МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних наук,
управління та адміністрування
Кафедра інформаційних
технологій

Кваліфікаційна робота бакалавра

на тему: Розробка мобільного додатка для Android з реалізацією методу LSB для стеганографії

Виконав студент групи К-20і
спеціальності 122 Комп'ютерні науки
Нищик Віктор Іванович

Керівник доктор філософії, доцент
Бучинська Ірина Вікторівна

Рецензент д.ф.-м.н., професор
Ковальчук Володимир Володимирович

Одеса 2022

ЗМІСТ

ВСТУП

У роботі представлено техніку приховування даних, засновану на техніці цифрових зображень LSB. Приховування даних є однією з найкращих тем у секретному спілкуванні. У цій статті представлена техніка приховування даних без втрат за допомогою LSB в зображеннях. Техніка приховування даних LSB не впливає на видимі властивості зображення. Стеганографія – це мистецтво і наука приховування факту спілкування. Секрети можуть бути приховані в усіх типах носіїв: тексті, аудіо, відео та зображення. Стеганографія є важливою сферою досліджень останніх років, що включає низку застосувань. Це наука про вбудовування інформації в зображення обкладинки, тобто текст, відео та зображення (корисне навантаження), не спричиняючи статистично значущих змін до зображення обкладинки. Сучасна безпечна стеганографія зображень представляє складне завдання передачі вбудованої інформації до місця призначення без виявлення. У цій статті йдеться про приховування тексту у файлі зображення за допомогою техніки найменшого значущого біта (LSB, Least Significant Bit). Алгоритм LSB реалізований у просторовій області, в якій біти корисного навантаження вбудовуються в найменші значущі біти зображення обкладинки для отримання стегозображення.

Оскільки все більша кількість даних зберігається на комп’ютерах і передається через мережі, не дивно, що стеганографія вступила в цифрову епоху. На комп’ютерах і в мережах програми stego дозволяють комусь приховати будь-який тип двійкових файлів у багатьох інших типах двійкових файлів, хоча графічні та аудіофайли є сьогодні найпоширенішими носіями.

Метою роботи є розробка мобільного додатку на Android для стеганографії з реалізацією метода LSB

Об’єктом роботи є зображення, що містить стеганографію

Предметом дослідження є алгоритм виявлення та вилучення повідомлень, вбудованих у зображення методами LSB

Для виконання роботи необхідно виконати наступні завдання:

– дослідження проблеми забезпечення автентичності зображення;

– аналіз сучасних методів забезпечення автентичності зображень;

– виявлення найефективніших методів для створення власної програми;

– опис алгоритму на основі методу LSB.

Робота складається з 4 розділів, 18 рисунків, 1 таблиці та посилань

1 АНАЛІЗ ТЕХНІКИ СТЕГАНОГРАФІЯ

Використання мультимедійного цифрового сигналу стало дуже популярним в останнє десятиліття через поширення бездротових послуг Інтернету, таких як впровадження систем мобільного зв'язку четвертого покоління, користувач може передавати дані зі швидкістю до 1 Гбіт/с [1]. наявність недорогих інструментів редагування, цифрові дані можуть бути легко скопійовані, змінені та повторно передані в мережі будь-яким користувачем. Щоб ефективно підтримувати розвиток мультимедійних комунікацій, важливо розробити інструменти, які захищають та аутентифікують цифрову інформацію. У цьому внеску ми представляємо нову схему вбудовування, засновану на техніці LSB. [2] Якщо значення пікселя зображення змінено на значення «1», це не впливає на зовнішній вигляд зображення. Ця ідея допомагає нам приховати дані в зображенні.

Стеганографія походить від грецьких слів: Steganos – «покритий», Graphie – «писати». [3] Зазвичай відправник пише нешкідливе повідомлення, а потім приховує секретне повідомлення на тому ж аркуші паперу. Основна мета стеганографії — безпечне спілкування в абсолютно непомітний спосіб і уникнути підозри щодо передачі прихованих даних. Це не для того, щоб інші не дізналися про приховану інформацію, а щоб інші не думали, що ця інформація взагалі існує. Дані можуть бути приховані в основних форматах, таких як аудіо, відео, текст, зображення тощо. Різні типи стеганографії включають:

– стеганографія зображення (процес, у якому приховуються дані всередині зображення, щоб не було видимих змін у вихідному зображенні, звичайний алгоритм стеганографії зображень − це алгоритм вбудовування LSB);

– аудіостеганографія (може застосовуватися до аудіофайлів, тобто приховування інформації в аудіофайлі, це можна назвати аудіостеганографія, аудіофайл повинен бути непомітним);

– стеганографія відео (також може застосовуватися до відеофайлів. відеофайл повинен бути невидимим зловмисником);

– текстові файли (може застосовуватися до текстових файлів)

За видами стеганографії розрізняють:

1. Зображення до зображення – у стеганографії зображення зображення вставляється в інше зображення за допомогою клавіші stego [2]. Текст до зображення: у цьому випадку текст вставляється в зображення і надсилає зображення за допомогою симетричного ключа. Відео в голос: приховування або вбудовування повідомлення у відео схоже на мистецтво приховування інформації, тому що відправник не тільки приховує повідомлення, але й тому, що ніхто, крім одержувача, не відкриває це повідомлення. Приховування повідомлень у відео є частиною мистецтва приховування інформації. Методи стеганографії на основі відео є такими ж, як і зображення. У сучасному світі інформаційна безпека – це чутливий випадок безпеки, і він стає необхідним для захисту даних від підробки. Щоб дані не були підроблені, ми використовуємо цю техніку стеганографії. Захист потрібен, коли дані вже розміщені при передачі

На рис. 1 показано загальну таксономію стеганографічних методів [1,3]:

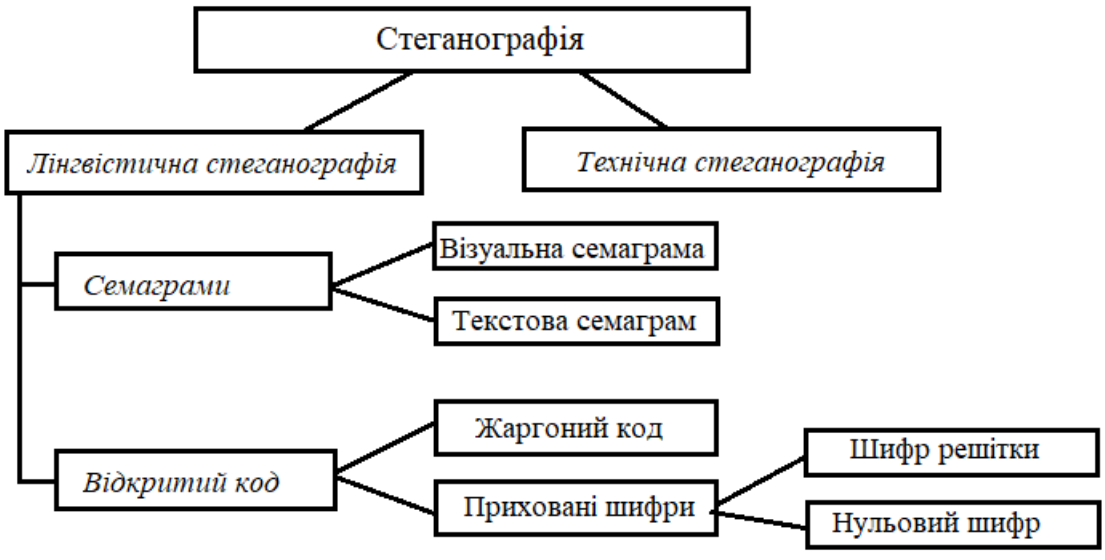

Рисунок 1 – Класифікація методик стеганографії

1) Технічна стеганографія використовує наукові методи, щоб приховати повідомлення, наприклад, використання невидимих чорнил або мікроточок та інші методи зменшення розміру;
2) Лінгвістична стеганографія приховує повідомлення всередині носія деякими неочевидними, та порозділяються на:
   a) Семаграми (приховує інформацію за допомогою символів або знаків):
      – візуальна семаграма (використовує невинні на вигляд або повсякденні фізичні об’єкти, щоб передати повідомлення, такі як каракулі, розміщення елементів на столі чи веб-сайті);
      – текстова семаграма (приховує повідомлення, змінюючи зовнішній вигляд тексту-носія, наприклад, незначні зміни розміру або типу шрифту, додавання додаткових пробілів або різні розмахи літер або рукописного тексту);
   b) Відкритий код (приховує повідомлення в легітимному повідомленні перевізника способами, неочевидними для спостерігачів, які не підозрюю; повідомлення-носій іноді називають відкритим спілкуванням, тоді як приховане повідомлення – прихованим:
      – жаргонічний код (використовує мову, яку розуміє група людей, але не має сенсу для інших, дані коди включають warchalking (символи, що використовуються для вказівки на наявність і тип сигналу бездротової мережі), підземну термінологію або невинну розмову, які передають особливий сенс через факти, відомі лише мовцям; підмножина кодів – це коди підказок, де певні заздалегідь підготовлені фрази передають значення;
      – приховані шифри відкрито приховують повідомлення на носії, для того, щоб його міг відновити кожен, хто знає секрет того, як воно було приховано:

- шифр решітки використовує шаблон, який використовується для покриття повідомлення-носія (слова, що з'являються у отворах шаблону, є прихованим повідомленням);
- нульовий шифр приховує повідомлення згідно з деяким заздалегідь підготовленим набором правил, наприклад «читати кожне п’яте слово» або «дивитися на третій символ у кожному слові».

1.1 Опис методів просторової області

Метод просторової області (Spatial Domain Methods) це техніка стеганографії просторової області яка відноситься до методів, у яких приховування даних виконується безпосередньо на пікселі зображення обкладинки таким чином, що ефект повідомлення не видно на обкладинці. Методи просторової області класифікуються таким чином (рис. 2):

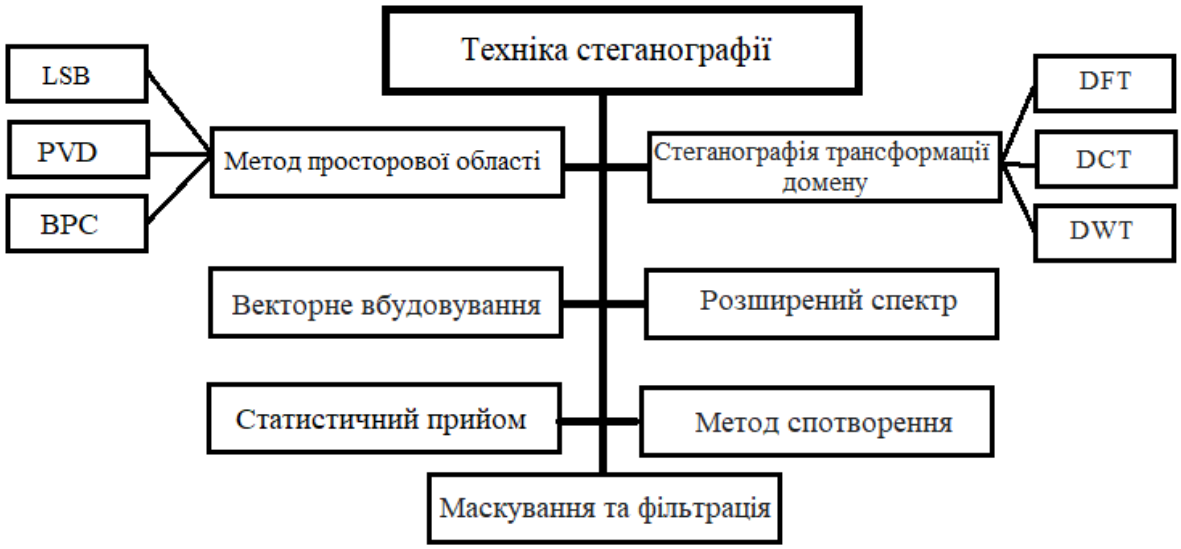

Рисунок 2 – Техніка стеганографії

1.1.1 Опис техніки LSB

LSB є одним із методів методів просторової області, він є простим, але сприйнятливим до стиснення з втратами та маніпуляцій із зображеннями.

Деякі біти змінюються безпосередньо в значеннях пікселів зображення під час приховування даних. Зміни значення LSB непомітні для людських очей.

1.1.2 Опис техніки PVD

Розрізнення значень пікселів (PVD, Pixel Value Differencing) для вбудовування даних вибираються два послідовні пікселі. Метод використовує значення різниці між двома послідовними пікселями в блоці, щоб визначити, скільки секретних бітів має бути вбудовано.

1.1.3 Опис BPC

Складність двійкового шаблону (BPC, Binary Pattern complexity) використовується для вимірювання коефіцієнта шуму в складності зображенн. Шумна частина замінюється двійковим шаблоном і відображається з секретних даних. Зображення залишиться незмінним, коли буде визначено коефіцієнт зворотного шуму.

1.2 Опис стеганографії трансформації домену

Стеганографія трансформації домену (Transform Domain Steganography) це більш складний спосіб приховати інформацію в зображенні. Для приховування інформації в зображеннях використовуються різні алгоритми та перетворення. У частотній області процес вбудовування даних сигналу набагато сильніший, ніж принципи вбудовування, які діють у часовій області. Методи області перетворення над техніками просторової області покликані приховати інформацію в зображеннях, які менше піддаються стисканню, обробці зображень і обрізанню. Деякий домен трансформації не залежать від формату зображення і виконують перетворення форматів без втрат. Методи області перетворення класифікуються за різними категоріями, такими як дискретне перетворення Фур'є, дискретне косинусне перетворення, дискретне вейвлет-перетворення.

1.2.1 Опис дискретного перетворення Фур'є

Дискретне перетворення Фур'є (DFT, Discrete Fourier transform) – це перетворення, яке є суто дискретним: сигнали дискретного часу перетворюються в дискретну кількість частот. DFT перетворює кінцевий список однаково розташованих вибірок функції в список коефіцієнтів кінцевої комбінації складних синусоїд, упорядкованих за їх частотами. Можна сказати, що вона перетворює вибіркова функція з її початкової області часто за часом або положення вздовж лінії в частотну область. При перетворенні Фур’є дискретного часу використовується дискретний час, але воно перетворюється в безперервну частоту. Алгоритм обчислення DFT дуже швидкий на сучасних комп’ютерах. Цей алгоритм відомий як швидке перетворення Фур’є (FFT, Fast Fourier Transform), і він дає той самий результат, що й DFT, використовуючи зворотне дискретне перетворення Фур’є.

1.2.2 Опис дискретно косинусного перетворення

Дискретне косинусне перетворення (DCT, Discrete Cosine Transform) подібне до дискретного перетворення Фур’є, він перетворює сигнал або зображення з просторової області в частотну область. Математичні перетворення перетворюють пікселі таким чином, щоб дати ефект «розповсюдження» розташування значень пікселів на частині зображення. DCT використовується в стеганографії, оскільки зображення розбивається на блоки 8×8 пікселів і перетворює ці блоки пікселів у 64 DCT. Працюючи зліва направо, вгору вниз, DCT застосовується до кожного блоку. За допомогою таблиці квантування кожен блок стискається для масштабування коефіцієнтів DCT, а повідомлення вбудовується в коефіцієнти DCT. Масив стиснених блоків, які утворюють зображення, зберігається в значно зменшеному просторі. За бажанням зображення відновлюється за допомогою декомпресії, процесу, який використовує зворотне дискретне косинусне перетворення, тобто IDCT.

1.2.3 Опис дискретного вейвлет-перетворення

Дискретне вейвлет-перетворення (DWT, Discrete Wavelet transformation) використовується для перетворення зображення з просторової області в частотну область. У процесі стеганографії DWT ідентифікує високочастотну і низькочастотну інформацію кожного пікселя зображення. Це математичний інструмент для ієрархічного розкладання зображення. В основному використовується для обробки нестаціонарних сигналів. Вейвлет-перетворення засноване на малих хвилях, відомих як вейвлети, різної частоти та обмеженої тривалості. Він забезпечує як частотний, так і просторовий опис зображення. Вейвлети створюються шляхом трансляції та розширення фіксованої функції, відомі як материнський вейвлет. DWT виконується в одному вимірі та у двовимірній площині. DWT є точнішою моделлю, ніж DFT або DCT, і це опис зображення з багатьма роздільною здатністю. Сучасний стандарт стиснення зображень JPEG 2000 заснований на вейвлет-перетвореннях [4].

1.3 Опис методу векторного вбудовування

Метод векторного вбудовування (Vector Embedding), який використовує надійний алгоритм зі стандартом кодеків (MPEG-1 і MPEG-2). Цей метод вбудовує звукову інформацію в пікселі кадрів у відео хосту. Він заснований на стандарті кодування відео H.264/AVC. Алгоритм розробив компонент вектора руху для керування вбудовуванням, а також для того, щоб бути секретним носієм. Вбудована інформація не вплине істотно на візуальну та статистичну невидимість відеопослідовності. Алгоритм має велику ємність вбудовування з високим використанням несучої, і може бути реалізований швидко та ефективно [5].

1.4 Опис методу розширеного спектру

Розширений спектр (Spread spectrum) використовує концепцію розширеного спектру. У цьому методі секретні дані поширюються на широку

смугу частот. Співвідношення сигналу до шуму в кожній смузі частот має бути настільки малим, щоб виявити наявність даних було важко. Навіть якщо частини даних видаляються з кількох діапазонів, в інших діапазонах все одно буде достатньо інформації для відновлення даних. Таким чином, важко повністю видалити дані без повного знищення обкладинки. Це дуже надійний підхід, який використовується у військовій комунікації.

1.5 Опис методу статистичного прийому

У техніці повідомлення даний метод вбудовується шляхом зміни кількох властивостей обкладинки. Це передбачає розбиття покриття на блоки, а потім вбудовування одного біта повідомлення в кожен блок. Обкладинка блок змінюється лише тоді, коли розмір біта повідомлення дорівнює один, інакше модифікація не потрібна.

1.6 Методи спотворення

Метод спотворення (Distortion Techniques) використовується для зберігання секретних даних шляхом спотворення сигналу. Кодер застосовує послідовність модифікацій до зображення обкладинки, а фаза декодера декодує зашифровані дані до вихідних даних із секретними даними, використовуючи деякий секретний ключ.

1.7 Опис методу маскування та фільтрація

Маскування та фільтрація (Masking and Filtering) використовується для приховування даних шляхом позначення зображення. Цей підхід є цінним, коли водяні знаки стають частиною зображення. Дані будуть вбудовані в більш значущу частину зображення, а не приховані в шумну частину. Техніки нанесення водяних знаків більш інтегровані в зображення, і їх можна застосовувати без страху знищення зображення. Ця техніка використовується в 24-бітових і сірих зображеннях [6].

В таблиці 1 представлено порівняння різних технік стеганографії

Таблиця 1 – Порівняння різних технік стеганографії.

Техніка	Домен	Невидимість	Ємність	Виявлення	Міцність	Повнота	Коментарі
Найменше значуще біта	Просторовий	Висока	Висока	Висока	Низька	Низька	незалежний від формату і текстури зображення
Розширений спектр	Просторовий	Висока	Низька	Низька	Середня	Середня	розчиняє інформацію на всьому зображенні
Розрізнення значень пікселі	Просторовий	Висока	Середня	Середня	Низька	Низька	Підходить для високо контрастних зображень
Дискретне косинусне перетворення	Перетворений	Висока	Середня	Низька	Середня	Середня	найпростіший в області перетворення
Дискретне перетворення Фур'є	Перетворений	Висока	Середня	Низька	Середня	Середня	передбачає складний розрахунок
Дискретне вейвлет-перетворення	Перетворений	Висока	Середня	Низька	Висока	Висока	тісно відповідає зоровому сприйняттю людини

Фактори, що входять до стеганографії

Ефективність методики стеганографії можна визначити, порівнявши зображення обкладинки зі стегозображенням. Різні фактори:

– надійність (відноситься до здатності вбудованих даних залишатися неушкодженими, якщо стего-зображення зазнає перетворень, таких як лінійні та нелінійні фільтрація, збільшення різкості або розмиття, додавання випадкових шумів, повороти та масштабування, обрізання або децимація, стиснення з втратами);

– непомітність (означає невидимість алгоритму стеганографії. Тому що це перша і головна вимога, оскільки сила стеганографії полягає в її здатності бути непомітною для людського ока);

– частота бітових помилок (приховану інформацію можна успішно відновити з каналу зв'язку; це має бути ідеальним, але для реального каналу зв’язку помилка виникає під час отримання прихованої

інформації, і це вимірюється BER, це відношення кількості помилок до загальної кількості бітів, надісланих у зображенні);

– середня квадратична помилка обчислюється шляхом побайтного порівняння двох зображень; представлення пікселя з 8 бітами та представлення зображень рівня сірого до 256 рівнів; спотворення зображення можна виміряти за допомогою MSE. Нехай I – зображення обкладинки, K – стего-зображення, а m*n – загальна кількість пікселів.

$$MSE = \frac{1}{m\,n}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2 \quad (1)$$

– співвідношення пікового сигналу до шуму система стеганографії зображення повинна вставляти вміст прихованої інформації в зображення, щоб якість зображення не змінилася. PSNR зазвичай використовується для вимірювання якості реконструкції методів стиснення з втратами. Більше значення PSNR вказує на кращу якість зображення, тобто менше спотворення. PSNR – це відношення максимального сигналу до шуму в стего-зображенні [7].

2 ІНСТРУМЕНТИ ДЛЯ ВИКОНАННЯ СТЕГАНОГРАФІЇ

2.1 Переваги стеганографії

Криптографія – це процес, який використовується для перетворення звичайного тексту в зашифрований текст за допомогою симетричного ключа, і цей процес відомий як шифрування. Основним недоліком криптографії є те, що відкритий текст можна знати, а зашифрований текст видно, але ми не можемо його прочитати[8]. Стеганографія – це метод, за допомогою якого звичайний текст приховується в цифровому носії. У цьому процесі порушник не може бачити відкритий текст або зашифрований текст, оскільки він приховується в іншому носії. Порушник не може підозрювати, чи є якісь конфіденційні дані. Для кращого захисту даних у комп’ютерній мережі використовується метод стеганографії.

Процес стеганографії складається з таких етапів:

– секретне повідомлення (дані, які потрібно вставити в цифровий носій);

– Stego-key (ключ, який використовується в процесі стеганографії);

– Cover Media (носій, який використовується в процедурі стеганографії, наприклад, зображення, відео та аудіо);

– алгоритм відправника (техніка, яка використовується в цій стеганографії процес);

– Stego-Media (ЗМІ з'являються через включення таємничого повідомлення в поширений медіа, використовуючи Stego-ключ і обчислення кодування);

– алгоритм приймача (техніка, яка використовується для вилучення таємничого повідомлення зі Stego-media за допомогою Stego-ключа).

Стеганографія має унікальні переваги для агентів мережевого шпигунства. Навіть якщо відомо або підозрюється, що файл містить програмне забезпечення Steganographic, майже неможливо витягти

інформацію, доки не буде отримано правильний пароль. Стеганографія корисна для безпечного зберігання конфіденційних даних, таких як приховування системних паролів або ключів в інших файлах. У місцях, де стандартна криптографія та шифрування заборонені, стеганографія може використовуватися для прихованої передачі даних.

Стеганографія може використовуватися як додаток до криптографії, водяних знаків і відбитків пальців. [9]. Стеганографія може використовуватися для приховування та передачі зашифрованого документа, що містить деяку отриману інформацію у військових програмах.

Але все одно потрібно докласти зусиль, щоб збільшити здатність вбудовування та зберегти секретність. У методах можна приховати текстовий файл, який дорівнює розміру зображення. Можна докласти зусиль, щоб приховати текстові файли, розмір яких перевищує розмір зображення. Секретні ключі повинні бути відомі як відправнику, так і одержувачу. Ключі не надсилаються у вигляді обкладинок, а роздаються окремо. Можна вдосконалити техніку, щоб ці ключі можна було створювати та поширювати приховано. Метод Transform Domain можна використовувати, якщо потрібна додаткова безпека. Якщо стеганографія використовується з криптографією, вона виявиться неперевершеним інструментом у безпечних комунікаційних каналах. Безпека схеми можна підвищити за допомогою передових методів криптографії, а також підвищити ефективність за допомогою методів стиснення даних [10].

Існує багато програм, які пропонують варіанти стеганографії. Деякі пропонують звичайну стеганографію, але деякі пропонують шифрування, перш ніж приховати дані. Деякі можуть приховати дані лише в зображенні, але деякі можуть приховати дані в будь-якому файлі.

В даному розділі описані найкращі інструменти які використовуються для шифрування даних.

2.1 Опис програми Hide'N'Send

Hide'N'Send є одним із найкращих інструментів стеганографії зображень. Він включає шифрування та приховування даних у файлі зображення (формат JPEG), також шифрує дані за допомогою алгоритму стеганографії F5. Приховування даних здійснюється за допомогою алгоритму LSB (найменшого значення) для стеганографії зображення. Замість того, щоб ховатися в структурі файлу, ці алгоритми приховують дані всередині зображення.

Інтерфейс інструмента простий і пропонує дві вкладки (рис. 3) − одна для приховування даних, а інша − для вилучення даних. Є можливість вибрати відповідні параметри. Необхідно просто запустити інструмент, обрати файл зображення, а потім обрати файл, який потрібно приховати, визначивши тип шифрування, і приховати дані в зображенні. Для того, щоб витягти приховану інформацію на зображенні, потрібно використовувати той самий інструмент.

Рисунок 3 − Головне вікно Hide'N'Send

2.2 Опис програмного забезпечення SteganPEG

SteganPEG використовує формат зображення JPEG для виконання стеганографії (рис. 4). Програма може приховати будь-які приватні дані або секретні повідомлення в зображенні JPEG, не змінюючи якість зображення. В

результаті не можна візуально знайти різницю між стеганографічним зображенням і звичайним зображенням, якщо використовувати формат JPEG. Оскільки JPEG забезпечує високоякісне стиснення, він займає менше місця і менше часу для завантаження з Інтернету.

Інтерфейс програмного забезпечення простий з меншою кількістю опцій. Можна легко використовувати цей інструмент, щоб приховати дані у файлі зображення JPG

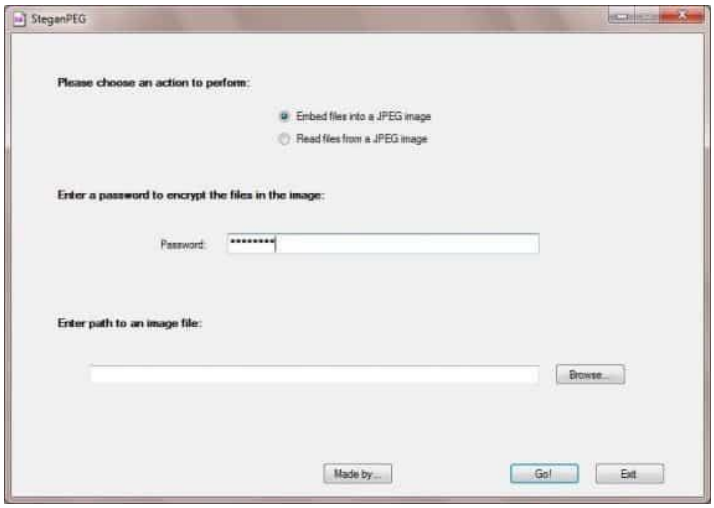

Рисунок 4 − Головне вікно SteganPEG

2.3 Опис програми OpenStego

OpenStego забезпечує приховування даних, а також водяні знаки. Приховування даних є головним пріоритетом, коли справа доходить до стеганографії. Використовуючи OpenStego, можна ефективно виконувати стеганографію з файлами зображень типу JPEG, JPG, BMP, GIF, PNG тощо. Результатом OpenStego є файл PNG. OpenStego безкоштовний інструмент Steganography з відкритим вихідним кодом, розроблений за допомогою Java. Водяний знак дає право надрукувати невидимий водяний знак на файлі зображення. Програма використовується для виявлення несанкціонованих копій файлів зображень (рис. 5).

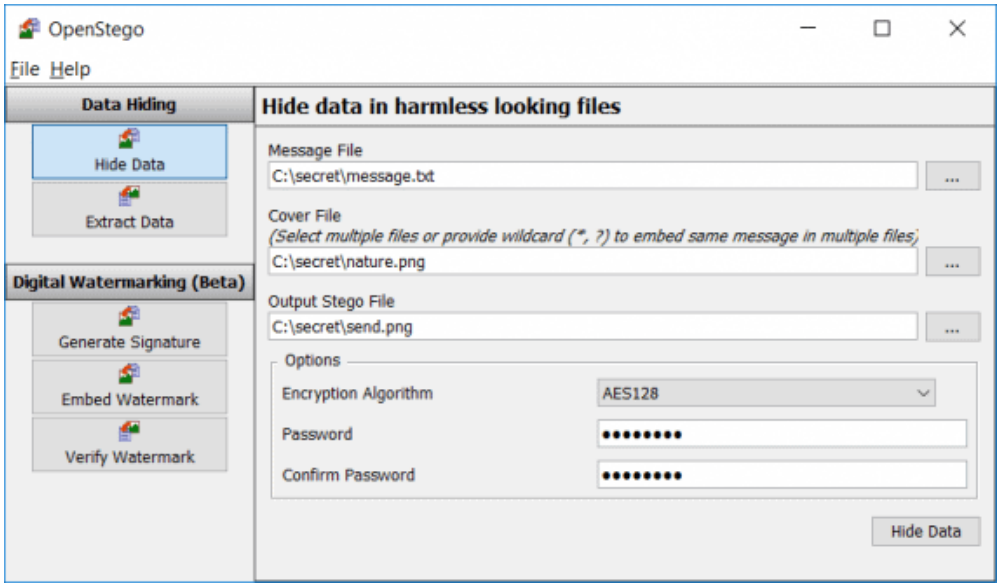

Рисунок 5 – Головне вікно OpenStego

2.4 Опис програмного інструменту Our Secret

Our Secret (рис. 6) – це ще один інструмент стеганографії зображень, який дозволяє приховати файли, текст, повідомлення на фотографіях. Якщо ці файли об’єднати із зображеннями, то кінцеве зображення буде більшого розміру. Але цей інструмент гарантує, що розмір кінцевого зображення не є ненормальним.

Рисунок 6 – Головне вікно Our Secret

Вхідним файлом бажано є файл JPEG із збереженням невеликого розміру зображення. Навіть аудіофайли можна переносити за допомогою файлів зображень за допомогою Our Secret. Використовувати потрібно той самий інструмент, щоб показати дані.

2.5 Опис продукту SS Suite Piscel

SSuite Piscel (рис. 7) є одним із основних продуктів офісного програмного забезпечення SSuite. Це окрема програма, яка є переносною. Це означає, що не потрібно її встановлювати. Piscel використовує зображення як ключ, це означає, що для шифрування не потрібен пароль.

Рисунок 7 − Головне вікно SSuite Piscel

Просто завантажуємо зображення та текст, натискаємо кнопку зашифрувати. Під час дешифрування просто завантажуємо файл зображення в те саме програмне забезпечення для виявлення Steganography, і секретний текст відобразиться в текстовому редакторі. Файли зображень, які використовує Piscel − BMP, JPG, JPEG та WMF.

2.6 Опис програмного інструменту RSteg

RSteg − це ще один інструмент стеганографії, розроблений за допомогою Java. Для запуску RSteg вашому комп'ютері повинна бути встановлена Java**.** Ще одна вражаюча перевага – портативна функція, не потрібно встановлювати його, просто запустіти, і з'явиться вікно програмного забезпечення. Виконати стеганографію за допомогою RSteg (рис. 8) дуже просто − потрібно файл зображення; текст, який потрібно зашифрувати; і пароль для розшифрування. Остаточний результат зберігається у форматі PNG. Підключити зображення необхідно до того ж інструменту виявлення стеганографії для розшифрування разом із паролем [11].

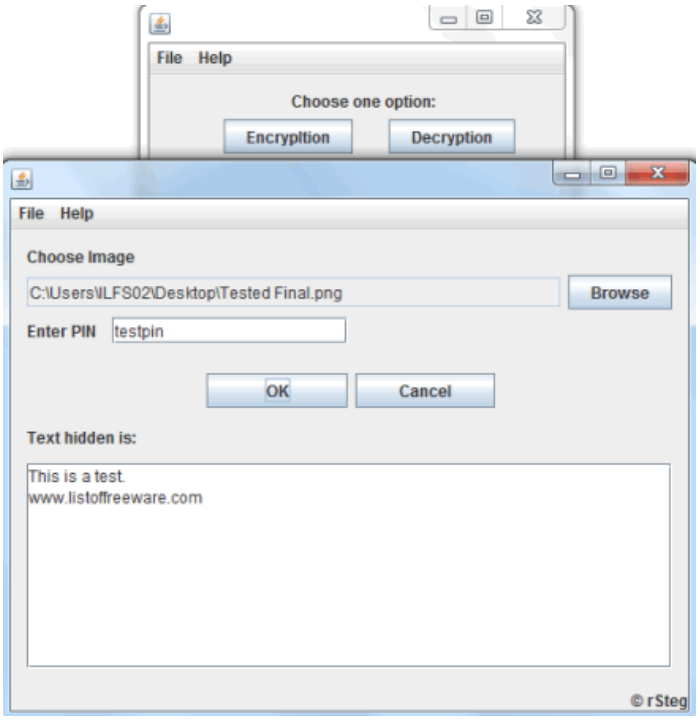

Рисунок 8 − Головне вікно RSteg

2.7 Опис програми Crypture

Crypture використовує інтерфейс командного рядка для виконання стеганографії. Використовуючи Crypture, є можливість приховати конфіденційні дані за допомогою файлів зображень BMP. Але є одна вимога, щоб ця стеганографія відбулася − файл зображення має бути у 8 разів більшим за файл даних, який потрібно зашифрувати. Crypture (рис. 9) дуже малий і має розмір лише 6 КБ, також програма не потребує будь-якої установки.

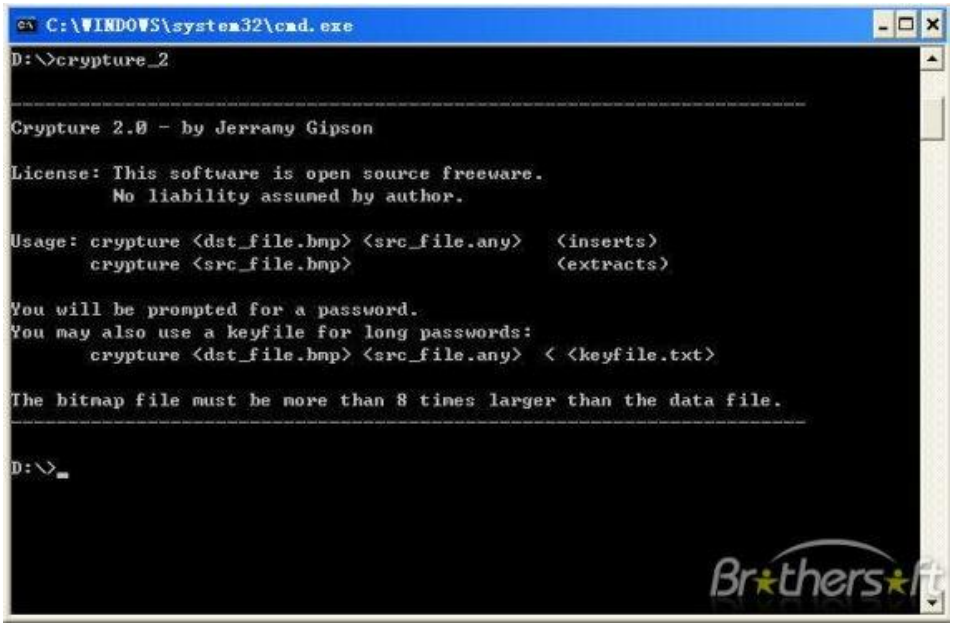

Рисунок 9 − Головне вікно Crypture

2.8 Опис додатку Steghide

Steghide може приховувати зображення та аудіофайли. Основні особливості Steghide включають стиснення, шифрування та автоматичну цілісність. Формати вхідних файлів включають JPEG, BMP, WAV та AU (звуковий файл). Секретні дані не обмежуються текстом, це може бути будь-який тип даних. Використовується тут алгоритм шифрування Rijndael з ключем 128 біт (AES – Advanced Encryption Service). Це програмне забезпечення командного рядка, тому його рекомендують як найкраще програмне забезпечення для стеганографії для користувачів Mac і Linux.

Тому необхідно знати про основні команди для шифрування та дешифрування, щоб відчувати себе комфортно при використанні Steghide (рис. 10).

Цей інструмент був розроблений багато років тому, але досі працює добре. Він працює лише на 32-розрядних версіях Windows.

Рисунок 10 – Головне вікно Steghide

2.9 Опис програми Image Steganography

Інструмент Image Steganography (рис. 11) – це ще один безкоштовний інструмент для виконання надійної стеганографії за допомогою зображень. Можна приховати текстові повідомлення або файли всередині файлу

зображення. Просто обираємо вихідний файл, у якому необхідно приховати секретне повідомлення, а потім о бараємофайл, який потрібно приховати, або написати текстове повідомлення, яке потрібно приховати. Обираємо розташування вихідного зображення і запускаємо, щоб почати кодування файлу.

У закодованому зображенні буде секретне повідомлення всередині зображення. Можна використовувати опцію декодування того ж інструменту, щоб декодувати прихований файл або повідомлення.

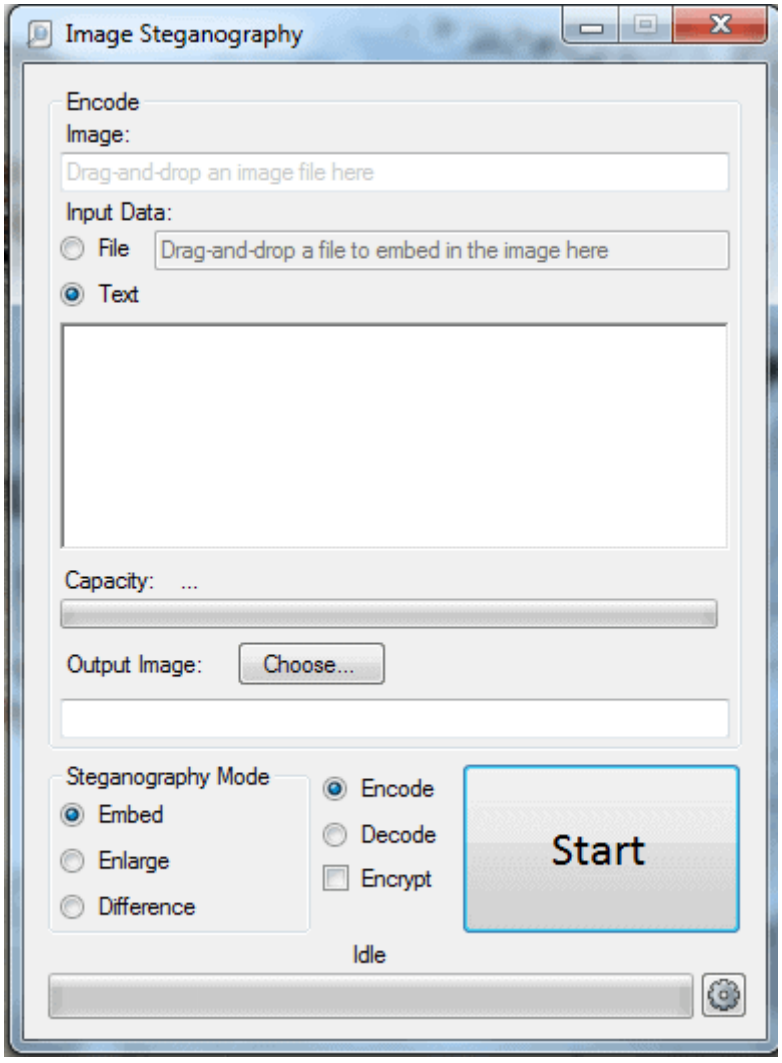

Рисунок 11 − Головне вікно Image Steganography

2.10 Опис інструменту Xiao Steganography

Xiao Steganography – це безкоштовний і найкращий інструмент стеганографії, який можна використовувати для приховування секретних файлів у зображенні, а також аудіофайлів (рис. 12). Найбільш часто використовувані формати файлів – це BMP для зображень і WAV для аудіофайлів. Для роботи відкриваємо інструмент, завантажуємо в нього потрібні файли та секретне повідомлення. В програмі можна вибрати будь-який з наступних алгоритмів шифрування, наприклад DES, DES 112, RC2. Хешування включає SHA, MD4, MD2 і MD5.

Щоб прочитати приховане повідомлення з цього файлу, доведеться знову використовувати це саме програмне забезпечення. Це програмне забезпечення прочитає файл і декодує з нього прихований файл. Не можна розпакувати прихований файл за допомогою будь-якого іншого програмного забезпечення. CNET відомий тим, що пропонує встановлення сторонніх розширень для браузера.

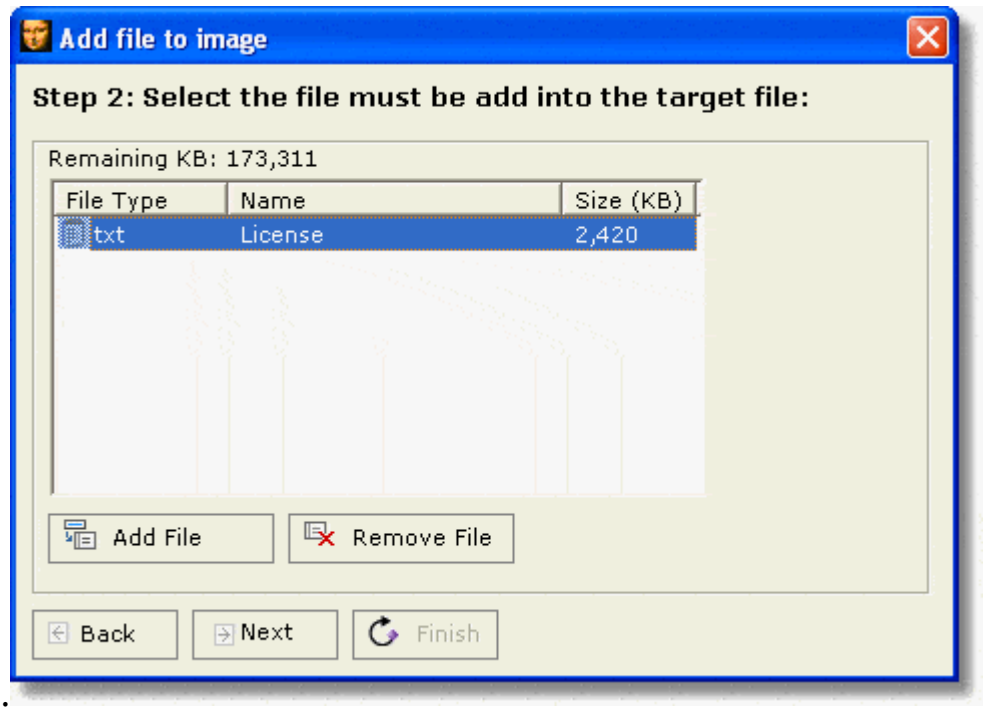

Рисунок 12 – Головне вікно Xiao Steganography

Це кілька інструментів для приховування даних у файлі. Є багато інших безкоштовних інструментів на різних веб-сайтах, що розміщують програмне забезпечення. Однак можна отримати бажані результати від цих інструментів.

Ця техніка була розроблена для безпечного спілкування. Однак злочинці та терористичні організації також почали використовувати це для спілкування. Тому дуже важливо розробити систему виявлення стеганографії у файлі.

Немає простого способу виявити прихований файл у файлі, не можна просто відкрити файл, щоб побачити, чи є щось підозріле, має бути належне спостереження. [11]

3 ПРОГРАМНІ ІНСТРУМЕНТИ ВИКОРИСТАНІ В ДОДАТКУ

3.1 Опис методу LSB

Суть методу заміна найменш значущого біта (Least Significant Bits – LSB) полягає в приховуванні інформації шляхом зміни останніх бітів зображення, які кодують колір на біти приховуваного повідомлення [12]. Різниця між порожнім і заповненим контейнерами повинна бути не відчутна для органів сприйняття людини. Принцип приховування інформації показано на рис. 13.

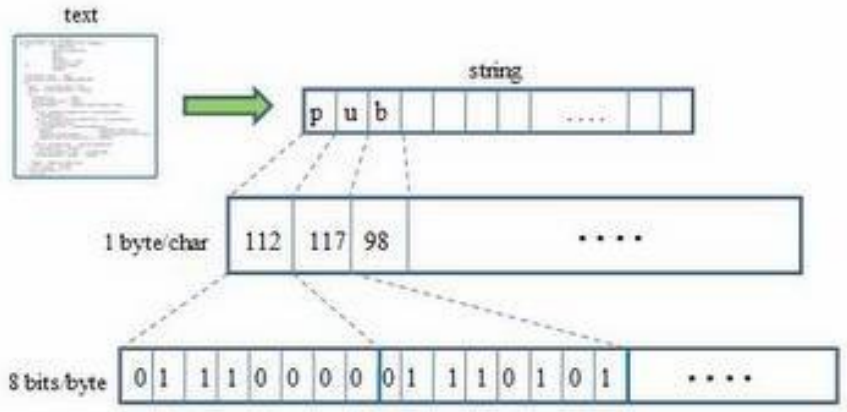

Рисунок 13 – Принцип приховування інформації

Перетворення тексту в байтову послідовність. Як вже описано раніше, в форматі BMP зображення зберігається як матриця значень відтінків кольору для кожної точки зображення, що зберігається. Якщо кожна з компонент простору RGB (їх ще називають каналами кольору) зберігається в одному байті, вона може набувати значень від 0 до 255 включно, що відповідає 24-х бітній глибині кольору. Особливість зору людини полягає в тому, що воно слабо розрізняє незначні коливання кольору. Для 24-х бітного кольору зміна в кожному з трьох каналів одного найменш значимого біта (тобто крайнього правого) призводить до зміни менш ніж на 1% інтенсивності даної точки, що дозволяє змінювати їх непомітно для ока на свій розсуд [12].

Розрахуємо пропускну здатність методу. Якщо відкинути в розрахунках, зазвичай незначне щодо розміру зображення (рис. 14), службову інформацію на початку файлу, то ми маємо можливість потай передати повідомлення розміром в 1/8 розміру контейнера ("розмазати" за останніми бітам в кожному байті матриці кольорів пікселів) або ж розміром в 1/4 контейнери (відповідно при використанні 2 останніх бітів в байтах).

Принцип роботи стеганографічного методу полягає в наступному. Нехай, є 24-х бітове зображення в градаціях сірого. Піксель кодується 3 байтами, і в них розташовані значення каналів RGB.

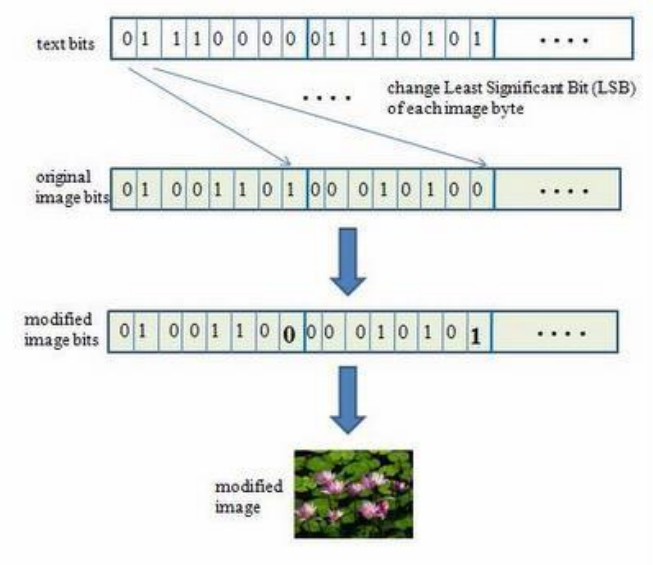

Рисунок 14 – Приховування інформації в зображенні

Змінюючи найменш значущий біт ми міняємо значення байта на одиницю. Такі градації, мало того що непомітні для людини, можуть взагалі не відобразитися при використанні низькоякісних пристроїв виведення.

Наведений нижче приклад показує, як повідомлення може бути приховано в перших восьми байтах, що відносяться до трьох пікселів у 24-бітному зображенні:

Пікселі: (00100111 11101001 11001000)

(00100111 11001000 11101001)

(11001000 00100111 11101001)

A: 01000001

Результат: (0010011 0 1110100 1 1100100 0)

(0010011 0 1100100 0 1110100 0)

(1100100 0 0010011 1 1110100 1)

У прикладі підкреслені тільки біти тільки ті три біта, які були фактично змінені. Застосування стеганографічного методу LSB в середньому вимагає, щоб тільки половина біт зображення-контейнера були змінені.

Невелика модифікація цієї стеганографічної техніки дозволяє використовувати для вбудовування повідомлення два або більш молодших бітів на байт. Це збільшує обсяг прихованої інформації в об'єкті-контейнері, але скритність сильно знижується, що полегшує процес розпізнавання стеганографії. Інші варіації цього методу включають в себе нівелювання статистичних змін в зображенні. Деяке інтелектуальне програмне забезпечення для стеганоаналізу перевіряє області, які складаються з одного суцільного кольору [13]. Для підвищення скритності слід уникнути запису змін в ці пікселі.

Методи LSB є нестійкими до всіх видів атак і можуть бути використані тільки при відсутності шуму в каналі передачі даних. Виявлення LSB-кодованого контейнера здійснюється за аномальними характеристиками розподілу значень діапазону молодших бітів відліків цифрового сигналу.

В якості мови програмування було обрано Java – це об'єктно-орієнтована мова програмування. Мова програмування зародилася як частина проекту створення передового програмного забезпечення для різних побутових приладів.

Java надзвичайно портативна. Одна й та сама програма Java працюватиме однаково на будь-якому комп’ютері, незалежно від апаратних можливостей чи операційної системи, якщо в ній є інтерпретатор Java. Крім портативності, ще однією з ключових переваг Java є набір функцій безпеки, які захищають ПК, на якому запущена програма Java, не тільки від проблем, викликаних помилковим кодом, але й від шкідливих програм (наприклад,

вірусів). Ви можете безпечно запускати аплет Java, завантажений з Інтернету, оскільки функції безпеки Java перешкоджають цим типам аплетів отримати доступ до жорсткого диска ПК або мережевих підключень. Аплет, як правило, є невеликою програмою на Java, яка вбудована в сторінку HTML.

Java можна вважати як компільованою, так і інтерпретованою мовою, оскільки її вихідний код спочатку компілюється у двійковий байт-код. Цей байт-код працює на віртуальній машині Java (JVM), яка зазвичай є програмним інтерпретатором. Використання скомпільованого байт-коду дозволяє інтерпретатору (віртуальній машині) бути невеликим і ефективним (і майже таким же швидким, як і процесор, який виконує рідний скомпільований код). Крім того, цей байт-код надає Java переносимість: він працюватиме на будь-якій JVM, яка правильно реалізована, незалежно від апаратної чи програмної конфігурації комп’ютера. Більшість веб-браузерів (наприклад, Microsoft Internet Explorer або Netscape Communicator) містять JVM для запуску Java-аплетів.

Три ключові елементи об'єдналися в технології мови Java:

– Java надає для широкого використання свої аплети (applets) – невеликі, надійні, динамічні, які не залежать від платформи активні мережеві додатки, що вбудовуються в сторінки Web. Аплети Java можуть налаштовуватися і поширюватися споживачам з такою ж легкістю, як будь-які документи HTML.
– Java вивільняє міць об'єктно-орієнтованої розробки додатків, поєднуючи простий і знайомий синтаксис з надійним і зручним в роботі середовищем розробки. Це дозволяє широкому колу програмістів швидко створювати нові програми і нові аплети.
– Java надає програмісту багатий набір класів об'єктів для ясного абстрагування багатьох системних функцій, використовуваних при роботі з вікнами, мережею і для введення-виведення. Ключова риса цих класів полягає в тому, що вони забезпечують створення незалежних від використовуваної платформи абстракцій для широкого спектра

системних інтерфейсів.

3.2 Основні поняття мови Java

Програми на Java є більш високоструктурованими, ніж еквіваленти C++. Усі функції (або методи Java) та виконувані інструкції в Java повинні перебувати в класі, тоді як C++ дозволяє визначенням функцій і рядкам коду існувати за межами класів. Глобальні дані та методи не можуть перебувати за межами класу в Java, тоді як C++ дозволяє це. Ці обмеження, хоча часом і громіздкі, допомагають підтримувати цілісність і безпеку програм Java і змушують їх бути повністю об’єктно-орієнтованими. Ще однією ключовою особливістю Java є те, що це відкритий стандарт із загальнодоступним вихідним кодом. Sun Microsystems контролює мову Java та пов'язані з нею продукти, але ліберальна політика ліцензування Sun сприяла тому, що Інтернет-спільнота прийняла Java як стандарт.

Java не є процедурною мовою: будь-яка функція може існувати тільки всередині класу. Це підкреслює термінологія мови Java, де немає понять «функція» або «функція-член» (англ. Member function), а тільки метод. В методи перетворилися і стандартні функції. Наприклад, в Java немає функції sin (), а є метод Math.sin () класу Math (що містить, крім sin (), методи cos (), exp (), sqrt (), abs () і багато інших). Конструктори в Java не вважаються методами. Деструкторів в Java не існує, а метод finalize () ні в якому разі не можна вважати аналогом деструктора.

Конструктор – це спеціальний метод, який обов'язково викликається при створенні нового об'єкта, тобто об'єкт (екземпляр класу) не може бути створений без виклику конструктора класу. Не завжди зручно форматувати всі змінні класу при створенні його примірника, тому змінні екземпляра часто оголошують всередині тіла конструктора, а ініціалізують як аргументи конструктора при створенні екземпляра класу. Іноді простіше, щоб якісь значення були б створені за замовчуванням при створенні об'єкта. В такому випадку змінні оголошуються і не започатковано всередині тіла

конструктора.

Конструктор ініціалізує об'єкт безпосередньо під час створення. Ім'я конструктора збігається з ім'ям класу, включаючи регістр, а по синтаксису конструктор схожий на метод без значення, що повертається.

На відміну від методу, конструктор ніколи нічого не повертає.

Конструктор визначає дії, що виконуються при створенні об'єкта класу, і є важливою частиною класу. Як правило, програмісти намагаються явно вказати конструктор. Якщо явного конструктора немає, то Java автоматично створить його (порожнім) для використання за замовчуванням.

Клас – ключове поняття в об'єктно-орієнтованому програмуванні, під яке і заточена Java. Клас визначає зміст і поведінку якоїсь сукупності даних і дій над цими даними. Оголошення класу проводиться за допомогою ключового слова class. Приклад: class <ім'я_класу> {// вміст класу}.

Спадкування – реалізується за допомогою ключового слова extends (class <ім'я_класу> extends <імя_суперкласса>). Класи можуть наслідувати методи і дані один іншого, крім конструкторів і ініціалізаторів. Якщо існують ящик і кімната, обсяг яких обчислюється перемноженням трьох параметрів, то можна визначити материнський клас для двох перерахованих вище класів, щоб в ньому визначити обчислення обсягу, а спадкоємці будуть тільки користуватися успадкованою властивістю, а не переписувати його кілька разів. У той же час при бажанні будь-який із спадкоємців може перевантажити успадковану властивість. Так, наприклад, якщо в кімнаті знаходиться якийсь предмет і обсяг кімнати не повинен включати обсягу цього предмета, то функція обчислення обсягу вже не буде однаковою для ящика і кімнати.

Object – це екземпляр класу. У нашому прикладі це може бути якась конкретна кімната з конкретними розмірами, причому кількість кімнат не обмежена. Припустимо, у нас є два примірника кімнат: спальня і кабінет. Тепер ми можемо, абсолютно не знаючи, з якою кімнатою маємо справу, дізнатися її обсяг, тому що обчислення обсягу – це властивість, яка працює

для будь-якої кімнати.

Interface – описує передбачувану поведінку класу, не згадуючи конкретних дій. Створюється інтерфейс за допомогою ключового слова interface (interface <імя_інтерфейса>). Для того щоб успадкувати (реалізувати) класом інтерфейс, використовується ключове слово implements (class <ім'я_класу> implements <імя_інтерфейса>). А між собою інтерфейси успадковуються все тим же словом extends. Для нашого прикладу можна створити інтерфейс Об'ємний, в якому буде сказано, що клас, який підтримує даний інтерфейс, повинен вміти повертати обсяг. У такому випадку ми можемо сказати, що і Кімната, і Ящик підтримують інтерфейс Об'ємний.

Посилання в Java – покажчики на об'єкти. Іншими словами, посилання – це змінна, яка містить адресу комірки пам'яті, в якій зберігається об'єкт. Крім того, посилання може бути инициализирована як null – нульова посилання, які не вказує ні на який об'єкт в пам'яті (саме це значення є значенням за замовчуванням).

В Java розрізняють змінні примітивного і посилального типу. До примітивним відносять 8 типів – п'ять цілочисельних, два для чисел з плаваючою точкою і логічний тип boolean. Всі інші типи даних відносяться до посилальним типам. Це означає, що оголошена змінна цього типу зберігає не стан об'єкта, а посилання на нього, сам же об'єкт зберігається в певній області пам'яті (купі).

3.3 Засоби і технології розробки

3.3.1 Обґрунтування вибору середовища розробки

Android Studio – це інтегроване середовище розробки (IDE) для роботи з платформою Android.

IDE перебувала у вільному доступі починаючи з версії 0.1, опублікованій в травні 2013, а потім перейшла в стадію бета-тестування, починаючи з версії 0.8, яка була випущена в червні 2014 року. Перша стабільна версія 1.0 була випущена в грудні 2014 року, тоді ж припинилася

підтримка плагіна Android Development Tools (ADT) для Eclipse.

Android Studio, заснована на програмному забезпеченні IntelliJ IDEA від компанії JetBrains, – офіційне засіб розробки Android додатків. Дане середовище розробки доступна для Windows, OS X і Linux. 17 травня 2017, на щорічній конференції Google I/O, Google анонсував підтримку мови Kotlin, використовуваного в Android Studio, як офіційної мови програмування для платформи Android на додаток до Java і C ++.

Програмне забезпечення для Android OS, розробляється за допомогою Android SDK.

SDK (від англ. Software Development Kit) – набір із засобів розробки, утиліт і документації, який дозволяє програмістам створювати прикладні програми за визначеною технологією або для певної платформи (програмної або програмно-апаратної).

Програміст, як правило, отримує SDK безпосередньо від розробника цільової технології або системи. Часто SDK розповсюджується через Інтернет. Багато SDK розповсюджуються безкоштовно для того, щоб заохотити розробників використовувати дану технологію або платформу.

Android SDK – універсальний засіб розробки мобільних додатків для операційної системи Android. Відмінною рисою від звичайних редакторів для написання кодів є наявність широких функціональних можливостей, що дозволяють запускати тестування і налагодження вихідних кодів, оцінювати роботу програми в режимі сумісності з різними версіями ОС Android і спостерігати результат в реальному часі (опціонально). Підтримує велику кількість мобільних пристроїв, серед яких виділяють: мобільні телефони, планшетні комп'ютери, розумні окуляри (в тому числі Google Glass), сучасні автомобілі з бортовими комп'ютерами на ОС Android, телевізори з розширеним функціоналом, особливі види наручних годинників і багато інших мобільні гаджети, габаритні технічні пристосування.

3.3.2 Можливості Android Studio

Візуальний редактор макетів – можливість створювати складні макети з ConstraintLayout, додаючи обмеження з кожного уявлення в інші уявлення і рекомендації. Потім можно переглянути макет на будь-якому розмірі екрану, вибравши одну з різних конфігурацій пристрою або просто змінивши розмір вікна попереднього перегляду.

APK Analyzer – можливість зменшити розмір застосування Android, перевіривши вміст файлу APK додатка, навіть якщо він не був створений за допомогою Android Studio.

Швидкий емулятор – можливість встановлювати та запускати програми швидше, ніж з фізичним пристроєм, а також моделювати різні конфігурації та функції, включаючи ARCore, платформу Google для створення досвіду розширеної реальності.

Інтелектуальний редактор коду – можливість написати кращий код, працювати швидше і працювати більш продуктивно з інтелектуальним редактором коду, який надає код для мов Kotlin, Java та C++ / C.

Гнучка система збірки – створена за допомогою Gradle, система збірки Android Studio дозволяє налаштовувати вашу конструкцію для створення декількох варіантів збірки для різних пристроїв з одного проекту.

Профайлери в реальному часі – вбудовані інструменти профілювання надають статистику реального часу для процесора, пам'яті та мережевої активності вашої програми. Можливість визначати вузькі місця в продуктивності, записуючи сліди методів, перевіряючи купу і розподіли, і бачити вхідні та вихідні корисні навантаження мережі.

3.3.3 Опис технічних засобів для розробки програми

Розширювана мова розмітки (скорочено XML) – запропонований консорціумом World Wide Web стандарт побудови мов розмітки ієрархічно структурованих даних для обміну між різними застосунками, зокрема, через Інтернет. Є спрощеною підмножиною мови розмітки SGML. XML-документ

складається із текстових знаків, і придатний до читання людиною.

Стандарт XML визначає набір базових лексичних та синтаксичних правил для побудови мови описання інформації шляхом застосування простих тегів. Цей формат достатньо гнучкий для того, аби бути придатним для застосування в різних галузях. Іншими словами, запропонований стандарт визначає метамову, на основі якої шляхом запровадження обмежень на структуру та зміст документів визначаються специфічні, предметно-орієнтовані мови розмітки даних. Ці обмеження описуються мовами схем (Schema), такими як XML Schema (XSD), DTD або RELAX NG. Прикладами мов, заснованих на XML, є: XSLT, XAML, XUL, RSS, MathML, GraphML, XHTML, SVG, а також XML Schema.

4 ОПИС ПРОГРАМНОГО ДОДАТКУ

Після запуску додатку на екрані пристрою користувач бачить головний екран (рис. 15).

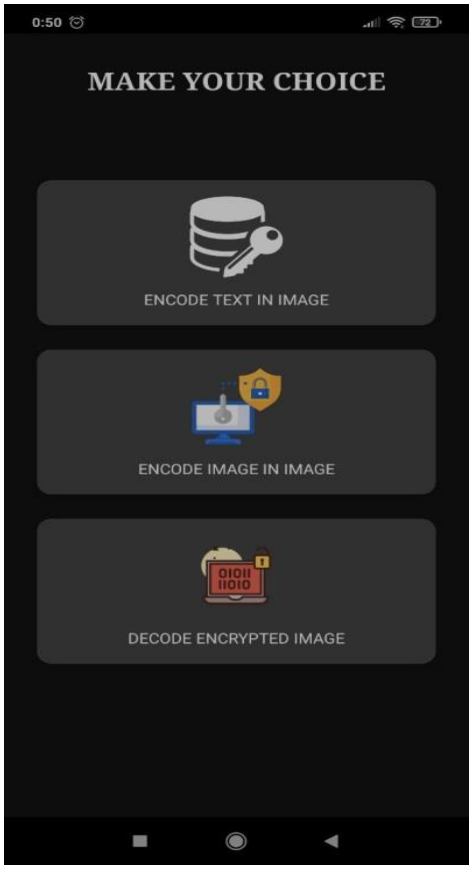

Рисунок 15 – Головний екран додатку

На головному екрані розміщено три кнопки:

– «Encode text in image»;
– «Encode image in image»;
– «Decode encrypted image»

При переході до пункту «Encode text in image» відчиняється екран для вбудовування текстового повідомлення у зображення (рис 16).

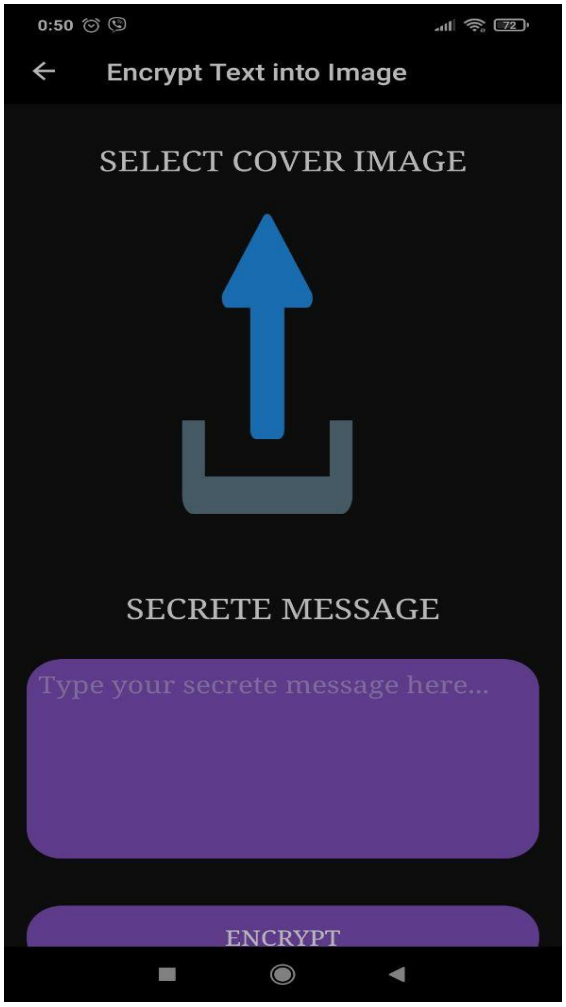

Рисунок 16 – Екран вбудовування текстового повідомлення

Для вбудовування текстового повідомлення у зображення потрібно натиснути на іконку додавання зображення, натиснути «Select Image», обрати встановлений на пристрої додаток (наприклад, «Галерея» чи «Провідник»), та за допомогою обраного додатку вибрати зображення, яке буде контейнером для текстового повідомлення.

В текстове поле необхідно ввести повідомлення, яке необхідно приховати, та натиснути кнопку «Encrypt».

Результат додавання буде показано на відповідному екрані (рис. 17).

Зображення потрібно зберегти на пристрої, натиснувши кнопку «Save in mobile’s database», після чого його можна використовувати у інших додатках чи передавати через мережу.

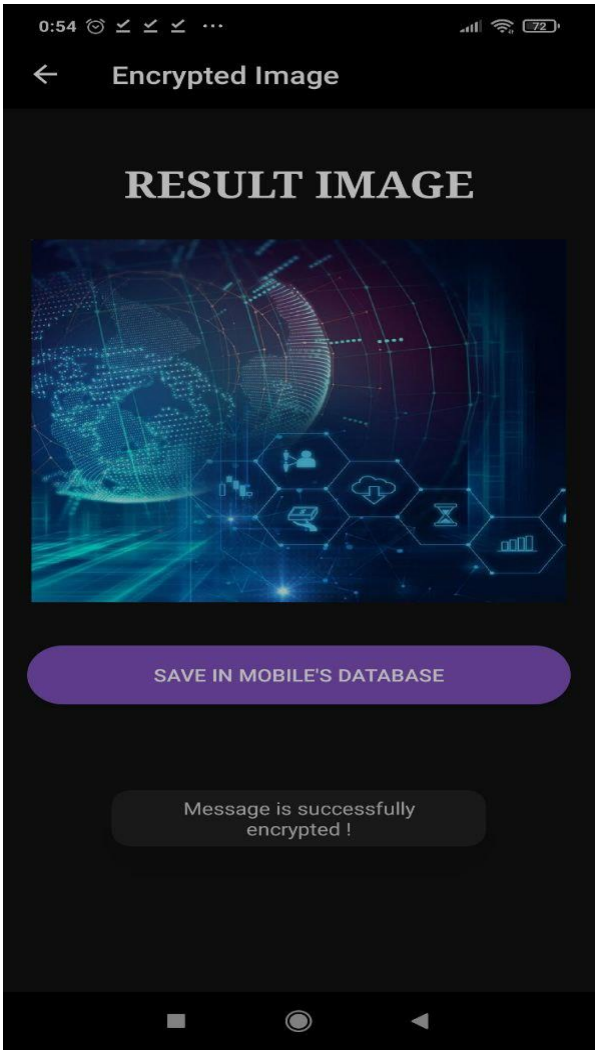

Рисунок 17 – Результат приховання

У пункті головного екрану «Encode image in image» користувачу доступна можливість приховання одного зображення у інше (рис. 18).

Для приховання зображення у зображенні спочатку потрібно обрати зображення контейнер через один зі встановлених на пристрої додатків. Наступним кроком необхідно обрати зображення, яке необхідно приховати та натиснути кнопку «Encrypt».

Після успішного завершення приховання користувач побачить екран на якому буде показано результат приховання та можливість зберегти отримане зображення.

Рисунок 18 – Екран приховання зображення у зображенні

Для виконання операції вилучення інформації із контейнеру, необхідно на головному екрані обрати пункт «Decode encrypted image», після чого на наступному екрані обрати з пристрою зображення і натиснути кнопку «Decrypt the secrete image». Додаток автоматично визначає тип інформації у контейнері. Результат операції вилучення показується на відповідному екрані (рис. 19).

Для реалізації методів LSB створено два класи Embedding та Extracting.

Клас Embedding реалізовує методи для вбудовування таємного тексту або зображення у зображення-контейнер.

У класі Extracting реалізовано методи для вилучення з зображення контейнеру прихованого тексту або зображення.

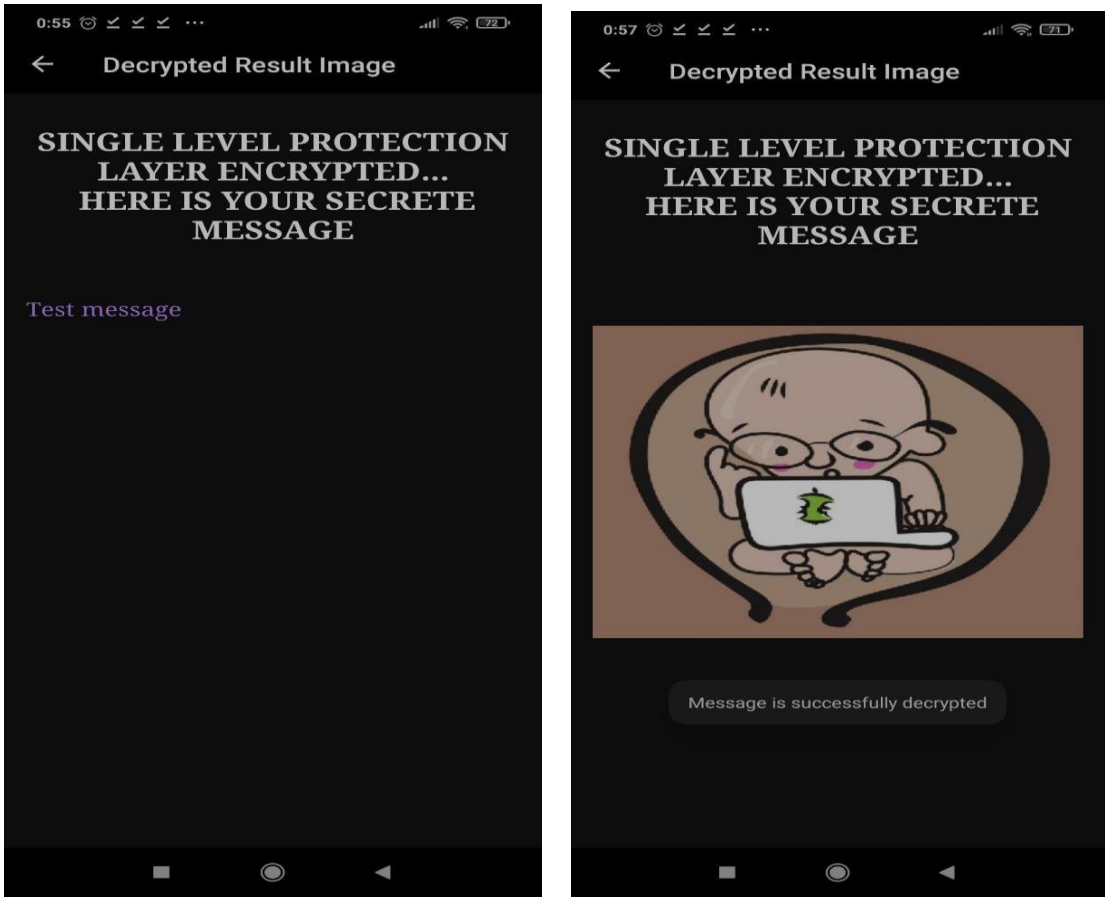

Рисунок 19 – Результати операції вилучення

Клас Embedding налічує 6 статичних методів.

Метод generateKey генерує випадковий масив типу int, що буде включати 24 біта.

```
private static int[] generateKey() {
    final int[] bits = {0, 1};
    int[] result = new int[24];
    int n, i;
    Random random = new Random();
    for (i = 0; i < result.length; ++i) {
      n = random.nextInt(2);
      result[i] = bits[n];
    }
    return result;
  }
```

Методи LSB та LSB2 повертають значення останнього та передостаннього біта переданого в якості параметру значення.

```
private static int LSB(int number) {
    return number & 1;
  }
  private static int LSB2(int number) {
    return (number >> 1) & 1;
  }
```

Метод action порівнює біт кольору та біт повідомлення, що приховується та повертає одне з трьох значень: -1, 0, 1.

```
private static int action(int color, char bit) {
    if (LSB(color) == 1 && bit == '0') {
      return -1;
    } else if (LSB(color) == 0 && bit == '1') {
      return 1;
    } else {
      return 0;
    }
  }
```

Метод embedSecretImage відповідає за вбудовування зображення, що потрібно приховати у зображення контейнер. Метод приймає два аргументи – зображення для приховання, та зображення контейнер. Зображення відчиняться у 32 бітному форматі, тобто кожен піксель займає 32 біти, по 8 біт на 3 канали кольору RGB, та 8 біт на прозорість.

```
    public static Bitmap embedSecretImage(Bitmap coverImage,
Bitmap secretImage) {
        Bitmap stegoImage =
coverImage.copy(Bitmap.Config.ARGB_8888, true);
        stegoImage.setPremultiplied(false);
        String sImageInBin =
HelperMethods.bitmapToBinaryStream(secretImage);
        int secretImageLen = sImageInBin.length();
        int action, embImPos = 0, keyPos = 0;
        int width = coverImage.getWidth();
        int height = coverImage.getHeight();
```

Перевірка довжини потайного зображення: якщо зображення більше зображення контейнеру – завершуємо метод.

```
        if (secretImageLen > width * height * 2) {
          return null;
        }
```

Наступна частина коду відповідає за генерацію 24 бітного випадкового масиву біт та розміщення його у пікселі (0,0):

```
        int key[] = generateKey();
        int temp_number;
        int red_sum = 0;
        for (int j = 0; j <= 7; ++j) {
          if (key[j] == 1) {
            temp_number = (int) Math.pow(2, 7 - j);
          } else {
            temp_number = 0;
          }
          red_sum += temp_number;
        }
        int green_sum = 0;
        for (int j = 8; j <= 15; ++j) {
          if (key[j] == 1) {
            temp_number = (int) Math.pow(2, 15 - j);
          } else {
            temp_number = 0;
          }
          green_sum += temp_number;
        }
        int blue_sum = 0;
        for (int j = 16; j <= 23; ++j) {
          if (key[j] == 1) {
            temp_number = (int) Math.pow(2, 23 - j);
          } else {
            temp_number = 0;
          }
          blue_sum += temp_number;
        }
        stegoImage.setPixel(0, 0, Color.rgb(red_sum, green_sum,
blue_sum));
```

Змінюємо піксель (0,1) та помічаємо у ньому, що зображення контейнер буде зберігати приховане зображення

```
        stegoImage.setPixel(0, 1, Constants.COLOR_RGB_IMAGE);
```

Починаючи з пікселя (0,2) вбудовуємо повідомлення у контейнер

```
        int endX = 0, endY = 2;
        outerloop:
        for (int x = 0; x < width; x++) {
          for (int y = 2; y < height; y++) {
            int pixel = coverImage.getPixel(x, y);
            if (embImPos < secretImageLen) {
              int      colors[]      =      {Color.red(pixel),
Color.green(pixel), Color.blue(pixel)};
              for (int c = 0; c < 3; c++) {
                if (embImPos == secretImageLen) {
                  break;
                }
                //Action for LSB
                if ((key[keyPos] ^ LSB2(colors[c])) == 1) {
                  action           =           action(colors[c],
sImageInBin.charAt(embImPos));
                  colors[c] += action;
                  embImPos++;
                  keyPos = (keyPos + 1) % key.length;
                }
              }
              int  newPixel  =  Color.rgb(colors[0],  colors[1],
colors[2]);
              stegoImage.setPixel(x, y, newPixel);
            } else {
              if (y < height - 1) {
                endX = x;
                endY = y + 1;
              } else if (endX < width - 1) {
                endX = x + 1;
                endY = y;
              } else {
                endX = width - 1;
                endY = height - 1;
              }
              break outerloop;
            }
          }
        }
```

У наступний піксель поміщаємо код, що позначає завершення повідомлення, та повертаємо зображення контейнер.

```
        stegoImage.setPixel(endX, endY,
Constants.COLOR_RGB_END);
              return stegoImage;
            }
```

У класі Extracting реалізовано метод extractSecretMessage, що приймає у якості аргументу зображення-контейнер, та повертає вилучене повідомлення у якості Map.

```
public static Map extractSecretMessage(Bitmap stegoImage) {
    Map<String, Object> map = new HashMap<String, Object>();
    int width = stegoImage.getWidth();
    int height = stegoImage.getHeight();
    int key[] = new int[24];
```

У наступній частині коду зчитуємо ключ з пікселя (0, 0).

```
            int keyPixel = stegoImage.getPixel(0, 0);
            int red = Color.red(keyPixel);
            int green = Color.green(keyPixel);
        int blue = Color.blue(keyPixel);
        StandardMethods.showLog("EXT", "Key2: " + red + " " +
green + " " + blue);
        String red_bin = Integer.toBinaryString(red);
        red_bin = "00000000" + red_bin;
        red_bin = red_bin.substring(red_bin.length() - 8);
        for (int i = 0; i <= 7; i++) {
          key[i] = (red_bin.charAt(i) == '1' ? 1 : 0);
        }
        String green_bin = Integer.toBinaryString(green);
        green_bin = "00000000" + green_bin;
        green_bin = green_bin.substring(green_bin.length() - 8);
        for (int i = 0; i <= 7; i++) {
          key[i + 8] = (green_bin.charAt(i) == '1' ? 1 : 0);
        }
        String blue_bin = Integer.toBinaryString(blue);
        blue_bin = "00000000" + blue_bin;
        blue_bin = blue_bin.substring(blue_bin.length() - 8);
        for (int i = 0; i <= 7; i++) {
          key[i + 16] = (blue_bin.charAt(i) == '1' ? 1 : 0);
        }
```

Перевіряємо піксель (0, 1) та зчитуємо з нього тип прихованого повідомлення: зображення або текст. Якщо контейнер пусти – то тип буде не визначений.

```
        int typePixel = stegoImage.getPixel(0, 1);
        int tRed = Color.red(typePixel);
        int tGreen = Color.green(typePixel);
        int tBlue = Color.blue(typePixel);
```

```
          if (tRed == 135 && tGreen == 197 && tBlue == 245) {
            map.put(Constants.MESSAGE_TYPE, Constants.TYPE_TEXT);
          } else if (tRed == 255 && tGreen == 105 && tBlue == 180)
{
            map.put(Constants.MESSAGE_TYPE, Constants.TYPE_IMAGE);
          } else {
            map.put(Constants.MESSAGE_TYPE,
Constants.TYPE_UNDEFINED);
            map.put(Constants.MESSAGE_BITS, "");
            return map;
          }
```

У наступній частині коду відбувається вилучення біт з контейнеру, доки не зустрінеться код завершення прихованого повідомлення.

```
          StringBuilder sb = new StringBuilder();
          int keyPos = 0;
          outerloop:
          for (int x = 0; x < width; ++x) {
            for (int y = 2; y < height; ++y) {
              int pixel = stegoImage.getPixel(x, y);
              int       colors[]       =       {Color.red(pixel),
Color.green(pixel), Color.blue(pixel)};
              //Colors.COLOR_RGB_END
              if (colors[0] == 96 && colors[1] == 62 && colors[2]
== 148) {
                break outerloop;
              } else {
                for (int c = 0; c < 3; c++) {
                  if ((key[keyPos] ^ LSB2(colors[c])) == 1) {
                    int lsb = LSB(colors[c]);
                    sb.append(lsb);
                    keyPos = (keyPos + 1) % key.length;
                  }
                }
              }
            }
          }
          String sm = sb.toString();
          int sL = sm.length();
          //Cut unnecessary [0-7] pixels
          sm = sm.substring(0, sL - sL % 8);

          map.put(Constants.MESSAGE_BITS, sm);
          return map;
        }
```

ВИСНОВКИ

Стеганографія є ефективним способом безпечного спілкування. Є можливість спочатку зашифрувати конфіденційний файл, а потім приховати його всередині зображення іншого типу файлу, перш ніж надіслати його іншому. Це зменшить шанси бути перехопленим.

Якщо просто відправити файл, зашифрувавши його, зловмисник спробує розшифрувати його різними способами. Однак, якщо він знайде лише звичайний файл зображення, він не матиме поняття.

Ця техніка дуже проста у використанні, але її дуже важко виявити. Його можуть використовувати державні організації для безпечного надсилання та отримання файлів.

У роботі розглядаються різні типи стеганографії та методи виявлення стеганографії. Всі методи, розглянуті в цій роботі, здатні захистити приховані дані. З іншого боку, деякі алгоритми мають дуже високу тимчасову складність і дуже невелику кількість даних, що зберігаються в зображеннях. Таким чином, існує потреба в розробці ефективних і точних алгоритмів стеганографії, або шляхом поєднання існуючих методів, або шляхом розробки нових методів.

ПЕРЕЛІК ПОСИЛАНЬ

1. P. Selvigrija, E. Ramya, Video by Linked List Method, no. March, 2015.

2. M. Hussain, M. Hussain, A Survey of video Steganography Techniques. Int. J. Adv. Sci. Technol., 2013. vol. 54, pp. 113–124.

3. G. Kaur, A. Kochhar, A Steganography Implementation based on LSB & DCT, 2012. vol. 4, no. 1, pp. 35–41.

4. M. Dixit, N. Bhide, S. Khankhoje, R. Ukarande. Video Steganography, 2015 Int. Conf. Pervasive Comput., vol. 00, c, pp. 1–4.

5. R.J. Mstafa, I. Studen, A High Payload Video Steganography Algorithm in DWT Domain Based on BCH Codes 2015. vol. 15, no. 11.

6. J. Gupta, A Review on Steganography techniques and methods, 2015. vol. 1, no. 1, pp. 1–4.

7. C. Science, B. Bridgeport, A Novel Video Steganography Algorithm in the Wavelet Domain Based on the KLT Tracking Algorithm and BCH Codes, 2015

8. Stuti Goel, Arun Rana, Manpreet Kaur. A Review of Comparison Techniques of Image Steganography, IOSR Journal of Electrical and Electronics Engineering (IOSR-JEEE) e-ISSN: 2278-1676,p-ISSN: 2320-3331, Vol 6, Issue 1 May. - Jun. 2013, pp. 41-48.

9. Електроний ресурс: Student_Web. URL: http://studentweb.niu.edu/9/~Z172699/Conclusion.html, (дата звернення 01.03.2022).

10. Harpreet Kaur, Jyoti Rani. A Survey on different techniques of steganography. MATEC Web of Conferences CSE Department, GZSCCET Bathinda, Punjab, India, 2016. DOI: 10.1051/ 57, 02003.

11. Електроний ресурс: Best tools to perform steganography URL:https://resources.infosecinstitute.com/topic/steganography-and-tools-to-perform-steganography/. (дата звернення 15.03.2022).

12. M. Pavani1, S. Naganjaneyulu, C. Nagaraju. A Survey on LSB Based Steganography Methods. International Journal Of Engineering And Computer Science. 2013. Vol. 2, Issue 8, pp. 2464-2467, ISSN:2319-7242.