = \mathbf{v\$} \\
& \quad\quad \textbf{if}\ \ \Sigma_{\mathbf{LF}} < \mathbf{P^*_L} \wedge \Sigma_{\mathbf{HF}} > \mathbf{P^*_H} \\
& \quad\quad\quad \mathbf{w} \leftarrow \mathbf{c} \\
& \quad\quad\quad \textbf{break} \\
& \quad \varpi_1 \leftarrow \Omega'_{\upsilon^*_1,\nu^*_1} \\
& \quad \varpi_2 \leftarrow \Omega'_{\upsilon^*_2,\nu^*_2} \\
& \quad \varpi_3 \leftarrow \Omega'_{\upsilon^*_3,\nu^*_3} \\
& \quad \varpi_{\min} \leftarrow \min\left(\begin{pmatrix}\varpi_1 \\ \varpi_2\end{pmatrix}\right) \\
& \quad \varpi_{\max} \leftarrow \max\left(\begin{pmatrix}\varpi_1 \\ \varpi_2\end{pmatrix}\right) \\
& \quad \mathbf{m^*_i} \leftarrow 0 \ \ \textbf{if}\ \ \varpi_3 < \varpi_{\min} \\
& \quad \mathbf{m^*_i} \leftarrow 1 \ \ \textbf{if}\ \ \varpi_3 > \varpi_{\max} \\
& \quad \mathbf{i} \leftarrow \mathbf{i} + 1 \ \ \textbf{if}\ \ \mathbf{rows(m^*)} < 8 \\
& \quad \textbf{if}\ \ \mathbf{rows(m^*)} = 8 \\
& \quad\quad \mathbf{i} \leftarrow 1 \\
& \quad\quad \mathbf{M^*_j} \leftarrow \mathbf{B2D(m^*)} \\
& \quad\quad \mathbf{m^*} \leftarrow 0 \\
& \quad\quad \mathbf{j} \leftarrow \mathbf{j} + 1 \\
& \mathbf{vec2str(M^*)}
\end{aligned}
\tag{M.54}
$$

5.3.3.3. Метод Хсу і Ву

Chiou-Ting Hsu та *Ja-Ling Wu* [104] запропонували свого часу алгоритм вбудовування ЦВЗ до масивів коефіцієнтів ДКП блоків зображення-контейнера. Наведемо основні положення, закладені авторами до основи алгоритму. Нехай $\boldsymbol{C}$ – первинне напівтонове зображення розміром $\boldsymbol{X} \times \boldsymbol{Y}$, а $\boldsymbol{W}$ – ЦВЗ, що являє собою двійкове зображення розміром $\boldsymbol{A} \times \boldsymbol{Z}$. У ЦВЗ піксель може приймати значення або «1», або «0». Тому очевидно, що безпосереднє спостереження такого зображення неможливе, оскільки інтенсивності 0 і 1 відповідають чорному кольору (остання – у

певному наближенні). Зображення ЦВЗ можна створити чорно-білим, а перед приховуванням замінити інтенсивність білих пікселів (255) на 1, наприклад, шляхом ділення всього масиву ЦВЗ на 255. При видобуванні, навпаки, для візуального спостереження масив ЦВЗ необхідно помножити на 255.

Оскільки, як буде показано далі, під час вбудовування ЦВЗ оброблятиметься лише середньочастотний діапазон сигналу-контейнера, необхідною передумовою є те, що розмір ЦВЗ повинен бути меншим за розмір контейнера. Так, для контейнера, розбитого на блоки 8×8, для вбудовування ЦВЗ оптимальним є використання $64 \cdot \boldsymbol{A} \cdot \boldsymbol{Z} / (\boldsymbol{X} \cdot \boldsymbol{Y})$ коефіцієнтів ДКП. Відношення $\boldsymbol{A} \cdot \boldsymbol{Z} / (\boldsymbol{X} \cdot \boldsymbol{Y})$ визначає кількість інформації, яка може бути вбудована до обраного в якості контейнера зображення (у наведеному випадку – до 64-х коефіцієнтів блоку 8×8). Для більшої стійкості та прихованості результатів застосування стеганометоду кількість вбудовуваної інформації намагаються зменшити.

Зображення-контейнер $\boldsymbol{C}$ і ЦВЗ $\boldsymbol{W}$ представимо як

$$\boldsymbol{C} = \{\boldsymbol{c}(\boldsymbol{x}, \boldsymbol{y});\ 1 \le \boldsymbol{x} \le \boldsymbol{X};\ 1 \le \boldsymbol{y} \le \boldsymbol{Y}\}, \tag{5.23}$$

$$\boldsymbol{W} = \{\boldsymbol{w}(\boldsymbol{a}, \boldsymbol{z});\ 1 \le \boldsymbol{a} \le \boldsymbol{A};\ 1 \le \boldsymbol{z} \le \boldsymbol{Z}\}, \tag{5.24}$$

де $\boldsymbol{c}(\boldsymbol{x}, \boldsymbol{y}) \in \{0, ..., 2^{\boldsymbol{L}}-1\}$ – інтенсивність пікселя $(\boldsymbol{x}, \boldsymbol{y})$; $\boldsymbol{L}$ – кількість біт, що використовується для квантування інтенсивностей; $\boldsymbol{w}(\boldsymbol{a}, \boldsymbol{z}) \in \{0, 1\}$ – двійкові значення пікселя $(\boldsymbol{a}, \boldsymbol{z})$ ЦВЗ.

Контейнер $\boldsymbol{C}$ можна розбити на $\frac{\boldsymbol{X}}{8} \times \frac{\boldsymbol{Y}}{8}$ блоків розмірністю 8×8. Для одержання цієї ж кількості блоків, ЦВЗ розбивається на блоки розмірністю $\frac{8 \cdot \boldsymbol{A}}{\boldsymbol{X}} \times \frac{8 \cdot \boldsymbol{Z}}{\boldsymbol{Y}}$. Наприклад, якщо $\boldsymbol{A} = \boldsymbol{X}/2$ та $\boldsymbol{Z} = \boldsymbol{Y}/2$, розмірність блоку ЦВЗ складе 4×4; якщо $\boldsymbol{A} = \boldsymbol{X}/4$ та $\boldsymbol{Z} = \boldsymbol{Y}/4$, – 2×2 і т.п. Для доповнення контейнера або ЦВЗ до необхідної розмірності можуть бути добавлені додаткові стовпці або рядки.

Псевдовипадкова перестановка пікселів ЦВЗ. У деякому наближенні, кожен блок ЦВЗ вбудовується до середньочастотних коефіцієнтів ДКП кожного блоку контейнера шляхом використання блочного перетворення замість перетворення всього контейнера. Тому, замість загального контейнера, кожен блок ЦВЗ буде розсіяний лише по відповідному блоку першого. Очевидно, що за відсутності належного регулювання просторових зв'язків ЦВЗ, звичайне масштабування контейнера може зруйнувати ЦВЗ.

Для забезпечення стійкості до масштабування, з метою зміни порядку ЦВЗ для розосередження його просторових зв'язків, авторами було запропоновано використати швидкий метод генерації двомірного ПВЧ:

$$\boldsymbol{W}_{rnd} = \mathbf{permute}(\boldsymbol{W}); \tag{5.25}$$

$$\boldsymbol{W}_{rnd} = \{\boldsymbol{w}_{rnd}(\boldsymbol{a}, \boldsymbol{z}) = \boldsymbol{w}_{rnd}(\boldsymbol{a}', \boldsymbol{z}');\ 1 \le \boldsymbol{a} \le \boldsymbol{A};\ 1 \le \boldsymbol{z} \le \boldsymbol{Z}\},$$

де піксель $(\boldsymbol{a}', \boldsymbol{z}')$ – являє собою переставлений у відповідності до псевдовипадкової перестановки піксель $(\boldsymbol{a}, \boldsymbol{z})$.

Перестановка блоків ЦВЗ, в залежності від характеристик блоків контейнера. У відповідності до збільшення рівня прихованості, повинні бути враховані характеристики контейнера (наприклад, відомо, що модифікація високих частот або ділянок з більшою яскравістю є менш помітною). Подібні, залежні від контейнера властивості можуть бути використані для перестановки псевдовипадково змішаного ЦВЗ для одержання більшої відповідності чутливості ЗСЛ. Автори пропонують впорядкувати блоки контейнера за зміною дисперсій інтенсивностей пікселів (наприклад, по їх зменшенню). У свою чергу, блоки ЦВЗ сортуються за кількістю інформації (тобто кількістю значущих (одиничних) пікселів). Вид сортування блоків ЦВЗ (за зростанням / за убуванням) повинен відповідати аналогічній операції над блоками контейнера. Таким чином, кожному блоку контейнера відповідає свій блок ЦВЗ, тобто

$$\boldsymbol{W}_{sort} = \mathbf{permute}(\boldsymbol{W}_{rnd}). \tag{5.26}$$

На рис.5.31 наведено приклад сортування і перестановки блоків.

Індекс блоку контейнера	*Значення дисперсії*
7	67.1
2	61.2
3	53.0
6	41.9
1	32.3
5	14.5
4	7.4

Індекс блоку ЦВЗ	*Кількість одиниць*
4	15
1	13
3	10
2	9
6	8
7	6
5	3

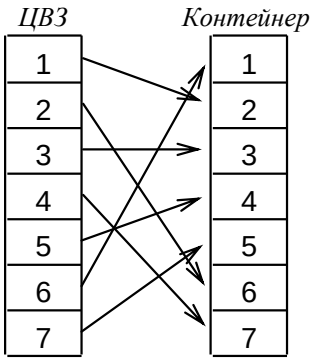


Рис.5.31. Приклад перестановки блоків ЦВЗ в залежності від характеристик блоків контейнера.

Перетворення блоків контейнера. Оскільки ДКП, використовуване при JPEG-компресії, оперує над блоками розмірністю 8×8, бачиться за доцільне розбити контейнер $\boldsymbol{C}$ на блоки зазначеної розмірності. До кожного блоку застосовується пряме ДКП:

$$\boldsymbol{\Omega} = \mathbf{FDCT}(\boldsymbol{C}). \qquad (5.27)$$

Обрання середньочастотних коефіцієнтів ДКП. Для того, щоб вбудований ЦВЗ був візуально непомітним і залишався стійким до компресії даних із втратами, очевидним компромісом є його вбудовування до діапазону середніх частот контейнера. При цьому, для кожного блоку 8×8 контейнера з наявних 64-х відбираються $64\cdot\boldsymbol{A}\cdot\boldsymbol{Z}/(\boldsymbol{X}\cdot\boldsymbol{Y})$ коефіцієнти ДКП, розміщені вздовж другої діагоналі матриці ДКП. Відібрані коефіцієнти для зручності подальших дій згортаються до зменшеної матриці розмірністю $\frac{8\cdot\boldsymbol{A}}{\boldsymbol{X}}\times\frac{8\cdot\boldsymbol{Z}}{\boldsymbol{Y}}$:

$$\boldsymbol{\Omega}_{mid} = \mathbf{reduce}(\boldsymbol{\Omega}). \qquad (5.28)$$

Зокрема, якщо $\boldsymbol{A} = \boldsymbol{X}/2$ та $\boldsymbol{Z} = \boldsymbol{Y}/2$, під час вбудовування ЦВЗ обробляються лише 16 коефіцієнтів ДКП, а інші 48 залишаються незміненими. Викладений вище процес формування масиву СЧ-коефіцієнтів ДКП проілюстровано на рис.5.32.

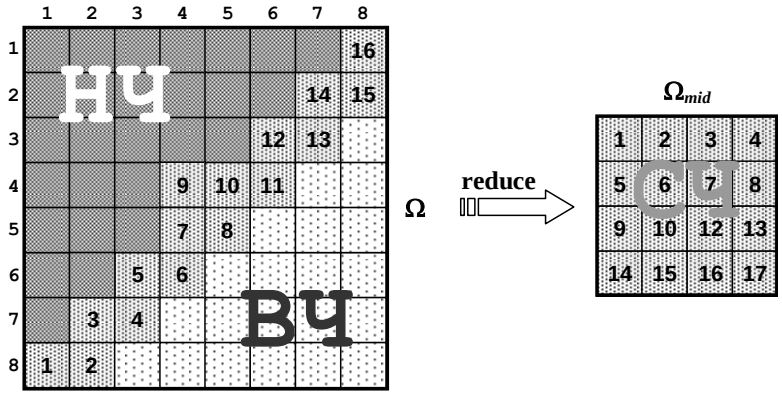


Рис.5.32. Конфігурація матриці $\boldsymbol{\Omega}$ коефіцієнтів ДКП і приклад зведення СЧ-коефіцієнтів до окремої матриці $\boldsymbol{\Omega}_{mid}$

Модифікація коефіцієнтів ДКП. В результаті попередніх дій маємо: переставлений у псевдовипадковому порядку і поставлений у відповідність до блоку контейнера блок ЦВЗ, а також призведене частотне відображення контейнеру (яке містить лише СЧ-компоненти первинного зображення), обидва розмірністю $\frac{8\cdot\boldsymbol{A}}{\boldsymbol{X}}\times\frac{8\cdot\boldsymbol{Z}}{\boldsymbol{Y}}$. Елементи масиву $\boldsymbol{\Omega}_{mid}$ можуть бути модифіковані у відповідності до пікселів вбудовуваного блоку ЦВЗ.

На думку авторів алгоритму, ефективним засобом досягнення непомітності ЦВЗ і стійкості стеганосистеми при низьких коефіцієнтах JPEG-компресії є вбудовування кожного пікселя ЦВЗ шляхом модифікації полярності між відповідними пікселями сусідніх блоків. Проте, зазначають вони, такий метод не стійкий до атак шляхом компресії з високим (≥ 6) коефіцієнтом компресії. Пропонується розглянути технічні аспекти проблеми вбудовування при зазначеному підході, а також покращений метод, який є більш стійким до атак компресії.

Вбудовування шляхом модифікації відношення між значеннями коефіцієнтів сусідніх блоків. Для підрахунку полярності обраних СЧ-коефіцієнтів сусідніх блоків використовується так звана "залишкова" маска. На рис.5.33 зображено приклад такої маски, де кожен елемент (від «А» до «И») містить у собі призведене відображення коефіцієнтів ДКП контейнера $\boldsymbol{\Omega}_{mid}$ певного блоку, причому позиції «Д» відповідає поточне відображення ДКП.

Наприклад, якщо А = Б = В = Е = Ж = З = И = *0*, Г = *–1*, а Д = *1*, то полярність становитиме собою двійковий образ – масив нулів та одиниць, який характеризуватиме, що коефіцієнт ДКП поточної позиції даного блоку відображення коефіцієнтів ДКП є більшим (полярність дорівнює 1) або меншим (полярність дорівнює 0) за коефіцієнт на відповідній позиції попереднього блоку. Тобто, для наведеного прикладу

А	Б	В
Г	Д	Е
Ж	З	И

Рис.5.33. Приклад залишкової маски

$$\boldsymbol{P} = \mathbf{polarity}(\boldsymbol{\Omega}_{mid}) = \begin{cases} 1, \text{ при } \omega_{\mathbf{mid}_b}(\upsilon,\nu) > \omega_{\mathbf{mid}_{b-1}}(\upsilon,\nu); \\ 0, \text{ при } \omega_{\mathbf{mid}_b}(\upsilon,\nu) \le \omega_{\mathbf{mid}_{b-1}}(\upsilon,\nu), \end{cases} \quad (5.29)$$

де $\omega_{\mathbf{mid}_b}(\upsilon,\nu)$ – середньочастотний коефіцієнт ДКП $\boldsymbol{b}$-го блоку.

При інших значеннях елементів залишкової маски відповідно змінюється і вираз (5.29). При обчисленні полярності порівняння значення коефіцієнту ДКП поточного блоку із значеннями відповідних коефіцієнтів декількох сусідніх блоків у більшості випадків дозволяє, крім підвищення рівня захищеності від зламу стеганосистеми, отримати менше спотворення контейнера.

Після одержання відображень полярності $\boldsymbol{P}$ для всіх блоків контейнера, проводиться виявлення коефіцієнтів ДКП, які потребують модифікації для приховання окремого пікселя переставленого ЦВЗ. Пошук проводиться у відповідності до залишкової маски шляхом зміни поточної полярності (**XOR** або знак «⊕» – додавання за модулем 2):

$$\hat{\boldsymbol{P}} = \boldsymbol{XOR}(\boldsymbol{P}, \boldsymbol{W}_{sort}); \quad (5.30)$$

$$\hat{\boldsymbol{P}} = \{\hat{\boldsymbol{p}}(\upsilon,\nu);\ 1 \le \upsilon \le \frac{8\cdot\boldsymbol{A}}{\boldsymbol{X}};\ 1 \le \nu \le \frac{8\cdot\boldsymbol{Z}}{\boldsymbol{Y}}\},$$

де $\hat{\boldsymbol{p}}(\upsilon,\nu) = \begin{cases} 1-\boldsymbol{p}(\upsilon,\nu), \text{ при } \boldsymbol{w}_{sort}(\upsilon,\nu)=1; \\ \boldsymbol{p}(\upsilon,\nu), \text{ при } \boldsymbol{w}_{sort}(\upsilon,\nu)=0. \end{cases} = \boldsymbol{p}(\upsilon,\nu) \oplus \boldsymbol{w}_{sort}(\upsilon,\nu).$

Далі, на основі масивів полярності $\hat{\boldsymbol{P}}$ для кожного блоку контейнера формують масив $\hat{\boldsymbol{\Omega}}_{mid}$ модифікованих СЧ-коефіцієнтів ДКП за умови, щоб різниця між $\boldsymbol{\Omega}_{mid}$ та $\hat{\boldsymbol{\Omega}}_{mid}$ була зведена до мінімуму або була меншою за встановлений поріг η:

$$\hat{\boldsymbol{\Omega}}_{mid} = \mathbf{expand}(\hat{\boldsymbol{P}}), \text{ при } \sum_{\upsilon,\nu}\left[\boldsymbol{w}_{sort}(\upsilon,\nu) - \hat{\boldsymbol{w}}_{sort}(\upsilon,\nu)\right]^2 < \eta. \quad (5.31)$$

Наприклад, задаючись початковим коефіцієнтом $\hat{\boldsymbol{w}}_{sort}(\upsilon_1,\nu_1) = \boldsymbol{w}_{sort}(\upsilon_1,\nu_1)$, необхідно додавати/віднімати коефіцієнти сусідніх блоків (у відповідності до залишкової маски) таким чином, щоб провівши згодом операцію, аналогічну (5.29), можна було одержати відповідну полярність $\hat{\boldsymbol{p}}(\upsilon_1,\nu_1)$. Далі слід перейти до наступних коефіцієнтів, змінюючи лише ті з них, які не будуть впливати на полярність попередньо опрацьованих коефіцієнтів.

Для того, щоб зменшити деградацію зображення (як наслідок вбудовування ЦВЗ), автори методу пропонують обчислювати полярність для абсолютних значень коефіцієнтів ДКП, що дозволить гарантовано зберегти знак (плюс або мінус) модифікованого коефіцієнта.

Крім того, для підвищення стійкості стеганосистеми до JPEG-компресії із втратами, повинен бути врахований ефект квантування, що використовується у технології JPEG. На рис.5.34*а* наведено таблицю квантування яскравості, пропоновану стандартом JPEG, яка, зазвичай, викликає помітні спотворення (так звані "артефакти") зображення. На рис.5.34*б*

зображено іншу таблицю квантування, використовувану у більшості програм, які працюють з JPEG. Видно, що значення при цьому є майже вдвічі меншими за відповідні у попередній таблиці.

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

а)

8	6	5	8	12	20	26	31
6	6	7	10	13	29	30	28
7	7	8	12	20	29	35	28
7	9	11	15	26	44	40	31
9	11	19	28	34	55	52	39
12	18	28	32	41	52	57	46
25	32	39	44	52	61	60	51
36	47	48	49	56	50	52	50

б)

Рис.5.34. Приклади таблиць квантування яскравості: стандартна JPEG (*а*) та Image Alchemy, Handmade Software Inc. (*б*)

Заснована на таблиці квантування полярність являє собою результат обчислення різниці між квантованими і згодом деквантованими коефіцієнтами ДКП відповідних блоків. Для тривіального випадку, коли порівняння ведеться з коефіцієнтами попереднього блоку (див. приклад до рис.5.33), формула (5.29) набуває вигляду

$$P_b = \begin{cases} 1, \text{ при } \left[\dfrac{\left| \omega_{mid_b}(\upsilon,\nu) \right|}{Q_{mid}(\upsilon,\nu)} \right] \cdot Q_{mid}(\upsilon,\nu) > \left[\dfrac{\left| \omega_{mid_{b-1}}(\upsilon,\nu) \right|}{Q_{mid}(\upsilon,\nu)} \right] \cdot Q_{mid}(\upsilon,\nu); \\ 0, \text{ при } \left[\dfrac{\left| \omega_{mid_b}(\upsilon,\nu) \right|}{Q_{mid}(\upsilon,\nu)} \right] \cdot Q_{mid}(\upsilon,\nu) \le \left[\dfrac{\left| \omega_{mid_{b-1}}(\upsilon,\nu) \right|}{Q_{mid}(\upsilon,\nu)} \right] \cdot Q_{mid}(\upsilon,\nu), \end{cases} \quad (5.32)$$

де $Q_{mid}(\upsilon, \nu)$ – значення квантування для СЧ-коефіцієнту з координатами (υ, ν); квадратні дужки вказують на те, що повертається ціла частина від результату ділення.

При цьому, у випадку атаки квантування, попереднє урахування ефекту останнього значно підвищує імовірність правильного розпізнання ознак пікселів при видобуванні. Проте, оскільки квантування має тенденцію до зведення значень багатьох коефіцієнтів у нуль (особливо для більш високочастотних коефіцієнтів), деяка частина СЧ-коефіцієнтів ДКП також в результаті дорівнюватиме нулеві. Крім того, для збереження встановленої полярності і після проведення квантування, тим самим значенням повинні бути модифіковані не лише визначені СЧ-коефіцієнти у поточному блоці, але й в усіх сусідніх блоках у відповідності до маски залишковості.

Вбудовування шляхом модифікації відношення між значеннями коефіцієнтів в межах одного блоку. Для подолання описаних вище технічних недоліків, Хсу і Ву запропонували замість порівняння з СЧ-коефіцієнтами ДКП сусідніх блоків використовувати DC-коефіцієнт поточного блоку. У цьому випадку,

$$P_b = \begin{cases} 1, \text{ при } \left[\dfrac{\left| \omega_{mid_b}(\upsilon,\nu) \right|}{Q_{mid}(\upsilon,\nu)} \right] \cdot Q(\upsilon,\nu) > \left[\dfrac{\omega_b(1,1)}{\psi \cdot Q(1,1)} \right] \cdot Q(1,1); \\ 0, \text{ при } \left[\dfrac{\left| \omega_{mid_b}(\upsilon,\nu) \right|}{Q_{mid}(\upsilon,\nu)} \right] \cdot Q(\upsilon,\nu) \le \left[\dfrac{\omega_b(1,1)}{\psi \cdot Q(1,1)} \right] \cdot Q(1,1), \end{cases} \quad (5.33)$$

де ψ – масштабний коефіцієнт; $Q(1,1)$ – значення квантування для DC-коефіцієнту.

Зворотне перетворення блоків контейнера. Модифіковані матриці СЧ-коефіцієнтів ($\hat{\Omega}_{mid}$) відображуються до загальних матриць коефіцієнтів ДКП (Ω):

$$\hat{\Omega} = \mathbf{put}\,(\hat{\Omega}_{mid})\,. \tag{5.34}$$

До результату проведеного об'єднання застосовується зворотне ДКП:

$$\hat{C} = \mathbf{IDCT}(\hat{\Omega})\,. \tag{5.35}$$

Видобування ЦВЗ з контейнера. Процес видобування вимагає наявності оригінального зображення-контейнера, зображення із вбудованим ЦВЗ, а також зображення-ЦВЗ.

Обидва зображення (оригінальне – $\boldsymbol{C}$, і досліджуване на наявність вбудованого ЦВЗ – $\boldsymbol{C}^*$) піддаються прямому ДКП: $\boldsymbol{\Omega} = \mathbf{FDCT}(\boldsymbol{C})$, $\boldsymbol{\Omega}^* = \mathbf{FDCT}(\boldsymbol{C}^*)$.

З одержаних масивів коефіцієнтів ДКП проводиться виокремлення матриць СЧ-коефіцієнтів, які, у свою чергу, використовуються для одержання шаблонів полярності:

$$\boldsymbol{\Omega}_{mid} = \mathbf{reduce}(\boldsymbol{\Omega}),\quad \boldsymbol{P} = \mathbf{polarity}(\boldsymbol{\Omega}_{mid});$$
$$\boldsymbol{\Omega}^*_{mid} = \mathbf{reduce}(\boldsymbol{\Omega}^*),\quad \boldsymbol{P}^* = \mathbf{polarity}(\boldsymbol{\Omega}^*_{mid}).$$

Застосовуючи операцію додавання за модулем 2 до одержаних масивів полярностей, одержують двійкові дані (поки що переставлені у просторі і псевдовипадково змішані):

$$\boldsymbol{W}^*_{sort} = \mathbf{XOR}(\boldsymbol{P}, \boldsymbol{P}^*), \tag{5.36}$$

де $\boldsymbol{w}^*_{sort}(\upsilon, \nu) = \boldsymbol{p}(\upsilon, \nu) \oplus \boldsymbol{p}^*(\upsilon, \nu)$.

Виконується зворотна просторова перестановка блоків одержаних даних (індекси відповідних пар блоків контейнера і даних, що видобуваються, можуть бути отримані або шляхом їх зчитування з попередньо збережених у файлі на етапі вбудовування ЦВЗ, або безпосередньо при видобуванні шляхом аналогічних дій над зображенням-оригіналом і зображенням-ЦВЗ):

$$\boldsymbol{W}^*_{rnd} = \mathbf{re\text{-}permute}(\boldsymbol{W}^*_{sort}).$$

Аналогічно проводиться зворотна псевдовипадкова перестановка даних в одержаному масиві: $\boldsymbol{W}^* = \mathbf{re\text{-}permute}(\boldsymbol{W}^*_{rnd})$.

В розглянутому алгоритмі можна виділити 3 особливості, які можна використати в якості секретного ключа: **1)** початкове число генератора ПВЧ, яке визначатиме перший елемент псевдовипадкової перестановки (будь-яке ціле число на проміжку $[1, \boldsymbol{A}\cdot\boldsymbol{Z}-1]$); **2)** обрання СЧ-коефіцієнтів ДКП (необхідно обрати $64\cdot\boldsymbol{A}\cdot\boldsymbol{Z}/(\boldsymbol{X}\cdot\boldsymbol{Y})$ коефіцієнти з 64-х для кожного блоку, отже для кожного блоку можна обирати свій набір коефіцієнтів); **3)** зведення обраних коефіцієнтів до окремої матриці (на рис.5.32 показано лише один з можливих способів такого відображення).

Розглянемо приклад реалізації даного методу.

1) Нехай контейнер $\mathbf{C} := \mathbf{READBMP}(\text{"C.bmp"})$, а ЦВЗ – $\mathbf{W} := \mathbf{READBMP}(\text{"W.bmp"})$ (рис.5.35).

При цьому $\mathbf{X} := \mathbf{rows}(\mathbf{C})$, $\mathbf{X} = 128$; $\mathbf{Y} := \mathbf{cols}(\mathbf{C})$, $\mathbf{Y} = 128$; $\mathbf{A} := \mathbf{rows}(\mathbf{W})$, $\mathbf{A} = 64$; $\mathbf{Z} := \mathbf{cols}(\mathbf{W})$, $\mathbf{Z} = 64$.

2) Проводимо нормування масиву ЦВЗ:

$$\mathbf{W} := \overrightarrow{\mathbf{round}(\mathbf{W} \div \mathbf{max}(\mathbf{W}))}$$

При цьому елементи вихідного масиву $\mathbf{W}$ приймають значення 0 або 1.

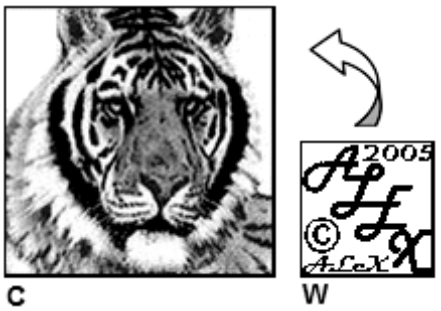


Рис.5.35. Контейнер і ЦВЗ

3) Розмірність блоків, на які розбивається контейнер, $\mathbf{N} := 8$. Кількість одержуваних при цьому блоків: $\aleph_{\mathbf{C}} := \mathbf{X}\cdot\mathbf{Y} \div \mathbf{N}^2$, $\aleph_{\mathbf{C}} = 256$.

Розмірність, яку повинні мати блоки ЦВЗ для одержання їх кількості $\aleph_{\mathbf{W}} = \aleph_{\mathbf{C}}$, $\mathbf{n} := \mathbf{A}\cdot\mathbf{N} \div \mathbf{X}$, $\mathbf{n} = 4$. Виконуємо перевірку кількості одержуваних блоків ЦВЗ: $\aleph_{\mathbf{W}} := \mathbf{A}\cdot\mathbf{Z} \div \mathbf{n}^2$, $\aleph_{\mathbf{W}} = 256$.

4) Програмний модуль розбиття контейнера на $\aleph_{\mathbf{C}}$ блоків розмірністю $\mathbf{N}\times\mathbf{N}$ – (М.55).

```
Бс := | c1 ← 1
      | c2 ← N
      | for b ∈ 1 .. ℵс
      |     | r1 ← mod[N·(b − 1) + 1 , X]
      |     | r2 ← r1 + N − 1
      |     | Бс_b ← submatrix(C , r1 , r2 , c1 , c2)
      |     | if r2 = X
      |     |     | c1 ← c1 + N
      |     |     | c2 ← c2 + N
      | Бс
```

(M.55)

5) Псевдовипадкову перестановку елементів ЦВЗ проведемо у тому порядку, який був запропонований Хсу і Ву. По-перше, проведемо індексацію кожного пікселя ЦВЗ (від 1 до **A·Z** = 4096), для чого просто розгорнемо масив ЦВЗ у вектор (модуль (M.56)). По-друге, отримані індекси розставимо у довільному (псевдовипадковому) порядку, для чого використаємо *лінійний регістр зсуву із зворотним зв'язком* (ЛРЗЗЗ або LFSR –*Linear Feedback Shift Register*).

```
Wvec := | Wvec ← W^⟨1⟩
        | for z ∈ 2 .. Z
        |     Wvec ← stack(Wvec , W^⟨z⟩)
```

(M.56)

Як відомо, ЛРЗЗЗ складається з двох частин: власне регістру зсуву і функції зворотного зв'язку (рис.5.36) [65]. Регістр зсуву становить собою послідовність бітів (розрядів) ***R***, кількість яких ***d*** визначається довжиною регістру зсуву. Зворотний зв'язок являє собою суму по модулю 2 визначених бітів регістру (ці біти називаються *відвідною послідовністю*). Теоретично, ***d***-бітовий ЛРЗЗЗ може перебувати в одному з 2^d–1 внутрішніх станів, тобто може генерувати ПВП з періодом у $T = 2^d$–1 біт. Всі ***T*** внутрішніх стани регістр пройде лише за певних відвідних послідовностей. Такі ЛРЗЗЗ мають максимальний період, а одержаний при цьому результат називають *M-послідовністю*.

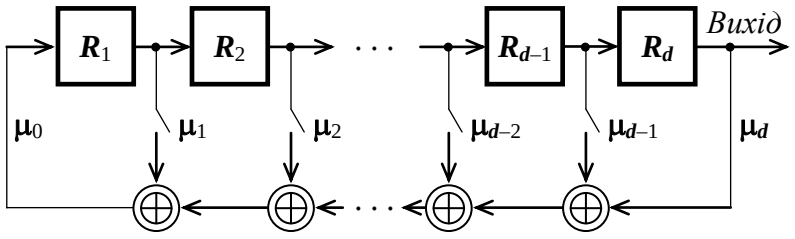


Рис.5.36. Узагальнений лінійний регістр зсуву із зворотним зв'язком

На рис.5.36 значення μ_i (i = 0, 1, ..., ***d***) є ваговим коефіцієнтами поліному $\rho(x)$ степені ***d***, асоційованого з відвідною послідовністю:

$$\rho(x) = \mu_0 \cdot x^0 + \mu_1 \cdot x^1 + \mu_2 \cdot x^2 + \ldots + \mu_{d-2} \cdot x^{d-2} + \mu_{d-1} \cdot x^{d-1} + \mu_d \cdot x^d.$$

Якщо μ_i = 1, то відповідний ключ замкнений. У випадку μ_i = 0 ключ розімкнений. Невдале ввімкнення суматорів до зворотного зв'язку може призвести до одержання ПВП, період повторення якої буде меншим за максимально можливий при наявній розрядності регістру. Для того, щоб конкретний ЛРЗЗЗ мав максимальний період, поліном $\rho(x)$ повинен бути примітивним по модулю 2 (тобто не розкладатися на добуток двійкових поліномів меншої степені). При цьому коефіцієнти μ_0 і μ_d завжди дорівнюють 1, оскільки, у випадку μ_0 = 0, поліном $\rho(x)$ ділиться на x і не є примітивним; у випадку μ_d = 0, навіть якщо поліном і примітивний, його степінь є меншою за ***d***. Інші коефіцієнти обраного поліному і визначатимуть схему формувача ПВП.

У нашому випадку, для перестановки чисел у діапазоні від 1 до **A·Z** необхідна і достатня кількість розрядів регістру **d** := **log**(**A·Z**, 2), **d** = 12. При цьому період повторення ПВП складе **A·Z** – 1.

Для **d**-розрядного регістра в якості примітивного поліному по модулю 2 оберемо наступний: $\rho(x) = 1 + x + x^2 + x^8 + x^{12}$. Цей та деякі інші можливі види примітивних поліномів степеню **d** зведено до табл.5.2.

Таблиця 5.2.

Приклади примітивних по модулю 2 поліномів степені d = 12

x^0	x^1	x^2	x^3	x^4	x^5	x^6	x^7	x^8	x^9	x^{10}	x^{11}	x^{12}
1	1	1	0	0	0	0	0	1	0	0	0	1
1	1	1	0	0	0	0	0	0	0	1	0	1
1	0	1	0	0	0	0	1	1	0	0	0	1
1	0	0	1	1	0	1	0	0	1	1	0	1
1	0	0	1	0	0	1	1	1	0	0	1	1
1	1	0	1	1	0	0	0	0	1	0	1	1

ЛРЗЗЗ, що має **d** розрядів, реалізує програмний модуль (М.57), в якому аргумент **s** визначає собою початковий стан регістра (у десятковому представленні) – довільне ціле число в межах від 1 до **A·Z**–1. На початку модуля задається вектор вагових коефіцієнтів примітивного поліному $\rho(x)$ для елементів відвідної послідовності (для наочності вектор зображено як матрицю-рядок з наступним транспонуванням. Циклом зміни **i** проводиться зміна стану регістра. Кожен **i**-й стан з двійкового формату конвертується до десяткового і зберігається у відповідному елементі вектора $\mathbf{R_{dec}}$. Оскільки період послідовності, що генерується даним регістром, дорівнює $2^{\mathbf{d}} - 1$, а псевдовипадкова перестановка застосовуватиметься до вектору, кількість елементів якого дорівнює $\mathbf{A \cdot Z} = 2^{\mathbf{d}}$, в кінці модуля до сформованого вектора $\mathbf{R_{dec}}$ дописується ще один елемент, значення якого враховує верхній граничний індекс елементів вектора $\mathbf{W_{vec}}$.

```
Vrnd(s) := | μ ← (1  1  0  0  0  0  0  1  0  0  0  1)^T      (x^1 x^2 x^3 ... x^d)
           | for i ∈ 1 .. 2^d − 1
           |   if i = 1
           |     R_dec_i ← s
           |     R_bin ← D2B(R_dec_i)
           |   if i ≥ 2
           |     bit ← 0
           |     for j ∈ 1 .. d
           |       bit ← R_bin_j ⊕ bit   if μ_j = 1
           |     R ← R_bin
           |     for j ∈ 1 .. d
           |       R_bin_j ← R_(j−1)   if j > 1
           |       R_bin_j ← bit   if j = 1
           |     R_dec_i ← B2D(R_bin)
           | r_2^d ← 2^d
           | r
```

(М.57)

Отримання масиву **Vrnd** індексів елементів вектора $\mathbf{W_{vec}}$, розставлених у псевдовипадковому порядку, дозволяє провести генерування пар координат (по рядкам і стовпцям) кожного пікселя шляхом перетворення послідовності ПВЧ на двовимірну послідовність. Це, в свою чергу, робить можливим після псевдовипадкового обрання елементу з вектору $\mathbf{W_{vec}}$ помістити його значення до визначеного згенерованою парою координат елементу масиву,

розмірність якого є ідентичною розмірності ЦВЗ. Вищенаведена процедура реалізована у модулі (М.58), в якому для кожного елементу М-послідовності обчислюються індекси **a** і **z** елементу масиву $\mathbf{W_{rnd}}$, до якого заноситься поточний елемент вектору $\mathbf{W_{vec}}$. Функція **trunc**(***x***) повертає цілу частину від аргументу ***x***, відкидаючи його мантису; функція **mod**(***k***, ***m***) повертає залишок від ділення ***k*** на ***m***. Додаванням одиниці врахована можливість повернення зазначеними функціями нульового результату. Результат обрахунку модуля (М.58) при **s** := 12 наведено на рис.5.37.

$$\mathbf{W_{rnd}} := \left| \begin{array}{l} \mathfrak{I} \leftarrow \mathbf{Vrnd(s)} \\ \mathbf{for}\ \ i \in 1..A \cdot Z \\ \quad \left| \begin{array}{l} a \leftarrow \mathbf{trunc}\left(\dfrac{\mathfrak{I}_i - 1}{A}\right) + 1 \\ z \leftarrow \mathbf{mod}(\mathfrak{I}_i, Z) + 1 \\ W_{rnd_{a,z}} \leftarrow W_{vec_i} \end{array} \right. \\ W_{rnd} \end{array} \right. \qquad \text{(М.58)}$$

Рис.5.37. Результат псевдовипадкової перестановки елементів ЦВЗ

6) Модуль розбиття масиву ЦВЗ на $\aleph_W$ блоків розмірністю **n**×**n** за своєю побудовою є аналогічним модулю (М.55). Відмінності полягають у наступному: змінна, якій присвоюється результат виконання модуля (як, власне, і відповідна змінна в тілі модуля), позначається як $\mathbf{Б_W}$ (замість $\mathbf{Б_C}$); виокремлення блоків проводиться з масиву $\mathbf{W_{rnd}}$ (замість **C**); замість розмірності масиву **N** використовується розмірність **n**; відповідно, змінюється і граничне значення індексу рядка (замість **X** – **A**). Загальну кількість блоків, на яку розбивався контейнер ($\aleph_C$), з огляду на таку ж їх кількість у ЦВЗ, можна не змінювати.

7) Формуємо таблиці результатів сортування блоків контейнера (за значенням стандартного відхилення елементів блоків) і блоків ЦВЗ (за кількістю значущих елементів).

$$\mathbf{T_C} := \left| \begin{array}{l} \mathbf{for}\ \ b \in 1..\aleph_C \\ \quad \left| \begin{array}{l} T_{b,1} \leftarrow b \\ \sigma \leftarrow \sqrt{\dfrac{1}{N^2} \cdot \sum_{i=1}^{N} \sum_{j=1}^{N} \left[\left(Б_{C_b}\right)_{i,j} \right]^2 - \left[\dfrac{1}{N^2} \cdot \sum_{i=1}^{N} \sum_{j=1}^{N} \left(Б_{C_b}\right)_{i,j} \right]^2} \\ T_{b,2} \leftarrow \sigma \end{array} \right. \\ \mathbf{reverse}(\mathbf{csort}(\mathbf{T}, 2)) \end{array} \right. \qquad \text{(М.59)}$$

До першого стовпця таблиць характеристик блоків контейнера ($\mathbf{T_C}$) і ЦВЗ ($\mathbf{T_W}$) вносяться порядкові індекси досліджуваних блоків. До другого – результат обчислення, відповідно, стандартного відхилення (σ) і кількості одиничних елементів ($\Sigma\mathbf{1}$). Після формування таблиць, останні сортируються за значеннями другого стовпця (функція **csort**(**T**,2)).

$$\mathbf{T_W} := \left| \begin{array}{l} \mathbf{for}\ \ b \in 1..\aleph_W \\ \quad \left| \begin{array}{l} T_{b,1} \leftarrow b \\ \Sigma 1 \leftarrow 0 \\ \mathbf{for}\ \ \upsilon \in 1..n \\ \quad \mathbf{for}\ \ \nu \in 1..n \\ \qquad \Sigma 1 \leftarrow \Sigma 1 + \left(Б_{W_b}\right)_{\upsilon,\nu} \\ T_{b,2} \leftarrow \Sigma 1 \end{array} \right. \\ \mathbf{reverse}(\mathbf{csort}(\mathbf{T}, 2)) \end{array} \right. \qquad \text{(М.60)}$$

Фрагмент результату сортування для обраних контейнера і ЦВЗ наведено у табл.5.3.

Шляхом виокремлення перших стовпців з масивів $\mathbf{T_C}$ і $\mathbf{T_W}$, співставляємо індекси блоків контейнера з індексами блоків ЦВЗ:

$$\mathbf{T_\Sigma} := \mathbf{augment}\left(\mathbf{T_C}^{\langle 1 \rangle}, \mathbf{T_W}^{\langle 1 \rangle}\right).$$

Таблиця 5.3.

Приклад сортування блоків контейнера і ЦВЗ

№ п/п	Індекси блоків контейнера	Значення станд. відхилення	Індекси блоків ЦВЗ	Кількість значущих елементів
1	232	66,676	84	15
2	54	65,551	183	15
3	55	65,236	185	15
...	...	...	...	...
64	45	44,565	152	12
65	190	44,394	153	12
66	98	44,318	154	12
...	...	...	...	...
128	81	33,562	169	11
129	79	33,462	117	11
130	209	33,287	245	11
...	...	...	...	...
192	175	22,564	48	10
193	29	22,396	193	10
194	239	21,151	194	10
...	...	...	...	...
254	3	0	138	7
255	1	0	50	7
256	241	0	46	6

8) У відповідності до отриманого масиву поставлених у відповідність один одному індексів, проводимо перестановку блоків ЦВЗ у порядку, що відповідає даній відповідності.

```
W_sort := | for b ∈ 1 .. ℵC
          |     i1 ← T_Σb,1
          |     i2 ← T_Σb,2
          |     W_sort_i1 ← Б_W_i2
          | W_sort
```

(М.61)

Результат виконання модуля (М.61) для першого рядка табл.5.3:

$$Б_{W_{84}} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix};\quad W_{sort_{232}} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

9) Використовуючи програмний модуль (М.47) (граничну кількість блоків контейнера у циклі **for** замінити з N_C на $\aleph_C$; а замість C_b під знаком суми слід використати $Б_{C_b}$), виконуємо пряме ДКП блоків контейнера.

З одержаних масивів коефіцієнтів ДКП Ω_b, що мають розмірність $N \times N$, виокремлюємо лише середньочастотні коефіцієнти (див. рис.5.32), паралельно згортаючи їх до масиву $n \times n$. Перед початком проведення виокремлення, формується масив координат виокремлюваних СЧ-коефіцієнтів:

$$\Theta := \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 16 \\ 0 & 0 & 0 & 0 & 0 & 0 & 14 & 15 \\ 0 & 0 & 0 & 0 & 0 & 12 & 13 & 0 \\ 0 & 0 & 0 & 9 & 10 & 11 & 0 & 0 \\ 0 & 0 & 0 & 7 & 8 & 0 & 0 & 0 \\ 0 & 0 & 5 & 6 & 0 & 0 & 0 & 0 \\ 0 & 3 & 4 & 0 & 0 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

```
θ := | for i ∈ 1 .. N
     |   for j ∈ 1 .. N
     |     if Θ_i,j > 0
     |       θ_Θ_i,j,1 ← i
     |       θ_Θ_i,j,2 ← j
     | θ
```

(М.62)

У цьому випадку, масив θ містить 16 рядків, елементи кожного з яких несуть інформацію про індекси рядка і стовпця відповідного СЧ-коефіцієнта у масиві Θ (рис.5.38).

$\theta^T =$

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	⟸ порядкові № СЧ-коефіцієнтів
1	8	8	7	7	6	6	5	5	4	4	4	3	3	2	2	1	⟸ індекс рядка масиву Θ
2	1	2	2	3	3	4	4	5	4	5	6	6	7	7	8	8	⟸ індекс стовпця масиву Θ

Рис.5.38. Таблиця координат СЧ-коефіцієнтів ДКП

У відповідності таблиці рис.5.38, для кожного блоку **b** виконується формування матриці Ω_{mid} обраних для модифікації коефіцієнтів (М.63). Результат виокремлення проілюстровано на рис.5.39.

$$\Omega_{mid} := \left| \begin{array}{l} \text{for } b \in 1..\aleph_C \\ \quad q \leftarrow 1 \\ \quad \text{for } \upsilon \in 1..n \\ \quad\quad \text{for } \nu \in 1..n \\ \quad\quad\quad \Omega'mid_{\upsilon,\nu} \leftarrow (\Omega_b)_{(\theta_q,1),(\theta_q,2)} \\ \quad\quad\quad q \leftarrow q+1 \\ \quad \Omega mid_b \leftarrow \Omega'mid \\ \Omega_{mid} \end{array} \right. \qquad \text{(М.63)}$$

Ω_{232} =

833.62	–494.34	109.89	99.26	–72.62	–12.92	28.49	**–10.5**
–37.49	18.15	27.92	–25.43	–4.7	12.04	**0.32**	**–14.48**
–48.45	30.19	31.45	–33.88	–11.36	**29.37**	**0.62**	–10.97
7.72	1	–10.55	**–0.22**	**21.76**	**–0.45**	–11.96	16.57
–6.12	1.32	11.16	**–14.34**	**–0.37**	1.08	–0.16	0.65
0.01	0.16	**14.31**	**–0.02**	–13.12	0.63	0.15	–0.04
–0.36	**–0.35**	**0.62**	–0.55	–0.3	–0.37	–0.2	–0.54
–0.29	**–0.06**	0.55	0	0.85	0.84	0.15	–0.06

;

Ωmid_{232} =

–0.29	–0.06	–0.35	0.62
14.31	–0.02	–14.34	–0.37
–0.22	21.76	–0.45	29.37
0.62	0.32	–14.48	–10.5

.

Рис.5.39. Приклад виокремлення СЧ-коефіцієнтів з масиву Ω для 232-го блоку контейнера

10) Проводимо обчислення масиву полярностей блоків контейнера. При цьому до основи програмного модуля (М.64) покладено реалізацію виразу (5.29); матриця СЧ-коефіцієнтів 1-го блоку порівнюється з матрицею останнього ($\aleph_C$) блоку. Для створення більш стійкого вбудовування ЦВЗ даний модуль можна модифікувати у відповідності до (5.33).

$$P := \left| \begin{array}{l} \text{for } b \in 1..\aleph_C \\ \quad \left.\begin{array}{l} \Delta \leftarrow \Omega mid_b - \Omega mid_{\aleph_C} \quad \text{if } b = 1 \\ \Delta \leftarrow \Omega mid_b - \Omega mid_{b-1} \quad \text{if } b > 1 \end{array}\right\} \xrightarrow{(5.33)} \\ \quad \boxed{\Delta \leftarrow \overrightarrow{\left(\text{trunc}\left(\frac{\left|\Omega mid_b\right|}{Q_{mid}} \right) \cdot Q_{mid} \right) - \text{trunc}\left[\frac{(\Omega_b)_{1,1}}{\psi \cdot Q_{1,1}} \right] \cdot Q_{1,1}}} \\ \quad \text{for } \upsilon \in 1..n \\ \quad\quad \text{for } \nu \in 1..n \\ \quad\quad\quad P'_{\upsilon,\nu} \leftarrow 1 \quad \text{if } \Delta_{\upsilon,\nu} > 0 \\ \quad\quad\quad P'_{\upsilon,\nu} \leftarrow 0 \quad \text{if } \Delta_{\upsilon,\nu} \le 0 \\ \quad P_b \leftarrow P' \\ P \end{array} \right. \qquad \text{(М.64)}$$

11) Згідно виразу (5.30), проводимо зміну поточного масиву полярності у відповідності до елементів переставленого ЦВЗ – модуль (М.65).

$$P\text{^} := \left| \begin{array}{l} \text{for } b \in 1..\aleph_C \\ \quad P\text{^}_b \leftarrow \overrightarrow{(P_b \oplus W_{sort_b})} \\ P\text{^} \end{array} \right. \qquad \text{(М.65)}$$

12) На основі поточних матриць СЧ-коефіцієнтів (Ω_{mid}) формуємо нові ($\Omega\text{^}_{mid}$), причому зазнають зміни ті елементи первинної матриці, за координатами яких виконується нерівність виду $P\text{^}_{\upsilon,\nu} \neq P_{\upsilon,\nu}$ (М.66). Модифікація

проводиться таким чином, щоб при виконанні програмного модуля (М.64), якщо при обчисленні параметру **Δ** в якості зменшуваного виступатиме елемент матриці $\Omega\hat{}_{mid}$, можна було одержати масив **P***, ідентичний масиву **P^**. У випадку модифікації відношення між значеннями коефіцієнтів в межах одного блоку (тобто полярність **P** визначалася за формулою (5.33)), у модулі (М.66 *а*) вираз, який обчислюється при виконанні умови $P\hat{}_{\upsilon,\nu} \neq P_{\upsilon,\nu}$, треба змінити на підмодуль (М.66 *б*).

$$
\begin{array}{l}
\Omega\hat{}_{mid} := \left|\begin{array}{l}
\textbf{for}\ \ b \in 1..\aleph_C \\
\quad \left|\begin{array}{l}
\Omega\hat{}'_{mid} \leftarrow \Omega_{mid_b} \\
\textbf{for}\ \ \upsilon \in 1..n \\
\quad \textbf{for}\ \ \nu \in 1..n \\
\qquad \textbf{if}\ \ \left(P\hat{}_b\right)_{\upsilon,\nu} \neq \left(P_b\right)_{\upsilon,\nu} \\
\qquad \left|\begin{array}{l}
\text{змінити при } P - \text{за (5.33)} \\
\left|\begin{array}{l}
\textbf{if}\ \ \left(P\hat{}_b\right)_{\upsilon,\nu} = 0 \\
\quad \left|\begin{array}{l}
\Omega\hat{}'_{mid_{\upsilon,\nu}} \leftarrow \left(\Omega_{mid_{\aleph_C}}\right)_{\upsilon,\nu} - \eta \quad \textbf{if}\ \ b = 1 \\
\Omega\hat{}'_{mid_{\upsilon,\nu}} \leftarrow \left(\Omega_{mid_{b-1}}\right)_{\upsilon,\nu} - \eta \quad \textbf{if}\ \ b > 1
\end{array}\right. \\
\textbf{if}\ \ \left(P\hat{}_b\right)_{\upsilon,\nu} = 1 \\
\quad \left|\begin{array}{l}
\Omega\hat{}'_{mid_{\upsilon,\nu}} \leftarrow \left(\Omega_{mid_{\aleph_C}}\right)_{\upsilon,\nu} + \eta \quad \textbf{if}\ \ b = 1 \\
\Omega\hat{}'_{mid_{\upsilon,\nu}} \leftarrow \left(\Omega_{mid_{b-1}}\right)_{\upsilon,\nu} + \eta \quad \textbf{if}\ \ b > 1
\end{array}\right.
\end{array}\right.
\end{array}\right. \\
\Omega\hat{}_{mid_b} \leftarrow \Omega\hat{}'_{mid}
\end{array}\right. \\
\Omega\hat{}_{mid}
\end{array}\right.
\end{array}
\qquad \text{(М.66 } a\text{)}
$$

$$
\left|\begin{array}{l}
\textbf{while}\ \ \text{trunc}\left(\dfrac{\left|\Omega\hat{}'_{mid_{\upsilon,\nu}}\right|}{Q_{mid_{\upsilon,\nu}}}\right)\cdot Q_{mid_{\upsilon,\nu}} - \text{trunc}\left[\dfrac{\left(\Omega_b\right)_{1,1}}{\psi\cdot Q_{1,1}}\right]\cdot Q_{1,1} > 0 \quad \textbf{if}\ \ \left(P\hat{}_b\right)_{\upsilon,\nu} = 0 \\
\quad \left|\begin{array}{l}
\Omega\hat{}'_{mid_{\upsilon,\nu}} \leftarrow \Omega\hat{}'_{mid_{\upsilon,\nu}} - \eta \quad \textbf{if}\ \ \Omega\hat{}'_{mid_{\upsilon,\nu}} > 0 \\
\Omega\hat{}'_{mid_{\upsilon,\nu}} \leftarrow \Omega\hat{}'_{mid_{\upsilon,\nu}} + \eta \quad \textbf{if}\ \ \Omega\hat{}'_{mid_{\upsilon,\nu}} < 0
\end{array}\right. \\
\textbf{while}\ \ \text{trunc}\left(\dfrac{\left|\Omega\hat{}'_{mid_{\upsilon,\nu}}\right|}{Q_{mid_{\upsilon,\nu}}}\right)\cdot Q_{mid_{\upsilon,\nu}} - \text{trunc}\left[\dfrac{\left(\Omega_b\right)_{1,1}}{\psi\cdot Q_{1,1}}\right]\cdot Q_{1,1} \leq 0 \quad \textbf{if}\ \ \left(P\hat{}_b\right)_{\upsilon,\nu} = 1 \\
\quad \left|\begin{array}{l}
\Omega\hat{}'_{mid_{\upsilon,\nu}} \leftarrow \Omega\hat{}'_{mid_{\upsilon,\nu}} + \eta \quad \textbf{if}\ \ \Omega\hat{}'_{mid_{\upsilon,\nu}} \geq 0 \\
\Omega\hat{}'_{mid_{\upsilon,\nu}} \leftarrow \Omega\hat{}'_{mid_{\upsilon,\nu}} - \eta \quad \textbf{if}\ \ \Omega\hat{}'_{mid_{\upsilon,\nu}} < 0
\end{array}\right.
\end{array}\right.
\qquad \text{(М.66 } b\text{)}
$$

13) Модифіковані для кожного блоку матриці СЧ-коефіцієнтів (Ω_{mid}) відображуються до загальних матриць коефіцієнтів ДКП ($\Omega\hat{}$) – програмний модуль (М.67). При цьому також використовується таблиця координат середньочастотних коефіцієнтів ДКП (див. (М.62) і рис.5.38).

$$
\begin{array}{l}
\Omega\hat{} := \left|\begin{array}{l}
\textbf{for}\ \ b \in 1..\aleph_C \\
\quad \left|\begin{array}{l}
\Omega\hat{}' \leftarrow \Omega_b \\
q \leftarrow 1 \\
\textbf{for}\ \ \upsilon \in 1..n \\
\quad \textbf{for}\ \ \nu \in 1..n \\
\qquad \left|\begin{array}{l}
\Omega\hat{}'\left(\theta_{q,1}\right),\left(\theta_{q,2}\right) \leftarrow \left(\Omega\hat{}_{mid_b}\right)_{\upsilon,\nu} \\
q \leftarrow q + 1
\end{array}\right. \\
\Omega\hat{}_b \leftarrow \Omega\hat{}'
\end{array}\right. \\
\Omega\hat{}
\end{array}\right.
\end{array}
\qquad \text{(М.67)}
$$

14) Після об'єднання, до модифікованих матриць **Ω^** необхідно застосувати зворотне ДКП (модуль (М.68)) і сформувати на основі блоків загальний масив контейнера (модуль (М.69)).

$$\begin{aligned}
&\text{Б^} := \Big| \ \text{for}\ \ b \in 1..\aleph_C \\
&\quad \Big| \ \text{for}\ \ x \in 0..N-1 \\
&\quad\quad \text{for}\ \ y \in 0..N-1 \\
&\quad\quad\quad \text{Б^}'_{x+1,y+1} \leftarrow \frac{1}{\sqrt{2\cdot N}} \cdot \sum_{\upsilon=0}^{N-1} \sum_{\nu=0}^{N-1} \left[\zeta(\upsilon)\cdot\zeta(\nu)\cdot(\Omega\text{^}_b)_{\upsilon+1,\nu+1} \times \right. \\
&\quad\quad\quad\quad \left. \times \cos\left[\frac{\pi\cdot\upsilon\cdot(2\cdot x+1)}{2\cdot N}\right]\cdot\cos\left[\frac{\pi\cdot\nu\cdot(2\cdot y+1)}{2\cdot N}\right]\right] \\
&\quad \Big| \ \text{Б^}_b \leftarrow \text{Б^}' \\
&\Big| \ \text{Б^}
\end{aligned} \qquad \text{(М.68)}$$

Отримане при цьому зображення із вбудованим ЦВЗ за певних обставин може занадто втратити у яскравості, що викликано декількома причинами: по-перше, для спрощення програмних модулів не була проведена оптимізація вбудовування за формулою (5.31); по-друге, сусідні блоки контейнера можуть мати досить різні значення інтенсивностей і, відповідно, СЧ-коефіцієнтів ДКП, що викликає необхідність при побудові алгоритму за (5.29) істотно змінювати значення цих коефіцієнтів для задоволення поставлених умов. У комплексі ці дві причини викликають появу пікселів контейнера, яскравість яких після проведення зворотного ДКП, виходить за межі [0, 255]. Останнє враховується нормуванням значень елементів масиву **C^** наприкінці модуля (М.69), яке і викликає зниження загальної яскравості зображення. Результати вбудовування ЦВЗ до контейнера шляхом модифікації відношення між значеннями коефіцієнтів сусідніх блоків і в межах одного блоку приведені на рис.5.39 *а* і *б*, відповідно.

$$\begin{aligned}
&\text{C^} := \Big| \ \text{C^} \leftarrow \text{Б^}_1 \\
&\quad \text{for}\ \ b \in 2..\frac{X}{N} \\
&\quad\quad \text{C^} \leftarrow \text{stack}\left(\text{C^}, \text{Б^}_b\right) \\
&\quad \text{C^}' \leftarrow 0 \\
&\quad \text{for}\ \ b \in \frac{X}{N}+1..\aleph_C \\
&\quad\quad \Big| \ \text{C^}' \leftarrow \text{Б^}_b \ \ \text{if}\ \ \text{C^}' = 0 \\
&\quad\quad \Big| \ \text{C^}' \leftarrow \text{stack}\left(\text{C^}', \text{Б^}_b\right) \ \ \text{otherwise} \\
&\quad\quad \Big| \ \text{if}\ \ \text{mod}\left(b, \frac{X}{N}\right) = 0 \\
&\quad\quad\quad \Big| \ \text{C^} \leftarrow \text{augment}(\text{C^}, \text{C^}') \\
&\quad\quad\quad \Big| \ \text{C^}' \leftarrow 0 \\
&\quad \text{C^} \leftarrow \frac{\text{C^} + |\min(\text{C^})|}{\max(\text{C^} + |\min(\text{C^})|)} \cdot 255
\end{aligned} \qquad \text{(М.69)}$$

15) Відтворимо процес видобування ЦВЗ з зображення-контейнера. Як було зазначено вище, для видобування ЦВЗ крім, власне, контейнера із можливо вбудованим ЦВЗ (**C***) є необхідною наявність оригінального (незаповненого) контейнера (**C**) і зображення ЦВЗ (**W**).

Зображення **C** і **C*** розбиваємо на блоки (модуль (М.55), з відповідною заміною змінних при розбитті **C*** на такі, що характеризують саме це зображення).

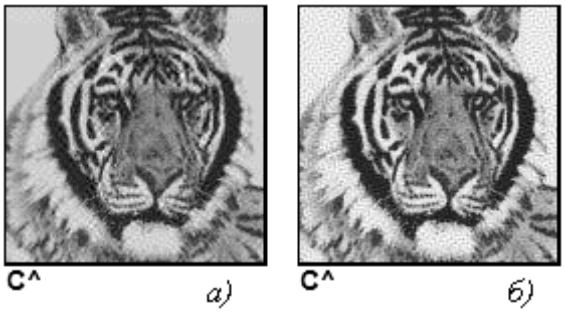


Рис.5.39. Контейнери із вбудованим ЦВЗ

До кожного блоку оригінального і досліджуваного на ЦВЗ зображення застосовуємо пряме ДКП (модуль (М.47)), а на основі одержаних матриць коефіцієнтів (**Ω** і **Ω***) формуємо матриці СЧ-коефіцієнтів Ω_{mid} і Ω^*_{mid} (модуль (М.63)), які використовуємо при обчисленні шаблонів полярності **P** і **P*** (модуль (М.64)).

Шляхом поблочного додавання по модулю 2 одержаних матриць полярностей одержуємо двійкові дані, які, якщо контейнер дійсно містить ЦВЗ, відповідають переставленим у просторі і псевдовипадково змішаним елементам ЦВЗ. Операцію додавання виконуємо за допомогою програмного модуля (М.70).

$$W^*_{sort} := \left| \begin{array}{l} \text{for } b \in 1 .. \aleph_{C^*} \\ \quad W^*_{sort_b} \leftarrow \overrightarrow{(P_b \oplus P^*_b)} \\ W^*_{sort} \end{array} \right. \qquad \text{(М.70)}$$

Формуємо масив T_Σ індексів співставлених пар блоків контейнера і оригінального ЦВЗ (див. (М.59), (М.60)), на підставі якого виконуємо зворотну просторову перестановку блоків масиву W^*_{sort} – модуль (М.71).

$$Б^*W := \left| \begin{array}{l} \text{for } b \in 1 .. \aleph_{C^*} \\ \quad s1 \leftarrow T_{\Sigma_{b,1}} \\ \quad s2 \leftarrow T_{\Sigma_{b,2}} \\ \quad Б^*W_{s2} \leftarrow W^*_{sort_{s1}} \\ Б^*W \end{array} \right. \qquad \text{(М.71)}$$

Просторово переставлені блоки $Б^*_W$ об'єднуємо у спільний масив W^*_{rnd} (М.72), елементи якого гіпотетично є псевдовипадково змішаними елементами оригінального ЦВЗ.

$$W^*_{rnd} := \left| \begin{array}{l} W^*_{rnd} \leftarrow Б^*W_1 \\ \text{for } b \in 2 .. A \div n \\ \quad W^*_{rnd} \leftarrow \text{stack}(W^*_{rnd}, Б^*W_b) \\ W^{*'}_{rnd} \leftarrow 0 \\ \text{for } b \in A \div n + 1 .. \aleph_W \\ \quad W^{*'}_{rnd} \leftarrow Б^*W_b \text{ if } W^{*'}_{rnd} = 0 \\ \quad W^{*'}_{rnd} \leftarrow \text{stack}(W^{*'}_{rnd}, Б^*W_b) \text{ otherwise} \\ \quad \text{if } \text{mod}(b, A \div n) = 0 \\ \quad\quad W^*_{rnd} \leftarrow \text{augment}(W^*_{rnd}, W^{*'}_{rnd}) \\ \quad\quad W^{*'}_{rnd} \leftarrow 0 \\ W^*_{rnd} \end{array} \right. \qquad \text{(М.72)}$$

Проводимо зворотну псевдовипадкову перестановку даних, використовуючи (М.57) для одержання ПВЧ, на основі яких генерується пара координат елементу в масиві W^*_{rnd}, значення якого присвоюється **i**-му елементові вектора W^*_{vec} – модуль (М.73).

$$W^*_{vec} := \left| \begin{array}{l} \Im \leftarrow \text{Vrnd}(s) \\ \text{for } i \in 1 .. A \cdot Z \\ \quad a \leftarrow \text{trunc}\left(\frac{\Im_i - 1}{A}\right) + 1 \\ \quad z \leftarrow \text{mod}(\Im_i, Z) + 1 \\ \quad W^*_{vec_i} \leftarrow W^*_{rnd_{a,z}} \\ W^*_{vec} \end{array} \right. \qquad \text{(М.73)}$$

Одержаний при цьому вектор згортаємо до масиву **W*** з розмірністю оригінального ЦВЗ (модуль (М.74)).

Графічне представлення видобутих ЦВЗ, вбудовування яких до контейнера було проведене шляхом зміни відношень між значеннями коефіцієнтів ДКП сусідніх блоків і в межах одного блоку зображено відповідно на рис.5.40 *а* і *б*.

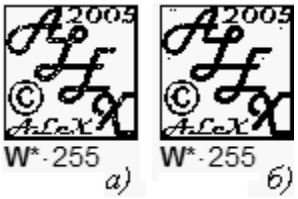


Рис.5.40. Видобуті з контейнера **C^** ЦВЗ

$$W^* := \text{for } z \in 1 .. Z \quad W^{*\langle z \rangle} \leftarrow \text{submatrix}\left[W^*_{vec}, (z-1)\cdot A + 1, z \cdot A, 1, 1\right] \qquad \text{(М.74)}$$

Результати обчислення показників візуального спотворення для двох розглянутих різновидів методу занесені до табл.5.4. Зауважимо, що при порівнянні отриманих результатів з результатами інших методів, слід брати до уваги, що до контейнера вбудовувалася інформація, об'єм (у пікселях) якої був всього у 4 рази меншим за об'єм контейнера.

5.3.3.4. Метод Фрідріх

Алгоритм, що запропонувала *J. Fridrich* у [106], по суті є комбінацією двох алгоритмів: за одним з них приховувані дані вбудовуються до низькочастотних, а за іншим – до середньочастотних коефіцієнтів ДКП. Як було показано автором, каскадне використання двох різних алгоритмів дозволяє одержати хороші результати стосовно стійкості стеганосистеми до атак.

Зображення, яке планується використати в якості контейнера, конвертується у сигнал з нульовим математичним очікуванням і певним стандартним відхиленням таким чином, що НЧ-коефіцієнти ДКП, які будуть одержані в подальшому, потрапляли у попередньо заданий незмінний діапазон. Запропоноване перетворення

$$\boldsymbol{G} = \frac{1024}{\sqrt{\boldsymbol{X}\cdot\boldsymbol{Y}}}\cdot\frac{\boldsymbol{C}-\overline{\boldsymbol{C}}}{\sigma(\boldsymbol{C})}, \tag{5.37}$$

де $\boldsymbol{X},\boldsymbol{Y}$ – розмірність зображення $\boldsymbol{C}$ у пікселях; $\overline{\boldsymbol{C}}$ і $\sigma(\boldsymbol{C})$ – відповідно, математичне очікування і стандартне відхилення значень яскравості пікселів зображення, – трансформує напівтонове зображення $\boldsymbol{C}$ у двомірний сигнал $\boldsymbol{G}$ з нульовим математичним очікуванням, такий, що абсолютне значення максимального НЧ-коефіцієнта ДКП $\boldsymbol{G}$ не перевищує (200...250). При цьому стверджується, що дане перетворення застосовне для широкого кола різноманітних зображень: як з великими однорідними областями, так і дуже текстурованих.

Приховуване повідомлення $\boldsymbol{W}$ являє собою послідовність чисел $\{-1, 1\}$.

Для зображення $\boldsymbol{G}$ виконується обчислення коефіцієнтів ДКП, з яких низькочастотні модифікуються таким чином, щоб в них було закодовано сигнал $\boldsymbol{W}$. Для цього попередньо необхідно визначити геометричну прогресію дійсних чисел:

$$\tau_{\boldsymbol{i}+1} = \frac{1+\alpha}{1-\alpha}\cdot\tau_{\boldsymbol{i}}; \quad \tau_1 = 1, \tag{5.38}$$

параметризовану за допомогою параметру $\alpha\in(0,1)$.

Для $\boldsymbol{t} > 1$, $\tau_{\boldsymbol{i}} \le \boldsymbol{t} < \tau_{\boldsymbol{i}+1}$ визначається індексна функція

$$\boldsymbol{ind}(\boldsymbol{t}) = (-1)^{\boldsymbol{i}}, \text{ якщо } \boldsymbol{t}\in[\tau_{\boldsymbol{i}}, \tau_{\boldsymbol{i}+1}). \tag{5.39}$$

Таким чином, для кожного дійсного числа $\boldsymbol{t} > 1$ можна визначити його індекс (+/–1). Цілком очевидно, що цей індекс може бути змінений шляхом додавання/віднімання числа, яке не перевищує $\alpha\cdot\boldsymbol{t}$. На рис.5.41 зображено індексні функції для α = 0.1, 0.2 і 0.3.

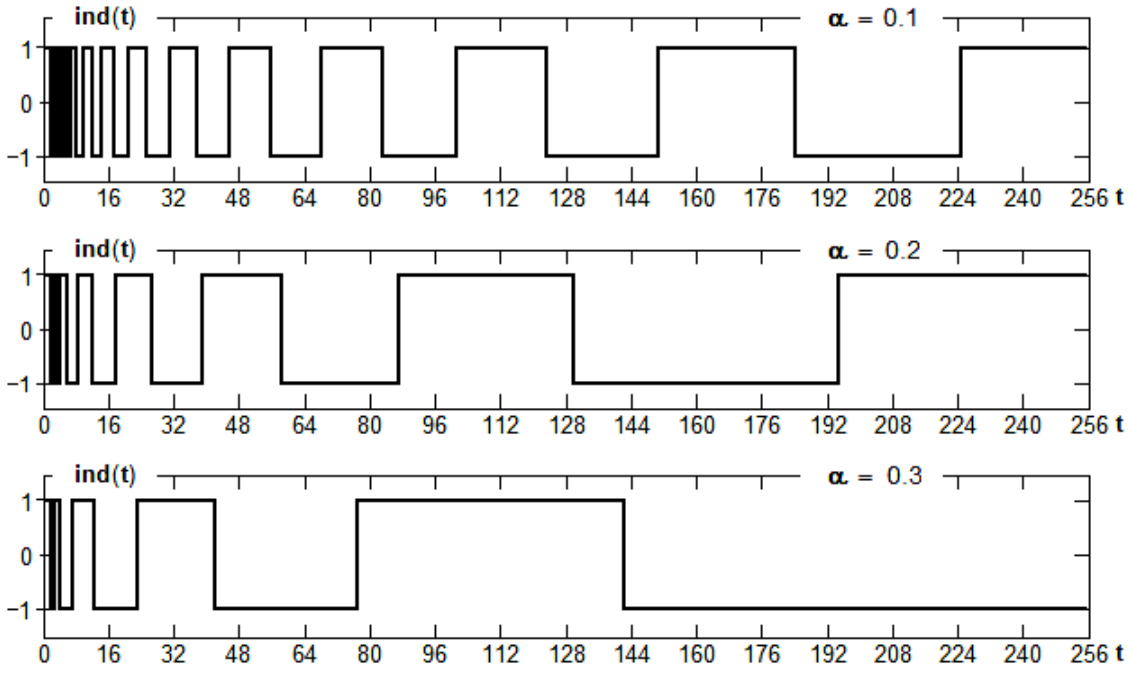


Рис.5.41. Індексна функція $\boldsymbol{ind}(\boldsymbol{t})$ при α = 0.1, 0.2 і 0.3.

Для вбудовування масиву повідомлення $\boldsymbol{W}$, кожен окремий біт якого може приймати значення $W_j \in \{-1, 1\}$, $j \in \{1, 2, ..., \aleph_W\}$, обираються $\aleph_{\Omega low} = \aleph_W$ НЧ-коефіцієнтів ДКП – Ω_j, значення яких змінюється таким чином, щоб $ind\left(\left|\Omega\hat{}_j\right|\right) = W_j$. Якщо $\left|\Omega_j\right| < 1$, коефіцієнт для вбудовування не використовується.

Завдяки властивостям індексної функції, як вказує автор, кожен коефіцієнт буде змінений не більше ніж на $100 \cdot \alpha$ відсотків[9]. Також зазначається, що зміни носитимуть випадковий характер, оскільки не існує жодних підстав, за яких коефіцієнти ДКП на початковому етапі кодування були б наслідком певного повідомлення.

Найбільша стійкість стеганосистеми до спотворювань контейнера досягається при встановленні нових значень коефіцієнтів ДКП як середини інтервалів $[\tau_i, \tau_{i+1})$. Проте, це може послужити з'явленню скупчень однакових коефіцієнтів ДКП, що робить таку систему ненадійною з точки зору можливого стеганоаналізу. Значення параметру α обирається таким, щоб вбудовування повідомлення не призводило до помітних оку спотворень контейнера.

Операція видобування проводиться шляхом виконання аналогічних до операції вбудовування перетворень контейнера, підозрюваного на наявність вбудованого повідомлення: конвертація (5.37), ДКП, обчислення для заздалегідь оговорених коефіцієнтів ДКП індексної функції (5.39) при заданому параметрі α, формування з одержаних індексів масиву видобутого повідомлення.

Крім того, автор пропонує метод детектування наявності/відсутності вбудованого повідомлення у контейнері (дана операція припускає поінформованість стосовно змісту прихованого повідомлення), що може бути корисним при захищенні цифрового контенту за допомогою ЦВЗ. Оскільки більшість з $\aleph_{\Omega low}$ НЧ-коефіцієнтів зазнала модифікації під час кодування, просте обчислення кореляції між $ind\left(\left|\Omega\hat{}_j\right|\right)$ та W_j зумовлювало б собою нестійкість методу, оскільки малі, візуально незначущі коефіцієнти ДКП роблять внесок тієї ж

---

9) Більш точно максимально можливу зміну, яку може зазнати коефіцієнт ДКП, можна обчислити, розглянувши граничний випадок, коли первинне значення коефіцієнту ДКП потрапляє на інтервал $[\tau_b, \tau_{b+1})$, якнайближче до τ_{b+1}, (див. рис.5.42, значення індексної функції умовні) і шляхом випадкового обрання на інтервалі $[\tau_{b-1}, \tau_b)$ (якщо заміна відбувається у бік менших значень) йому було присвоєно значення τ_{b-1}. При цьому значення $\tau_{b+1}^{\approx}$ (знак "≈" означає наближеність) більше за τ_{b-1} у $\left(\frac{1+\alpha}{1-\alpha}\right)^2$ разів або на $\left[\left(\frac{1+\alpha}{1-\alpha}\right)^2 - 1\right] \cdot 100 \Big/ \left(\frac{1+\alpha}{1-\alpha}\right)^2$ відсотків. Графік залежності максимально можливої зміни коефіцієнту ДКП від параметру α наведено на рис.5.43.

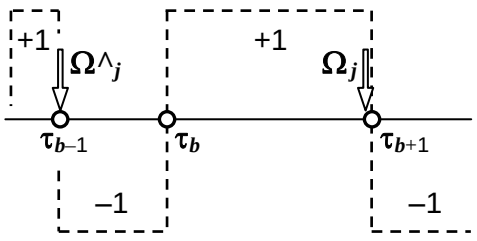


Рис.5.42. До пояснення визначення максимально можливої зміни, яку може зазнати коефіцієнт ДКП

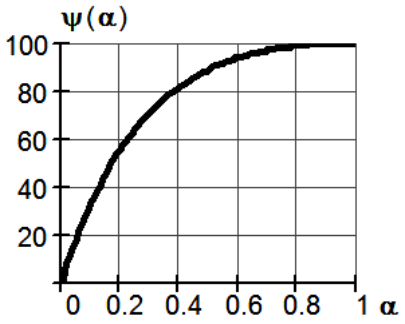


Рис.5.43. Залежність максимально можливої зміни коефіцієнту ДКП від α

ваги, що й великі, візуально більш значущі коефіцієнти. Оскільки попередньо була поставлена умова, що контейнер із вбудованим повідомленням не повинен привертати увагу, ми не можемо вбудовувати дані лише до коефіцієнтів, що мають велике значення. Крім того, позиції найбільших коефіцієнтів ДКП первинного і модифікованого зображень можуть не співпадати, що унеможливить ідентифікацію тих з них, до яких було проведено вбудовування. В пропонованій автором системі вбудовування відбувається до всіх НЧ-коефіцієнтів, незалежно від їх значення (звичайно, крім тих, що менше за 1), але лише найбільші з них враховуються при обчисленні коефіцієнту кореляції, зважуваного з енергією абсолютних значень коефіцієнтів ДКП:

$$\boldsymbol{K} = \frac{\sum_{j=1}^{\aleph_{\Omega low}} \left|\Omega_j^*\right|^{\beta} \cdot \boldsymbol{ind}\left(\left|\Omega_j^*\right|\right) \cdot \boldsymbol{W_j}}{\sum_{j=1}^{\aleph_{\Omega low}} \left|\Omega_j^*\right|^{\beta}}. \tag{5.40}$$

Таке зважування автоматично робить більш виразними найбільші значення коефіцієнтів, одночасно пригнічуючи незначні, які могли зазнати змін в результаті операцій по обробці зображення. Параметр β встановлює важливість зважування. Якщо $\beta = 0$, обчислюється звичайний, незважений коефіцієнт кореляції. Значення β, надто наближене до 1, призводить до сингулярності (виродженості) системи детектування: функція виявлення залежатиме лише від значення лише одного біту, що відповідає найбільшому коефіцієнту ДКП. Автор рекомендує використовувати $\alpha \in (0.5,1)$.

Більш стійкою до атак дану систему можна зробити шляхом пошуку максимального значення коефіцієнта кореляції відносно стандартного відхилення значень яскравостей пікселів зображення, що підозрюється на наявність вбудованого повідомлення. Масштабування (5.37) залежить від стандартного відхилення значень яскравості пікселів, яке може бути суттєво змінене, якщо зображення із вбудованим повідомленням зазнало згладжування або додавання шуму. Як наслідок, коефіцієнти ДКП такого зображення будуть промасштабовані за допомогою фіксованого коефіцієнта (відношення стандартних відхилень оригінального і досліджуваного на наявність прихованого повідомлення зображень, $\boldsymbol{d} = \sigma(\boldsymbol{C})/\sigma(\boldsymbol{S})$). Проте, повідомлення, закодоване у коефіцієнтах ДКП, лінійні зміни не торкнуться. Останнє й наводить на думку про доцільність використання простого одновимірного пошуку правильного масштабу $\boldsymbol{d}$, який би максимізував значення коефіцієнта кореляції (оскільки, як було зазначено, первинне зображення у детекторі відсутнє). Таким чином, доповнена функція детектування має наступний вигляд:

$$\boldsymbol{K'} = \max_{\boldsymbol{d} \in (1-\delta,\, 1+\delta)} \boldsymbol{K}(\boldsymbol{d}) = \frac{\sum_{j=1}^{\aleph_{\Omega low}} \left|\Omega_j^*\right|^{\beta} \cdot \boldsymbol{ind}\left(\boldsymbol{d} \cdot \left|\Omega_j^*\right|\right) \cdot \boldsymbol{W_j}}{\sum_{j=1}^{\aleph_{\Omega low}} \left|\Omega_j^*\right|^{\beta}}. \tag{5.41}$$

Автором [106] встановлено, що навіть для значних спотворень зображення в результаті атак, достатнім є крок $\delta = 0.25$.

Ускладнення при детектуванні вимагають зменшення інформаційного змісту повідомлення і додавання корегувальних біт. Таким чином, оскільки внесок у виявлення повідомлення вкладають лише найбільші коефіцієнти ДКП, інформаційний зміст повідомлення довжиною $\aleph_W$ становить собою лише певну частину від $\aleph_W$. Крім того, є очевидним, що одномірний пошук масштабного коефіцієнта, який максимізує коефіцієнт кореляції, збільшує відсоток помилкових виявлень.

Для досягнення властивостей високої стійкості до атак на стеганосистему при найменшому (наскільки це є можливим) спотворенні контейнера автором було запропоновано

вбудувати до останнього додаткове повідомлення, використовуючи методику розширення спектру. Повідомлення вбудовується шляхом додавання шумоподібного сигналу до СЧ-коефіцієнтів ДКП зображення (кількість яких $\aleph_{\Omega mid}$ становить близько 30% від загальної кількості коефіцієнтів ДКП). Вважається, що інформація, яку несе додаткове повідомлення, складається з $\aleph_{W^+}$ символів W_j^+, кожен з яких може бути представлений десятковим цілим числом, $1 \le W_j^+ \le \max(W^+)$.

Для кожного j-го символу генерується послідовність $\xi^{(j)}$ ПВЧ, рівномірно розподілених на інтервалі [0, 1]. Початковий стан генератора ПВЧ виступає в ролі секретного ключа. Потужність j-ї множини ПВЧ: $\left|\xi^{(j)}\right| \ge \aleph_{\Omega mid} + \max(W^+)$. Для представлення окремого символу повідомлення W^+, з множини $\xi^{(j)}$ виокремлюється сегмент $\eta^{(j)} = \xi_{W_j^+}^{(j)}, \ldots, \xi_{W_j^+ + \aleph_{\Omega mid} - 1}^{(j)}$, який містить $\aleph_{\Omega mid}$ елементів. В результаті, повідомлення з $\aleph_{W^+}$ символів може бути представлене у вигляді наступної суми[10]:

$$Spr = \frac{\left[\sum_{j=1}^{\aleph_{W^+}} \eta^{(j)}\right] - \frac{\aleph_{W^+}}{2}}{\sqrt{\aleph_{W^+}/12}}. \qquad (5.42)$$

Сигнал з розширеним спектром Spr має приблизно нормальний (гаусівський) розподіл з нульовим математичним очікування і одиничним стандартним відхиленням (точність апроксимації зростає при збільшенні значення $\aleph_{W^+}$). В подальшому сигнал Spr помножується на параметр γ (який регулює відношення "стійкість/помітність вбудовування") і поелементно додається до $\aleph_{\Omega mid}$ обраних СЧ-коефіцієнтів.

Видобування повідомлення W^+ проводиться шляхом попереднього обчислення коефіцієнтів ДКП зображення і виокремлення серед них середньочастотних (дана операція повинна бути узгодженою з відповідною дією на етапі вбудовування). Використовуючи секретний ключ/алгоритм, проводиться генерація послідовностей ПВЧ (загальною кількістю $\aleph_{W^+}$, якщо даний параметр є відомим; в іншому випадку – за обстановкою, виходячи з аналізу вже видобутої частини повідомлення) довжиною $\aleph_{\Omega mid} + \max(W^+)$. З кожної послідовності $\xi^{(j)}$ виокремлюється $\max(W^+)$ сегментів довжиною $\aleph_{\Omega mid}$ елементів, для яких обчислюється взаємна кореляція з вектором виокремлених СЧ-коефіцієнтів. Позиція найбільшого значення кореляції в одержаному при цьому векторі і визначатиме значення, яке мав вбудований символ W_j^+.

Розглянемо реалізацію даного методу на практиці.

---

[10] Взагалі, для формування нормального розподілу на основі рівномірного у [106] пропонується використати вираз $Spr = \frac{1}{\sqrt{\aleph_{W^+}}} \cdot \sum_{j=1}^{\aleph_{W^+}} \eta^{(j)}$, який, як відомо з теорії статистичних розподілів, не відповідає поставленим вимогам (див., наприклад, [107]).

1) Нехай контейнер **C** := **READBMP**("C.bmp"), а ЦВЗ – **W** := **READBMP**("W.bmp") (рис.5.44). При цьому **X** := **rows**(**C**), **X** = 256; **Y** := **cols**(**C**), **Y** = 256; **A** := **rows**(**W**), **A** = 16; **Z** := **cols**(**W**), **Z** = 16.

2) Для більшої стійкості стеганосистеми, контейнер розіб'ємо на блоки розмірністю **N**×**N**, **N** := 128 (див. програмний модуль (М.55)) до кожного з яких вбудовуватимемо ЦВЗ, використовуючи перший (низькочастотний) алгоритм Фрідріх. Кількість блоків – $\aleph_C := X \cdot Y \div N^2$, $\aleph_C = 4$.

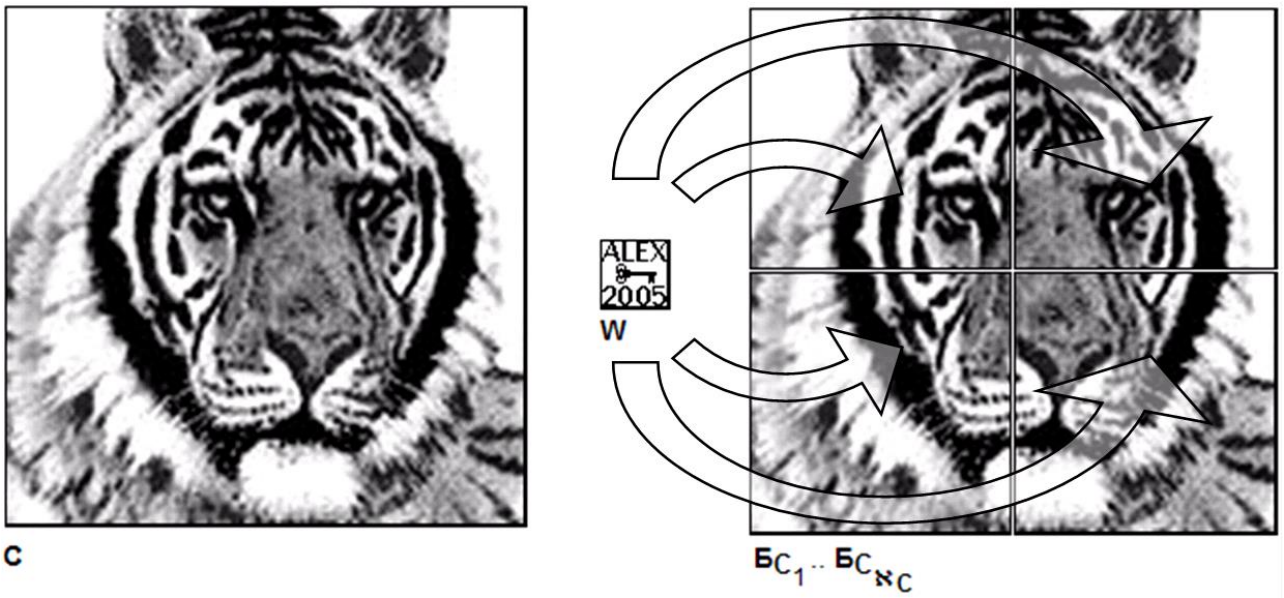


Рис.5.44. Приклад контейнера-оригіналу (***C***), ЦВЗ (***W***) і розбитого контейнера на $\aleph_C = 4$ блоки **Бс**

3) Трансформацію (5.37) реалізовано для кожного блоку окремо за допомогою програмного модуля (М.75). Функції **mean**(***M***) та **stdev**(***M***) повертають, відповідно, середнє значення та стандартне відхилення для елементів масиву ***M***.

$$БG := \left| \begin{array}{l} \textbf{for}\ \ b \in 1 .. \aleph_C \\ \quad БG_b \leftarrow \dfrac{1024}{N} \cdot \dfrac{БC_b - \textbf{mean}(БC_b)}{\textbf{stdev}(БC_b)} \\ БG \end{array} \right. \qquad \text{(М.75)}$$

Формування прогресії (5.38) здійснюється модулем (М.76). Верхня межа 300 змінної циклу **i** та переривання циклу при перевищенні елементом τ_i порогу 256 обрано умовно, виходячи з того, що в результаті трансформації (М.75) найбільші значення НЧ-коефіцієнтів не перевищуватимуть 200...250, а для проходження всіх 300 відліків значення параметру α повинне бути меншим за 0.01, що на практиці не використовується через низьку стійкість до атак одержаної за цих умов стеганосистеми. Вигляд прогресії τ при $\alpha = 0.1$ наведено на рис.5.45.

$$\tau := \left| \begin{array}{l} \textbf{for}\ \ i \in 1 .. 300 \\ \quad \tau_i \leftarrow \left(\dfrac{1+\alpha}{1-\alpha}\right)^{i-1} \\ \quad \textbf{break}\ \ \textbf{if}\ \ \tau_i > 256 \\ \tau \end{array} \right. \qquad \text{(М.76)}$$

$\tau =$

1	1	6	2.727	11	7.439	16	20.289	21	55.335	26	150.923
2	1.222	7	3.334	12	9.092	17	24.797	22	67.632	27	184.461
3	1.494	8	4.074	13	11.112	18	30.308	23	82.662	28	225.452
4	1.826	9	4.980	14	13.582	19	37.043	24	101.031	29	275.553
5	2.232	10	6.086	15	16.600	20	45.274	25	123.482	30	

Рис.5.45. Прогресія τ при параметрі $\alpha = 0.1$

Програмний модуль (М.77) визначає індексну функцію для аргументу **t**. У пропонованому варіанті остання є визначеною і для $0 \le t < 1$. Приклади індексних функцій при різних значеннях параметру α наведено на рис.5.41.

```
ind(t) := | for i ∈ 1 .. rows(τ) − 1
          |     ind ← 1      if 0 ≤ t < τ1
          |     ind ← (−1)^i  if τi ≤ t < τi+1
          | ind
```
(М.77)

4) На наступному етапі необхідно провести ДКП трансформованих блоків зображення. Очевидно, що час, необхідний для обчислення матриці ДКП одного блоку зображення за формулами (5.18), суттєво залежить від розмірності блоку. Для обчислення повної матриці ДКП використовуються два вкладених цикли перебирання індексів елементів матриці, загальна кількість яких дорівнює N^2 (див., наприклад, (М.47) і (М.49)). Для обчислення одного елементу матриці необхідно провести обчислення двох вкладених підсумовувань (N^2 операції), всередині яких обчислити аргументи двох косинусів (2·6 операцій), власне косинуси (2·1 операції) і два добутки. Крім того, одержаний по цьому результат (позначимо його як **ΣΣ**) помножується на коефіцієнт $\frac{\zeta(\upsilon)\cdot\zeta(\nu)}{\sqrt{2\cdot N}}\cdot\Sigma\Sigma$ – ще 5 операцій. Отже, загальна кількість кроків або арифметико-логічних операцій, необхідних для розв'язання даної обчислювальної проблеми (формування матриці ДКП одного блоку) складає:

$$\mathbf{O}(N) = N^2\cdot\left[5+N^2\cdot(2\cdot6+2\cdot1+2)\right] = N^2\cdot\left(5+16\cdot N^2\right). \qquad (5.43)$$

Відповідним чином зростає і час обчислення алгоритму. На рис.5.46 наведено графік залежності часу обчислення за допомогою модуля (М.47) ДКП блоку зображення розмірністю **N**×**N** від значення **N**, одержаний за допомогою обчислювальної системи наступної конфігурації: процесор Intel Pentium 4HT 2.4 ГГц, FSB 4×200 МГц, Dual DDR SDRAM 2×256 Мб.

У нашому випадку, при проведенні прямого ДКП за модулем (М.47), час обчислення всіх $\aleph_C = 4$ блоків склав би більше ніж півгодини.

Значно більш ефективний варіант обчислення коефіцієнтів ДКП реалізується через добуток матриць. При такому підході формула *прямого ДКП* може бути записана у наступному вигляді

$$\Omega = \zeta\cdot C\cdot\zeta^{\mathrm{T}}, \qquad (5.44)$$

де Ω – матриця коефіцієнтів ДКП; ζ – трансформаційна матриця ДКП розмірністю $N\times N$, елементи якої визначаються по формулі (5.45); C – матриця яскравостей пікселів зображення розмірністю $N\times N$; ζ^{T} – транспонована матриця ζ.

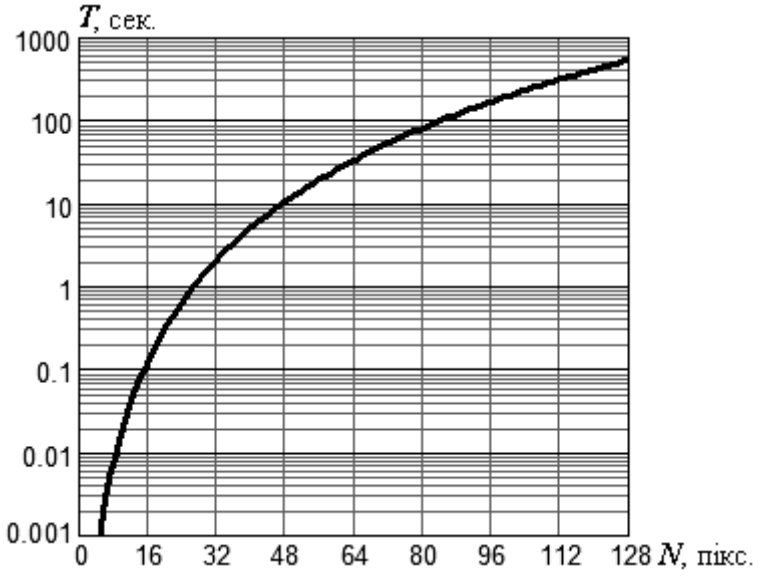


Рис.5.46. Залежність тривалості T проведення ДКП блоку від його розмірності N при використанні для обчислення модуля (М.47)

$$\zeta_{\upsilon,\nu} = \begin{cases} \frac{1}{\sqrt{N}}, \text{ при } \upsilon = 0,\ 0 \le \nu \le N-1; \\ \sqrt{\frac{2}{N}}\cdot\cos\left[\frac{\pi\cdot\upsilon\cdot(2\cdot\nu+1)}{2\cdot N}\right], \text{ при } 1 \le \upsilon \le N-1,\ 0 \le \nu \le N-1. \end{cases} \qquad (5.45)$$

У (5.44) результатом добутку матриць $\boldsymbol{\zeta}\cdot\boldsymbol{C}$ є матриця розмірністю $\boldsymbol{N}\times\boldsymbol{N}$, чиї стовпці містять результат одномірного ДКП стовпців $\boldsymbol{C}$. Добуток одержаної матриці на $\boldsymbol{\zeta}^{\mathbf{T}}$ дозволяє одержати двомірне ДКП. Оскільки $\boldsymbol{\zeta}$ є ортонормованою матрицею дійсних елементів, транспонована матриця є еквівалентною зворотній: $\boldsymbol{\zeta}^{\mathbf{T}}=\boldsymbol{\zeta}^{-1}$. Таким чином, формула *зворотного ДКП*:

$$\boldsymbol{S}=\boldsymbol{\zeta}^{-1}\cdot\boldsymbol{\Omega}\cdot\boldsymbol{\zeta}=\boldsymbol{\zeta}^{\mathbf{T}}\cdot\boldsymbol{\Omega}\cdot\boldsymbol{\zeta}. \tag{5.46}$$

При добутку двох матриць "ціна" обчислення одного елементу результуючої матриці складає $\boldsymbol{N}$ добутків і $\boldsymbol{N}$ підсумовувань, а при обчисленні повної матриці – $(\boldsymbol{N}+\boldsymbol{N})\cdot\boldsymbol{N}^2=2\cdot\boldsymbol{N}^3$. Оскільки у (5.44),(5.46) присутні по два добутки, обчислення матриці $\boldsymbol{\Omega}$ міститиме $4\cdot\boldsymbol{N}^3$ кроки:

$$\boldsymbol{O}'(\boldsymbol{N})=4\cdot\boldsymbol{N}^3. \tag{5.47}$$

Порівняно з (5.43) це суттєве підвищення швидкості обчислення. Так, у випадку $\boldsymbol{N}$ = 8 час обчислення скорочується приблизно у 32 рази, при $\boldsymbol{N}$ = 128 виграш вже становить майже 512.

Програмний модуль прямого ДКП трансформованих блоків зображення – (М.78).

```
ζ := | for υ ∈ 0 .. N – 1
     |   for ν ∈ 0 .. N – 1
     |     ζ_{υ+1,ν+1} ← 1/√N                                  if υ = 0
     |     ζ_{υ+1,ν+1} ← √(2/N) · cos[π·υ·(2·ν+1)/(2·N)]       if 1 ≤ υ ≤ N – 1
     | ζ
Ω := | for b ∈ 1 .. ℵC
     |   Ω_b ← ζ · Б_{G_b} · ζ^T
     | Ω
```

(М.78)

Графік залежності часу обчислення ДКП одного блоку за допомогою модуля (М.78) наведено на рис.5.47.

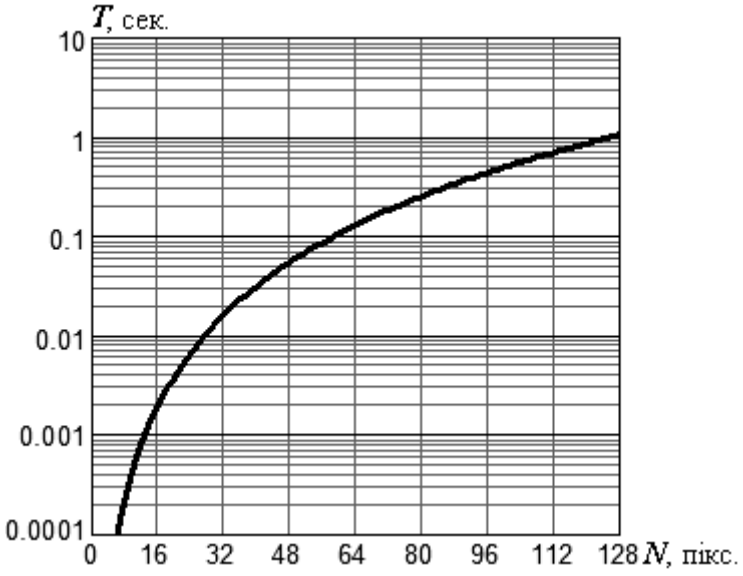


Рис.5.47. Залежність тривалості $\boldsymbol{T}$ проведення ДКП блоку від його розмірності $\boldsymbol{N}$ при використанні для обчислення модуля (М.78)

5) Для зручності вбудовування, розгорнемо масив ЦВЗ у вектор, використовуючи програмний модуль (М.56). Очевидно, що загальна кількість елементів одержаного вектора $\mathbf{W_{vec}}$ дорівнюватиме $\aleph_W := A\cdot Z$; $\aleph_W = 1024$.

6) Обрання НЧ-коефіцієнтів ДКП з матриці $\boldsymbol{\Omega_b}$ проводитимемо виходячи з того, що для елементів матриці, які знаходяться вище побічної діагоналі (НЧ-коефіцієнти) сума індексів є меншою за $N+1$, а для тих, що нижче (ВЧ-коефіцієнти) – більшою за $N+1$. Введемо мітки **L** і **H**, які визначатимуть діапазон суми індексів. Елемент матриці $\boldsymbol{\Omega_b}$, сума індексів якого не виходить за встановлені межі, обиратиметься для вбудовування. Програмний модуль пошуку мітки **H** при заданій мітці **L** (М.79)

побудований на основі підрахунку кількості елементів, сума індексів яких задовольнятиме поставленій вимозі невиходу за межі (**L**, **H**). Верхня межа **H** збільшується до моменту, коли загальна кількість вказаних елементів не перевищить загальну кількість $\aleph_W$ елементів вектора W_{vec}. Наприклад, для **L** := 40 модуль (М.79) повертає значення **H** = 62, що дозволяє використати для приховування 1050 НЧ-коефіцієнтів кожної з матриць Ω_b.

```
H := | H ← L
     | Σbit ← Σ_{i=L-1}^{H-3} (1 + i)
     | while Σbit < ℵ_W
     |     H ← H + 1
     |     Σbit ← Σ_{i=L-1}^{H-3} (1 + i)
     | H
```

(М.79)

```
Ω^ := | for b ∈ 1 .. ℵ_C
      |     j ← 1
      |     Ω^' ← Ω_b
      |     for υ ∈ 1 .. N
      |         break if j > ℵ_W
      |         for ν ∈ 1 .. N
      |             if L < (υ + ν) < H
      |                 d ← |Ω^'_{υ,ν}|
      |                 # ← 1 if Ω^'_{υ,ν} ≥ 0
      |                 # ← -1 if Ω^'_{υ,ν} < 0
      |                 if ind(d) = W_vec_j
      |                     for i ∈ 1 .. rows(τ)
      |                         if d < τ_i
      |                             if i = 1
      |                                 t1 ← 0
      |                                 t2 ← τ_i
      |                             if i > 1
      |                                 t1 ← τ_{i-1}
      |                                 t2 ← τ_i
      |                             break
      |                     if (|d - t1| < 0.01) ∨ (|d - t2| < 0.01)
      |                         Ω^'_{υ,ν} ← #·rnorm(1, (t1 + t2)/2, (t2 - t1)/13)_1
      |                 if ind(d) ≠ W_vec_j
      |                     for i ∈ 1 .. rows(τ)
      |                         if d < τ_i
      |                             if i ≤ 2
      |                                 t1 ← τ_i
      |                                 t2 ← τ_{i+1}
      |                             if i > 2
      |                                 t1 ← τ_{i-2}
      |                                 t2 ← τ_{i-1}
      |                             break
      |                     Ω^'_{υ,ν} ← #·rnorm(1, (t1 + t2)/2, (t2 - t1)/13)_1
      |                 j ← j + 1
      |             break if j > ℵ_W
      |     Ω^_b ← Ω^'
      | Ω^
```

(М.80)

Вбудовування ЦВЗ до обраних НЧ-коефіцієнтів блоків зображення реалізує програмний модуль (М.80). Якщо індексна функція **ind** від абсолютного значення **d** НЧ-коефіцієнту відповідає **j**-му елементу вектора ЦВЗ (**ind**(**d**) = $\mathbf{W_{vec_j}}$), проводиться пошук елементів прогресії $\boldsymbol{\tau}$, на інтервалі між якими знаходиться поточне значення **d**. Якщо останнє є занадто близьким до однієї з границь інтервалу (що знижує стійкість створюваної стеганосистеми), коефіцієнту присвоюється нове значення – перший елемент вектору елементів, розподілених за нормальним законом при математичному сподіванні (**t1** + **t2**)/2 і стандартному відхиленні (**t2** – **t1**)/13. Зазначене стандартне відхилення дозволяє системі бути адаптованою до різних інтервалів (τ_i, τ_{i+1}), а також запобігти випадку, коли й нове значення коефіцієнту знову виявиться близьким до границі інтервалу. У свою чергу, обрання випадкового числа з визначеного інтервалу (а не, наприклад, значення, що відповідає середині інтервалу) унеможливлює утворення підозрілих скупчень однакових значень коефіцієнтів ДКП.

Якщо індексна функція **ind**(**d**) ≠ $\mathbf{W_{vec_j}}$, коефіцієнту присвоюється випадкове значення з найближчого інтервалу – (τ_i, τ_{i+1}), при **i** ≤ 2; і (τ_{i-2}, τ_{i-1}), при **i** > 2.

На рис.5.48 схематично зображено масиви ДКП чотирьох блоків зображення – темні шумоподібні діагональні смуги у верхніх лівих кутах блоків відповідають модифікованим НЧ-коефіцієнтам, світлі елементи масивів – немодифікованим.

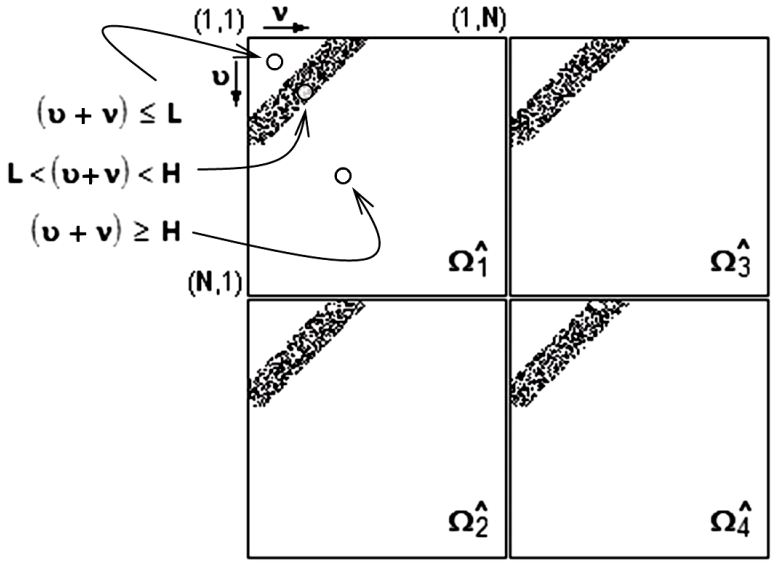


Рис.5.48. Масиви модифікованих (зображено чорним) НЧ-коефіцієнтів ДКП

7) Для реалізації середньочастотного алгоритму Фрідріх необхідно для кожного елементу ЦВЗ згенерувати послідовність рівномірно розподілених на інтервалі [0, 1] ПВЧ.

За основу генератора ПВЧ можна використати ЛРЗЗЗ, реалізований програмним модулем (М.57), який генерує дійсні числа, рівномірно розподілені в інтервалі [1, $2^{\mathbf{d}}$]. Для одержання ПВЧ, рівномірно розподілених на інтервалі [0, 1], достатньо розділити одержаний вектор **Vrnd**() на значення максимального його елементу ($2^{\mathbf{d}}$), що й виконано у модулі (М.81). Останній дозволяє

$$\begin{aligned}&\xi := \left|\begin{array}{l}\mathbf{start} \leftarrow 74\\ \mathbf{for}\ \ j \in 1..\aleph_W\\ \quad\left|\begin{array}{l}\xi' \leftarrow \mathbf{Vrnd}(\mathbf{start})\\ \mathbf{start} \leftarrow \xi'_{74}\\ \xi_j \leftarrow \dfrac{\xi'}{\mathbf{max}(\xi')}\end{array}\right.\\ \xi\end{array}\right.\end{aligned} \qquad \text{(М.81)}$$

сформувати для кожного з $\aleph_W^+ = \aleph_W$ символів ЦВЗ ПВП довжиною $2^{\mathbf{d}}$ елементів. Початковий стан (змінна **start**) генератора ПВЧ для 1-го символу ЦВЗ обрано рівним 74. В подальшому він визначатиметься значенням елементу ПВП, який має індекс 74. У ході опрацювання методу встановлено, що алгоритм розширення спектру є більш надійним, якщо відношення $\dfrac{|\xi_j|}{\aleph_W} \geq 4$.

Тому попередньо приймаємо **d** := **log**(4·$\aleph_W$, 2), **d** = 12. При цьому $2^{\mathbf{d}}$ = 4096.

Кількість СЧ-коефіцієнтів ДКП блоків зображення, до яких проводитимемо вбудовування ЦВЗ, приймемо рівною $\aleph_{\Omega mid} := 4080$.

Елемент ЦВЗ, який дорівнює «+1», представлятимемо випадковим цілим числом з інтервалу [1, 7], а елемент, що має значення «–1» – випадковим цілим з інтервалу [10, 16].

З урахуванням заданих вище початкових даних, проведемо виокремлення сегменту η_j, що представлятиме **j**-й символ ЦВЗ, для чого скористаємося програмним модулем (М.82), в якому парою функцій **round**(**runif**(...)) здійснюється генерування випадкового за рівномірним законом цілого числа на зазначеному інтервалі. За допомогою функції **submatrix**(...) виконується виокремлення сегменту довжини $\aleph_{\Omega mid}$ з вектору ξ_j.

Розділимо ЦВЗ на $\aleph_C$ частин, кожну з яких представимо у вигляді окремої суми (5.42). Дана процедура реалізована програмним модулем (М.83).

```
η := | for j ∈ 1 .. ℵW+
     |   if Wvec_j = 1
     |     m ← round(runif(1, 1, 7)_1)
     |     η_j ← submatrix(ξ_j, m, ℵΩmid + m − 1, 1, 1)
     |   if Wvec_j = −1
     |     m ← round(runif(1, 10, 16)_1)
     |     η_j ← submatrix(ξ_j, m, ℵΩmid + m − 1, 1, 1)
     | η
```

(М.82)

Результатом обчислення (М.83) є вектори $\mathbf{Spr_b}$, елементи яких повинні мати гаусівський розподіл з нульовим математичним очікування і одиничним стандартним відхиленням. У нашому випадку для кожного блоку були одержані наступні результати:

```
Spr := | for b ∈ 1 .. ℵC
       |                  (ℵW+/ℵC)·b
       |                    Σ      η_i      − (ℵW+/ℵC)·(1/2)
       |                i = (ℵW+/ℵC)·(b−1)+1
       |   Spr_b ← ─────────────────────────────────────────
       |                      √((ℵW+/ℵC)·(1/12))
       | Spr
```

(М.83)

mean(Spr_b)=	stdev(Spr_b)=
0.013	1.015
0.002	0.973
-0.001	0.976
-0.012	0.991

8) Обрання СЧ-коефіцієнтів з матриці Ω_b будемо проводити виходячи з того, що для елементів матриці, які формують побічну діагональ, сума індексів дорівнює **N**+1, сама ж діагональ складається з **N** елементів. Введемо мітки **L'** і **H'**, які визначатимуть діапазон суми індексів, потрапляння у який відноситиме розгляданий СЧ-коефіцієнт до розряду тих, в які проводитиметься вбудовування векторів $\mathbf{Spr_b}$. При $\aleph_{\Omega mid}$ = 4080, **N** = 128 одержано результати: **L'** = 111, **H'** = 147, що робить доступними для вбудовування 4174 СЧ-коефіцієнтів кожної з матриць Ω_b.

```
(L')
(H') := | L' ← N
        | H' ← N + 2
        | Σbit ← N
        | j ← 1
        | while Σbit < ℵΩmid
        |   Σbit ← Σbit + 2·(N − j)
        |   j ← j + 1
        |   L' ← L' − 1
        |   H' ← H' + 1
        | (L')
        | (H')
```

(М.84)

Вбудовування елементів векторів $\mathbf{Spr_b}$ до обраних СЧ-коефіцієнтів блоків контейнера виконує програмний модуль (М.85). Схематичне зображення результату вбудовування (як відмінності від оригінальних матриць $\mathbf{\Omega_b}$) приведене на рис.5.49.

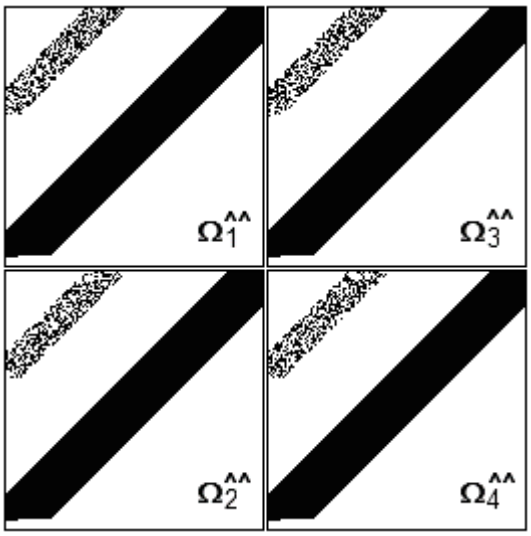


Рис.5.49. Масиви модифікованих (зображено чорним) НЧ і СЧ-коефіцієнтів ДКП

```
Ω^^ := | for b ∈ 1 .. ℵC
       |   j ← 1
       |   Ω^^' ← Ω^b
       |   for υ ∈ 1 .. N
       |     break if j > ℵΩmid
       |     for ν ∈ 1 .. N
       |       if L' < (υ + ν) < H'
       |         Ω^^'υ,ν ← Ω^^'υ,ν +
       |                   + γ·(Spr_b)_j
       |         j ← j + 1
       |       break if j > ℵΩmid
       |   Ω^^b ← Ω^^'
       | Ω^^
```
(М.85)

9) До модифікованих матриць $\mathbf{\Omega\hat{}\hat{}_b}$ застосовуємо зворотне ДКП (модуль (М.86)) і на основі відтворених блоків формуємо загальний масив контейнера (модуль (М.87)). Контейнер із вбудованим ЦВЗ зображено на рис.5.50.

```
БE := | for b ∈ 1 .. ℵC
      |   БE_b ← ζ^T · Ω^^b · ζ
      | БE
```
(М.86)

S

Рис.5.50. Зображення із вбудованим ЦВЗ при $\alpha = 0.1$ і $\gamma = 1$

```
S := | S ← БE_1
     | for b ∈ 2 .. X ÷ N
     |   S ← stack(S, БE_b)
     | S' ← 0
     | for b ∈ X ÷ N + 1 .. ℵC
     |   S' ← БE_b if S' = 0
     |   S' ← stack(S', БE_b) otherwise
     |   if mod(b, X ÷ N) = 0
     |     S ← augment(S, S')
     |     S' ← 0
     | S ← (S + |min(S)|) / max(S + |min(S)|) · 255
```
(М.87)

10) При видобуванні ЦВЗ контейнер $\mathbf{S^*} \approx \mathbf{S}$ попередньо розбивається на блоки розмірністю $\mathbf{N^* \times N^*}$, $\mathbf{N^*} := \mathbf{N}$, загальною кількістю $\mathbf{\aleph^*_C} := \mathbf{\aleph_C}$ (модуль (М.55) з відповідною підстановкою вказаних змінних).

Для кожного блоку виконується трансформація (5.37), для чого використовується програмний модуль, подібний до (М.75).

Для параметру $\alpha^* = \alpha$ формується

прогресія **τ*** і визначається індексна функція **ind*(t)**.

Виконується пряме ДКП трансформованих блоків зображення (модуль (М.78)).

Для видобування даних з матриць ДКП повинні бути заданими наступні параметри: $\aleph^*_W := \aleph_W$, $L^* := L$, $H^* := H$, $L^{*\prime} := L'$, $H^{*\prime} := H'$, $\aleph^*_{\Omega mid} := \aleph_{\Omega mid}$, $\xi^* := \xi$. Програмний модуль видобування – (М.88).

$$
\begin{array}{l}
W^*vec := \\
\quad \text{for } b \in 1 .. \aleph^*_C \\
\quad\quad j \leftarrow 1 \\
\quad\quad \text{for } \upsilon \in 1 .. N^* \\
\quad\quad\quad \text{break if } j > \aleph^*_W \\
\quad\quad\quad \text{for } \nu \in 1 .. N^* \\
\quad\quad\quad\quad \text{if } L^* < (\upsilon + \nu) < H^* \\
\quad\quad\quad\quad\quad low_j \leftarrow ind\left[\left|\left(\Omega^*_b\right)_{\upsilon,\nu}\right|\right] \\
\quad\quad\quad\quad\quad j \leftarrow j + 1 \\
\quad\quad\quad\quad \text{break if } j > \aleph^*_W \\
\quad\quad W^{*\prime}vec_b \leftarrow low \\
\quad \text{for } b \in 1 .. \aleph^*_C \\
\quad\quad j \leftarrow 1 \\
\quad\quad \text{for } \upsilon \in 1 .. N^* \\
\quad\quad\quad \text{break if } j > \aleph^*_{\Omega mid} \\
\quad\quad\quad \text{for } \nu \in 1 .. N^* \\
\quad\quad\quad\quad \text{if } L^{**} < (\upsilon + \nu) < H^{**} \\
\quad\quad\quad\quad\quad mid_j \leftarrow \left(\Omega^*_b\right)_{\upsilon,\nu} \\
\quad\quad\quad\quad\quad j \leftarrow j + 1 \\
\quad\quad\quad\quad \text{break if } j > \aleph^*_{\Omega mid} \\
\quad\quad \text{for } n \in 1 .. \dfrac{\aleph^*_W}{\aleph^*_C} \\
\quad\quad\quad \text{for } m \in 1 .. 16 \\
\quad\quad\quad\quad \eta \leftarrow submatrix\left[\xi^*_{\,n+\frac{\aleph^*_W}{\aleph^*_C}\cdot(b-1)}, m, \aleph^*_{\Omega mid} + m - 1, 1, 1\right] \\
\quad\quad\quad\quad K_m \leftarrow \sum_{i=1}^{\aleph^*_{\Omega mid}} mid_i \cdot \eta_i \\
\quad\quad\quad \text{for } m \in 1 .. 16 \\
\quad\quad\quad\quad MAX \leftarrow m \text{ if } K_m = \max(K) \\
\quad\quad\quad V_n \leftarrow 1 \text{ if } MAX < 8 \\
\quad\quad\quad V_n \leftarrow -1 \text{ if } MAX > 8 \\
\quad\quad W^{*\prime}vec_{\aleph^*_C+1} \leftarrow V \text{ if } b = 1 \\
\quad\quad W^{*\prime}vec_{\aleph^*_C+1} \leftarrow stack\left(W^{*\prime}vec_{\aleph^*_C+1}, V\right) \text{ if } b > 1 \\
\quad \text{for } j \in 1 .. \aleph^*_W \\
\quad\quad \text{for } b \in 1 .. \aleph^*_C + 1 \\
\quad\quad\quad \omega_b \leftarrow \left(W^{*\prime}vec_b\right)_j \\
\quad\quad W^*vec_j \leftarrow 1 \text{ if } mean(\omega) > 0 \\
\quad\quad W^*vec_j \leftarrow -1 \text{ if } mean(\omega) < 0 \\
\quad W^*vec
\end{array}
\qquad \text{(М.88)}
$$

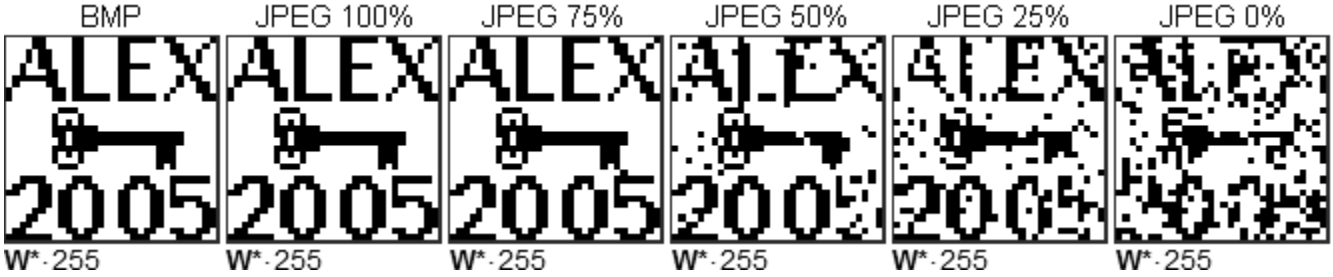
Першим циклом перебирання індексів блоків контейнера з одержаних матриць ДКП обираються НЧ-коефіцієнти, до яких було проведено вбудовування. Дані коефіцієнти виступають аргументами індексної функції, формуючи вектор **low**. Одержаний для кожного блоку результат обчислення заноситься до **b**-го елементу масиву $\mathbf{W^{*\prime}_{vec}}$.

Наступним циклом перебирання індексів блоків з матриць ДКП обираються модифіковані СЧ-коефіцієнти, які формують вектор **mid**. Попередньо знаючи, що ЦВЗ було розділено порівну між кожним блоком і що елементи ЦВЗ представлені цілими числами від 1 до 16, з елементу масиву ξ, що має індекс $\mathbf{n}+(\mathbf{b}-1)\cdot\aleph^*_{\mathbf{W}}/\aleph^*_{\mathbf{C}}$, виокремлюються всі 16 сегментів довжиною по $\aleph^*_{\Omega\mathbf{mid}}$ елементів. Для кожного з цих сегментів обчислюється взаємна кореляція з вектором **mid** виокремлених СЧ-коефіцієнтів. Проводиться пошук індексу, що відповідає найбільшому значенню кореляції. Якщо значення цього індексу є меншим/більшим за 8, робиться висновок, що вбудований елемент ЦВЗ мав значення $+1/-1$, яке й присвоюється вектору **V**. Останній має розмірність $(\aleph^*_{\mathbf{W}}/\aleph^*_{\mathbf{C}})\times 1$ і формується окремо для кожного блоку, утворюючи згодом $(\aleph^*_{\mathbf{C}}+1)$-й елемент масиву $\mathbf{W^{*\prime}_{vec}}$.

Таким чином, результуючий масив $\mathbf{W^{*\prime}_{vec}}$ складається з 5 елементів, кожен з яких, у свою чергу, є вектором розмірністю $\aleph^*_{\mathbf{W}}\times 1$, і, по суті, становлять собою 5 гіпотез щодо можливого вигляду ЦВЗ, які в ідеалі повинні бути ідентичними. Для формування єдиної гіпотези використовується "принцип більшості" – **j**-му елементу результуючого вектору $\mathbf{W^*_{vec}}$ присвоюється таке значення, яке переважає серед усіх 5 елементів з відповідним індексом. У пропонованому в модулі випадку з цих 5 елементів формується вектор $\boldsymbol{\omega}$, для якого підраховується середнє значення. Якщо останнє є додатним (кількість одиниць є більшою), елемент $\mathbf{W^*_{vec_j}} = 1$. В іншому випадку – $\mathbf{W^*_{vec_j}} = -1$.

Одержаний вектор $\mathbf{W^*_{vec}}$ згортається до масиву **W*** з розмірністю оригінального ЦВЗ (див. модуль (М.74)).

Видобуті з контейнера, який зазнав попередньої JPEG-компресії, ЦВЗ представлено на рис.5.51. У випадках *а)* і *б)* встановлена повна відповідність видобутого ЦВЗ оригіналу. Кількість помилково розпізнаних елементів ЦВЗ у випадку *в)* становила 3 пікселі (або 0.29% від загальної кількості $\aleph^*_{\mathbf{W}}$); для *г)* – 58 (5.66%); для *д)* – 109 (10.64%); для *е)* – 194 (18.945%). Збільшення параметрів α і, особливо, γ робить дану стеганосистему ще більш стійкою, але при цьому погіршується якість зображення.



а) *б)* *в)* *г)* *д)* *е)*

Рис.5.51. ЦВЗ, видобуті з контейнера **S*** ($\alpha = 0.1$, $\gamma = 1$): формат **S*** не змінено (*а*), застосована JPEG-компресія із збереженням 100% якості зображення (*б*), те саме при збереженні 75, 50, 25 і 0% якості (*в*, *г*, *д*, *е*).

Результати обчислення показників візуального спотворення контейнера при значеннях параметрів $\alpha = 0.1$, $\gamma = 1$ занесено до табл.5.4. Слід мати на увазі, що контейнер було збільшено учетверо.

Таблиця 5.4.

Показники візуального спотворення у випадку приховування даних в частотній області зображення

Назва показника спотворення	*Оригінал*	*Методи приховування у частотній області*								
		Коха-Жао (**P** = 0.5)	*Коха-Жао* (**P** = 25)	*Бенгама-Мемона* (**P** = 1)	*Бенгама-Мемона* (**P** = 35)	*Хсу-Ву (а)*	*Хсу-Ву (б)*	*Фрідріх* ($\alpha = 0.1$, $\gamma = 1$)	*Фрідріх* ($\alpha = 0.1$, $\gamma = 2$)	*Фрідріх* ($\alpha = 0.2$, $\gamma = 1$)
Макс. абсолютна різниця, ***MD***	0	39	45	45	51	122	88	68	94	73
Середня абсолютна різниця, ***AD***	0	9.504	11.392	1.846	3.042	36.805	26.457	19.233	25.990	22.413
Норм. середня абс. різниця, ***NAD***	0	0.074	0.088	0.014	0.024	0.251	0.181	0.132	0.179	0.154
Середньоквадр. помилка, ***MSE***	0	124.383	178.342	15.427	31.417	1794.673	962.154	543.072	942.113	718.449
Нормована середньоквадр. помилка, ***NMSE***	0	$5.065 \cdot 10^{-3}$	$7.263 \cdot 10^{-3}$	$6.283 \cdot 10^{-4}$	$1.279 \cdot 10^{-3}$	0.062	0.033	0.019	0.033	0.025
$\boldsymbol{L^p}$-норма, $\boldsymbol{p} = 2$	0	11.153	13.354	3.928	5.605	42.364	31.019	23.304	30.694	26.804
Лапласова середньоквадр. помилка, ***LMSE***	0	0.020	0.037	0.027	0.053	0.444	0.297	0.352	1.010	0.362
Відношення с/ш, ***SNR***	∞	197.423	137.690	$1.592 \cdot 10^{3}$	781.605	16.056	29.948	52.178	30.078	39.441
Макс. відношення с/ш, ***PSNR***	∞	522.782	364.608	$4.215 \cdot 10^{3}$	$2.070 \cdot 10^{3}$	36.232	67.583	119.735	69.020	90.507
Якість зображення, ***IF***	1	0.994935	0.992737	0.999372	0.998721	0.937717	0.966609	0.980835	0.966753	0.974646
Норм. взаємна кореляція, ***NC***	1	0.993261	0.986178	0.996790	0.994154	0.788233	0.863134	0.903486	0.873017	0.876281
Якість кореляції, ***CQ***	190.182	188.901	187.554	189.572	189.071	155.023	169.754	176.042	170.106	170.742
Структурний зміст, ***SC***	1	1.008484	1.020804	1.005826	1.010523	1.565560	1.316381	1.210451	1.283235	1.285486
Загальне сигма-відношення с/ш, ***GSSNR***	∞	87.792	60.244	$4.388 \cdot 10^{3}$	$2.167 \cdot 10^{3}$	6.326	74.183	18.153	10.080	13.899
Сигма-відношення с/ш, ***SSNR***	∞	72.9	67.3	73.3	69.2	30.1	40.8	198.8	158.1	178.9
Нормоване відношення с/помилка, ***NSER***	256	100	99	127	116	114	41	184	233	190
Подібність гістограм, ***HS***	0	11096	10714	2446	2736	15960	12712	43062	50984	45806

5.3.4. Методи розширення спектру

Початково методи розширення спектру (РС або SS – *Spread-Spectrum*) використовувалися при розробці військових систем управління і зв'язку. Під час другої світової війни в радіолокації розширення спектру використовувалося для боротьби з навмисними завадами, а в останні роки розвиток даної технології пояснюється бажанням створити ефективні системи радіозв'язку для забезпечення високої завадостійкості при передачі вузькосмужних сигналів каналами з шумами й ускладнення процесу їх перехоплення. Система зв'язку є системою з розширеним спектром у наступних випадках [65]:

1) Смуга частот, що використовується при передачі, є значно ширшою за мінімальну, необхідну для передавання поточної інформації. При цьому енергія інформаційного сигналу розширюється по всій ширині смуги частот при низькому співвідношенні сигнал/шум, в результаті чого сигнал важко виявити, перехопити або перешкодити його передачі шляхом внесення завад. Хоча повна потужність сигналу може бути великою, відношення сигнал/шум у будь-якому діапазоні частот є малим, що робить сигнал з розширеним спектром таким, що важко визначається при радіозв'язку і, в контексті приховування інформації стеганографічними методами, таким, що важко розрізнюється людиною.

2) Розширення спектра виконується за допомогою так званого розширюючого (або кодового) сигналу, який не залежить від інформації, що передається. Присутність енергії сигналу в усіх частотних діапазонах робить радіосигнал з розширеним спектром стійким до внесення завад, а інформацію, вбудовану до контейнера методом розширення спектру, стійкою до її усунення/видобування з контейнера. Компресія та інші види атак на систему зв'язку можуть усунути енергію сигналу з деяких ділянок спектра, але оскільки остання була поширена по всьому діапазону, в інших смугах залишиться достатньо даних для відновлення інформації. В результаті, якщо не розголошувати ключ, що використовувався для генерації кодового сигналу, імовірність видобування інформації неавторизованими особами суттєво знижується.

3) Відновлення початкової інформації (тобто "звуження спектру") здійснюється шляхом співставлення одержаного сигналу та синхронізованої копії кодового сигналу.

У радіозв'язку застосовують три основних способи розширення спектра:

– за допомогою прямої ПВП (РСПП);

– за допомогою стрибкоподібного перестроювання частот;

– за допомогою компресії з використанням лінійної частотної модуляції (ЛЧМ).

У першому методі інформаційний сигнал модулюється функцією, яка приймає псевдовипадкові значення у встановлених межах, і помножується на часову константу – частоту (швидкість) слідування елементарних посилок (елементів сигналу). Даний псевдовипадковий сигнал містить складові на всіх частотах, які, при їх розширенні, модулюють енергію сигналу на широкому діапазоні частот. В методі розширення спектру шляхом стрибкоподібного перестроювання частот передавач миттєво змінює одну частоту несучого сигналу на іншу. Секретний ключ при цьому – псевдовипадковий закон зміни частот. При компресії з використанням ЛЧМ сигнал модулюється функцією, частота якої змінюється у часі. Очевидно, що вказані методи можуть бути поширені на використання в просторовій області при побудові стеганографічних систем.

Розглянемо один з варіантів реалізації методу РСПП, авторами якого є *J.R. Smith* та *B.O. Comiskey* [88]. Алгоритм модуляції наступний: кожен біт повідомлення m_i представляється деякою базисною функцією φ_i розмірності $X \times Y$, помноженою, в залежності від значення біта (1/0), на +1 або –1:

$$E(x,y) = \sum_i m_i \cdot \varphi_i(x,y) . \qquad (5.47)$$

Модульоване повідомлення $E(x, y)$, одержане при цьому, попіксельно додається до зображення-контейнера $C(x, y)$, в якості якого використовується напівтонове зображення розміром $X \times Y$. Результатом є стеганозображення $S(x, y) = C(x, y) + E(x, y)$, $x \in 1...X$, $y \in 1...Y$.

Для унеможливлення спотворення вже вбудованого біту повідомлення, базисні функції повинні бути взаємно ортогональними:

$$\left\langle \varphi_i, \varphi_j \right\rangle = \sum_{x,y}^{X,Y} \varphi_i(x,y) \cdot \varphi_j(x,y) = n_\varphi \cdot G^2 \cdot \delta_{i,j}, \tag{5.48}$$

де n_φ – кількість значущих пікселів у базисній функції; G^2 – середня потужність на піксель, $n_\varphi \cdot G^2 = \sum_{x,y} \varphi_i^2(x,y)$; $\delta_{i,j} = \begin{cases} 1, & \text{при } i = j; \\ 0, & \text{при } i \neq j \end{cases}$ – дельта-символ Кронекера.

В ідеальному випадку, усі базисні функції φ_i повинні бути некорельовані з зображенням-контейнером C, тобто повинні бути ортогональними до нього: $\left\langle \tilde{N}, \varphi_i \right\rangle = 0 \ \forall i$. Але на практиці важко знайти контейнер, який би був повністю ортогональним до всіх базисних функцій φ_i. У такому випадку до розгляду вводиться величина похибки $\left\langle \tilde{N}, \varphi_i \right\rangle = \Delta \approx 0$, яку необхідно врахувати збільшенням потужності G^2.

Для ефективного приховання інформації необхідна значна кількість базисних функцій, ортогональних до типових зображень. Кодування ж зображень висуває протилежну вимогу: ідеальною є невелика кількість базисних функцій, які приблизно перекривають всю область зображення. Ці вимоги вступають у конфлікт, коли зображення, що містить приховану інформацію, зазнає компресії: ідеальна схема компресії не здатна повністю відобразити базиси, які використовувалися для приховування.

Базисні функції можуть бути організовані і порівняні у відповідності до таких властивостей як повна потужність, ступінь просторового розширення (або локалізації), а також ступінь просторового частотного розширення (або локалізації) [88].

У випадку РСПП модулююча функція складається з постійного коефіцієнту підсилення G (ціле число), помноженого на псевдовипадковий блок (масив) базисних функцій φ_i значень ±1. Кожен масив φ_i має індивідуальне розміщення в (x, y)-масиві. Крім того, масиви φ_i є такими, що не перетинаються (тобто є завідомо ортогональними один до одного) і перекривають (x, y)-масив без проміжків. Також вважатимемо, що всі N_φ базисні функції мають однакову кількість значущих елементів (n_φ). У цьому випадку повну потужність можна записати як

$$P \equiv \sum_{x,y}^{X,Y} \left(\sum_i G \cdot m_i \cdot \varphi_i(x,y) \right)^2 = \sum_i \sum_{x,y}^{X,Y} \left(G \cdot m_i \cdot \varphi_i(x,y) \right)^2 = G^2 \cdot X \cdot Y = N_\varphi \cdot n_\varphi \cdot G^2. \tag{5.49}$$

На етапі видобування інформації немає потреби знати первинний контейнер C. Операція декодування полягає у відновленні прихованого повідомлення шляхом проектування одержаного стеганозображення S^* на усі базисні функції φ_i:

$$\sigma_i = \left\langle S^*, \varphi_i \right\rangle = m_i \cdot n_\varphi \cdot G^2.$$

Значення m_i можуть бути легко відновлені за допомогою знакової функції:

$$m_i^* = sign(\sigma_i) = \begin{cases} -1, & \text{при } \sigma_i < 0; \\ 1, & \text{при } \sigma_i > 0; \\ ?, & \text{при } \sigma_i = 0, \end{cases} \quad \text{за умови, що } G^2 >> 0. \tag{5.50}$$

Якщо $\sigma_i = 0$, то приховувана інформація була втрачена. За малих значень середньої потужності G^2 зростає імовірність видобування помилкового значення біта інформації, але при цьому менше спотворюється зображення.

Основна перевага стеганографічних методів на основі розширення спектру – порівняно висока стійкість до різного виду атак на зображення і, оскільки приховувана інформація розподілена в широкій смузі частот і її важко видалити без повного руйнування контейнера. Спотворення стеганозображення збільшують значення Δ і, якщо $|\Delta| < |n_\varphi \cdot G^2|$, то приховане повідомлення не постраждає.

Наведемо приклад реалізації методу стеганографічного приховання шляхом розширення спектру у програмі MathCAD.

1) Імпортуємо зображення-контейнер: **C** := **READBMP**("C.bmp"); **X** := **rows**(**C**); **X** = 128; **Y** := **cols**(**C**); **Y** := 128.

2) Формуємо масив **Φ** ортогональних базисних функцій, який повинен мати розмірність сигналу-контейнера (**X**×**Y**) і становити собою суму всіх базисних функцій ϕ_i, які не перекриваються. Очевидно, що для одержання ортогональних базисних функцій ϕ_i, достатньо провести поділ масиву **Φ** на неперетинні сегменти, кожен з яких помістити у масив нульових елементів розмірністю **X**×**Y** за тими самими координатами. Генерування масиву **Φ** здійснюється програмним модулем (М.89). При цьому +1 і –1 чергуються у шаховому порядку. Звичайно, при створенні більш надійної стеганосистеми слід обрати більш складний спосіб формування масиву **Φ**.

```
Φ := | d ← −1
     | for x ∈ 1 .. X
     |   | d ← −d
     |   | for y ∈ 1 .. Y
     |   |   | Φ[x,y] ← d
     |   |   | d ← −d
     | Φ
```
(М.89)

Нехай загальна кількість базисних функцій N_ϕ := 256.

3) Розмірність значущого підмасиву окремої базисної функції (розмірність виокремлюваного сегменту) визначимо, виходячи із розмірності масиву **Φ** та загальної кількості базисних функцій N_ϕ:

$$n := \mathrm{floor}\left(\sqrt{\frac{X \cdot Y}{N_\phi}}\right); \quad n = 8.$$

Формування масиву базисних функцій (ϕ) здійснюємо шляхом виокремлення відповідного базисній функції значущого підмасиву $n \times n$ із загального масиву **Φ** і наступного його вбудовування у відповідні позиції нульового масиву розмірністю **X**×**Y** за допомогою функції **putregion**(...) – програмний модуль (М.90). Для спрощення, нульовий масив формується помноженням на 0 масиву **Φ**. Результат присвоюється псевдовипадковому елементу масиву ϕ. Для генерування ПВП використовується програмний модуль (М.57) при наступних вихідних значеннях:

– примітивний поліном 8-го степеня (**d** := **log**(N_ϕ, 2), **d** = 8), наприклад, наступного виду: $\rho(x) = 1 + x^3 + x^5 + x^7 + x^8$, що означає $\mu \leftarrow (0\ 0\ 1\ 0\ 1\ 0\ 1\ 1)^T$;

– значення **s** – довільне ціле число на проміжку [1; N_ϕ).

```
φ := | ℑ ← Vrnd(s)
     | c1 ← 1
     | c2 ← n
     | for i ∈ 1 .. N_φ
     |   | r1 ← mod(⌈n·(i − 1) + 1, X⌉)
     |   | r2 ← r1 + n − 1
     |   | φ[ℑ_i] ← putregion(0·Φ, submatrix(Φ, r1, r2, c1, c2), r1, c1)
     |   | if r2 = X
     |   |   | c1 ← c1 + n
     |   |   | c2 ← c2 + n
     | φ
```
(М.90)

Графічне відображення масиву **Φ**, та деякі з масивів базисних функцій наведено на рис.5.52.

4) Розглянемо ступінь ортогональності сигналу контейнера **C** до одержаних базисних функцій ϕ_i, для чого використаємо програмний модуль (М.77).

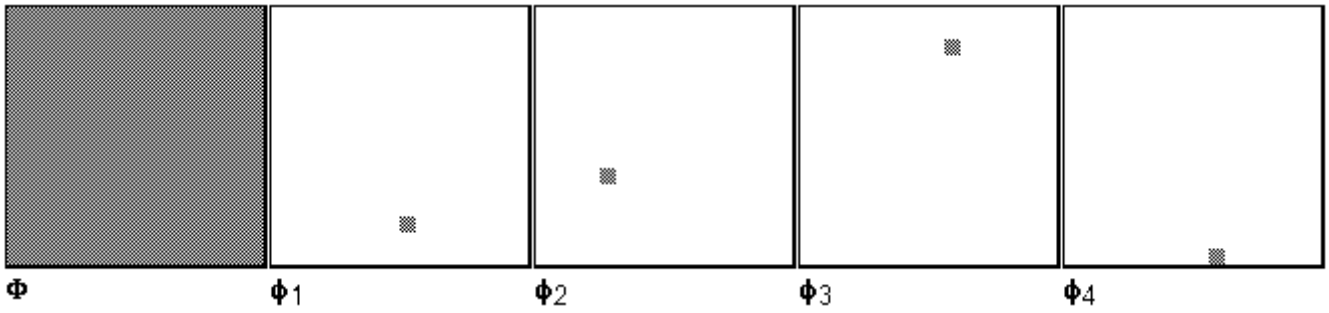


Рис.5.52. Графічна інтерпретація масивів ортогональних базисних функцій.

Результат обчислення модуля (М.91) зображено на рис.5.53. При обраному контейнері та алгоритмі формування масиву **Φ** максимальне абсолютне значення похибки ортогональності **Δ** складає 312.

$$\Delta := \left| \begin{array}{l} \text{for } i \in 1..N_{\phi} \\ \quad \Delta_i \leftarrow \sum_{x=1}^{X} \sum_{y=1}^{Y} (\phi_i)_{x,y} \cdot C_{x,y} \\ \Delta \end{array} \right. \qquad \text{(M.91)}$$

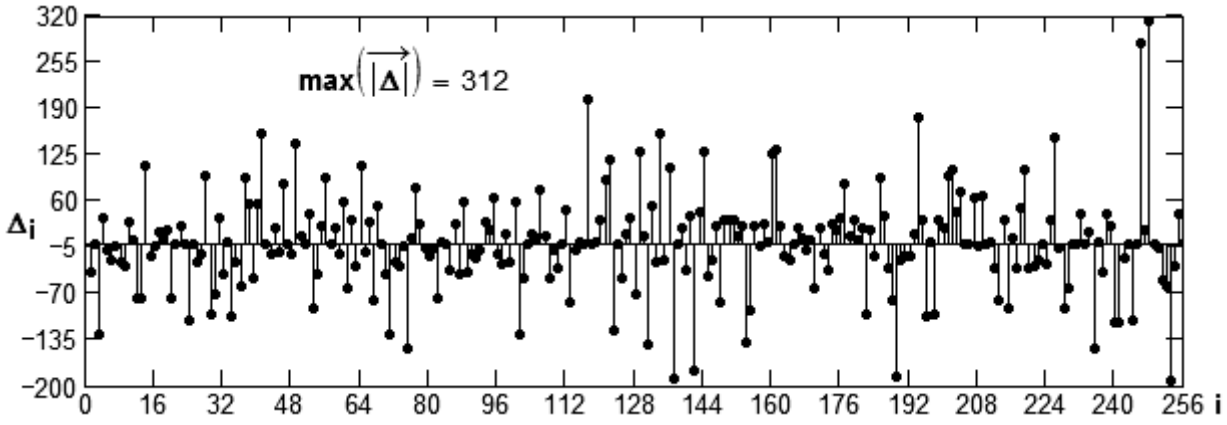


Рис.5.53. Ступінь ортогональності масиву контейнера **C** до базисної функції ϕ_i

5) Нехай повідомлення має наступний зміст: **M** := "© Пузиренко О.Ю., 2005 р.", двійкова довжина якого складає L_M = 200 біт, що є меншим за загальну кількість базисних функцій N_ϕ.

Проведемо модуляцію повідомлення базисними функціями (див. формулу (5.37)), попередньо присвоївши елементам двійкового вектора повідомлення, які мали значення 0, значення –1 – програмний модуль (М.92). Результат модуляції наведено на рис.5.54. Наявність суцільно чорних сегментів **n** × **n** говорить про те, що деякі базисні функції не брали участі в модуляції через відсутність у повідомленні бітів з відповідними їм індексами (у даному випадку кількість таких базисних функцій: $N_\phi - L_M = 56$).

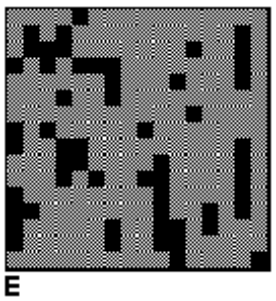


Рис.5.54. Приклад модульованого повідомлення

$$E := \left| \begin{array}{l} \text{Ⓘ -- див. (М.22), крім } S \leftarrow C \\ \text{for } j \in 1..L_M \\ \quad M_{vec\_bin_j} \leftarrow -1 \text{ if } M_{vec\_bin_j} = 0 \\ (*)\left\{ \begin{array}{l} \text{for } x \in 1..X \\ \quad \text{for } y \in 1..Y \\ \qquad E_{x,y} \leftarrow \sum_{i=1}^{L_M} M_{vec\_bin_i} \cdot (\phi_i)_{x,y} \end{array} \right. \\ E \end{array} \right. \qquad \text{(M.92)}$$

З урахуванням максимального значення похибки Δ, проведемо обчислення достатнього коефіцієнту підсилення потужності вбудовуваного до контейнера промодульованого повідомлення. Для цього використаємо спеціальний обчислювальний модуль розв'язку нерівностей (М.93), що відкривається директивою **Given**, після якої міститься логічна нерівність, у виконанні якої ми зацікавлені. Функція **Find**(K_G) повертає значення змінної K_G, попередньо заданої як $K_G := 1$, для точного розв'язку нерівності.

$$\begin{aligned} &K_G := 1; \\ &n_\phi := n^2; \\ &\textbf{Given} \quad n_\phi \cdot K_G > \max\left(\overrightarrow{|\Delta|}\right); \\ &\qquad K_G := \text{ceil}\left(\text{Find}\left(K_G\right)\right),\ K_G = 5. \end{aligned} \tag{М.93}$$

З урахуванням коефіцієнту K_G проводимо попереднє нормування масиву контейнера (інакше можливі випадки, коли значення яскравості окремого пікселя вийде за межі [0, 255]; при цьому яскравість λ обчислюється функцією запису до файлу зображення як значення $\lambda' = \textbf{mod}[\textbf{mod}(\lambda, 256) + 256, 256]$, що розумітиме під собою відчутний зором людини стрибок значення яскравості пікселя через весь діапазон її зміни і, очевидно, є неприпустимим[11]).

$$C_{norm} := \left| \begin{array}{l} \textbf{for}\ \ x \in 1..X \\ \quad \textbf{for}\ \ y \in 1..Y \\ \quad\quad C_{norm_{x,y}} \leftarrow \text{round}\left[\dfrac{C_{x,y}}{255}\cdot\left(255 - 2\cdot K_G\right) + K_G\right] \\ C_{norm} \end{array} \right. \tag{М.94}$$

До нормування: $\min(C) = 0$, $\max(C) = 255$; після: $\min(C_{norm}) = 5$, $\max(C_{norm}) = 250$.

Очевидно, що навіть у випадку додавання до граничних значень яскравості пікселя контейнера $C_{norm_{x,y}}$ елементу модульованого певною базисною функцією повідомлення, який може приймати значення $\pm K_G \cdot E_{x,y}$, яскравість пікселя заповненого контейнера $S_{x,y}$ не вийде за припустимі межі: $S := C_{norm} + K_G \cdot E$; $\min(S) = 0$, $\max(S) = 255$ (рис.5.55).

6) Для видобування повідомлення повинні бути відомими: стеганоконтейнер **S***, його розмірність **X*** і **Y***, загальна кількість базисних функцій N^*_ϕ, конфігурація (**n***) і алгоритм одержання базисних функцій **φ***. Програмний модуль видобування прихованого шляхом РСПП повідомлення – (М.95).

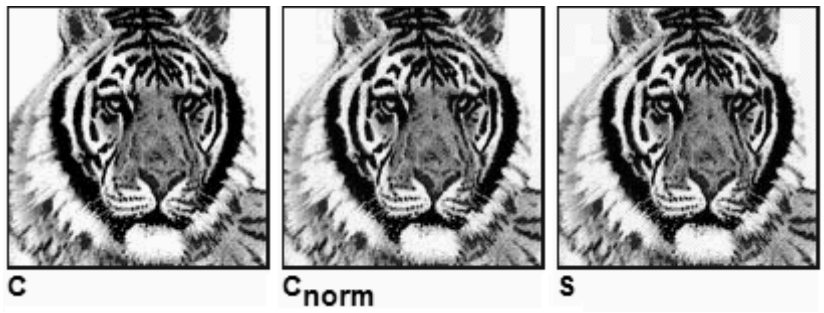


Рис.5.55. Пустий (**C**), нормований (C_{norm}) і заповнений при $K_G = 5$ (**S**) контейнери

Результат обчислення показників спотвореності контейнера зведено до таблиці 5.5.

$$\begin{aligned} M^* := \Bigg| \ & \textbf{for}\ \ i \in 1..N^*_\phi \\ & \left| \begin{array}{l} \mu \leftarrow \displaystyle\sum_{x=1}^{X^*}\sum_{y=1}^{Y^*} S^*_{x,y}\cdot\left(\phi^* i\right)_{x,y} \\ M^*vec\_bin_i \leftarrow 1 \quad \textbf{if}\ \ \mu > 0 \\ M^*vec\_bin_i \leftarrow 0 \quad \textbf{if}\ \ \mu < 0 \\ M^*vec\_bin_i \leftarrow \text{round}(\text{rnd}(1)) \quad \textbf{if}\ \ \mu = 0 \end{array} \right\} (*) \\ & \textbf{for}\ \ j \in 1..\text{rows}\left(M^*vec\_bin\right) \div 8 \\ & \quad M^*vec_j \leftarrow \text{B2D}\left(\text{submatrix}\left(M^*vec\_bin, 8\cdot j - 7, 8\cdot j, 1, 1\right)\right) \\ & M^*vec \end{aligned} \tag{М.95}$$

---

[11] Наприклад, при $\lambda = -8$ $\lambda' = 248$, при $\lambda = 259$ $\lambda' = 3$.

Більш ефективний алгоритм реалізації методу РСПП, на думку авторів [88], полягає у використанні “подвійного каналу” вбудовування: кожна базисна функція ϕ_i заздалегідь обумовленим чином ділиться на дві рівні частини (значущі), які модулюються відповідно (–1)(+1) для вбудовування «0», і (+1)(--1) для вбудовування «1». При видобуванні повідомлення з контейнера, кожен біт є результатом подвійного демодулювання – для випадків (–1)(+1) і (+1)(–1) проводиться обчислення значень кореляції. Біт повідомлення («0» або «1») визначається більшим значенням кореляції.

Реалізація наведеного алгоритму вимагає внесення змін до блоків “(*)” програмних модулів вбудовування (М.92) і видобування (М.95). Можливий варіант заміни представлено фрагментами програмних модулів (М.96), (М.97).

Як при вбудовуванні, так і під час видобування, для поділу значущих елементів базисних функцій на дві підмножини використовується значення змінної-лічильника **V** (у першому випадку – як вектора).

Значення показників спотвореності контейнера у разі використання даного алгоритму занесені до табл.5.5.

```
E ← C·0
V ← M_vec_bin·0
for x ∈ 1 .. X
  for y ∈ 1 .. Y
    for j ∈ 1 .. L_M
      if M_vec_bin_j = −1
        E_x,y ← E_x,y − (φ_j)_x,y   if V_j < n²/2
        E_x,y ← E_x,y + (φ_j)_x,y   if V_j ≥ n²/2
        V_j ← V_j + 1   if (φ_j)_x,y ≠ 0
      if M_vec_bin_j = 1
        E_x,y ← E_x,y + (φ_j)_x,y   if V_j < n²/2
        E_x,y ← E_x,y − (φ_j)_x,y   if V_j ≥ n²/2
        V_j ← V_j + 1   if (φ_j)_x,y ≠ 0
```

(М.96)

```
μ−− ← 0
μ++ ← 0
V ← 0
for x ∈ 1 .. X*
  for y ∈ 1 .. Y*
    if V < n*²/2
      μ−− ← μ−− − S*_x,y·(φ*i)_x,y
      μ++ ← μ++ + S*_x,y·(φ*i)_x,y
    if V ≥ n*²/2
      μ−− ← μ−− + S*_x,y·(φ*i)_x,y
      μ++ ← μ++ − S*_x,y·(φ*i)_x,y
    V ← V + 1   if (φ*i)_x,y ≠ 0
M*vec_bin_i ← 1   if μ++ > μ−−
M*vec_bin_i ← 0   if μ++ < μ−−
M*vec_bin_i ← round(rnd(1))   if μ−− = μ++
```

(М.97)

Таблиця 5.5.

Показники візуального спотворення у випадку приховування даних методом розширення спектру

Назва показника спотворення	*Оригінал*	*РСПП-1*	*РСПП-2*
Максимальна абсолютна різниця, ***MD***	0	10	10
Середня абсолютна різниця, ***AD***	0	4.614	4.624
Нормована середня абсолютна різниця, ***NAD***	0	0.031	0.032
Середньоквадратична помилка, ***MSE***	0	31.721	31.817
Нормована середньоквадр. помилка, ***NMSE***	0	$1.101 \cdot 10^{-3}$	$1.104 \cdot 10^{-3}$
L^p-норма, при $p = 2$	0	5.632	5.641
Лапласова середньоквадр. помилка, ***LMSE***	0	0.113	0.102
Відношення сигнал/шум, ***SNR***	∞	908.381	905.628
Максимальне відношення сигнал/шум, ***PSNR***	∞	$2.050 \cdot 10^{3}$	$2.044 \cdot 10^{3}$
Якість зображення, ***IF***	1	0.998899	0.998896
Норм. взаємна кореляція, ***NC***	1	0.986079	0.986052
Якість кореляції, ***CQ***	196.672	193.934	193.929
Структурний зміст, ***SC***	1	1.027477	1.027529
Загальне сигма-відношення с/ш, ***GSSNR***	∞	568.540	563.058
Сигма-відношення сигнал/шум, ***SSNR***	∞	103.8	103.5
Нормоване відношення сигнал/помилка, ***NSER***	256	62	61
Подібність гістограм, ***HS***	0	5702	5736

5.3.5. Інші методи приховування даних у нерухомих зображеннях

5.3.5.1. Статистичні методи

До основи статистичних методів приховування конфіденційних даних покладена модифікація певних статистичних властивостей зображення (або його фрагментів) з наступною перевіркою статистичних гіпотез під час видобування або перевірки наявності зазначених даних у контейнері. Сутність статистичних методів зводиться до такої модифікації визначених статистичних характеристик контейнера, за якої приймальна сторона матиме можливість розпізнати заповнений контейнер від пустого.

Як і у вищерозглянутих методах, багаторозрядну статистичну стеганосистему можна отримати шляхом розбиття контейнера на достатню кількість неперетинаючихся блоків (у загальному випадку ця кількість дорівнює кількості біт l_M у приховуваному повідомленні): $b_1, ..., b_{l_M}$. У цьому випадку, окремий біт повідомлення M_i вбудовується до i-го блоку контейнера. Детектування прихованого у блоці біту реалізується шляхом використання так званої *перевірочної функції*, яка дозволяє розпізнати наявність модифікації блоку:

$$f(b_i) = \begin{cases} 1, & \text{блок } b_i \text{ було змінено;} \\ 0, & \text{блок } b_i \text{ незмінено.} \end{cases}$$

Одержання функції $\boldsymbol{f}$ є найпроблемнішою задачею при реалізації статистичного методу. Її побудова здійснюється на основі теорії перевірки статистичних гіпотез. Для роботи з даними, що мають двійковий формат, у більшості випадків проводиться оперування двома гіпотезами: основною – *"блок* $\boldsymbol{b_i}$ *незмінено"*, і альтернативною – *"блок* $\boldsymbol{b_i}$ *було змінено"*. При видобуванні прихованих даних, функцію $\boldsymbol{f}$ послідовно застосовують до всіх блоків контейнера. Якщо статистика $\boldsymbol{q}(\boldsymbol{b_i})$ розподілу елементів аналізованого блоку контейнера перевищує деяке граничне значення, то вважається, що до блоку було вбудовано «1», в протилежному випадку – «0».

Статистичні методи складно застосовувати на практиці [3,89]. Причинами цього є необхідність мати вичерпну статистику $\boldsymbol{q}(\boldsymbol{b_i})$ для контейнера-оригіналу, на основі якої приймається рішення про можливу модифікацію досліджуваного контейнера (або його блоку). Крім того, розподіл $\boldsymbol{q}(\boldsymbol{b_i})$ повинен бути заздалегідь відомим на приймальній стороні, що в більшості випадків є досить складною задачею.

I. Pitas у своїй роботі [89] пропонує використовувати статистичний метод для вбудовування до напівтонового зображення $\boldsymbol{C}$ розмірністю $\boldsymbol{X}\times\boldsymbol{Y}$ цифрового підпису $\boldsymbol{W}$ (ЦВЗ), що являє собою псевдовипадковий двійковий шаблон розмірністю $\boldsymbol{X}\times\boldsymbol{Y}$, у якому кількість "одиниць" дорівнює кількості "нулів":

$$\boldsymbol{W}=\left\{\boldsymbol{w}_{\boldsymbol{x},\boldsymbol{y}},\ \boldsymbol{x}\in\{1,2,...,\boldsymbol{X}\},\ \boldsymbol{y}\in\{1,2,...,\boldsymbol{Y}\}\right\}, \tag{5.51}$$

де $\boldsymbol{w}_{\boldsymbol{x},\boldsymbol{y}}\in\{0,1\}$.

Оригінальне зображення представляється наступним чином:

$$\boldsymbol{C}=\left\{\boldsymbol{c}_{\boldsymbol{x},\boldsymbol{y}},\ \boldsymbol{x}\in\{1,2,...,\boldsymbol{X}\},\ \boldsymbol{y}\in\{1,2,...,\boldsymbol{Y}\}\right\}, \tag{5.52}$$

де $\boldsymbol{c}_{\boldsymbol{x},\boldsymbol{y}}\in\{0,1,...,255\}$ – рівень інтенсивності (яскравості) пікселя $(\boldsymbol{x},\boldsymbol{y})$.

Множина $\boldsymbol{C}$ розділяється на дві підмножини рівної потужності $\boldsymbol{P}=\boldsymbol{X}\times\boldsymbol{Y}/2$:

$$\boldsymbol{A}=\left\{\boldsymbol{c}_{\boldsymbol{x},\boldsymbol{y}}\in\boldsymbol{C},\ \boldsymbol{w}_{\boldsymbol{x},\boldsymbol{y}}=1\right\}; \tag{5.53}$$

$$\boldsymbol{Z}=\left\{\boldsymbol{c}_{\boldsymbol{x},\boldsymbol{y}}\in\boldsymbol{C},\ \boldsymbol{w}_{\boldsymbol{x},\boldsymbol{y}}=0\right\}. \tag{5.54}$$

Вбудовування ЦВЗ $\boldsymbol{W}$ здійснюється шляхом зміни елементів підмножини A на додатній цілий коефіцієнт $\boldsymbol{k}>0$:

$$\boldsymbol{V}=\left\{\boldsymbol{c}_{\boldsymbol{x},\boldsymbol{y}}+\boldsymbol{k},\ \boldsymbol{c}_{\boldsymbol{x},\boldsymbol{y}}\in\boldsymbol{A}\right\}. \tag{5.55}$$

Зображення з вбудованим ЦВЗ одержується шляхом об'єднання двох множин:

$$\boldsymbol{S}=\boldsymbol{V}\cup\boldsymbol{Z}\ . \tag{5.56}$$

Незмінність зорового сприйняття зображення (непомітність вбудованих сторонніх даних) зумовлюється тим, що величина $\boldsymbol{k}$, яка додається до яскравості пікселя $\boldsymbol{c}_{\boldsymbol{x},\boldsymbol{y}}\in\boldsymbol{A}$ для одержання множини $\boldsymbol{V}$, зазвичай є достатньо малою (з урахуванням $\boldsymbol{k}/\boldsymbol{c}_{\boldsymbol{x},\boldsymbol{y}}\to 0$).

Автори [89] вказують на можливість доволі точного виявлення вбудованої інформації, шляхом дослідження змін, викликаних вбудовуванням. Головна ідея при цьому – експертиза відмінності середніх значень (математичних очікувань) $\bar{\boldsymbol{v}}$ і $\bar{\boldsymbol{z}}$ двох виокремлених підмасивів зображення $\boldsymbol{V}$ і $\boldsymbol{Z}$. До результатів обчислення різниці $\bar{\boldsymbol{u}}=\bar{\boldsymbol{v}}-\bar{\boldsymbol{z}}$ застосовується теорія перевірки гіпотези. Статистика, що лежить в основі критерію:

$$\boldsymbol{q}=\bar{\boldsymbol{u}}/\hat{\sigma}_{\bar{\boldsymbol{u}}}\ , \tag{5.57}$$

де $\hat{\sigma}^2_{\bar{\boldsymbol{u}}}=\left[\mathbf{var}(\boldsymbol{V})+\mathbf{var}(\boldsymbol{Z})\right]/\boldsymbol{P}$; $\mathbf{var}(\bullet)$ – оцінка дисперсії випадкових змінних у відповідній підмножині.

Основна і альтернативна гіпотези, відповідно, становлять собою:

$\mathbf{H_0}$: ЦВЗ у зображенні відсутній ($\bar{\boldsymbol{u}}=0$);

$\mathbf{H_1}$: у зображення вбудоване ЦВЗ ($\bar{\boldsymbol{u}}=\boldsymbol{k}$).

Виходячи з основної гіпотези, статистика $\boldsymbol{q}$ відповідає розподілу Стьюдента з нульовим математичним очікуванням і $(2\cdot\boldsymbol{P}-2)$ степенями свободи, який можна з достатньою точністю

апроксимувати нормальним розподілом з нульовим математичним очікуванням і одиничною дисперсією.

У випадку альтернативної гіпотези, статистика $\boldsymbol{q}$ розподілена за так званим нецентрованим розподілом Стьюдента з математичним очікуванням $\boldsymbol{k}/\hat{\boldsymbol{\sigma}}_{\bar{\boldsymbol{u}}}$. Для великого об'єму вибірки розподіл $\boldsymbol{q}$ може бути апроксимований нормальним розподілом з одиничною дисперсією та математичним очікуванням $\boldsymbol{k}/\hat{\boldsymbol{\sigma}}_{\bar{\boldsymbol{u}}}$.

Під час виявлення ЦВЗ можливі наступні помилки:

– *помилка першого роду*: прийнято рішення про наявність вбудованого ЦВЗ, у той час, як він у зображенні відсутній;

– *помилка другого роду*: наявність вбудованого ЦВЗ не встановлена, при тому, що він у зображенні присутній.

Якщо $\boldsymbol{t}_{1-\alpha}$ є $\boldsymbol{t}$-процентилем[12)], що мінімізує обидві помилки, то

$$\boldsymbol{k} = \left[2 \cdot \hat{\boldsymbol{\sigma}}_{\bar{\boldsymbol{u}}} \cdot \boldsymbol{t}_{1-\alpha} \right]. \tag{5.58}$$

Як наслідок, перед вбудовуванням ЦВЗ до контейнера існує можливість задатися ступенем достовірності $(1 - \alpha)$, з якою на стадії детектування можна зробити припущення про відсутність чи наявність вбудованого у контейнері ЦВЗ.

Таким чином, пропонується наступний алгоритм *вбудовування ЦВЗ*:

1) Підраховуються значення $\mathbf{var}(\boldsymbol{A})$ і $\mathbf{var}(\boldsymbol{Z})$, які використовуються для визначення $\hat{\boldsymbol{\sigma}}_{\bar{\boldsymbol{u}}}$, використовуючи формулу $\hat{\boldsymbol{\sigma}}_{\bar{\boldsymbol{u}}}^2 = \left[\mathbf{var}(\boldsymbol{A}) + \mathbf{var}(\boldsymbol{Z}) \right] / \boldsymbol{P}$.

2) За виразом (5.58) обчислюється значення $\boldsymbol{k}$. Проте, використане у даному виразі квантування дещо змінює ступінь достовірності до деякого значення $(1 - \alpha')$. Крім того, авторами було зроблено припущення, що $\mathbf{var}(\boldsymbol{V}) = \mathbf{var}(\boldsymbol{A})$. Це не є цілком справедливим через відсікання, що виникають у випадку, коли результат дії $\boldsymbol{c}_{\boldsymbol{x},\boldsymbol{y}} + \boldsymbol{k}$ виходить за межі дозволеного діапазону [0; 255].

3) Створюється "підписане" ЦВЗ зображення $\boldsymbol{S}$ шляхом заміни підмножини $\boldsymbol{A}$ з $\boldsymbol{C}$ на підмножину $\boldsymbol{V}$ (вирази (5.55), (5.56)).

Детектування ЦВЗ відбувається таким чином:

1) Визначаються математичні очікування $\bar{\boldsymbol{v}}$ і $\bar{\boldsymbol{z}}$ виокремлених підмасивів $\boldsymbol{V}$ і $\boldsymbol{Z}$, за якими обчислюється різниця $\bar{\boldsymbol{u}} = \bar{\boldsymbol{v}} - \bar{\boldsymbol{z}}$.

2) Визначаються оцінки дисперсії $\mathbf{var}(\boldsymbol{V})$ і $\mathbf{var}(\boldsymbol{Z})$, на основі яких проводиться розрахунок $\hat{\boldsymbol{\sigma}}_{\bar{\boldsymbol{u}}}$ (див. коментар до виразу (5.47)).

3) На підставі (5.57) створюється статистика $\boldsymbol{q}$, яка порівнюється з процентилем $\boldsymbol{t}_{1-\alpha}$. У випадку, якщо $\boldsymbol{q} < \boldsymbol{t}_{1-\alpha}$, роблять висновок про відсутність ЦВЗ у зображенні. В іншому випадку з імовірністю $(1 - \alpha)$ ЦВЗ у зображенні присутнє.

5.3.5.2. Структурні методи

Методи, що зазнали на сьогодні найбільшого поширення, в основному використовують інформаційну надлишковість на рівні пікселів (просторова область) або ж виконують перетворення в частотній області зображення. *Ю.М. Коростиль* та *М.Є. Шелест* запропонували метод, в якому приховування інформації здійснюється на *змістовному рівні* з використанням структурних та інформаційних параметрів зображення [3,91]. Пропонований метод є розвитком відомої стеганографічної технології – *семаграм*. Суть методу полягає у проведенні послідовних перетворень окремих фрагментів графічного зображення, що в остаточному підсумку призводить до формування приховуваного тексту.

[12)] Процентиль є різновидом квантиля порядку ρ одномірного розподілу – такого значення $\boldsymbol{t}_\rho$ випадкової величини $\boldsymbol{t}$, для якого $\mathrm{P}\{\boldsymbol{t} < \boldsymbol{t}_\rho\} = \boldsymbol{F}(\boldsymbol{t}_\rho) = \rho$, $(0 < \rho < 1)$. Процентилі $\boldsymbol{t}_{0.01}$, $\boldsymbol{t}_{0.02}$, ..., $\boldsymbol{t}_{0.99}$ ділять область зміни ρ на 100 інтервалів, потрапляння в які мають рівні імовірності. Більш докладно див., наприклад, [105].

У структурних методах виділяють наступні етапи стеганоперетворення:

1) Перетворення захищуваного секретного повідомлення $\boldsymbol{M}$ у цифрову форму $\boldsymbol{D_M}$, наприклад, за допомогою будь-якого криптографічного кодування, що розуміє під собою шифрування тексту з усіма відповідними атрибутами.

2) Перетворення послідовності чисел $\boldsymbol{D_M}$ у графічну структуру $\boldsymbol{G_M}$ (граф, піктограму тощо), яка тим або іншим способом піддається формальному описові.

3) Перетворення графічної структури $\boldsymbol{G_M}$ у візуальне інформаційне середовище $\boldsymbol{V_M}$ (наприклад, мультимедійне або програмне).

4) Сукупність методів і відповідних процедур, за допомогою яких формується сюжет з візуальних образів із вбудованими в них прихованими повідомленнями.

Отже, всю послідовність перетворень можна записати наступним чином: $\boldsymbol{M \Rightarrow D_M \Rightarrow G_M \Rightarrow V_M \Rightarrow S_M}$, де $\boldsymbol{S_M}$ – опис сюжету, який складається з окремих графічних образів.

5.4. Приховування даних в аудіосигналах

Особливий розвиток знайшли цифрові методи стеганографії в аудіосередовищі. Приховування даних у звукових (аудіо) сигналах є особливо перспективним, оскільки слухова система людини (ССЛ), працює у надширокому динамічному діапазоні. ССЛ сприймає більше ніж мільярд до одного у діапазоні потужності та більше ніж тисяча до одного у частотному діапазоні. Проте, гострою є і чутливість до адитивного флуктуаційного (білого) шуму. Відхилення у звуковому файлі можуть бути виявлені впритул до однієї десятимільйонної (на 70 дБ нижче за рівень зовнішніх шумів).

Не дивлячись на це, існують певні можливості для приховування інформації і в аудіосередовищі. Хоча ССЛ і має широкий динамічний діапазон, вона характеризується досить малим різницевим діапазоном. Як наслідок, гучні звуки сприяють маскуванню тихих звуків. Крім того, ССЛ не спроможна розрізнювати абсолютну фазу, розрізнюючи тільки відносну. Зрештою, існують деякі спотворення, викликані оточуючим середовищем, які є настільки звичайними, що у більшості випадків ігноруються слухачем. Подібні особливості слухового апарату дозволяють вдало використовувати аудіосередовища з метою стеганографічного захисту конфіденційної інформації. Особливий внесок у розвиток аудіостеганографії зробили *W. Bender, N. Morimoto* та ін. У подальшому викладенні даного підрозділу пропонується розглянути основні відомості і методи, викладені даними авторами у своїй роботі [14].

5.4.1. Кодування найменших значущих біт (часова область)

Кодування молодших розрядів – найпростіший спосіб включити конфіденційні дані до інших структур даних. Використовуючи звуковий сигнал, шляхом заміни НЗБ кожної точки здійснення вибірки, представленої двійковою послідовністю, можна зашифрувати великий об'єм інформації. Теоретично, ПЗ каналу складає 1 Кб/сек на 1 кГц у каналі без завад, бітова швидкість передачі даних складе 8 Кб/сек у послідовності, що оцифрована з частотою 8 кГц, та 44 Кб/сек у послідовності з частотою дискретизації 44 кГц. Платою за велику пропускну здатність каналу є відчутний на слух НЧ шум. Відчутність даного шуму безпосередньо залежить від вмісту сигналу-контейнера. Наприклад, шум глядачів під час ефіру спортивного змагання маскував би шум модифікованих шифруванням наймолодших бітів, який, проте, останній буде відчутним на слух під час гри струнного квартету. Для компенсації цієї зміни є доцільним використання *адаптивного послаблення даних*.

Головним недоліком даного методу, як і у випадку з графічним контейнером, є його слабка стійкість до сторонніх впливів. Вбудована інформація може бути зруйнована шумом у каналі, передискретизацією вибірки тощо, за виключенням випадків, коли вона вбудовується з використанням методів надлишковості. Для того, щоб бути стійкими до завад, зазначені методи зменшують швидкість передачі даних, часто на один/два порядки. На практиці метод є корисним лише в закритих, повністю цифрових середовищах, які не вимагають додаткового перетворення.

Як і у випадку із використанням в якості контейнеру файлу зображення, перед імпортом аудіоконтейнера до документу MathCAD його необхідно підготувати у відповідному звуковому (музичному) редакторі. Зазначимо, що MathCAD підтримує лише файли *WAV*-формату імпульсно-кодово-модульованих сигналів (*pulse-code modulated signals* – PCM-signals), який, проте, є одним з найпоширеніших на сьогодні. Для запобігання можливості порівняння порушником перехопленого контейнеру з наявними у нього аудіофайлами, а на цій підставі – доведення факту існування прихованого повідомлення і, можливо, видобування чи модифікації останнього, в якості контейнеру рекомендується використовувати саме унікальні (створені власноруч) записи. Підготовлений аудіоконтейнер слід помістити в поточну для формованого документу MathCAD директорію. У нашому випадку попередньо створено файл ІКМ-сигналу "C.wav".

1) Для отримання інформації стосовно WAV-файлу використовується вбудована функція MathCAD **GETWAVINFO**("файл"), де під аргументом "файл" розуміється текстовий рядок, що містить у собі ім'я файлу (або повний шлях і ім'я файлу), який передбачається використати в якості контейнеру. Зазначена функція повертає 4-елементний вектор з інформацією про файл, що виступив його аргументом. Перший елемент вектору характеризує *кількість каналів*; другий – *частоту дискретизації (Гц)*; третій – *кількість біт, якими кодується один відлік* (*кількість рівнів квантування*); четвертий – *середню кількість біт на секунду*, яку повинен обробляти аудіопрогравач, щоб програвати цей звук у реальному часі. Нижче наведено два можливих варіанти використання даної функції:

$$\mathbf{GETWAVINFO}(\text{"C.wav"}) = \begin{pmatrix} 2 \\ 22050 \\ 16 \\ 88200 \end{pmatrix}; \quad \text{або} \quad \begin{pmatrix} N_к \\ f_д \\ Q \\ B \end{pmatrix} \equiv \mathbf{GETWAVINFO}(\text{"C.wav"}) \qquad \begin{aligned} N_к &= 2 \text{ канал(-и);} \\ f_д &= 22050 \text{ Гц;} \\ Q &= 16 \text{ біт;} \\ B &= 88200 \text{ біт/сек.} \end{aligned}$$

2) Користуючись одержаною інформацією, отримаємо часовий вектор, який відповідає амплітудам звуку в окремі відліки дискретизації. Дані амплітуди можуть бути зчитані за допомогою функції **READWAV**("файл"), яка повертає масив, кожен стовпець якого являє собою окремий канал (так, для моносигналу масив міститиме лише 1 стовпець, для стерео – 2 і т.д.), а кожен рядок відповідає моменту часу, що визначається номером відліку і частотою дискретизації сигналу. Окремий елемент масиву, в залежності від рівня квантування **Q**, може приймати значення або від 0 до $2^8 - 1 = 255$ (при **Q** = 1...8), або від $-2^{16-1} = -32768$ до $2^{16-1} - 1 = 32767$ (при **Q** = 9...16).

Нехай **C** := **READWAV**("C.wav"). Фрагмент імпортованого звуку (коди відліків від 1000-го по 1010-й у десятковому і двійковому вигляді) зображено на рис.5.56.

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	1010011111011b
1004	3131	4088	110000111011b	111111111000b
1005	2446	3274	100110001110b	110011001010b
1006	1842	2485	11100110010b	100110110101b
1007	1174	1631	10010010110b	11001011111b
1008	304	494	100110000b	111101110b
1009	-953	-1127	-1110111001b	-10001100111b
1010	-2466	-3080	-100110100010b	-110000001000b

Рис.5.56. Фрагмент імпортованого аудіофайлу у вигляді масиву квантованих амплітуд

Загальна кількість відліків на кожен з каналів: $\aleph_{\text{С/кан.}} := \mathbf{rows}(\mathbf{C})$, де **rows**(**C**) – функція, що повертає кількість рядків масиву **C**. У нашому випадку: $\aleph_{\text{С/кан.}} = 20191$ відлік на канал.

Визначимо часові координати кожного з відліків: $\mathbf{n} := 1 .. \aleph_{\text{С/кан.}}$, $\mathbf{t_n} := \mathbf{n}/\mathbf{f_д}$ (сек.). Значення часових інтервалів, що відповідають наведеному вище (рис.5.56) фрагменту відліків імпортованого звуку приведені на рис.5.57.

Інтервал дискретизації $\Delta\mathbf{t} := \mathbf{f_д}^{-1} = \mathbf{t_1} - 0 = \mathbf{t_2} - \mathbf{t_1} = ... = \mathbf{t_n} - \mathbf{t_{n-1}}$. Отже, $\Delta\mathbf{t} = 45.351 \cdot 10^{-6}$ сек. Загальна тривалість аудіофайлу: $\mathbf{t_\Sigma} := \mathbf{max}(\mathbf{t}) = 915.692 \cdot 10^{-3}$ сек.

Очевидно, що розділивши загальну кількість відліків $\aleph_{\text{С/кан.}}$ на загальну тривалість звучання $\mathbf{t_\Sigma}$, ми отримаємо значення частоти дискретизації $\mathbf{f_д}$.

Маючи інформацію про часові координати кожного з відліків і відповідні цим відлікам амплітуди (у квантованому вигляді) можна зобразити "осцилограму" імпортованого аудіофайлу (рис.5.58).

3) В якості секретного повідомлення використаємо наступне: **M** := **str2vec**("© Пузиренко О.Ю., 2005 р."). Аналогічно до того, як це було показано вище (програмні модулі (М.1) і (М.2)), проведемо криптографічне кодування повідомлення, результатом чого є масив **M_cod**.

$\mathbf{t_i}$ =

$45.351 \cdot 10^{-3}$
$45.397 \cdot 10^{-3}$
$45.442 \cdot 10^{-3}$
$45.488 \cdot 10^{-3}$
$45.533 \cdot 10^{-3}$
$45.578 \cdot 10^{-3}$
$45.624 \cdot 10^{-3}$
$45.669 \cdot 10^{-3}$
$45.714 \cdot 10^{-3}$
$45.760 \cdot 10^{-3}$
$45.805 \cdot 10^{-3}$

Рис.5.57. Фрагмент відліків дискретизації імпортованого аудіофайлу

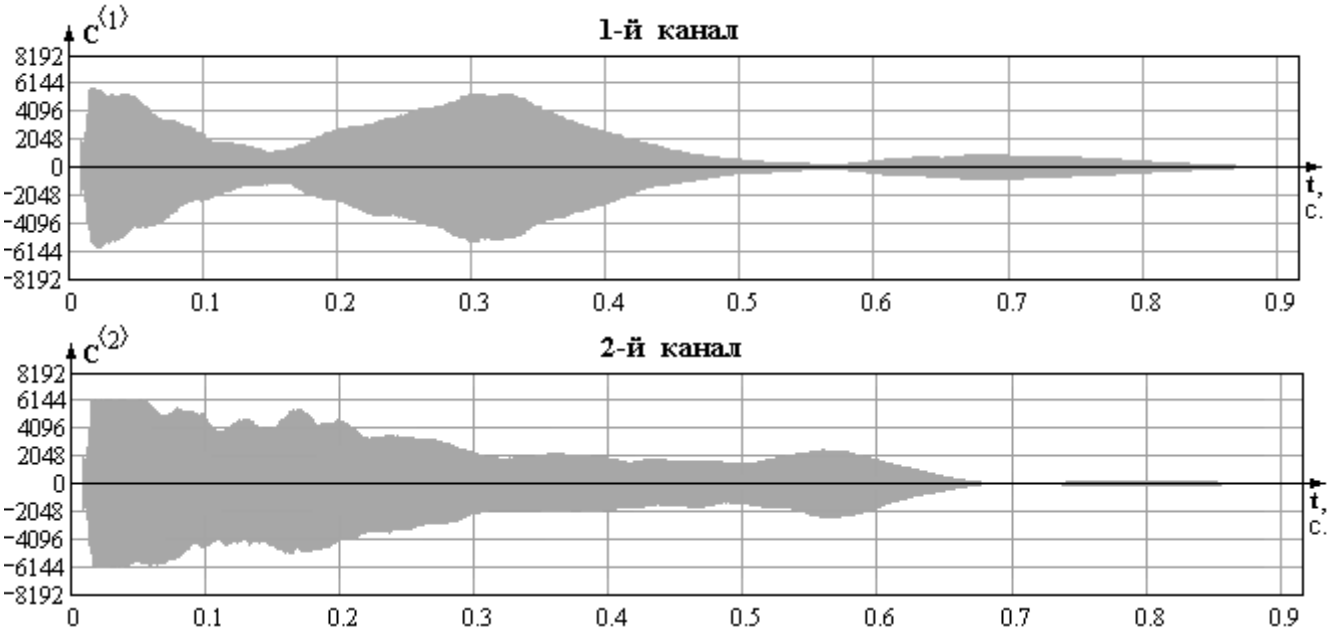


Рис.5.58. Часові діаграми каналів ІКМ сигналу "C.wav"

4) Для приховання одного символу повідомлення достатньо 8 бітів звуку. Для того, щоб при видобуванні прихованих даних з отриманого ряду символів можна було чітко визначити початок і кінець останніх, введемо відповідні мітки, які б тим або іншим чином вказували на границі саме корисного змісту. Приймемо, що стартова мітка μ_s визначає порядковий номер елементу контейнера, починаючи з якого в останній заноситимуться дані (див. нижче). Нехай $\mu_s := 74$. Мітка μ_e сигналізуватиме про завершення корисної частини серед розпакованих символів, μ_e := "KiHeu,6".

Обмежуючу мітку μ_e, перетворивши її на вектор ASCII-кодів, введемо до тексту закодованого повідомлення: **Me** := **stack**(**M_cod**, **str2vec**(μ_e)).

Таким чином, загальна кількість символів у повідомленні, що підлягає прихованню: **rows**(**Me**) = 32 симв. Кількість потрібних для цього біт (8 біт на символ): 8·**rows**(**Me**) = 256 біт.

Загальна кількість НЗБ аудіоконтейнеру: **rows(C)·cols(C)** = 40382 >> 256 біт. Отже, аудіофайл має достатній об'єм і є придатним для використання в якості контейнера для даного повідомлення.

5) Для подальших розрахунків також буде потрібним перетворення форматів чисел з десяткового на двійковий і назад. Для цього використаємо програмні модулі (М.98) і (М.99), які є подібними до (М.3) і (М.4) з тією лише різницею, що верхня межа індексування в останніх (вісімка) замінюється на **rows(x)** і **Q** відповідно. Крім того, при перетворенні формату аргументу враховується його знак.

Зазначимо, що в модулі (М.99) функція **sign(x)** повертає 0 (якщо **x** = 0); 1 (якщо **x** > 0); –1 (якщо **x** < 0). Це дозволяє зберегти знак числа **x** при переведенні його формату на двійковий.

$$B2D(x) := \sum_{i=1}^{rows(x)} \left(x_i \cdot 2^{i-1} \right) \qquad \text{(М.98)}$$

Внесення бітів повідомлення до контейнеру проводитимемо із змінним кроком, величина якого обумовлюватиметься кількістю одиниць у двійковому значенні номеру елементу контейнера, який модифікувався попередньо. Для визначення величини кроку використаємо програмний модуль (М.16).

$$\begin{aligned} &D2B(x) := \\ &\quad \text{знак} \leftarrow sign(x) \\ &\quad \text{for } i \in 1..Q \\ &\quad\quad V_i \leftarrow mod(|x|, 2) \\ &\quad\quad x \leftarrow floor\left(\frac{|x|}{2}\right) \\ &\quad \text{знак} \cdot V \end{aligned} \qquad \text{(М.99)}$$

6) Для зручності подальших дій, проведемо векторизацію масиву **C** (у разі, якщо кількість каналів у звуці більше одного) – (М.100), попередньо виконавши перестановку елементів кожного із стовпців у зворотному порядку (за допомогою функції **reverse**()), що підвищить прихованість інформації.

$$\begin{aligned} &Cv := \\ &\quad Cv \leftarrow reverse\left(C^{\langle 1 \rangle}\right) \\ &\quad \text{for } i \in 2..N_K \quad \text{if } N_K \geq 2 \\ &\quad\quad Cv \leftarrow stack\left(Cv, reverse\left(C^{\langle i \rangle}\right)\right) \\ &\quad Cv \end{aligned} \qquad \text{(М.100)}$$

На основі масиву **Cv** формуємо новий масив, який вже міститиме приховане закодоване повідомлення. Для цього застосовуємо програмний модуль (М.101). При цьому кожен символ кодованого повідомлення (операція циклу **for** $\tau \in$ 1..**rows(Me)**) переводиться у двійковий формат (змінна **b**), кожен розряд якого записується замість НЗБ числа, яке відповідає значенню амплітуди того чи іншого відліку. При цьому елементи масиву **Cv** проходяться не послідовно, а із змінним кроком, величина якого у даному випадку обумовлюється кількістю одиниць у двійковому значенні номеру елементу, який модифікувався попередньо – функція **step**(...) модуля (М.16) при **κ** = 20 (останнє повинне враховувати загальну кількість біт, що необхідна для вбудовування повідомлення, і наявну кількість елементів масиву контейнера). Стартовий елемент задається міткою μ_S.

$$\begin{aligned} &Sv := \\ &\quad Sv \leftarrow Cv \\ &\quad Z \leftarrow \mu_S \\ &\quad \text{for } \tau \in 1..rows(Me) \\ &\quad\quad b \leftarrow D2B\left(Me_\tau\right) \\ &\quad\quad \text{for } i \in 1..8 \\ &\quad\quad\quad Z \leftarrow Z + step(D2B(Z)) \\ &\quad\quad\quad \text{знак} \leftarrow 1 \quad \text{if } Cv_Z \geq 0 \\ &\quad\quad\quad \text{знак} \leftarrow -1 \quad \text{if } Cv_Z < 0 \\ &\quad\quad\quad P \leftarrow D2B\left(Cv_Z\right) \\ &\quad\quad * \quad P_1 \leftarrow \text{знак} \cdot b_i \\ &\quad\quad\quad Sv_Z \leftarrow B2D(P) \\ &\quad Sv \end{aligned} \qquad \text{(М.101)}$$

Значення елемента масиву **Cv**, до якого вестиметься запис, попередньо переводиться у двійковий формат (змінна **P**)). Також враховується знак амплітуди відліку. Після проведеної

зміни модифіковане двійкове число **P** переводиться у формат десяткового і записується у відповідну позицію вектора **Sv**, який на початку модуля був прийнятий рівним вектору **Cv.**

7) Проводимо зворотне згортання вектора **Sv** до масиву **S**, який має розмірність первинного масиву **C**. Одночасно виконуємо зворотну перестановку, повертаючи елементи відліків на свої місця:

$$\mathbf{S} := \text{for } i \in 1..\mathbf{cols}(\mathbf{C}) \quad \mathbf{S}^{\langle i \rangle} \leftarrow \mathbf{reverse}[\mathbf{submatrix}[\mathbf{Sv}, (i-1)\cdot\mathbf{rows}(\mathbf{C})+1, i\cdot\mathbf{rows}(\mathbf{C}), 1, 1]] \tag{M.102}$$

На рис.5.59 зображено часові діаграми аудіоконтейнеру, що містить приховане повідомлення, відновлені за відповідними значеннями амплітуд відліків.

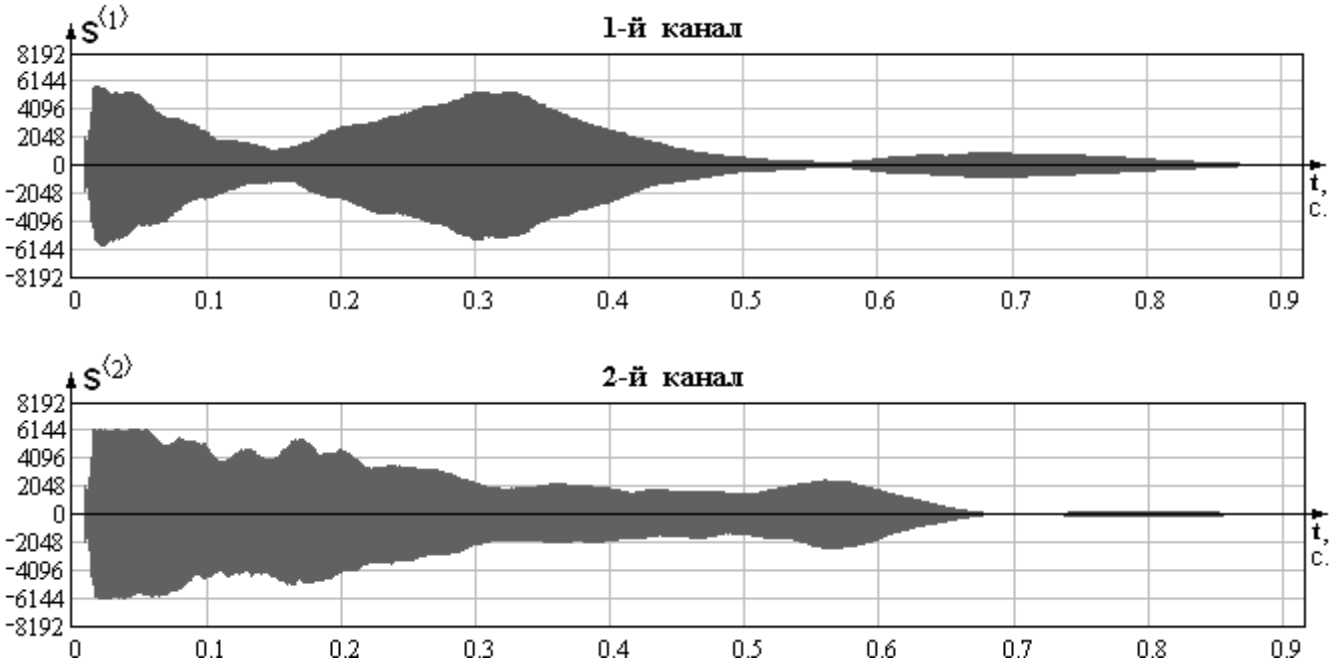


Рис.5.59. Часові діаграми каналів ІКМ сигналу з модифікованими НЗБ

Порівнюючи рис.5.59 з рис.5.58 можна зробити висновок про їх цілковиту візуальну ідентичність, а це дає певні підстави стверджувати про ідентичність заповненого і первинного контейнерів на слух. Для того, щоб переконатися в цьому необхідно записати масив **S** у файл, що можна зробити вбудованою функцією **WRITEWAV**("файл", частота дискретизації, кількість рівнів квантування):

WRITEWAV("S_LSB.wav", $\mathbf{f_д}$, **Q**) := **S.**

8) Щоб дослідити вплив на ступінь прихованості того, до якого з розрядів числа, яке характеризує рівень сигналу певного відліку, заноситиметься конфіденційна інформація, в модулі (М.101) у поміченому символом "∗" рядку замість індексу "1" слід внести індекс, який відповідає розряду, що модифікуватиметься. Пересічна людина не відчуватиме на слух різниці між звучанням оригіналу і контейнера з повідомленням, якщо в якості “носіїв” використовувати не лише наймолодший, але й наступні 1-2 біти (особливо, якщо використовується 16-бітове квантування і в аудіосигналі відсутні великі ділянки пауз або ділянки з тривалим низьким рівнем звучання). Отже, ще одним із можливих варіантів захисту є використання змінюваного за певним законом запису до цих кількох молодших бітів. Або ж, знижуючи рівень прихованості, можна в кілька разів збільшити пропускну здатність створюваного аудіостеганоканалу.

На рис.5.60 зображено наслідок внесення даних до 13-го біта числа квантованого відліку.

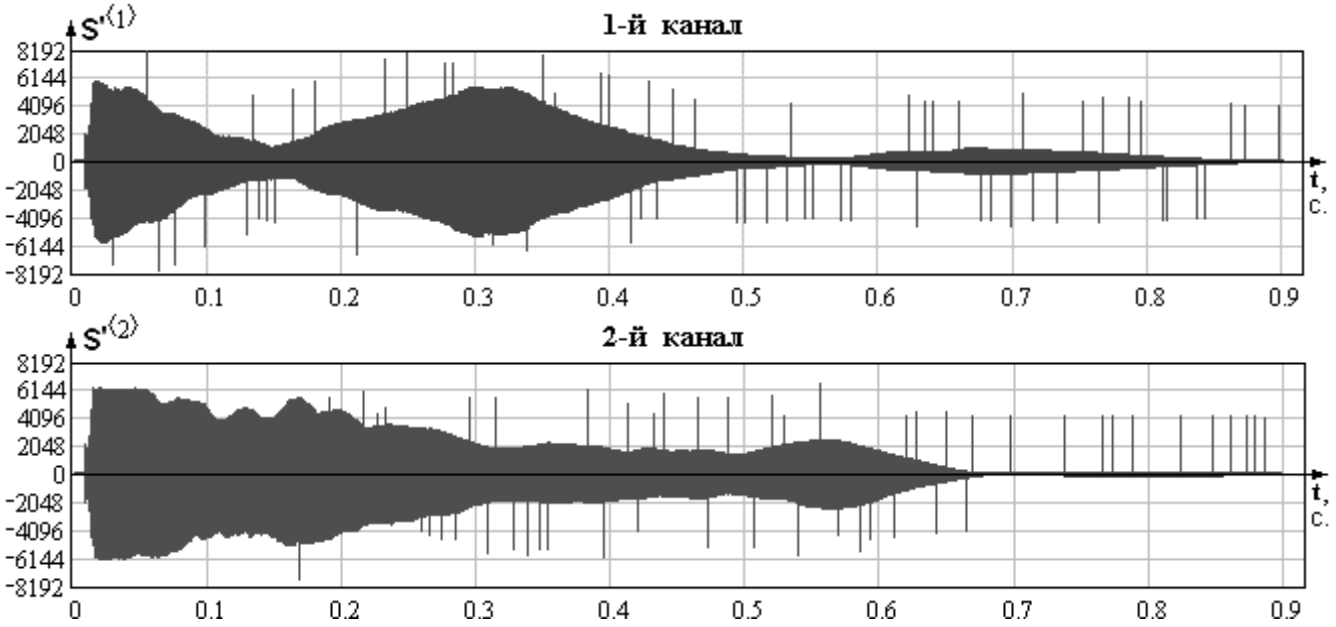


Рис.5.60. Часові діаграми каналів ІКМ сигналу при внесенні приховуваних даних до 13-го біту числа, яке характеризує рівень сигналу

9) Розглянемо процес *розпакування прихованого повідомлення*.

Формуємо масив значень амплітуд дискретних відліків, а також зчитуємо службову інформацію з аудіоконтейнеру.

$$\mathbf{S^*} := \mathbf{READWAV}(\text{"S\_LSB.wav"});$$

$$\begin{pmatrix} N_K \\ f_Д \\ Q \\ B \end{pmatrix} \equiv \mathbf{GETWAVINFO}(\text{"S\_LSB.wav"}) \qquad \begin{matrix} N_K = 2 \text{ канал(-и);} \\ f_Д = 22050 \text{ Гц;} \\ Q = 16 \text{ біт;} \\ B = 88200 \text{ біт/сек.} \end{matrix}$$

Отриманий масив **S*** розгортаємо у вектор **Sv***, змінюючи порядок елементів у стовпцях (використовується масив (М.100) з відповідними замінами **Cv** → **Sv***, **C** → **S***).

Використовуючи програмний модуль (М.103), проводимо розпакування прихованого повідомлення.

Оскільки у переважній більшості випадків одержувачеві заздалегідь невідомо, повідомлення якої довжини було приховано у контейнері, беруться до уваги всі елементи вектора **Sv***. Значення кожного елементу формованого при цьому вектора **Mf*** представляють собою коди символів гіпотетичного повідомлення, які обчислюються у зворотному до (М.101) порядку: модуль кожного молодшого розряду вісімки перетворених у двійковий формат елементів вектора **Sv*** (обраних з урахуванням змінного кроку і відомого значення мітки $\mu^*_S := \mu_S = 74$ формує двійкове число коду символу, формат якого потім перетворюється на десятковий. Отримане число присвоюється τ-му елементу вектора **Mf***.

```
Mf* := | Z ← μ*_S
       | err ← 0
       | for τ ∈ 1 .. rows(Sv*)
       |     for i ∈ 1 .. 8
       |         Z ← Z + step(D2B(Z))
       |         if Z ≥ rows(Sv*)
       |             err ← 1
       |             break
       |         P ← D2B(Sv*_Z)
       |         b_i ← |P_1|
       |     Mf*_τ ← B2D(b)
       |     Mf*_τ ← Mf*_τ + 32.5  if  Mf*_τ < 32
       |     break  if  err = 1
       | Mf*
```

(М.103)

Через зазначену вже неможливість оброблення MathCAD 12-ї версії символів, ASCII-код яких має значення від 0 до 31 включно, додатково вноситься заміна значень 0, 1, 2, ..., 31 додаванням до кожного з них коефіцієнту 32.5 (причина обрання саме такого коефіцієнту пояснена у коментарях до програмних модулів (М.10) і (М.11)).

Змінна **err** необхідна для переривання циклу в разі виходу аргументу **Z** за межі масиву **Sv***. Номери рядків вектора **Mf***, елементи яких мають дробові значення, формують масив **N** (див. модуль (М.11)).

Володіючи інформацією про те, якою є кінцева мітка корисного повідомлення (у нашому випадку $\mu^*_e = \mu_e$ = "KiHeu,6"), виокремлюємо його з видобутого повідомлення, використовуючи модуль (М.104). Пояснення до цього модуля аналогічні поясненням, що були дані до програмного модуля (М.12).

```
M_cod* := | e ← 0
          | β_e ← strlen(μ*_e)
          | Mf* ← vec2str(Mf*)
          | for τ ∈ 1 .. strlen(Mf*)
          |   | e ← τ  if  substr(Mf*, τ, β_e) = μ*_e ∧ e = 0
          |   | break  if  e ≠ 0
          | Mf* ← substr(Mf*, 0, e)
          | M_cod* ← str2vec(Mf*)
          | for n ∈ 1 .. cols(N)
          |   for i ∈ 1 .. rows(N)
          |     | break  if  N_i,n = 0
          |     | M_cod*_(N_i,n) ← n − 1  if  0 < N_i,n ≤ rows(M_cod*)
          | M_cod*
```
(М.104)

10) Розпаковане повідомлення *декодуємо*, застосовуючи модулі (М.13) і (М,14). Декодоване повідомлення записуємо до файлу: **WRITEBIN**("M_dec.txt","byte",0) := **M***.

11) Проведемо обчислення показників звукового спотворення, наведених у розділі 3. Отримані результати зведено до табл.5.6.

5.4.2. Метод фазового кодування (частотна область)

До методу фазового кодування закладено заміну фази первинного звукового сегмента на *опорну фазу*, що представляє собою дані, які необхідно приховати. Для того, щоб зберегти різницеву фазу між сегментами, фази останніх відповідним чином узгоджуються.

Фазове кодування, коли воно може бути використане, є одним з найефективніших методів кодування за критерієм відношення сигнал/сприйманий шум. Суттєва зміна співвідношення фаз між кожною частотною складовою призводить до значного розсіювання фази. Тим не менш, до тих пір, поки модифікація фази є в достатній мірі малою (звичайно, достатньо малою в залежності від спостерігача; фахівці зі спектрального аналізу здатні виявити видозміни, що здаються незначними пересічному спостерігачеві), може бути досягнуте приховання, невідчутне на слух.

Процедура фазового кодування полягає в наступному:

1) Звукова послідовність $S[i]$, $(1 \le i \le I)$ розбивається на серію N коротких сегментів (блоків) $S_n[i]$, $(1 \le n \le N)$ – рис.5.61, *а, б*.

2) До n-го сегмента сигналу $S_n[i]$ застосовується K-точкове ДПФ, де $K = I/N$, та створюються масиви фаз $\phi_n(\omega_k)$ і амплітуд $A_n(\omega_k)$ для $1 \le k \le K$ (рис.5.61, *в*).

3) Запам'ятовується різниця фаз між кожними сусідніми сегментами для $1 < n \le N$ (рис.5.61, *г*):

$$\Delta\phi_n(\omega_k) = \phi_n(\omega_k) - \phi_{n-1}(\omega_k); \quad \Delta\phi_1(\omega_k) = 0. \tag{5.59}$$

4) Двійкова послідовність даних представляється як $\phi_{data} = \pi/2$ або $\phi_{data} = -\pi/2$, відображуючи відповідно «1» або «0» (рис.5.61, *д*): $\phi'_1(\omega_k) = \phi_{data}$.

5) З урахуванням різниці фаз відтворюється новий масив фаз для $n > 1$ (рис.5.61, *е*):

$$\left\| \begin{array}{c} \phi'_1(\omega_k) = \phi_{data} \\ \phi'_2(\omega_k) = \phi'_1(\omega_k) + \Delta\phi_2(\omega_k) \\ \cdots \\ \phi'_n(\omega_k) = \phi'_{n-1}(\omega_k) + \Delta\phi_n(\omega_k) \\ \cdots \\ \phi'_N(\omega_k) = \phi'_{N-1}(\omega_k) + \Delta\phi_N(\omega_k) \end{array} \right\|. \quad (5.60)$$

6) Відновлення звукового сигналу проводиться шляхом застосування оберненого ДПФ до первинної матриці амплітуд і модифікованої матриці фаз (рис.5.61, *ж*, *з*).

Перед процесом розшифрування повинна бути проведена *синхронізація* послідовності. Одержувачу повинні бути відомими довжина сегменту, точки ДПФ і інтервал даних. Значення основної фази першого сегменту виявляється як «0» або «1», які представляють закодовану двійкову послідовність.

Оскільки фаза $\phi'_1(\omega_k)$ є зміненою, відповідним чином будуть змінені й абсолютні фази наступних сегментів. Але відносна різниця фаз кожного суміжного сегменту буде збережена. Саме до цієї відносної різниці в фазі і є найбільш чутливим людський слух.

Розкид фаз – спотворення, викликане порушенням взаємозв'язку фаз між кожною з частотних складових. Зменшення розкиду фаз обмежує швидкість передачі даних при фазовому кодуванні. Однією з причин розкиду фаз є заміщення фази $\phi'_1(\omega_k)$ на двійковий код. Для зменшення спотворень, значення модифікованої фази повинне бути близьким до її первинного значення. Для того щоб знизити чутливість вбудованих даних до шуму, повинна збільшуватися різниця між структурами модифікованої фази. У нашому випадку, значенню біта «0» відповідає $-\pi/2$, а значенню «1» $+\pi/2$.

Іншим джерелом спотворення є швидкість зміни модифікованої фази. Якщо спотворення застосоване до кожного елементу дискретизації ДПФ, це з великою імовірністю зруйнує зв'язки між фазами

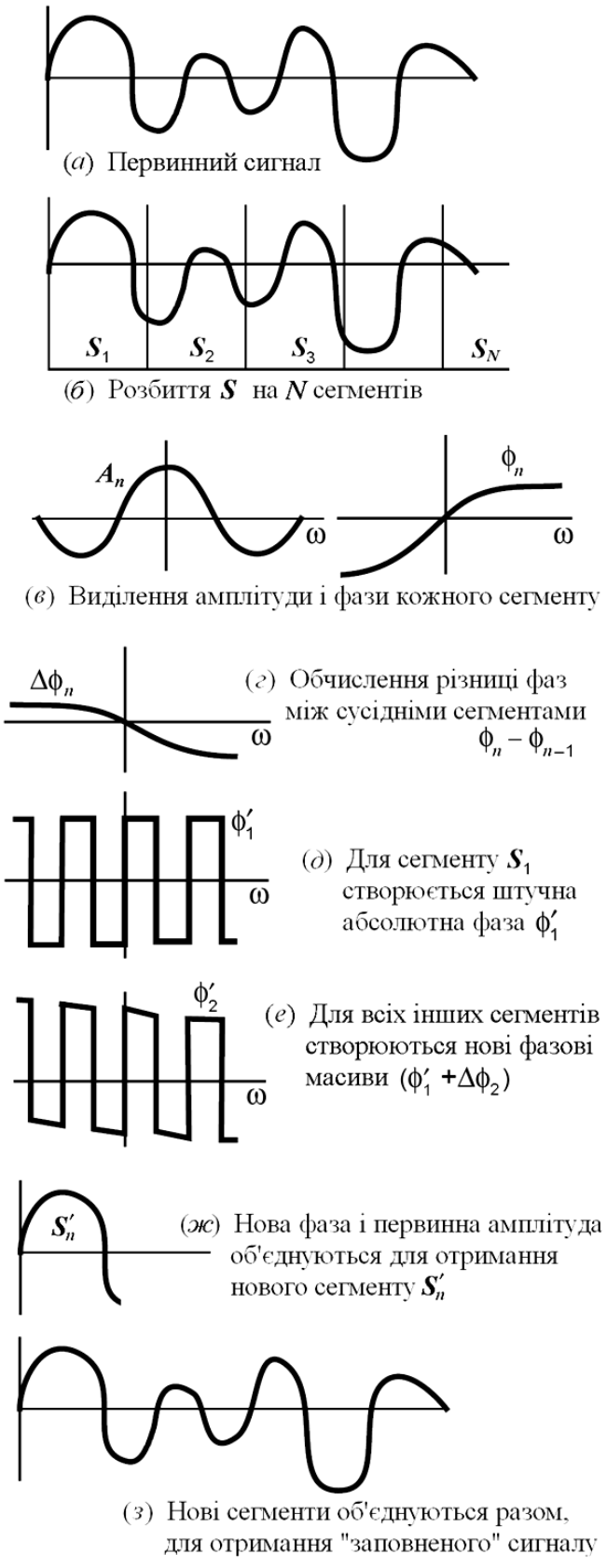


Рис.5.61. Блок-схема фазового кодування

сусідніх частотних складових, що призведе до накладання фонового биття. Шляхом більш повільної зміни фази і узгодження переходів між змінами фази, досягається суттєве зниження відчутних на слух спотворювань.

На рис.5.62 зображено різкі переходи у порівнянні зі згладженими – круті фронти фазових переходів, викликають значні спотворення контейнера (*а*); спотворення зменшуються, якщо фронти є згладженими (*б*). Слід зазначити, що в обох випадках інформаційні точки присутні в одному і тому ж місці. Така плавна зміна характеризується недоліком спричинення скорочення смуги пропускання, оскільки для того щоб зробити можливим плавний перехід, необхідно залишити місце між кожною інформаційною точкою.

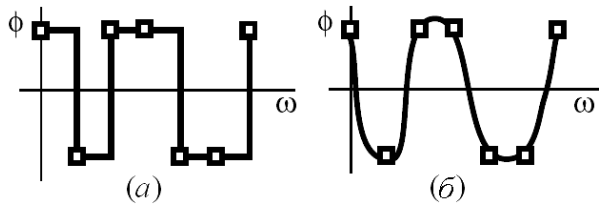


Рис.5.62. Різкі переходи у порівнянні зі згладженими

Недоліком цієї схеми є її низька пропускна здатність. В експериментах авторів [14] ПЗ каналу варіювалася від 8 до 32 біт/сек. в залежності від звукового контексту.

Нами пропонується наступна реалізація методу фазового кодування.

1) Імпортуємо файл аудіоконтейнера до масиву квантованих амплітуд дискретних відліків: **C** := **READWAV**("C.wav"). Приховувані дані вноситимемо до 1-го каналу даного звукового файлу: **S** := $\mathbf{C}^{\langle 1\rangle}$. Кількість елементів у даному контейнері: **I** := **rows**(**S**), **I** = 20191; **i** := 1.. **I.** Часова діаграма даного каналу ІКМ сигналу зображена на рис.5.58.

2) Нехай приховуване повідомлення має зміст: **M** := "© Пузиренко О.Ю., 2005 р.". Довжина повідомлення (у бітах): $\mathbf{L_M}$:= 8·**strlen**(**M**), $\mathbf{L_M}$ = 200 біт.

3) Проведемо розбиття звукової послідовності на сегменти. Кількість сегментів **N** визначається довжиною **K** окремого сегмента. Як буде показано нижче, параметр **K** повинен бути результатом возведення двійки до **ν**-го степеню, де **ν** є цілим числом і залежить від довжини приховуваного повідомлення $\mathbf{L_M}$. При цьому останнє заноситиметься до масиву фаз, який отримується в результаті обчислення для сегменту швидкого перетворення Фур'є (ШПФ). Зазначений масив має розмірність вдвічі меншу за розмірність сегменту, для якого проводилося обчислення. Отже, значення параметру **ν** можна знайти з розв'язку рівняння:

$$\textbf{Given}\ 2^{\nu} = 2\cdot \mathbf{L_M}\ ; \qquad \nu := \textbf{ceil}(\textbf{Find}(\nu)),$$

де **Given** – ключове слово (директива), яке відкриває блок розв'язання рівняння (або системи рівнянь); **Find**(**var1**, **var2**, ...) – вектор змінних **var1, var2, ...** , які дають розв'язок рівняння у блоці (кількість повертаних значень дорівнює кількості аргументів); функція **ceil**(**x**) повертає найменше ціле, що перевищує або дорівнює аргументу **x.**

Результатом розв'язку є: **ν** = 9. Отже, **K** := $2^{\nu+1}$ (степінь збільшена на одиницю для зменшення спотворюваності контейнера), **K** = 1024; **k** := 1..**K.**

Визначимо кількість сегментів **N**, на яку необхідно поділити послідовність аудіоданих: **N** := **ceil**(**I**/**K**), **N** = 20; **n** := 1..**N**; **I**/**K** = 19,718.

Якщо ціле значення **N** є більшим від результату **I**/**K**, то контейнер необхідно розширити, наприклад шляхом дописування нулів (М.105).

Таким чином, нове значення **I** := **rows**(**S**), **I** = 20480.

Остаточно визначившись з основними розмірностями, проводимо розбиття первинної звукової послідовності **S** за допомогою модуля (М.106).

$$\mathbf{S} := \left| \begin{array}{l} \textbf{for}\ \ \mathbf{i} \in 1..\mathbf{N}\cdot\mathbf{K} \\ \quad \left| \begin{array}{l} \mathbf{S_i} \leftarrow \mathbf{S_i}\ \ \textbf{if}\ \ \mathbf{i} \le \mathbf{I} \\ \mathbf{S_i} \leftarrow 0\ \ \textbf{if}\ \ \mathbf{i} > \mathbf{I} \end{array} \right. \\ \mathbf{S} \end{array} \right. \qquad \text{(М.105)}$$

$$\mathbf{s} := \left| \begin{array}{l} \textbf{for}\ \ \mathbf{n} \in 1..\mathbf{N} \\ \quad \mathbf{s_n} \leftarrow \textbf{submatrix}[\mathbf{S}, (\mathbf{n}-1)\cdot\mathbf{K}+1, \mathbf{n}\cdot\mathbf{K}, 1, 1] \\ \mathbf{s} \end{array} \right. \qquad \text{(М.106)}$$

Схема розбиття зображена на рис.5.63.

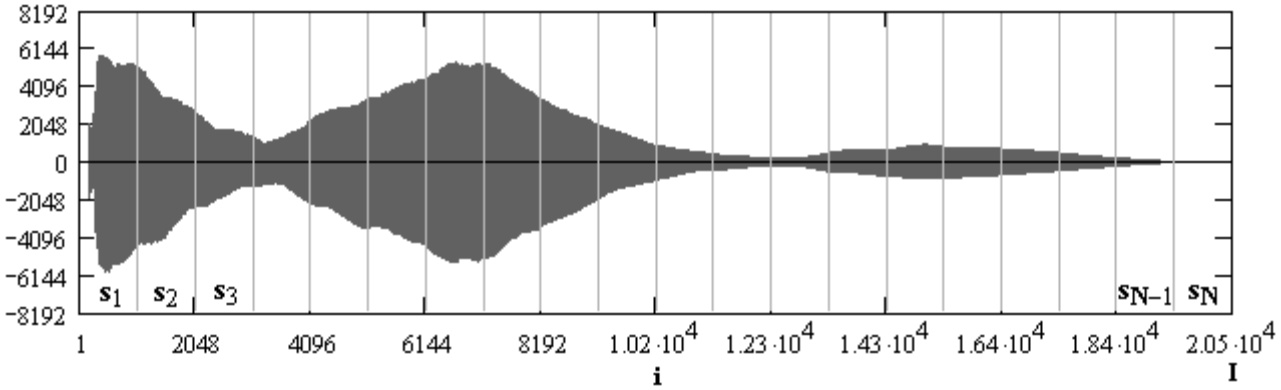


Рис.5.63. Сигнал S_i ($i = 1..I$) розбитий на N сегментів s_n ($n = 1..N$)

4) За допомогою вбудованої функції **FFT(V)** виконуємо ШПФ для даних, записаних в аргументі-векторі **V**. Останній повинен містити 2^ν елементів, де ν – ціле число. Результат виконання функції – вектор розмірністю $(2^{\nu-1} + 1)$. У нашому випадку, в якості аргументу функції виступатимуть вектори окремих сегментів s_n (М.107,*а*).

Кожен n-й елемент отриманого масиву σ містить підмасив з $[(K/2) + 1]$ елементів, що представляють собою результат обчислення ШПФ для сегменту s_n. Використовуючи відомі залежності, отримуємо масиви амплітуд і фаз (М.107, *б*, *в*).

$$\sigma := \left| \begin{array}{l} \text{for } n \in 1..N \\ \quad \sigma_n \leftarrow \text{FFT}(s_n) \\ \sigma \end{array} \right. \qquad A := \left| \begin{array}{l} \text{for } n \in 1..N \\ \quad A_n \leftarrow \overrightarrow{|\sigma_n|} \\ A \end{array} \right. \qquad \phi := \left| \begin{array}{l} \text{for } n \in 1..N \\ \quad \phi_n \leftarrow \overrightarrow{\arg(\sigma_n)} \\ \phi \end{array} \right. \qquad \text{(М.107)}$$

(*а*) (*б*) (*в*)

Запис $\overrightarrow{\psi(V)}$ означає операцію векторизації – виконання заданої операції ψ для всіх елементів масиву **V**).

В якості прикладу, наведемо результат обчислення (перші 13 елементів) для 1-го сегмента (рис.5.64).

$s_1 =$

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

$\sigma_1 =$

	1
1	-16.171
2	-16.039-0.37i
3	-16.548-0.428i
4	-11.696+0.936i
5	-22.515+1.469i
6	-18.979-1.948i
7	-18.442-2.908i
8	-18.062-2.386i
9	-13.601-1.661i
10	-21.488-11.192i
11	-21.555-1.861i
12	-16.348-1.132i
13	-17.928-7.227i
14	-20.536-6.973i
15	-22.17-4.883i

$A_1 =$

	1
1	16.171
2	16.043
3	16.554
4	11.733
5	22.562
6	19.078
7	18.670
8	18.219
9	13.702
10	24.228
11	21.635
12	16.387
13	19.329
14	21.687
15	22.701

$\phi_1 =$

	1
1	3.142
2	-3.118
3	-3.116
4	3.062
5	3.076
6	-3.039
7	-2.985
8	-3.010
9	-3.020
10	-2.661
11	-3.055
12	-3.072
13	-2.758
14	-2.814
15	-2.925

$\overrightarrow{(A_1 \cdot \exp(j \cdot \phi_1))} =$

	1
1	-16.171
2	-16.039-0.37i
3	-16.548-0.428i
4	-11.696+0.936i
5	-22.515+1.469i
6	-18.979-1.948i
7	-18.442-2.908i
8	-18.062-2.386i
9	-13.601-1.661i
10	-21.488-11.192i
11	-21.555-1.861i
12	-16.348-1.132i
13	-17.928-7.227i
14	-20.536-6.973i
15	-22.17-4.883i

Рис.5.64. Приклад обчислення програмних модулів (М.106), (М.107)

На рис.5.65 представлено графічну інтерпретацію амплітудного і фазового масивів для 1-го сегмента.

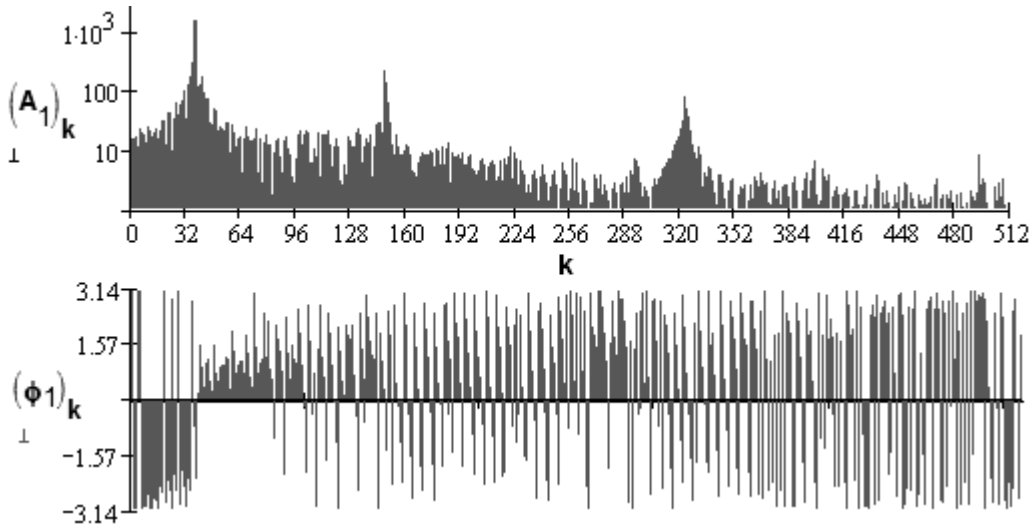


Рис.5.65. Приклад виділеного амплітудного і фазового спектру для сегмента $\mathbf{s}_1$

5) За допомогою програмного модуля (М.108) зберігаємо інформацію про різницю фаз між кожними сусідніми сегментами.

На рис.5.66 проілюстровано результат обчислення різниці фаз між 2-м і 1-м сегментами.

$$\Delta\phi := \left| \begin{array}{l} \Delta\phi_1 \leftarrow 0 \cdot \phi_1 \\ \text{for } n \in 2..N \\ \quad \Delta\phi_n \leftarrow \phi_n - \phi_{n-1} \\ \Delta\phi \end{array} \right. \qquad \text{(М.108)}$$

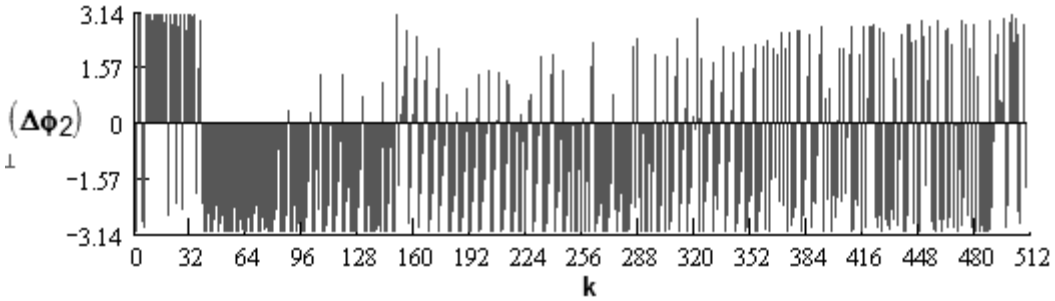


Рис.5.66. Приклад обчислення різниці фаз між сегментами $\mathbf{s}_2$ і $\mathbf{s}_1$

6) З урахуванням різниці фаз $\Delta\phi$ відтворюємо нову матрицю фаз – програмний модуль (М.109). При цьому двійкову послідовність, на яку попередньо перетворюємо приховувані дані, вбудовуємо як значення фази, що дорівнює $\pi/2$, якщо приховується біт «1» і як $-\pi/2$, якщо приховується біт «0». Результат обчислення модуля, на прикладі нових фаз сегментів $\mathbf{s}_1$ і $\mathbf{s}_2$, приведено на рис.5.67.

При формуванні масиву нових фаз внесення приховуваних даних починається з ВЧ складових. Крім того, залишаються без змін перша і остання складові одержаного фазового спектру, оскільки їх модифікація призводить до значного спотворення вихідного сигналу. Після внесення даних, які необхідно було приховати, масив дописується елементами з первинного фазового масиву.

$$
\begin{aligned}
\phi' := \;& m \leftarrow D2B\left(str2vec(M)_1\right) \\
& \textbf{for}\ \tau \in 2..strlen(M) \qquad \textbf{if}\ strlen(M) > 1 \\
& \quad m \leftarrow stack\left(m, D2B\left(str2vec(M)_\tau\right)\right) \\
& \textbf{for}\ k \in 1..\frac{K}{2}+1 \\
& \quad \textbf{if}\ k \le rows(m)+1 \\
& \quad\quad \phi'data_{\frac{K}{2}+1} \leftarrow (\phi_1)_{\frac{K}{2}+1} \quad \textbf{if}\ k = 1 \vee k = \frac{K}{2}+1 \\
& \quad\quad \textbf{if}\ 1 < k \le \frac{K}{2} \\
& \quad\quad\quad \phi'data_{\frac{K}{2}+2-k} \leftarrow -\frac{\pi}{2} \quad \textbf{if}\ m_{k-1} = 1 \\
& \quad\quad\quad \phi'data_{\frac{K}{2}+2-k} \leftarrow \frac{\pi}{2} \quad \textbf{if}\ m_{k-1} = 0 \\
& \quad \phi'data_{\frac{K}{2}+2-k} \leftarrow (\phi_1)_{\frac{K}{2}+2-k} \quad \textbf{if}\ k > rows(m)+1 \\
& \phi'_1 \leftarrow \phi'data \\
& \textbf{for}\ n \in 2..N \\
& \quad \phi'_n \leftarrow \phi'_{n-1} + \Delta\phi_n \\
& \phi'
\end{aligned}
\tag{M.109}
$$

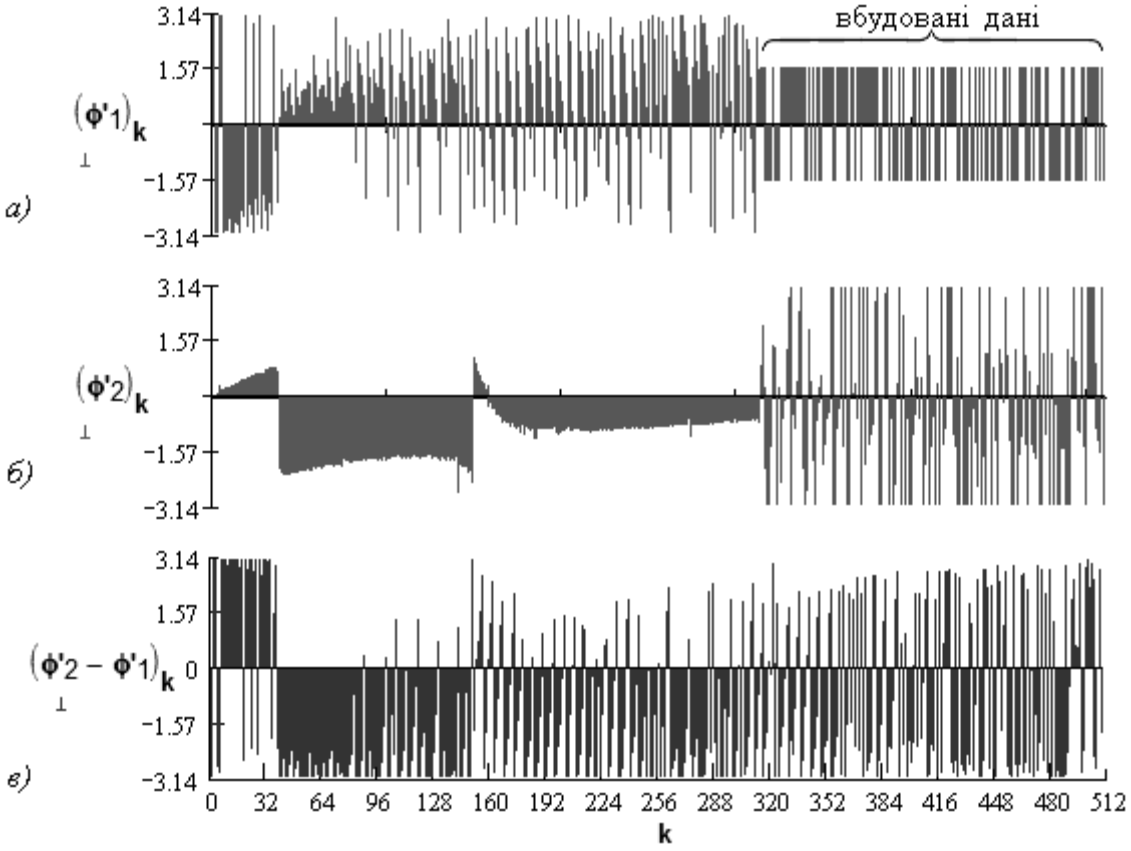


Рис.5.67. Результат вбудовування даних до фазового спектру сегмента $\mathbf{s}_1$ (*а*), обчислення нових значень фаз спектру сегмента $\mathbf{s}_2$ (*б*), перевірка різниці фаз між спектрами сегментів $\mathbf{s}_2$ і $\mathbf{s}_1$ (*в*).

7) Відновлюємо сегменти, шляхом застосування оберненого ШПФ до **n** первинних масивів амплітуд (**A_n**) і модифікованих масивів фаз (**ϕ'_n**) – програмний модуль (М.110). При цьому використовуємо функцію ОШПФ виду **IFFT(W)**, де вектор **W** повинен містити [(**K**/2) + 1] елементів. Дана функція повертає вектор **V**, кількість елементів у якому дорівнює **K.**

Відтворені сегменти об'єднуємо до спільного масиву – (М.111). Результат об'єднання зображено на рис.5.68, *а* (нагадаємо, що в даному випадку нами було приховано 200-бітове повідомлення). На рис.5.68, *б* зображено відновлений звуковий сигнал, у випадку приховання 512-бітового повідомлення при залишенні незмінною довжини сегменту (**K** = 1024). Очевидно, що в останньому випадку рівень прихованості конфіденційного повідомлення є неприпустимо низьким.

$$\mathbf{s'} := \left| \begin{array}{l} \textbf{for}\ \ n \in 1..N \\ \quad \left| \begin{array}{l} \sigma \leftarrow \overrightarrow{\left(A_n \cdot \exp\left(j \cdot \phi'_n\right)\right)} \\ s'_n \leftarrow \mathrm{IFFT}(\sigma) \end{array} \right. \\ s' \end{array} \right. \qquad \text{(М.110)}$$

$$\mathbf{S'} := \left| \begin{array}{l} S' \leftarrow s'_1 \\ \textbf{for}\ \ n \in 2..N \\ \quad S' \leftarrow \mathrm{stack}\left(S', s'_n\right) \\ S' \end{array} \right. \qquad \text{(М.111)}$$

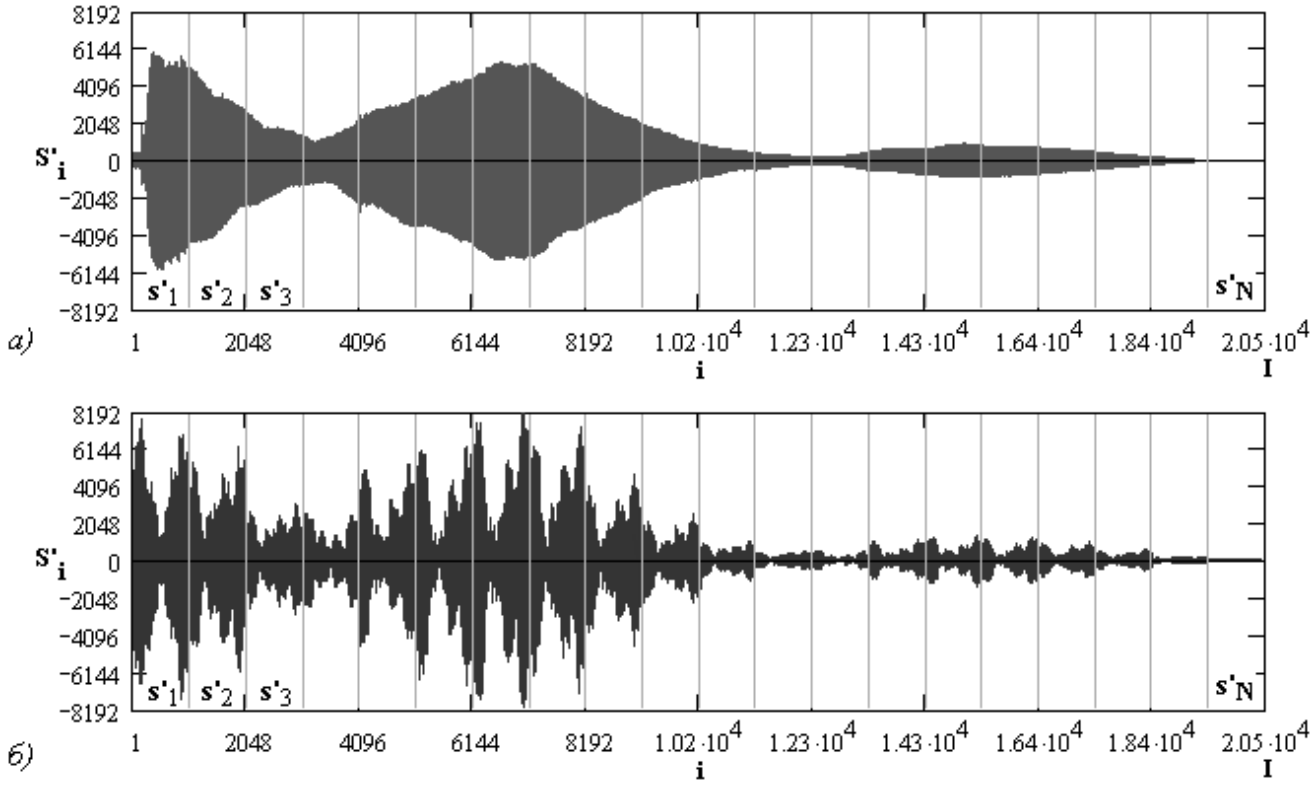


Рис.5.68. Сигнал, відновлений за сегментами **s'_n** (**n** = 1..**N**) при об'ємі прихованого повідомлення 200 (*а*) та 512 біт (*б*)

8) Перед записом звукового файлу необхідно "подовжити" вектор, що відповідає немодифікованому (у нашому випадку – 2-му) каналу, до розмірності модифікованого (1-го) каналу. Для цього можна скористатися модулем (М.112), який за необхідності дописує нулі в кінець початкового масиву.

$$\mathbf{C2} := \left| \begin{array}{l} \textbf{for}\ \ i \in 1..\mathrm{rows}(S') \\ \quad \left| \begin{array}{l} C2_i \leftarrow \left(C^{\langle 2 \rangle}\right)_i \ \ \textbf{if}\ \ i \le \mathrm{rows}\left(C^{\langle 2 \rangle}\right) \\ C2_i \leftarrow 0 \ \ \textbf{if}\ \ i > \mathrm{rows}\left(C^{\langle 2 \rangle}\right) \end{array} \right. \\ C2 \end{array} \right. \qquad \text{(М.112)}$$

В результаті з'являється можливість об'єднати масиви 1-го і 2-го каналів до спільного масиву: **CM** := **augment**(**S'**, **C2**) і, зрештою, провести запис до файлу: **WRITEWAV**("CM_Phase.wav", **$f_д$**, **Q**) := **CM.**

9) Розглянемо алгоритм *видобування прихованої інформації*. Формуємо масив квантованих амплітуд дискретних відліків: **CM*** := **READWAV**("CM_Phase.wav"). Одержувачеві повинні бути відомими: довжина кожного з сегментів **K*** := 1024; точки ШПФ, до яких вбудовувалися дані; алгоритм вбудовування даних (канал, до якого заносилася інформація, метод її занесення і т.п.).

Нехай відомо, що приховані дані містяться у 1-му каналі. Отже, **S*** := $\mathbf{CM}^{*\langle 1\rangle}$. Загальна кількість елементів у даному контейнері: **I*** := **rows**(**S***), **I*** = 20480. Кількість сегментів, на яку необхідно поділити контейнер: **N*** := **I***/**K***, **N*** = 20.

Відповідно до програмного модуля (М.106) проводимо розбиття звукової послідовності **S*** на сегменти $\mathbf{s}^*_n$.

Для 1-го сегмента обчислюємо ШПФ і визначаємо масив фаз: σ^*_1 := **FFT**($\mathbf{s}^*_1$); $\phi^*_1 := \overrightarrow{\arg(\sigma^*_1)}$.

Для видобування прихованої інформації з отриманого масиву фаз використовуємо програмний модуль (М.113).

У даному модулі послідовно аналізуються значення фаз кожної частотної складової спектру 1-го сегмента на факт перевищення встановленого порогу (як у бік від'ємних, так і в бік додатних значень). В якості порогу не рекомендується використовувати строгу рівність + чи $-\pi/2$, оскільки аудіофайл в процесі передавання може зазнати певних змін. Крім того, операції ШПФ повертають в певній мірі округлені значення. Виходячи з цього поріг рекомендується знизити (у нашому випадку було встановлено рівень $|\pi/3|$). При значенні фази, меншому за від'ємне значення порогу приймається рішення про прихований біт «1», у випадку значення фази, що перевищує додатний поріг, робиться висновок про приховання біту «0». Результатом виконання модуля (М.113) є рядок символів: **M*** = "© Пузиренко О.Ю., 2005 р.".

```
M* := | for τ ∈ 1 .. K*/2
      |     d ← (φ*1)_(rows(φ*1)−τ)
      |     b_τ ← 1   if d < −π/3
      |     b_τ ← 0   if d > π/3
      |     break   otherwise
      | for i ∈ 1 .. floor(rows(b)/8)
      |     B ← submatrix[b, (i − 1)·8 + 1, i·8, 1, 1]
      |     M*_i ← B2D(B)
      | vec2str(M*)
```

(М.113)

10) Обчислені показники звукового спотворення зведено до табл.5.6.

5.4.3. Метод розширення спектру (часова область)

У стандартному каналі зв'язку часто є бажаним зосередити інформацію у як можливо вужчій ділянці частотного спектру, для того щоб зберегти наявну смугу пропускання і зменшити потужність сигналу. З іншого боку, основний метод розширення спектру призначений для шифрування потоку інформації шляхом "розсіяння" кодованих даних по всьому можливому частотному спектру. Це робить можливим прийом сигналу навіть за наявності завад на певних частотах.

У [14] розглядається технологія розширення спектру сигналу *прямою послідовністю* (РСПП). Метод РСПП розширює сигнал даних (повідомлення), помножуючи його на елементарну посилку – ПВП максимальної довжини, модульовану відомою частотою. Оскільки аудіосигнали, що використовуються в якості контейнерів, мають дискретний формат, то для кодування як частоту елементарної посилки можна використовувати частоту дискретизації. Як наслідок, дискретний характер сигналу усуває найскладнішу проблему, яка виникає при одержанні сигналу з розширеним прямою послідовністю спектром, – встановлення правильного

початку і кінця квантів елементарної посилки з метою фазової синхронізації. Отже, існує можливість використання набагато вищої частоти елементарної посилки, а, отже, й одержання більш високої пов'язаної з нею швидкості передачі даних. За відсутності цього можуть застосовуватися різноманітні алгоритми блокування сигналу, але у обчислювальному відношенні вони є досить складними.

У РСПП для шифрування і дешифрування інформації необхідний один і той самий ключ – псевдовипадковий шум, який в ідеалі має плоску частотну характеристику в усьому діапазоні частот (тобто білий шум). Ключ застосовується до приховуваної інформації, щоб трансформувати її послідовність у послідовність з розширеним спектром.

Метод РСПП при цьому полягає в наступному. Сигнал даних помножується на сигнал несучої і псевдовипадкову шумову послідовність, яка має широкий частотний спектр. В результаті спектр даних розширюється на всю доступну смугу. Тоді послідовність розширених даних послаблюється і додається до первинного файлу як адитивний випадковий шум (рис.5.69).

РСПП використовує двійкову фазову маніпуляцію, оскільки фаза сигналу ПВП раз від разу чергується з фазою модульованої двійкової послідовності повідомлення (рис.5.70). На стадії видобування фазові значення ϕ_0 та $\phi_0 + \pi$ інтерпретуються, відповідно, як біти «1» і «0», якими кодувалася двійкова послідовність даних. При цьому передбачається наступне:

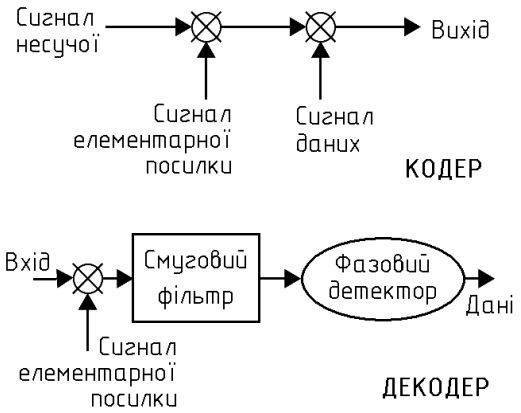


Рис.5.69. Кодування з розширенням спектру

1) Псевдовипадковий ключ являє собою М-послідовність (він має максимально можливу кількість комбінацій, що рівномірно розподілені в заданому діапазоні, і максимально довго не повторюється). Отже, він має відносно плоский частотний спектр.

2) Приймальній стороні є відомим потік ключів для шифрування. Виконана синхронізація сигналу, а також відомі точки початку і кінця розширених даних.

3) Приймальній стороні також відомі наступні параметри: частота слідування елементарних посилок, швидкість передачі даних и частота (вигляд) несучої.

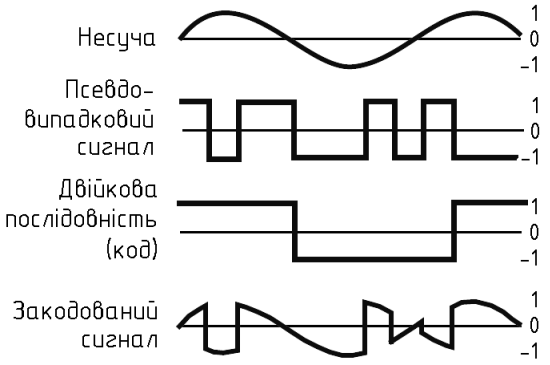


Рис.5.70. Інформація, синтезована розширенням спектру і шифрована методом прямої послідовності

На відміну від фазового кодування, метод РСПП вводить до звуку адитивний випадковий шум. Для того щоб тримати рівень шуму низьким і невідчутним на слух, розширений код послаблюється (без адаптації) приблизно до рівня 0.5% динамічного діапазону звукового файлу-контейнера. Поєднання нескладної техніки повторення з кодуванням із виправленням помилок гарантує цілісність коду. Короткі сегменти двійкової кодової комбінації об'єднуються і додаються до сигналу контейнера таким чином, щоб зменшити шуми перехідних процесів шляхом усереднення по всьому сегменту в процесі декодування. Під час досліджень методу РСПП, авторами [14] була отримана швидкість передачі даних у 4 біти за секунду.

Наведемо приклад реалізації методу РСПП.

1) Початкові дані наступні: контейнер **C** := **READWAV**("C.wav"); повідомлення **M** := "© Пузиренко О.Ю., 2005 р." довжиною $L_M := 8 \cdot \mathbf{strlen(M)}$, L_M = 200 біт, яке будемо вносити до 1-го каналу контейнера: $\mathbf{C1} := \mathbf{C}^{<1>}$.

Для вбудовування L_M-бітового повідомлення до контейнеру, який має **rows**(**C1**) = 20191 дискретних відліків, останній необхідно розбити на **N** := **floor**(**rows**(**C1**)/L_M), **N** = 100 сегментів, до кожного з яких вбудовуватиметься один біт повідомлення.

2) Для кожного біту повідомлення необхідно згенерувати ПВП з +/–1, довжиною як мінімум **N** елементів. За основу генератора ПВЧ візьмемо ЛРЗЗЗ, описаний у п.5.3.3.3. Достатня кількість розрядів регістру: **d** := **ceil**(**log**(**N**+1, 2)), **d** = 7. При цьому період генерованої ПВП складе $2^{d} - 1 = 127 > N$.

Програмний модуль (М.114) дозволяє одержувати ПВП при різних значеннях **d**. Вихідна послідовність визначається найменшим значущим бітом стану регістра (**CHIP**$_i \leftarrow$ **R**$_{bin_1}$). Керуючись достатністю, процес генерації триває до отримання **N**-го біту ПВП. На завершальному етапі одержана ПВП {0, 1}, перетворюється на ПВП {–1, 1}.

$$\mu_1 := (1)^T$$
$$\mu_2 := (1\ \ 1)^T$$
$$\mu_3 := (1\ \ 0\ \ 1)^T$$
$$\mu_4 := (0\ \ 0\ \ 1\ \ 1)^T$$
$$\mu_5 := (0\ \ 1\ \ 0\ \ 0\ \ 1)^T$$
$$\mu_6 := (1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 1)^T$$
$$\mu_7 := (1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 1)^T$$
$$\mu_8 := (0\ \ 1\ \ 1\ \ 1\ \ 0\ \ 0\ \ 0\ \ 1)^T$$
$$\mu_9 := (0\ \ 0\ \ 0\ \ 1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 1)^T$$
$$\mu_{10} := (0\ \ 0\ \ 1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 1)^T$$
$$\mu_{11} := (0\ \ 1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 1)^T$$
$$\mu_{12} := (1\ \ 0\ \ 0\ \ 1\ \ 0\ \ 1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 1)^T$$
$$\vdots$$

```
CHIP(start) := | μ ← μ_d
               | for i ∈ 1 .. N
               |   if i = 1
               |     R_dec_i ← start
               |     R_bin ← D2B(R_dec_i)
               |   if i ≥ 2
               |     bit ← 0
               |     for j ∈ 1 .. d
               |         bit ← R_bin_j ⊕ bit  if μ_j = 1
               |     R ← R_bin
               |     for j ∈ 1 .. d
               |       | R_bin_j ← R_(j−1)  if j > 1
               |       | R_bin_j ← bit  if j = 1
               |     R_dec_i ← B2D(R_bin)
               |   CHIP_i ← R_bin_1
               | CHIP·2 − 1
```

(М.114)

3) Безпосередньо вбудовування бітів повідомлення до контейнера виконується програмним модулем (М.115), на початку якого рядок **M** перетворюється на вектор {–1, 1}. **m**-й елемент одержаного вектора за допомогою відповідної йому ПВП **CHIP**(**m**) поширюється на **m**-й сегмент контейнера **c** (параметр α обирається виходячи з вимог стійкості системи і непомітності модифікації носія, рекомендуються значення порядку однієї сотої). Модифіковані сегменти **s** об'єднуються до спільного вектора **S**. Після вбудовування останнього (L_M-го) біту повідомлення вектор **S** подовжується на довжину контейнера **C1** кінцевими його елементами, що не зазнали модифікації. Результат вбудовування зображено на рис.5.71.

```
S1 := | M_vec ← str2vec(M)
      | M_vec_bin ← D2B(M_vec_1)
      | for j ∈ 2.. strlen(M)
      |   M_vec_bin ← stack(M_vec_bin, D2B(M_vec_j))
      | M_vec_bin ← 2·M_vec_bin − 1
      | m ← 1
      | while m ≤ L_M
      |       c ← submatrix[C1, [N·(m − 1) + 1], N·m, 1, 1]
      |       s ← c + α·((c·CHIP(m))→ · M_vec_bin_m)
      |       S1 ← s  if m = 1
      |       S1 ← stack(S1, s)  if m > 1
      |       m ← m + 1
      | S1 ← stack(S1, submatrix(C1, L_M·N + 1, rows(C), 1, 1))
      | S1
```

(M.115)

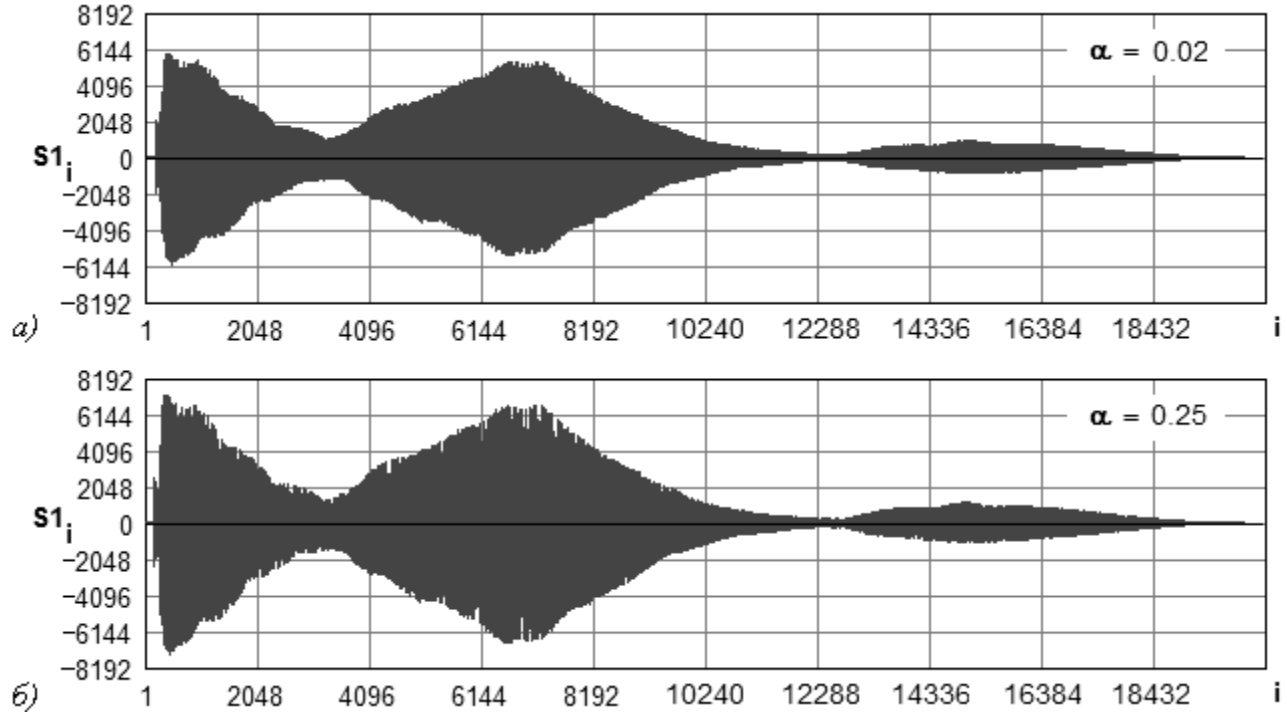


Рис.5.71. Часові діаграми 1-го каналу сигналу-контейнера при внесенні приховуваних даних шляхом РСПП при α = 0.02 (*а*) і 0.25 (*б*)

Об’єднуємо масиви 1-го (модифікованого) і 2-го (первинного) каналів до спільного масиву: **S** := **augment**(**S1**, $\mathbf{C}^{<2>}$) і, виконуємо запис до файлу: **WRITEWAV**("S.wav", $\mathbf{f_д}$, **Q**) := **S**.

4) Процес видобування полягає в наступному. Після імпортування аудіофайлу до документу MathCAD (**S*** := **READWAV**("S.wav")) з одержаного масиву виокремлюється канал, до якого було проведено вбудовування (у нашому випадку – 1-й): **S1*** := $\mathbf{S}^{*<1>}$. Приймальна сторона повинна мати оригінальний аудіофайл, з якого також виокремлюється відповідний канал: **C1** := $\mathbf{C}^{<1>}$. Відомою повинна бути кількість сегментів, на яку розбивається сигнал-носій: **N*** := 100. Програмний модуль видобування повідомлення (М.116), вбудованого методом РСПП, представлено нижче.

```
M* := | m ← 1
      | while  m·N* ≤ rows(S*)
      |        s* ← submatrix[S1* , [N*·(m − 1) + 1] , N*·m , 1 , 1]
      |        c ← submatrix[C1 , [N*·(m − 1) + 1] , N*·m , 1 , 1]
      |        c' ← [(s* − c)·CHIP*(m)]
      |        M*bin_m ← 1  if  sign(mean(c)) = sign(mean(c'))
      |        M*bin_m ← 0  if  sign(mean(c)) ≠ sign(mean(c'))
      |        m ← m + 1
      | for  j ∈ 1.. rows(M*bin) ÷ 8
      |        M*vec_j ← B2D(submatrix(M*bin , 8·j − 7 , 8·j , 1 , 1))
      | vec2str(M*vec)
```
(M.116)

Якщо до сегменту було вбудовано «1», середні значення оригінального (**c**) і модифікованого (**c'**) сегментів матимуть однакові знаки, якщо «0» – різні. Недоліком такої системи є необхідність наявності оригінального аудіофайлу у приймальної сторони. Можливим виходом з цієї ситуації є запис модифікованого каналу замість 2-го, при цьому оригінальний 1-й канал залишається без змін: **S** := **augment**(**C**$^{\langle 1 \rangle}$, **S1**). Таким чином, одночасно передаватимуться як пустий, так і заповнений контейнери. Зазначена можливість вбудовування враховується наступною зміною у модулі (М.116):

```
| s* ← submatrix[S*⟨2⟩ , [N*·(m − 1) + 1] , N*·m , 1 , 1]
| c ← submatrix[S*⟨1⟩ , [N*·(m − 1) + 1] , N*·m , 1 , 1]
|  ⋮
```
(М.117)

Звичайно, модифікувати вказаним чином весь стереофонічний аудіосигнал є недоцільним через повне зникнення стереоефекту. Тому можна обмежитися однією чи кількома заздалегідь обумовленими з приймальною стороною ділянками або навіть окремими відліками по всій тривалості сигналу, на основі яких створюється вибірка, в яку або з якої й виконується вбудовування / видобування бітів повідомлення.

5) Обчислені показники звукового спотворення аудіоконтейнера вбудовуванням повідомлення з розширеним спектром занесено до табл.5.6.

5.4.4. Приховування даних з використанням ехо-сигналу

Даний метод розуміє під собою вбудовування даних до аудіосигналу-контейнеру шляхом введення до нього *ехо-сигналу* [14]. Дані приховуються зміною трьох параметрів ехо-сигналу: початкової амплітуди, швидкості загасання і зсуву (рис.5.72). Коли зсув (або затримка) між первинним і ехо-сигналом зменшується, два сигнали змішуються. Починаючи з деякого значення затримки ССЛ стає не спроможною виявити різницю між цими двома сигналами, а ехо-сигнал сприймається лише як додатковий резонанс. Вказане значення важко визначити точно, оскільки воно залежить від якості первинного звукозапису, типу звуку, для якого формується ехо-сигнал, і, зрештою, слухача. У загальному випадку, автори [14] прийшли до висновку, що для більшості звуків і більшості слухачів злиття відбувається при затримці близько однієї мілісекунди.

Кодер використовує два часи затримки: один для представлення двійкового нуля (зсув), а інший – для представлення двійкової одиниці (зсув + δ). Обидва часи затримки є меншими граничного часу, за якого ССЛ здатна розпізнати ехо-сигнал. Крім зменшення часу затримки

для забезпечення непомітності також можна встановити рівні початкової амплітуди і часу загасання, які були б нижчими за відчутні ССЛ.

Процес вбудовування даних може бути представлено як пристрій, що має одну з двох можливих системних функцій. У часовій області системні функції – це дискретні у часі експоненти (рис.5.73), відмінні лише у затримці між імпульсами.

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

На рис.5.74 представлено системні функції для кодування двійкових «1» і «0». Обробка сигналу у відповідності до рис.5.74, *а* або *б* матиме результатом закодований сигнал (рис.5.75).

Затримка δ_{bit} між первинним і ехо-сигналом є залежною від того, яке представлення або системна функція була використана. Представлення «1» створюється затримкою у δ_1 секунд, тоді як представлення «0» – затримкою у δ_0 секунд.

Для того щоб закодувати більше одного біта, первинний сигнал розкладається на менші сегменти. Кожен сегмент розглядається як окремий сигнал і в нього може бути вбудовано (шляхом ехо-відображення) один необхідний біт інформації. Результуючий закодований сигнал (утримуючий декілька біт) являє собою нове об'єднання всіх незалежно закодованих сегментів первинного сигналу.

На рис.5.76 зображено приклад, при якому сигнал було розділено 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 треба застосувати системну функцію представлення одиниці (рис.5.74, *а*). Кожен сегмент індивідуально згортається з системною функцією. Нулі, поміщені до сегментів ***b***, ***e*** та ***f***, кодуються аналогічним чином, використовуючи спосіб представлення нуля (рис.5.74, *б*). Одержані після згортання з відповідною системною функцією результати повторно об'єднуються. Для досягнення мінімальної помітності повторного з'єднання, у [14] попередньо

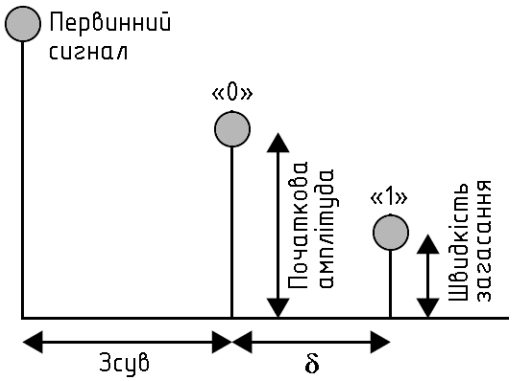


Рис.5.72. Регульовані параметри ехо-сигналу

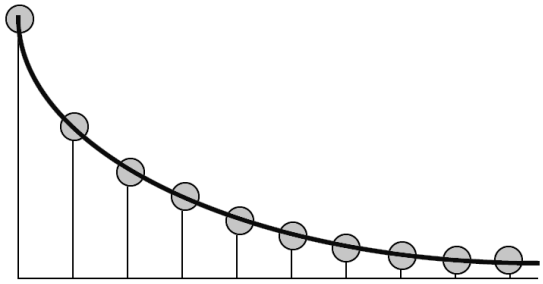


Рис.5.73. Дискретний у часі експоненціал

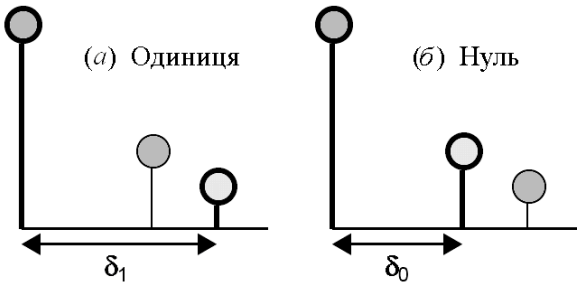


Рис.5.74. Представлення ехо-сигналу

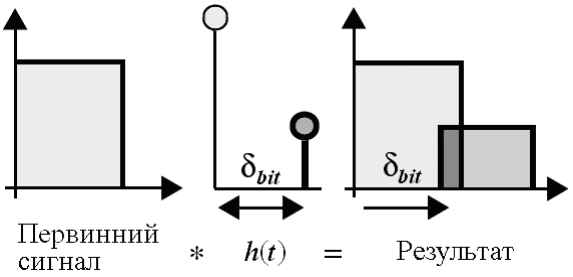


Рис.5.75. Приклад ехо-відображення

пропонується створити окремі “одиничний” і “нульовий” ехо-сигнали, повторюючи первинний і використовуючи відповідні представлення «1» і «0». Отримані в результаті сигнали зображені на рис.5.77.

“Одиничний” і “нульовий” ехо-сигнали містять, відповідно, лише одиниці і нулі. Для того щоб об’єднати ці два сигнали, також створюються два змішувальних сигнали (рис.5.78), що являють собою двійкову послідовність, стан якої залежить від того, який біт треба приховати в тому або іншому сегменті первинного сигналу.

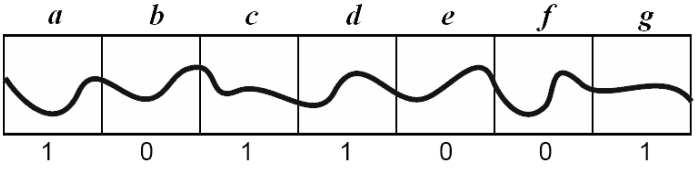


Рис.5.76. Розбиття первинного сигналу на менші сегменти для вбудовування інформації, що являє собою двійкову послідовність

“Одиничний” і “нульовий” змішувальні сигнали помножуються на відповідні їм ехо-сигнали. Іншими словами, останні масштабуються одиницею, або нулем протягом всієї тривалості сигналу в залежності від того, який біт передбачається помістити до будь-якого його окремого сегменту. В подальшому два результати додаються один до одного. Необхідно зауважити, що “нульовий” змішувальний сигнал являє собою інверсію “одиничного” змішувального сигналу, крім того, фронти переходів кожного сигналу є похиленими. Сума обох змішувальних сигналів завжди дорівнює одиниці. Все це дозволяє одержати плавний перехід між сегментами, кодованими різними бітами, а також запобігає різким змінам у звучанні результуючого (змішаного) сигналу.

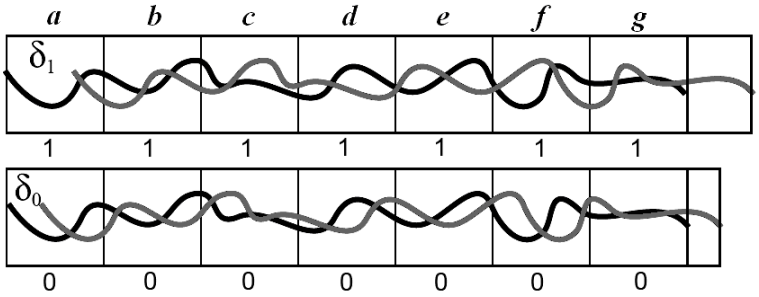


Рис.5.77. Створення “одиничного” і “нульового” ехо-сигналів (світліша лінія)

Блок-схема, що представляє повний процес вбудовування, зображена на рис.5.79.

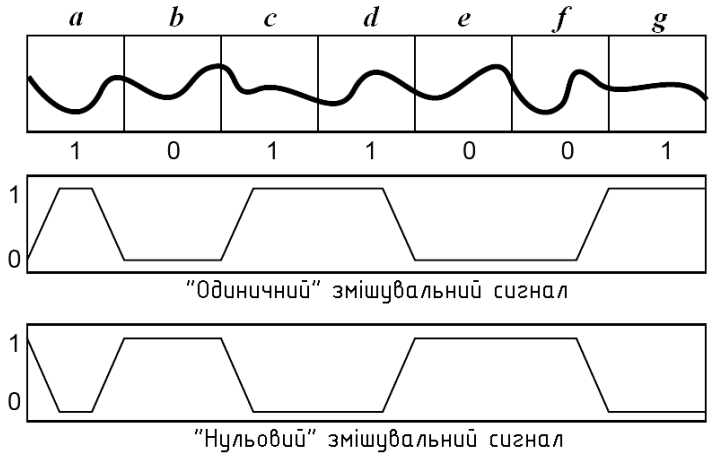


Рис.5.78. Змішувальні сигнали

Видобування вкладеної інформації розуміє під собою виявлення інтервалу між ехо-сигналами окремих сегментів. Для цього необхідно дослідити у двох позиціях амплітуду автокореляційної функції (АКФ) косинус-перетворення Фур’є натурального логарифму спектру потужності (або так званого

кепстра) кодованого сигналу [92,93]:

$$\boldsymbol{F}^{-1}\left\{\left[\ln_{\boldsymbol{compl.}}(\boldsymbol{F}(\boldsymbol{x}))\right]^2\right\}. \quad (5.61)$$

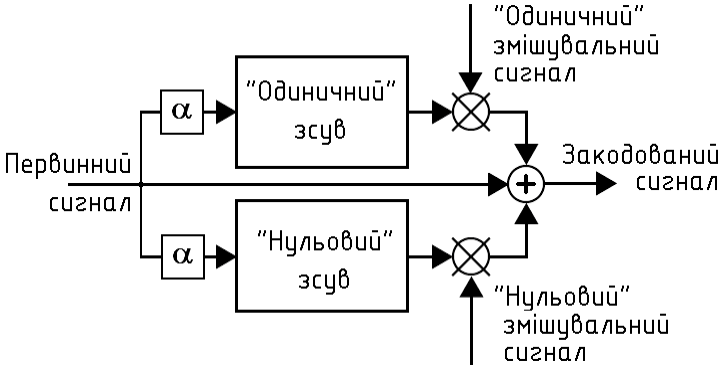


Рис.5.79. Процес вбудовування

Розглянемо приклад процесу видобування, наведений у [14]. Нехай одержано закодований сигнал, що являє собою таку послідовність імпульсів, в якій останні відділені один від одного визначеним інтервалом і характеризуються експонентним загасанням амплітуди. В усіх інших точках сигнал дорівнює нулеві (рис.5.80).

Наступним кроком є пошук кепстра ехо-версії. Результат обчислення кепстра робить інтервал між ехо-сигналом і первинним сигналом дещо більш виразним.

Нажаль, результат обчислення кепстра до того ж дублює ехо-сигнал через кожні δ секунд. На рис.5.81 це зображено послідовністю імпульсів на виході. Крім того, амплітуда імпульсів, що представляють ехо-сигнали, є малою по відношенню до первинного сигналу. Як наслідок, їх важко виявити. Розв'язок даної проблеми полягає в обчисленні АКФ кепстра. Одноразовим відображенням сигналу із затримкою δ (рис.5.82), отримуємо результат, зображений на рис.5.83.

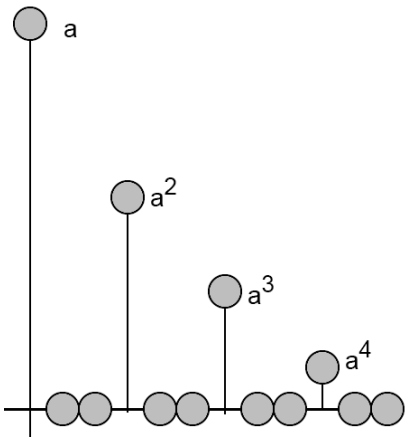


Рис.5.80. Зразок сигналу: $\boldsymbol{x}[\boldsymbol{n}]= a^{\boldsymbol{n}}\cdot\boldsymbol{u}[\boldsymbol{n}]$; $(0 < a < 1)$

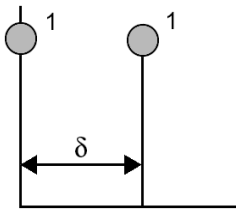


Рис.5.82. Зразок відображеного сигналу

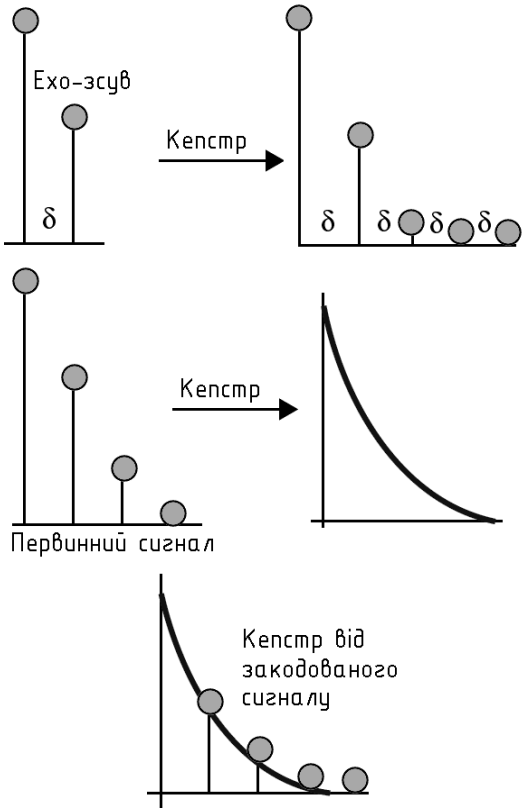


Рис.5.81. Кепстр від ехо-кодованого сигналу

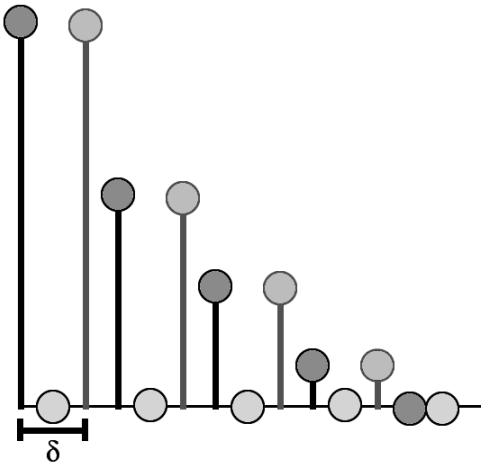


Рис.5.83. Відображена версія зразка сигналу

Значно посиленим є лише перший імпульс, оскільки його підкріплено наступними за ним імпульсами. Таким чином, в позиції першого імпульсу ми отримуємо сплеск. Подібно до першого імпульсу, сплеск повторюється через δ_1 або δ_0 секунд після первинного сигналу. Остаточні складові імпульсів наближаються до нуля, що дозволяє ефективно послаблювати шуми.

Розглянемо критерій прийняття рішення стосовно того, який біт («1» чи «0») прихований у часовій затримці між первинним сигналом і затримкою δ перед сплеском АКФ. Згадаємо, що «1» кодувалася розміщенням ехо-сигналу через δ_1, а «0» – через δ_0 секунд після оригіналу. При видобуванні, біту присвоюється одиниця, якщо значення АКФ через δ_1 секунд є більшим, ніж через δ_0 секунд. В іншому випадку – присвоюється нуль.

За твердженням авторів [14], за допомогою даного методу цілком можливо приховувати і видобувати інформацію у вигляді двійкового коду в потік аудіоданих з мінімальною зміною первинного сигналу при ПЗ приблизно у 16 біт/сек. Під мінімальною зміною розуміється те, що середньостатистична людина не здатна відчувати жодної суттєвої різниці між модифікованим і первинним сигналами. Проте, у своїй же роботі [108] автори вказують на те, що розроблений метод не є універсальним – для деяких аудіосигналів неможливо одержати достатньо високий коефіцієнт правильно розпізнаних при видобуванні бітів навіть за відсутності завад у каналі зв'язку.

Розглянемо реалізацію методу ехо-кодування за допомогою системи MathCAD.

1) Початкові дані: контейнер з частотою дискретизації $f_д$ = 22050 Гц та кількістю рівнів квантування **Q** = 16 — C_{total} := **READWAV**("C.wav"); **C** := $C_{total}^{<1>}$ (**rows**(**C**) = 20191) повідомлення **M** := "Олександр" довжиною N_M := **strlen**(**M**), N_M = 9 символів або $L_M := 8 \cdot N_M$, L_M = 72 біт.

2) Нехай нульова затримка між первинним і ехо-сигналом складає δ_0 := 20 дискретних відліків (або $\delta_0 / f_д$ = 0.907 мс), а одинична – δ_1 := 30 відліків (або $\delta_1 / f_д$ = 1.361 мс).

"Одиничний" і "нульовий" ехо-сигнали одержимо шляхом простого зміщення на $\delta_{0(1)}$ відліків елементів контейнера-оригіналу і поелементного підсумовування одержаних векторів (попередньо помножених на коефіцієнт загасання α) з вектором **C** – програмні модулі (М.118) і (М.119).

$$C_0 := \left| \begin{array}{l} \text{for } i \in 1..\delta_0 \\ \quad C_{0_i} \leftarrow 0 \\ C_0 \leftarrow \alpha \cdot \text{stack}(C_0, C) \end{array} \right. \qquad C_1 := \left| \begin{array}{l} \text{for } i \in 1..\delta_1 \\ \quad C_{1_i} \leftarrow 0 \\ C_1 \leftarrow \alpha \cdot \text{stack}(C_1, C) \end{array} \right. \tag{M.118}$$

$$\text{rows}(C_0) = 20211 \qquad \text{rows}(C_1) = 20221$$

$$\text{rows}(C_0) - \text{rows}(C) = 20 \qquad \text{rows}(C_1) - \text{rows}(C) = 30$$

$$\Sigma C_0 := \left| \begin{array}{l} \text{for } i \in 1..\text{rows}(C) \\ \quad \Sigma C_{0_i} \leftarrow C_i + C_{0_i} \\ \Sigma C_0 \end{array} \right. \qquad \Sigma C_1 := \left| \begin{array}{l} \text{for } i \in 1..\text{rows}(C) \\ \quad \Sigma C_{1_i} \leftarrow C_i + C_{1_i} \\ \Sigma C_1 \end{array} \right. \tag{M.119}$$

$$\text{rows}(\Sigma C_0) = 20191 \qquad \text{rows}(\Sigma C_1) = 20191$$

Фрагменти (перші 70 відліків) результату зсуву сигналу **C** на δ_0 і δ_1 відліків зображено на рис.5.84.

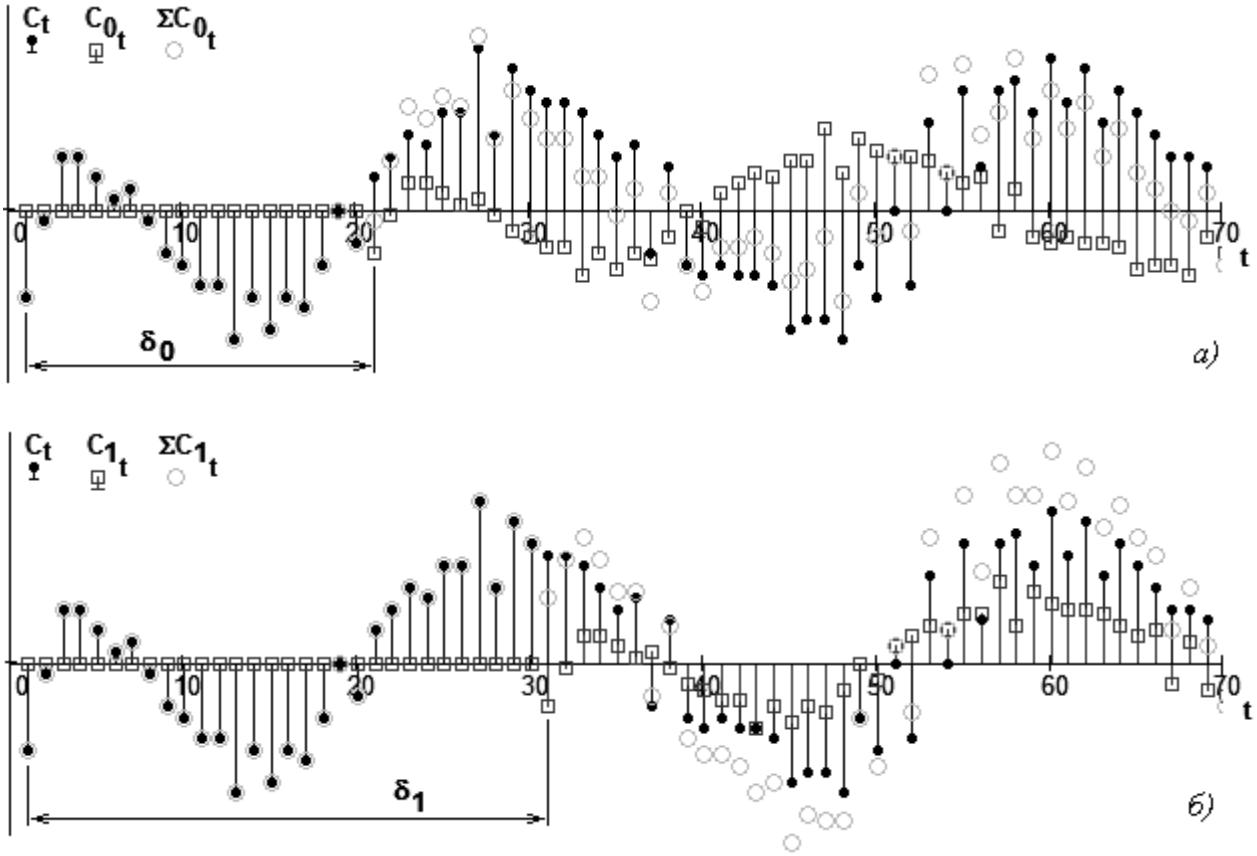


Рис.5.84. "Нульовий" (*а*) та "одиничний" (*б*) ехо-сигнали сигналу **C** при $\alpha = 0.5$

3) Для можливості приховання більше одного біта первинний сигнал **C** необхідно розділити на менші сегменти, кожен з яких розглядатиметься як окремий сигнал і до якого шляхом ехо-відображення може бути вбудовано необхідний біт даних. Обчислимо кількість відліків в одному сегменті, виходячи з бітової довжини L_M повідомлення (шляхом округлення до найближчого найменшого цілого): $N_Б := \text{floor}(\text{rows}(C) / L_M)$, $N_Б = 280$.

4) Виходячи із зазначеної при описі методу необхідності у похилених фронтах імпульсів змішувальних сигналів (трапецеїдальні імпульси), попередньо задаємося наступним:

– розмах імпульсу: $U := 1$;

– тривалість фронтів: $\tau := 20$ відліків;

– тривалість імпульсу на рівні **U**: $T := N_Б - \tau$, $T = 260$ відліків.

Амплітуди відліків імпульсу формуються за допомогою комплексного програмного модуля (М.120).

Графічну інтерпретацію обчислення модуля зображено на рис.5.85.

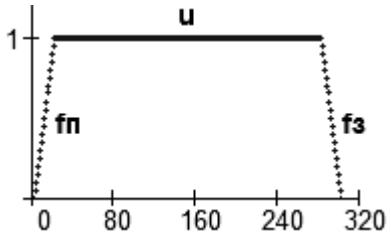


Рис.5.85. Трапецеїдальний імпульс

```
fп := for n ∈ 1 .. τ
        fп_n ← (n − 1)/τ   if τ > 0
        fп_1 ← 0           if τ = 0
      fп

u := for n ∈ (τ + 1) .. (τ + T)
       u_(n−τ) ← 1
     u

fз := for n ∈ (τ + T + 1) .. (2·τ + T)
        fз_(n−(τ+T)) ← (n − (2·τ + T))/(−τ)   if τ > 0
        fз_1 ← 0                              if τ = 0
      fз
```

(М.120)

Формування змішувальних сигналів виконаємо, користуючись програмним модулем (М.121). При цьому приймаємо, що кожен біт повідомлення кодується $1/2$ переднього фронту імпульсу, його одиничним рівнем та $1/2$ заднього фронту. Після формування змішувального сигналу для всіх L_M біт повідомлення, даний сигнал дописується нульовим або одиничним рівнем, в залежності від значення останнього (L_M-го) біту. Нульовий змішувальний сигнал одержується з одиничного з урахуванням того, що їх сума дорівнює одиниці.

Змішувальні сигнали для першої вісімки бітів повідомлення представлено на рис.5.86.

```
μ1 := M_vec ← str2vec(M)
      M_vec_bin ← D2B(M_vec_1)
      for j ∈ 2 .. N_M
          M_vec_bin ← stack(M_vec_bin, D2B(M_vec_j))
      μ ← stack[0·submatrix[fз, (round(τ/2) + 1), τ, 1, 1], 0·u]   if M_vec_bin_1 = 0
      μ ← stack[0·submatrix[fп, (round(τ/2) + 1), τ, 1, 1] + 1, u]   if M_vec_bin_1 = 1
      for m ∈ 2 .. L_M
          μ ← stack(μ, fз, 0·u)   if M_vec_bin_m = 0 ∧ M_vec_bin_m ≠ M_vec_bin_(m−1)
          μ ← stack(μ, 0·fз, 0·u)   if M_vec_bin_m = 0 ∧ M_vec_bin_m = M_vec_bin_(m−1)
          μ ← stack(μ, fп, u)   if M_vec_bin_m = 1 ∧ M_vec_bin_m ≠ M_vec_bin_(m−1)
          μ ← stack[μ, (0·fп + 1), u]   if M_vec_bin_m = 1 ∧ M_vec_bin_m = M_vec_bin_(m−1)
      while rows(μ) ≤ rows(C)
          μ ← stack(μ, 0·fп, 0·u)   if M_vec_bin_(L_M) = 0
          μ ← stack[μ, (0·fп + 1), u]   if M_vec_bin_(L_M) = 1
      μ

μ0 := (−1·μ1 + 1)
```

(М.121)

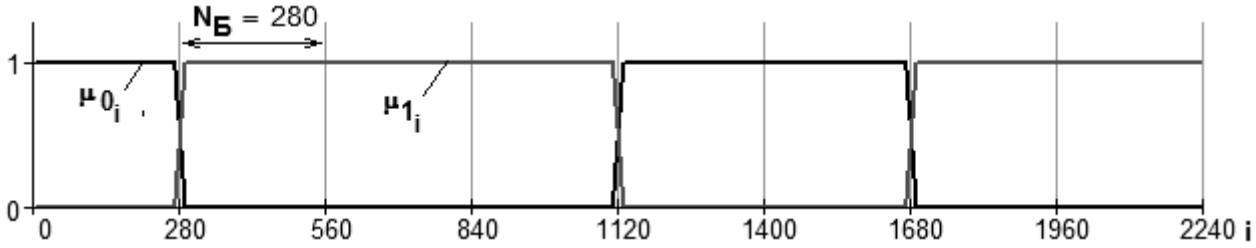


Рис.5.86. "Нульовий" (μ_0) і "одиничний" (μ_1) змішувальні сигнали

5) Безпосередньо вбудовування бітів повідомлення до аудіоконтейнера виконує програмний модуль (М.122). В залежності від значення поточного біту повідомлення, проводиться виокремлення сегменту заданої розмірності ($N_Б$) з відповідного ехо-сигналу та зі змішувального сигналу, які в подальшому поелементно перемножуються (для цього можна використати й операцію векторизації). Отримані для кожного біту вектори **s'** формують загальний вектор заповненого контейнера **S**, до якого після вбудовування останнього символу повідомлення дописуються елементи контейнера-оригіналу, що не зазнали модифікації. Очевидно, що кількість елементів сформованого вектора **S** відповідатиме аналогічному показнику для **C**: **rows**(**S**) = 20191.

Одержаний вектор об'єднуємо з немодифікованим другим каналом і записуємо результат до файлу: **WRITEWAV**("S_echo.wav", $f_д$, **Q**) := **augment**(**S**, $C_{total}^{<1>}$).

6) Для видобування прихованого повідомлення передбачається наступне: одержувачеві відомі розмірність блоків, на які розбивається контейнер ($N^*_Б := N_Б$), значення нульової та одиничної затримок ($\delta^*_0 := \delta_0$, $\delta^*_1 := \delta_1$). Програмний модуль видобування даних – (М.123). До його основи покладено обчислення автокореляційної функції кепстру (5.61) (для більш надійного видобування аналізується окіл відліків δ^*_0 і δ^*_1).

$$
\begin{array}{l}
S := \left|
\begin{array}{l}
M_{vec} \leftarrow \text{str2vec}(M) \\
M_{vec\_bin} \leftarrow \text{D2B}\left(M_{vec_1}\right) \\
\text{for } j \in 2..N_M \\
\quad M_{vec\_bin} \leftarrow \text{stack}\left(M_{vec\_bin}, \text{D2B}\left(M_{vec_j}\right)\right) \\
\text{for } m \in 1..L_M \\
\quad \left|
\begin{array}{l}
\text{if } M_{vec\_bin_m} = 0 \\
\quad \left|
\begin{array}{l}
\Sigma \leftarrow \text{submatrix}\left[\Sigma C_0, \left[N_Б \cdot (m-1)+1\right], N_Б \cdot m, 1, 1\right] \\
\mu \leftarrow \text{submatrix}\left[\mu_0, \left[N_Б \cdot (m-1)+1\right], N_Б \cdot m, 1, 1\right] \\
\text{for } n \in 1..N_Б \\
\quad S'_n \leftarrow \Sigma_n \cdot \mu_n
\end{array}
\right. \\
\text{if } M_{vec\_bin_m} = 1 \\
\quad \left|
\begin{array}{l}
\Sigma \leftarrow \text{submatrix}\left[\Sigma C_1, \left[N_Б \cdot (m-1)+1\right], N_Б \cdot m, 1, 1\right] \\
\mu \leftarrow \text{submatrix}\left[\mu_1, \left[N_Б \cdot (m-1)+1\right], N_Б \cdot m, 1, 1\right] \\
\text{for } n \in 1..N_Б \\
\quad S'_n \leftarrow \Sigma_n \cdot \mu_n
\end{array}
\right. \\
S \leftarrow S' \text{ if } m = 1 \\
S \leftarrow \text{stack}(S, S') \text{ if } m > 1
\end{array}
\right. \\
\text{for } i \in \text{rows}(S)+1..\text{rows}(C) \text{ if } \text{rows}(S) < \text{rows}(C) \\
\quad S_i \leftarrow C_i \\
S
\end{array}
\right.
\end{array}
\qquad \text{(М.122)}
$$

```
M* := | m ← 1
      | while  N*Б·m < rows(S*)
      |        s ← submatrix[S*, [N*Б·(m − 1) + 1], N*Б·m, 1, 1]
      |        F ← CFFT(s)
      |        L ← ln(F⃗ + 10^−20)^2
      |        AC ← ICFFT(L)
      |        "0" ← Σ_{j = δ*0−2}^{δ*0+2} |ACj|
      |        "1" ← Σ_{j = δ*1−2}^{δ*1+2} |ACj|
      |        B ← 1   if  "1" > "0"
      |        B ← 0   if  "1" < "0"
      |        B ← round(rnd(1))   if  "1" = "0"
      |        M* bin m ← B
      |        m ← m + 1
      | for  j ∈ 1 .. rows(M*bin) ÷ 8
      |      M*vecj ← B2D(submatrix(M*bin, 8·j − 7, 8·j, 1, 1))
      | vec2str(M*vec)
```

(M.123)

На рис.5.87 наведено результат обчислення АКФ кепстру для 1-го і 8-го бітів повідомлення ($M_{vec\_bin_1} = 0$, $M_{vec\_bin_8} = 1$).

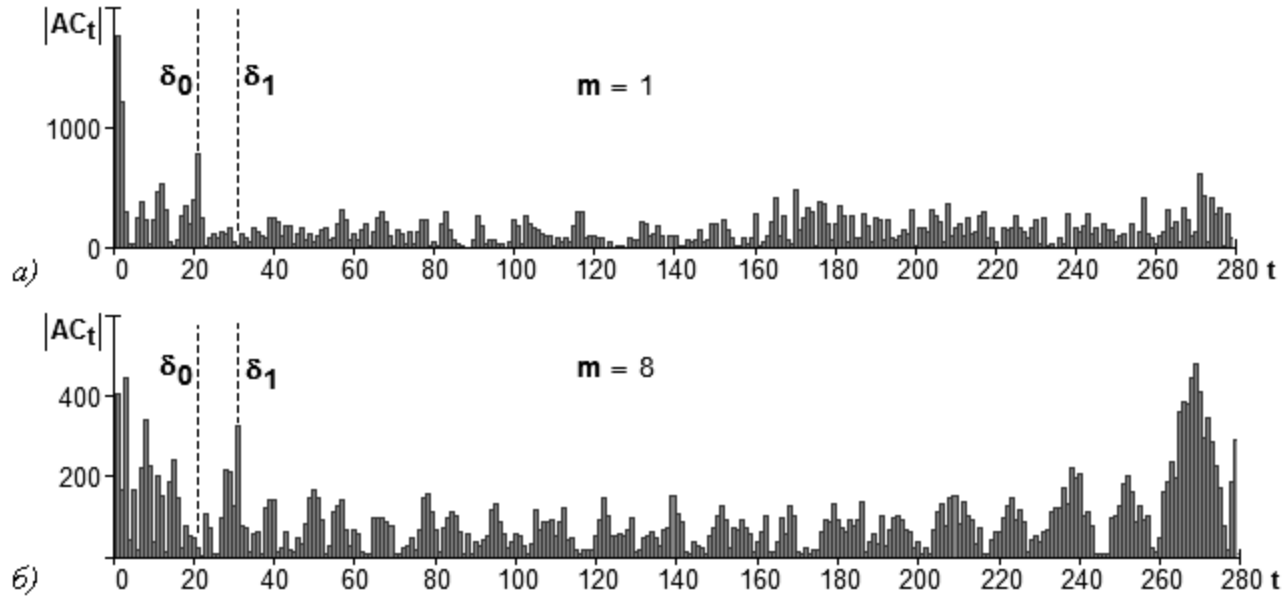


Рис.5.87. Приклад вигляду АКФ кепстру сигналу, що містить нульове (*а*) і одиничне (*б*) ехо-відображення.

7) Результати обчислення показників звукового спотворення контейнера при вбудовуванні до нього даних шляхом ехо-кодування зведено до табл.5.6.

5.5. Приховування даних у тексті

Для приховування конфіденційних повідомлень у тексті (або так звана *лінгвістична стеганографія*) використовується або звичайна надлишковість мови, або формати представлення тексту.

Електронна версія тексту за багатьма причинами є найскладнішим місцем для приховування даних, на відміну від її "жорсткої" копії (наприклад, паперової), яка може бути оброблена як високоструктуроване зображення і є такою, що легко піддається різноманітним методам приховування, таким як незначні зміни формату текстових зразків, регулювання відстані між певними парами символів (кернінг), відстані між рядками тощо. В значній мірі це викликане відносним дефіцитом у текстовому файлі надлишкової інформації, особливо у порівнянні із зображеннями чи звуковими фрагментами. У той час, як до зображення/звуку є можливим у більшості випадків внести непомітні/невідчутні модифікації, навіть додаткова літера або крапка у тексті можуть бути помічені випадковим читачем. Приховування даних у тексті вимагає пошуку таких модифікацій, які були б непомітними переважною більшістю читачів. Автори [14] розглядають три групи методів, що зазнали найбільшого поширення при вбудовуванні приховуваних даних до тексту:

– *методи довільного інтервалу*, що здійснюють вбудовування шляхом маніпуляції з пробільними символами (вільним місцем на друкованій полосі);

– *синтаксичні методи*, які працюють з пунктуацією;

– *семантичні методи*, до основи яких покладене залежне від приховуваних бітів даних маніпулювання словами.

Таблиця 5.6.

Показники звукового спотворення у випадку приховування даних в аудіосередовищі

Назва показника спотворення	*Оригінал*	*Методи приховування даних в аудіосередовищі*			
		НЗБ (ПВ інтервал)	*Фазове кодування*	*Розширення спектру*	*Ехо-кодування*
Макс. абсолютна різниця, ***MD***	0	1	1124	148	2963
Середня абсолютна різниця, ***AD***	0	$3.244 \cdot 10^{-3}$	15.298	14.321	572.093
Норм. середня абс. різниця, ***NAD***	0	$2.625 \cdot 10^{-6}$	0.012	0.012	0.5
Середньоквадр. помилка, ***MSE***	0	0.003	$2.657 \cdot 10^{3}$	956.742	$7.640 \cdot 10^{5}$
Нормована середньоквадратична помилка, ***NMSE***	0	$9.402 \cdot 10^{-10}$	$7.702 \cdot 10^{-4}$	$2.773 \cdot 10^{-4}$	0.250
L^p-норма, $p = 2$	0	0.057	51.551	30.931	874.01
Відношення сигнал/шум, ***SNR***	∞	$1.064 \cdot 10^{9}$	$1.298 \cdot 10^{3}$	$3.606 \cdot 10^{3}$	4.006
Макс. відношення с/шум, ***PSNR***	∞	$1.177 \cdot 10^{10}$	$1.437 \cdot 10^{4}$	$3.992 \cdot 10^{4}$	45.968
Якість звучання, ***AF***	1	≈1	0.999230	0.999723	0.750371
Норм. взаємна кореляція, ***NC***	1	≈1	0.999615	0.999806	1.255078
Якість кореляції, ***CQ***	$2.792 \cdot 10^{3}$	$2.792 \cdot 10^{3}$	$2.790 \cdot 10^{3}$	$2.791 \cdot 10^{3}$	$3.355 \cdot 10^{3}$
Структурний зміст, ***SC***	1	≈1	≈1	1.000110	0.568251
Загальне сигма-відношення сигнал/шум, ***GSSNR***	∞	$1.152 \cdot 10^{14}$	$1.51 \cdot 10^{20}$	$3.279 \cdot 10^{8}$	9.377
Сигма-відношення сигнал/шум, ***SSNR***	∞	140.6	201.8	85.2	9.721
Відношення сигнал/помилка, ***SER***	∞	$1.064 \cdot 10^{9}$	$1.298 \cdot 10^{3}$	$3.606 \cdot 10^{3}$	4.006
Подібність гістограм, ***HS***	0	240	13854	14708	15548

5.5.1. Методи довільного інтервалу

Існує щонайменше дві причини, за якими маніпулювання вільним місцем у певних випадках показує непогані результати. По-перше, зміна кількості пробілів у кінці текстового рядка не спричиняє істотних змін у значенні фрази або речення. По-друге, пересічний читач навряд чи помітить незначні модифікації вільного місця.

У [14] наведено три методи, що використовують для приховування даних вільне місце у тексті. Дані методи оперують з інтервалами між реченнями, пропусками в кінці текстових рядків та інтервалами між словами у тексті, вирівняному по ширині.

5.5.1.1. Метод зміни інтервалу між реченнями

Вказаний метод дозволяє вбудовувати повідомлення, що має двійковий формат, до тексту шляхом розміщення одного або двох пробілів після кожного символу завершення речення (наприклад, крапки у звичайному тексті або крапки з комою для коду програм на мові C++ тощо). Одиночним пробілом кодується, наприклад, біт «1», подвійним – біт «0». Проте, такий простий метод має низку недоліків. По-перше, він є неефективним, вимагаючи тексту значного обсягу для вбудовування незначної кількості біт (так, один біт, що можна приховати в одному реченні є еквівалентним швидкості передачі даних приблизно 1 біт даних на 160 байт текстового контейнера, за умови, що у середньому речення становить собою 2 рядки по 80 символів кожен). По-

друге, можливість приховування залежить від структури тексту-контейнера (деякі тексти, як наприклад, верлібри або вільні вірші характеризуються відсутністю стійких узгоджених або однозначних знаків завершення рядка). По-третє, деякі з текстових редакторів автоматично встановлюють після крапки наприкінці речення один-два пробіли. Зрештою, як зазначається у [14], непослідовне використання вільних місць є досить помітним для читача.

Наведемо приклад реалізації даного методу.

1) Нехай в якості контейнера використовується текст, фрагмент якого наведено на рис.5.88 (**C** := **READBIN**("C.txt", "byte")). Приховуване ж повідомлення має наступний зміст: **M** := "Алгоритм".

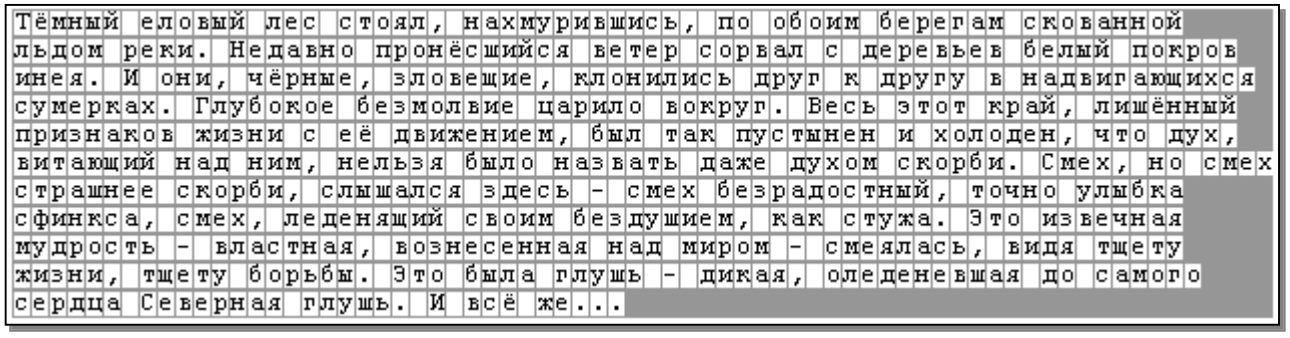

Тёмный еловый лес стоял, нахмурившись, по обоим берегам скованной
льдом реки. Недавно пронёсшийся ветер сорвал с деревьев белый покров
инея. И они, чёрные, зловещие, клонились друг к другу в надвигающихся
сумерках. Глубокое безмолвие царило вокруг. Весь этот край, лишённый
признаков жизни с её движением, был так пустынен и холоден, что дух,
витающий над ним, нельзя было назвать даже духом скорби. Смех, но смех
страшнее скорби, слышался здесь - смех безрадостный, точно улыбка
сфинкса, смех, леденящий своим бездушием, как стужа. Это извечная
мудрость - властная, вознесенная над миром - смеялась, видя тщету
жизни, тщету борьбы. Это была глушь - дикая, оледеневшая до самого
сердца Северная глушь. И всё же...

Рис.5.88. Фрагмент оригіналу тексту-контейнера, використовуваного для приховування даних

2) Змінною π := ". " (крапка + пробіл) позначимо фрагмент контейнера, який сигналізуватиме про закінчення речень. У ASCII-кодировці: $\mathbf{str2vec}(\pi)^T = (46\ \ 32)$.

Перевірку достатності речень у тексті для приховування заданої кількості біт повідомлення ($L_M := 8 \cdot \mathbf{strlen}(M)$, $L_M = 64$ біти) виконаємо за допомогою програмного модуля (М.124), який підраховує кількість елементів π в обраному тексті.

$$
\begin{aligned}
N_\pi := \;& N_\pi \leftarrow 0 \\
& \textbf{for}\ \ i \in 1 .. \mathbf{rows}(C) \\
& \quad \Pi \leftarrow \mathbf{submatrix}(C, i, i+1, 1, 1) \ \ \textbf{if}\ \ i < \mathbf{rows}(C) \\
& \quad N_\pi \leftarrow N_\pi + 1 \ \ \textbf{if}\ \ \Pi = \mathbf{str2vec}(\pi) \\
& N_\pi
\end{aligned}
\qquad \text{(М.124)}
$$

Для представленого на рис.5.88 фрагменту $N_\pi = 8$, що є достатнім для приховання одного символу повідомлення (8 біт).

3) Стеганографічне приховування виконує програмний модуль (М.125). На його початку рядок символів **M** перетворюється на вектор двійкових даних M_{vec_bin}. Для кожного елементу вектора M_{vec_bin} проводиться пошук кінця речення (елемент π) у представленому в ASCII-кодах тексті **C'**. Останній являє собою текст, який ще не зазнав модифікації вбудовуваними бітами. Паралельно формується стеганограма **S**, шляхом дописування в її кінець **i**-го елемента вектору **C'**.

У випадку знайдення фрагменту вектору **C'**, що відповідає вектору π, відбувається збереження подальшої частини тексту як немодифікованого (перевизначення змінної **C'**) і в залежності від значення поточного елемента M_{vec_bin} до стеганограми **S** дописується один або два пробіли (ASCII-код яких 32). Для запобігання можливому виникненню неоднозначності у тих випадках, коли в оригінальному тексті речення відділялися між собою крапкою і довільною кількістю пробілів, виконується видалення можливих пробілів на початку немодифікованого фрагменту **C'**.

По завершенні вбудовування всіх бітів повідомлення, до стеганограми **S** приписується текст **C'**, який залишився без змін.

```
S := | M_vec ← str2vec(M)
     | M_vec_bin ← D2B(M_vec_1)
     | for j ∈ 2 .. strlen(M)                                   if strlen(M) > 1
     |     M_vec_bin ← stack(M_vec_bin , D2B(M_vec_j))
     | C' ← C
     | for μ ∈ 1 .. 8·strlen(M)
     |     for i ∈ 1 .. rows(C')
     |         S_rows(S)+1 ← C'_i
     |         Π ← submatrix(C' , i , i + 1 , 1 , 1)   if i < rows(C')
     |         if M_vec_bin_μ = 1                                if Π = str2vec(π)
     |             C' ← submatrix(C' , i + 2 , rows(C') , 1 , 1)
     |             S_rows(S)+1 ← 32
     |             while C'_1 = 32
     |                 C' ← submatrix(C' , 2 , rows(C') , 1 , 1)
     |             break
     |         if M_vec_bin_μ = 0                                if Π = str2vec(π)
     |             C' ← submatrix(C' , i + 2 , rows(C') , 1 , 1)
     |             S_rows(S)+1 ← 32
     |             S_rows(S)+1 ← 32
     |             while C'_1 = 32
     |                 C' ← submatrix(C' , 2 , rows(C') , 1 , 1)
     |             break
     | stack(S , C')
```

(M.125)

Результат приховання першого символу повідомлення ("A"), двійковий вираз ASCII-коду якого: $\mathbf{D2B(str2vec("A"))}^{T} = (0_{LSB}\ 0\ 0\ 0\ 0\ 0\ 1\ 1_{MSB})$, зображено на рис.5.89.

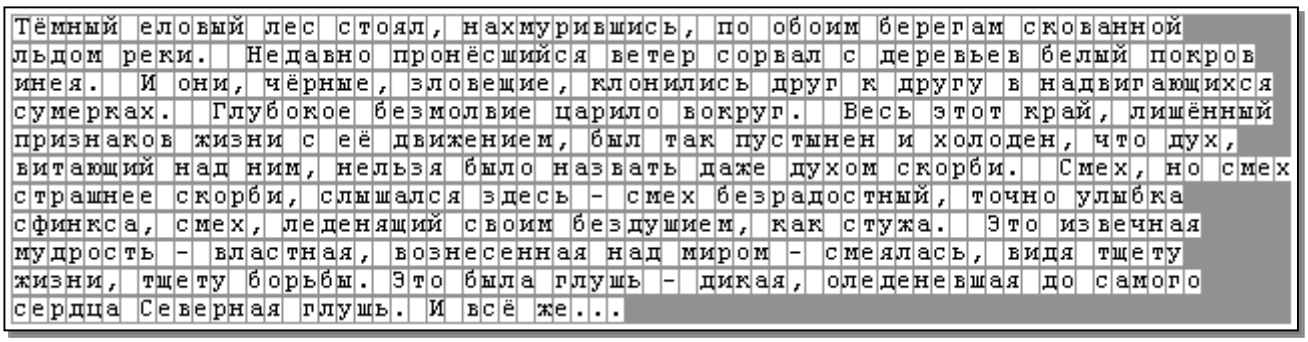


Рис.5.89. Фрагмент заповненого методом зміни інтервалу між реченнями текстового контейнера

4) Видобування прихованої інформації проводиться програмним модулем (М.126) при **S*** := **S**, **π*** := **π**. Алгоритм дій наступний: виконується пошук кінця речення (елемент **π***) у представленому в ASCII-кодах тексті **S***. У випадку знайдення, аналізується кількість пробілів після крапки, на підставі чого робиться висновок про значення поточного біту повідомлення, що видобувається. Наприкінці модуля вектор двійкових даних перетворюється на рядок символів.

```
M* := | μ ← 1
      | for i ∈ 1 .. rows(S*)
      |     Π ← submatrix(S* , i , i + 1 , 1 , 1)  if  i < rows(S*)
      |     if  Π = str2vec(π*)
      |         M*bin_μ ← 0  if  S*_{i+1} = S*_{i+2} = 32
      |         M*bin_μ ← 1  if  S*_{i+1} ≠ S*_{i+2}
      |         μ ← μ + 1
      | for j ∈ 1 .. rows(M*bin) ÷ 8
      |     M*vec_j ← B2D(submatrix(M*bin , 8·j − 7 , 8·j , 1 , 1))
      | vec2str(M*vec)
```
(M.126)

5.5.1.2. Метод зміни кількості пробілів в кінці текстових рядків

Другий метод використання вільних місць для вбудовування конфіденційних даних полягає у додаванні пробілів в кінець кожного текстового рядка. Кількість додаваних пробілів залежить від біту, що вбудовується. Два пробіли кодують один біт на рядок, чотири пробіли – два біти, вісім – три і т.д., істотно збільшуючи, порівняно з попереднім методом, кількість інформації, яку виникає можливість приховати. Додаткові переваги даного методу полягають у тому, що він може бути застосовний до будь-якого тексту; зміни у форматі є досить непомітними, оскільки вільні місця, що використовуються, є периферійними по відношенню до основного тексту. Недоліком даного (як, зрештою, і попереднього) методу є те, що деякі програми можуть ненавмисно видаляти додатково внесені пробіли. Крім того, характерним недоліком даного методу є неможливість видобування прихованих даних з паперової копії тексту.

Нами запропоновано наступний шлях реалізації даного методу.

1) Порожній текстовий контейнер і приховуване повідомлення ті самі, що й у попередньо розглянутому методі.

2) Вбудовування бітів повідомлення виконуємо за допомогою програмного модуля (М.127). Після одержання вектору двійкового представлення повідомлення, яке підлягає прихованню, виконується підрахунок кількості символів у кожному текстовому рядку **r** (нагадаємо, що кожен рядок завершується парою службових символів повернення каретки *CR* (ASCII-код 13) і переведення рядка *LF* (ASCII-код 10), що в звичайному режимі не виводяться на екран і на друк). Підрахована кількість заноситься до **r**-го елементу вектора **L**.

Стеганограма формується шляхом дописування після останнього елементу вектора **S**, що її представляє, значення **i**-го елементу вектора **C'** (останній, як і у попередньо розглянутому методі, являє собою текст, який ще не зазнав модифікації), якщо той не являє собою символ *CR*. В протилежному випадку здійснюється перевизначення змінної **C'** як такої, що містить немодифіковану частину тексту. Кількість біт **κ**, яку можна приховати у рядку визначається різницею між кількістю символів у найдовшому рядку тексту (максимальне значення вектора **L**) і кількістю символів у поточному (**r**-му) рядку. При цьому, якщо приховується біт «0», до стеганограми дописується значення 32 (звичайний пробіл), а якщо біт «1» – значення 160 (нерозривний пробіл). Після вбудовування **κ** біт, рядок завершується парою символів *CR*/*LF*. Після вбудовування всіх 8·**strlen(M)** біт повідомлення цикл переривається. Якщо рядок, що модифікувався останнім, не є останнім у тексті (загальну кількість рядків тексту можна визначити як кількість елементів вектора **L**), стеганограма доповнюється немодифікованим текстом **C'**.

```
S := | M_vec ← str2vec(M)
     | M_vec_bin ← D2B(M_vec_1)
     | for j ∈ 2 .. strlen(M)                                   if strlen(M) > 1
     |     M_vec_bin ← stack(M_vec_bin , D2B(M_vec_j))
     | n ← 0
     | r ← 1
     | for i ∈ 1 .. rows(C)                                                    }
     |     n ← n + 1   if  C_i ≠ 10                                           }
     |     if  C_i = 10                                                       }
     |         L_r ← n − 1                                                    }
     |         n ← 0                                                          }
     |         r ← r + 1                                                      }
     | C' ← C                                                                  } #
     | r ← 1
     | μ ← 1
     | while  μ ≤ 8·strlen(M)
     |       for  i ∈ 1 .. rows(C')
     |           S_rows(S)+1 ← C'_i   if  C'_i ≠ 13
     |           if  C'_i = 13
     |               C' ← submatrix(C', i + 2, rows(C'), 1, 1)   if  i+2 ≤ rows(C')
     |               κ ← 1
     |               while  max(L) − L_r ≥ κ
     |                       S_rows(S)+1 ← 32   if  M_vec_bin_μ = 0
     |                       S_rows(S)+1 ← 160   if  M_vec_bin_μ = 1
     |                       μ ← μ + 1
     |                       break   if  μ > 8·strlen(M)
     |                       κ ← κ + 1
     |               S_rows(S)+1 ← 13
     |               S_rows(S)+1 ← 10
     |               r ← r + 1
     |               break
     | S ← stack(S , C')   if  r < rows(L)
     | S
```

(M.127)

Результат вбудовування повідомлення **M** до контейнера **C** зображено на рис.5.90.

```
Тёмный еловый лес стоял, нахмурившись, по обоим берегам скованной
льдом реки. Недавно пронёсшийся ветер сорвал с деревьев белый покров
инея. И они, чёрные, зловещие, клонились друг к другу в надвигающихся
сумерках. Глубокое безмолвие царило вокруг. Весь этот край, лишённый
признаков жизни с её движением, был так пустынен и холоден, что дух,
витающий над ним, нельзя было назвать даже духом скорби. Смех, но смех
страшнее скорби, слышался здесь - смех безрадостный, точно улыбка
сфинкса, смех, леденящий своим бездушием, как стужа. Это извечная
мудрость - властная, вознесенная над миром - смеялась, видя тщету
жизни, тщету борьбы. Это была глушь - дикая, оледеневшая до самого
сердца Северная глушь. И всё же...
```

Рис.5.90. Фрагмент заповненого методом зміни кількості пробілів в кінці рядків текстового контейнера

3) Видобування бітів прихованих даних при **S*** := **S** здійснюється за допомогою програмного модуля (М.128). На всій довжині вектора **S*** ведеться пошук символів *CR*. У випадку знайдення проводиться аналіз символів, які розміщені до нього, на предмет відповідності їх кодів кодам звичайного (32) і нерозривного (160) пробілів. На підставі останніх формується вектор двійкових даних, який, у свою чергу, згодом конвертується на рядок символів **M*.**

```
M* := | μ ← 1
      | for i ∈ 1 .. rows(S*)
      |     if S*_i = 13
      |         | j ← i
      |         | while S*_(j-1) = 32 ∨ S*_(j-1) = 160
      |         |       j ← j - 1
      |         | for m ∈ j .. i - 1                    if j ≠ i
      |         |     | M*bin_μ ← 0   if S*_m = 32
      |         |     | M*bin_μ ← 1   if S*_m = 160
      |         |     | μ ← μ + 1
      | for j ∈ 1 .. rows(M*bin) ÷ 8
      |     M*vec_j ← B2D(submatrix(M*bin, 8·j - 7, 8·j, 1, 1))
      | vec2str(M*vec)
```

(М.128)

5.5.1.3. Метод зміни кількості пробілів між словами вирівняного по ширині тексту

Даний метод дозволяє приховувати дані у вільних місцях тексту, вирівняного по ширині. При цьому дані вбудовуються шляхом керованого обрання позицій, в яких буде розміщено додаткові пробіли. Один пробіл між словами інтерпретується як «0». Два пробіли – як «1». Метод дозволяє вбудувати у середньому по декілька біт до одного рядка.

Через обмеження, що накладаються вирівнюванням тексту по ширині, не кожен пробіл між словами може використовуватися для вбудовування даних. Для можливості прийняття однозначного рішення приймальною стороною при визначенні, які ж саме з пробілів між словами приховують вбудовану інформацію, а які є частиною оригінального тексту, у [14] запропоновано використовувати метод вбудовування, подібний до манчестерського кодування. Таке кодування групує біти попарно, причому «01» інтерпретується як «1», «10» – як «0», а пари «00» і «11» є порожніми. Наприклад, видобуте повідомлення «1010000111» зводиться до «001», тоді як «0011110011» є порожнім рядком.

Промоделюємо даний метод у програмі MathCAD.

1) Порожній текстовий контейнер **C** і приховуване повідомлення **M** є аналогічними використаним у вищерозглянутих методах.

2) Вбудовування повідомлення до контейнера здійснюється програмним модулем (М.129). Одержання вектору двійкового представлення приховуваного повідомлення та підрахунок кількості символів у **r**-му текстовому рядку виконується таким самим чином, що й у програмному модулі (М.127) (позначено символом «#»).

В процесі приховування **μ**-го біту повідомлення обчислюються індекси **i1** та **i2**, що обмежують **r**-й текстовий рядок у спільному векторі **C**. До рядку, що має максимальну кількість символів у своєму складі (назвемо його довгим, $L_r = \max(L)$), дані не вбудовуються. До всіх інших рядків (назвемо їх короткими, $\max(L) - L_r > 0$) вбудовування бітів виконується таким чином, щоб додавання пробілів розширило поточний рядок до розмірів довгого рядка.

Для виокремленого зі спільного вектору **C** підмасиву **c**, який містить ASCII-коди символів короткого **r**-го рядка формується масив **space**, розмірність якого відповідає загальній кількості пробілів у даному рядку, а кожні окремі елементи містять інформацію про порядкові

номери відповідних їм пробілів серед усіх символів рядку. Наприклад, при **r =** 1, для контейнера, зображеного на рис.5.88, $\mathbf{space}^{\mathbf{T}} = (7 \;\; 14 \;\; 18 \;\; 25 \;\; 39 \;\; 42 \;\; 48 \;\; 56)$.

З опису методу відомо, що один біт даних вбудовується до текстового контейнера шляхом модифікації пари пробілів, які охоплюють одне слово речення, – додаванням одного пробілу на початку або в кінці слова, в залежності від значення біту. Отже, кількість біт **Nb**, яку можна вбудувати до **r**-го рядка, визначається співвідношенням між кількістю можливих пар пробілів і кількістю символів, на яку короткий рядок "відстає" від довгого. Якщо $\mathbf{t}/2 < \mathbf{max}(\mathbf{L}) - \mathbf{L_r}$, для вбудовування біт можна використати лише $\mathbf{floor}(\mathbf{t}/2) - 1$ пар пробілів; інші використовуватимуться для подовження рядка до розмірів довгого – шляхом створення порожніх пар (для спрощення алгоритму, приймається, що видобування на певній ділянці рядка порожньої пари пробілів вказує на те, що у подальшій частині рядка вбудованих даних не міститься). Якщо $\mathbf{t}/2 \geq \mathbf{max}(\mathbf{L}) - \mathbf{L_r}$, для вбудовування біт використовується кількість пар пробілів, яка дорівнює $\mathbf{max}(\mathbf{L}) - \mathbf{L_r}$, адже саме цієї кількості символів не вистачає короткому рядку, щоб зрівнятися з довгим. Так, для обраного контейнера довгим є 6-й рядок, який містить 70 символів. 1-й рядок є коротким: $\mathbf{L}_1 = 65$. Для нього кількість можливих пар пробілів $\mathbf{t}/2 = 8/2 = 4$, що є меншим за $70 - 65 = 5$. Таким чином, **Nb** = 3.

Після визначення кількості пар пробілів, до яких можуть бути вбудовані біти повідомлення, з рядка виокремлюються сегменти $\mathbf{seg}_\tau$, що, відповідно, містять τ-пару пробілів. У нашому випадку для першого рядка одержуються наступні сегменти (для наочності, зображено відповідні ASCII-кодам символи):

$\mathbf{seg}_1 \rightarrow$ Тёмный еловый ; $\mathbf{seg}_2 \rightarrow$ лес стоял, ; $\mathbf{seg}_3 \rightarrow$ нахмурившись, по .

Для кожного сегменту формується вектор **ssp** з двох елементів, кожен з яких містить індекс відповідного йому пробілу. Наприклад, для зазначених вище сегментів:

$$\mathbf{ssp} = \begin{pmatrix} 7 \\ 14 \end{pmatrix} \text{ для } \mathbf{seg}_1; \quad \mathbf{ssp} = \begin{pmatrix} 4 \\ 11 \end{pmatrix} \text{ для } \mathbf{seg}_2; \quad \mathbf{ssp} = \begin{pmatrix} 14 \\ 17 \end{pmatrix} \text{ для } \mathbf{seg}_3.$$

За допомогою додатково визначеної функції **ins_vec2vec** формуються нові сегменти, в яких додано один пробіл до або до першого, або до другого пробілу (в залежності від значення поточного вбудовуваного біта). У загальному випадку функція **ins_vec2vec**(**A**, **B**, ρ) (див. програмний модуль (М.130)) дозволяє проводити вбудовування вектора **B** до вектора **A**, починаючи з позиції ρ. У нашому застосуванні в якості вектора, до якого відбувається вбудовування, виступає сегмент $\mathbf{seg}_\tau$. Вбудовується ж ASCII-код пробілу (**B** = 32) після першого ($\rho = \mathbf{ssp}_1+1$), якщо $\mathbf{M_{vec\_bin}} = 1$, або після другого ($\rho = \mathbf{ssp}_2+1$), якщо $\mathbf{M_{vec\_bin}} = 0$, пробілу сегмента.

Зокрема, для наведених вище сегментів одержуються наступні, модифіковані сегменти (нагадаємо, що першим вбудовується символ "A" повідомлення, ASCII-код якого у двійковому еквіваленті: $\mathbf{D2B}(\mathbf{str2vec}(\text{"A"}))^{\mathbf{T}} = (0_{LSB}\ 0\ 0\ 0\ 0\ 0\ 1\ 1_{MSB})$. Вбудовування проводиться починаючи з наймолодших біт):

$\mathbf{seg_{new}}_1 \rightarrow$ Тёмный еловый ; $\mathbf{seg_{new}}_2 \rightarrow$ лес стоял, ; $\mathbf{seg_{new}}_3 \rightarrow$ нахмурившись, по .

Одержані нові сегменти формують **r**-й рядок стеганограми ($\mathbf{s_r}$).

Після вбудовування останнього (**Nb**-го) біта, дозволеного у даному рядку, і формування початку рядка $\mathbf{s_r}$, залишається сегмент оригінального рядка **c**, який не зазнав модифікації – $\mathbf{seg_{end}}$. Для першого рядка:

$\mathbf{seg_{end}} \rightarrow$ обоим берегам скованной.

Якщо $\mathbf{t}/2 < \mathbf{max}(\mathbf{L}) - \mathbf{L_r}$, до перших двох пробілів даного завершального сегменту додаються ще по пробілу, що дозволить після приєднання цього сегменту вже наявної частини рядка $\mathbf{s_r}$ одержати рядок, кількість символів у якому дорівнюватиме $\mathbf{max}(\mathbf{L})$. Якщо ж виконується умова $\mathbf{t}/2 \geq \mathbf{max}(\mathbf{L}) - \mathbf{L_r}$, завершальний сегмент не змінюється. Для розгляданого вище прикладу, останній сегмент зазнає модифікації, після якої набуває наступного вигляду:

$\mathbf{seg_{end\_new}} \rightarrow$ обоим берегам скованной.

```
S := # (M.127)
Σ ← 0 / r ← 1 / μ ← 1
while μ ≤ 8·strlen(M)
    break if r > rows(L)
    if r = 1
        i1 ← 1
        i2 ← L_r + 2
    if r ≥ 2
        i1 ← i2 + 1
        i2 ← i1 + L_r + 1
    if (max(L) − L_r) > 0
        c ← submatrix(C, i1, i2, 1, 1)
        Σ ← Σ + rows(c)
        t ← 0
        for i ∈ 1.. rows(c)
            if c_i = 32
                space_{t+1} ← i
                t ← t + 1
        Nb ← floor(t ÷ 2) − 1 if t ÷ 2 < (max(L) − L_r)
        Nb ← (max(L) − L_r) if t ÷ 2 ≥ (max(L) − L_r)
        τ ← 1
        while τ ≤ Nb
            break if μ > 8·strlen(M)
            sp2_τ ← space_{2·τ}
            seg_τ ← submatrix(c, 1, sp2_τ, 1, 1) if τ = 1
            seg_τ ← submatrix(c, sp2_{τ−1} + 1, sp2_τ, 1, 1) if τ > 1
            g ← 1
            for m ∈ 1.. rows(seg_τ)
                if (seg_τ)_m = 32
                    ssp_g ← m
                    g ← g + 1
            seg_new ← ins_vec2vec(seg_τ, 32, ssp_1 + 1) if M_vec_bin_μ = 1
            seg_new ← ins_vec2vec(seg_τ, 32, ssp_2 + 1) if M_vec_bin_μ = 0
            s_r ← seg_new if τ = 1
            s_r ← stack(s_r, seg_new) if τ > 1
            if τ = Nb
                seg_end ← submatrix(c, sp2_τ + 1, rows(c), 1, 1)
                if t ÷ 2 < (max(L) − L_r)
                    g ← 1
                    for m ∈ 1.. rows(seg_end)
                        if seg_end_m = 32
                            ssp_g ← m
                            g ← g + 1
                    seg_end_new ← ins_vec2vec(seg_end, 32, ssp_1 + 1)
                    seg_end_new ← ins_vec2vec(seg_end_new, 32, ssp_2 + 1)
                seg_end_new ← seg_end if t ÷ 2 ≥ (max(L) − L_r)
                s_r ← stack(s_r, seg_end_new)
            μ ← μ + 1
            τ ← τ + 1
    if L_r = max(L)
        c ← submatrix(C, i1, i2, 1, 1)
        Σ ← Σ + rows(c)
        s_r ← c
    r ← r + 1
S ← s_1
for r ∈ 2.. rows(s)
    S ← stack(S, s_r)
S ← stack(S, submatrix(C, Σ + 1, rows(C), 1, 1)) if rows(s) < rows(L)
S
```

(M.129)

$$\text{ins\_vec2vec}(A, B, \rho) \equiv \left| \begin{array}{l} A' \leftarrow \text{submatrix}(A, 1, \rho - 1, 1, 1) \\ \text{if } \rho \le \text{rows}(A) \\ \quad \left| \begin{array}{l} A'' \leftarrow \text{submatrix}(A, \rho, \text{rows}(A), 1, 1) \\ A''' \leftarrow \text{stack}(A', B, A'') \end{array} \right. \\ A''' \leftarrow \text{stack}(A', B) \quad \text{if } \rho = \text{rows}(A) + 1 \\ A''' \end{array} \right. \tag{M.130}$$

Як вже зазначалося, модифікований або ні завершальний сегмент дописується в кінець **r**-го рядка стеганограми.

Одержані рядки стеганограми об'єднуються до спільного масиву **S.**

Паралельно, у програмному модулі (М.129) проводився підрахунок кількості символів оброблених рядків **Σ**. Це дозволяє визначити індекс символу оригінального тексту, після якого останній не зазнав змін (через завершення повідомлення раніше, ніж було досягнуто кінець тексту) і дописати немодифікований текст в кінець стеганограми.

Фрагмент результату вбудовування повідомлення **M** до текстового контейнера **C** наведено на рис.5.91. Неважко помітити, що у даному фрагменті приховано 24 біти повідомлення, що відповідає його першим трьом символам ("Алг"). Так, у 1-му рядку приховано біти 000, у 2-му – 00, у 3-му – 0, у 4-му – 11 (маємо символ "А"), у 5-му – 11, у 6-му – жодного біта, у 7-му – 010, у 8-му – 111 (маємо символ "л"), у 9-му – 110, у 10-му – 0011, і, нарешті, у 11-му – 1 (маємо символ "г").

Рис.5.91. Фрагмент заповненого методом зміни кількості пробілів між словами вирівняного по ширині текстового контейнера

3) Можливий програмний модуль видобування з контейнера **S*** := **S** вбудованих до нього бітів, представлено нижче.

На початку модуля виконується підрахунок кількості символів у кожному з рядків тексту **S***. На підставі одержаного при цьому масиву **L** обчислюються індекси **i1** та **i2**, що обмежують **r**-й текстовий рядок у спільному векторі **S***.

Для виокремленого з вектору **S*** рядку-підмасиву **s** формується вектор **space**, розмірність якого відповідає загальній кількості пробілів у даному рядку, а елементи містять інформацію про порядкові номери відповідних їм пробілів серед множини символів рядку.

На основі одержаного вектора **space** формується вектор **b**, значення кожного елементу якого залежить від результату обчислення різниці між наступним і поточним значеннями елементів вектора **space**. Якщо дана різниця дорівнює одиниці, елемент $\mathbf{b_j}$ приймає значення 1. Якщо ж різниця є більшою за одиницю, елементу $\mathbf{b_j}$ присвоюється значення 0.

З одержаного вектору **b** виокремлюються підмасиви з трьох елементів (так звані трибіти), на основі яких робиться висновок про поточне значення вбудованого біту: якщо трибіт становить собою послідовність (0 1 0) – було приховано біт «0»; якщо ж (1 0 0) – приховано біт «1». В усіх інших випадках приймається рішення про відсутність вбудованих бітів. Сформована при цьому двійкова послідовність перетворюється на рядок символьних даних, який і повертається змінній **M***.

```
M* := | n ← 0
      | r ← 1
      | for i ∈ 1 .. rows(S*)
      |     | n ← n + 1  if  S*_i ≠ 10
      |     | if  S*_i = 10
      |     |     | L_r ← n − 1
      |     |     | n ← 0
      |     |     | r ← r + 1
      | r ← 1
      | μ ← 1
      | while  r ≤ rows(L)
      |     | if  r = 1
      |     |     | i1 ← 1
      |     |     | i2 ← L_r + 2
      |     | if  r ≥ 2
      |     |     | i1 ← i2 + 1
      |     |     | i2 ← i1 + L_r + 1
      |     | s ← submatrix(S* , i1 , i2 , 1 , 1)
      |     | t ← 0
      |     | for  i ∈ 1 .. rows(s)
      |     |       if  s_i = 32
      |     |         | space_{t+1} ← i
      |     |         | t ← t + 1
      |     | for  j ∈ 1 .. t − 1
      |     |     | b_j ← 0  if  (space_{j+1} − space_j) > 1
      |     |     | b_j ← 1  if  (space_{j+1} − space_j) = 1
      |     | for  τ ∈ 1 .. rows(b) / 3
      |     |     | tribit ← submatrix(b , 3·τ − 2 , 3·τ , 1 , 1)
      |     |     | if  tribit^T = (0  1  0)
      |     |     |     | M*bin_μ ← 0
      |     |     |     | μ ← μ + 1
      |     |     | if  tribit^T = (1  0  0)
      |     |     |     | M*bin_μ ← 1
      |     |     |     | μ ← μ + 1
      |     |     | break  otherwise
      |     | r ← r + 1
      | for  j ∈ 1 .. rows(M*bin) ÷ 8
      |       M*vec_j ← B2D(submatrix(M*bin , 8·j − 7 , 8·j , 1 , 1))
      | vec2str(M*vec)
```

(M.131)

Розглянуті методи довільного інтервалу є ефективними, за умови, що текст представлено у форматі ASCII. Як було вже зазначено вище, деякі дані можуть виявитися втраченими після роздрукування тексту. Друковані документи висувають до приховування даних такі вимоги, що далеко виходять за можливості текстового файлу при кодуванні ASCII. Приховування даних у "жорстких" копіях тексту виконується шляхом незначних змін відстані між словами і окремими літерами, зміною позицій базових ліній (ліній, на яких лежать найнижчі елементи літер або знаків пунктуації рядка), зміною форм літер тощо.

5.5.2. Синтаксичні і семантичні методи

Той факт, що вільне місце для вбудовування обирається довільно, є одночасно перевагою і недоліком з точки зору прихованості даних. Пересічний читач може й не помітити маніпуляції з текстом, тоді як текстовий редактор може автоматично змінити кількість і розміщення пробілів, руйнуючи таким чином приховані дані. Низька стійкість до атак, в світлі можливого переформатування документу, є однією з причин пошуку інших методів вбудовування даних до текстових контейнерів. Крім того, синтаксичні і семантичні методи взагалі жодним чином не використовують вільні місця у тексті, докорінно відрізняючись від розглянутих вище методів. Проте, всі ці методи можуть використовуватися одночасно, дублюючи або доповнюючи один одного.

До *синтаксичних методів* текстової стеганографії відносяться методи зміни пунктуації та методи зміни структури і стилю тексту [14]. Існує чимало випадків, коли правила пунктуації є неоднозначними і відступ від них не суттєво не впливає на загальний зміст тексту. Так, наприклад, фрази "*червоний, зелений, синій*" та "*червоний, зелений і синій*" є еквівалентними одна одній. Той факт, що вибір подібних форм є довільним (але довільним, звичайно, з огляду на використовуваний в якості контейнера текст, оскільки зрозуміло, що стеганосистема, побудована на основі видозміни широковідомого тексту навряд чи може вважатися надійною), й використовується при побудові стеганосистем на основі синтаксичних методів. Періодична зміна форм може бути поставлена у відповідність до двійкових даних. Наприклад, поява у тексті форми перерахування із сполучником "*і*" розуміє під собою вбудований біт «1», у той час відсутність сполучника при перерахуванні говорить про те, що було вбудовано біт «0». Серед інших прикладів – використання скорочень і абревіатур. Середня швидкість передавання даних такими методами становить декілька біт на один кілобайт тексту [14].

Проте, у той час як писемна мова надає достатньо можливостей для синтаксичного приховування даних, ці можливості зникають у характерних прозаїчних творах. Крім того, хоча деякі з правил пунктуації і є неоднозначними, їхнє суперечливе використання може стати об'єктом уваги для цензора. Також можливі випадки, коли зміна пунктуації призводить до зниження зрозумілості тексту або ж до набування текстом протилежного смислу. Тому автори [14] рекомендують з обережністю застосовувати цей метод.

До синтаксичних методів також відносяться методи зміни стилю та структури тексту без значної зміни його смислового навантаження. Наприклад, речення "*Існує чимало випадків, коли правила пунктуації є неоднозначними*" можна сформулювати як "*Правила пунктуації є неоднозначними у багатьох випадках*". Такі методи, порівняно з методами зміни пунктуації, є більш непомітними для сторонніх, але можливість їх використання обмежена складністю автоматизування процесу стеганографічного вбудовування/видобування бітів повідомлення.

Семантичні методи є подібними до синтаксичних. Замість того, щоб вбудовувати двійкові дані, використовуючи двозначність граматичної форми, дані методи призначають два синоніми, які відповідають значенням бітів. Наприклад, слово "*але*" може бути поставлене у відповідність до «0», а слово "*однак*" – до «1». Для проведення приховування з використанням семантичних методів необхідна наявність таблиці синонімів. Крім того, як зазначається у [14], якщо слову відповідає досить велика кількість синонімів, виникає можливість одночасного кодування більшої кількості бітів. Скажімо, вибір між синонімами "*секретний*", "*таємний*", "*прихований*", "*конфіденційний*", "*негласний*", "*невідомий*" дає можливість представити три біти даних. Проблеми можуть виникнути, коли бажанню вбудувати біт інформації перешкоджає нюанс значення слова.

5.6. Системні вимоги

Для ефективного застосування пропонованих програмних комплексів рекомендуються наступні системні вимоги до робочого місця.

Мінімальна конфігурація: процесор Intel Pentium III / Athlon ~ 1 ГГц, оперативна пам'ять 256 Мб, відеоадаптер з 32 Мб, операційна система Microsoft Windows XP, універсальна математична система MathCAD v.11 або 12, ~ 400 Мб вільного простору на диску (переважно для встановлення системи MathCAD).

Рекомендована конфігурація: процесор Intel Pentium 4 / Athlon XP ~ 2.4 ГГц, оперативна пам'ять 512 Мб, відеоадаптер з 64 Мб, аудіоплата з периферією (для можливості слухової оцінки результатів приховання даних у звукових файлах, а також запису звуків через мікрофон), ОС Microsoft Windows XP, універсальна математична система MathCAD v.12, ~ 400 Мб вільного простору на диску.

Для формування/обробки зображень використовується будь-який графічний редактор. У більшості випадків достатнім є наявність графічного редактору *MS Paint*, що поставляється разом з ОС MS Windows XP і дозволяє працювати з точковими зображеннями формату JPG, GIF або BMP. За допомогою MS Paint можна моделювати такі спотворення як масштабування зображення, його поворот, нахилення, фрагментація, відображення по горизонталі чи вертикалі, інвертування кольорів. Для можливості моделювання атак на зразок додавання гаусівського шуму, фільтрації, експозиції, зміни глибини кольорів, масштабування зображення з використанням різних типів фільтрів передискретизації тощо крім власне вбудованих функцій MathCAD по обробці зображення, рекомендується наявність одного із спеціалізованих пакетів обробки зображення: *ACD FotoCanvas, Adobe Photoshop, Deformer, Focus Photoeditor, Gimp, JPEG Imager, PhotoStudio* або ін.

Для роботи з аудіостеганометодами у більшості випадків достатньо можливостей програми *Звукозапис/Sound Recorder*, що є прикладною програмою з інструментарію ОС MS Windows XP. Дана програма дозволяє здійснювати запис, змішування, відтворення і редагування звукозаписів. Крім того, вона дозволяє виконувати наступні операції над імпортованим аудіофайлом: видалення частини звукозапису, зміна швидкості і гучності відтворення, зміна напрямку відтворення, зміна і конвертація типів звукозаписів, додавання луни. Для моделювання спотворень рекомендується використовувати такі програми як *Nero Wave Editor v.2.0.0.55* з пакету *NERO, AudioEdit Deluxe, SoundForge,* GoldWave, *AV Voice Changer Software, Audio Editor Gold* та ін.

При дослідженні методів лінгвістичної стеганографії можуть використовуватися такі текстові редактори як *Notebook, WordPad, MS Office Word* тощо.

5.7. Висновки

У даному розділі було докладно розглянуті відомі на сьогодні стеганографічні методи приховання даних у нерухомих зображеннях, в аудіосигналах та текстових файлах. Наведено приклади програмних комплексів, що демонструють принципи, закладені до основи більше ніж двадцяти методів стеганографічного приховування інформації в просторовій, часовій, частотній областях використовуваного контейнера. Розробка комплексів проведена з використанням популярної математичної системи MathCAD.

Всі етапи приховування супроводжуються відповідними програмними модулями (більше 130). Отримані авторами під час моделювання результати належним чином проілюстровані. Наведено наслідки можливих атак на стеганоповідомлення та рекомендації щодо захисту від них. Також обчислено основні показники візуального та звукового спотворення, що дозволяє проводити аналіз оптимальності обраного формату контейнера для приховування визначених даних.

ЗАГАЛЬНІ ВИСНОВКИ

Як показує практика, за останні декілька років актуальність проблеми інформаційної безпеки невпинно зростала, постійно стимулюючи при цьому пошук нових методів захисту інформації. Збільшення попиту на ефективні системи захисту інформації є наслідком не лише всім зрозумілої зацікавленості державних структур, яким, як правильно відзначено у [5], необхідно тепер протистояти як розвідкам інших держав, так і внутрішнім противникам – адептам світового тероризму чи хакерства, але й очевидним бажанням керівників великих і малих фірм та, зрештою, пересічних громадян вберегти наявні в них конфіденційні дані від витоку і розголошення. Аналіз існуючої тенденції дає можливість стверджувати, що й в наступні найближчі роки інтерес до впровадження і розвитку методів ефективного захисту інформації тільки зростатиме, чому, зокрема, сприятиме і бурхливий розвиток інформаційних технологій, який ми спостерігаємо сьогодні.

Зробити помітний внесок у загальну справу збереження державної таємниці, впевненості фірми у чесній грі конкурентів, а громадянина – у дієвості призабутого на сьогодні такого поняття як свобода особистості, наряду з іншими покликані й стеганографічні методи захисту інформації, зокрема, методи комп'ютерної стеганографії, теоретичні та практичні основи якої викладено у пропонованому підручнику. Одержані при цьому результати полягають в наступному:

– проведено аналіз спеціалізованих літературних джерел та ресурсів мережі Internet щодо перспективних напрямків, за якими можливе використання стеганографії як інструменту захисту інформації в автоматизованих системах обробки даних, що дозволило на відміну від поширеної у переважній більшості праць схеми ув'язнених перейти до схеми стеганографічної системи, яка відповідає основним принципам теорії зв'язку;

– на підставі дослідження відомих публікацій вітчизняних і закордонних авторів здійснено системне викладення проблем надійності і стійкості довільної стеганографічної системи по відношенню до видів здійснюваних на неї атак, а також оцінки пропускної здатності каналу прихованого обміну даними, яким, по суті, є стеганосистема. Виділено як спільні, так і характерні лише для стеганографічних систем види атак, на можливість існування яких необхідно зважати при організації каналу прихованого зв'язку. Наведено результати існуючих інформаційно-теоретичних досліджень проблеми інформаційного приховання у випадку активної протидії порушника. В результаті цього було закладено обґрунтовану теоретичну базу для розробки комп'ютерних систем стеганографічного приховання конфіденційної інформації;

– викладені принципи, що покладені до основи відомих на сьогодні стеганографічних методів, спрямованих на приховання конфіденційних даних у комп'ютерних файлах графічного, звукового та текстового форматів (розглянуто більше двадцяти методів);

– для зазначених методів проаналізовано особливості відповідних апаратів людини (зорового і слухового), зроблено акцент на характерні нюанси, які дозволяють скористатися існуючими обмеженостями ЗСЛ і ССЛ в стеганографічних цілях. Сформульовано рекомендації стосовно можливих способів вбудовування бітів приховуваного повідомлення до контейнера з метою підвищення рівня прихованості вбудованих даних при застосуванні відомих алгоритмів стеганоаналізу;

– для демонстрації принципів, закладених у методи стеганографічного захисту інформації, наведено приклади комп'ютерних стеганосистем на їх основі, розроблених з використанням потужної математичної системи MathCAD. Всі етапи приховування супроводжені відповідними програмними модулями, загальна кількість яких перевищила 130. Одержані у ході моделювання результати проілюстровані значною кількістю графічного матеріалу. Проведено обчислення показників спотворення медіаконтейнерів при вбудовуванні до них бітів приховуваних даних, що дозволяє проводити аналіз оптимальності обрання того або іншого стеганометоду або формату контейнера;

– розроблені системи дозволяють проводити стеганографічне приховання файлів будь-якого формату у файлах растрового зображення форматів BMP, JPG, GIF, у файлах ІКМ аудіосигналу формату WAV, а також у текстових файлах. Основні вимоги, що при цьому повинні виконуватися, – обрання файлу-контейнеру належного об’єму і, зрештою, апаратна спроможність використовуваної обчислювальної системи.

Порівняння стеганографії з технологіями криптографічного захисту інформації дозволяє припустити їх взаємну інтеграцію вже протягом найближчого майбутнього. При цьому виникне можливість позбутися слабких сторін відомих на сьогодні методів захисту інформації і розробити більш ефективні й оптимальні з позицій обчислювальної складності і стійкості до зламу нові методи забезпечення інформаційної безпеки.

На завершення, не можна не погодитися з авторами [5] щодо перспективного майбутнього комп’ютерної стеганографії, оскільки з часу її зародження пройшло лише 10 років, а як показує нам історія, час, за який технологія з моменту виникнення досягає стадії поширеного промислового використання, зазвичай становить близько 20 років.

СПИСОК ЛІТЕРАТУРИ

1. Артёхин Б.В. *Стеганография* // Журнал "Защита информации. Конфидент". — 1996. — №4. — С. 47-50.

2. D. Kahn, *The Codebreakers: The Story of Secret Writing*. New York, Macmillan, 1983.

3. Хорошко В.О., Азаров О.Д., Шелест М.Є., Яремчук Ю.Є. *Основи комп'ютерної стеганографії* : Навч. посіб. для студентів і аспірантів. — Вінниця: ВДТУ, 2003.

4. Барсуков В.С., Романцов А.П. *Компьютерная стеганография: вчера, сегодня, завтра. Технологии информационной безопасности XXI века*. — матеріали Internet-ресурсу "Специальная техника" (http://st.ess.ru/).

5. Грибунин В.Г., Оков И.Н., Туринцев И.В. *Цифровая стеганография*. — М.: Солон-Пресс, 2002.

6. A. Kerckhoffs, *La Cryptographie Militaire*. Journal des sciences militaires, pp: 5–83, Jan. 1883, pp: 161-191, Feb. 1883.

7. Gustavus J. Simmons, *The Prisoner's Problem And The Subliminal Channel*, Advances in Cryptology: Proc. Workshop on Communications Security (Crypto'83, David Chaum, ed.), Plenum Press, 1984, pp. 51-67.

8. Gustavus J. Simmons, *The History of Subliminal Channels*. // IEEE Journal on Selected Areas of Communications. 1998. Vol.16, №4. pp. 452-461.

9. J. Fridrich, R. Du, M. Long, *Steganalysis of LSB Encoding in Color Images*, Proceedings of ICME 2000, New York City, July 31 – August 2, New York, USA.

10. B. Pfitzmann, *Information Hiding Terminology*. In: Information Hiding, Springer Lecture Notes in Computer Science, v.1174, 1996, pp. 347-350.

11. B. Schneier, *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, 2nd ed. New York // John Wiley and Sons, 1996.

12. S. Craver, *On Public-Key Steganography in the Presence of an Active Warden* // Technical report RC 20931, IBM, 1997. 13p.

13. Ross J. Anderson, *Stretching the Limits of Steganography*. In: Information Hiding, Springer Lecture Notes in Computer Science, v.1174, 1996, pp. 39-48.

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

15. C. Cachin, *An Information-Theoretic Model for Steganography*. In: Information Hiding — 2nd International Workshop, Springer as Lecture Notes in Computing Science, vol.1525, April 1998, pp.306-318.

16. P. Bassia, I. Pitas, *Robust Audio Watermarking In The Time Domain* // Department of Informatics, University of Thessalonica (http://poseidon.csd.auth.gr/voyatzis/creus.zip).

17. R. Popa, *An Analysis of Steganographic Techniques*. The Polytechnic University of Timisoara, Faculty of Automatics and Computers, Department of Computer Science and Software Engineering. 1998.

18. N.F. Johnson, S. Jajodia, *Steganalysis of Images Created using Current Steganography Software*, Workshop on Information Hiding Proceedings, Portland Oregon, Springer as Lecture Notes in Computer Science, vol.1525, 1998.

19. Husrev T. Sencar , Mahalingam Ramkumar, Ali N. Akansu, *Data Hiding Fundamentals And Applications. Content Security In Digital Multimedia*. ELSEVIER science and technology books, 2004. 364p.

20. Neil F. Johnson, Zoran Duric, Sushil Jajodia, *Information Hiding : Steganography and Watermarking. - Attacks and Countermeasures*. Kluwer Academic Publishers. 2001. 160p.

21. S. Katzenbeisser, Fabien A. P. Petitcolas (Editors), *Information Hiding Techniques for Steganography And Digital Watermark*. Artech House Publishers. 1999. 220p.

22. Генне О.В. *Основные положения стеганографии* // Журнал "Защита информации. Конфидент", №3, 2000.

23. Кустов В.Н., Федчук А.А. *Методы встраивания скрытых сообщений* // Журнал "Защита информации. Конфидент", №3, 2000, с.34.

24. Быков С.Ф. *Алгоритм сжатия JPEG с позиции компьютерной стеганографии* // "Защита информации. Конфидент", №3, 2000, с.26.

25. Очков В.Ф. *MathCAD 12 для студентов и инженеров*. — СПб.: BHV, 2005.
26. Дьяконов В.П. *Энциклопедия MathCAD 2001i и MathCAD 11*. — М.: Солон-Пресс, 2004.
27. Kefa Rabah, *Steganography – The Art of Hiding Data*. Information Technology Journal 3 (3): pp. 245-269. 2004.
28. D. Verton, *Experts Debate Biggest Network Security Threats*. USA Today, 12 Apr. 2002. (http://www.usatoday.com/life/cyber/tech/cw1.htm).
29. K. Maney, *Bin Laden's Messages Could Be Hiding In Plain Sight*. USA Today, 19 Dec. 2001. (http://www.usatoday.com/life/cyber/ccarch/2001/12/19/maney.htm).
30. R. Mohanakrishnan, *Steganography: Snake In the Grass*. The Hindu Newspaper (India), Dec. 6, 2001.
31. D.L. Schilling, ed., *Meteor Burst Communications: Theory And Practice*. Wiley series in telecommunications, New York, USA: Wiley, 1993.
32. John H. Nugent, Mahesh S. Raisinghani, *The Information Technology And Telecommunications Security Imperative: Important Issues And Drivers*. Journal of Electronic Commerce Research, Vol.3, №1, 2002.
33. Закон України *"Про захист інформації в інформаційно-телекомунікаційних системах"*, №2594-IV від 31 травня 2005 року.
34. Анин Б.Ю. *Защита компьютерной информации*. — СПб.: BHV, 2000.
35. Соколов А.В., Шаньгин В.Ф. *Защита информации в распределённых корпоративных сетях и системах*. — М.: ДМК, 2002.
36. Хорошко В.О., Огаркова І.М., Чирков Д.В., Голего А.Г., Горохова Т.Б. *Термінологічний довідник з питань технічного захисту інформації*. / За ред. проф. Хорошка В.О. — К.: Ей-Бі-Сі, 2002.
37. Голубєв В.О., Гавловський В.Д., Цимбалюк В.С. *Інформаційна безпека: проблеми боротьби зі злочинами у сфері використання комп'ютерних технологій*. — Запоріжжя: Просвіта, 2001. — с.198-201.
38. Інформаційний ресурс дослідницької групи "CNews Analytics" (http://www.cnews.ru/).
39. Інформаційний ресурс дослідницького центру "DataPro Research Corporation". (http://www.datapro.com/).
40. N.F. Johnson, S. Jajodia, *Steganalysis: The Investigation of Hidden Information*, IEEE Information Technology Conference, Syracuse, New York, USA, Sept. 1st-3rd. 1998.
41. S. Voloshynovskiy, S. Pereira, V. Iquise, T. Pun, *Attack Modelling: Towards a Second Generation Watermarking Benchmark*. // Preprint. University of Geneva, 2001. 58p.
42. Грибунин В.Г. *Критерии оценки надёжности паролей*. — М.: РУСКАРД, 2003. (http://www.ruscard.org/).
43. Офіційний сайт Національного Інституту Стандартів і Технологій (НІСТ) США — http://www.nist.gov/.
44. M. Ramkumar, *Data Hiding in Multimedia*. PhD Thesis. New Jersey Institute of Technology, 1999. 72p.
45. C.E. Shannon, *A Mathematical Theory of Communication*. Bell System Technical Journal, 27 (1948), p.p. 379-423, 623-656.
46. Marvel L. *Image Steganography for Hidden Communication*. PhD Thesis. Univ.of Delaware, 1999. 115p.
47. Cox J., Miller M., McKellips A. *Watermarking as Communications With Side Information* // Proceedings of the IEEE. 1999. Vol.87. №7. P.1127-1141.
48. Барсуков В.С. *Стеганографические технологии защиты документов, авторских прав и информации* // Обзор специальной техники. — 2000. — №2. — С.31-40.
49. M. Kutter and F.A.P. Petitcolas, *A Fair Benchmark For Image Watermarking Systems*. Electronic Imaging '99. Security and Watermarking of Multimedia Contents, vol. 3657, Sans Jose, CA, USA, 25-27 January 1999.
50. Khalid Sayood, *Introduction to Data Compression*, chapter 7, p. 142. Morgan Kaufmann Publishers, 1996.
51. Ahmet M. Eskicioglu and Paul S. Fisher. *Image Auality Measures And Their Performance*. IEEE Transactions on Communication, 43(12): 2959-2965, December 1995.

52. P.R.R.L. Nunes, A. Alcaim, and M.R.L. Fragoso da Silva. *Quality Measures of Compressed Images for Classification Purposes*. Technical Report CCR-146, IBM Brasil, Rio Scientific Center, P.O. Box 4624, 20.0001 Rio de Janeiro, Brazil, October 1992.
53. S. Winkler. *A Perceptual Distortion Metric for Digital Color Video*. In: SPIE Proceedings of Human Vision and Electronic Imaging, vol. 3644, San Jose, CA, January 1999.
54. S. Winkler. *A Perceptual Distortion Metric for Digital Color Images*. In: Proc. ICIP, vol. 3, pp. 399-403, Chicago, IL, October 1998.
55. C.J. van den Branden Lambrecht and J.E. Farrell. *Perceptual Quality Metric for Digitally Coded Color Images*. In: Proceeding of EUSIPCO, pp. 1175-1178, Trieste, Italy, September 1996.
56. S.J.P. Westen, R.L. Lagendijk, and J. Biemond. *Perceptual Image Quality Based on a Multiple Channel HVS Model*. In: Proceeding of ICASP, vol. 4, pp. 2351-2354, 1995.
57. I.J. Cox, J. Kilian, F.T. Leighton and T. Shamoon, *Secure Spread Spectrum Watermarking for Multimedia*. IEEE Trans. on Image Processing, Vol. 6, No.12, December 1997.
58. L. von Ahn and N.J. Hopper, *Public-Key Steganography*. In: Advances in Cryptology: Eurocrypt'2004 (C. Cachin and J. Camenisch, eds.), vol. 3027 of Lecture Notes in Computer Science, pp. 322–339, Springer, 2004.
59. M. Backes, C. Cachin, *Public-Key Steganography with Active Attacks*. IBM Research Zurich Research Laboratory, CH-8803. Ruschlikon, Switzerland, August 26, 2004.
60. Шеннон К. *Работы по теории информации и кибернетики.* / Пер. с англ. — М.: Иностр. литература, 1963. — 829 с.
61. P. Moulin, J.A. O'Sullivan, *Information-Theoretic Analysis of Information Hiding*. IEEE Transactions on Information Theory, vol.49, № 3, pp. 563-593, March 2003.
62. J.K. Su, J.J. Eggers, B. Girod, *Analysis of Digital Watermarks Subjected to Optimum Linear Filtering and Additive Noise* // Signal Processing. Special Issue on Information Theoretic Issues in Digital Watermarking, vol.81. № 6, pp. 1141-1175, 2001.
63. F. Petitcolas, R.J. Anderson, M.G. Kuhn, *Information Hiding — A Survey* // Proceedings IEEE, Special Issue on Identification and Protection of Multimedia Information, vol.87, № 7. pp. 1069-1078, 1999.
64. F. Hartung, M. Kutter, *Multimedia Watermarking Techniques* // Proceedings IEEE, Special Issue on Identification and Protection of Multimedia Information, vol.87, № 7, pp. 1079-1107, 1999.
65. Б. Скляр, *Цифровая связь: Теоретические основы и практическое применение*. Изд. 2-е, исправл. — М.: Вильямс, 2003. — 1104 с.
66. M.D. Swanson, M. Kobayahi, A.H. Tewfik, *Multimedia Data-Embedding and Watermarking Strategies*. // Proceeding of IEEE, vol. 86, № 6, pp. 1064-1087, 1998.
67. T. Basar and G.J. Olsder, *Dynamic Noncooperative Game Theory* // SIAM Classics in Applied Mathematics. Philadelphia, PA: SIAM, 1999.
68. T.M. Cover and J.A. Thomas, *Elements of Information Theory*. New York: Wiley, 1991.
69. S.I. Gel'fand and M.S. Pinsker, *Coding For Channel With Random Parameters*, Probl. Contr. Inform. Theory, vol.9, №1, pp. 19–31, 1980.
70. Игнатов В.А. *Теория информации и передачи сигналов*. — М.: Радио и связь, 1991. — 280 с.
71. A.D. Wyner, *The Wire-tap Channel*. // Bell System Tech. J. 1975. Vol. 54. № 8. pp. 1355-1387.
72. A.D. Wyner and J. Ziv, *The Rate-Distortion Function For Source Coding With Side Information At The Decoder*, IEEE Trans. Inform. Theory, vol. IT–22, pp. 1–10, Jan. 1976.
73. Яковлев В.А. *Защита информации на основе кодового зашумления. Ч.1. Теория кодового зашумления*. / Под ред. В.И. Коржика. — С.Пб.: ВАС, 1993. — 245 с.
74. H. Hastur, *MandelSteg*, ftp://idea.sec.dsi.unimi.it/pub/security/crypt/code/.
75. Шиффман Х. *Ощущение и восприятие*. Изд. 5-е. — СПб.: Питер, 2003.
76. Максименко С.Д., Соловієнко В.О. *Загальна психологія:* Навч. посібник. — К.: МАУП, 2000. — 256 с.
77. A. Watson, *The Cortex Transform: Rapid Computation of Simulated Neural Images* // Computer Vision, Graphics, and Image Processing. 1987. Vol. 39. № 3. PP. 311-327.
78. S. Moller, A. Pfitzmann, I. Stirand, *Computer Based Steganography: How It Works And Why Therefore Any Restriction On Cryptography Are Nonsense, At Best*. // Information Hiding: First

International Workshop "InfoHiding'96", Springer as Lecture Notes in Computing Science, vol.1174, 1996. — pp.7-21.
79. T. Aura, *Practical Invisibility In Digital Communication*. // Information Hiding: First International Workshop "InfoHiding'96", Springer as Lecture Notes in Computing Science, vol.1174, 1996. — pp.265-278.
80. A. Westfeld, A. Pfitzmann, *Attacks on Steganographic Systems. Breaking the Steganographic Utilities EzStego, Jsteg, Steganos, and S-Tools – and Some Lessons Learned* // Proceeding of the Workshop on Information Hiding. 1999. — 16 p.
81. J. Fridrich, *A New Steganographic Method For Palette-Based Image*. // Proceedings of the ISBT PISP conference, Savannah, Georgia, Apr.1998, pp.285-289.
82. K. Matsui, K. Tanaka, *Video-steganography: How To secret Embed A Signature In A Picture*. // IMA intellectual properly project proceeding, vol.1, №1, 1994, pp.187-205.
83. M. Kutter, F. Jordan, F. Bossen, *Digital Signature Of Color Images Using Amplitude Modulation* // Proc. of the SPIE Storage and Retrieval for Image and Video Databases V. 1997. Vol. 3022. Pp. 518-526.
84. J. Hernandez, F. Perez-Gonzalez, J. Rodriguez, G. Nieto, *Performance Analysis of a 2-D Multipulse Amplitude Modulation Scheme for Data Hiding and Watermarking of Still Images* // IEEE Journal on Selected Areas in Communications. 1998. Vol. 16, № 5. Pp. 510-525.
85. A.N. Akansu, R.A. Haddad, *Multiresolution Signal Decomposition: Transforms, Subbands and Wavelets*, Academic Press Inc., New York, 1992.
86. J. Zhao, E. Koch, *Embedding Robust Labels into Images for Copyright Protection*. // Proceeding of the Int. Congress on Intellectual Property Rights for Specialized Information, Knowledge and New Techniques, Munich-Vienna, Verlag, Aug.1995, pp.242-251.
87. E. Koch, J. Zhao, *Towards Robust and Hidden Image Copyright Labeling*. // IEEE Workshop on Nonlinear Signal and Image Processing, Greece, June 20-22, 1995. Pp.123-132.
88. J. Smith, B. Comiskey, *Modulation and Information Hiding in Image*. // Information Hiding: First Int. Workshop "InfoHiding'96", Springer as Lecture Notes in Computing Science, vol.1174, 1996. — pp.207-227.
89. I. Pitas, *A Method for Signature Casting on Digital Images*. // Int. Conference on Image Processing, vol.3, IEEE Press, Sept. 1996, pp.215-218.
90. M.T. Sandford, T.G. Handel, J.M. Ettinger, *Data Embedding Method*. // Proceeding of the SPIE 2615, Integration issues in large commercial media delivery systems, 1996, pp. 226-259.
91. Коростиль Ю.М., Шелест М.Е. *Принципы построения стеганографических систем со структурной технологией*. // Праці VII міжнародної конференції з автоматичного управління "Автоматика-2000", Львів, 11-15 вересня 2000 р., секція 7, ч.1. — Львів: ДНДІІІ. — С.273-286.
92. A.V. Oppenheim and R.W. Shaffer, *Discrete-Time Processing*, Prentice-Hall, Inc., Englewood Cliffs, NJ. 1989.
93. Л. Рабинер, Б. Гоулд, *Теория и применение цифровой обработки сигналов*. // Пер. с англ.; Под ред. Ю.И. Александрова. — М.: Мир, 1978. — 848 с.
94. Конахович Г.Ф., Пузиренко О.Ю. *Використання пакету MathCAD v.12 для стеганографічного захисту секретних повідомлень у графічних файлах* // Захист інформації: Збірник наукових праць. — НАУ, 2005.
95. Кошкин А., Кошкина Н. *Стеганография – особенности использования программ на основе метода наименьшего значащего бита*. — www.delphikingdom.ru, 2004.
96. Конахович Г.Ф., Пузиренко О.Ю. *Використання пакету MathCAD v.12 для стеганографічного захисту секретних повідомлень в аудіофайлах* // Захист інформації: Збірник наукових праць. — НАУ, 2005.
97. *Малая математическая энциклопедия*. // Э. Фрид, И. Пастор, И. Рейман, П. Ревес, И. Ружа. — Будапешт: Изд-во Академии наук Венгрии, 1976. — 694 с.
98. M. Luby, C. Rackoff, *How to Construct Pseudorandom Permutations from Pseudorandom Functions*. // SIAM Journal on Computing, 17(2): 373-386, April 1998.
99. Paul C. Kocher, IEEE Personal Communication, Oct. 1995.
100. V. Darmstaedter, J.-F. Delaigle, J.J. Quisquater, B. Macq, *Low Cost Spatial Watermarking* // Computers and Graphics. 1998. Vol. 5. P. 417-423.

101. Фильчаков П.Ф. *Справочник по высшей математике*. — К.: Наукова думка, 1974. — 744 с.
102. G. Langelaar, R. Lagendijk, J. Biemond, *Robust Labeling Methods for Copy Protection of Images* // Proc. of the SPIE Storage and Retrieval for Image and Video Databases V. 1997. Vol. 3022.
103. D. Benham, N. Memon, B.-L. Yeo, M. Yeung, *Fast Watermarking of DCT-based Compressed Images* // Proc. of the International Conference on Image Science, Systems and Technology. Las Vegas, Nevada, June 30 – July 3, 1997, vol. 1, pp. 243-252.
104. C.-T. Hsu, J.-L. Wu, *DCT-based Watermarking for Video* // IEEE Transactions on Consumer Electronics, Vol. 44, No. 1, Feb 1998, pp. 206-216.
105. Г. Корн, Т. Корн. Справочник по математике для научных работников и инженеров. Определения, теоремы, формулы. // Пер. с англ. под ред. И.Г. Арамановича. Изд. 2-е. — М.: Наука, 1970. — 720 с.
106. J. Fridrich, *Combining Low-Frequency and Spread Spectrum Watermarking* // Proc. of the SPIE Conference on Mathematics of Data / Image Coding, Compression and Encryption. 1998. Vol. 3456. P. 2-12.
107. Н. Хастингс, Дж. Пикок, *Справочник по статистическим распределениям* // Пер. с англ. А.К. Звонкина. — М.: Статистика, 1980. — 95 с.
108. D. Gruhl, A. Lu, W. Bender, *Echo Hiding*. Information Hiding Workshop, Cambridge, UK, (1996).

Додаток А

Вбудовані оператори MathCAD

У приведеному нижче переліку операторів використовуються наступні умовні позначення:

A і **B** – масиви (вектори або матриці);
u і **v** – вектори з дійсними або комплексними елементами;
M – квадратна матриця;
z і **w** – дійсні або комплексні числа;
x і **y** – дійсні числа;
m і **n** – цілі числа;
i – діапазон змінних (дискретний аргумент);
t – будь-яке ім'я змінної;
f – будь-яка функція;
X і **Y** – змінні або вирази будь-якого типу.

На їх місце можуть підставлятися об'єкти відповідного типу (змінні, вектори, матриці і т.п.) з будь-якими іншими іменами і вирази з результатом їх обчислення відповідного типу.

Оператор	*Позначення*	*Введення*	*Призначення оператора (операції)*
1	*2*	*3*	*4*
Присвоєння значення напівглобальній змінній (елементу масиву, стовпцю матриці), визначення функції користувача	■ := ■ **z** := **x** + **y** $\mathbf{B}_{\mathbf{n},\mathbf{m}}$:= 8 $\mathbf{A}^{<\mathbf{n}>}$:= **v** **f**(**x**, [**y**,...]) := **x**+7	:	Присвоює значення змінній (елементу масиву), визначає функцію користувача, видиму правіше і нижче даного оператора
Присвоєння значення глобальній змінній (елементу масиву, стовпцю матриці), визначення функції користувача	■ ≡ ■ **z** ≡ **x** + **y** $\mathbf{B}_{\mathbf{n},\mathbf{m}}$ ≡ 8 $\mathbf{A}^{<\mathbf{n}>}$ ≡ **v** **f**(**x**, [**y**,...]) ≡ **x**+7	~	Присвоює значення змінній (елементу масиву), визначає функцію користувача, видиму в усьому документі MathCAD
Присвоєння значення локальній змінній (елементу масиву, стовпцю матриці)	■ ← ■ **z** ← **x** + **y** $\mathbf{B}_{\mathbf{n},\mathbf{m}}$ ← 8 $\mathbf{A}^{<\mathbf{n}>}$ ←**v**	{	Присвоює значення локальній змінній (елементу масиву) в межах програмного модуля
Обчислення числове значення	■ = ■·[■] **z** = 1980 $\mathbf{B}_{\mathbf{n},\mathbf{m}}$ = 8 $\mathbf{v}_{\mathbf{n}}$ = 1·s **f**(**x**) = 74	=	Обчислює і виводить на екран у 2-й операнд числове значення змінної, виразу, функції, записаної у 1-ому операнді. Можна задавати розмірність (3-й операнд)
Круглі дужки	(**X**)	'	Зміна пріоритету виконання операцій, група операторів
Нижній індекс	$\mathbf{A}_{\mathbf{n}}$	[	Задання індексованої змінної
Верхній індекс	$\mathbf{A}^{<\mathbf{n}>}$	[Ctrl]6	Обрання **n**- го стовпця з масиву **A**
Векторизація	$\overrightarrow{\mathbf{f}(\mathbf{A})}$	[Ctrl]-	Виконання заданої операції **f** для всіх елементів масиву **A**
Факторіал	**n**!	!	Обчислення факторіалу для цілого додатного числа **n**
Спряжене комплексне число	$\bar{\mathbf{z}}$	"	Обчислення спряженого комплексного числа (інвертований сигнал уявної частини **z**)
Транспонування	$\mathbf{A}^{\mathbf{T}}$	[Ctrl]1	Транспонування масиву **A**
Піднесення до степеня	$\mathbf{z}^{\mathbf{w}}$	^	Піднесення **z** до степеня **w**
Степінь матриці, інверсна матриця	$\mathbf{M}^{\mathbf{n}}$	^	Піднесення квадратної матриці **M** до степеня **n** (при **n** = –1 інверсія матриці)
Заперечення	–**X**	–	Добуток **X** на –1

1	*2*	*3*	*4*
Сума елементів вектора	$\Sigma \mathbf{v}$	[Ctrl]4	Обчислення суми елементів вектора **v** (повертається скалярне значення)
Квадратний корінь	$\sqrt{\mathbf{w}}$	\	Обчислення квадратного кореня з **w**
Корінь **n**- го степеня	$\sqrt[n]{\mathbf{w}}$	[Ctrl]\	Обчислення кореня степені **n** з **w**
Модуль комплексного числа	$\lvert \mathbf{z} \rvert$	|	Обчислення модуля $\sqrt{\mathbf{Re}(\mathbf{z})^2 + \mathbf{Im}(\mathbf{z})^2}$ комплексного числа **z**
Абсолютна величина вектора (евклідова норма або розмірність вектора)	$\lvert \mathbf{v} \rvert$	|	Обчислення $\sqrt{\mathbf{v} \cdot \mathbf{v}}$, якщо всі елементи **v** є дійсними, і $\sqrt{\mathbf{v} \cdot \overline{\mathbf{v}}}$, якщо елементи **v** є комплексними
Детермінант матриці **M**	$\lvert \mathbf{M} \rvert$	|	Повертає визначник (детермінант) квадратної матриці **M**
Ділення	$\frac{\mathbf{X}}{\mathbf{z}}$ або $\mathbf{X} \div \mathbf{z}$	/	Ділення виразу **X** на скаляр **z**, який не дорівнює 0 (якщо **X** є масивом, то на **z** ділиться кожен елемент масиву)
Множення	$\mathbf{X} \cdot \mathbf{Y}$	*	Обчислення добутку **X** на **Y**, якщо **X** і **Y** є скалярами. Множення кожного елемента **Y** на **X**, якщо **Y** є масивом, а **X** – скаляром. Обчислення скалярного добутку, якщо **X** та **Y** – вектори однакового розміру. Множення матриць, якщо **X** та **Y** є подібними матрицями
Крос-добуток	$\mathbf{u} \cdot \mathbf{v}$	[Ctrl]8	Обчислення векторного добутку векторів **u** і **v**
Підсумовування для скінченого ряду	$\sum_{\mathbf{i}=\mathbf{m}}^{\mathbf{n}} \mathbf{X}$	[Ctrl] [Shift] 4	Обчислення суми членів **X** для **i** = **m**, **m**+1,..., **n** , причому **X** може бути будь-яким виразом
Добуток для скінченого ряду	$\prod_{\mathbf{i}=\mathbf{m}}^{\mathbf{n}} \mathbf{X}$	[Ctrl] [Shift] 3	Перемножування елементів **X** для **i** = **m**, **m**+1,..., **n** , причому **X** може бути будь-яким виразом
Підсумовування для нескінченого ряду	$\sum_{\mathbf{i}} \mathbf{X}$	$	Обчислення суми нескінченої кількості членів **X**, причому **X** може бути будь-яким виразом
Добуток для нескінченого ряду	$\prod_{\mathbf{i}} \mathbf{X}$	#	Перемноження нескінченої кількості членів **X**, причому **X** може бути будь-яким виразом
Границя функції в заданій точці	$\lim_{\mathbf{x} \to \mathbf{a}} \mathbf{f}(\mathbf{x})$	[Ctrl]L	Обчислення границі функції **f(x)** при **x**, що прямує до **a** (лише у символьному вигляді)
Границя функції ліворуч від заданої точки	$\lim_{\mathbf{x} \to \mathbf{a}^-} \mathbf{f}(\mathbf{x})$	[Ctrl]B	Обчислення границі функції **f(x)** при **x**, що прямує до **a** зліва (лише у символьному вигляді)
Границя функції праворуч від заданої точки	$\lim_{\mathbf{x} \to \mathbf{a}^+} \mathbf{f}(\mathbf{x})$	[Ctrl]A	Обчислення границі функції **f(x)** при **x**, що прямує до **a** праворуч (лише у символьному вигляді)
Визначений інтеграл	$\int_{\mathbf{a}}^{\mathbf{b}} \mathbf{f}(\mathbf{t})\, d\mathbf{t}$	&	Обчислення визначеного інтеграла від підінтегральної функції **f(t)** з межами інтегрування **a** і **b**
Невизначений інтеграл	$\int \mathbf{f}(\mathbf{t})\, d\mathbf{t}$	[Ctrl]I	Обчислення у символьному вигляді невизначеного інтегралу від підінтегральної функції **f(t)**

1	*2*	*3*	*4*
Похідна заданої функції по змінній **t**	$\frac{d}{dt}\mathbf{f(t)}$	?	Обчислення першої похідної функції **f(t)** по змінній **t**
n-а похідні заданої функції по змінній **t**	$\frac{d^n}{dt^n}\mathbf{f(t)}$	[Ctrl]?	Обчислення **n**- ої похідної функції **f(t)** по змінній **t**
Додавання	**X+Y**	+	Скалярне додавання, якщо **X**, **Y** є скалярами. Додавання елементів, якщо **X** і **Y** – масиви однакового розміру. Якщо **X** є масивом, а **Y** – скаляром, складання кожного елементу **X** з **Y**
Віднімання	**X–Y**	−	Виконує віднімання скалярів, векторів або матриць **X** і **Y**
Перенесення на інший рядок	**X** ... **+Y**	[Ctrl] ↵	Перенесення частини виразу на наступний рядок. Лінія розриву має редакційне значення
Більше, чим	**x** > **y** "**T1**" > "**T2**"	>	Повертає 1, якщо **x** > **y**, інакше – 0. **x** і **y** повинні бути дійсними скалярними змінними. Для текстових змінних **T1** і **T2** – повертає 1, якщо ASCII-кодування змінної **T1** більше, чим у змінної **T2**
Менше, чим	**x** < **y** "**T1**" < "**T2**"	<	Повертає 1, якщо **x** < **y**, інакше – 0. **x** і **y** повинні бути дійсними скалярними змінними. Для текстових змінних **T1** і **T2** – повертає 1, якщо ASCII-кодування змінної **T1** менше, чим у змінної **T2**
Більше, чим або дорівнює	**x** ≥ **y** "**T1**" ≥ "**T2**"	[Ctrl]0	Повертає 1, якщо **x** ≥ **y**, 0 – в іншому випадку. **x** і **y** повинні бути дійсними скалярними змінними. Для текстових змінних **T1** і **T2** – повертає 1, якщо ASCII-кодування **T1** ≥ **T2**
Менше, чим або дорівнює	**x** ≤ **y** "**T1**" ≤ "**T2**"	[Ctrl]9	Повертає 1, якщо **x** ≤ **y**, 0 – в іншому випадку. **x** і **y** повинні бути дійсними скалярними змінними. Для текстових змінних **T1** і **T2** – повертає 1, якщо ASCII-кодування **T1** ≤ **T2**
Дорівнює	**x = y** "**T1**" = "**T2**"	[Ctrl]=	Повертає 1, якщо **x = y**, 0 – в іншому випадку. **x** і **y** повинні бути дійсними скалярними змінними. Для текстових змінних **T1** і **T2** – повертає 1, якщо ASCII-кодування **T1 = T2**
Не дорівнює	**x** ≠ **y** "**T1**" ≠ "**T2**"	[Ctrl]3	Повертає 1, якщо **x** ≠ **y**, 0 – в іншому випадку. **x** і **y** повинні бути дійсними скалярними змінними. Для текстових змінних **T1** і **T2** – повертає 1, якщо ASCII-кодування **T1** ≠ **T2**

Додаток В

Основні вбудовані функції та директиви MathCAD

У приведених нижче функціях MathCAD використовуються такі умовні позначення:

A і **B**	– масиви (вектори або матриці);
M і **N**	– квадратні матриці;
X і **Y**	– змінні або вирази будь-якого типу;
F	– вектор-функція;
f	– функція-скаляр;
S	– рядкова змінна або константа
x і **y**	– дійсні числа;
z і **w**	– дійсні або комплексні числа;
m, **n**, **i**, **j**, **k**	– цілі числа;
u і **v**[...]	– вектори;
file	– ім'я файлу або файлова змінна, приєднана до імені файлу.

Всі кути в тригонометричних функціях виражені у радіанах. Багатозначні функції і функції з комплексним аргументом завжди повертають головне значення. Імена приведених функцій нечутливі до шрифту, але чутливі до регістру – їх слід вводити в точності, як вони приведені. Всі функції повертають вказане для них значення.

Функції для роботи з комплексними числами

arg(**z**) — повертає аргумент комплексного числа **z** між $-\pi$ та π

csgn(**z**) — повертає комплексний знак числа: 0, якщо **z** = 0; 1, якщо **Re**(**z**) > 0 або (**Re**(**z**) = 0 та **Im**(**z**) > 0); –1 в іншому випадку

Im(**z**) — уявна частина комплексного аргументу **z**

Re(**z**) — повертає дійсну частина комплексного аргументу **z**

signum(**z**) — повертає комплексний знак числа: 1, якщо **z** = 0; **z**/|**z**| в іншому випадку

Кусочно-неперервні функції

if(**cond**, **f1**, **f2**) — умовний вираз, що повертає вираз **f1**, якщо логічна умова **cond** є правдивою, і вираз **f2** в інших випадках

δ(**m**, **n**) — функція Кронекера: 1, якщо **m** = **n**; 0, якщо **m** ≠ **n**

Φ(**x**) — одинична ступінчата функція (функція Хевісайда): 1 при **x** ≥ 0; 0 при **x** < 0

sign(**x**) — повертає знак числа **x**: 0, якщо **x** = 0; 1, якщо **x** > 0; –1, якщо **x** < 0

Статистичні функції і функції аналізу даних

corr(**A**, **B**) — коефіцієнт кореляції елементів масивів **A** і **B** по Пірсону. Обчислюється як $\frac{\mathbf{cvar}(\mathbf{A},\mathbf{B})}{\mathbf{stdev}(\mathbf{A})\cdot\mathbf{stdev}(\mathbf{B})}$

cvar(**A**, **B**) — коваріація елементів масивів **A** і **B**, що мають розмірність **m**×**n**,

$$\frac{1}{\mathbf{m}\cdot\mathbf{n}}\cdot\sum_{\mathbf{i}=0}^{\mathbf{m}-1}\sum_{\mathbf{j}=0}^{\mathbf{n}-1}\left[\mathbf{A}_{\mathbf{i},\mathbf{j}}-\mathbf{mean}(\mathbf{A})\right]\cdot\overline{\left[\mathbf{B}_{\mathbf{i},\mathbf{j}}-\mathbf{mean}(\mathbf{B})\right]}$$

gmean(**A**,**B**,**C**,...) — повертає середнє геометричне для аргументів **A**, **B**, **C**, ... розмірністю **m**×**n** за виразом $\sqrt[\mathbf{m}\cdot\mathbf{n}]{\prod_{\mathbf{i}=0}^{\mathbf{m}-1}\prod_{\mathbf{j}=0}^{\mathbf{n}-1}\frac{1}{\mathbf{M}_{\mathbf{i},\mathbf{j}}}}$, де **M** – масив, утворений аргументами **A**, **B**, **C**, ...

hist(**int**, **d**) — повертає вектор значення частот, з якими величини, що містяться у векторі **d**, потрапляють до інтервалів, представлених межами, заданими у векторі **int**

histogram(**n**, **d**) — повертає матрицю, що має дві колонки: перша містить середини **n** підінтервалів діапазону рівної довжини, друга є ідентичною обчисленню функції **hist**(**int**, **d**)

hmean(**A**,**B**,**C**,...) — повертає середнє гармонічне для аргументів **A**, **B**, **C**, ... за формулою

$$\left(\frac{1}{\mathbf{m}\cdot\mathbf{n}}\cdot\sum_{\mathbf{i}=0}^{\mathbf{m}-1}\sum_{\mathbf{j}=0}^{\mathbf{n}-1}\frac{1}{\mathbf{M}_{\mathbf{i},\mathbf{j}}}\right)^{-1}$$

kurt(**A**, **B**, **C**, ...)— повертає ексцес аргументів **A**, **B**, **C**, ... за формулою

$$\left[\frac{\mathbf{m}\cdot\mathbf{n}\cdot(\mathbf{m}\cdot\mathbf{n}+1)}{(\mathbf{m}\cdot\mathbf{n}-1)\cdot(\mathbf{m}\cdot\mathbf{n}-2)\cdot(\mathbf{m}\cdot\mathbf{n}-3)}\cdot\sum_{\mathbf{i}=0}^{\mathbf{m}-1}\sum_{\mathbf{j}=0}^{\mathbf{n}-1}\left(\frac{\mathbf{M_{i,j}}-\mathbf{mean}(\mathbf{M})}{\mathbf{Stdev}(\mathbf{M})}\right)^4\right]-\frac{3\cdot(\mathbf{m}\cdot\mathbf{n}-1)^2}{(\mathbf{m}\cdot\mathbf{n}-2)\cdot(\mathbf{m}\cdot\mathbf{n}-3)},$$

причому масив **M**, утворений на основі аргументів **A**, **B**, **C**, ..., повинен мати не менше 4-х елементів, а стандартне відхилення цих елементів $\sigma \neq 0$

mean(**A**, **B**, **C**, ...) — повертає середнє арифметичне для аргументів **A**, **B**, **C**, ...: $\frac{1}{\mathbf{m}\cdot\mathbf{n}}\cdot\sum_{\mathbf{i}=0}^{\mathbf{m}-1}\sum_{\mathbf{j}=0}^{\mathbf{n}-1}\mathbf{M_{i,j}}$

median(**A**,**B**,**C**,...) — повертає медіану для аргументів **A**, **B**, **C**, ...

mode(**A**, **B**, **C**, ...) — повертає значення елементу аргументів **A**,**B**,**C**,..., яке зустрічається найбільш часто

skew(**A**, **B**, **C**, ...) — повертає коефіцієнт асиметрії значень аргументів **A**, **B**, **C**, ..., обчислений як $\frac{\mathbf{m}\cdot\mathbf{n}}{(\mathbf{m}\cdot\mathbf{n}-1)\cdot(\mathbf{m}\cdot\mathbf{n}-2)}\cdot\sum_{\mathbf{i}=0}^{\mathbf{m}-1}\sum_{\mathbf{j}=0}^{\mathbf{n}-1}\left(\frac{\mathbf{M_{i,j}}-\mathbf{mean}(\mathbf{M})}{\mathbf{Stdev}(\mathbf{M})}\right)^3$

stderr(**u**, **v**) — повертає середньоквадратичну помилку, що відповідає простій лінійній регресії для точок, описаних векторами **u** і **v** (для кожного вектора кількість елементів $\mathbf{n} \geq 3$). При цьому визначається, наскільки близько результати обробки даних розміщені до лінії регресії, $\sqrt{\frac{1}{\mathbf{n}-2}\cdot\sum_{\mathbf{i}}[\mathbf{v_i}-\mathbf{intercept}(\mathbf{u},\mathbf{v})+\mathbf{slope}(\mathbf{u},\mathbf{v})\cdot\mathbf{u_i}]^2}$

Stdev(**A**, **B**, **C**, ...) — повертає значення вибіркового стандартного відхилення значень аргументів, як корінь квадратний з дисперсії **Var**(**A**, **B**, **C**, ...)

stdev(**A**, **B**, **C**, ...) — повертає значення стандартного відхилення генеральної сукупності значень аргументів, як корінь квадратний з дисперсії **var**(**A**, **B**, **C**, ...)

Var(**A**, **B**, **C**, ...) — повертає значення вибіркової дисперсії значень аргументів, обчислене як $\sum_{\mathbf{i}=0}^{\mathbf{m}-1}\sum_{\mathbf{j}=0}^{\mathbf{n}-1}(\mathbf{M_{i,j}}-\mathbf{mean}(\mathbf{M}))^2 \Big/ (\mathbf{m}\cdot\mathbf{n}-1)$. Ділення квадратів відхилення на "об'єм вибірки мінус один" дозволяє одержати кращу оцінку істинної дисперсії генеральної сукупності

var(**A**, **B**, **C**, ...) — повертає значення дисперсії генеральної сукупності, обчислене за формулою $\sum_{\mathbf{i}=0}^{\mathbf{m}-1}\sum_{\mathbf{j}=0}^{\mathbf{n}-1}(\mathbf{M_{i,j}}-\mathbf{mean}(\mathbf{M}))^2 \Big/ \mathbf{m}\cdot\mathbf{n}$. Таким чином, $\mathbf{var}\cdot\frac{\mathbf{m}\cdot\mathbf{n}}{\mathbf{m}\cdot\mathbf{n}-1}=\mathbf{Var}$

Функції статистичних розподілів і генераторів випадкових чисел

Позначення наступних функцій здійснені наступним чином: **d...** – функція щільності розподілу імовірності (імовірність того, що випадкова величина прийме значення **x**); **p...** – функція кумулятивного розподілу імовірностей (імовірність того, що випадкова величина прийме значення $\mathbf{X} \leq \mathbf{x}$); **q...** – функція оберненого кумулятивного розподілу імовірностей – квантиль (таке значення **x**, при якому імовірність не перевищує задане значення **p**); **r...** – функція, що повертає вектор з **m** випадкових чисел, які мають розподіл "**...**".

β-розподіл, $\frac{\Gamma(\mathbf{s1}+\mathbf{s2})}{\Gamma(\mathbf{s1})\cdot\Gamma(\mathbf{s2})}\cdot\mathbf{x}^{\mathbf{s1}-1}\cdot(1-\mathbf{x})^{\mathbf{s2}-1}$, при параметрах форми $(\mathbf{s1},\mathbf{s2})>0$ та $0 < \mathbf{x} < 1$: **dbeta**(**x**, **s1**, **s2**); **pbeta**(**x**, **s1**, **s2**); **qbeta**(**p**, **s1**, **s2**); **rbeta**(**m**, **s1**, **s2**).

Біноміальний розподіл, $\frac{\mathbf{n!}}{\mathbf{k!}\cdot(\mathbf{n}-\mathbf{k})\mathbf{!}}\cdot\mathbf{p}^{\mathbf{k}}\cdot(1-\mathbf{p})^{\mathbf{n}-\mathbf{k}}$, де **n** – кількість незалежних випробувань з імовірністю успіху **p**, $0 \leq \mathbf{p} \leq 1$; **q** – імовірність успіху при однократному випробуванні; **k** – кількість успіхів, $0 \leq \mathbf{k} \leq \mathbf{n}$: **dbinom**(**k**, **n**, **p**); **pbinom**(**k**, **n**, **p**); **qbinom**(**p**, **n**, **q**); **rbinom**(**m**, **n**, **p**).

Розподіл Коші, $\left[\pi\cdot\mathbf{s}\cdot\left[1+\left(\frac{\mathbf{x}-\mathbf{a}}{\mathbf{s}}\right)^2\right]\right]^{-1}$, де **a** – параметр розташування; $\mathbf{s} > 0$ – параметр масштабу: **dcauchy**(**x**, **a**, **s**); **pcauchy**(**x**, **a**, **s**); **qcauchy**(**p**, **a**, **s**); **rcauchy**(**m**, **a**, **s**).

Розподіл χ^2, $\frac{\exp(-x/2)}{2\cdot\Gamma(d/2)}\cdot\left(\frac{x}{2}\right)^{d/2-1}$, де **d** – параметр форми (кількість ступенів свободи), **d** > 0; **x** > 0: **dchisq**(**x**, **d**); **pchisq**(**x**, **d**); **qchisq**(**p**, **d**); **rchisq**(**m**, **d**).

Експонентний (показниковий) розподіл, $r\cdot\exp(-r\cdot x)$, де **r** – параметр масштабу, **r** > 0; **x** > 0: **dexp**(**x**, **r**); **pexp**(**x**, **r**); **qexp**(**p**, **r**); **rexp**(**m**, **r**).

F-розподіл Фішера, $\frac{\sqrt{d1^{d1}\cdot d2^{d2}}\cdot\Gamma\left(\frac{d1+d2}{2}\right)}{\Gamma\left(\frac{d1}{2}\right)\cdot\Gamma\left(\frac{d2}{2}\right)}\cdot\sqrt{\frac{x^{d1-2}}{(d1+d2\cdot x)^{d1+d2}}}$, де **d1**,**d2** > 0 – кількості ступенів свободи; **x** > 0: **dF**(**x**, **d1**, **d2**); **pF**(**x**, **d1**, **d2**); **qF**(**p**, **d1**, **d2**); **rF**(**m**, **d1**, **d2**).

γ-розподіл, $x^{s-1}\cdot\exp(-x)/\Gamma(s)$, де **s** > 0 – параметр форми; **x** ≥ 0: **dgamma**(**x**, **s**); **pgamma**(**x**, **s**); **qgamma**(**p**, **s**); **rgamma**(**m**, **s**).

Геометричний розподіл, $p\cdot(1-p)^n$, де 0 < **p** < 1 – імовірність успіху; **n** – кількість випробувань, **n** ≥ 0: **dgeom**(**n**, **p**); **pgeom**(**n**, **p**); **qgeom**(**p**, **q**); **rgeom**(**m**, **p**).

Гіпергеометричний розподіл, $C_a^m\cdot\frac{C_b^{n-m}}{C_{a+b}^n}=\mathbf{combin}(a,m)\cdot\frac{\mathbf{combin}(b,n-m)}{\mathbf{combin}(a+b,n)}$, де **a** + **b** – популяція елементів, з яких **a** мають деяку властивість (видобування такого елементу є успіхом); **n** – обсяг вибірки з популяції без повернення; **m** – кількість успіхів у виборці. Необхідна умова $\max\{0,n-b\}\le m\le\min\{n,a\}$: **dhypergeom**(**m**, **a**, **b**, **n**) і **phypergeom**(**m**, **a**, **b**, **n**), для яких 0 ≤ **m** ≤ **a**, 0 ≤ (**n**–**m**) ≤ **b**, 0 ≤ **n** ≤ (**a**+**b**); **qhypergeom**(**p**, **a**, **b**, **n**) і **rhypergeom**(**m**, **a**, **b**, **n**), для яких 0 ≤ **p** < 1, **m** > 0, (**a**, **b**) ≥ 0, 0 ≤ **n** ≤ (**a**+**b**).

Логнормальний розподіл, $\frac{1}{\sqrt{2\cdot\pi}\cdot\sigma\cdot x}\cdot\exp\left[-\frac{1}{2\cdot\sigma^2}\cdot(\ln(x)-\mu)^2\right]$, де μ – натуральний логарифм від середнього значення; σ > 0 – параметр форми (натуральний логарифм середньоквадратичного відхилення); **x** > 0: **dlnorm**(**x**, μ, σ); **plnorm**(**x**, μ, σ); **qlnorm**(**p**, μ, σ); **rlnorm**(**m**, μ, σ).

Логістичний розподіл, $\exp\left(\frac{a-x}{s}\right)\Big/\left[s\cdot\left(1+\exp\left(\frac{a-x}{s}\right)\right)^2\right]$, де **a** – параметр розташування; **s** > 0 – параметр масштабу: **dlogis**(**x**, **a**, **s**); **plogis**(**x**, **a**, **s**); **qlogis**(**p**, **a**, **s**); **rlogis**(**m**, **a**, **s**).

Від'ємний біноміальний розподіл, $C_{n+k-1}^k\cdot p^n\cdot(1-p)^k=\mathbf{combin}(n+k-1,k)\cdot p^n\cdot(1-p)^k$, де 0 < **p** ≤ 1 – імовірність успіху; **x** > 0 – кількість успіхів; **k** ≥ 0 – кількість невдач: **dnbinom**(**k**, **n**, **p**); **pnbinom**(**k**, **n**, **p**); **qnbinom**(**p**, **n**, **q**); **rnbinom**(**m**, **n**, **p**).

Нормальний (гаусівський) розподіл, $\frac{1}{\sqrt{2\cdot\pi}\cdot\sigma}\cdot\exp\left[-\frac{1}{2\cdot\sigma^2}\cdot(x-\mu)^2\right]$, де μ – параметр розташування (математичне очікування); σ > 0 – параметр масштабу (середньоквадратичне відхилення): **dnorm**(**x**, μ, σ); **pnorm**(**x**, μ, σ); **qnorm**(**p**, μ, σ); **rnorm**(**m**, μ, σ); **cnorm**(**x**) – те саме, що й функція **pnorm** при μ = 0 і σ = 1 (стандартизований нормальний розподіл).

Розподіл Пуассона, $\lambda^k\cdot\exp(-\lambda)/k!$, де λ > 0 – параметр розподілу; **k** ≥ 0: **dpois**(**k**, λ); **ppois**(**k**, λ); **qpois**(**k**, λ); **rpois**(**m**, λ).

t-розподіл Стьюдента, $\frac{\Gamma\left(\frac{d+1}{2}\right)}{\Gamma\left(\frac{d}{2}\right)\cdot\sqrt{\pi\cdot d}}\cdot\left[\sqrt{\left(1+\frac{x^2}{d}\right)^{d+1}}\right]^{-1}$, де **d** > 0 – кількість ступенів свободи; **x** – дійсне: **dt**(**x**, **d**); **pt**(**x**, **d**); **qt**(**p**, **d**); **rt**(**m**, **d**).

Рівномірний (прямокутний) розподіл, $(b-a)^{-1}$, де **a** і **b** – відповідно, нижня і верхня границі області значень, причому **a** < **b**, **a** ≤ **x** ≤ **b**: **dunif**(**x**, **a**, **b**); **punif**(**x**, **a**, **b**); **qunif**(**x**, **a**, **b**); **runif**(**m**, **a**, **b**); **rnd**(**x**) – повертає рівномірно розподілене випадкове число на інтервалі між 0 і **x**.

Розподіл Вейбулла, $s\cdot x^{s-1}\cdot\exp\left(-x^s\right)$, де **s** > 0 – параметр форми; **x** > 0: **dweibull**(**x**, **s**); **pweibull**(**x**, **s**); **qweibull**(**p**, **s**); **rweibull**(**m**, **s**).

Seed(**x**) — скидає значення початкового числа при генерації ПВЧ в **x** і повертає попереднє значення. Причому $1\le x\le 2147483647$, ціле

Функції комбінаторного аналізу і теорії чисел

combin(**n**, **k**) — повертає кількість підмножин (комбінацій), кожна розмірністю **k**, що може бути утворена з **n** об'єктів – еквівалент $\frac{\mathbf{n!}}{\mathbf{k!}\cdot(\mathbf{n}-\mathbf{k})!}$, де $\mathbf{n} \geq \mathbf{k}$; $\mathbf{n}, \mathbf{k} \geq 0$

gcd(**A**, **B**, **C**, ...) — повертає найбільший спільний дільник – найбільше число, на яке без залишку діляться кожен з аргументів **A**, **B**, **C**, ...

lcm(**A**, **B**, **C**, ...) — повертає найменше спільне кратне – найменше додатне ціле, яке без залишку ділиться на кожен з аргументів **A**, **B**, **C**, ...

mod(**x**, **y**) — залишок від ділення **x** на **y** (аргументи повинні бути дійсними, $\mathbf{y} \neq 0$; результат має той самий знак, що й **x**)

permut(**n**, **k**) — повертає кількість можливих способів обрання (перестановки) **k** різних об'єктів з **n** існуючих – еквівалент $\mathbf{n!}/(\mathbf{n}-\mathbf{k})\mathbf{!}$, де $\mathbf{n} \geq \mathbf{k}$; $\mathbf{n}, \mathbf{k} \geq 0$

Функції інтерполяції та екстраполяції

cspline(**vx**, **vy**) — повертає вектор **vs** других похідних для даних, представлених векторами **vx** і **vy**. Побудована за цим вектором сплайнова крива є кубічною у кінцевих точках

interp(**vs**,**vx**,**vy**,**x**) — повертає значення сплайн-інтерпольованої функції для векторів **vx** (впорядкованого по зростанню) і **vy** опорних точок і аргументу **x** на основі вектора **vs**. Вектори **vx** і **vy** мають однакову розмірність.

linterp(**vx**, **vy**, **x**) — повертає значення лінійно-інтерпольованої функції для заданих векторів **vx** (впорядкованого по зростанню) і **vy** опорних точок і заданого аргументу **x**. Вектори **vx** і **vy** мають однакову розмірність. При цьому точки на графіку, одержані за даними векторів, з'єднуються відрізками прямих

lspline(**vx**, **vy**) — те саме, що й **cspline**, але сплайнова крива у кінцевих точках є лінійною

predict(**v**, **m**, **n**) — повертає вектор з **n** лінійно прогнозованих значень на основі **m** послідовних елементів з вектору даних **v**

pspline(**vx**, **vy**) — те саме, що й **cspline**, але сплайнова крива у кінцевих точках є параболічною

Функції регресії

expfit(**vx**,**vy**,[**vg**]) — експонентна регресія даних, визначених векторами **vx** і **vy**. Вектор **vg**, якщо використовується, містить прогнозовані значення параметрів **a**, **b** і **c** у степеневому рівнянні $\mathbf{a}\cdot\mathbf{exp}(\mathbf{b}\cdot\mathbf{x})+\mathbf{c}$

genfit(**vx**,**vy**,**vg**,**F**) — повертає вектор **K** параметрів функції **F**, які дають мінімальну середньоквадратичну похибку опису функцією $\mathbf{F}(\mathbf{x}, \mathbf{K}_1, ..., \mathbf{K}_\mathbf{n})$ первинних даних **vx**, **vy**. Вектор **vg** повинен містити прогнозовані значення **n**-параметрів

intercept(**vx**, **vy**) — повертає значення вільного члену **a** лінійної регресії $\mathbf{a}+\mathbf{b}\cdot\mathbf{x}$

lgsfit(**vx**, **vy**, **vg**) — регресія логістичною функцією $\mathbf{a}/(1+\mathbf{b}\cdot\mathbf{exp}(-\mathbf{c}\cdot\mathbf{x}))$

line(**vx**, **vy**) — повертає вектор коефіцієнтів лінійної регресії функцією $\mathbf{a}+\mathbf{b}\cdot\mathbf{x}$

linfit(**vx**, **vy**, **F**) — повертає вектор коефіцієнтів лінійної регресії загального виду. Вектор **F** повинен містити функції $\mathbf{F}_1(\mathbf{x})$, ..., $\mathbf{F}_\mathbf{n}(\mathbf{x})$

lnfit(**vx**, **vy**) — регресія логарифмічною функцією $\mathbf{a}\cdot\mathbf{ln}(\mathbf{x})+\mathbf{b}$

loess(**Mx**, **My**, **sp**) — повертає вектор коефіцієнтів **vs** для регресії відрізками поліномів (використовується в парі з функцією **interp**). Параметр **sp** вказує розмір локальної області наближуваних даних

logfit(**vx**, **vy**, **vg**) — регресія логарифмічною функцією $\mathbf{a}\cdot\mathbf{ln}(\mathbf{x}+\mathbf{b})+\mathbf{c}$

medfit(**vx**, **vy**) — повертає вектор коефіцієнтів лінійної регресії функцією $\mathbf{a}+\mathbf{b}\cdot\mathbf{x}$, використовуючи медіанну регресію

pwrfit(**vx**, **vy**, **vg**) — регресія степеневою функцією $\mathbf{a}\cdot\mathbf{x}^\mathbf{b}+\mathbf{c}$

regress(**Mx**,**vy**,**n**) — повертає вектор коефіцієнтів **vs** для поліноміальної регресії при степені поліному **n** (використовується в парі з функцією **interp**)

sinfit(**vx**, **vy**, **vg**) — регресія синусоїдою $\mathbf{a}\cdot\mathbf{sin}(\mathbf{x}+\mathbf{b})+\mathbf{c}$

slope(**vx**, **vy**) — повертає значення кутового коефіцієнту **b** лінійної регресії $\mathbf{a}+\mathbf{b}\cdot\mathbf{x}$

Функції статистичного згладжування даних

ksmooth(**vx**,**vy**,**b**) — повертає **m**-мірний вектор згладжених елементів **vy**, обчислених на основі розподілу Гауса. **vx**, **vy** – **m**-мірні вектори дійних чисел. Параметр **b** – ширина вікна згладжування

medsmooth(**vy**,**n**) — повертає **m**-мірний вектор згладжених елементів **vy**, обчислених з використанням методу "ковзних медіан". Параметр **n** – ширина вікна згладжування

supsmooth(**vx**,**vy**) — повертає **m**-мірний вектор згладжених елементів **vy**, обчислених на основі адаптивної процедури лінійного згладжування методом найменших квадратів

expsmooth(**B**, α) — згладжування даних масиву **B** шляхом застосування експонентного згладжування з ваговим коефіцієнтом α

Функції дискретного перетворення

cfft(**A**) — пряме комплексне перетворення Фур'є для масиву комплексних чисел **A** (повертає масив того ж розміру, що й **A**). Якщо **A** є вектором, $\mathbf{c_j} = \frac{1}{\sqrt{\mathbf{n}}} \cdot \sum_{\mathbf{k}} \mathbf{A_k} \cdot \mathbf{exp}[\mathbf{i} \cdot (2 \cdot \pi \cdot \mathbf{j}/\mathbf{n}) \cdot \mathbf{k}]$, де **n** – кількість елементів вектору **A**

CFFT(**A**) — те саме, але при нормуванні $1/\mathbf{n}$ і від'ємному аргументі експоненти

fft(**v**) — пряме ШПФ для даних, записаних у векторі **v** у вигляді дійних чисел з $2^{\mathbf{n}}$ елементами, де **n** – ціле число (повертається вектор розмірністю $2^{\mathbf{n}-1}+1$): $\mathbf{c_j} = \frac{1}{\sqrt{\mathbf{n}}} \cdot \sum_{\mathbf{k}} \mathbf{v_k} \cdot \mathbf{exp}[\mathbf{i} \cdot (2 \cdot \pi \cdot \mathbf{j}/\mathbf{n}) \cdot \mathbf{k}]$

FFT(**v**) — те саме, але при нормуванні $1/\mathbf{n}$ і від'ємному аргументі експоненти

icfft(**B**) — обернене комплексне перетворення Фур'є, яке відповідає **cfft** (повертає масив того ж розмірності аргументу **B**): $\mathbf{c_j} = \frac{1}{\sqrt{\mathbf{n}}} \cdot \sum_{\mathbf{k}} \mathbf{B_k} \cdot \mathbf{exp}[-\mathbf{i} \cdot (2 \cdot \pi \cdot \mathbf{j}/\mathbf{n}) \cdot \mathbf{k}]$

ICFFT(**B**) — те саме, але при нормуванні $1/\mathbf{n}$ і додатному аргументі експоненти

ifft(**u**) — обернене ШПФ, яке відповідає **fft**. Вектор **u** має розмір $2^{\mathbf{n}-1}+1$, де **n** – ціле число (повертається вектор розмір $2^{\mathbf{n}}$): $\mathbf{c_j} = \frac{1}{\sqrt{\mathbf{n}}} \cdot \sum_{\mathbf{k}} \mathbf{u_k} \cdot \mathbf{exp}[-\mathbf{i} \cdot (2 \cdot \pi \cdot \mathbf{j}/\mathbf{n}) \cdot \mathbf{k}]$

IFFT(**u**) — те саме, але при нормуванні $1/\mathbf{n}$ і додатному аргументі експоненти

wave(**v**) — пряме вейвлет-перетворення (вектор **v** повинен містити $2^{\mathbf{n}}$ дійних елементів, де **n** – ціле число)

iwave(**u**) — вектор оберненого вейвлет-перетворення (**u** – вектор частотних даних вейвлет-спектру, розмірністю $2^{\mathbf{n}}$)

Функції розв'язку рівнянь

Given — директива, що відкриває блок розв'язку системи рівнянь, нерівностей

Find(**x**, **y**, ...) — вектор скалярних значень **x**, **y**, ..., які дають розв'язок системи рівнянь в блоці, об'явленому директивою **Given** (кількість повертаних значень дорівнює кількості аргументів)

lsolve(**M**, **v**) — повертає вектор невідомих **x**, що дають розв'язок системи лінійних алгебраїчних рівнянь виду **M**·**x** = **v**

Maximize(**f**, **x**, **y**, ...) — повертає вектор значень аргументів, при яких функція **f** досягає максимуму (можливе задання додаткових умов у блоці з **Given**)

Minerr (**x1**, **x2**, ...) — вектор значень для **x1**, **x2**, ..., які представляють розв'язок системи рівнянь у блоці, об'явленому директивою **Given** з мінімальною середньоквадратичною похибкою

Minimize(**f**, **x**, **y**, ...) — повертає вектор значень аргументів, при яких функція **f** досягає мінімуму (можливе задання додаткових умов у блоці з **Given**)

polyroots(**v**) — повертає вектор всіх коренів поліному, коефіцієнти якого містяться у **v**

root(**f**(**var**), **var**, [**a**, **b**]) — повертає значення змінної **var**, при якій вираз **f** = 0. [**a**, **b**] – можливий інтервал пошуку кореня

Функції помилок (інтеграли імовірностей)

erf(**x**) — функція помилок (функція Лапласа), $\frac{2}{\sqrt{\pi}} \cdot \int_0^{\mathbf{x}} \mathbf{exp}\left(-\mathbf{t}^2\right)\mathbf{dt}$

erfc(**x**) — комплементарна функція помилок, $\frac{2}{\sqrt{\pi}} \cdot \int_{\mathbf{x}}^{\infty} \mathbf{exp}\left(-\mathbf{t}^2\right)\mathbf{dt}$; **erfc**(**x**) = 1 – **erf**(**x**)

Експонентна та логарифмічні функції

exp(**z**) — основа натурального логарифму (число **e**) в степені **z**
log(**z**) — десятковий логарифм від аргументу **z**
log(**z**, **b**) — логарифм від аргументу **z** по основі **b**
ln(**z**) — натуральний логарифм від аргументу **z**

Функції повернення типу виразу

IsArray(**x**) — повертає 1, якщо **x** – масив, і 0 в інших випадках
IsScalar(**x**) — повертає 1, якщо **x** – скаляр, і 0 в інших випадках
IsString(**x**) — повертає 1, якщо **x** – тестовий рядок, і 0 в інших випадках

Функції імпортування та експортування файлів

APPENDPRN(*file*) — дописує зміст масиву в кінець розмежованого ASCII-файлу. Кількість стовпців в масиві повинна відповідати кількості стовпців в існуючому файлі

GETWAVINFO(*file*) — повертає вектор, що містить наступні характеристики WAV-файлу: кількість каналів, частота дискретизації, кількість рівнів квантування і середня кількість байт на секунду

READRGB(*file*) — масив, що складається з трьох підмасивів, які представляють відповідно червону, зелену і синю компоненти кольорового зображення (BMP, GIF, JPG, TGA, PBM, PGM, PPM або TIF формату), що знаходиться у *file*

READ_RED(*file*)
..._GREEN(*file*)
..._BLUE(*file*) — масиви, що відповідають відповідно червоній, зеленій і синій компонентам об’єкта (зображення), що міститься у *file*

READ_HLS(*file*) — масив, що представляє дані про колір об’єкта у *file* (відтінки кольору, яскравість та інтенсивність)

READ_HLS_HUE(*file*)
..._HLS_LIGHT(*file*)
..._HLS_SAT(*file*) — масиви, що представляють дані відповідно про відтінки, яскравість та інтенсивність кольору для об’єкта у *file*

READ_HSV(*file*) — масив, що представляє значення відтінку кольору, яскравості і насиченості для компонента у *file*

READ_HSV_HUE(*file*)
..._HSV_SAT(*file*)
..._HSV_VALUE(*file*) — масиви, що представляють значення відповідно відтінку, насиченості та інтенсивності кольору для компонента у *file*

READ_IMAGE(*file*) — масив, що представляє BMP, GIF, JPG, TGA, PCX, PBM, PGM, PPM або TIF-файл градаціями сірого

READBIN(*file*, *type*, [**t**]) — присвоєння масиву двійкових значень з файлу, тип (*type*) якого – *byte* (8 біт беззнакове ціле), *double* (64 біт з плаваючою комою) та ін.

READBMP(*file*) — масив, що представляє об’єкт BMP-формату з *file* градаціями сірого

READFILE(*file*, *type*, [**t**]) — присвоєння масиву даних з файлу типу *type*. **t** – додаткові параметри

READPRN(*file*) — присвоєння масиву значень з файлу з ім’ям *file.prn*

READWAV(*file*) — масив, кожен стовпець якого являє собою окремий канал, а кожен рядок відповідає моменту часу, що визначається номером відліку і частотою дискретизації сигналу

WRITERGB(*file*) — запис кольорового (16 млн. кольорів) зображення до файлу

WRITE_HSV(*file*) — те саме

WRITE_HLS(*file*) — те саме

WRITEBIN(*file*, *type*, 1/0) — запис масиву або скаляру до файлу двійкових даних

WRITEBMP(*file*) — запис зображення у відтінках сірого

WRITEPRN(*file*) — запис даних (масиву) до текстового файлу

WRITEWAV(*file*) — запис даних до звукового файлу

Тригонометричні та гіперболічні функції

a...(**z**) — обернена тригонометрична або гіперболічна функція "**...**" від аргументу **z**
...h(**z**) — гіперболічна функція "**...**"
cos(**z**) — косинус
csc(**z**) — косеканс
cot(**z**) — котангенс
sec(**z**) — секанс
sin(**z**) — синус
tan(**z**) — тангенс

Функції округлення числа

ceil(**z**) — повертає найменше ціле, що перевищує або дорівнює **z**
Ceil(**z**, **y**) — повертає найменше число кратне **y**, таке що ≥ **z** (при **y** ≠ 0)
floor(**z**) — повертає найбільше ціле число, що є меншим або дорівнює **z**
Floor(**z**, **y**) — повертає найбільше число кратне **y**, таке що ≤ **x** (при **y** ≠ 0)
round(**z**, [**n**]) — повертає аргумент **z**, округлений до **n** (опціонально) розрядів десяткового дробу. Якщо **n** опущене, повертає аргумент **z**, округлений до найближчого цілого (тобто вважається, що **n** = 0). При цьому функція є аналогічною **floor**(**z**), якщо дробова частина менша за 0.5, і **ceil**(**z**) в іншому випадку. Якщо **n** < 0, повертає аргумент **z**, округлений до **n** знаків ліворуч від десяткової коми
Round(**z**, **y**) — округлює **z** до найближчого кратного **y** шляхом обчислення **round**(**z**/**y**)·**y**
trunc(**z**) — повертає цілу частину числа **z** шляхом відкидання мантиси
Trunc(**z**, **y**) — повертає результат обчислення **trunc**(**z**/**y**)·**y**

Рядкові функції

concat(**S1**,**S2**, ...) — об'єднання рядкових змінних **S1**, **S2**, ... шляхом додавання рядка **S2** до кінця рядка **S1** і т.д.
error(**S**) — виведення повідомлення **S** про помилку
num2str(**z**) — повертає рядок, чиї символи відповідають десятковому значенню числа **z.**
search(**S**, **sub**, **m**) — здійснює пошук підрядка **sub** у рядку **S**, починаючи з позиції **m**
str2num(**Sn**) — перетворення рядкового представлення числа **Sn** на дійсне число. Число може бути комплексним, мати інженерний запис, формат двійкового і т.п.
str2vec(**S**) — перетворення рядка символів на вектор їх ASCII-кодів
strlen(**S**) — повертає кількість символів у рядку **S**
substr(**S**, **n**, **m**) — починаючи з позиції **n**, виокремлює з рядка **S** підрядок довжиною **m** символів
vec2str(**v**) — конвертує елементи вектора **v** ASCII-кодів у символьний рядок. Допустимі значення елементів вектора – 9, 10, 13, 32-255.

Векторні і матричні функції

Функції об'єднання масивів

augment(**A**, **B**, ...) — об'єднує бік у бік масиви **A**, **B**, ... з однаковою кількістю рядків зліва направо
stack(**A**, **B**, ...) — об'єднує масиви **A**, **B**, ... з однаковою кількістю стовпців зверху вниз

Функція розділення масивів

submatrix(**A**,**m**,**n**,**i**, **j**) — повертає підмасив з масиву **A**, що складається з елементів, спільних для рядків від **m** до **n** та стовпців від **i** до **j**. Необхідна умова: **m** ≤ **n** та **i** ≤ **j**

Функції створення масивів

diag(**v**) — повертає діагональну матрицю, елементи головної діагоналі якої – вектор **v**
identity(**n**) — повертає одиничну матрицю розмірністю **n**×**n**
matrix(**m**, **n**, **f**) — повертає матрицю, (**i**, **j**)-й елемент якої містить значення заданої попередньо деякої функції **f**(**i**, **j**), де **i** = 0, 1, ..., **m** та **j** = 0, 1, ..., **n**

Функції визначення розмірності масивів

cols(**A**) — повертає кількість стовпців у масиві **A**
last(**v**) — повертає індекс останнього елемента вектора **v**
length(**v**) — повертає загальну кількість елементів у векторі **v**
rows(**A**) — повертає кількість рядків у масиві **A**

Функції визначення екстремумів значень елементів масивів

max(**A**, **B**, ...) — повертає найбільший за значенням елемент серед масивів **A**, **B**, ... Якщо значення елементів є комплексними, максимум повертається окремо для дійсної і уявної частин
min(**A**, **B**, ...) — повертає найменший за значенням елемент серед масивів **A**, **B**, ... Якщо значення елементів є комплексними, мінімум повертається окремо для дійсної і уявної частин

Функції сортировки масивів

csort(**A**, **n**) — сортування масиву **A** шляхом перестановки рядків у порядку зростання значень елементів у стовпці **n**. Якщо масив містить комплексні елементи, уявна частина ігнорується (це стосується і наступних функцій сортировки)
reverse(**v**) — змінює порядок слідування елементів вектора **v** на протилежний
rsort(**A**, **n**) — сортування масиву **A** шляхом перестановки стовпців у порядку зростання значень елементів у рядку **n**
sort(**v**) — сортування елементів вектора **v** в порядку зростання їх значень

Функції пошуку

hlookup(**z**,**A**,**r**) — проводить пошук у верхньому рядку масиву **A** заданого значення **z**. У стовпцю, в якому знайдено **z**, обирається елемент в рядку **r**. Якщо знайдено декілька значень – повертається вектор (справедливо для всіх функцій пошуку)
lookup(**z**, **A**, **B**) — виконує пошук у масиві **A** заданого значення **z**. У випадку знайдення – повертає значення відповідного (за рядком і стовпцем) елементу масиву **B**
match(**z**, **A**) — проводить пошук у масиві **A** елементів, що мають значення **z** і повертає індекс (індекси) їх позицій в **A**
vlookup(**z**,**A**,**c**) — проводить пошук у лівому стовпцю масиву **A** заданого значення **z**. У рядку, в якому знайдено **z**, обирається елемент в стовпці **c**

Функції визначення числа обумовленості матриць

cond1(**M**) — повертає число обумовленості квадратної матриці **M**, обчислене в нормі $\mathbf{L_1}$ як $\mathbf{L_1}(\mathbf{M})\cdot\mathbf{L_1}(\mathbf{M}^{-1}) \xrightarrow{\text{MathCAD}} \mathbf{norm1}(\mathbf{M})\cdot\mathbf{norm1}(\mathbf{M}^{-1})$
cond2(**M**) — те саме, в нормі $\mathbf{L_2}$
conde(**M**) — те саме, в евклідовій нормі
condi(**M**) — те саме, у нескінченій нормі

Функції визначення норми матриць

norm1(**M**) — повертає $\mathbf{L_1}$-норму як максимальне значення серед результатів підсумовування абсолютних значень елементів стовпців матриці: $\mathbf{L_1}(\mathbf{M}) = \max_{j}\sum_{i=1}^{m}\left|\mathbf{M}_{i,j}\right|$
norm2(**M**) — повертає $\mathbf{L_2}$-норму як корінь квадратний з найбільшого власного значення матриці – λ: $\mathbf{L_2}(\mathbf{M}) = \sqrt{\max(\lambda)} \xrightarrow{\text{MathCAD}} \sqrt{\max\left(\text{eigenvals}\left(\mathbf{M}\cdot\overline{\mathbf{M}^{\mathbf{T}}}\right)\right)}$, де $\overline{\mathbf{M}^{\mathbf{T}}}$ – спряжена транспонована (можливий зворотній порядок) матриця **M**
norme(**M**) — повертає евклідову $\mathbf{L_e}$-норму як корінь квадратний з суми квадратів всіх елементів: $\mathbf{L_e}(\mathbf{M}) = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{m}\left(\mathbf{M}_{i,j}\right)^2}$
normi(**M**) — повертає нескінчену $\mathbf{L_\infty}$-норму як максимальне значення серед результатів підсумовування абсолютних значень елементів рядків матриці: $\mathbf{L_\infty}(\mathbf{M}) = \max_{i}\sum_{j=1}^{m}\left|\mathbf{M}_{i,j}\right|$

Функції визначення рангу і лінійних властивостей матриць

geninv(**A**) — створення лівої, зворотної до **A** (що має повний ранг) матриці, яка задовольняє рівнянню $\mathbf{L}\cdot\mathbf{A} = \mathbf{E}$, де **E** – одинична матриця $\mathbf{n}\times\mathbf{n}$, **L** – матриця $\mathbf{n}\times\mathbf{m}$, **A** – матриця $\mathbf{m}\times\mathbf{n}$, при $\mathbf{m} \geq \mathbf{n}$. Тобто $\mathbf{L} = (\mathbf{A}^{\mathbf{T}}\cdot\mathbf{A})^{-1}\cdot\mathbf{A}^{\mathbf{T}}$. Якщо **A** – квадратна, несингулярна матриця, $\mathbf{L} = \mathbf{A}^{-1}$

rank(**A**) — повертає ранг (кількість лінійно незалежних стовпців) масиву **A**

rref(**A**) — повертає масив – ступінчасту форму масиву **A** зі скороченою кількістю рядків

tr(**M**) — повертає слід (суму діагональних елементів) квадратної матриці **M**

Функції визначення власних векторів і власних значень матриць

eigenvals(**M**) — повертає вектор λ власних значень квадратної матриці **M**, кожен елемент якого задовольняє рівності $\mathbf{M}\cdot\mathbf{V}^{\langle i\rangle} = \lambda_i\cdot\mathbf{V}^{\langle i\rangle}$, де $\mathbf{V}^{\langle i\rangle}$ – **i**-й власний вектор матриці **M**. Підсумовування всіх елементів вектору λ еквівалентне функції обчислення сліду **tr**(**M**)

eigenvec(**M**, **z**) — повертає нормований власний вектор $\mathbf{V}^{\langle i\rangle}$ матриці **M**, що відповідає її власному значенню $\mathbf{z} = \lambda_i$

eigenvecs(**M**) — повертає матрицю **V**, стовпцями якої є власні вектори матриці **M** (порядок розміщення власних векторів відповідає порядку власних значень λ_i – **eigenvals**)

genvals(**M**, **N**) — повертає вектор узагальнених власних значень λ'_i матриці **M**, який відповідає матричному виразові $\mathbf{M}\cdot\mathbf{V'}^{\langle i\rangle} = \lambda'_i\cdot\mathbf{N}\cdot\mathbf{V'}^{\langle i\rangle}$. **M** і **N** – квадратні матриці дійсних елементів

genvecs(**M**, **N**) — повертає матрицю **V'**, стовпці якої містять нормовані узагальнені власні вектори (порядок розміщення векторів відповідає порядку узагальнених власних значень λ'_i, що повертаються функцією **genvals**)

Функції розкладання матриць

cholesky(**M**) — реалізує розкладання матриці за Холецьким (метод квадратних коренів), повертаючи нижньо-трикутну матрицю **L** (всі елементи над головною діагоналлю є нульовими), що задовольняє рівність $\mathbf{L}\cdot\mathbf{L}^{\mathbf{T}} = \mathbf{M}$, де **M** – дійсна, позитивно визначена квадратна матриця. При обчисленні використовується лише верхньо-трикутна частина **M**

lu(**M**) — реалізує розкладання квадратної матриці **M** шляхом повернення масиву, що містить три об'єднані бік у бік квадратні матриці **P**, **L** і **U**, однакової розмірності з **M**. Дані матриці задовольняють рівнянню $\mathbf{P}\cdot\mathbf{M} = \mathbf{L}\cdot\mathbf{Q}$, де **L** і **U** – відповідно нижньо- і верхньо-трикутні матриці. Для виділення **P**, **L** і **U** зі спільного масиву треба використати функцію **submatrix** (це ж стосується функцій **qr** і **svd**)

qr(**A**) — реалізує розкладання масиву **A** розмірністю $\mathbf{m}\times\mathbf{n}$ шляхом повернення масиву, чиї перші **m** стовпців є квадратною ортонормованою матрицею **Q**, яка має однакову з **A** кількість рядків, а **n** наступних стовпців містять верхньо-трикутну матрицю **R**, яка задовольняє рівнянню $\mathbf{A} = \mathbf{Q}\cdot\mathbf{R}$.

svd(**A**) — реалізує розкладання масиву **A** розмірністю $\mathbf{m}\times\mathbf{n}$ ($\mathbf{m} \geq \mathbf{n}$) по його сингулярним числам. Повертає масив, що складається з розміщених одна над одною матриць **U** і **V**, які задовольняють рівнянню $\mathbf{A} = \mathbf{U}\cdot\mathbf{D}\cdot\mathbf{V}^{\mathbf{T}}$, де матриці **U** – верхня, розмірністю $\mathbf{m}\times\mathbf{n}$; **V** – нижня, розмірністю $\mathbf{n}\times\mathbf{n}$; **D** – діагональна, на діагоналі якої розміщені сингулярні числа масиву **A**, одержані за допомогою функції **svds**: $\mathbf{D} \xrightarrow{\text{MathCAD}} \mathbf{diag}(\mathbf{svds}(\mathbf{A}))$

svds(**A**) — повертає вектор сингулярних чисел (корінь квадратний від власних значень матриці $\mathbf{A}^{\mathbf{T}}\cdot\mathbf{A}$) масиву **A**. Останній може мати розмірність $\mathbf{m}\times\mathbf{n}$, $\mathbf{m} \geq \mathbf{n}$.

Функції одержання логарифмічно рознесених точок

logspace(**min**, **max**, **npts**) — повертає вектор з **npts** логарифмічно рознесених точок, починаючи з **min** і завершуючи **max** (**min** і **max** – позитивні дійсні числа)

logpts(**minexp**, **dec**, **dnpts**) — повертає вектор, що охоплює **dec** декад з **dnpts** рівновіддалених точок, починаючи з $10^{\mathbf{minexp}}$, по **dnpts** точок на декаду. Кількість елементів результуючого вектора – $\mathbf{dec}\cdot\mathbf{dnpts}$

Функція обчислення кореляції векторів

correl(**vx**, **vy**) — повертає одномірний коефіцієнт кореляції векторів **vx** і **vy**. Результуючий вектор має довжину **length**(**vx**) + **length**(**vy**) − 1, де кожен елемент є результатом векторного добутку вектора **vx** із зсунутою версією **vy**

Деякі додаткові функції обробки масивів зображення

addnoise(**A**, **p**, **n**) — повертає масив **A** з доданим шумом. Причому до значення пікселя додається або віднімається значення **n** ($0 \le |\mathbf{n}| \le 255$) з імовірністю $\mathbf{p}/2$ ($0 \le \mathbf{p} \le 1$)

and(**A**, **B**) / **or**(**A**, **B**) — повертає результат виконання булевих операцій "`і`"/"`або`" над масивами зображень **A** і **B** (останні повинні мати однакову розмірність)

augment3(**R**, **G**, **B**) — призначення аналогічне функції **augment**, але обов'язково три аргументи

binarize(**A**, **thresh**) — повертає двійкову версію масиву **A**: пікселям, значення яких перевищують поріг **thresh** присвоюється значення 1, іншим – 0

binarize_auto(**A**) — те саме, але значення порогу обирається програмою автоматично

binarize2(**A**, **t1**, **t2**, **x**, **y**) — повертає двійкову версію масиву **A**: пікселям, значення яких не виходять за межі [**t1**, **t2**], присвоюється значення **x**, іншим – **y**

blend(**A**, **B**) — повертає суміш співрозмірних масивів **A, B** шляхом попіксельного обчислення виразу: $\mathbf{A}_{i,j} + \mathbf{B}_{i,j} - (\mathbf{A}_{i,j} \cdot \mathbf{B}_{i,j} / 255)$

canny(**A**, **σ**, **low**, **high**) — повертає двійкове контурне зображення, одержане в результаті виділення контурів зображення за алгоритмом Кенні. **σ** > 0 – середньоквадратичне відхилення (зазвичай, не більше 2). **low** < **high** – гістерезисні пороги (дійсні)

center(**A**) — повертає результат ДПФ масиву **A**, трансформований таким чином, що постійна складова знаходиться у центрі

centsmooth(**A**) — повертає масив **A**, згладжений за допомогою центрально зваженого елементу розмірністю 3×3

clip(**A**, **Min**, **Max**) — повертає масив **A**, значення елементів якого обмежені порогами **Min** і **Max**

colgrad(**A**) — повертає постовпцевий градієнт (відмінність між стовпцями) масиву **A**

dct2d(**A**) — повертає двомірне швидке ДКП масиву **A**

distform(**A**, **fg**) — трансформація масиву **A** за евклідовою відстанню для відтінку сірого **fg**

equalize(**A**) — повертає масив **A** зображення з інтенсивністю пікселів шкали рівнів сірого, врегульованою таким чином, щоб формувати лінійну сукупну гістограму

extract(**A**, **n**) — виокремлює підмасив **n**-го колірного компоненту (**n** = 1_R, 2_G, 3_B) з три-компонентного масиву **A**

funmap(**A**, **f**) — застосовує функцію однієї змінної **f** до кожного елементу масиву **A**

getnoise(**A**) — повертає відмінність між первинним і медіанно-відфільтрованим за допомогою функції **medfilt** масивом **A**

hist2d(**A**, **B**, **n**) — повертає двомірну **n**-стовпцеву колірну гістограму для масивів **A** і **B** однакової розмірності

horzflip(**A**) — дзеркальне відображення навколо горизонтальної вісі масиву **A**

idct2d(**A**) — повертає обернене двомірне швидке ДКП масиву **A**

imhist(**A**, **n**) — повертає **n**-стовпцеву колірну гістограму масиву **A**, для значень між 0 і 255 включно (значення, що виходять за цей діапазон, ігноруються)

imhist2(**A**, **n**) — повертає **n**-стовпцеву колірну гістограму масиву **A** для наявного діапазону значень інтенсивностей

immse(**A**, **B**) — повертає середньоквадратичну помилку між масивами зображення **A** і **B**

imquant(**A**, **n**) — повертає квантовану версію масиву **A**, яка містить лише **n** рівнів сірого між 0 і 255, що рівновіддалені один від одного

imquant2(**A**, **v**) — повертає квантовану версію **A**, яка містить зазначені у векторі **v** рівні сірого

imsnr(**A**, **B**) — повертає відношення "сигнал/шум" між масивами зображення **A** і **B**

invert(**A**) — повертає інвертований масив зображення **A** шляхом поелементного віднімання від 255 значень інтенсивностей всіх пікселів

invert2(**A**) — повертає масив **A**, нове значення елементів якого обчислюється шляхом віднімання від **max**(**A**) значення поточного елементу і додавання до одержаного результату **mix**(**A**)

levelmap(**A**, **v**) — повертає масив **A** зі зміненою інтенсивністю на значення, яке має елемент вектору **v** ($0 \le \mathbf{v}_i \le 255$), індекс **i** якого відповідає первинній інтенсивності пікселя. Розмірність **v** не повинна бути меншою за наявне максимальне значення інтенсивності на всій множині пікселів зображення

mask(**A**, **B**) — повертає масив **A**, кожен елемент якого замінюється на нуль, якщо відповідний елемент масиву-маски **B** дорівнює 0

medfilt(**A**) — повертає медіанно-відфільтрований масив **A**

orthosmooth[5](**A**) — повертає масив **A**, згладжений за допомогою ортогонально зваженого елементу розмірністю 3×3 (або 5×5 для **orthosmooth5**)

putregion(A, B, r, c) — вбудовує масив **B** до масиву **A**, починаючи з рядка **r** і стовпця **c**. Необхідні умови: **r** + **rows**(**B**) –1 < **rows**(**A**) та **c** + **cols**(**B**) –1 < **cols**(**A**)

quantfilt(**A**, **W**, **quant**) — повертає квантильно-відфільтрований масив **A**, використовуючи при цьому суміжну матрицю **W** (довільної розмірності) та квантиль 0 ≤ **quant** ≤ 1

relerror(**A**, **B**) — повертає значення відносної похибки між масивами зображення **A** і **B**

replace(**A**, **B**, **n**) — повертає трикомпонентний масив **A** із заміненим на матрицю **B** підмасивом **n**-го колірного компоненту. Матриця **B** повинна мати однакову з **A** кількість рядків і третину від кількості стовпців **A**

rotate(**A**, α) — обертає масив **A** на α градусів проти ходу годинникової стрілки

rotate90[180,270](**A**) — обертає масив **A** на 90, 180 або 270° проти ходу годинникової стрілки

rowgrad(**A**) — повертає відрядковий градієнт (відмінність між рядками) масиву **A**

scale(**A**, **Min**, **Max**) — повертає масив **A**, елементи якого віднормовані у діапазоні [**Min**, **Max**]

shape_features(**A**) — повертає матрицю моментів і особливостей відображення кожного окремого пікселя зображення **A**

submatrix(**A**,**m**,**n**,**i**, **j**) — повертає RGB-підмасив з трикомпонентного масиву **A**, обмежений рядками від **m** до **n** та стовпцями від **i** до **j**. Необхідна умова: **m** ≤ **n** та **i** ≤ **j**

threshold(**A**, **thresh**) — повертає масив **A**, із присвоєнням елементам, значення яких є меншими порогу **thresh**, нульових значень. Якщо **thresh** є від'ємним числом, нуль присвоюється тим елементам, значення яких перевищують значення |**thresh**|

translate(**A**, **r**, **c**, **pad**) — повертає масив однієї розмірності з **A**, причому оригінальні елементи **A** зсунуті на **r** рядків і **c** стовпців, незаповнені елементи нового масиву що при цьому з'явилися, набувають інтенсивності 0 ≤ **pad** ≤ 255. Знак **r** і **c** визначає напрямок зсуву (плюс – вниз або праворуч, мінус – вгору або ліворуч)

unismooth[5](**A**) — повертає масив **A**, згладжений за допомогою рівномірно зваженого елементу розмірністю 3×3 (або 5×5 для **unismooth5**)

vertflip(**A**) — дзеркальне відображення навколо вертикальної вісі масиву **A**

warp(**A**, **T**) — застосовує білінійну деформацію зображення, що представлене масивом **A**, використовуючи точки прив'язки, що містяться в масиві **T**. Останній має розмірність 8×2 і містить, таким чином, 4 пари точок

zoom(**A**, **hsc**, **vsc**) — повертає масив **A**, промасштабований з коефіцієнтами **hsc**, **vsc** (відповідно, по горизонталі і вертикалі). Останні повинні бути додатними, ненульовими дійсними числами

Деякі додаткові функції обробки масивів сигналів

ccepstrum(**A**) — повертає масив – результат обчислення комплексного кепстру багатоканального сигналу **A**

cepstrum(**v**) — повертає кепстр вектора **v**

chirpz(**A**, **lo**, **hi**, **d**) — повертає частотний спектр сигналу **A** між **lo** та **hi** з інтервалом частот **d**, причому 0 ≤ **lo** < **hi** ≤ 0.5, а (**hi** – **lo**)/**d** ≥ 2

costr(**A**) — косинусне перетворення масиву **A**, що містить парну кількість елементів

icostr(**A**) — обернене косинусне перетворення масиву **A**

sintr(**A**) — синусне перетворення масиву **A**, що містить парну кількість елементів

isintr(**A**) — обернене синусне перетворення масиву **A**

phase(**A**) — повертає масив фаз, обчислених на основі комплексних елементів масиву **A**

mag(**A**) — повертає масив амплітуд на основі комплексних елементів масиву **A**

makeri(**Mag**, **Phase**) — повертає масив комплексних чисел на основі амплітудних і фазових елементів, що містяться відповідно у співрозмірних масивах **Mag** і **Phase**

snr(**vx**, **vy**, **n**, **r**, [**w**]) — обчислює відношення "сигнал/шум" для векторів сигналу **vx** і **vy**. Вектори поділяються на 1 < **n** < **length**(**vx**) перекриваючихся сегментів з коефіцієнтом перекриття 0 ≤ **r** > 1. Кожен сегмент оброблюється вікном зі смугою **w**

whiten(**n**) — повертає вектор розмірністю **n**, елементи якого являють собою рівномірно розподілений білий шум

convol(**A**, **B**) — обчислення згортки масивів **A** і **B**, які повинні мітити як мінімум 2 елементи

deconvol(**B**, **A**) — обернення згортки – знаходження оригіналу **B** на основі **A**

Додаток С

Математичні і системні константи MathCAD

Нижче подано найменування математичних і системних констант, правила їх введення та призначення. Зазначеним константам можна присвоїти й інші значення: або за допомогою оператора присвоєння безпосередньо в робочому документі, або через діалогове вікно *Worksheet Options* з меню *Tools*. У дужках подано встановлені значення за умовчанням.

Константа	*Введення*	*Призначення*
π	[Ctrl][Shift]p або p[Ctrl]g	Число **π**. У чисельних розрахунках MathCAD використовує значення **π** з урахуванням 16 значущих цифр (3,1415926535897931). У символьних обчисленнях **π** зберігає своє точне значення
e	e	Основа натурального логарифму (2,7182818284590451). У символьних обчисленнях **e** зберігає своє точне значення
∞	[Ctrl][Shift]z	Системна нескінченність (10^{307})
%	%	Відсоток (0,01)
1j (**1i**)	1j (1i)	Комплексна одиниця ($\sqrt{-1}$)
NaN		Не числове значення
ORIGIN		Нижня границя індексації масивів (0)
TOL		Похибка чисельних методів різноманітних алгоритмів апроксимації, яка визначає умови припинення ітерацій чисельним алгоритмом (0,001)
CTOL		Похибка збіжності у блоках розв'язку рівнянь, яка обмежує нев'язку, задаючи точність виконання рівнянь (0,001)
Seed		Початкове число при генерації ПВЧ (1)
PRNPRECISION		Кількість десяткових знаків, використовуваних при запису файлів оператором **WRITEPRN** (4)
PRNCOLWIDTH		Ширина стовпця (кількість символів), яка використовується при запису файлів оператором оператора **WRITEPRN** (8)
CWD		Текстова змінна, що зберігає адресу поточного (робочого) документа на диску
FRAME		Змінна лічильника кадрів при роботі з анімаційними рисунками (0)

Додаток D

Програмні оператори MathCAD

Програмний оператор	*Шаблон*	*Призначення*
Add Line		Виконує функції розширення програмного модуля. Розширення фіксується подовженням вертикальної риски програмних модулів та їх деревовидним розширенням. Завдяки цьому можна створювати як завгодно великі програми
←		Оператор внутрішнього (в тілі програмного модуля) локального присвоєння
if		Оператор створення умовних виразів: *"вираз"* **if** *"умова"* Якщо *"умова"* виконується, то повертається значення *"вираз"*. У парі з даним оператором зазвичай використовуються оператори **break** і **otherwise**
for		Оператор організації циклів із заданою кількістю повторень: **for** *"змінна"* ∈ **Nmin** .. **Nmax** *"вираз"* Якщо *"змінна"* змінюється, наприклад, з кроком +1 від значення **Nmin** до **Nmax**, то *"вираз"* виконуватиметься. Сама ж *"змінна"* може бути використана у виразах програмного модуля
while		Оператор організації циклів, діючих то тих пір, поки виконується деяка *"умова"*: **while** *"умова"* *"вираз"*
otherwise		Оператор альтернативної умови. Зазвичай використовується спільно з оператором **if** . Наприклад, модуль $f(x) := \begin{cases} 1 & \text{if } x < 0 \\ -1 & \text{otherwise} \end{cases}$ повертає 1, якщо змінна **x** < 0, в усіх інших випадках повертається значення –1
break		Оператор виклику зупинки роботи програми (циклу). Зазвичай використовується спільно з оператором умовного виразу **if** та операторами циклів **for** і **while**, забезпечуючи перехід на кінець тіла циклу
continue		Оператор продовження. Використовується для продовження роботи після переривання програми. Використовується в парі з операторами задання циклів **for** і **while**, забезпечуючи після переривання перехід на початок циклу
return		Оператор-функція повернення. Перериває виконання програми і повертає значення свого операнда-аргумента. Наприклад, модуль $f(x) := \begin{cases} \text{return } 20 & \text{if } x < 1 \\ 1 \end{cases}$ повертає 20 при будь-якому **x** < 1 , в усіх інших випадках повертається значення 1
on error		Оператор обробки помилок. Дозволяє створювати конструкції обробників помилок: *"вираз 1"* **on error** *"вираз 2"* Якщо при виконанні *"виразу 2"* виникає помилка, виконується *"вираз 1"*

Додаток E

Таблиця ASCII-кодів

DEC	*BIN*	*Симв.*	*Опис*	*DEC*	*BIN*	*Симв.*	*Опис*
0	00000000	*NuL*	Не визначений	64	01000000	@	Комерційне "ет"
1	00000001	☺	Початок заголовку	65	01000001	A	Велика лат. *A*
2	00000010	☻	Початок тексту	66	01000010	B	Велика лат. *B*
3	00000011	♥	Кінець тексту	67	01000011	C	Велика лат. *C*
4	00000100	♦	Кінець передачі	68	01000100	D	Велика лат. *D*
5	00000101	♣	Запит	69	01000101	E	Велика лат. *E*
6	00000110	♠	Підтвердження прийому	70	01000110	F	Велика лат. *F*
7	00000111	•	Звуковий сигнал	71	01000111	G	Велика лат. *G*
8	00001000	◘	Повернення, забій	72	01001000	H	Велика лат. *H*
9	00001001	○	Горизонт. табуляція	73	01001001	I	Велика лат. *I*
10	00001010	◙	Переведення рядка	74	01001010	J	Велика лат. *J*
11	00001011	♂	Вертик. табуляція	75	01001011	K	Велика лат. *K*
12	00001100	♀	Переведення сторінки	76	01001100	L	Велика лат. *L*
13	00001101	♪	Повернення каретки	77	01001101	M	Велика лат. *M*
14	00001110	♫	Верхній регістр	78	01001110	N	Велика лат. *N*
15	00001111	☼	Нижній регістр	79	01001111	O	Велика лат. *O*
16	00010000	►	Відключ. від лінії	80	01010000	P	Велика лат. *P*
17	00010001	◄	Керування 1	81	01010001	Q	Велика лат. *Q*
18	00010010	↕	Керування 2	82	01010010	R	Велика лат. *R*
19	00010011	‼	Керування 3	83	01010011	S	Велика лат. *S*
20	00010100	¶	Керування 4	84	01010100	T	Велика лат. *T*
21	00010101	§	Нема підтвердження	85	01010101	U	Велика лат. *U*
22	00010110	▬	Синхронізація	86	01010110	V	Велика лат. *V*
23	00010111	↨	Кінець перед. блоку	87	01010111	W	Велика лат. *W*
24	00011000	↑	Відміна	88	01011000	X	Велика лат. *X*
25	00011001	↓	Кінець носія	89	01011001	Y	Велика лат. *Y*
26	00011010	→	Заміна	90	01011010	Z	Велика лат. *Z*
27	00011011	←	Переривання	91	01011011	[	Ліва квадратна дужка
28	00011100	∟	Роздільник файлів	92	01011100	\	Зворотна коса риска
29	00011101	↔	Роздільник груп	93	01011101	]	Права квадратна дужка
30	00011110	▲	Роздільник записів	94	01011110	^	Знак вставки
31	00011111	▼	Роздільник елементів	95	01011111	_	Підкреслення
32	00100000		Пробіл	96	01100000	`	Тупий наголос
33	00100001	!	Знак оклику	97	01100001	a	Мала лат. *a*
34	00100010	"	Подвійні лапки	98	01100010	b	Мала лат. *b*
35	00100011	#	Знак числа	99	01100011	c	Мала лат. *c*
36	00100100	$	Знак долара	100	01100100	d	Мала лат. *d*
37	00100101	%	Знак проценту	101	01100101	e	Мала лат. *e*
38	00100110	&	Амперсанд	102	01100110	f	Мала лат. *f*
39	00100111	'	Апостроф	103	01100111	g	Мала лат. *g*
40	00101000	(	Ліва кругла дужка	104	01101000	h	Мала лат. *h*
41	00101001	)	Права кругла дужка	105	01101001	i	Мала лат. *i*
42	00101010	*	Зірочка	106	01101010	j	Мала лат. *j*
43	00101011	+	Знак плюс	107	01101011	k	Мала лат. *k*
44	00101100	,	Кома	108	01101100	l	Мала лат. *l*
45	00101101	-	Знак мінус	109	01101101	m	Мала лат. *m*
46	00101110	.	Крапка	110	01101110	n	Мала лат. *n*
47	00101111	/	Ліва коса риска	111	01101111	o	Мала лат. *o*
48	00110000	0	Цифра нуль	112	01110000	p	Мала лат. *p*
49	00110001	1	Цифра один	113	01110001	q	Мала лат. *q*
50	00110010	2	Цифра два	114	01110010	r	Мала лат. *r*
51	00110011	3	Цифра три	115	01110011	s	Мала лат. *s*
52	00110100	4	Цифра чотири	116	01110100	t	Мала лат. *t*
53	00110101	5	Цифра п’ять	117	01110101	u	Мала лат. *u*
54	00110110	6	Цифра шість	118	01110110	v	Мала лат. *v*
55	00110111	7	Цифра сім	119	01110111	w	Мала лат. *w*
56	00111000	8	Цифра вісім	120	01111000	x	Мала лат. *x*
57	00111001	9	Цифра дев’ять	121	01111001	y	Мала лат. *y*
58	00111010	:	Двокрапка	122	01111010	z	Мала лат. *z*
59	00111011	;	Крапка з комою	123	01111011	{	Ліва фігурна дужка
60	00111100	<	Знак менше	124	01111100	|	Вертикальна риска
61	00111101	=	Знак рівності	125	01111101	}	Права фігурна дужка
62	00111110	>	Знак більше	126	01111110	~	Тильда
63	00111111	?	Знак питання	127	01111111	⌂	Видалення

DEC	BIN	Симв.	Опис	DEC	BIN	Симв.	Опис
128	10000000	Ђ	Велика кирил. *Дже*	192	11000000	А	Велика кирил. *А*
129	10000001	Ѓ	Велика кирил. *Ге*	193	11000001	Б	Велика кирил. *Б*
130	10000010	‚	Нижній апостроф	194	11000010	В	Велика кирил. *В*
131	10000011	ѓ	Мала кирил. *ге*	195	11000011	Г	Велика кирил. *Г*
132	10000100	„	Подвійні нижні лапки	196	11000100	Д	Велика кирил. *Д*
133	10000101	…	Горизонт. три крапки	197	11000101	Е	Велика кирил. *Е*
134	10000110	†	Початок виділ. даних	198	11000110	Ж	Велика кирил. *Ж*
135	10000111	‡	Кінець виділ. даних	199	11000111	З	Велика кирил. *З*
136	10001000	€	Знак євро	200	11001000	И	Велика кирил. *И*
137	10001001	‰	Знак проміле	201	11001001	Й	Велика кирил. *Й*
138	10001010	Љ	Велика кирил. *Ле*	202	11001010	К	Велика кирил. *К*
139	10001011	‹	Ліва куткова лапка	203	11001011	Л	Велика кирил. *Л*
140	10001100	Њ	Велика кирил. *Не*	204	11001100	М	Велика кирил. *М*
141	10001101	Ќ	Велика кирил. *Ке*	205	11001101	Н	Велика кирил. *Н*
142	10001110	Ћ	Велика кирил. *Ше*	206	11001110	О	Велика кирил. *О*
143	10001111	Џ	Велика кирил. *Же*	207	11001111	П	Велика кирил. *П*
144	10010000	ђ	Мала кирил. *де*	208	11010000	Р	Велика кирил. *Р*
145	10010001	‘	Ліва одиночна лапка	209	11010001	С	Велика кирил. *С*
146	10010010	’	Права одиночна лапка	210	11010010	Т	Велика кирил. *Т*
147	10010011	“	Ліві лапки	211	11010011	У	Велика кирил. *У*
148	10010100	”	Праві лапки	212	11010100	Ф	Велика кирил. *Ф*
149	10010101	•	Очікув. повідомлення	213	11010101	Х	Велика кирил. *Х*
150	10010110	–	Коротке тире	214	11010110	Ц	Велика кирил. *Ц*
151	10010111	—	Довге тире	215	11010111	Ч	Велика кирил. *Ч*
152	10011000	□	Не визначений	216	11011000	Ш	Велика кирил. *Ш*
153	10011001	™	Торгівельна марка	217	11011001	Щ	Велика кирил. *Щ*
154	10011010	љ	Мала кирил. *ле*	218	11011010	Ъ	Велика кирил. *Ъ*
155	10011011	›	Права куткова лапка	219	11011011	Ы	Велика кирил. *Ы*
156	10011100	њ	Мала кирил. *не*	220	11011100	Ь	Велика кирил. *Ь*
157	10011101	ќ	Мала кирил. *ке*	221	11011101	Э	Велика кирил. *Э*
158	10011110	ћ	Мала кирил. *ше*	222	11011110	Ю	Велика кирил. *Ю*
159	10011111	џ	Мала кирил. *же*	223	11011111	Я	Велика кирил. *Я*
160	10100000		Нерозривний пробіл	224	11100000	а	Мала кирил. *а*
161	10100001	Ў	Вел. кирил. *У кратке*	225	11100001	б	Мала кирил. *б*
162	10100010	ў	Мала кирил. *у кратке*	226	11100010	в	Мала кирил. *в*
163	10100011	Ј	Велика кирил. *Йе*	227	11100011	г	Мала кирил. *г*
164	10100100	¤	Валюта	228	11100100	д	Мала кирил. *д*
165	10100101	Ґ	Велика кирил. *Ге*	229	11100101	е	Мала кирил. *е*
166	10100110	¦	Розірвана риска	230	11100110	ж	Мала кирил. *ж*
167	10100111	§	Знак параграфу	231	11100111	з	Мала кирил. *з*
168	10101000	Ё	Велика кирил. *Ё*	232	11101000	и	Мала кирил. *и*
169	10101001	©	Авторське право	233	11101001	й	Мала кирил. *й*
170	10101010	Є	Велика кирил. *Є*	234	11101010	к	Мала кирил. *к*
171	10101011	«	Ліві куткові лапки	235	11101011	л	Мала кирил. *л*
172	10101100	¬	Куткове тире	236	11101100	м	Мала кирил. *м*
173	10101101	-	М’який перенос	237	11101101	н	Мала кирил. *н*
174	10101110	®	Зареєстр. торг. знак	238	11101110	о	Мала кирил. *о*
175	10101111	Ї	Велика кирил. *Ї*	239	11101111	п	Мала кирил. *п*
176	10110000	°	Градус	240	11110000	р	Мала кирил. *р*
177	10110001	±	"Плюс або мінус"	241	11110001	с	Мала кирил. *с*
178	10110010	І	Велика кирил. *І*	242	11110010	т	Мала кирил. *т*
179	10110011	і	Мала кирил. *і*	243	11110011	у	Мала кирил. *у*
180	10110100	ґ	Мала кирил. *Ге*	244	11110100	ф	Мала кирил. *ф*
181	10110101	µ	Знак мікро (мала мю)	245	11110101	х	Мала кирил. *х*
182	10110110	¶	Абзац	246	11110110	ц	Мала кирил. *ц*
183	10110111	·	Серединна точка	247	11110111	ч	Мала кирил. *ч*
184	10111000	ё	Мала кирил. *ё*	248	11111000	ш	Мала кирил. *ш*
185	10111001	№	Номер	249	11111001	щ	Мала кирил. *щ*
186	10111010	є	Мала кирил. *Є*	250	11111010	ъ	Мала кирил. *ъ*
187	10111011	»	Праві куткові лапки	251	11111011	ы	Мала кирил. *ы*
188	10111100	j	Мала кирил. *йе*	252	11111100	ь	Мала кирил. *ь*
189	10111101	S	Велика кирил. *Зе*	253	11111101	э	Мала кирил. *э*
190	10111110	s	Мала кирил. *зе*	254	11111110	ю	Мала кирил. *ю*
191	10111111	ї	Мала кирил. *ї*	255	11111111	я	Мала кирил. *я*

Таблиця 5.1.

Показники візуального спотворення у випадку приховування даних в просторовій області зображення

Назва показника спотворення	*Оригінал*	*Методи приховування у просторовій області*							
		НЗБ	*ПВ інтервал*	*ПВП*	*Блокового кодування*	*Заміни палітри*	*Квантування*	*Куттера-Джордона*	*Дармстедтера-Делейгла*
Макс. абсолютна різниця, ***MD***	0	1	1	1	1	3	3	38	54
Середня абсолютна різниця, ***AD***	0	0.494	$7.690\cdot10^{-3}$	$5.920\cdot10^{-3}$	$6.165\cdot10^{-3}$	$9.827\cdot10^{-3}$	$7.141\cdot10^{-3}$	4.588	17.704
Норм. середня абс. різниця, ***NAD***	0	$3.823\cdot10^{-3}$	$5.956\cdot10^{-5}$	$4.585\cdot10^{-5}$	$4.774\cdot10^{-5}$	$7.611\cdot10^{-5}$	$5.535\cdot10^{-5}$	0.050	0.137
Середньоквадр. помилка, ***MSE***	0	0.494	$7.690\cdot10^{-3}$	$5.920\cdot10^{-3}$	$6.165\cdot10^{-3}$	0.017	$9.460\cdot10^{-3}$	235.708	456.887
Нормована середньоквадр. помилка, ***NMSE***	0	$2.010\cdot10^{-5}$	$3.132\cdot10^{-7}$	$2.411\cdot10^{-7}$	$2.510\cdot10^{-7}$	$7.084\cdot10^{-7}$	$3.853\cdot10^{-7}$	$9.599\cdot10^{-3}$	0.019
L^p-норма, $p = 2$	0	0.703	0.088	0.077	0.079	0.132	0.097	11.301	21.375
Лапласова середньоквадр. помилка, ***LMSE***	0	$9.815\cdot10^{-4}$	$1.560\cdot10^{-5}$	$1.263\cdot10^{-5}$	$1.253\cdot10^{-5}$	$1.990\cdot10^{-5}$	$1.855\cdot10^{-5}$	0.240	0.420
Відношення с/ш, ***SNR***	∞	$4.975\cdot10^{4}$	$3.193\cdot10^{6}$	$4.148\cdot10^{6}$	$3.983\cdot10^{6}$	$1.412\cdot10^{6}$	$2.596\cdot10^{6}$	192.271	53.746
Макс. відношення с/ш, ***PSNR***	∞	$1.317\cdot10^{5}$	$8.455\cdot10^{6}$	$1.044\cdot10^{7}$	$1.055\cdot10^{7}$	$3.738\cdot10^{6}$	$6.873\cdot10^{6}$	509.139	142.322
Якість зображення, ***IF***	1	0.999980	≈1	≈1	≈1	0.999999	≈1	0.994799	0.981394
Норм. взаємна кореляція, ***NC***	1	0.999439	0.999992	0.999998	0.999988	0.999942	1.000001	0.988343	0.942705
Якість кореляції, ***CQ***	190.182	190.076	190.181	190.182	190.180	190.172	190.183	187.966	179.286
Структурний зміст, ***SC***	1	1.001103	1.000016	1.000004	1.000025	1.000114	0.999999	1.018447	1.106175
Загальне сигма-відношення с/ш, ***GSSNR***	∞	$1.298\cdot10^{5}$	$9.751\cdot10^{6}$	$8.026\cdot10^{6}$	$5.306\cdot10^{6}$	$5.797\cdot10^{5}$	$3.648\cdot10^{6}$	187.522	31.555
Сигма-відношення с/ш, ***SSNR***	∞	142.5	62	42.4	39.7	7.6	19.5	41.8	57.3
Нормоване відношення с/помилка, ***NSER***	256	60	155	175	179	241	214	111	83.5

Назва показника спотворення	*Оригінал*	*Методи приховування у просторовій області*							
		НЗБ	*ПВ інтервал*	*ПВП*	*Блокового кодування*	*Заміни палітри*	*Квантування*	*Куттера-Джордона*	*Дармстедтера-Делейгла*
Подібність гістограм, ***HS***	0	3918	176	138	154	184	150	2068	10372

Таблиця 5.4.

Показники візуального спотворення у випадку приховування даних в частотній області зображення

Назва показника спотворення	*Оригінал*	*Методи приховування у частотній області*								
		Коха- Жао (**P** = 0.5)	*Коха- Жао* (**P** = 25)	*Бенгама-Мемона* (**P** = 1)	*Бенгама-Мемона* (**P** = 35)	*Хсу-Ву (а)*	*Хсу-Ву (б)*	*Фрідріх* (α = 0.1, γ = 1)	*Фрідріх* (α = 0.1, γ = 2)	*Фрідріх* (α = 0.2, γ = 1)
Макс. абсолютна різниця, ***MD***	0	39	45	45	51	122	88	68	94	73
Середня абсолютна різниця, ***AD***	0	9.504	11.392	1.846	3.042	36.805	26.457	19.233	25.990	22.413
Норм. середня абс. різниця, ***NAD***	0	0.074	0.088	0.014	0.024	0.251	0.181	0.132	0.179	0.154
Середньоквадр. помилка, ***MSE***	0	124.383	178.342	15.427	31.417	1794.673	962.154	543.072	942.113	718.449
Нормована середньоквадр. помилка, ***NMSE***	0	$5.065\cdot10^{-3}$	$7.263\cdot10^{-3}$	$6.283\cdot10^{-4}$	$1.279\cdot10^{-3}$	0.062	0.033	0.019	0.033	0.025
L^p-норма, $p = 2$	0	11.153	13.354	3.928	5.605	42.364	31.019	23.304	30.694	26.804
Лапласова середньоквадр. помилка, ***LMSE***	0	0.020	0.037	0.027	0.053	0.444	0.297	0.352	1.010	0.362
Відношення с/ш, ***SNR***	∞	197.423	137.690	$1.592\cdot10^{3}$	781.605	16.056	29.948	52.178	30.078	39.441
Макс. відношення с/ш, ***PSNR***	∞	522.782	364.608	$4.215\cdot10^{3}$	$2.070\cdot10^{3}$	36.232	67.583	119.735	69.020	90.507
Якість зображення, ***IF***	1	0.994935	0.992737	0.999372	0.998721	0.937717	0.966609	0.980835	0.966753	0.974646
Норм. взаємна кореляція, ***NC***	1	0.993261	0.986178	0.996790	0.994154	0.788233	0.863134	0.903486	0.873017	0.876281
Якість кореляції, ***CQ***	190.182	188.901	187.554	189.572	189.071	155.023	169.754	176.042	170.106	170.742
Структурний зміст, ***SC***	1	1.008484	1.020804	1.005826	1.010523	1.565560	1.316381	1.210451	1.283235	1.285486

Назва показника спотворення	*Оригінал*	*Методи приховування у частотній області*								
		Коха- Жао (**P** = 0.5)	*Коха- Жао* (**P** = 25)	*Бенгама-Мемона* (**P** = 1)	*Бенгама-Мемона* (**P** = 35)	*Хсу-Ву (а)*	*Хсу-Ву (б)*	*Фрідріх* (α = 0.1, γ = 1)	*Фрідріх* (α = 0.1, γ = 2)	*Фрідріх* (α = 0.2, γ = 1)
Загальне сигма-відношення с/ш, ***GSSNR***	∞	87.792	60.244	$4.388 \cdot 10^3$	$2.167 \cdot 10^3$	6.326	74.183	18.153	10.080	13.899
Сигма-відношення с/ш, ***SSNR***	∞	72.9	67.3	73.3	69.2	30.1	40.8	198.8	158.1	178.9
Нормоване відношення с/помилка, ***NSER***	256	100	99	127	116	114	41	184	233	190
Подібність гістограм, ***HS***	0	11096	10714	2446	2736	15960	12712	43062	50984	45806

Таблиця 5.5.

Показники візуального спотворення у випадку приховування даних методом розширення спектру

Назва показника спотворення	*Оригінал*	*РСПП-1*	*РСПП-2*
Максимальна абсолютна різниця, ***MD***	0	10	10
Середня абсолютна різниця, ***AD***	0	4.614	4.624
Нормована середня абсолютна різниця, ***NAD***	0	0.031	0.032
Середньоквадратична помилка, ***MSE***	0	31.721	31.817
Нормована середньоквадр. помилка, ***NMSE***	0	$1.101 \cdot 10^{-3}$	$1.104 \cdot 10^{-3}$
L^p-норма, при $p = 2$	0	5.632	5.641
Лапласова середньоквадр. помилка, ***LMSE***	0	0.113	0.102
Відношення сигнал/шум, ***SNR***	∞	908.381	905.628
Максимальне відношення сигнал/шум, ***PSNR***	∞	$2.050 \cdot 10^3$	$2.044 \cdot 10^3$
Якість зображення, ***IF***	1	0.998899	0.998896
Норм. взаємна кореляція, ***NC***	1	0.986079	0.986052

Назва показника спотворення	*Оригінал*	*РСПП-1*	*РСПП-2*
Якість кореляції, ***CQ***	196.672	193.934	193.929
Структурний зміст, ***SC***	1	1.027477	1.027529
Загальне сигма-відношення с/ш, ***GSSNR***	∞	568.540	563.058
Сигма-відношення сигнал/шум, ***SSNR***	∞	103.8	103.5
Нормоване відношення сигнал/помилка, ***NSER***	256	62	61
Подібність гістограм, ***HS***	0	5702	5736

Таблиця 5.6.

Показники звукового спотворення у випадку приховування даних в аудіосередовищі

Назва показника спотворення	*Оригінал*	*Методи приховування даних в аудіосередовищі*			
		НЗБ (ПВ інтервал)	*Фазове кодування*	*Розширення спектру*	*Ехо-кодування*
Макс. абсолютна різниця, ***MD***	0	1	1124	148	2963
Середня абсолютна різниця, ***AD***	0	$3.244 \cdot 10^{-3}$	15.298	14.321	572.093
Норм. середня абс. різниця, ***NAD***	0	$2.625 \cdot 10^{-6}$	0.012	0.012	0.5
Середньоквадр. помилка, ***MSE***	0	0.003	$2.657 \cdot 10^{3}$	956.742	$7.640 \cdot 10^{5}$
Нормована середньоквадратична помилка, ***NMSE***	0	$9.402 \cdot 10^{-10}$	$7.702 \cdot 10^{-4}$	$2.773 \cdot 10^{-4}$	0.250
L^p-норма, $p = 2$	0	0.057	51.551	30.931	874.01
Відношення сигнал/шум, ***SNR***	∞	$1.064 \cdot 10^{9}$	$1.298 \cdot 10^{3}$	$3.606 \cdot 10^{3}$	4.006
Макс. відношення с/шум, ***PSNR***	∞	$1.177 \cdot 10^{10}$	$1.437 \cdot 10^{4}$	$3.992 \cdot 10^{4}$	45.968

Назва показника спотворення	*Оригінал*	*Методи приховування даних в аудіосередовищі*			
		НЗБ (ПВ інтервал)	*Фазове кодування*	*Розширення спектру*	*Ехо-кодування*
Якість звучання, ***AF***	1	≈1	0.999230	0.999723	0.750371
Норм. взаємна кореляція, ***NC***	1	≈1	0.999615	0.999806	1.255078
Якість кореляції, ***CQ***	$2.792 \cdot 10^3$	$2.792 \cdot 10^3$	$2.790 \cdot 10^3$	$2.791 \cdot 10^3$	$3.355 \cdot 10^3$
Структурний зміст, ***SC***	1	≈1	≈1	1.000110	0.568251
Загальне сигма-відношення сигнал/шум, ***GSSNR***	∞	$1.152 \cdot 10^{14}$	$1.51 \cdot 10^{20}$	$3.279 \cdot 10^8$	9.377
Сигма-відношення сигнал/шум, ***SSNR***	∞	140.6	201.8	85.2	9.721
Відношення сигнал/помилка, ***SER***	∞	$1.064 \cdot 10^9$	$1.298 \cdot 10^3$	$3.606 \cdot 10^3$	4.006
Подібність гістограм, ***HS***	0	240	13854	14708	15548

ЗМІСТ

Арк.