УДК 004.056, 621.3

РЕАЛІЗАЦІЯ СТЕГАНОГРАФІЧНОГО АЛГОРИТМУ ЗАХИСТУ ДАНИХ З ВИКОРИСТАННЯМ ФАЙЛІВ ЗОБРАЖЕНЬ

Денисюк Валерій Олександрович, к.т.н., доцент
Вінницький національний аграрний університет

V. Denysiuk, PhD, Associate Professor
Vinnytsia National Agrarian University

Обрано програмне забезпечення середовища створення програмного коду стеганографічного алгоритму за методом LSB. Розроблено стеганографічний алгоритм за методом LSB. Алгоритм підвищує рівень захисту інформації від несанкціонованого доступу. Алгоритм використовується для втаємничення даних у молодших бітах пікселів файлів зображень. Візуальна якість зображення не змінюється. Виявлення факту приховування інформації ускладнюється. Розглянуто приклад уміщення таємної інформації у файл-контейнер.

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

Рис. 8. Літ. 13.

1. Постановка проблеми

У сучасному світі зростає цінність його інформаційної складової частини. Тому при зберіганні, передачі та використанні фінансово-економічної, технічної або соціальної інформації виникає потреба у її приховуванні від сторонніх очей. Більш цінна інформація потребує більших зусиль з захисту. Інформація обробляється за допомогою повідомлень [3].

Захистити вміст повідомлення можливо блокуванням несанкціонованого доступу до інформації шляхом шифрування повідомлення, або приховуванням повідомлення так, щоб його неможливо було знайти [1, 2, 4, 6, 10-13].

Перший спосіб використовують криптографічні методи захисту, але вони легко автоматично виділяються з інформаційного потоку.

Другий спосіб застосовує стеганографічні методи захисту, які значно знижують ймовірність виявлення таємної інформації. Крім того повідомлення можна ще й додатково зашифрувати. У цьому випадку стеганографія являє собою більш високий рівень захисту інформації у порівнянні з методами криптографії [2, 5, 10-13].

2. Аналіз останніх досліджень і публікацій

Комп'ютерна стеганографія – це приховування одного файлу в іншому.

Нові методи безпеки передачі даних по каналах телекомунікацій дозволяють приховувати повідомлення в комп'ютерних файлах (контейнерах) за рахунок природньої неточності пристроїв дискретизації і надмірність аналогового відео або аудіо сигналу. Стеганографічні системи активно використовуються для вирішення таких основних завдань [1]: захист конфіденційної інформації від несанкціонованого доступу; подолання систем моніторингу та управління мережевими ресурсами; камуфлювання програмного забезпечення; захист авторського права на деякі види інтелектуальної власності. Захист інформації - найефективніша задача, яку розв'язує стеганографія. Можливості стеганографії вражають. Так, наприклад, зміна у 1% оцифрованого звуку (частота дискретизації 44100 Гц, 8-бітний рівень відліку, стерео-режим) дозволяє приховати повідомлення у 10 Кбайт [2].Причому, людина нічого не помітить при прослуховуванні.

У якості контейнера (повідомлення) може використовуватися будь-яка інформація призначена для приховування таємних повідомлень [2]. Повідомленням може бути як текст або зображення, так і, наприклад, аудіодані (файли мультимедіа) тощо. Порожній контейнер - контейнер без вбудованого повідомлення; заповнений контейнер (стег) – контейнер, що містить вбудовану інформацію. Вбудоване (приховане) повідомлення – повідомлення, вбудовується в контейнер [4]. Стеганографічний канал або просто стегоканал - канал передачі стег. Стегоключ або просто ключ - секретний ключ, необхідний для приховування інформації [9]. В залежності від кількості рівнів захисту (наприклад, вбудовування попередньо зашифрованого повідомлення) в стегосистемі може бути один або декілька стегоключів.

Узагальнена модель стегостистеми представлена на рис.1 [8-13].

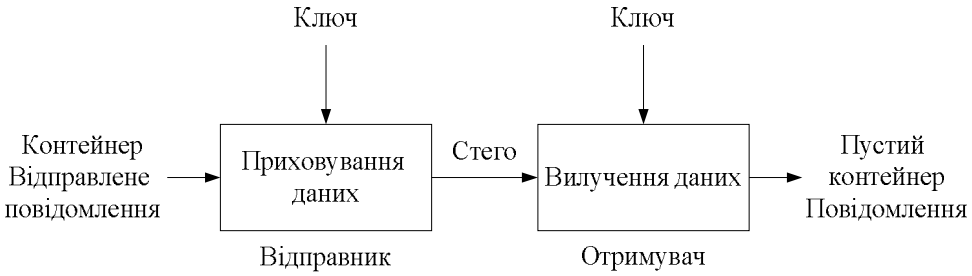

Рис. 1. Модуль стегосистем

За типом стегоключа стегосистеми можна поділяються на системи з секретним ключем та системи з відкритим ключем. У стегосистемі 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 стегосистемі 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 [2].

Найбільш поширеним, але найменш стійким є метод заміни найменших значущих бітів або LSB-метод (Least Significant Bit, найменший значущий біт) [4]. Він полягає у використанні похибки дискретизації, яка завжди існує в оцифрованих зображеннях, аудіо- або відеофайлах. Дана похибка дорівнює найменшому значущому розряду числа, що визначає величину колірної складової елемента зображення (пікселя). Тому модифікація молодших бітів в більшості випадків не викликає значної трансформації зображення і не виявляється візуально.

3. Постановка завдання

За мету в даній роботі було поставлено забезпечення високого рівня інформаційного захисту методом кодування даних у мультимедійних матеріалах. Для розробки програмного алгоритму завершеної системи кодування інформації та її приховування файлі JPG, слід розв’язати такі задачі:

1. Визначити структуру стегомоделі;
2. Обрати конкретний метод;
3. Розробити алгоритм за обраним методом;
4. Обрати програмне забезпечення та апаратну платформу реалізації алгоритма;
5. Написати програмний код;
6. Розробити дружній (зручний) користувацький інтерфейс;
7. Провести тестування алгоритму у цілому.

Додатково на кожному із етапів треба передбачити дії по перевірці чи тестуванню розробленого програмного продукту, а за його результатами – корекцію, налагодження, документування та надання рекомендацій користувачам [4]. Також треба вважати на значний об’єм повної реалізації поставлених цілей та спрямувати зусилля на реалізацію пп. 1-4 та частково п.5 у даній роботі, а повну реалізацію п.5 та пп.6-7 залишити на майбутні дослідження.

4. Дослідження стеганографічного алгоритму

Методи вбудовування прихованої інформації поділяються на такі [1, 2]:

1. Синтаксичні методи вбудовування прихованої інформації в текстові файли;
2. Лексичні методи вбудовування прихованої інформації в текстові файли;
3. Мімікрія;
4. Методи вбудовування прихованої інформації в графічні файли.

Методи, що працюють з графічними файлами, використовують у якості контейнера графічні файли. Ці методи дозволяють вбудовувати не тільки текстову інформацію, але і зображення та інші файли. Єдиною умовою є те, що об’єм прихованого повідомлення не повинен перевищувати розмір зображення-контейнера. Для досягнення цієї мети, кожна програма використовує свою технологію, але всі вони зводяться до заміни певних пікселів у зображенні [7]. Характерним прикладом цієї групи методів є метод LSB [8]. Суть цього методу 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 [9].

Практично в будь-якому типі мультимедіа-файлів після області з даними міститься прапор, після якого може знаходитися службова інформація. Якщо дописати приховуване повідомлення після даного прапора, то при перегляді або прослуховуванні помітити це буде неможливо [10]. Так як, для неущільнених типів файлів легко встановити розмір, знаючи параметри (розподільна спроможність зображень, частота дискретизації, розрядність, тривалість - для звуку), повідомлення за допомогою даного методу краще приховувати в ущільнених форматах (MP3, AVI тощо). Існує ціла множина методів стеганографії, що є простими у реалізації, і які можна використовувати практично до всіх типів файлів мультимедіа. Вони стійкий і мають високу перепускну спроможність, але їх легко помітити і видалити [1, 2, 4, 8].

Огляд існуючих методів вирішення задачі показав, що найбільш перспективними за інформаційною ємністю є контейнери у вигляді файлів зображень у форматі BMP [13]. 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 [6, 7]. Тому їх і краще використовувати у подальших реалізаціях.

У якості стеганографічного алгоритму доцільно використовувати LSB-метод та BMP-файли зображення у якості контейнерів. BMP-файл можна умовно поділити на 4 частини: заголовок файлу, заголовок зображення, палітра, саме зображення [2]. 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. BMP-контейнери поділяються на «чисті» та «зашумлені». У «чистих» файлах-картинках існує зв'язок між молодшим бітом, який змінюється, і рештою 7-ма бітами елементів кольору, а також є істотна залежність самих молодших бітів між собою. Впровадження повідомлення в «чистий» файл руйнує існуючі залежності, що легко виявляється спостерігачем. Якщо файл «зашумлений» (наприклад, зі сканера або фотокамери), то визначити вкладення стає на порядок складніше. Отже, у якості файлів-контейнерів для LSB-методу рекомендується використовувати файли які не були створені на комп'ютері спочатку [2].

Робота спрощеного алгоритму за LSB-методом така [2]:

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 незначно спотворить якість картинки; даний процес виконується у циклі, щоб рівномірно програти потрібні біти по файлу, і щоб вбудовування було непомітним, це значно підвищує як безпеку збереженої інформації так і 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.

На рис.2 представлена граф-схема алгоритму за LSB-методом.

Алгоритм також реалізує зворотній процес та витягнення даних з файлу і 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 [5]. Сучасна об'єктно-орієнтована мова пропонує, як правило, наступний обов'язковий набір синтаксичних засобів:

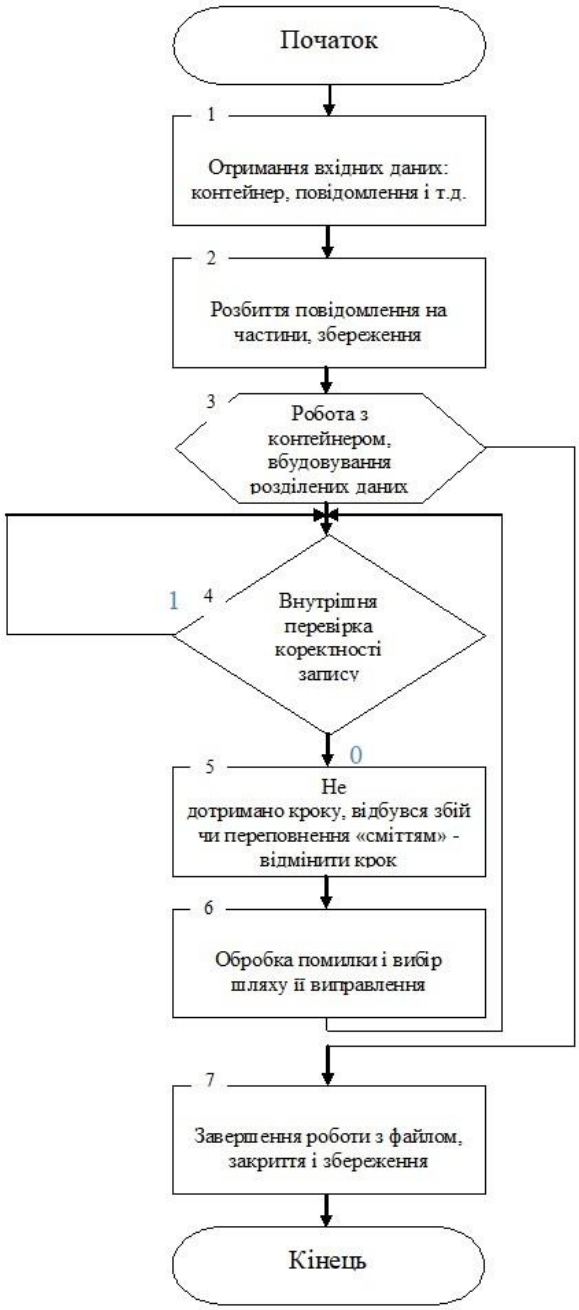

Рис. 2. Граф-схема алгоритму за LSB-методом

1. Оголошення класів з полями (даними - членами класу) і методами (функціями - членами класу) [7];
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 [5];
4. Поліморфна поведінка примірників класів за рахунок використання віртуальних методів.
5. Більшість мов додають до зазначеного мінімального набору ті чи інші додаткові плагіни:
6. Конструктори, деструктори, фіналізатори;
7. Властивості;
8. Індексатори;
9. Інтерфейси (наприклад, в Java використовуються також як альтернатива множинного спадкоємства – будь-який клас може реалізувати скільки завгодно інтерфейсів) [6];
10. Перевизначення операторів для класів;
11. 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]. 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.

6. Розробка програмних модулів для приховування даних

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 (рис.3).

```
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;
      }
```

Рис. 3. Методи перетворення байта у біти і бітів у байт

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.

На рис.4 надано код запису в піксель 0.0 ознаки зашифрованого файлу.

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.

```
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);
```

Рис. 4. Код запису в піксель 0.0 ознаки зашифрованого файлу

Ознака того, що в файлі є інформація записана в bmp файл. Спосіб запису інформації, тобто 2, 3 і 3 біти вибрані для зручності роботи, бо в один піксель записується одразу байт інформації.

Метод перевірки ознаки того, що в файлі є інформація записана в bmp- файл надано на рис.5.

```
private bool isEncryption(Bitmap scr)
    {
        byte[] rez = new byte[1];
        Color color = scr.GetPixel(0, 0);
        BitArray colorArray = ByteToBit(color.R); // отримуємо 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;
    }
```

Рис. 5. Метод перевірки ознаки запису в файл

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

Метод занесення в файл розміру текстової інформації надано на рис.6.

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.

```
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
        }
    }
```

Рис. 6. Метод занесення в файл розміру текстової інформації

Метод читання розміру текстової інформації надано на рис.7.

```
private int ReadCountText(Bitmap src)
    {
        byte[] rez = new byte[3]; //масив на 3 елементи, шифрується
                                    максимум 999 символів
        for (int i = 1; i < 4; i++)
        {
            Color color = src.GetPixel(0, i + 1); //колір 1, 2, 3 пікселів
            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);
    }
```

Рис. 7. Метод читання розміру текстової інформації

Для роботи алгоритму 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.

Було виконано дослідження з метою експериментальної перевірки надійності приховування даних.

У якості приховуваної інформації обрано текст: «Тестування програми до статті за темою «СТЕГАНОГРАФІЧНИЙ ЗАХИСТ ДАНИХ З ВИКОРИСТАННЯМ ФАЙЛІВ-ЗОБРАЖЕНЬ» автора к.т.н., доцента Денисюка Валерія Олександровича».

При проведенні експериментальних досліджень в якості контейнера використовувалися зображення представлені у форматі *.BMP розрядністю 24 розряди на піксель. Обрано стандартне зображення з ОС Windows Безмятежность.bmp. На рис.8 надано два зображення Безмятежность.bmp:

а) це початкове зображення, яке використане у якості контейнера;

б) зображення, що містить приховані дані, причому, контейнер заповнено на 99%.

а) б)

Рис. 8. Файл-зображення "Безмятежность.bmp": а) початкове зображення; б) контейнер заповнений на 99%

До запису Пакета контейнер мав розмір 1440054 байт, після запису розмір контейнера становив 1440054 байт, тобто розмір контейнера не змінився Розмір даних, що приховуються, становить 158 байт. Візуально розрізнити оригінал контейнера та записаний контейнер у процесі дослідження не вдалось. Погіршення якості зображення не спостерігається. Час приховування (вилучення) даних, - 5 с.

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 XP та тактовою частотою процесора 1.8 Гц).

7. Висновки

Аналіз існуючих методів вирішення задачі показав, що найбільш перспективними за інформаційною ємністю є контейнери у вигляді файлів зображень у форматі *.BMP. Отримав подальший розвиток метод приховування даних у файлах зображень, відмінністю якого є приховування даних в молодших бітах значень пікселів, що не змінює візуальну якість зображення і унеможливлює тим самим виявлення факту приховування. Аналіз середовищ та мов програмування показав, що найбільш перспективним для розробки програми приховування даних є середовище програмування Microsoft Visual Studio і мова програмування C#, оскільки цим середовищем забезпечуються умови безпечного виконання програм і автоматизація проектування інтерфейсу користувача. З використанням мови програмування C# в середовищі Visual Studio 2010 розроблено алгоритм для виконання приховування даних за методом LSB.

Запропонована реалізація стеганографічного алгоритму захисту даних з використанням файлів зображень може бути використана при зберіганні, передачі та використанні фінансово-економічної, технічної, соціальної інформації. Добре підійде при приховуванні даних у процесі передавання великих об’ємів графічної інформації з штучних супутників Землі (знімки поверхні планети, метеорологічні мапи тощо) у різноманітних геоінформаційних системах.

У якості перспективних розробок увага буде приділена аналізу роботи стеганографічного алгоритму захисту даних з використанням файлів зображень та його тестування, розробці дружнього користувацького інтерфейсу для використання отриманих результатів з метою втаємничення різноманітної економічної, фінансової або технічної інформації.

Список використаних джерел

1. Барсуков В.С., Романцов А.П. Компьютерная стеганография вчера, сегодня, завтра [Электронный ресурс]. – Режим доступу: http://www.bnti.ru/showart. asp?aid=330&lvl=03.07.06 (дата звернення 11.01.2018). – Назва з екрану.
2. 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. (дата звернення 11.01.2018). – Назва з екрана.
3. 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.
4. Конахович Г.Ф. Компьютерная стеганография. Теория и практика/ Г.Ф. Конахович, А.Ю. Пузыренко.– К.: МК-Пресс, 2006. – 288 с.
5. Программирование для MS Windows. [Электронный ресурс]. – Режим доступа: http://abc.vvsu.ru/Books/Prog_win/page0024.asp (дата звернення 11.01.2018). – Назва з екрана.
6. Стеганография [Электронный ресурс]. – Режим доступу: https://ru.wikipedia.org/wiki/ Стеганография#.D0.9C.D0.B5.D1.82.D0.BE.D0.B4_LSB. (дата звернення 11.01.2018). – Назва з екрану.
7. Стеганография в GIF [Электронный ресурс]. – Режим доступу: http://habrahabr.ru/post/ 128327 (дата звернення 11.01.2018). – Назва з екрану.
8. Стеганозавр [Электронный ресурс]. – Режим доступу: http://www.gagin.ru/ internet/8/12.html (дата звернення 11.01.2018). – Назва з екрану.
9. Ярмолик С.В. Стеганографические методы защиты информации/ С.В.Ярмолик, Ю. Н. Листопад // Информатизация образования. – 2005. – N 1. – С. 64-74.
10. Cachin C. An information-theoretic model for steganography [Electronic resourse]/ Christian Cachin // [Information and Computation, 2004], vol. 192, pp. 41-56. - Mode of access: http://www.sciencedirect.com/ science/article/pii/ S0890540104000409 (Accessed 11 January 2018). – Title from the screen.
11. Cox I. Digital Watermarking and Steganography/ Cox Ingermar, Miller Matthew, Bloom Jeffrey, Fridrich Jessica, Kalker Ton. – London: Elsevier, 2008. – 593 p.
12. Pfitzmann B. Information Hiding Terminology [Electronic resourse]/ Birgit Pfitzmann// [Information Hiding, 1996], vol. 1174, pp. 347-350. - Mode of access: https://www.bibsonomy.org/bibtex/ 2a2a3c63d0ffb9905c76e9842a2419 cbb/dblp (Accessed 11 January 2018). – Title from the screen.
13. 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 11 January 2018).
[2] Denysiuk V.O. (2017), “Steganography Algorithm of Data Protection by Image Files”, Efektyvnaia ekonomika, [Online], available at: http://www.economy.nayka.com.ua (Accessed 11 January 2018).
[3] Kuz'myn, Y. V. and Kedrus, V.A. (1986), Osnovy teoryy ynformatsyy y kodyrovanyia, Vyscha shkola, Holovnoe yzdatel'stvo, Kyiv, Ukraine.
[4] Konahovich, G.F. and Puzyirenko, A.Y. (2006), Kompyuternaya steganografiya. Teoriya i praktika, [Computer steganography. Theory and practice], MK-Press, Kyiv, Ukraine.
[5] ‘MS Windows Programming”, [Online], available at: http://abc.vvsu.ru/ Books/Prog_win/ page0024.asp (Accessed 11 January 2018).
[6] “Steganography”, Vikipedija, [Online], available at: https://ru.wikipedia.org/ /Стеганография #.D0.9C.D0.B5.D1.82.D0.BE.D0.B4_LSB. (Accessed 11 January 2018).
[7] Tihulev, M. (2011), “Steganography in GIF”, Habrahabr, [Online], available at: http://habrahabr.ru/ post/128327/ (Accessed 11 January 2018).
[8] Tihulev, M. (1998), ”Steganosaur”, Zhurnal “Internet”, [Online], vol.98-3(8), available at: http://www.gagin.ru/internet/8/12.html (Accessed 11 January 2018).
[9] Yarmolyk, S.V. (2005), “Steganography methods of priv”, Ynformatyzatsyia obrazovanyia. – vol. 1, pp. 64-74.
[10] Cachin, C., (2004), “An information-theoretic model for steganography”, Information and Computation, [Online], vol.192, available at: http:// http://www.sciencedirect.com/science/article/pii/ S0890540104000409 (Accessed 11 January 2018).

[11] Cox, I. Miller, M. Bloom, J. Fridrich, J. and Kalker, T. (2008), Digital Watermarking and Steganography, Elsevier, London, UK.
[12] Pfitzmann, B.(1996), "Information Hiding Terminology", Information Hiding, [Online], vol. 1174, available at: https://www.bibsonomy.org/bibtex/2a2a3c63d0ffb9905c76e9842a2419cbb/dblp (Accessed 11 January 2018).
[13] Wayner, P. (2009), Disappearing Cryptography: Information Hiding: Steganography and Watermarking, Elsevier, London, UK.

РЕАЛИЗАЦИЯ СТЕГАНОГРАФИЧЕСКОГО АЛГОРИТМА ЗАЩИТЫ ДАННЫХ С ИСПОЛЬЗОВАНИЕМ ФАЙЛОВ ИЗОБРАЖЕНИЙ

Выбрано программное обеспечение среды создания программного кода алгоритма по методу LSB. Разработан стеганографический алгоритм по методу LSB. Алгоритм підвищує рівень захисту інформації від несанкціонованого доступу. Алгоритм используется для сокрытия данных в младших битах пикселей файлов зображений. Визуальное качество изображения не меняется. Выявление факта сокрытия информации усложняется. Рассмотрен пример помещения тайной информации в файл-контейнер.
Ключевые слова: экономическая информация, стеганография, LSB-метод, алгоритм, защита информации, файл изображения, сокрытие информации.
Рис. 8. Лит. 13.

REALISATION OF DATA PROTECTION STEGANOGRAPHY ALGORITHM BY IMAGE FILES

LSB-method algorithm software is decised. Steganography LSB-method algorithm is decised. An algorithm promotes the level of priv from an unauthorized division. The algorithm of concealment of data got further development in the junior bits of pels of files of images. It does not change visual quality of image. The exposure of fact of the information hiding becomes complicated. The example of apartment of private information is considered in a file-container.
Keywords: economic information, steganography, LSB-method, algorithm, priv, file of image, concealment of information.
Fig. 8. Ref. 13.

ВІДОМОСТІ ПРО АВТОРІВ

Денисюк Валерій Олександрович – кандидат технічних наук, доцент кафедри «Економічної кібернетики» Вінницького національного аграрного університету (вул. Сонячна, 3, м. Вінниця, 21008, Україна, e-mail: vad64@i.ua).

Денисюк Валерий Александрович – кандидат технических наук, доцент кафедры «Экономической кибернетики» Винницкого национального аграрного университета (ул. Солнечная, 3, г. Винница, 21008, Украина, e-mail: vad64@i.ua).

Denisyuk Valeriy – PhD, Associate Professor of the Department of Economic Cybernetics, Vinnytsia National Agrarian University (3, Sunny St., Vinnytsia, 21008, Ukraine, e-mail: vad64@i.ua).