Електронне наукове фахове видання **"Ефективна економіка"** включено до переліку наукових фахових видань України з питань економіки (Наказ Міністерства освіти і науки України від 29.12.2014 № 1528)
www. economy.nayka.com. ua | № 4, 2018

УДК 004.056, 621.3

В. О. Денисюк,
к. т. н., доцент, доцент кафедри економічної кібернетики,
Вінницький національний аграрний університет, м. Вінниця
А. В. Денисюк,
асистент кафедри програмного забезпечення,
Вінницький національний технічний університет, м. Вінниця
Н. В. Денисюк,
магістр з програмного забезпечення систем, м. Вінниця

ПРОГРАМНА РЕАЛІЗАЦІЯ СТЕГАНОГРАФІЧНОГО АЛГОРИТМУ ЗАХИСТУ ІНФОРМАЦІЇ

V. Denysiuk,
PhD, associate professor, associate professor of economic cybernetics department,
Vinnytsya national agrarian university, Vinnytsya
A. Denysiuk,
assistant of software departmen, Vinnytsya national technical university, Vinnytsya
N. Denysiuk,
software of the automated systems master's degree, Vinnytsya

STEGANOGRAPHY ALGORITHM PROGRAMM REALIZATION OF INFORMATION DEFENCE

Розроблено програмний код реалізації стеганографічного алгоритму за методом LSB. Алгоритм приховує конфіденційну інформацію у файлах зображень. Підвищується рівень захисту інформації від несанкціонованого доступу. Метод використовує приховування даних у молодших бітах пікселів файлів зображень. Алгорим не змінює візуальну якість зображення та ускладнює виявлення факту приховування інформації.

The programm code of stego algorithm realization by LSB-method is worked. An algorithm hides confidential information in the files of images. The level of defence of information rises from an unauthorized division. A method uses the concealment of data in the junior pixels of image files. The algorithm does not change the visual quality of the image and complicates the detection of the fact of concealing information.

Ключові слова: *економічна інформація, стеганографія, LSB-метод, алгоритм, захист інформації, файл зображення, приховування інформації.*

Keywords: *economic information, steganography, LSB- method, algorithm, priv, file of image, concealment of information.*

Постановка проблеми

При зберіганні, передаванні або використанні інформації економічного, технологічного чи державного спрямування, виникає потреба у її приховуванні від сторонніх очей. Чим більша цінність інформації тим більшого захисту вона потребує. Не існує абсолютно надійного способу зашифрувати інформацію. Кращий спосіб захистити її - це приховати сам факт її існування. Цим стеганографія перевершує криптографію [6, 15]. У якості носія таємних даних може використовуватися будь-яка інформація: текст, повідомлення, зображення, файли мультимедіа тощо. В загальному випадку доцільно використовувати слово «повідомлення» бо повідомленням може бути як текст або зображення, так і, наприклад, аудіодані (інформація обробляється за допомогою повідомлень) [3, 8, 14]. Метою роботи є підвищення рівня захисту інформації від несанкціонованого доступу за рахунок приховування її у мультимедійних файлах.

Аналіз останніх досліджень і публікацій

Прогрес розвитку глобальних комп'ютерних мереж і засобів мультимедіа привів до розробки нових методів безпеки передачі даних по каналах телекомунікацій і використання їх в різноманітних цілях. Ці методи, враховуючи природні неточності пристроїв дискретизації і надмірність аналогового відео або аудіо сигналу, дозволяють приховувати повідомлення в комп'ютерних файлах (контейнерах). Досягнення науково-технічного прогресу дозволили стеганографії зайняти певну нішу у галузі кодування інформації та її приховування, з'явився такий напрям в області захисту інформації, як комп'ютерна стеганографія (фактично приховування одного файлу в іншому) [2-6].

Методи, що використовують графічні файли, є найбільш вигідними і перспективними, оскільки їх використання немає обмежень по об'єму переданих даних, все залежить від підібраного контейнеру. У якості основного стегано-алгоритму доцільно використовувати LSB-алгоритм, що дозволяє зробити таку заміну, яка в загальному випадку не помітна людському оку. Більш того, багато старих пристроїв виведення, навіть не зможуть відобразити такі незначні зміни. Також можна змінювати не тільки два молодших біта, але і будь-яку їх кількість. Тут є наступна закономірність - чим більшу кількість бітів замінюється, тим більший обсяг інформації можна заховати, але тим більші завади будуть у вихідному зображенні [6, 7, 12].

Постановка завдання

Завдання стеганографії – це приховування певного повідомлення в іншому таким чином, що неможливо побачити присутність або сенс прихованого повідомлення [2,15]. У якості контейнерів зазвичай використовуються цифрові зображення, аудіо- і відео-файли [2, 7, 15]. Результат вбудовування – стего – передається по каналу зв'язку, який контролює порушник. Основна задача порушника полягає у визначенні наявності вбудованої інформації в прехопленому цифровому об'єкті [1, 7, 15]. Реалізація стеганографічного алгоритму захисту інформації з використанням файлів зображень може бути використана при зберіганні, передачі та використанні фінансово-економічної, технічної, соціальної інформації, великих об'ємів графічної інформації з штучних супутників Землі. Крім того, необхідно приділити увагу аналізу роботи стеганографічного алгоритму захисту інформації та його тестуванню, розробці дружнього користувацького інтерфейсу для використання отриманих результатів в практичних задачах.

Викладення основного матеріалу

Існують спеціальні алгоритми для приховування інформації у цифровому вигляді за допомогою стеганографії [7, 12, 13, 15]. Додавання інформації до наявних файлів (наприклад, мультимедійні) призводить до спотворень, які перебувають нижче порогу чутливості людини, тому це не викликає помітних змін у сприйнятті вхідних файлів. Вбудоване (приховане) повідомлення - повідомлення, вбудовується в контейнер. Контейнер - будь-яка інформація, призначена для приховання таємних повідомлень. Порожній контейнер - контейнер без вбудованого повідомлення; заповнений контейнер або стег - контейнер, що містить вбудовану інформацію. Стеганографічний канал або просто стегоканал - канал передачі стег. Стегоключ або просто ключ - секретний ключ, необхідний для приховування інформації. В залежності від кількості рівнів захисту (наприклад, вбудовування попередньо зашифрованого повідомлення) в стегосистеми може бути один або декілька стегоключів [3, 7, 10, 15].

За аналогією з криптографією, за типом стегоключа стегосистеми можна поділити на системи з секретним ключем та системи з відкритим ключем. У стегосистеми з секретним ключем використовується один ключ, який повинен бути визначений або до початку обміну секретними повідомленнями, або переданий по захищеному каналу. Такий варіант вибору ключа є менш ефективним, оскільки зловмисник, перехопивши цей ключ отримує подальший доступ до даних. У стегосистеми з відкритим ключем для вбудовування і отримання повідомлення використовуються різні ключі, такі, що за допомогою обчислень неможливо вивести один ключ з іншого. Тому один ключ (відкритий) може передаватися вільно по незахищеному каналу зв'язку. Крім того, дана схема добре

працює і при взаємній недовірі відправника і одержувача. Тому вибір такого ключа є більш корисним і забезпечує високий рівень захисту.

Стегосистема має відповідати таким вимогам:

1) властивості контейнера повинні бути модифіковані, щоб зміни неможливо було виявити при візуальному контролі, що визначає якість приховування повідомлення (для безперешкодного проходження стегоповідомлення по каналу зв'язку воно жодним чином не повинно привернути увагу);
2) стегоповідомлення повинно бути стійким до спотворень, в тому числі і до зловмисних (в процесі передачі зображення, звука або використання інших контейнерів можуть відбуватися різні трансформації зі зменшення або збільшення, перетворення в інший формат, ущільнення, в тому числі і з використанням алгоритмів з втратою даних тощо);
3) для збереження цілісності вбудованого повідомлення необхідно використовувати код з виправленням помилок;
4) для підвищення надійності вбудоване повідомлення має бути продубльовано.

Суть методу LSB полягає в заміні останніх значущих бітів у контейнері (зображення, аудіо або відеозапису) на біти приховуваного повідомлення. Різниця між порожнім і заповненим контейнерами повинна бути непомітною для органів чуття людини.

Огляд існуючих методів вирішення задачі показав, що найбільш перспективними за інформаційною ємністю є контейнери у вигляді файлів зображень у форматі BMP. Методи, що використовують графічні файли, є найбільш вигідними і перспективними, оскільки практично не обмежують по об’єму переданих даних, все залежить від обраного контейнеру. Також вони надають високий ступінь захисту. Тому їх і краще використовувати у подальших реалізаціях. Для стеганографічного алгоритму доцільно використовувати LSB-метод та BMP-файли зображення у якості контейнерів [5, 7, 13, 15]. BMP-файл можна умовно поділити на 4 частини:

1) заголовок файлу;
2) заголовок зображення;
3) палітра;
4) саме зображення.

Для поставлених цілей треба знати тільки те, що записано в заголовку. Перші два байти заголовка - це сигнатура BMP, далі в подвійному слові записаний розмір файлу в байтах, наступні 4 байти зарезервовані і повинні містити нулі, в наступному подвійному слові записано зсув від початку файлу до байтів зображення. У 24-бітному BMP-файлі кожен піксель кодується трьома байтами RGB.

Робота спрощеного алгоритму за LSB-методом така:

1) початок роботи програми, отримання вхідних даних – файлу-контейнеру;

2) отримуючи на вході файл з даними їх потрібно певним чином обробити і перетворити в послідовність бітів, для подальшого вбудовування в графічний файл;

3) розбиття отриманих байтів зображення на нові пари, по кілька бітів в кожному каналі; у результаті отримується новий відтінок, дуже схожий на попередній; цю різницю важко помітити навіть при великій площі заливки, і згідно результатів досліджень, заміна двох молодших бітів не сприймається людським оком; також можна зайняти три розряди, але це трошки спотворить якість картинки; даний процес виконується у циклі, щоб рівномірно програти потрібні біти по файлу, і щоб вбудовування не кидалося в очі, це значно підвищує безпеку збереженої так інформації і відносно зберігає якість вхідного контейнера-картинки;

4) при записі/читанні файлу можуть виникнути помилки, слід передбачити уникання і перевірку їх; саме цей крок і буде контролювати процес перевірки;

5) при умові, що при перевірці отримано позитивне значення – знайдено збій, сміття тощо, – буде виконуватися послідовність кроків для її уникнення (п.6) і вже потім повторно відбувається прогін по умові, і повернення до головної частини процесу кодування даних;

6) обробка помилки та вибір шляху її виправлення;

7) при завершені роботи і вбудовуванні всіх необхідних даних у зображення, слід коректно переписати файл-контейнер, передбачити додатковий файл для декодування.

Зворотній процес та отримання даних з файлу має аналогічний вигляд [5].

Вибір середовища розробки та мови програмування. Visual Studio 2010 включає в себе багато мов програмування. Наприклад, в ньому можна писати на C++, C#, Java, які є об'єктно-орієнтованими мовами [9]. Сучасна об'єктно-орієнтована мова пропонує, як правило, наступний обов'язковий набір синтаксичних засобів:

1) оголошення класів з полями (даними - членами класу) і методами (функціями - членами класу);

2) механізм розширення класу (успадкування) - породження нового класу від існуючого з автоматичним включенням всіх особливостей реалізації класу-предка до складу класу-нащадка;

3) поліморфні змінні і параметри функцій (методів), що дозволяють привласнювати одні і ті ж змінні екземпляри різних класів;

4) поліморфна поведінка примірників класів за рахунок використання віртуальних методів.

Більшість мов додають до зазначеного мінімального набору ті чи інші додаткові плагіни:

1) конструктори, деструктори, фіналізатори;
2) властивості;
3) індексатори;
4) інтерфейси (наприклад, в Java використовуються також як альтернатива множинного спадкоємства - будь-який клас може реалізувати скільки завгодно інтерфейсів);
5) перевизначення операторів для класів;
6) засоби захисту внутрішньої структури класів від несанкціонованого використання ззовні.

Зазвичай це модифікатори доступу до полів і методів, типу public, private, також protected, іноді деякі інші.

Отже, з точки зору об'єктно-орієнтованого програмування, наведені вище мови однакові, бо в них представлені подібні методи роботи. Тільки порівнюючи синтаксичні та семантичні характеристики мов, важливо розглянути їх у відповідному контексті. Мови націлені на різні потреби, що означає, що вони вирішують різні проблеми різними способами і використовуються в дуже різних середовищах програмування. Хоча як мови, так і їх середовище копіює характеристики одне одного, вони були сконструйовані для різних потреб, і в цьому можна переконатися, порівнюючи їх характеристики. Мета C++ - потужність і контроль за рахунок складності. Мета Java - мобільність, навіть за рахунок деякої відмови від швидкості, і розподілення програми коли виконується зміст [5, 9].

Тому для виконання поставленої задачі обрано C#, бо він може забезпечити і потрібний контроль і багатофункціональність, за умови розумного підходу до справи, дозволяє організувати зрозумілий для користувача інтерфейс, за допомогою використання форм схожих з формами ОС Windows, знайомої всім звичайним користувачам комп’ютерів.

Метою роботи є реалізація приховування текстової інформації в зображенні за допомогою засобів мови C# платформи Net Framework. Програма повинна здійснювати додавання інформації в обране зображення, а також вилучення з нього прихованої раніше інформації.

Розробка програмних модулів для приховування даних.

Повний текст програмного коду наведено у додатку.

Оскільки робота програми буде основана на обробці бітів інформації, а колір одного пікселя займає три байти, то будуть необхідні методи перетворення байта у біти і бітів у байт:

private BitArray ByteToBit(byte src).

Розміщення інформації в bmp буде таким:

1) піксель 0.0: ознака того, що у файлі є текстова інформація, ознакою є символ «/».
2) пікселі 0.1 – 0.3 – розмір текстової інформації записаної в файл.
3) піксель 0.4 і до кінця файлу – текстова інформація.

Код запису в піксель 0.0 ознаки зашифрованого файлу:

byte [] Symbol = Encoding.GetEncoding(1251).GetBytes("/").

В коді змінної Symbol зберігається код символу «/». Далі цей код перетворюється в масив біт (змінна ArrBeginSymbol). Цей колір пікселю 0.0 зберігається в змінній curColor. Далі кожний з трьох складових кольору пікселя перетворюється в масив біт, потім в червоному кольору змінюються молодші два біта на біти символу «/» і в синьому також змінюються молодші 3 біти на біти символу «/», в зеленому теж змінюються аналогічно три біти.

З трьох нових отриманих кольорів створюється новий колір пікселя (nColor), і встановлюється замість попереднього кольору.

Ознака того, що в файлі є інформація записана в bmp файл. Для зручності роботи в один піксель записується одразу байт інформації. Метод перевірки ознаки того, що в файлі є інформація записана в bmp-файл надано у:

private bool isEncryption(Bitmap scr).

Якщо в пікселі 0.0 записаний символ «/», то функція повертає true, інакше – false.

Метод занесення в файл розміру текстової інформації:

private void WriteCountText(int count, Bitmap src).

В CountSymbols записується кількість символів вихідного тексту. Кожна цифра займає один байт, тому максимальна довжина вихідного тексту: 999-4=955 символів (4 – це один піксель на ознаку присутності інформації в файлі і три пікселя на розмір текстової інформації). При необхідності можна збільшити, взявши пікселі не з 0.1 по 0.3, а з 0.1 по 0.4 тощо. В циклі for кожна цифра кількості вихідного тексту перетворюється в масив біт і записується в молодші пікселі кольору за вже описаним методом.

Метод читання розміру текстової інформації:

private int ReadCountText(Bitmap src).

Для роботи програми необхідно:

- комп’ютер з процесором, тактова частота якого не нижча 1ГГц;
- щонайменше 64 Мб оперативної пам’яті;

- на комп’ютері повинна бути встановлена операційна система Windows XP або вища версія;
- мати на комп’ютері Visual Studio відповідного року (2010);
- файл.txt з вже занесеною інформацією для приховування;
- файли-зображення.bmp (готових контейнерів) для роботи.

Програма сформована як файл StegProject.exe.

Створення зовнішнього доступного інтерфейсу виконано за допомогою Windows Form Visual Studio 2010, і представляє наступне вікно (рис.1):

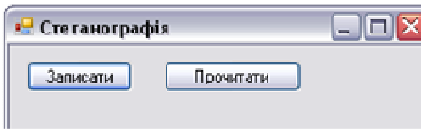

Рис.1. Головне вікно програми.

Користувачеві запропоновано дві кнопки – «Записати» і «Прочитати», які відповідають за кодування і декодування інформації відповідно. При виборі тої чи іншої кнопки, викинеться діалогове вікно, для вибору файлу.

Кодування інформації. При натисненні кнопки «Записати», в діалоговому вікні спочатку пропонується обрати зображення, в яке буде записана інформація (рис.2).

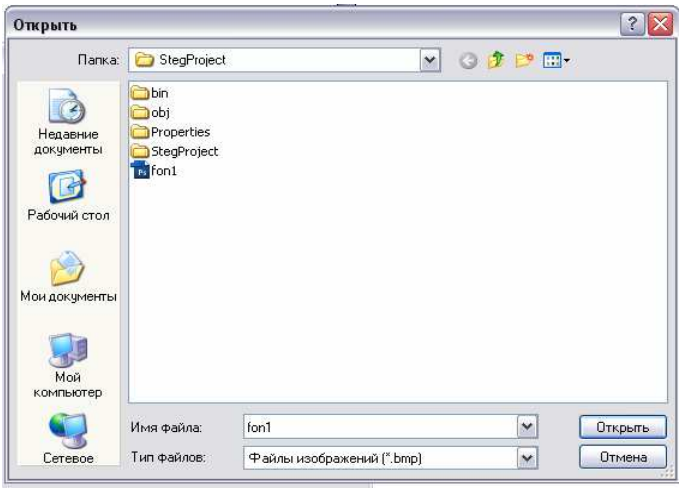

Рис.2. Діалогове вікно вибору зображення контейнеру

Потім програма пропонує обрати файл з інформацією (розширення txt), з якого буде передаватися інформація в зображення і зберігатися (рис.3).

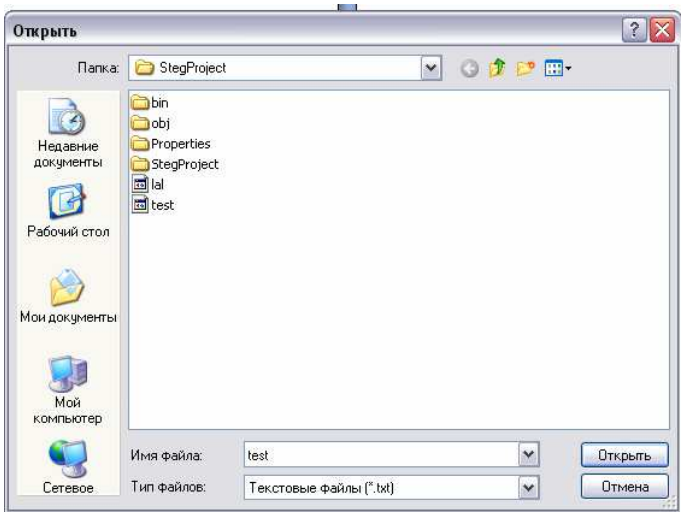

Рис. 3. Діалогове вікно вибору текстового файлу

Після цього необхідно зберегти закодований файл (рис.4).

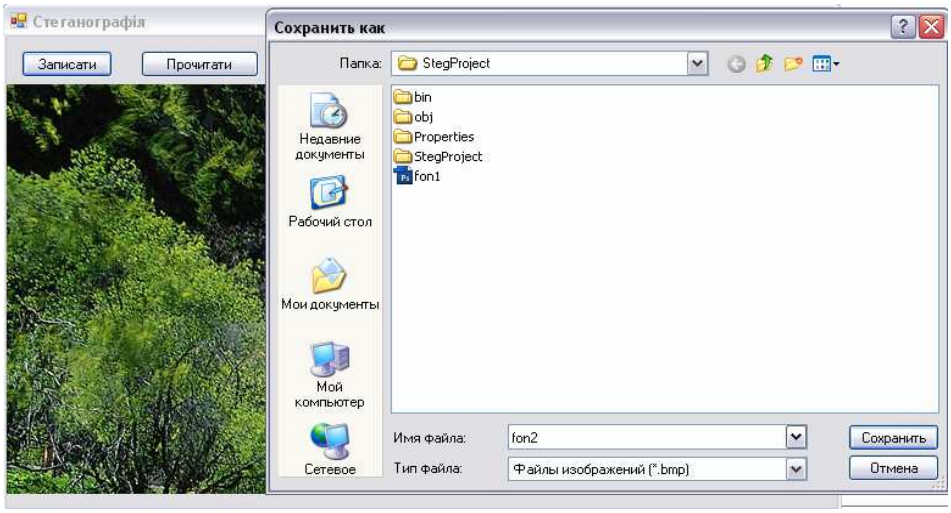

Рис. 4. Збереження зашифрованої інформації

Декодування збереженої інформації з утвореного файлу. Для цього використовуємо «Прочитати», обираємо файл з зображенням з діалогового меню, подібного до того що на рис. 2. Щоб отримати збережену інформацію, необхідно також зберегти її в новому файлі (рис.5).

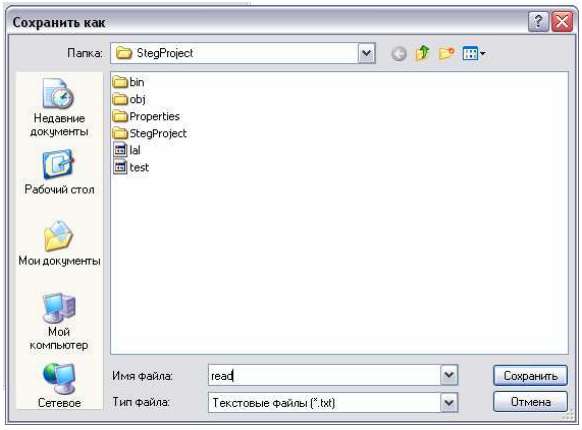

Рис. 5. Збереження витягнутої інформації

Після цього, в результаті успішного виконання, програма виведе нам повідомлення (рис.6):

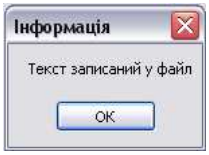

Рис. 6. Результат виконання операції декодування

Після цього можна відкрити збережений файл і дослідити його вміст (рис.7).

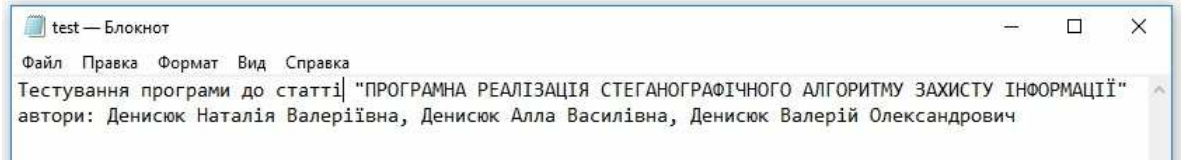

Рис.7. Декодована інформація з файлу

Також, у програмі передбачено випадки виникнення помилок, для таких ситуацій підготовлено також виведення довідкової інформації та повідомлень про помилки. Існує обмеження на розмір контейнеру. Контейнер не має бути надто малим, щоб без завад вмістити інформацію, і зберегти свою якість. Програма при виникненні надлишку шуму чи нестачі простору контейнеру виведе повідомлення про помилку (рис.8).

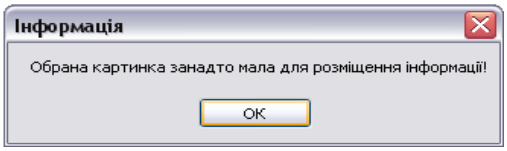

Рис.8. Один з варіантів помилок і попереджень з боку програми, під час кодування/декодування

З метою експериментальної перевірки надійності приховування даних виконано тестування програми приховування даних. При проведенні експериментальних досліджень в якості контейнера використовувалися зображення представлені у форматі *.BMP , з 24-ма розрядами на піксель.

При проведенні експериментальних досліджень розглянуто два зображення Windows 10.bmp наведені на рис.9. Перше з них (рис.9, а) це початкове зображення, яке використовується в якості контейнера, а друге зображення (рис.9, б) містить приховані дані, причому контейнер заповнено на 99%. У якості приховуваної інформації обрано текст: «Тестування програми до статті“ ПРОГРАМНА РЕАЛІЗАЦІЯ СТЕГАНОГРАФІЧНОГО АЛГОРИТМУ ЗАХИСТУ ІНФОРМАЦІЇ” автори: Денисюк Наталія Валеріївна, Денисюк Алла Василівна, Денисюк Валерій Олександрович» (рис.7).

а) б)

Рис. 9. Файл-зображення Windows 10.bmp:
а) початкове зображення; б) контейнер заповнений на 99%.

До запису контейнер (рис.9. а) мав розмір 2764854 байт, після запису (рис.9. б) розмір контейнера становив 2764854 байт, тобто розмір контейнера не змінився. Візуально розрізнити оригінал контейнера та записаний контейнер у процесі дослідження не вдалось. Погіршення якості зображення не спостерігається.

а) б)

Рис. 10. Зображення little.bmp:
а) початкове зображення; б) контейнер заповнений на 99%.

а) б)

Рис. 11. Зображення fon1.bmp:
а) початкове зображення; б) контейнер заповнений на 99%.

Дослідження інших двох зображень (рис.10 та рис.11) дали такі ж результати. Результати тестування наведено у табл.1.

Аналіз наведених результатів показує, що приховування даних не змінює візуальну якість зображення, що свідчить про правильність роботи алгоритму. Швидкість приховування достатньо висока (тести проводились на персональному комп'ютері з ОС Windows 10 та тактовою частотою процесора 2.3 ГГц).

Таблиця 1.
Експериментальні дані по файлам зображень тестування

Назва файлу-контейнеру, bmp	Розмір початкового файлу, байт	Розмір файлу після приховування, байт	Розмір даних, що приховуються, байт	Візуальна оцінка якості після приховування	Час приховування вилучення даних, с
Windows 10	2764854	2764854	190	Відмінна	4
fon1	3932160	3932160	190	Відмінна	6
little	76800	76800	190	Відмінна	2

Висновки

Розроблено алгоритм стеганографічного захисту інформації за методом LSB та реалізовано програмний код, що підвищує рівень захисту інформації від несанкціонованого доступу за рахунок приховування її у мультимедійних файлах, а саме у файлах зображень. Приховування даних відбувається у молодших бітах значення пікселів файлів зображень (по 1-му, 2-м або 3-м бітам приховуваного повідомлення на піксель контейнера). Алгоритм не змінює візуальну якість зображення, що унеможливлює виявлення факту приховування інформації. Програмний код розроблено з використанням обраної мови програмування C# в середовищі Visual Studio 2010. Проведене тестування повністю підтвердило правильність алгоритму та програмного засобу.

Література.

1. Барсуков В.С., Романцов А.П. Компьютерная стеганография вчера, сегодня, завтра [Электронный ресурс]. – Режим доступа: http://www.bnti.ru/showart.asp?aid=330&lvl=03.07.06 (дата звернення 27.03.2018). – Назва з екрана.

2. Денисюк В.О. Алгоритм стеганографічного захисту інформації/ В.О. Денисюк // Матеріали Міжнародної науково-практичної інтернет- конференції «Майбутнє економіки в епоху інформаційного суспільства» (11 квітня 2017 р.). - Вінниця: ВНАУ, 2017. – С.59-60.

3. Денисюк В.О. Дослідження стеганографічних засобів захисту інформації/ В.О. Денисюк // Матеріали Міжнародної науково-практична конференція «Інституціональні ринкові трансформації у розвитку аграрного сектору економіки» (16 - 17 лютого 2017 р.). - Вінниця: ВНАУ, 2017. – С.112-115.

4. Денисюк В.О. Захист інформації з використанням стеганографічних алгоритмів / В.О. Денисюк // Матеріали 2 Всеукраїнської інтернет-конференції «Професійна підготовка фахівця в контексті потреб сучасного ринку праці» (28 лютого 2017 р.). - Вінниця: ВНАУ, 2017. – С.156-159.

5. Денисюк В.О.Стеганографічний алгоритм захисту даних з використанням файлів зображень / В.О.Денисюк// Ефективна економіка. – 2017. –№ 5.– Режим доступу до журналу : http://www.economy.nayka.com.ua.

6. Денисюк В.О. Стеганографічний захист даних з використанням файлів зображень/ Денисюк В.О., Денисюк А.В.// Матеріали І Всеукраїнської науково-практичної інтернет-конференції «Моделювання та кількісні методи в економіці» (27 жовтня 2016 року, м. Вінниця).– Вінниця, ВНАУ, 2016. – С.140-143.

7. Конахович Г.Ф. Компьютерная стеганография. Теория и практика/ Г.Ф.Конахович, А.Ю.Пузыренко.– К.: МК-Пресс, 2006. – 288 с.

8. Кузьмин И. В. Основи теории информации и кодирования/ И.В.Кузьмин, В.А.Кедрус. – 2-е изд., перераб. и доп. – К.: Вища шк. Головное изд-во, 1986.– 238 с.

9. Программирование для MS Windows. [Электронный ресурс]. – Режим доступа: http://abc.vvsu.ru/Books/Prog_win/page0024.asp (дата звернення 27.03.2018). – Назва з екрана.

10. Стеганография [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Стеганография#.D0.9C.D0.B5.D1.82.D0.BE.D0.B4_LSB. (дата звернення 27.03.2018). – Назва з екрана.

11. Стеганозавр [Электронный ресурс]. – Режим доступа: http://www.gagin.ru/internet/8/12.html (дата звернення 27.03.2018). – Назва з екрана.

12. Ярмолик С.В. Стеганографические методы защиты информации/ С.В.Ярмолик, Ю. Н. Листопад // Информатизация образования. – 2005. – N 1. – С. 64-74.

13. Cox I. Digital Watermarking and Steganography/ Cox Ingermar, Miller Matthew, Bloom Jeffrey, Fridrich Jessica, Kalker Ton. – London: Elsevier, 2008. – 593 p.

14. Shannon C. E. A Mathematical Theory of Communication // Bell System Technical Journal. — 1948. — Т. 27. — С. 379—423, 623—656.

15. Wayner P. Disappearing Cryptography: Information Hiding: Steganography and Watermarking / Wayner Peter.- London: Elsever, 2009. – 440 p.

References.

1. Barsukov, V.S. and Romantsov, A.P. (2006), “Computer steganography yesterday, today, tomorrow”, *Tehnika dlja specsluzhb*, [Online], available at: http://www.bnti.ru/showart.asp?aid=330&lvl=03.07.06 (Accessed 27 March 2018).

2. Denysiuk V.O.(2017), “Steganography Algorithm of Information Defence”, *Materialy Mizhnarodnoi naukovo-praktychnoi internet- konferentsii* [Materials of International Research and Practice Internet-conferences], Majbutnie ekonomiky v epokhu informatsijnoho suspil'stva [Future of economy in the epoch of informative society], Vinnytsya national agrarian university, Vinnytsya, Ukraine, 11 April, pp. 59-60.

3. Denysiuk V.O.(2017), “Steganography Facilities Research of Information Defence”, *Materialy Mizhnarodnoi naukovo-praktychna konferentsiia* [Materials of International Research and Practice conference], Instytutsional'ni rynkovi transformatsii u rozvytku ahrarnoho sektoru ekonomiky [Institutional Market Transformations at the development of the agrarian sector of economy], Vinnytsya national agrarian university, Vinnytsya, Ukraine, 16-17 February, pp. 112-115.

4. Denysiuk V.O.(2017), “Information Defence by Steganography Algorithms”, *Materialy 2 Vseukrains'koi internet-konferentsii* [Materials of 3-rd All Ukraine Internet-conferences], Profesijna pidhotovka fakhivtsia v konteksti potreb suchasnoho rynku pratsi [Professional preparation of specialist is in the context of necessities of modern labour-market], Vinnytsya national agrarian university, Vinnytsya, Ukraine, 28 February, pp. 156-159.

5. Denysiuk V.O.(2017), “Steganography Algorithm of Data Protection by Image Files”, *Efektyvna ekonomika*, [Online], vol . 5, available at: http://www.economy.nayka.com.ua (Accessed 27 March 2018).

6. 4. Denysiuk V.O.(2016), “Steganography Defence by Images Files”, *Materialy I Vseukrains'koi naukovo-praktychnoi internet-konferentsii* [Materials of 1-st All Ukraine Research and Practice Internet-conferences], Modeliuvannia ta kil'kisni metody v ekonomitsi [A design and quantitative methods are in an economy],Vinnytsya national agrarian university, Vinnytsya, Ukraine, 27 October, pp. 140-143.

7. Konahovich, G.F. and Puzyirenko, A.Y. (2006*), Kompyuternaya steganografiya*. *Teoriya i praktika*, [Computer steganography. Theory and practice], MK-Press, Kyiv, Ukraine.

8. Kuz'myn, Y. V. and Kedrus, V.A. (1986), Osnovy teoryy ynformatsyy y kodyrovanyia, Vyscha shkola, Holovnoe yzdatel'stvo, Kyiv, Ukraine.

9. ‘MS Windows Programming”, [Online], available at: http://abc.vvsu.ru/ Books/Prog_win/ page0024.asp (Accessed 27 March 2018).

10. “Steganography”, *Vikipedija*, [Online], available at: https://ru.wikipedia.org/wiki/Стеганография #.D0.9C.D0.B5.D1.82.D0.BE.D0.B4_LSB. (Accessed 27 March 2018).

11. Tihulev, M. (1998), ”Steganosaur”, Zhurnal “Internet”, [Online], vol.98-3(8), available at: http://www.gagin.ru/internet/8/12.html (Accessed 27 March 2018).

12. Yarmolyk, S.V.(2005),“Steganography methods of priv”,*Ynformatyzatsyia obrazovanyia*, vol.1, pp.64-74.

13. Cox, I. Miller, M. Bloom, J. Fridrich, J. and Kalker, T. (2008), *Digital Watermarking and Steganography*, Elsevier, London, UK.

14. Shannon C. E.(1948).“A Mathematical Theory of Communication”, *Bell System Technical Journal*, vol. 27, pp. 379-423, 623-656.

15. Wayner, P. (2009), *Disappearing Cryptography: Information Hiding: Steganography and Watermarking*, Elsevier, London, UK.

Додаток
Текст програми приховування даних

Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;

namespace StegProject
{  static class Program
   {
      /// <summary>
      /// The main entry point for the application.
      /// </summary>
      [STAThread]
      static void Main()
      {
         Application.EnableVisualStyles();
         Application.SetCompatibleTextRenderingDefault(false);
         Application.Run(new Form1());
      }
   }
}
```

Form1.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;
using System.Collections;

namespace StegProject
{  public partial class Form1 : Form
   {   public Form1()
      {  InitializeComponent();
      }

      private BitArray ByteToBit(byte src) {
         BitArray bitArray = new BitArray(8);
         bool st = false;
         for (int i = 0; i < 8; i++)
         {  if ((src >> i & 1) == 1) {  st = true; }
          else st = false;
            bitArray[i] = st;
         }
         return bitArray;
      }

      private byte BitToByte(BitArray scr) {
         byte num = 0;
         for (int i = 0; i < scr.Count; i++)
            if (scr[i] == true)
               num += (byte)Math.Pow(2, i);
         return num;
      }
            /*перевірка, чи зашифрований файл, повертає true, якщо символ в першому пікселі рівний / інакше false */
      private bool isEncryption(Bitmap scr)
      {
```

```
    byte[] rez = new byte[1];
    Color color = scr.GetPixel(0, 0);
    BitArray colorArray = ByteToBit(color.R); //отримаємо байт кольору і перетворюємо його  в біт
    BitArray messageArray = ByteToBit(color.R); ;//ініціалізація масиву бітів результату
    messageArray[0] = colorArray[0];
    messageArray[1] = colorArray[1];

    colorArray = ByteToBit(color.G);//отримуємо байт кольору і перетворюємо в масив біт
    messageArray[2] = colorArray[0];
    messageArray[3] = colorArray[1];
    messageArray[4] = colorArray[2];

    colorArray = ByteToBit(color.B);//отримуємо байт кольору і перетворюємо в масив біт
    messageArray[5] = colorArray[0];
    messageArray[6] = colorArray[1];
    messageArray[7] = colorArray[2];
    rez[0] = BitToByte(messageArray); //отримуємо байт символу, записаного в одному пікселі
    string m = Encoding.GetEncoding(1251).GetString(rez);
    if (m == "/")
    {   return true;  }
    else return false;
}

/*метод, занесення в файл розміру текстової інформації */
private void WriteCountText(int count, Bitmap src) {
    byte[] CountSymbols = Encoding.GetEncoding(1251).GetBytes(count.ToString());
    for (int i = 0; i < 3; i++)
    {
        BitArray bitCount = ByteToBit(CountSymbols[i]); //біти кількості символів
        Color pColor = src.GetPixel(0, i + 1); //1, 2, 3 пікселі
        BitArray bitsCurColor = ByteToBit(pColor.R); //біт кольорів поточного пікселя
        bitsCurColor[0] = bitCount[0];
        bitsCurColor[1] = bitCount[1];
        byte nR = BitToByte(bitsCurColor); //новий біт кольору пікселя

        bitsCurColor = ByteToBit(pColor.G);//біт кольорів поточного пікселя
        bitsCurColor[0] = bitCount[2];
        bitsCurColor[1] = bitCount[3];
        bitsCurColor[2] = bitCount[4];
        byte nG = BitToByte(bitsCurColor);//новий колір пікселя

        bitsCurColor = ByteToBit(pColor.B);//біт кольорів поточного пікселя
        bitsCurColor[0] = bitCount[5];
        bitsCurColor[1] = bitCount[6];
        bitsCurColor[2] = bitCount[7];
        byte nB = BitToByte(bitsCurColor);//новий колір пікселя

        Color nColor = Color.FromArgb(nR, nG, nB); //новий колір з отриманих бітів
        src.SetPixel(0, i + 1, nColor); //записати отриманий колір в зображення
    }
}
                                /*читає кількість символів для дешифрування з перших біт картики*/
private int ReadCountText(Bitmap src) {
    byte[] rez = new byte[3]; //масив на 3 елементи, тобто максимум 999 символів шифрується
    for (int i = 0; i < 3; i++)
    {
        Color color = src.GetPixel(0, i + 1); //колір 1, 2, 3 пікселей
        BitArray colorArray = ByteToBit(color.R); //біти кольору
        BitArray bitCount = ByteToBit(color.R); ; //ініціалізація результуючого масиву біт
        bitCount[0] = colorArray[0];
        bitCount[1] = colorArray[1];

        colorArray = ByteToBit(color.G);
        bitCount[2] = colorArray[0];
        bitCount[3] = colorArray[1];
        bitCount[4] = colorArray[2];

        colorArray = ByteToBit(color.B);
        bitCount[5] = colorArray[0];
        bitCount[6] = colorArray[1];
        bitCount[7] = colorArray[2];
        rez[i] = BitToByte(bitCount);
    }
    string m = Encoding.GetEncoding(1251).GetString(rez);
    return Convert.ToInt32(m, 10);
}
                                            /* відкрити файл для шифрування */
private void button1_Click(object sender, EventArgs e)
```

```
{
   string FilePic;
   string FileText;
   OpenFileDialog dPic = new OpenFileDialog();
   dPic.Filter = "Файлы изображений (*.bmp)|*.bmp|Все файлы (*.*)|*.*";
   if (dPic.ShowDialog() == DialogResult.OK)
   {  FilePic = dPic.FileName;
   }
   else
   {  FilePic = "";  return;
   }

   FileStream rFile;  try
   {  rFile = new FileStream(FilePic, FileMode.Open); //відкриваємо потік
   }
   catch (IOException)
   { MessageBox.Show("Помилка відкриття файлу", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
      return;
   }
   Bitmap bPic = new Bitmap(rFile);

   OpenFileDialog dText = new OpenFileDialog();
   dText.Filter = "Текстовые файлы (*.txt)|*.txt|Все файлы (*.*)|*.*";
   if (dText.ShowDialog() == DialogResult.OK)
   {  FileText = dText.FileName;
   }
   else
   {  FileText = "";  return;
   }

   FileStream rText;  try
   { rText = new FileStream(FileText, FileMode.Open); //відкриваємо потік
   }
   catch (IOException)
   { MessageBox.Show("Помилка відкриття файлу", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
      return;
   }
   BinaryReader bText = new BinaryReader(rText, Encoding.ASCII);

   List<byte> bList = new List<byte>();
   while (bText.PeekChar() != -1) { //прочитали весь текстовий файл для занесення в лист байт
      bList.Add(bText.ReadByte());
   }
   int CountText = bList.Count; // в CountText - кількість в байтах тексту, який потрібно закодувати
   bText.Close();
   rFile.Close();

   //перевіряємо, чи вміститься текст в контейнері
   if (CountText > ((bPic.Width * bPic.Height)) - 4 )
   {  MessageBox.Show("Обрана картинка занадто мала для розміщення інформації!", "Інформація", MessageBoxButtons.OK);
      return;
   }

   //перевіряємо, чи вже закодована картинка
   if (isEncryption(bPic))
   {  MessageBox.Show("Файл вже містить інформацію", "Інформація", MessageBoxButtons.OK);
      return;
   }

   byte [] Symbol = Encoding.GetEncoding(1251).GetBytes("/");
   BitArray ArrBeginSymbol = ByteToBit(Symbol[0]);
   Color curColor = bPic.GetPixel(0, 0);
   BitArray tempArray = ByteToBit(curColor.R);
   tempArray[0] = ArrBeginSymbol[0];
   tempArray[1] = ArrBeginSymbol[1];
   byte nR = BitToByte(tempArray);

   tempArray = ByteToBit(curColor.G);
   tempArray[0] = ArrBeginSymbol[2];
   tempArray[1] = ArrBeginSymbol[3];
   tempArray[2] = ArrBeginSymbol[4];
   byte nG = BitToByte(tempArray);

   tempArray = ByteToBit(curColor.B);
   tempArray[0] = ArrBeginSymbol[5];
   tempArray[1] = ArrBeginSymbol[6];
   tempArray[2] = ArrBeginSymbol[7];
   byte nB = BitToByte(tempArray);
```

```
    Color nColor = Color.FromArgb(nR, nG, nB);
    bPic.SetPixel(0, 0, nColor);   //якщо в першому пікселі символ /, картинка вже закодована

    WriteCountText(CountText, bPic); //записуємо кількість символів для кодування
    int index = 0;
    bool st = false;
    for (int i = 4; i < bPic.Width; i++) {
       for (int j = 0; j < bPic.Height; j++) {
          Color pixelColor = bPic.GetPixel(i, j);
          if (index == bList.Count) {
             st = true;
             break;
          }
          BitArray colorArray = ByteToBit(pixelColor.R);
          BitArray messageArray = ByteToBit(bList[index]);
          colorArray[0] = messageArray[0]; //міняємо
          colorArray[1] = messageArray[1]; // в нашому кольорі біти
          byte newR = BitToByte(colorArray);

          colorArray = ByteToBit(pixelColor.G);
          colorArray[0] = messageArray[2];
          colorArray[1] = messageArray[3];
          colorArray[2] = messageArray[4];
          byte newG = BitToByte(colorArray);

          colorArray = ByteToBit(pixelColor.B);
          colorArray[0] = messageArray[5];
          colorArray[1] = messageArray[6];
          colorArray[2] = messageArray[7];
          byte newB = BitToByte(colorArray);

          Color newColor = Color.FromArgb(newR, newG, newB);
          bPic.SetPixel(i, j, newColor);
          index ++;
       }
       if (st) { break; }
    }
    pictureBox1.Image = bPic;

    String sFilePic;
    SaveFileDialog dSavePic = new SaveFileDialog();
    dSavePic.Filter = "Файлы изображений (*.bmp)|*.bmp|Все файлы (*.*)|*.*";
    if (dSavePic.ShowDialog() == DialogResult.OK)
    {  sFilePic = dSavePic.FileName;
    }
    else
    {  sFilePic = "";  return;
    };

    FileStream wFile;  try
    {  wFile = new FileStream(sFilePic, FileMode.Create); //відкриваємо потік на запис результату
    }
    catch (IOException)
    {  MessageBox.Show("Помилка відкриття файлу для запису", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
       return;
    }
    bPic.Save(wFile, System.Drawing.Imaging.ImageFormat.Bmp);
    wFile.Close(); //закриваємо потік
}
                                                    /*відкрити файл для декодування */
private void button2_Click(object sender, EventArgs e)
{
    string FilePic;
    OpenFileDialog dPic = new OpenFileDialog();
    dPic.Filter = "Файлы изображений (*.bmp)|*.bmp|Все файлы (*.*)|*.*";
    if (dPic.ShowDialog() == DialogResult.OK)
    {  FilePic = dPic.FileName;
    }
    else
    {  FilePic = "";  return;
    }

    FileStream rFile;   try
    { rFile = new FileStream(FilePic, FileMode.Open); //відкрити потік
    }
    catch (IOException)
    {
```

```
            MessageBox.Show("Помилка відкриття файлу", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            return;
        }
        Bitmap bPic = new Bitmap(rFile);
        if (!isEncryption(bPic)) {
            MessageBox.Show("У файлі немає зашифрованої інформації", "Інформація", MessageBoxButtons.OK);
            return;
        }

        int countSymbol = ReadCountText(bPic); //порахувати кількість закодованих символів
        byte[] message = new byte[countSymbol];
        int index = 0;
        bool st = false;
        for (int i = 4; i < bPic.Width; i++) {
            for (int j = 0; j < bPic.Height; j++) {
                Color pixelColor = bPic.GetPixel(i, j);
                if (index == message.Length) { st = true;  break; }
                BitArray colorArray = ByteToBit(pixelColor.R);
                BitArray messageArray = ByteToBit(pixelColor.R); ;
                messageArray[0] = colorArray[0];
                messageArray[1] = colorArray[1];

                colorArray = ByteToBit(pixelColor.G);
                messageArray[2] = colorArray[0];
                messageArray[3] = colorArray[1];
                messageArray[4] = colorArray[2];

                colorArray = ByteToBit(pixelColor.B);
                messageArray[5] = colorArray[0];
                messageArray[6] = colorArray[1];
                messageArray[7] = colorArray[2];
                message[index] = BitToByte(messageArray);
                index++;
            }
            if (st) {  break; }
        }
        string strMessage = Encoding.GetEncoding(1251).GetString(message);

        string sFileText;
        SaveFileDialog dSaveText = new SaveFileDialog();
        dSaveText.Filter = "Текстовые файлы (*.txt)|*.txt|Все файлы (*.*)|*.*";
        if (dSaveText.ShowDialog() == DialogResult.OK)
        { sFileText = dSaveText.FileName;
        }
        else
        {  sFileText = "";  return;
        };

        FileStream wFile;  try
        {
            wFile = new FileStream(sFileText, FileMode.Create); //відкрити потік на запис результату
        }
        catch (IOException)
        {
            MessageBox.Show("Помилка відкриття файлу для запису", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            return;
        }
        StreamWriter wText = new StreamWriter(wFile, Encoding.Default);
        wText.Write(strMessage);
        MessageBox.Show("Текст записаний у файл", "Інформація", MessageBoxButtons.OK);
        wText.Close();
        wFile.Close(); //закрити потік
    }
  }
}
```

Стаття надійшла до редакції 27.03.2018 р.