STEGANOGRAFIA – APLIKACJA DO ODCZYTYWANIA I UKRYWANIA INFORMACJI W GRAFICE

DOKUMENTACJA

28 MAJA 2021

MARTA KOT, MARIUSZ NIECIECKI

Spis treści

Cel zadania projektowego

Stworzenie aplikacji na telefon z systemem operacyjnym android, która umożliwiałaby ukrywanie treści w grafikach w taki sposób, aby osoba trzecia nie była w stanie wykryć jej obecności. Aplikacja powinna umożliwić również odczytanie wiadomości osobie, która zna klucz (podawany podczas kodowania wiadomości w grafice).

Steganografia

Steganografia jest to nauka o komunikacji, w której nie można wykryć obecności komunikatu. Pojęcie steganografii nie jest popularne, jednak ma wiele praktycznych zastosowań, z którymi można spotkać się na co dzień nie wiedząc o tym.

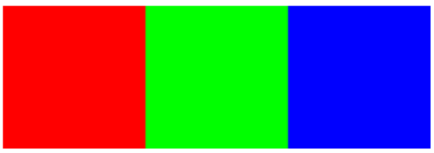
Steganografia znalazła szerokie zastosowanie w rzeczywistości wirtualnej. Polega na ukryciu wiadomości, zdjęcia lub pliku w nośniku tak, aby osoba trzecia nie była w stanie wykryć jej obecności. Nośnikiem może być zdjęcie lub inna wiadomość. Mamy wtedy do czynienia z cyfrową steganografią. Najczęściej wybór pada na ukrywanie wiadomości w plikach multimedialnych, ponieważ posiadają one duży rozmiar i zawierają pełno informacji nadmiarowych.

Podejście do problemu

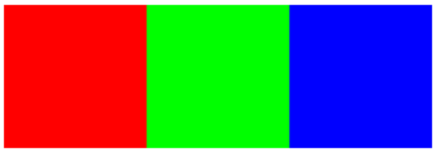
Pliki cyfrowe i wiadomości tekstowe można zapisać w postaci ciągu bitów. Są to ciągi zero-jedynkowe, których odkodowanie przy użyciu odpowiednich kodeków powoduje, że stają się czytelne nie tylko dla maszyny, ale również dla użytkownika.

Barwa piksela w obrazach cyfrowych opisywana jest za pomocą wartości liczbowych. Przykładowo w obrazie o 24 bitowej głębi koloru, barwa każdego piksela zapisywana jest na 24 bitach, czyli 3 bajtach. Każdy bajt odpowiada innej składowej koloru, a zapisuje się je najczęściej stosując model przestrzeni barw RGB. Każda wartość składowej koloru może być przedstawiona za pomocą postaci bitowej. Po zmianie trzech najmniej znaczących bitów każdej składowej, oko ludzkie nie jest w stanie zarejestrować zmiany barwy, ponieważ jest ona niezauważalna i znikoma.

Zależność ta odnajduje swoje zastosowanie w steganografii. Im więcej najmniej znaczących bitów można zmienić, tym więcej informacji można ukryć w obrazie, ale jednocześnie tym bardziej może on odstawać od oryginału. Na poniższych rysunkach zostało pokazane ukrycie litery S o kodzie binarnym: 01010011. Porównując Rys. 1 i Rys. 2 widać jak zmienił się obraz po modyfikacji najmniej znaczącego bitu każdej barwy piksela.



Rys. 1: Obraz przed ukryciem litery S



Rys. 2: Obraz po ukryciu litery S zmieniając najmniej znaczący bit

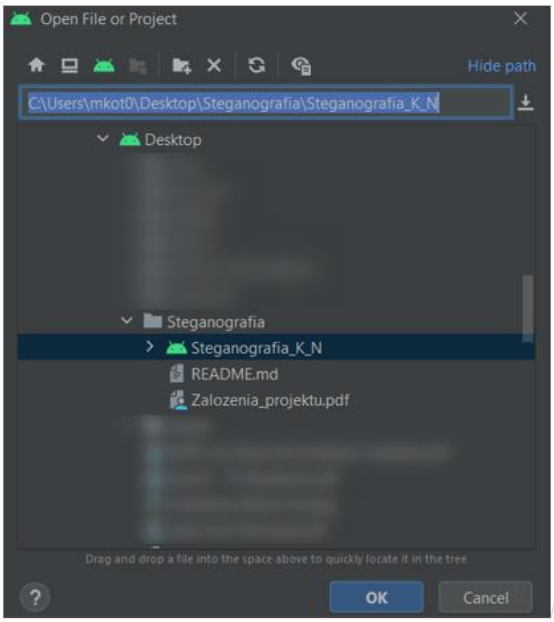
Użyte programy

Do wykonania aplikacji użyto języka programowania Java oraz zintegrowanego środowiska programistycznego dla systemu operacyjnego Android – Android Studio

Uruchomienie aplikacji

W celu uruchomienia aplikacji należy:

1. Otworzyć nowy projekt w programie Android Studio i z górnej zakładki wybrać *File* → *Open*
2. Wybrać ścieżkę do pliku *Steganografia_K_N* i zatwierdzenie przyciskiem *OK*



3. Wybrać jeden spośród wielu dostępnych emulatorów dostępnych w zakładce AVD Manager (sugerowany model telefonu – Nexus 5X).
4. Skompilować projekt przyciskiem *RUN*

Przyjęte rozwiązania programistyczne

1. Opis klas i znajdujących się w nich metod

a) Klasa MainActivity – jest to główna klasa, która w widoku użytkownika reprezentuje główne menu aplikacji.

Są w niej zadeklarowane przyciski obsługujące konkretne elementy w warstwie widoku (Przycisk zakoduj, zdekoduj itd.)

Zdefiniowane są w niej również następujące funkcje:

- SteganoInfo(View view) – funkcja przenosząca użytkownika na stronę www, gdzie opisane jest pojęcie Steganografii
- Exit(View view) – funkcja kończąca działanie aplikacji
- AuthorsView(View view) – nowy widok - Autorzy
- DecodeView(View view) – nowy widok - Dekodowanie

- EncodeView(View view) – nowy widok - Kodowanie

b) klasa Encode – jest to klasa odpowiedzialna za widok oraz wywołanie się odpowiednich metod podczas kodowania wiadomości

Są w do niej zaimportowane potrzebne biblioteki, zadeklarowane wszystkie niezbędne zmienne, oraz instancje obiektów potrzebne do zakodowania wiadomości.

Występują w niej również następujące funkcje:

- saveEncodedImg(Bitmap bitmapImage) – funkcja zapisująca zakodowane zdjęcie w katalogu *Downloads*
- imageChooser() – funkcja odpowiedzialna za utworzenie nowej aktywności – wybór grafiki w której będziemy kodować wiadomość
- onActivityResult(int requestCode, int resultCode, Intent data) – funkcja sprawdzająca czy podczas aktywności wybraliśmy zdjęcie – przypisuje ścieżkę do zdjęcia i samo zdjęcie do odpowiednich zmiennych, wyświetla informacje o wyborze zdjęcia
- onCompleteTextEncoding(ImageSteganography result) – funkcja przypisująca do zmiennej grafikę z zakodowaną wiadomością

c) klasa Decode – jest to klasa odpowiedzialna za widok oraz wywołanie się odpowiednich metod podczas odkodowywania wiadomości

Są w do niej zaimportowane potrzebne biblioteki, zadeklarowane wszystkie niezbędne zmienne, oraz instancje obiektów potrzebne do zdekodowania wiadomości.

- ImageChooser() – funkcja odpowiedzialna za utworzenie nowej aktywności – wybór grafiki z której będziemy odkodowywać wiadomość
- onActivityResult(int requestCode, int resultCode, Intent data) – funkcja sprawdzająca czy podczas aktywności wybraliśmy zdjęcie – przypisuje ścieżkę do zdjęcia i samo zdjęcie do odpowiednich zmiennych, wyświetla informacje o wyborze zdjęcia
- onCompleteTextEncoding(ImageSteganography result) – funkcja wyświetlająca informację o tym czy udało się odkodować wiadomość, oraz wyświetlająca samą wiadomość.

d) klasa Authors – jest to klasa udostępniająca widok dzięki któremu użytkownik może dowiedzieć się, kto jest autorem aplikacji.

2. Zewnętrzne biblioteki

W aplikacji skorzystano z ogólnodostępnej biblioteki posiadającej funkcje, które pozwalają zakodować wiadomość w obrazie oraz ją odkodować.

3. Zakodowanie wiadomości w obrazie

W pierwszym kroku (po naciśnięciu przycisku zakoduj) następuje utworzenie obiektu ImageSteganography i przekazanie do konstruktora wartości **message** (wiadomość), **secretKey** (klucz), **originalImg** (oryginalne zdjęcie).

Następnie tworzony jest obiekt TextEncoding, gdzie również do konstruktora zostają przekazane odpowiednie wartości.

Ostatnim finalnym krokiem jest wykonanie kodowania

```
public void onClick(View view) {
    if(filepath!=null){
        if(message.getText()!=null){
            imageSteganography = new
ImageSteganography(message.getText().toString(), secretKey.getText().toString(),
originalImg);
            textEncoding = new TextEncoding(Encode.this, Encode.this);
            textEncoding.execute(imageSteganography);
        }
    }
}
```

textEncoding.execute(imageSteganography);

Obiekt ImageSteganography korzysta z klasy z biblioteki *ImageSteganography.java*.

Wywołanie konstruktora powoduje przypisanie wartości do zmiennych:

```
this.message = message;
this.secret_key = convertKeyTo128bit(secret_key);
this.image = image;
```

Do klucza zostaje przypisana przekonwertowana wartość przy pomocy funkcji *convertKeyTo128bit(secret_key)*, która konwertuje ciąg znaków na bajty. Funkcja ta jest odpowiedzialna za ograniczenie związane z tym, że długość klucza nie może przekraczać 16 znaków.

W dalszej części, treść wiadomości kodowana jest w sekwencję bajtów i zwracana w postaci tablicy pod zmienną *encrypted_zip*.

Następnie wywoływana jest funkcja *encryptMessage*, która jest funkcją kodującą korzystającą z klasy Crypto.

```
encrypted_message = Crypto.encryptMessage(message, secret_key);
```

Metoda *encryptMessage* znajduje się w bibliotece, w klasie *Crypto.java*. Pobiera wiadomość oraz klucz do zakodowania. Tworzony jest cipher, który korzysta z kluczu kodowania typu AES.

```
cipher.init(Cipher.ENCRYPT_MODE, aesKey);
byte[] encrypted;
encrypted = cipher.doFinal(message.getBytes());
```

Zostaje zainicjowany szyfr przy pomocy klucza kodowania AES. Jest tworzona tablica bajtów do przechowywania zaszyfrowanej wiadomości a następnie tablica zostaje wypełniona zaszyfrowaną wiadomością, za co odpowiada metoda *cipher.doFinal(message.getBytes());*. Funkcja ostatecznie zwraca zaszyfrowaną wiadomość.

Następnie zaszyfrowaną wiadomość należy zakodować w zdjęciu.

Dzieje się to przy użyciu metody:

```
@Override
protected ImageSteganography doInBackground(ImageSteganography...
imageSteganographies)
```

Metoda ta pobiera oryginalne zdjęcie, przekształca je na bitmapę a następnie sprawdza jego wysokość i szerokość.

Kolejnym krokiem jest wywołanie metody *splitImage(bitmap);* z klasy *Utility.java*. Zadaniem tej funkcji jest podzielenie obrazu na kwadraciki – mniejsze elementy. Elementy te przechowywane są w formie bitmap w kontenerze. Cała metoda zwraca listę wyodrębnionych obrazów.

Następnie wywoływana jest funkcja *encodeMessage(splitted_images, encrypted_message, progressHandler)* z klasy *EncodeDecode.java*, która odpowiada za właściwe zakodowanie wiadomości w otrzymanych „kawałkach” zdjęcia.

Metoda pozwala na zaszyfrowanie wiadomości w zdjęciu na dwóch najmniej znaczących bitach. Metoda przechodzi przez każdy bit zdjęcia jak po dwuwymiarowej tablicy. Szyfrowanie następuje przez przesuniecie wartości całkowitej o 2 w lewo i zastąpienie dwóch najmniej znaczących cyfr wartościami *message_byte_array*. Wynikiem działania funkcji jest tablica bajtów zakodowanego już zdjęcia.

Wynik działania funkcji *encodeMessage()* przypisany jest do listy bitmapy *encoded_list*. Następnie znów wywoływana jest metoda z klasy *Utility.java - mergeImage(encoded_list, originalHeight, originalWidth)*, która na celu ma scalenie zakodowanych „kwadracików" w całość – czyli gotowe, zakodowane zdjęcie.

Finalnie zwracane jest zakodowane zdjęcie i użytkownik może je zapisać do folderu *Downloads*.

4. Odkodowanie wiadomości z obrazu

Odkodowanie wiadomości z zakodowanego zdjęcia jest operacją odwrotną do zakodowania.

Proces odkodowania jest uruchomiony przy pobraniu zdjęcia z zakodowaną wiadomością i prawidłowo wpisanego tajnego klucza. Najpierw obraz zamieniany jest na bitmapę i dzielony na mniejsze kwadratowe elementy.

Następnie program przechodzi po każdym elemencie i odczytuje z niego wartość dwóch najmniej znaczących bitów i zapisuje je. Z otrzymanego ciągu bitów, który zapisywany jest jako wartość *encrypted_message*, program jest w stanie odkodować wiadomość korzystając z funkcji *decryptMessage(encrypted_message, secret_key);* z klasy *Crypto.java*.

Pobiera wiadomość oraz klucz do odkodowania. Wywoływany jest obiekt klasy *SecretKeySpec.java*, który pobiera klucz w bajtach i przyjmuje typ kodowania AES. Następnie konstruuje klucz z podanej tablicy bajtów. Dalej tworzony jest cipher, który również korzysta z kluczu kodowania typu AES. Wywoływana jest funkcja cipher.doFinal(…).

Do zmiennej *decrypted* przypisany jest wynik działania metody *cipher.doFinal(…)*, której wynikiem jest odkodowana wiadomość.

Odkodowana wiadomość zostaje wyświetlona użytkownikowi.

Opis działania aplikacji

1. Kodowanie wiadomości – z menu startowego należy wybrać opcję „Zakoduj"

Aby zakodować wiadomość w grafice użytkownik musi z poziomu aplikacji wybrać dowolne zdjęcie znajdujące się w galerii. Następnie musi on podać wiadomość, którą chce zakodować, oraz wymyślony przez siebie klucz (bez polskich znaków, zalecana długość mniejsza niż 16 znaków). Kolejnym krokiem jest naciśnięcie przycisku „Zakoduj" – uruchamia on wykonanie bloku kodu, który jest odpowiedzialny za zakodowanie tekstu w grafice. Należy odczekać chwilę na wiadomość potwierdzającą pomyślne zakodowanie tekstu, a następnie zapisać nowy obraz naciskając przycisk „Zapisz" – nowy obraz wraz z zakodowaną wiadomością zostanie zapisany w folderze *Downloads*.

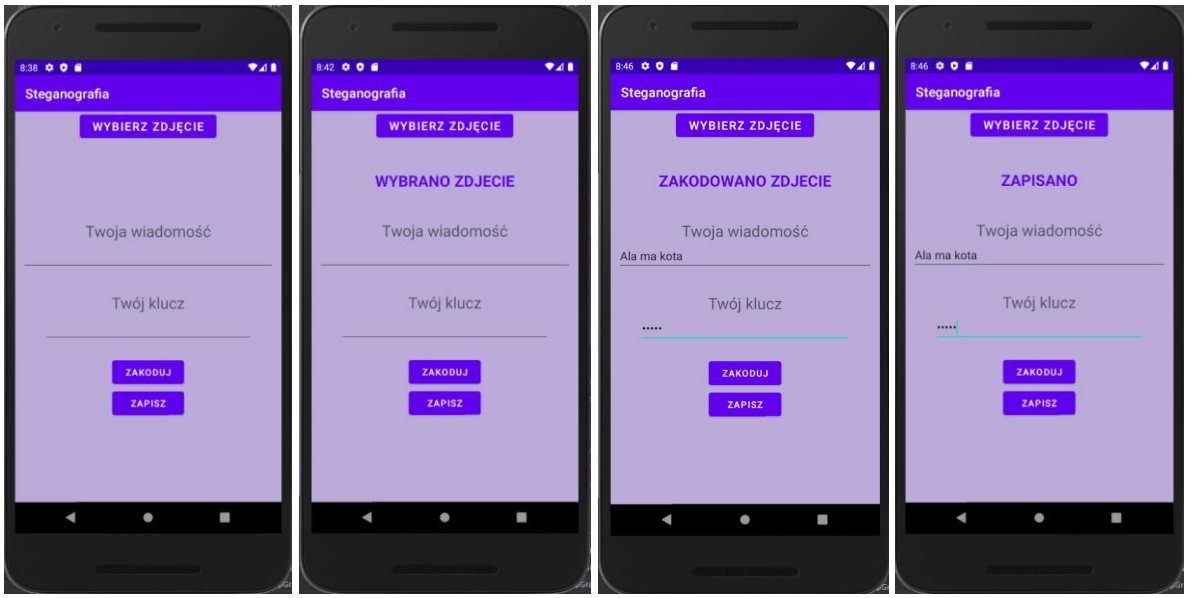
2. Dekodowanie wiadomości – z menu startowego należy wybrać opcję „Zdekoduj”

Aby zdekodować wiadomość ukrytą w grafice należy wybrać obraz z folderu *Downloads*, w którym wcześniej owa wiadomość została ukryta. Następnie należy podać klucz (taki sam jak klucz podany podczas kodowania) i nacisnąć przycisk „Dekoduj” – wciśnięcie przycisku uruchamia wykonanie bloku kodu, który jest odpowiedzialny za zdekodowanie wiadomości. Jeśli użytkownik wpisał poprawny klucz to w wyznaczonym miejscu pojawi się zakodowana wcześniej wiadomość. W przypadku podania złego klucza w wyznaczonym miejscu wyświetli się komunikat mówiący o jego niepoprawności.

Interfejs aplikacji

Uruchomienie aplikacji powoduje wyświetlenie głównego menu aplikacji:

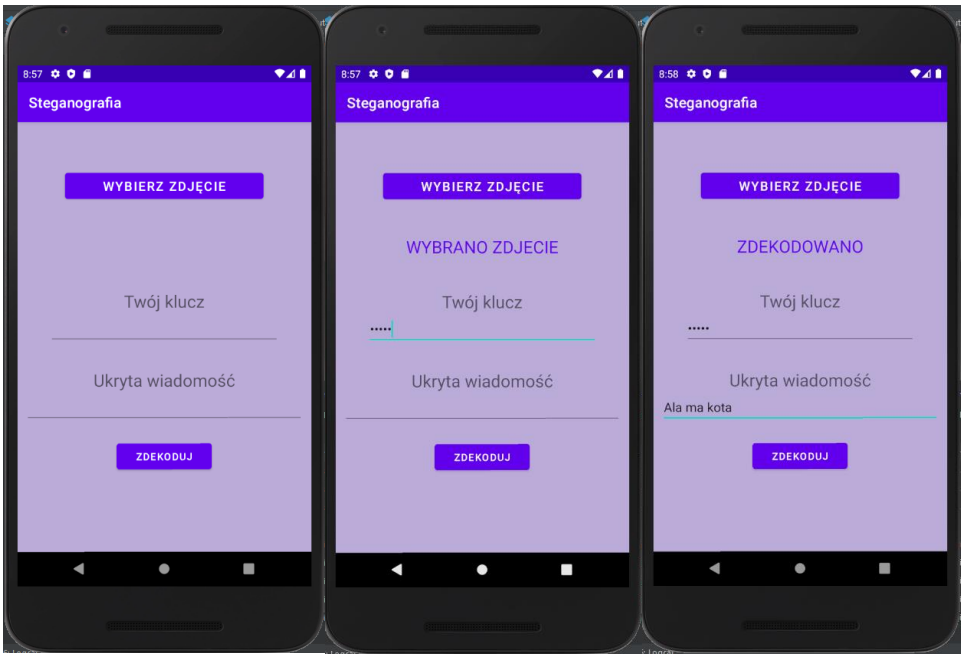
Po wybraniu opcji „Zakoduj” wyświetla się widok, w którym można wybrać zdjęcie, wpisać wiadomość oraz klucz, a także zakodować wiadomość i zapsisać grafikę z zakodowaną wiadomością.



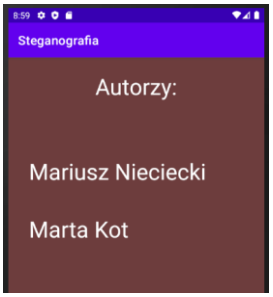
Zdjęcie przed i po zakodowaniu:

Zdjecie oryginalne *Zdjęcie z zakodowaną wiadomością*

Po wybraniu opcji „Zdekoduj” wyświetla się widok, w którym można wybrać zdjęcie, wpisać klucz oraz spróbować zdekodować zakodowaną wiadomość (uda się w przypadku podania prawidłowego klucza).



Po wybraniu opcji „Autorzy” wyświetla się widok, w którym pojawiają się autorzy aplikacji:



Wybranie opcji „Steganografia Info” powoduje przeniesienie do strony, na której wyjaśnione jest pojęcie steganografii.

Po wybraniu opcji „Wyjdź” aplikacja kończy swoje działanie.

Źródła

[1] http://ekryptografia.pl/steganografia/metoda-najmniej-znaczacego-bitu/

[2] https://pawelskaruz.pl/2017/03/steganografia-a-co-to-jest-a-komu-to-potrzebne/

[3] https://docs.oracle.com/en/java/javase/11/docs/api/java.base/javax/crypto/Cipher.html

[4] https://www.baeldung.com/java-aes-encryption-decryption

[5] https://github.com/aagarwal1012/Image-Steganography-Library-Android/tree/master/ImageSteganographyLibrary

[6] https://developer.android.com/studio/intro