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

ПРИХОВУВАННЯ ДАНИХ У ПРОСТОРОВІЙ ОБЛАСТІ НЕРУХОМИХ ЗОБРАЖЕНЬ ШЛЯХОМ МОДИФІКАЦІЇ НАЙМЕНШ ЗНАЧУЩОГО БІТА

Методичні рекомендації
до лабораторної роботи з дисципліни «Стеганографія»
для студентів спеціальності 125 «Кібербезпека»

Харків – 2019

УДК 004.415.24 (075.8)
П 77

Рецензенти:

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

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

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

П 77 **Приховування** даних у просторовій області нерухомих зображень шляхом модифікації найменш значущого біта : методичні рекомендації до лабораторної роботи з дисципліни «Стеганографія» для студентів спеціальності 125 «Кібербезпека» / уклад. О. О. Кузнецов, М. О. Полуяненко, Т. Ю. Кузнецова. – Харків : ХНУ імені В. Н. Каразіна, 2019. – 48 с.

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

УДК 004.415.24 (075.8)

ЗМІСТ

1. МЕТА І ЗАВДАННЯ ЛАБОРАТОРНОЇ РОБОТИ

Мета роботи: закріпити теоретичні знання за темою «Приховування даних у просторовій області нерухомих зображень шляхом модифікації найменш значущого біта даних (НЗБ, LSB – Least Significant Bit)», набути практичних вмінь та навичок щодо розробки стеганографічних систем, дослідити властивості стеганографічних методів, що засновані на низькорівневих властивостях зорової системи людини (ЗСЛ).

Лабораторна робота № 1 виконується у середовищі символьної математики MathCAD версії 12 або вище. Допускається виконання лабораторної роботи із використанням інших середовищ або мов програмування, що вивчалися студентами під час навчання.

Завдання лабораторної роботи

Завдання 1. Реалізація алгоритмів приховування і вилучення повідомлень методом заміни найменш значущого біта даних:

– реалізувати у середовищі символьної математики MathCAD (або в іншому середовищі / мові програмування) алгоритми приховування та вилучення даних у просторовій області зображень шляхом модифікації найменш значущого біта даних (методом LSB);

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

– виконати зорове порівняння пустого та заповненого контейнера та переконатися у відсутності помітних похибок;

– виконати вилучення вбудованого повідомлення, переконатися в його автентичності;

– отримати заповнені контейнери від інших груп розробників та переконатися у відсутності помітних похибок;

– вилучити повідомлення із отриманих стеганоконтейнерів інших груп розробників та переконатися у їхній автентичності.

Завдання 2. Експериментальні дослідження зорового порогу чутливості людини до змін яскравості зображень:

– виконати приховування даних у різні за значущістю біти зображення, починаючи з найменш значущого;

– експериментально встановити, модифікація яких бітів зображення не призводить до помітних похибок;

– розрахувати за отриманими емпіричними даними зоровий поріг чутливості до змін яскравості нерухомих зображень.

Завдання 3. Реалізація алгоритмів приховування і вилучення повідомлень методом псевдовипадкової перестановки:

– реалізувати у середовищі символьної математики MathCAD (або в іншому середовищі / мові програмування) алгоритм приховування та

вилучення даних у просторовій області зображень методом псевдовипадкової перестановки (методом ПВП);

– застосовуючи розроблену програмну реалізацію, виконати стеганографічне кодування інформаційного повідомлення (сформувати стеганограму). Вилучити вбудоване повідомлення, переконатися в його автентичності;

– ввести інший секретний ключ (таблицю псевдовипадкової перестановки) та спробувати вилучити інформаційне повідомлення з контейнера, переконатися в спотворенні інформації. Розрахувати кількість можливих секретних ключів та ймовірність вгадування секретного ключа зловмисником.

Завдання 4. Реалізація алгоритмів приховування і вилучення повідомлень методом псевдовипадкового інтервалу:

– реалізувати у середовищі символьної математики MathCAD (або в іншому середовищі / мові програмування) алгоритм приховування та вилучення даних у просторовій області зображень методом псевдовипадкового інтервалу (методом ПВІ);

– застосовуючи розроблену програмну реалізацію, сформувати стеганограму, вилучити вбудоване повідомлення, переконатися в його автентичності;

– ввести інший секретний ключ (який задає правило формування псевдовипадкового інтервалу) та спробувати вилучити інформаційне повідомлення з контейнера, переконатися в спотворенні інформації. Розрахувати кількість можливих секретних ключів та ймовірність вгадування секретного ключа зловмисником.

2. МЕТОДИЧНІ ВКАЗІВКИ З ОРГАНІЗАЦІЇ САМОСТІЙНОЇ РОБОТИ

1. Вивчити теоретичний матеріал лекції «Приховування даних у просторовій області зображень шляхом модифікації найменш значущого біта даних».

2. Вивчити матеріал основного джерела літератури (Конахович Г. Ф., Пузыренко А. Ю. Компьютерная стеганография):

- особливості зорової системи людини (ст. 73–75);
- приховування даних у просторовій області зображень (ст. 76);
- метод заміни найменш значущого біта (ст. 76–89);
- метод псевдовипадкового інтервалу (ст. 89–92);
- метод псевдовипадкової перестановки (ст. 92–96).

3. Вивчити основні команди у середовищі символьної математики MathCAD для роботи із зображеннями.

4. Підготувати відповіді на контрольні запитання.

5. Підготувати звіт з лабораторної роботи.

Допуск до виконання лабораторної роботи здійснюється за результатами письмового опитування (контрольної роботи).

3. ЗАГАЛЬНОТЕОРЕТИЧНІ ПОЛОЖЕННЯ ЗА ТЕМОЮ ЛАБОРАТОРНОЇ РОБОТИ

Властивості зорової системи людини, що використовуються в стеганографії

Властивості зорової системи людини (ЗСЛ), що використовуються в стеганографії, можна розділити на дві основні групи: *низькорівневі* («фізіологічні») і *високорівневі* («психофізіологічні»).

Виділяють три найважливіші *низькорівневі властивості,* що впливають на помітність стороннього шуму в зображенні:

– мала чутливість до незначних змін яскравості зображення;

– мала чутливість до незначних змін контрастності зображення;

– частотна чутливість;

– ефект маскування;

– мала чутливість до незначних змін яскравості каналу синього кольору зображення.

Розглянемо ці властивості більш докладно.

Чутливість до зміни яскравості. Здатність ока людини реагувати на світлове роздратування характеризується *чутливістю*. Чутливість ока до дії випромінювання визначається величиною, яка є зворотною до яскравості $L_П$, що викликає граничне роздратування: $v = 1/L_П$. Чутливість може виражатися і в одиницях, зворотних до граничної освітленості спостережуваного зображення. Дослідження показали, що поріг чутливості ЗСЛ до зміни яскравості дорівнює 2–3 %. Тобто, наприклад, якщо яскравість зображення у цифровому вигляді кодується цілими числами в діапазоні $0 \ldots 255$ (всього $L_m = 256$ рівнів квантування), тоді зміна яскравості на $\Delta L = 8$ рівнів змінить освітленість на

$$\Delta = \frac{\Delta \mathrm{L}}{\mathrm{L_m}} 100\% = \frac{8}{256} 100\% \approx 3\%$$

відносно граничної освітленості і ця зміна не призведе до видимих викривлень зображення.

Чутливість до зміни контрастності. Якщо на око людини впливає зображення з яскравістю ділянок L, тоді спостерігач реагує не тільки на абсолютну зміну яскравості ΔL, а і на її відносне значення $\Delta L/L$. Мінімальна відносна зміна яскравості $\Delta L/L$, що сприймається спостерігачем, називається *відносним різницевим порогом роздратування*.

На практиці вважають, що поріг чутливості ЗСЛ до незначних змін контрастності становить 2 ... 4 %, тобто спотворення, що вносяться, з відношенням $\Delta L/L < 0,02 \ldots 0,04$ ЗСЛ не сприймає. І навпаки, якщо спотворення

зображень приводять до $\Delta L/L > 0,04$, такі спотворення людина помітить. Оцінка $0,002\ldots0,004$ отримана емпіричним шляхом, тобто вона може бути відмінною для різних довжин хвиль (різних колірних складових зображення) і є індивідуальною характеристикою органів зору конкретної людини. Наприклад, якщо як зображення використовувати *.bmp файл з кодуванням кожного пікселя 24 байтами (по 8 біт на кожен канал червоного, зеленого і синього кольору), а відповідний поріг чутливості ЗСЛ до зміни контрастності (відносної зміни яскравості до граничної освітленості) становитиме:

$$\Delta L/L = \Delta L/256 = 0,02\ldots0,04 \quad \Delta\,\mathrm{L/L} = \Delta\,\mathrm{L/256} = 0{,}02\ldots0{,}04,$$

тоді при зміні $\Delta L = 5\ldots10$ в кодуванні файла *.bmp ЗСЛ спотворень не сприйме.

Частотна чутливість ЗСЛ проявляється в тому, що людина набагато більш сприйнятлива до низькочастотного (НЧ), ніж до високочастотного (ВЧ) шуму. Це пов'язано з нерівномірністю амплітудно-частотної характеристики системи зору людини.

Ефект маскування. Елементи ЗСЛ розділяють вступний відеосигнал на окремі компоненти. Кожна складова збуджує нервові закінчення ока через низку підканалів. Виокремлювані оком компоненти мають різні просторові й частотні характеристики, а також різну орієнтацію (горизонтальну, вертикальну, діагональну). У випадку одночасного впливу на око двох компонентів з подібними характеристиками збуджуються ті самі підканали. Це призводить до ефекту маскування, що полягає в збільшенні порога виявлення відеосигналу в присутності іншого сигналу, що володіє аналогічними характеристиками. Тому адитивний шум набагато помітніший на гладких ділянках зображення, ніж на високочастотних, тобто в останньому випадку спостерігається маскування. Найбільш сильно ефект маскування проявляється, коли обидва сигнали мають однакову орієнтацію й місце розташування.

Чутливість до зміни каналу синього кольору. Ще одним відомим феноменом ЗСЛ є її чутливість до змін колірових каналів. Якщо проаналізувати криві залежностей спектральної чутливості людського ока до потоку світлового випромінювання, можна помітити, що людина дуже добре сприймає зелені і зелено-жовті кольори, тоді як її чутливість до синіх кольорів помітно нижча. Крім того, очному кришталику важче фокусуватися на предметах, якщо вони забарвлені в синьо-фіолетові тони. Це пояснюється падінням спектральної чутливості ока в цих ділянках спектра. Тому окуляри іноді роблять не нейтрально-прозорими, а із забарвленого в жовтий або коричневий колір скла, яке фільтрує синьо-фіолетову складову спектра.

Існує припущення, що знижена чутливість ЗСЛ до змін у каналах синього кольору пов'язана з переважанням у природі предметів (об'єктів), що мають зелений колір, і практично повною відсутністю суто строго синіх

кольору. На практиці, різна чутливість ЗСЛ до кольорових складових растрових даних виражається в оцінці повнокольорової яскравості пікселя

$$Y = 0{,}58662 \cdot G + 0{,}2989 \cdot R + 0{,}11448 \cdot B,$$

тобто внесок каналу зеленого кольору в сприйману яскравість пікселя становить близько 60 %, відповідний внесок червоного кольору – близько 30 %, і лише близько 10 % – це внесок каналу синього кольору. Таким чином, будемо вважати, що чутливість ЗСЛ до зміни синього кольору приблизно в 6 разів нижче, ніж до зеленого і в 3 рази нижче, ніж до червоного.

Високорівневі властивості ЗСЛ поки що рідко враховуються при побудові стеганоалгоритмів. Вони відрізняються від низькорівневих тим, що виявляються «повторно» – обробивши первинну інформацію від ЗСЛ, мозок видає команди на «підстроювання» зорової системи під зображення.

Перерахуємо основні з цих властивостей:

– чутливість до контрасту – висококонтрастні ділянки зображення і перепади яскравості привертають до себе більше уваги;

– чутливість до розміру – великі ділянки зображення «помітніші» порівняно з меншими за розміром, причому існує поріг насиченості, коли подальше збільшення розміру не відіграє ролі;

– чутливість до форми – довгі і тонкі об’єкти викликають більше уваги, ніж заокруглені й однорідні;

– чутливість до кольору – деякі кольори (наприклад, червоний) «помітніші», ніж інші; цей ефект посилюється, якщо фон відрізняється від кольору фігур на ньому;

– чутливість до місця розміщення – людина схильна в першу чергу розглядати центр зображення; також уважніше розглядаються фігури переднього плану, ніж заднього;

– чутливість до зовнішніх подразників – рух очей спостерігачів залежить від конкретних умов, отриманих перед переглядом або під час його інструкцій чи додаткової інформації.

Високорівневі властивості ЗСЛ часто використовують у рекламі, в політиці, в різних шоу тощо, бо це дієвий спосіб впливу на свідомість людини за допомогою різних особливостей розумової діяльності, зумовлених загальним рівнем культури, освіти, національністю, професійною приналежністю, традиціями, менталітетом тощо.

Цифровий формат кодування зображень Bitmap Picture

Формат Windows BMP є одним із вбудованих форматів зображень в операційних системах Microsoft Windows. Він підтримує зображення з 1, 4, 8, 16, 24 і 32 бітами на піксель, хоча файли BMP з 16 і 32 бітами на піксель використовуються рідко. Для зображень з 4 і 8 бітами на піксель

формат BMP підтримує також просте RLE-стиснення (кодування довжин серій). Проте стиснення в BMP-форматі має ефект тільки за наявності в зображенні великих ділянок однакового кольору, що обмежує цінність вбудованого алгоритму стиснення.

Розглянемо структуру BMP-файла (тут і далі будемо розглядати тільки файли BMP-24 із кодуванням 24 бітів на піксель). Він містить точкове (растрове) зображення і складається із трьох основних розділів: заголовка файла, заголовка растра та растрових даних.

Заголовок файла містить інформацію про файл (його тип, обсяг і т. п.). До заголовка растра винесено інформацію про ширину й висоту зображення, кількість бітів на піксель, розмір растра, глибину кольору, коефіцієнт компресії і тощо. В першу чергу нас цікавлять растрові дані – інформація про колір кожного пікселя зображення.

Колір пікселя визначається об'єднанням трьох основних колірних складових: червоної, зеленої та синьої (скорочено *RGB*). Кожній з них відповідає певне значення інтенсивності, що може змінюватися від 0 до 255. Отже, за кожний з колірних каналів відповідає 8 бітів (1 байт), а глибина кольору зображення в цілому – 24 біта (3 байти).

Для обробки зображення необхідно перевести колірні характеристики кожного його пікселя в числову матрицю, що є масивом, який складається з трьох підмасивів розкладу кольорового зображення на компоненти R, G і B. При цьому три колірні компоненти розміщуються один за одним у загальному масиві C (див. рис. 1).

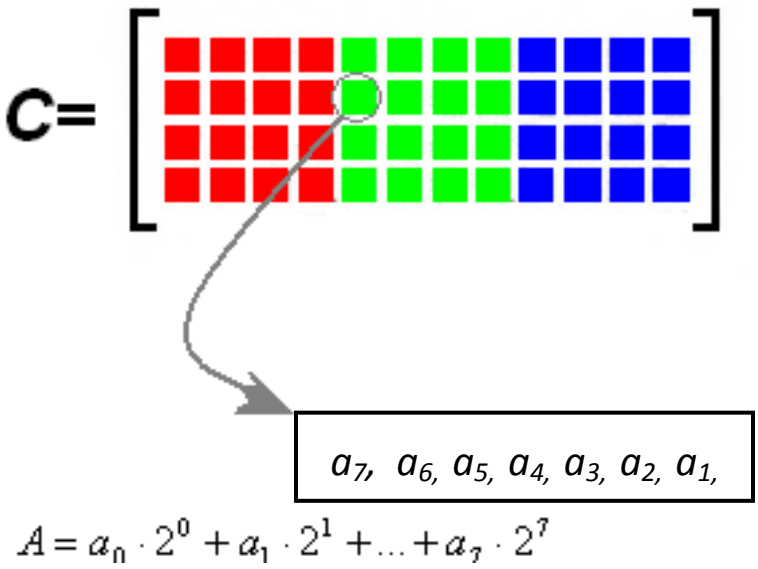


Рис. 1. Уявлення компонент кольоровості R, G і B у вигляді підмасивів масиву C

Таким чином, одна точка зображення у форматі BMP-24 кодується трьома байтами, кожний з яких відповідає за інтенсивність одного з трьох складових кольорів (див. рис. 2).

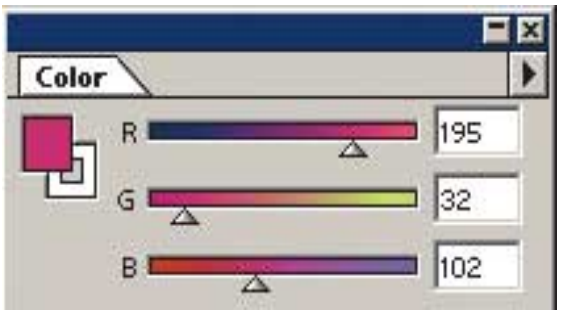


Рис. 2. Приклад кодування растрових даних зображення

1	1	0	0	0	0	1	1	R — 195
0	0	1	0	0	0	0	0	G — 32
0	1	1	0	0	1	1	0	B — 102

Рис. 3. Приклад опису трьох байтів у бітовому вигляді

В результаті змішення кольорів з червоного (R), зеленого (G) і синього (B) каналів піксель одержує потрібний відтінок.

Для більше наочної демонстрації принципу дії методу LSB, розпишемо кожний з трьох байтів у бітовий вигляд (рис. 3).

Молодші розряди (на рис. 3 вони розташовані правруч) меншою мірою впливають на підсумкове зображення, ніж старші. З цього можна зробити висновок, що заміна одного або декількох молодших бітів на інші біти тільки трохи спотворить відтінок пікселя і спостерігач не помітить зміни.

Метод заміни найменш значущого біта

Метод заміни найменш значущого біта (НЗБ, LSB – Least Significant Bit) найпоширеніший серед методів заміни в просторовій області зображення. Цей метод використовує першу низькорівневу властивість ЗСЛ – слабку чутливість до незначних змін яскравості зображення.

Молодший значущий біт при кодуванні яскравості пікселя несе в собі найменше інформації. Тобто людина в більшості випадків не здатна помітити змін у цьому біті. Фактично, НЗБ – це шум, тому його можна використовувати для вбудовування інформації шляхом заміни менш значущих бітів пікселів зображення бітами секретного повідомлення. При цьому, для зображення в градаціях сірого (кожний піксель зображення кодується одним байтом) обсяг вбудованих даних може становити 1/8 від загального обсягу контейнера. Наприклад, у зображення розміром 512×512 пікселів у форматі BMP-24 можна вмонтувати ~32 кбайт інформації. Якщо ж модифікувати два молодших біти (що також практично непомітно), то пропускну здатність такого стегаканалу можна збільшити вдвічі.

Популярність цього методу обумовлена його простотою й тим, що він дозволяє приховувати у відносно невеликих файлах досить великі обсяги інформації (пропускна спроможність створюваного прихованого каналу зв’язку становить при цьому від 12,5 до 30 %). Метод найчастіше працює з растровими зображеннями, представленими у форматі без компресії (наприклад, GIF і BMP).

Метод НЗБ має дуже низьку стеганографічну стійкість до атак пасивного й активного порушників. Основний недолік – висока чутливість до найменших спотворень контейнера. Для ослаблення цієї чутливості часто додатково застосовують завадостійке кодування.

Так, якщо інформаційне повідомлення вбудовується у зображення у форматі BMP-24 із використанням тільки LSB, маємо (у відсотках до максимального рівня яскравості):

$$\Delta = \frac{2^0}{2^8}100\,\% = \frac{1}{256}100\,\% < 0,4\,\% .$$

При вбудовуванні в перші три найменш значущі біти маємо:

$$\Delta = \frac{2^0 + 2^1 + 2^2}{2^8}100\,\% = \frac{7}{256}100\,\% < 3\,\% .$$

Отже, використання перших трьох бітів призводить до внесення похибок, що лежать нижче порогу ЗСЛ до змін яскравості (3 %).

Слід вказати на переваги та недоліки методу LSB.

Переваги:

1. Висока пропускна спроможність створюваного стегаканалу. Фактично мова йде щонайменше про 1/8 обсягу контейнера. Збільшення кількості бітів, що використовуються при вбудовуванні, веде до збільшення пропускної спроможності стегаканалу (до 2/8=1/4, або навіть до 3/8).

2. Простота практичної реалізації та обумовлена цим велика швидкість перетворення як при вбудовуванні, так і при видобуванні бітів інформації.

Недоліки:

1. Відсутність секретного ключа обумовлює дуже малу стійкість до атак зловмисників. Фактично, супротивник може зчитувати інформаційні повідомлення з LSB без будь-яких завад.

2. Дуже низька стійкість до детектування повідомлень супротивником. Наприклад, найпростіший статистичний тест LSB заповненого контейнера дає змогу супротивнику встановити факт вбудовування інформації.

3. Дуже низька стійкість до геометричних (повороти, масштабування, зміна пропорцій) атак та атак стиснення.

4. Псевдовипадкові зміни LSB контейнера або їх обнуління гарантовано руйнують вбудоване повідомлення.

Подальшим розвитком методу LSB є методи псевдовипадкового інтервалу та псевдовипадкового переставлення.

Метод псевдовипадкового інтервалу

У розглянутому вище простому прикладі виконується заміна найменш значущого біта всіх послідовно розміщених пікселів зображення, що спрощує атаки зловмисників та знижує ефективність стегаканалу. Інший підхід – метод випадкового інтервалу, що полягає у випадковому

розподілі бітів секретного повідомлення по контейнеру, внаслідок чого відстань між двома вбудованими бітами визначається псевдовипадково. Ця методика особливо ефективна у разі, коли бітова довжина секретного повідомлення істотно менша за кількість пікселів зображення.

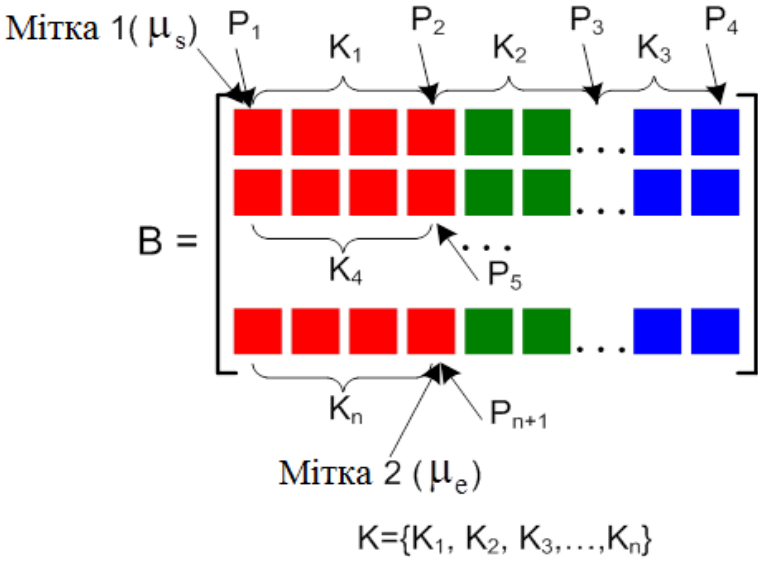


Рис. 4. Приклад використання підмасиву B синього колірного компоненту зображення як контейнера

Розглянемо простий випадок цього методу, коли інтервал між двома послідовними вбудовуваннями бітів повідомлення задається значенням секретного ключа $K = \{K_1, K_2, \ldots, K_n\}$.

Нехай M – повідомлення, що необхідно приховати. Як контейнер C використаємо підмасив B синього колірного компоненту зображення (див. рис. 4).

Визначимо мітки, що встановлюватимуть межі корисного повідомлення в контейнері. На відміну від попереднього методу, стартова мітка визначатиме порядковий номер елементу контейнера, починаючи з якого в останній заноситимуться дані. Нехай μ_S буде початковою міткою, а μ_e – міткою, що сигналізує про завершення корисної частини серед добутих символів.

Приймемо, що під час внесення бітів повідомлення в контейнер зі змінним кроком величина останнього обумовлена значенням секретного ключа K_i, i=1…n, де i – номер вбудованого біта.

Обмежуючу мітку μ_e додамо до тексту повідомлення, яке підлягає прихованню.

Кожен символ повідомлення переводимо в двійковий формат, кожен розряд якого записується замість наймолодшого біта числа P_i, що відповідає значенню інтенсивності синього кольору певного пікселя. При цьому елементи масиву контейнера перебираються не послідовно, а зі змінним кроком, що задається значенням секретного ключа.

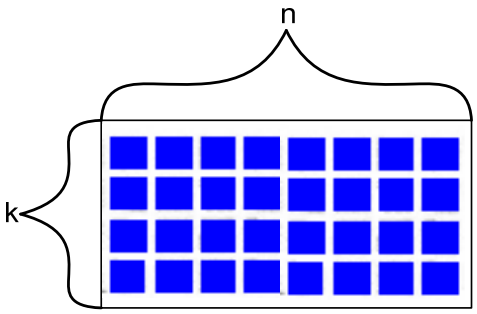
Стартовий елемент задається міткою μ_s. Після проведеної зміни модифіковане двійкове число P_i переводиться у формат десяткового і записується у відповідну позицію масиву B^*, який на початку був прийнятий рівним масиву B.

Результуюче кольорове зображення визначатиметься масивом об’єднання колірних масивів R, G, B^*.

При отриманні прихованого повідомлення слід знати параметри μ^*_s, μ^*_e, K^* і, зрозуміло, масив B^*, що, як передбачається, містить приховані дані.

Отримання повідомлення з масиву B^* виконується в зворотному по відношенню до операції вбудовування порядку, після чого одержуємо вектор інформаційних бітів даних. З одержаного вектора шляхом порівняння з мітками μ_s та μ_e виділеного фрагмента вилучається корисне повідомлення.

Найпростіше реалізувати розглянутий метод можна у такий спосіб. Нехай, наприклад, $M = \{m_0, m_1, m_2, \ldots m_n\}$ – інформаційне повідомлення із n бітів, тобто $m_i = \begin{cases} 0 \\ 1 \end{cases}$.



Як контейнер будемо використовувати масив даних розміром n стовпців та k рядків (одного з кольорів).

Як секретний ключ будемо використовувати вектор $K = \{k_0, k_1, \ldots k_{n-1}\}$, де K_i – деяке число, що лежить у діапазоні $[0 \ldots k]$.

Таким чином, вбудовування біта m_0 будемо виконувати у стовпець під номером N = 0, біта m_1 – у стовпець під номером 1 і т. д. Біт m_0 записується у LSB байта контейнера, у рядку k_0, m_1 – у рядку k_1 і т. п. Схема вбудовування зображення представлена на рис. 5.

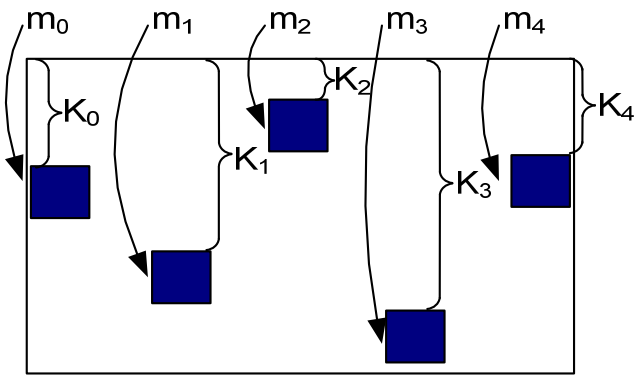


Рис. 5. Схема вбудовування зображення

Таким чином, один біт інформації записується у один стовпець контейнера у рядок, що задається секретним ключем.

Розглянутий метод має такі *переваги:*

1. Введення секретного ключа, який задає правило вбудовування інформації, значно підвищує стійкість методу до детектування та вставки інформаційних даних. Звісно, статистика інформаційних бітів даних дещо «просочується» у статистику LSB контейнера, але рознесення їх по контейнеру через псевдовипадковий інтервал значно ускладнює детектування повідомлення.

2. Простота реалізації методу та обумовлена цим велика швидкість перетворення як при вбудовуванні, так і при видобуванні бітів інформації.

Недоліки:

1. Значно зменшується пропускна спроможність стегаканалу. В середньому пропускна спроможність зменшується у N разів, де

$$N = \frac{1}{n}\sum_{i=1}^{n} K_i$$ – середнє значення псевдовипадкового інтервалу.

2. Дуже низька стійкість до геометричних (повороти, масштабування, зміна пропорцій) атак та атак стиснення.

3. Псевдовипадкові зміни контейнера руйнують вбудоване повідомлення.

Для подальшого підвищення стійкості до негативних дій зловмисників вбудовування інформаційного повідомлення доцільно попередньо шифрувати. Тому наступний метод – метод псевдовипадкової перестановки, саме і побудований на основі використання найпростішого перестановочного шифру.

Метод псевдовипадкової перестановки

Недоліком методу псевдовипадкового інтервалу є те, що біти повідомлення в контейнері розміщені в тій же послідовності, що і в самому повідомленні. І лише інтервал між ними змінюється псевдовипадково. Тому для контейнерів фіксованого розміру доцільнішим є використання методу псевдовипадкової перестановки. Сутність його полягає у використанні переставного шифру для попередньої обробки інформаційних бітів даних.

У переставному шифрі змінюється не відкритий текст, що підлягає шифруванню, а порядок символів. Наприклад, у простому стовпцевому переставному шифрі відкритий текст пишеться горизонтально на розграфленому аркуші паперу фіксованої ширини, а шифротекст прочитується по вертикалі. Розшифруванням є запис шифротексту вертикально на аркуші розграфленого паперу фіксованої ширини і потім зчитування відкритого тексту горизонтальне (див. табл. 1.1).

Таблиця 1.1

Простий стовпцевий переставний шифр

Відкритий текст:
ЦЕПРИКЛАДПРОСТОГОСТОВПЦЕВОГОПЕРЕСТАВНОГОШИФРУАААА

Ц	Е	П	Р	И	К	Л
А	Д	П	Р	О	С	Т
О	Г	О	С	Т	О	В
П	Ц	Е	В	О	Г	О
П	Е	Р	Е	С	Т	А
В	Н	О	Г	О	Ш	И
Ф	Р	У	А	А	А	А

Шифрограма:
ЦАОБОАИЕДГЦПВФППОЬЕНРРРСОРОУИОТВЕГАКСООСОАЛТВГТША

Оскільки символи шифротексту ті ж, що і у відкритому тексті, частотний аналіз шифротексту покаже, що кожна буква зустрічається приблизно з тією ж частотою, що і звичайно. Це дасть криптоаналітику можливість застосувати різні методи визначенно правильного порядку символів для отримання відкритого тексту. Застосування до шифротексту другого переставного фільтру значно підвищить безпеку. Існують і ще складніші переставні фільтри, але комп'ютери можуть розкрити майже всі з них.

Німецький шифр ADFCVX, використаний під час Першої світової війни, був переставним фільтром у поєднанні з простою підстановкою. Цей для свого часу дуже складний алгоритм був розкритий Жоржем Пенвеном (Georges Painvin), французьким криптоаналітиком.

Хоча багато сучасних алгоритмів використовують перестановку, з цим пов'язана проблема використання великого обсягу пам'яті, а також необхідність роботи з повідомленнями певного розміру.

Формалізуємо переставний шифр, стосовно використання його для стеганографічного перетворення цифрових повідомлень.

Нехай $m = \{m_0, m_1, \ldots m_{N\text{-}1}\}$ – інформаційне повідомлення із N бітів. Розіб'ємо його на блоки по n бітів, отримаємо

$$m = \left\{ M_0, M_1, \ldots M_{\frac{N}{n}-1} \right\}, \text{ де } M_i = \{m_{n\cdot i+0}, m_{n\cdot i+1}, m_{n\cdot i+2}, \ldots m_{n\cdot i+n-1}\}.$$

Тобто $m = \left\{ \underbrace{m_0, m_1, m_2, \ldots m_{n-1}}_{M_0}, \underbrace{m_n, m_{n+1}}_{M_1}, \underbrace{\ldots m_{N-1}}_{M_{N/n-1}} \right\}$.

Як ключ будемо використовувати переставну матрицю P розміром $n \times n$ двійкових елементів, причому у кожному стовпці та у кожному рядку матриці є тільки одна «1», решта заповнюється «0». Переставна матриця P задає правило псевдовипадкового перестановки:

$$M_i^* = M_i \cdot P = \{m_{n\cdot i+0}, m_{n\cdot i+1}, m_{n\cdot i+2}, \ldots m_{n\cdot i+n-1}\} \times [P] = \\ = \{m^*_{n\cdot i+0}, m^*_{n\cdot i+1}, m^*_{n\cdot i+2}, \ldots m^*_{n\cdot i+n-1}\}.$$

Для фіксованого значення n існують n! різних переставних матриць, кожна з яких задає своє правило перестановки. Після перестановки сформуємо масив:

$$m^* = \{m^*_0, m^*_1, \ldots m^*_{N-1}\}.$$

Для виконання зворотного перетворення треба обчислити

$$M_i = M_i^* \cdot P^{-1},$$

де P^{-1} – матриця, зворотна матриці P, тобто $P^{-1} \cdot P = I$.

Але для переставної матриці можна записати $P^{-1} = P^T$, отже:

$$M_i = M_i^* \cdot P^T.$$

Наведемо приклад:

$$m = \{1\,0\,1\,0\,1\,1\,0\,1\,0\,1\,1\,0\,1\,1\,0\},$$

$$n = 5,\ P = \begin{vmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{vmatrix}.$$

Маємо таке:

$$M_0 = \{1\,0\,1\,0\,1\},\qquad M^*_0 = M_0 \cdot P = \{0\,1\,0\,1\,1\},$$
$$M_1 = \{1\,0\,1\,0\,1\},\qquad M^*_1 = M_1 \cdot P = \{0\,1\,0\,1\,1\},$$
$$M_2 = \{1\,0\,1\,0\,1\},\qquad M^*_2 = M_2 \cdot P = \{0\,1\,1\,0\,1\}.$$

Отже,

$$m^* = \{0\,1\,0\,1\,1\,0\,1\,0\,1\,1\,0\,1\,1\,0\,1\}.$$

Для зворотного перетворення виконаємо

$$M_0 = M^*_0 \cdot P^T = \{0\,1\,0\,1\,1\} \cdot \begin{vmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{vmatrix} = \{1\,0\,1\,0\,1\},$$

$$M_1 = M^*_1 \cdot P^T = \{0\,1\,0\,1\,1\} \cdot \begin{vmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{vmatrix} = \{1\,0\,1\,0\,1\}.$$

$$M_2 = M^*_2 \cdot P^T = \{0\,1\,1\,0\,1\} \cdot \begin{vmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{vmatrix} = \{1\,0\,1\,1\,0\}.$$

Отже, маємо

$$m = \{1\,0\,1\,0\,1\,1\,0\,1\,0\,1\,1\,0\,1\,1\,0\}.$$

Розглянемо тепер використання переставного шифру в методі псевдовипадкової перестановки. Загальна схема вбудовування бітів даних зображена на рис. 6.

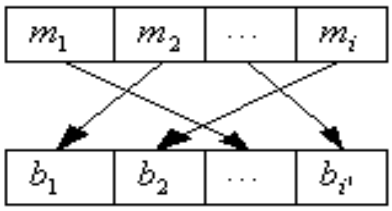
Сутність методу псевдовипадкової перестановки полягає у попередній обробці інформаційних даних переставним шифруванням (за розглянутою вище схемою) та вбудовуванні отриманих бітів даних у біти контейнера за допомогою методу LSB або псевдовипадкового інтервалу.

Вочевидь, додаткове шифрування інформаційних бітів контейнера поліпшує властивості стегасистеми, отже, маємо такі *переваги:*

1. Підвищення стійкості до детектування повідомлення зловмисниками. Використання шифру знижує статистику вбудованих бітів даних, отже «зашумляє» відповідні LSB. Виявляти таку повідомлення дуже складно.

2. Пропускна спроможність не змінюється, тобто при вбудовуванні у всі LSB контейнера пропускна спроможність дуже висока (1/8, 2/8 – 1/4, або навіть 3/8).

$$m=\{m_1,m_2..m_i\}=\{m_1,m_2..m_n\}\{m_{n+1},m_{n+2}..m_{2n}\}\{m_{2n+1},m_{2n+2}..m_i\}$$



$$b=\{b_1,b_2..b_{i'}\}$$

Рис. 6. Приклад здійснення псевдовипадкової перестановки

На жаль, метод псевдовипадкової перестановки має такі *недоліки:*

1. Дуже низька стійкість до геометричних (повороти, масштабування, зміна пропорцій) атак та атак стиснення.

2. Псевдовипадкові зміни LSB контейнера або їх обнуління руйнують вбудоване повідомлення.

4. ПИТАННЯ ДЛЯ ПОТОЧНОГО КОНТРОЛЮ ПІДГОТОВКИ СТУДЕНТІВ ДО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ

1. Математична модель та структурна схема криптографічної (секретної) та стеганографічної системи захисту інформації.
2. Основні галузі практичного використання стеганографічних методів захисту інформації. Історичні приклади використання стеганографічних систем захисту інформації.
3. Класифікація стеганографічних систем. Закриті, напівзакриті, відкриті стеганографічні системи. Хиткі, напівхиткі, робастні стеганографічні системи.
4. Класифікація та види контейнерів за різними ознаками. Приклади використання різних контейнерів для організації прихованої передачі інформації.
5. Низькорівневі властивості зорової системи людини. Практичні приклади використання низькорівневих властивостей зорової системи людини в стеганографії.
6. Високорівневі властивості зорової системи людини. Практичні приклади використання високорівневих властивостей зорової системи людини в стеганографії.
7. Сучасні формати нерухомих зображень. Структура файла нерухомого зображення у форматі bmp24. Растрові дані зображення.
8. Метод вбудовування інформації в нерухомі зображення на основі модифікації найменш значущого біта (метод LSB). Переваги та недоліки.
9. Найпростіші симетричні шифри. Простий перестановочний шифр.
10. Метод вбудовування інформації з використанням псевдовипадкової перестановки (метод ПВП). Переваги та недоліки.
11. Метод вбудовування інформації з використанням псевдовипадкового інтервалу (метод ПВІ). Переваги та недоліки.

5. ІНСТРУКЦІЯ ДО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ

Завдання 1. Реалізація алгоритмів вбудовування та вилучення повідомлень у просторовій області нерухомих зображень методом LSB

1.1. Завантажуємо вихідні дані: контейнер – нерухоме зображення (в форматі *.bmp24); інформаційне повідомлення – текстовий документ (у форматі *.txt). Для цього в середовищі MathCAD виконуємо такі дії.

1.1.1. Виконуємо команду читання растрових даних нерухомого зображення з заданого файла у вигляді двовимірного масиву цілих чисел:

«C:= READRGB(“[ім’я файла].bmp”)».

Елементи масиву $C_{i,j}$ знаходяться в інтервалі [0...255] і задають значення яскравості одного з трьох кольорів (червоного, зеленого або синього) зображення. Координати елементів масиву задають розташування окремих пікселів зображення. Масив С складається з трьох підмасивів рівного розміру (для зберігання значень яскравості відповідних кольорів).

Наприклад, нехай як контейнер виступає нерухоме зображення, що зберігається в файлі з ім’ям «1.bmp». Тоді після виконання команди читання растрових даних

«C:=READRGB(“1.bmp”)»

маємо:

C =

	0	1	2	3	4	5	6	7	8	9
0	86	79	72	72	72	69	71	74	77	85
1	110	97	90	86	78	71	70	70	77	79
2	132	120	112	105	96	88	81	83	80	80
3	122	116	105	104	103	102	101	99	93	90
4	131	122	117	118	118	116	105	107	105	110
5	147	147	148	148	150	153	145	135	127	120
6	169	164	167	170	173	175	164	155	152	144
7	189	195	193	189	183	172	173	173	177	168
8	191	192	194	199	194	187	182	182	183	182
9	186	188	194	198	192	187	187	180	175	177
10	195	196	199	200	201	190	192	186	174	167
11	185	189	202	203	203	199	203	199	200	199
12	192	196	198	199	204	202	206	199	194	197
13	177	185	187	186	180	178	179	177	175	180
14	173	176	174	166	165	165	163	161	158	162
15	160	162	158	153	156	158	157	156	153	...

Для графічного відображення завантажених даних виконуємо дії: «Вставити», «Зображення». В полі введення джерела зображень вносимо ім’я файла або ім’я змінної, в якій зберігається масив даних.

Після виконання команд графічного відображення контейнера для розглянутого прикладу маємо такі зображення:

"1.bmp"

C

Графічне відображення масиву растрових даних C складається з трьох фрагментів зображення, кожен фрагмент відповідає відображенню одного з кольорових каналів (червоного, зеленого або синього) в градаціях сірого кольору.

Виконуємо команду читання даних з каналу червоного кольору растрових даних нерухомого зображення з заданого файла у вигляді масиву цілих чисел:

«R:=READ_RED(“[ім’я файла].bmp”)».

Елементи масиву $R_{i,j}$ також знаходяться в інтервалі [0...255] і задають значення яскравості червоного кольору. Для графічного відображення завантажених даних виконуємо дії: «Вставити», «Зображення». В полі введення джерела зображень вносимо ім’я змінної R, в якій зберігається масив даних каналу червоного кольору.

Для розглянутого прикладу виконуємо команду

«R:=READ_RED(“1.bmp”)»,

після чого отримуємо:

R =

	0	1	2	3	4	5	6	7	8	9
0	86	79	72	72	72	69	71	74	77	85
1	110	97	90	86	78	71	70	70	77	79
2	132	120	112	105	96	88	81	83	80	80
3	122	116	105	104	103	102	101	99	93	90
4	131	122	117	118	118	116	105	107	105	110
5	147	147	148	148	150	153	145	135	127	120
6	169	164	167	170	173	175	164	155	152	144
7	189	195	193	189	183	172	173	173	177	168
8	191	192	194	199	194	187	182	182	183	182
9	186	188	194	198	192	187	187	180	175	177
10	195	196	199	200	201	190	192	186	174	167
11	185	189	202	203	203	199	203	199	200	199
12	192	196	198	199	204	202	206	199	194	197
13	177	185	187	186	180	178	179	177	175	180
14	173	176	174	166	165	165	163	161	158	162
15	160	162	158	153	156	158	157	156	153	...

R

Виконуємо аналогічні команди читання даних з каналів зеленого і синього кольору растрових даних нерухомого зображення у вигляді масивів цілих чисел:

«G:=READ_ GREEN (“[ім’я файла].bmp”)»,
«G:=READ_BLUE(“[ім’я файла].bmp”)».

Для розглянутого прикладу виконуємо команди

«B:=READ_ GREEN (“1.bmp”)»,
«B:=READ_BLUE(“1.bmp”)»,

після чого отримуємо:

G *B*

G =

	0	1	2	3	4	5	6	7	8	9
0	91	83	79	77	75	73	73	75	83	90
1	112	101	93	89	82	76	76	76	81	82
2	134	122	118	107	103	90	88	90	86	85
3	124	118	109	107	109	104	103	101	98	94
4	133	125	118	124	120	119	110	110	107	115
5	147	146	151	145	151	153	145	135	132	120
6	168	163	167	170	172	174	163	153	152	149
7	187	195	189	185	182	171	172	172	175	169
8	190	192	191	193	192	184	182	180	179	177
9	185	187	196	195	193	188	185	180	175	177
10	194	196	194	199	195	189	189	182	173	165
11	184	190	199	200	197	201	201	197	196	197
12	192	195	198	199	201	199	202	196	191	194
13	175	185	186	185	181	178	180	177	178	181
14	171	175	173	169	165	165	164	159	160	163
15	160	164	160	159	159	161	160	160	158	…

B =

	0	1	2	3	4	5	6	7	8	9
0	135	128	123	122	119	124	120	124	124	130
1	155	142	141	134	133	122	122	124	120	134
2	174	159	151	152	141	136	133	135	126	130
3	162	160	151	145	147	147	146	144	139	142
4	172	161	159	164	158	160	150	150	153	149
5	184	181	190	184	183	191	176	166	166	159
6	198	197	200	203	206	207	195	189	188	180
7	225	222	226	222	218	206	204	208	213	201
8	223	226	222	224	219	215	210	219	213	212
9	220	221	230	229	224	221	224	214	209	210
10	224	228	231	225	229	221	224	215	209	207
11	221	217	227	231	232	233	222	229	229	222
12	222	224	228	230	231	231	229	225	223	226
13	215	211	218	217	211	208	208	209	203	208
14	209	207	204	198	197	197	192	199	190	194
15	200	196	192	190	189	191	193	189	190	…

Завантажені масиви червоного, зеленого і синього кольору (масиви R, G і B) є подмасивами масиву растрових даних C. Аналогічно, графічна інтерпретація масивів R, G і B (у вигляді зображень у градаціях сірого кольору) об'єднана в графічному представленні масиву C. Повноколірне представлення зображення отримаємо в такий спосіб:

R, G, B

При порушенні порядку проходження масивів растрових даних, що характеризують інформацію про канали кольоровості, зображення буде спотворене, наприклад:

G, B, R

B, R, G

1.1.2. Виконуємо команду читання інформаційних даних текстового документа з заданого файла у вигляді одновимірного масиву цілих чисел:

«M:=READBIN("[ім'я файла].txt", byte)».

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

Наприклад, нехай інформаційними даними виступає текстовий документ, який зберігається в файлі з ім'ям «1.txt». Тоді після виконання команди читання символів інформаційного повідомлення в кодуванні ASCII

«M:=READBIN("1.txt", byte)»

отримуємо:

M =

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

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

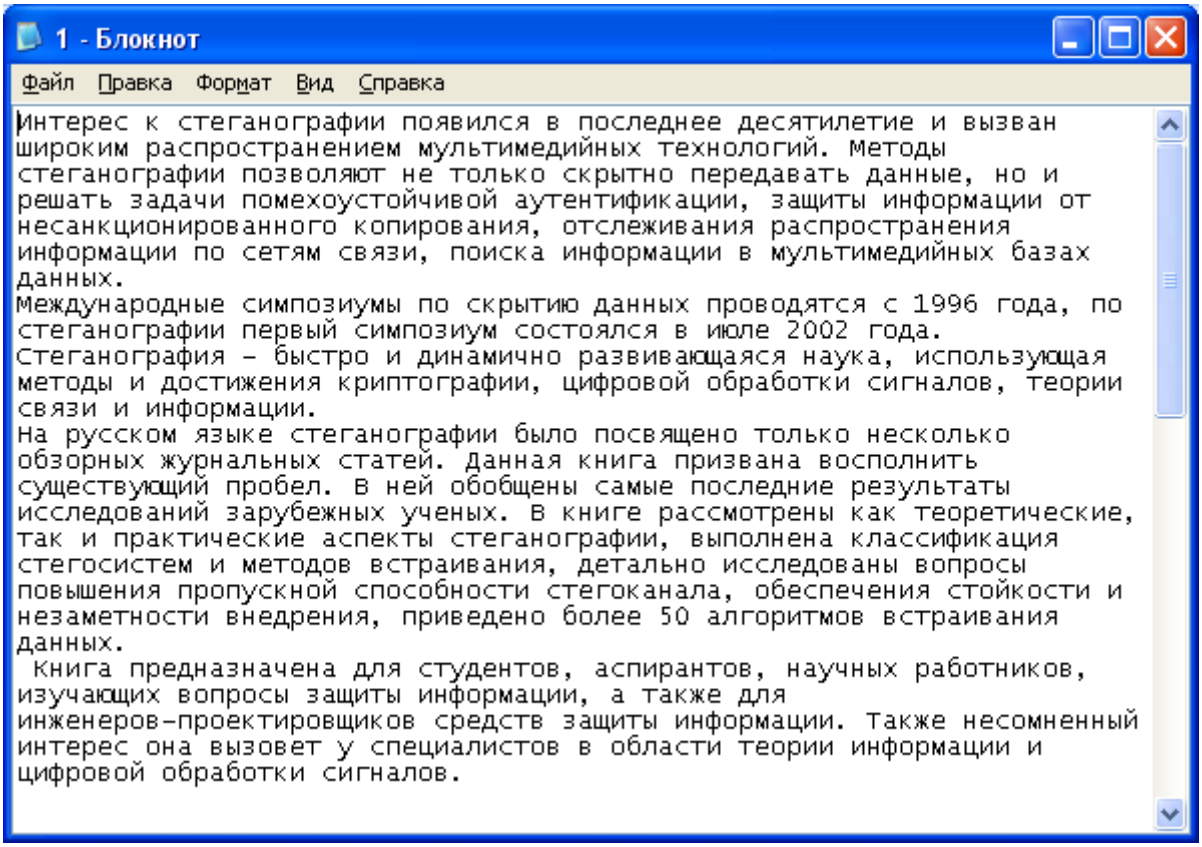
{200, 237, 242, 229, 240, 229, 241}

відповідає набору символів повідомлення

{И, н, т, е, р, е, с}

в кодуванні ASCII.

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



1.2. Перетворимо масив інформаційних даних

Інформаційні дані в масиві M представлені у вигляді набору цілих чисел з інтервалу [0...255] і задають значення символів інформаційного повідомлення в кодуванні ASCII. Для розглянутих стеганографічних методів вбудовування інформації в нерухомі зображення здійснюється побітово. Тобто інформаційні дані з масиву M слід попередньо підготувати,

перетворивши їх в бітовий масив. Для цього в середовищі MathCAD використовуємо такі функції.

1.2.1. Функція перетворення вектора-стовпця з восьми біт в десятковий код:

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

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

$$x := \left(x_0 \quad x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5 \quad x_6 \quad x_7 \right)^T$$

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

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

Наприклад, нехай аргумент x функції B_D (x) заданий таким чином:

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

Тоді значення функції B_D(x) дорівнює

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

1.2.2. Функція перетворення цілого числа в десятковому коді в двійковий вектор-стовпець з восьми біт:

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

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

Алгоритм обчислення значення функції D_B (x) є таким. На кожній ітерації (для кожного значення циклової змінної i) обчислюються наступні значення:

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

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

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

Наприклад, нехай аргумент функції D_B (x) дорівнює 27. Тоді значення функції D_B (x) обчислюється таким чином:

$$\begin{gathered} i=0\text{: } V_0=\mathrm{mod}(27,2)=1,\ x=\mathrm{floor}(27/2)=13; \\ i=1\text{: } V_1=\mathrm{mod}(13,2)=1,\ x=\mathrm{floor}(13/2)=6; \\ i=2\text{: } V_2=\mathrm{mod}(6,2)=0,\ x=\mathrm{floor}(6/2)=3; \\ i=3\text{: } V_3=\mathrm{mod}(3,2)=1,\ x=\mathrm{floor}(3/2)=1; \\ i=4\text{: } V_4=\mathrm{mod}(1,2)=1,\ x=\mathrm{floor}(1/2)=0; \\ i=5\text{: } V_5=\mathrm{mod}(0,2)=0,\ x=\mathrm{floor}(0/2)=0; \\ i=6\text{: } V_6=\mathrm{mod}(0,2)=0,\ x=\mathrm{floor}(0/2)=0; \\ i=7\text{: } V_7=\mathrm{mod}(0,2)=0,\ x=\mathrm{floor}(0/2)=0. \end{gathered}$$

Отже, обчислене значення функції D_B(x) дорівнює

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

1.2.3. Використовуючи функцію D_B (x), перетворимо масив M інформаційних цілих чисел в бітовий масив.

Для цього скористаємося такою процедурою:

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

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

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

M =

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

Виконання процедури перетворення масиву M у бітовий масив M_b відбувається таким чином:

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

j=0: $M\_b_0=0$,

j=1: $M\_b_1=0$,

j=2: $M\_b_2=0$,

j=3: $M\_b_3=1$,

j=4: $M\_b_4=0$,

j=5: $M\_b_5=0$,

j=6: $M\_b_6=1$,

j=7: $M\_b_7=1$;

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

j=0: $M\_b_8=0$,

j=1: $M\_b_9=0$,

j=2: $M\_b_{10}=0$,

j=3: $M\_b_{11}=1$,

j=4: $M\_b_{12}=0$,

j=5: $M\_b_{13}=0$,

j=6: $M\_b_{14}=1$,

j=7: $M\_b_{15}=1$;

…

1.3. Реалізуємо алгоритм вбудовування даних у просторову область зображень методом LSB. Для цього скористаємося такою процедурою:

```
S := | for j ∈ 0.. rows(R) − 1
     |   for i ∈ 0.. cols(R) − 1
     |     S[j,i] ← R[j,i]
     | for l ∈ 0.. rows(M_b) − 1
     |   i ← floor( l / rows(R) )
     |   j ← l − i·rows(R)
     |   V ← (D_B(R[j,i]))
     |   V_0 ← M_b_l
     |   S[j,i] ← B_D(V)
     | S
```

Наведена процедура реалізує поелементне вбудовування бітового масиву інформаційних даних M_b в найменш значущі біти послідовних

байтів масиву растрових даних каналу червоного кольору, тобто вбудовування здійснюється тільки в LSB масиву R. Перші три рядки процедури виконують перезаписування даних з масиву растрових даних каналу червоного кольору в новий масив S. Далі для всіх елементів бітового масиву інформаційних даних M_b обчислюються координати (номери стовпців і рядків) елементів масиву R, в які вони будуть вбудовані. Так, розділивши значення індексу l на кількість рядків у масиві R, отримаємо номер (індекс i) того стовпця, в елементи якого буде вбудований l-й біт повідомлення. Виконавши обчислення $1 - j * rows(R)$, отримаємо номер (індекс j) того рядка, в елементи якого буде вбудований l-й біт повідомлення. Бітове подання десяткового числа $R_{j,i}$ записується у змінну V (це перетворення реалізується за допомогою розглянутої вище функції). Поточний біт повідомлення M_b_l заноситься в нульовий (найменш значущий) біт масиву V_0, після чого виконуємо зворотне перетворення масиву V зі зміненим LSB. Отримане десяткове число записуємо в масив S з тими самими індексами i та j. Таким чином, усі біти інформаційного повідомлення з масиву M_b записуються в найменш значущі біти байтів каналу червоного кольору зображення. Якщо розмір повідомлення є невеликим, тоді невикористані елементи масиву S заповнюються вихідними даними з масиву R.

Для візуального перегляду результату вбудовування інформаційних даних виведемо вихідний масив растрових даних червоного кольору R та отриманий масив зі зміненими найменш значущими бітами. Для розглянутого прикладу маємо:

R =

	0	1	2	3	4
0	86	79	72	72	72
1	110	97	90	86	78
2	132	120	112	105	96
3	122	116	105	104	103
4	131	122	117	118	118
5	147	147	148	148	150
6	169	164	167	170	173
7	189	195	193	189	183
8	191	192	194	199	194
9	186	188	194	198	192
10	195	196	199	200	201
11	185	189	202	203	203
12	192	196	198	199	204
13	177	185	187	186	180
14	173	176	174	166	165
15	160	162	158	153	...

S =

	0	1	2	3	4
0	86	78	73	72	72
1	110	96	90	86	79
2	132	121	112	105	97
3	123	116	105	105	103
4	130	123	117	119	119
5	146	147	149	149	150
6	169	165	167	170	173
7	189	194	192	189	182
8	191	192	195	199	194
9	186	188	194	198	193
10	195	197	198	201	201
11	185	188	203	203	203
12	192	197	199	199	205
13	177	185	187	186	180
14	173	177	174	166	165
15	161	162	158	152	...

З представлених даних видно, що, наприклад, значення $R_{0,0}$, $R_{1,0}$, $R_{2,2}$ та інші повністю ідентичні відповідним значенням $S_{0,0}$, $S_{1,0}$, $S_{2,2}$. Практично це означає, що найменш значущі біти в цих елементах масиву R

співпали з вбудованими інформаційними бітами повідомлення. Навпаки, значення $R_{0,1}$, $R_{1,1}$, $R_{2,1}$ відрізняються на одиницю від відповідних значень масиву S. Це означає зміну найменш значущого біта елемента контейнера в процесі вбудовування повідомлення. Відзначимо, що внесені спотворення знаходяться нижче порога зорової чутливості людини.

Графічна інтерпретація порожнього і заповненого контейнера (каналу червоного кольору в градаціях сірого) наведена на наступному рисунку, з якого видно, що візуально внесені спотворення є непомітними, це підтверджує висновок про чутливість органів зору людини.

R

S

Отриманий заповнений масив S записуємо в канал червоного кольору контейнера. Для виконання цієї операції з використанням команди

«WRITERGB([ім’я файла].bmp):=augment(S,G,B)»

під відповідним ім’ям записуємо на фізичний носій сформований контейнер зі зміненими LSB в каналі червоного кольору. Для розглянутого прикладу виконуємо команду

«WRITERGB(Stego.bmp):=augment(S,G,B)»,

виконання якої формує на фізичному носії новий файл з ім’ям «Stego.bmp».

Для графічного відображення вихідного (порожнього) і заповненого контейнера виконаємо вставку відповідних зображень:

"1.bmp"

"Stego. bmp"

Переконаємося у відсутності видимих спотворень.

1.4. Реалізуємо алгоритм вилучення даних з просторової області зображень методом LSB. Для цього в новому вікні середовища MathCAD

виконуємо команди читання растрових даних нерухомого зображення із заданого файла (файла заповненого контейнера) у вигляді двовимірного масиву цілих чисел. Для розглянутого прикладу виконуємо команди:

«C:=READRGB(“Stego.bmp”)», «R:=READ_RED(“Stego.bmp”)»,
«G:=READ_ GREEN (“Stego.bmp”)»,«B:=READ_BLUE(“Stego.bmp”)».

Отримуємо такий результат:

R =

	0	1	2	3	4	5
0	86	78	73	72	72	69
1	110	96	90	86	79	70
2	132	121	112	105	97	88
3	123	116	105	105	103	102
4	130	123	117	119	119	117
5	146	147	149	149	150	152
6	169	165	167	170	173	174
7	189	194	192	189	182	...

R

Далі, скориставшись розглянутою функцією D_B(x), отримуємо найменш значущі біти з масиву даних каналу червоного кольору. Для цього використовуємо таку процедуру:

$$
\begin{aligned}
M\_b1 := \;& \text{for } i \in 0..\operatorname{cols}(R) - 1 \\
& \quad \text{for } j \in 0..\operatorname{rows}(R) - 1 \\
& \quad\quad V \leftarrow D\_B\left(R_{j,i}\right) \\
& \quad\quad M\_b1_{i\cdot\operatorname{rows}(R)+j} \leftarrow V_0 \\
& M\_b1
\end{aligned}
$$

M_b1 =

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

Процедура виконує для всіх елементів масиву R формування двійкового коду десяткового числа і записує його в змінну V. Нульовий (найменш значущий) біт масиву V заноситься до відповідного елемента масиву M_b1. Індекс елементів масиву M_b1 змінюється в залежності від номерів рядків і стовпців оброблюваного елемента масиву R. У результаті отримуємо лінійний бітовий масив M_b1, заповнений найменш значущими бітами масиву растрових даних каналу червоного кольору заповненого контейнера.

1.5. Перетворимо масив інформаційних даних.

Для формування текстового повідомлення за отриманими бітами сформуємо масив M1. Для цього скористаємося такою процедурою:

$$M1 := \left| \begin{array}{l} \text{for } i \in 0..\dfrac{\text{rows}(M\_b1)}{8} - 1 \\ \quad \left| \begin{array}{l} \text{for } j \in 0..7 \\ \quad V_j \leftarrow M\_b1_{i\cdot 8+j} \\ M1_i \leftarrow B\_D(V) \end{array} \right. \\ M1 \end{array} \right.$$

Алгоритм формування масиву M1 наступний. Для всіх елементів масиву M_b1 обчислюємо значення індексу l – поточного номера десяткового числа (коду інформаційного символу). Для цього беремо цілу частину від ділення i (індексу біта в масиві M_b1) на вісім (число бітів в одному інформаційному символі в кодуванні ASCII). Далі, всі біти поточного символу записуємо в службову змінну V, після чого за допомогою функції B_D(x) по черзі обчислюємо коди інформаційних символів. Отримані цілі числа (коди символів у кодуванні ASCII) записуємо в масив M1.

M1 =

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

Виконання процедури перетворення бітового масиву M_b1 в масив десяткових чисел M1 відбувається таким чином:

i=0:

j=0: $V_0=M\_b1_0=0$,
j=1: $V_1=M\_b1_1=0$,
j=2: $V_2=M\_b1_2=0$,
j=3: $V_3=M\_b1_3=1$,
j=4: $V_4=M\_b1_4=0$,
j=5: $V_5=M\_b1_5=0$,
j=6: $V_6=M\_b1_6=1$,
j=7: $V_7=M\_b1_7=1$;

$M1_0=B\_D(V)=B\_D((0\ 0\ 0\ 1\ 0\ 0\ 1\ 1)^T)=200$;

i=1:

j=0: $V_0=M\_b1_8=1$,
j=1: $V_1=M\_b1_9=0$,
j=2: $V_2=M\_b1_{10}=1$,
j=3: $V_3=M\_b1_{11}=1$,
j=4: $V_4=M\_b1_{12}=0$,
j=5: $V_5=M\_b1_{13}=1$,
j=6: $V_6=M\_b1_{14}=1$,
j=7: $V_7=M\_b1_{15}=1$,

$M1_1=B\_D(V)=B\_D((1\ 0\ 1\ 1\ 0\ 1\ 1\ 1)^T)=237$;

…

1.6. Отриманий масив цілих чисел записуємо на фізичний носій у вигляді текстового файла. Для цього скористаємося командою:

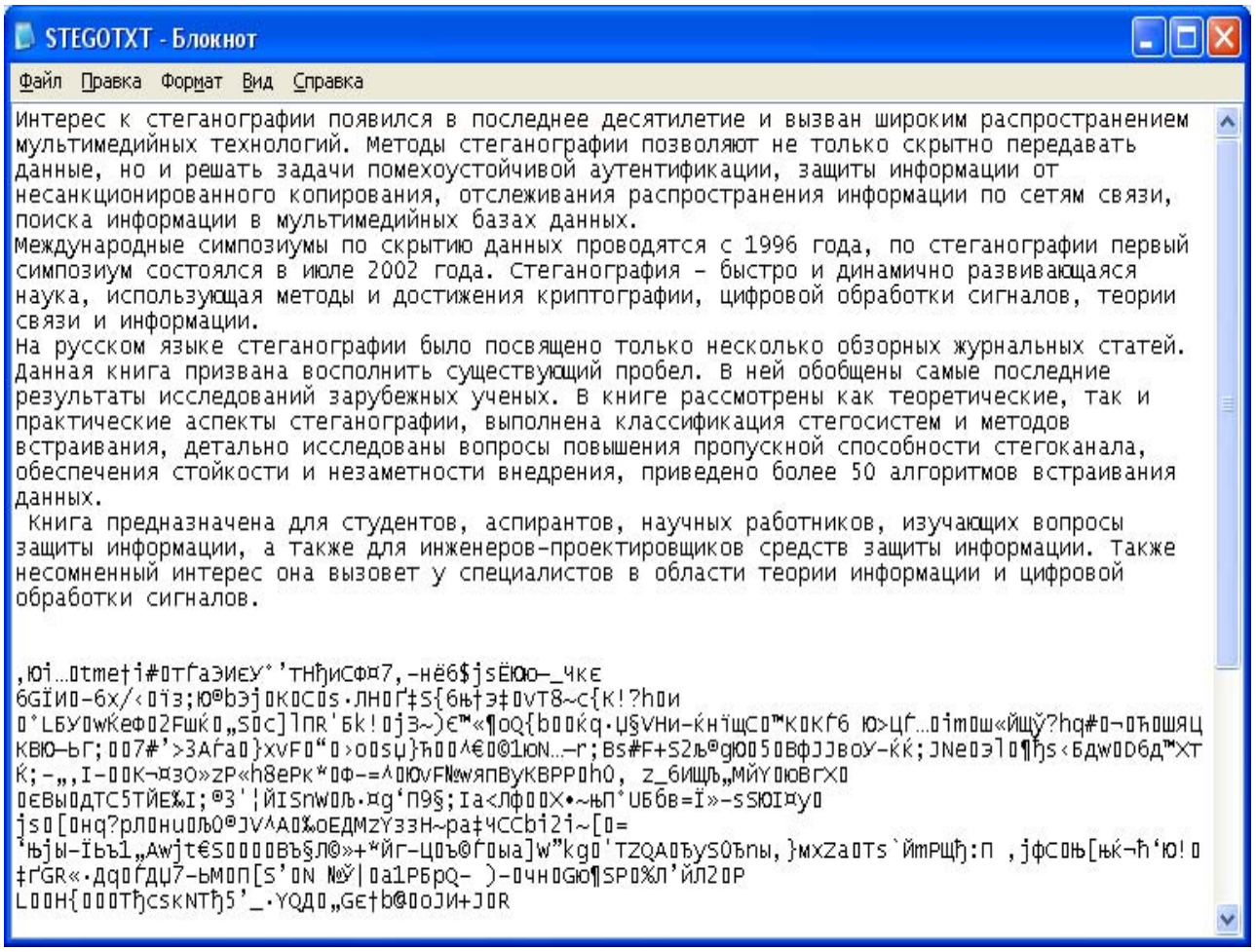
«WRITEBIN(“[ім’я файла].txt”, byte,1):=M1».

Для розглянутого прикладу виконуємо команду:

«WRITEBIN(“STEGOTXT.txt”, byte,1):=M1»,

в результаті якої на фізичному носії буде записаний текстовий файл з ім’ям "STEGOTXT.txt".

Необхідно зазначити, що наведена процедура дозволяє отримати всі найменш значущі біти контейнера (з каналу червоного кольору), тобто отримуються всі LSB навіть з тих байтів, в які не вбудовувалася інформація. Після формування повідомлення в кінці текстового файла "STEGOTXT.txt" можуть бути присутні символи, отримані в результаті вилучення LSB, немодифіковані в процесі вбудовування інформації, тобто так звані «випадкові» символи.



Останній рисунок наочно демонструє правильність роботи розглянутих процедур і функцій. Інформаційне повідомлення, вбудоване в просторову область нерухомого зображення методом LSB, вилучено правильно.

Завдання 2. Експериментальні дослідження зорового порогу чутливості людини до зміни яскравості зображень

2.1. Внесемо зміни до реалізації алгоритму вбудовування даних у нерухомі зображення методом LSB. Для цього змінимо порядковий номер біта, який використовується для вбудовування інформації в двійковому поданні елементів контейнера (окремих байтів яскравості конкретних пікселів зображення).

```
S := | for j ∈ 0.. rows(R) − 1
     |   for i ∈ 0.. cols(R) − 1
     |     S_{j,i} ← R_{j,i}
     | for l ∈ 0.. rows(M_b) − 1
     |   i ← floor(l / rows(R))
     |   j ← l − i·rows(R)
     |   V ← (D_B(R_{j,i}))
     |   V_0 ← M_b_l
     |   S_{j,i} ← B_D(V)
     | S
```

```
S := | for j ∈ 0.. rows(R) − 1
     |   for i ∈ 0.. cols(R) − 1
     |     S_{j,i} ← R_{j,i}
     | for l ∈ 0.. rows(M_b) − 1
     |   i ← floor(l / rows(R))
     |   j ← l − i·rows(R)
     |   V ← (D_B(R_{j,i}))
     |   V_1 ← M_b_l
     |   S_{j,i} ← B_D(V)
     | S
```

```
S := | for j ∈ 0.. rows(R) − 1
     |   for i ∈ 0.. cols(R) − 1
     |     S_{j,i} ← R_{j,i}
     | for l ∈ 0.. rows(M_b) − 1
     |   i ← floor(l / rows(R))
     |   j ← l − i·rows(R)
     |   V ← (D_B(R_{j,i}))
     |   V_2 ← M_b_l
     |   S_{j,i} ← B_D(V)
     | S
```

Перша наведена процедура реалізує вбудовування інформаційних даних в найменш значущі (нульові) біти контейнера (в біти V_0). Друга і третя процедури реалізують вбудовування інформаційних даних у наступні за значущістю біти контейнера (в біти V_1 та V_2). Наведемо приклад графічного зображення контейнера для кожного з розглянутих випадків.

S

S

S

Вочевидь, не вдається виявити видимих спотворень при вбудовуванні інформаційних повідомлень у нульовий, перший або другий за значущістю біти. Це пояснюється тим, що максимальні спотворення, які вносяться до окремих пікселів зображення за допомогою зміни рівня їх яскравості, для кожного з розглянутих випадків не перевищують величин 2^0=1, 2^1=2, 2^2=4 відповідно, що знаходиться нижче зорового порога чутливості людини до незначної зміни яскравості зображення.

Продовжимо змінювати порядковий номер біта, що використовується для вбудовування інформації, в двійковому поданні елементів контейнера, до тих пір, поки візуально не стануть видими спотворення яскравості окремих пікселів зображення.

```
S := | for j ∈ 0.. rows(R) − 1
     |   for i ∈ 0.. cols(R) − 1
     |     S_{j,i} ← R_{j,i}
     | for l ∈ 0.. rows(M_b) − 1
     |   | i ← floor( l / rows(R) )
     |   | j ← l − i·rows(R)
     |   | V ← (D_B(R_{j,i}))
     |   | V_3 ← M_b_l
     |   | S_{j,i} ← B_D(V)
     | S
```

```
S := | for j ∈ 0.. rows(R) − 1
     |   for i ∈ 0.. cols(R) − 1
     |     S_{j,i} ← R_{j,i}
     | for l ∈ 0.. rows(M_b) − 1
     |   | i ← floor( l / rows(R) )
     |   | j ← l − i·rows(R)
     |   | V ← (D_B(R_{j,i}))
     |   | V_4 ← M_b_l
     |   | S_{j,i} ← B_D(V)
     | S
```

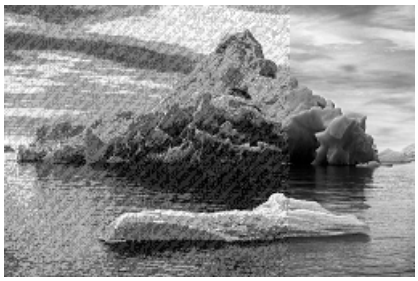
```
S := | for j ∈ 0.. rows(R) − 1
     |   for i ∈ 0.. cols(R) − 1
     |     S_{j,i} ← R_{j,i}
     | for l ∈ 0.. rows(M_b) − 1
     |   | i ← floor( l / rows(R) )
     |   | j ← l − i·rows(R)
     |   | V ← (D_B(R_{j,i}))
     |   | V_5 ← M_b_l
     |   | S_{j,i} ← B_D(V)
     | S
```

Наведені процедури реалізують вбудовування інформаційних даних у наступні за значущістю біти контейнера (в біти V_3, V_4 і V_5).

2.2. Експериментально встановимо, модифікація яких бітів зображення не призводить до помітних спотворень. Для цього наведемо приклад графічного зображення контейнера для кожного з розглянутих випадків (вбудовування здійснювалося в біти V_3, V_4 і V_5).

S

S



S

З наведених зображень зрозуміло, що вбудовування даних у треті за значущістю біти контейнера призводить до ледь помітних спотворень (яскравість відповідних пікселів змінилася на 2^3=8 рівнів). Модифікація четвертих і п'ятих за значущістю бітів призводить до значних спотворень зображення (яскравість відповідних пікселів змінилася на 2^4= 16 та 2^5=32 рівні, відповідно). Отже, з результатів експериментальних досліджень отримуємо, що зорова система людини для розглянутого прикладу зображення чутлива до зміни третього, четвертого і т. д. бітів (за їх значущістю) контейнера (окремих байтів яскравості конкретних пікселів зображення).

2.3. Розрахуємо зоровий поріг чутливості людини до незначної зміни яскравості зображення.

Використовуємо експериментальні дані для розрахунку зорового порога чутливості людини до незначної зміни яскравості зображення. Позначимо символом Δ величину внесених спотворень яскравості (як число рівнів квантування) окремих пікселів зображення при використанні стеганографічного алгоритму вбудовування інформації в нерухомі зображення на основі модифікації окремих бітів контейнера. За специфікацією формату зображень *bmp24 загальне число рівнів квантування яскравості окремих пікселів дорівнює 2^8=256. Тоді зоровий поріг чутливості (ПЧ) людини до незначної зміни яскравості зображення визначимо як

$$ПЧ=(Δ/256)*100\ \%.$$

Для розглянутого прикладу видимі спотворення були виявлені після модифікації третіх за значущістю бітів контейнера, відповідна величина Δ=8. Отже, зоровий поріг чутливості людини, обчислений за емпіричними даними, становить

$$ПЧ=(Δ/256)*100\ \%=(8/256)*100\ \%=3{,}125\ \%,$$

що узгоджується з відомими теоретичними даними.

Завдання 3. Реалізація алгоритмів вбудовування та вилучення повідомлень методом псевдовипадкової перестановки

3.1. Завантажуємо вихідні дані (див. п. 1.1).

3.2. Перетворюємо масив інформаційних даних (див. п. 1.2).

3.3. Вводимо секретне правило псевдовипадкової перестановки.

Вбудовувана інформація попередньо обробляється простим перестановочним шифром. Задамо секретний ключ – правило псевдовипадкової перестановки у вигляді перестановочної матриці розміром nxn, де n – розмір оброблюваного блока інформаційних бітів. Нехай, наприклад, n=10. Тоді перестановочна матриця складається з двомірного масиву 10x10 бітів, причому в кожному рядку і в кожному стовпці масиву міститься тільки один одиничний елемент, всі інші елементи – нулі.

Для розглянутого прикладу задамо перестановочну матрицю таким чином:

$$P := \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

3.4. Розбиваємо інформаційне повідомлення на блоки однакової довжини і обробляємо простим перестановочним шифром (шифруємо). Для цього скористаємося такою процедурою:

$$\begin{array}{l} \text{M\_b\_P} := \left| \begin{array}{l} \text{for } i \in 0..\left(\dfrac{\text{rows(M\_b)}}{n} - 1 \right) \\ \quad \left| \begin{array}{l} \text{for } j \in 0..n-1 \\ \quad V_j \leftarrow \text{M\_b}_{i \cdot n + j} \end{array} \right. \\ \quad V1 \leftarrow (V)^T \cdot P \\ \quad \text{for } j \in 0..n-1 \\ \quad \text{M\_b\_P}_{i \cdot n + j} \leftarrow \left(V1^T \right)_j \\ \text{M\_b\_P} \end{array} \right. \end{array}$$

Алгоритм перетворення працює так. Інформаційне повідомлення розбивається на блоки однакової довжини (довжини n), після чого кожен блок поелементно записується в службову змінну V. Тобто, на кожному циклі (для кожного блока даних) в змінній V зберігаються поточні n бітів повідомлення. Вектор V множиться на перестановочну матрицю P, чим забезпечується шифрування простим перестановочним шифром. Оброблені таким чином дані об’єднуються в єдиний бітовий масив M_b_P.

Для розглянутого прикладу інформаційних даних (див. п. 1.1) алгоритм розбиття інформаційного повідомлення на блоки однакової довжини і обробки простим перестановочним шифром функціонує таким чином:

M_b =

	0
0	0
1	0
2	0
3	1
4	0
5	0
6	1
7	1
8	1
9	0
10	1
11	1
12	0
13	1
14	1
15	1
16	0
17	1
18	0
19	0
20	1
21	1
22	1
23	1
24	...

i=0: $V=(0\ 0\ 0\ 1\ 0\ 0\ 1\ 1\ 1\ 0)^T$,
$V1=V^T*P=(0\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 0)$,
j=0: $M\_b\_P_0=0$,
j=1: $M\_b\_P_1=1$,
j=2: $M\_b\_P_2=0$,
j=3: $M\_b\_P_3=0$,
j=4: $M\_b\_P_4=1$,
j=5: $M\_b\_P_5=1$,
j=6: $M\_b\_P_6=0$,
j=7: $M\_b\_P_7=0$,
j=8: $M\_b\_P_8=1$,
j=9: $M\_b\_P_9=0$;

i=0: $V=(1\ 1\ 0\ 1\ 1\ 1\ 0\ 1\ 0\ 0)^T$,
$V1=V^T*P=(1\ 1\ 1\ 0\ 0\ 0\ 1\ 0\ 1\ 1)$,
j=0: $M\_b\_P_{10}=1$,
j=1: $M\_b\_P_{11}=1$,
j=2: $M\_b\_P_{12}=1$,
j=3: $M\_b\_P_{13}=0$,
j=4: $M\_b\_P_{14}=0$,
j=5: $M\_b\_P_{15}=0$,
j=6: $M\_b\_P_{16}=1$,
j=7: $M\_b\_P_{17}=0$,
j=8: $M\_b\_P_{18}=1$,
j=9: $M\_b\_P_{19}=1$;

...

M_b_P =

	0
0	0
1	1
2	0
3	0
4	1
5	1
6	0
7	0
8	1
9	0
10	1
11	1
12	1
13	0
14	0
15	0
16	1
17	0
18	1
19	1
20	1
21	1
22	1
23	1
24	...

3.5. Реалізуємо алгоритм вбудовування даних у просторову область зображень методом LSB (див. п. 1.3).

3.6. Реалізуємо алгоритм вилучення даних з просторової області зображень методом LSB (див. п. 1.4).

3.7. Розбиваємо отримане повідомлення (записане в бітовий масив M_b1) на блоки однакової довжини і обробляємо простим перестановочним шифром (розшифровуємо). Для цього скористаємося такою процедурою:

```
M_b1_P := | for i ∈ 0.. (rows(M_b1)/n − 1)
          |   for j ∈ 0.. n − 1
          |     V_j ← M_b1_(i·n+j)
          |   V1 ← (V)^T · P^(−1)
          |   for j ∈ 0.. n − 1
          |     M_b1_P_(i·n+j) ← (V1^T)_j
          | M_b1_P
```

Алгоритм перетворення працює таким чином. Отримане повідомлення розбивається на блоки однакової довжини (довжини n), після чого кожен блок поелементно записується в службову змінну V. Тобто, на кожному циклі (для кожного блока витягнутих даних) в змінній V зберігаються поточні n бітів. Вектор V множиться на матрицю P^{-1} (матрицю, зворотну до введеної вище перестановочної матриці P), завдяки чому забезпечується розшифрування отриманих даних простим перестановочним шифром. Оброблені таким чином дані об'єднуються в єдиний бітовий масив M_b1_P.

Для розглянутого прикладу інформаційних даних (див. п. 3.4) алгоритм розбиття отриманого повідомлення на блоки однакової довжини і обробки простим перестановочним шифром (розшифрування) функціонує таким чином:

M_b1 =

	0
0	0
1	1
2	0
3	0
4	1
5	1
6	0
7	0
8	1
9	0
10	1
11	1
12	1
13	0
14	0
15	0
16	1
17	0
18	1
19	1
20	1
21	1
22	1
23	1
24	...

i=0: $V=(0\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 0)^T$,
$V1=V^T*P=(0\ 0\ 0\ 1\ 0\ 0\ 1\ 1\ 1\ 0)$,
j=0: $M\_b\_P_0=0$,
j=1: $M\_b\_P_1=0$,
j=2: $M\_b\_P_2=0$,
j=3: $M\_b\_P_3=1$,
j=4: $M\_b\_P_4=0$,
j=5: $M\_b\_P_5=0$,
j=6: $M\_b\_P_6=1$,
j=7: $M\_b\_P_7=1$,
j=8: $M\_b\_P_8=1$,
j=9: $M\_b\_P_9=0$;

i=0: $V=(1\ 1\ 1\ 0\ 0\ 0\ 1\ 0\ 1\ 1)^T$,
$V1=V^T*P=(1\ 1\ 0\ 1\ 1\ 1\ 0\ 1\ 0\ 0)$,
j=0: $M\_b\_P_{10}=1$,
j=1: $M\_b\_P_{11}=1$,
j=2: $M\_b\_P_{12}=0$,
j=3: $M\_b\_P_{13}=1$,
j=4: $M\_b\_P_{14}=1$,
j=5: $M\_b\_P_{15}=1$,
j=6: $M\_b\_P_{16}=0$,
j=7: $M\_b\_P_{17}=1$,
j=8: $M\_b\_P_{18}=0$,
j=9: $M\_b\_P_{19}=0$;

…

M_b1_P =

	0
0	0
1	0
2	0
3	1
4	0
5	0
6	1
7	1
8	1
9	0
10	1
11	1
12	0
13	1
14	1
15	1
16	0
17	1
18	0
19	0
20	1
21	1
22	1
23	1
24	...

3.8. Перетворимо масив інформаційних даних (див. п. 1.5).

3.9. Отриманий масив цілих чисел записуємо на фізичний носій у вигляді текстового файла (див. п. 1.6).

Завдання 4. Реалізація алгоритмів вбудовування та вилучення повідомлень методом псевдовипадкового інтервалу

4.1. Завантажуємо вихідні дані (див. п. 1.1).
4.2. Перетворюємо масив інформаційних даних (див. п. 1.2).
4.3. Вводимо секретне правило псевдовипадкової перестановки.

Секретним ключем виступає правило, за яким окремі біти інформаційного повідомлення вбудовуються в LSB байт контейнера, тобто секретний ключ – це набір псевдовипадкових чисел, що задають величини інтервалів між пікселями зображення, які модифікуються в процесі вбудовування інформації.

key =

	0
0	153
1	155
2	25
3	96
4	159
5	97
6	43
7	59
8	134
9	11
10	99
11	33
12	108
13	102
14	74
15	...

Припустимо, що інформаційне повідомлення вбудовується побітово в блок даних зображень, причому один біт повідомлення вбудовується в один стовпець масиву даних контейнера. Номер вбудованого біта задає номер стовпчика контейнера, в який буде вбудовуватися біт повідомлення. Поточне значення секретного ключа задає номер рядка контейнера, в який буде вбудований поточний біт повідомлення. Таким чином, як секретний ключ будемо використовувати масив псевдовипадкових чисел в інтервалі допустимих номерів рядків контейнера, розмір масиву дорівнює числу стовпців контейнера. Для реалізації такого підходу використовуємо таку процедуру:

$$\text{key} := \left| \begin{array}{l} \text{for} \quad i \in 0 .. \text{cols}(R) - 1 \\ \quad \text{key}_i \leftarrow \text{floor}(\text{rnd}(\text{rows}(R))) \\ \text{key} \end{array} \right.$$

З наведеного прикладу видно, що перший біт повідомлення буде вбудований в 153-й рядок першого стовпця масиву даних контейнера, другий біт повідомлення буде вбудований в 155-й рядок другого стовпця повідомлення і т. д.

4.4. Реалізуємо алгоритм вбудовування даних у просторову область зображень методом ПСІ. Для цього скористаємося такою процедурою:

```
S := | for j ∈ 0.. rows(R) − 1
     |   for i ∈ 0.. cols(R) − 1
     |     S_{j,i} ← R_{j,i}
     | for i ∈ 0.. cols(R) − 1
     |   | V ← D_B(R_{key_i, i})
     |   | V_0 ← M_b_i
     |   | S_{key_i, i} ← B_D(V)
     | S
```

Алгоритм працює таким чином. Вихідний контейнер (масив R даних каналу червоного кольору) перезаписується в новий масив S. Далі, в кожному стовпці контейнера зчитується значення яскравості (десяткове число) з рядка з номером, що задається ключем. У отриманому числі замінюється найменш значущий біт даних на поточний вбудовуваний біт. Далі записуємо заповнений контейнер на фізичний носій (див. п. 1.3).

4.5. Реалізуємо алгоритм вбудовування даних у просторову область зображень методом ПСІ. Для цього після завантаження даних контейнера (див. п. 1.4) скористаємося такою процедурою:

```
M_b1 := | for i ∈ 0.. cols(R) − 1
        |   | V ← D_B(R_{key_i, i})
        |   | M_b1_i ← V_0
        | M_b1
```

Алгоритм працює таким чином. У всіх стовпцях контейнера по черзі зчитуються значення з рядків, номери яких задані значенням секретного ключа. З отриманих даних витягуються найменш значущі біти, що несуть інформаційний зміст вбудованого повідомлення.

4.6. Перетворюємо масив інформаційних даних (див. п. 1.5).

4.7. Отриманий масив цілих чисел записуємо на фізичний носій у вигляді текстового файла (див. п. 1.6).

6. ПРИКЛАД ОФОРМЛЕННЯ ЗВІТУ З ЛАБОРАТОРНОЇ РОБОТИ

Лабораторна робота № 1

Приховування даних у просторовій області зображень шляхом модифікації найменш значущого біта

Вихідні дані:

"1.bmp"

```
C :=READRGB("1.bmp" )
R := READ_RED("1.bmp" )
G := READ_GREEN( "1.bmp" )
B := READ_BLUE("1.bmp" )
M := READBIN("3.txt", "byte" )
```

R

C =

	0	1	2	3	4	5	6	7
0	86	79	72	72	72	69	71	74
1	110	97	90	86	78	71	70	70
2	132	120	112	105	96	88	81	83
3	122	116	105	104	103	102	101	99
4	131	122	117	118	118	116	105	107
5	147	147	148	148	150	153	145	135
6	169	164	167	170	173	175	164	155
7	189	195	193	189	183	172	173	173
8	191	192	194	199	194	187	182	182
9	186	188	194	198	192	187	187	180
10	195	196	199	200	201	190	192	186
11	185	189	202	203	203	199	203	199
12	192	196	198	199	204	202	206	199
13	177	185	187	186	180	178	179	177
14	173	176	174	166	165	165	163	161
15	160	162	158	153	156	158	157	156

R =

	0	1	2	3	4	5	6	7
0	86	79	72	72	72	69	71	74
1	110	97	90	86	78	71	70	70
2	132	120	112	105	96	88	81	83
3	122	116	105	104	103	102	101	99
4	131	122	117	118	118	116	105	107
5	147	147	148	148	150	153	145	135
6	169	164	167	170	173	175	164	155
7	189	195	193	189	183	172	173	173
8	191	192	194	199	194	187	182	182
9	186	188	194	198	192	187	187	180
10	195	196	199	200	201	190	192	186
11	185	189	202	203	203	199	203	199
12	192	196	198	199	204	202	206	199
13	177	185	187	186	180	178	179	177
14	173	176	174	166	165	165	163	161
15	160	162	158	153	156	158	157	...

M =

	0
0	200
1	237
2	242
3	229
4	240
5	229
6	241
7	32
8	234
9	32
10	241
11	242
12	229
13	...

Програмна реалізація алгоритмів приховування повідомлень методом LSB:

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

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

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

$$S := \left| \begin{array}{l} \text{for } j \in 0..\text{rows}(R) - 1 \\ \quad \text{for } i \in 0..\text{cols}(R) - 1 \\ \quad\quad S_{j,i} \leftarrow R_{j,i} \\ \text{for } l \in 0..\text{rows}(M\_b) - 1 \\ \quad i \leftarrow \text{floor}\left(\frac{l}{\text{rows}(R)}\right) \\ \quad j \leftarrow l - i \cdot \text{rows}(R) \\ \quad V \leftarrow D\_B\left(R_{j,i}\right) \\ \quad V_0 \leftarrow M\_b_l \\ \quad S_{j,i} \leftarrow B\_D(V) \\ S \end{array} \right.$$

M_b=

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

WRITERGB("Stego.bmp") := augment (S, G, B)

Візуальне порівняння порожньою та заповненого контейнерів:

"1.bmp"

"Stego. bmp"

Програмна реалізація алгоритмів приховування та вилучення повідомлень методом ПВП:

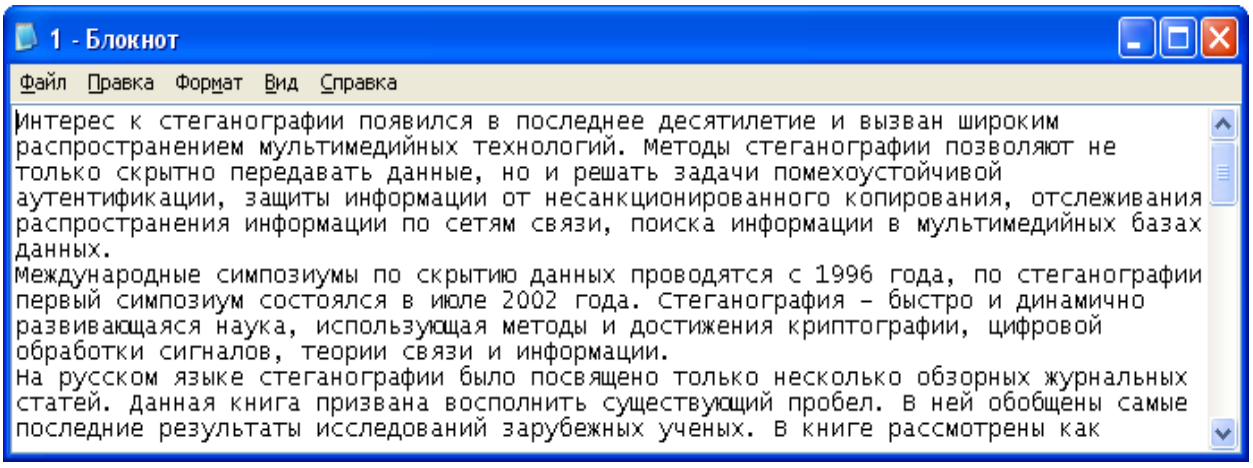
```
M1 := for i ∈ 0.. rows(M_b1) − 8
        l ← floor(i / 8)
        for j ∈ 0.. 7
          V_j ← M_b1_(l·8+j)
        M1_l ← B_D(V)
      M1

M_b1 := for i ∈ 0.. cols(R) − 1
          for j ∈ 0.. rows(R) − 1
            V ← D_B(R_(j,i))
            M_b1_(i·rows(R)+j) ← V_0
        M_b1

WRITEBIN("STEGOTXT.txt","byte",1) := M1
```

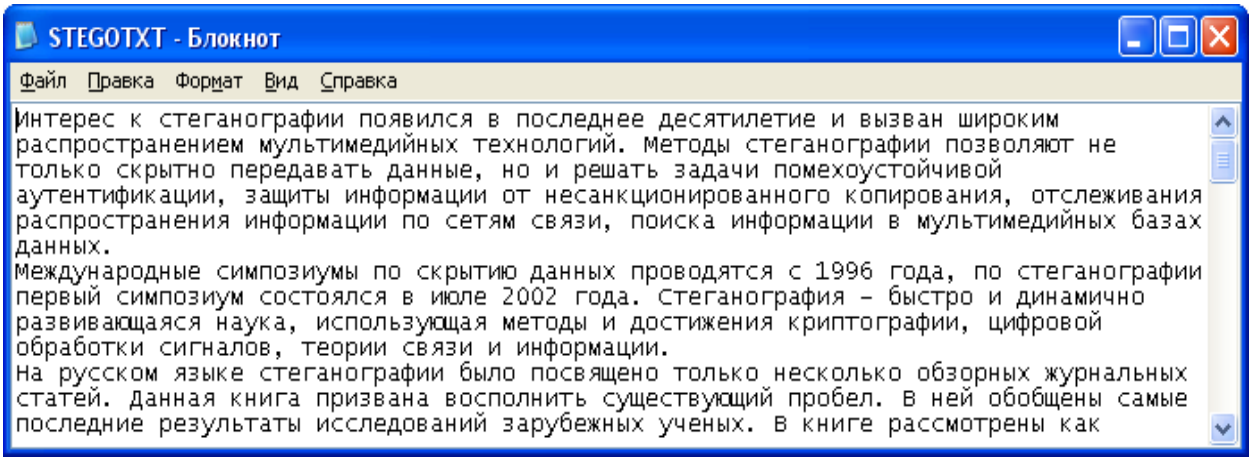
M_b1 =

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

M1 =

	0
0	240
1	109
2	233
3	239
4	168
5	196
6	117
7	226
8	59
9	6
10	231
11	194
12	108
13	253
14	28
15	...





Програмна реалізація алгоритмів приховування та вилучення повідомлень методом ПВІ:

$$P := \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \qquad n := 10$$

$$M\_b\_P := \left| \begin{array}{l} \text{for } i \in 0..\left(\frac{\text{rows}(M\_b)}{n} - 1\right) \\ \quad \left| \begin{array}{l} \text{for } j \in 0..n-1 \\ \quad V_j \leftarrow M\_b_{i \cdot n + j} \\ V1 \leftarrow (V)^T \cdot P \\ \text{for } j \in 0..n-1 \\ \quad M\_b\_P_{i \cdot n + j} \leftarrow \left(V1^T\right)_j \end{array} \right. \\ M\_b\_P \end{array} \right.$$

$M\_b =$

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

$M\_b\_P =$

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

$$M\_b1\_P := \left| \begin{array}{l} \text{for } i \in 0..\left(\frac{\text{rows}(M\_b1)}{n} - 1\right) \\ \quad \left| \begin{array}{l} \text{for } j \in 0..n-1 \\ \quad V_j \leftarrow M\_b1_{i \cdot n + j} \\ V1 \leftarrow (V)^T \cdot P^{-1} \\ \text{for } j \in 0..n-1 \\ \quad M\_b\_P1_{i \cdot n + j} \leftarrow \left(V1^T\right)_j \end{array} \right. \\ M\_b\_P1 \end{array} \right.$$

$M\_b1 =$

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

$M\_b1\_P =$

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

Програмна реалізація алгоритмів приховування та вилучення повідомлень методом ПВІ:

```
key := | for i ∈ 0.. cols(R) − 1
       |   key_i ← floor(rnd(rows(R)))
       | key

S := | for j ∈ 0.. rows(R) − 1
     |   for i ∈ 0.. cols(R) − 1
     |     S_{j,i} ← R_{j,i}
     | for i ∈ 0.. cols(R) − 1
     |   | V ← D_B(R_{key_i, i})
     |   | V_0 ← M_b_i
     |   | S_{key_i, i} ← B_D(V)
     | S
```

key=

	0
0	153
1	155
2	25
3	96
4	159
5	97
6	43
7	59
8	134
9	11
10	99
11	…

R =

	0	1	2	3	4
148	162	151	160	164	161
149	147	154	153	159	135
150	144	134	84	118	146
151	97	123	177	153	140
152	98	116	83	92	106
153	125	129	144	143	134
154	113	105	92	101	107
155	132	121	105	84	110
156	104	89	97	111	108
157	122	126	110	103	…

S =

	0	1	2	3	4
148	162	151	160	164	161
149	147	154	153	159	135
150	144	134	84	118	146
151	97	123	177	153	140
152	98	116	83	92	106
153	124	129	144	143	134
154	113	105	92	101	107
155	132	120	105	84	110
156	104	89	97	111	108
157	122	126	110	103	…

$$M\_b1 := \left| \begin{array}{l} \text{for } i \in 0..\text{cols}(R) - 1 \\ \quad \left| \begin{array}{l} V \leftarrow D\_B\left(S_{key_i}, i\right) \\ M\_b1_i \leftarrow V_0 \end{array} \right. \\ M\_b1 \end{array} \right.$$

$M\_b =$

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

$M\_b1 =$

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

Для нотаток

Навчальне видання

Кузнецов Олександр Олександрович
Полуяненко Микола Олександрович
Кузнецова Тетяна Юріївна

ПРИХОВУВАННЯ ДАНИХ У ПРОСТОРОВІЙ ОБЛАСТІ НЕРУХОМИХ ЗОБРАЖЕНЬ ШЛЯХОМ МОДИФІКАЦІЇ НАЙМЕНШ ЗНАЧУЩОГО БІТА

Методичні рекомендації
до лабораторної роботи з дисципліни «Стеганографія»
для студентів спеціальності 125 «Кібербезпека»

Коректор *Н. В. Мазепа*
Комп'ютерне верстання *Л. П. Зябченко*
Макет обкладинки *І. М. Дончик*

Формат 60×84 /16. Ум. друк. арк. 2,84. Наклад 50 пр. Зам № 144/19.

Видавець і виготовлювач
Харківський національний університет імені В. Н. Каразіна,
61022, м. Харків, майдан Свободи, 4
Свідоцтво суб'єкта видавничої справи ДК № 3367 від 13.01.2019
Видавництво ХНУ імені В. Н. Каразіна
Тел. 705-24-32