

SVEUČILIŠTE U ZAGREBU  
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

**Projekt iz predmeta Raspoznavanje uzorka**

# **Uporaba steganografije u revezibilnoj deidentifikaciji**

Katarina Matić, Hrvoje Backović, Marin Oršić,

Dino Rakipović, Ivan Relić, Filip Reškov

Voditelj: *Slobodan Ribarić*

Zagreb, siječanj 2016.

# SADRŽAJ

<b>1. Projektni zadatak</b>	<b>1</b>
1.1. Opis projektnog zadatka . . . . .	1
1.2. Pregled i opis srodnih rješenja . . . . .	1
1.3. Konceptualno rješenje zadatka . . . . .	7
<b>2. Postupak rješavanja zadataka</b>	<b>8</b>
<b>3. Ispitivanje rješenja</b>	<b>9</b>
3.1. Ispitna baza . . . . .	10
3.2. Rezultati ispitivanja . . . . .	11
3.3. Analiza rezultata . . . . .	27
<b>4. Opis programske implementacije rješenja</b>	<b>29</b>
4.1. Opis programske implementacije . . . . .	29
4.2. Način korištenja implementacije . . . . .	31
<b>5. Zaključak</b>	<b>33</b>
<b>6. Literatura</b>	<b>34</b>

# 1. Projektni zadatak

## 1.1. Opis projektnog zadatka

## 1.2. Pregled i opis srodnih rješenja

Steganografija se koristila od davnina i stoga su dosad razvijeni brojni postupci koji efektivno skrivaju i prenose tajne poruke putem raznih medija kao što su tekst, zvučni signali ili slike. Za potrebe ovog projekta, koncentriramo se na steganografske postupke koji se odnose na skrivanje podataka unutar slika jer svi oni mogu poslužiti u različitim situacijama za rješavanje problema koji se u ovom projektu rješava. Najprije ćemo nabrojati svojstva koja steganografski postupak treba zadovoljavati da bi bio praktično upotrebljiv. Ta svojstva ćemo koristiti kao mjeru po kojoj ćemo uspoređivati različite postupke. Nakon toga ćemo navesti i opisati nekoliko poznatih postupaka te ćemo ih usporediti sa našim algoritmom po svojstvima koje smo naveli.

Općenito, steganografske postupke nad slikama možemo klasificirati u dvije domene: **transformacijska domena** (tehnika frekvencijske domene) i **slikovna domena** (tehnika prostorne domene). Transformacijska domena primjenjuje transformacije slike kako bi ubacio tajne podatke u sliku, dok slikovna domena koristi modificiranje bitova boje/intenziteta piksela i manipulaciju šuma u originalnoj slici. LSB algoritam koji se koristi u ovom projektu spada u slikovnu domenu steganografskih tehniki. Veliki broj istraživača su predložili brojne tehnike za ove domene koje čitatelj može dodatno proučiti u navedenoj literaturi [LITERATURA], a mi ćemo opisati samo neke najvažnije.

Za početak navodimo svojstva koja su važna za svaki steganografski postupak i po kojima ih uspoređujemo:

**kapacitet** Vrlo je važno definirati kapacitet postupka, odnosno koliko podataka može

najviše sakriti unutar medija (slike) bez da uzrokuje vidljive promjene na slici koje će potencijalnom napadaču dati do znanja da se u slici krije neka skrivena poruka.

**transparentnost** Nakon procesa skrivanja poruke, originalna slika će se u određenoj mjeri promijeniti. Mjera u kojoj stego-slika odudara od originalne se naziva transparentnost i poželjno je da ta razlika bude što manja kako bi bila manja šansa za otkrivanje kodirane poruke.

**robustnost** Nakon što smo kodirali poruku unutar slike, poželjno je da ta poruka ostane nepromijenjena čak i ako slika bude podvrgnuta raznim transformacijama kao što je rezanje dijelova slike, skaliranje, filtriranje, dodavanje šuma i sl. Robustnost je svojstvo koje nam to opisuje.

**otpornost na promjene poruke** Za dobar steganografski postupak trebalo bi biti teško promijeniti tajnu poruku, jednom kada je ona kodirana u sliku. Ako je ta otpornost mala, moguće je da napadač otkrije poruku, promijeni njen sadržaj i prosljedi dalje do odredišta i time uzrokuje štetu.

**računska složenost** Često je poželjno da postupak skrivanja poruke unutar slike i njenu rekonstrukciju iz slike bude što je moguće brži, tako da omogući primjene u npr. aplikacijama koje rade u stvarnom vremenu. Stoga često uspoređujemo postupke po njihovoj računskoj složenosti

Kod prenošenja slika, često se provodi kompresija slike prije slanja. Ovaj koncept je jako bitan jer utječe na dobar odabir steganografskog postupka. Postoje brojni algoritmi za kompresiju slika, a klasificiramo ih u dvije skupine: *lossy* i *lossless*.

*Lossy* kompresija reducira količinu informacije koju je potrebno prenijeti tako da trajno gubi dio informacija sa slike, najčešće redundantne informacije. Miču se oni detalji koje ljudsko oko ne može razlikovati, što rezultira u dobroj aproksimaciji originalne slike, ali koja se ipak razlikuje od originala. *JPEG (Joint Photographic Experts Group)* je jedan od slikovnih formata koji koriste lossy kompresiju. Takav format slike nije pogodan za naš algoritam jer *LSB* steganografski postupak upisuje poruku u najmanje bitne dijelove slike kako bi minimizirali promjene na slici, ali to su upravo oni bitovi koje *JPEG* odbacuje kako bi sačuvao prostor.

S druge strane, *lossless* kompresija nikad ne odbacuje informacije iz slike nad kojom radi kompresiju, tako da se svaki bit informacije može povratiti u originalno stanje nakon dekompresije. *GIF (Graphical Interchange Format)* i *BMP (Bit Map File)* su formati slika koji koriste "lossless" kompresiju. BMP format je upravo onaj koji koristimo u našem projektu i najpogodniji je za *LSB* steganografski postupak koji smo implementirali. Također se isti postupak može primijeniti i na *GIF* slike, iako valja biti oprezan pri tome.

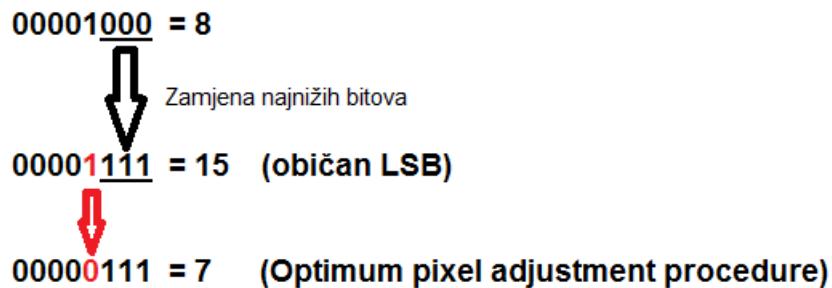
Algoritam koji se primjenjuje u ovom projektu se naziva *LSB (Least Significant Bit)* postupak, a poruke kodiramo unutar *BMP* slike. Algoritam funkcioniра na način da najniže bitove svakog piksela zamijenimo sa bitovima iz poruke. Zato što koristimo najniže bitove, razlika u odnosu na originalnu sliku će biti praktički nezamjetna što znači da ovaj algoritam ima visoku transparentnost. Dodatno se ta transparentnost može povećati za manje poruke tako da postavimo prioritete na kanale (crvena, zelena, plava) kojima mijenjamo bitove. Pokazalo se da je ljudsko oko najmanje osjetljivo na promjene u plavom kanalu, stoga je logično da prvo želimo te bitove mijenjati.

S druge strane, pošto koristimo samo najniže bitove, kapacitet odnosno veličina podataka koju možemo sakriti u slici je dosta ograničena. Ako koristimo *BMP* gdje svaki piksel zauzima 24 bita (po 8 bitova za crvenu, zelenu i plavu boju) te ako mijenjamo samo najniži bit, možemo spremiti 3 bita u svaki piksel tako da promijenimo najniži bit od svake boje. Za sliku rezolucije 800x600 piksela, moguće je sakriti poruku maksimalne veličine 1,440,000 bitova, odnosno 180,000 okteta. Moguće je zamijeniti najnižih nekoliko bitova bez da bude velike promjene na slici, ali više od toga dovodi do slabe transparentnosti. Za naš projekt želimo lica sa slike izdvojiti i kodirati u najniže bitove te zamutiti originalno lice tako da se na prvi pogled ne zna tko je na slici, ali je ta informacija sadržana u tajnoj poruci. Ovaj algoritam dobro funkcioniра za lica koja nisu prevelika, ali za velika lica, slika će biti vidljivo promijenjena zbog ograničenog kapaciteta.

Nadalje, *LSB* postupak ima prednost nad ostalim algoritmima po tome što je vrlo jednostavan za implementirati i ima malu računsku složenost, ali s druge strane nije robustan i vrlo je osjetljiv na bilo kakvo filtriranje slike, skaliranje, rotaciju i sl.

Također je ovaj postupak osjetljiv na promjene poruke, odnosno moguće je promijeniti poruku koja se kodira ako je poznat način na koji se poruka kodira u sliku. Tome

Slika 1.1: Optimum pixel adjustment procedure



je moguće pristupiti na način da pošiljatelj i primatelj dijele tajni ključ koji specifičira točno koji pikseli će se mijenjati (i kojim redoslijedom). U slučaju da napadač otkrije da postoji tajna poruka unutar slike, bez ključa će biti nemoguće znati točno koje piksele je potrebno promijeniti da bi promijenio poruku u nešto smisleno.

LSB postupak koji se koristi u ovom projektu je općenito dio porodice *LSB* steganografskih postupaka koji su razvijeni za različite primjene, ali svi u načelu imaju iste prednosti i mane te su neki varijacije ili poboljšanja istoga. Stoga ćemo spomenuti takozvanu *Optimum pixel adjustment procedure* što je poboljšanje običnog *LSB* postupka. Ta procedura funkcionira na način da dodatno mijenja još neke bitove u pikselima (koji nisu dio poruke koja se kodira) s ciljem da dobiveni intenzitet piksela bude što bliži originalnom. Slika 1.1 prikazuje jedan jednostavan primjer gdje se može vidjeti kako ova procedura daje bolji rezultat od običnog *LSB*-a. Prednost ove procedure je što steg-slika sadrži manje distorzije i stoga je sličnija originalnoj slici i time ima veću transparentnost.

Kao što je već spomenuto, *LSB* steganografski postupci se mogu koristiti i na *GIF* slikama jer one koriste lossless kompresiju (konkretno, *LZW* kompresiju), ali valja biti oprezan. Problem je u tome što *GIF* slike spremaju različite boje koje se koriste u takozvanu tablicu "paletu boja", gdje je svakoj boji dodijeljen indeks u toj tablici. U takvoj slici su svi pikseli definirani samo indeksom u toj paleti pa je bitno paziti jer bliski indeksi u paleti boja mogu rezultirati potpuno različitim bojama pa nije dobro izravno primjenjivati *LSB* postupak. Taj problem se može riješiti tako da se sortiraju boje u paleti, što će dovesti do toga da slični indeksi rezultiraju sličnim bojama pa je onda moguće raditi običan *LSB* algoritam.

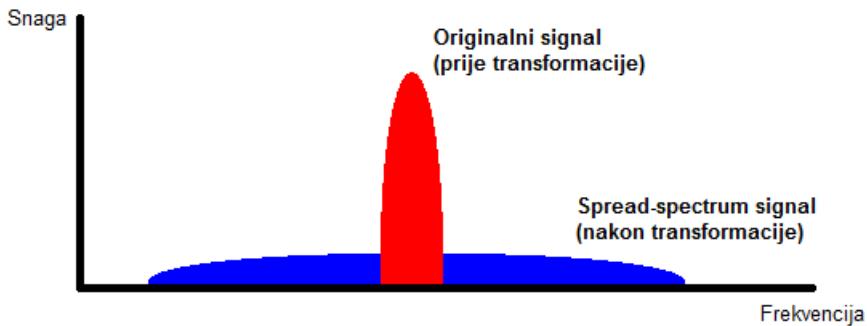
Drugi način za doskočiti ovom problemu je da uvodimo nove boje u paletu koje su vizualno slične već postojećim bojama u paleti. Ovo zahtijeva da je broj korištenih boja u paleti manji od maksimalnog broja boja pa je potrebno paziti u kakve slike skrivamo poruke. Još jedan način rješavanja ovog problema je da koristimo crno-bijele slike gdje imamo 256 različitih nijansi sive gdje je promjena u nijansama vrlo postepena i teža za uočiti.

Nadalje imamo još neke postupke iz slikovne domene, koje nazivamo *mod* postupci. Često se koristi mod10 koji svakom pikselu oduzme njegov ostatak pri dijeljenju sa 10 te mu nadoda bitove iz poruke, opet iz intervala [0,9]. Ovaj postupak je lakše za implementirati od običnog LSB algoritma te se dosta koristi pogotovo na crno-bijelim slikama. Rekonstrukcija poruke iz ovako kodirane slike se također lagano provodi: jednostavno iz svakog piksela izvadimo njegov ostatak pri dijeljenju sa 10.

Još jedan predstavnik steganografskih postupaka iz slikovne domene je *patchwork*. Taj postupak daje pseudoslučajni, statistički model koji višestruko skriva istu poruku na slici. Radi na principu da odabire slučajno po dva područja (*patch-a*) na slici te jednomo poveća intenzitet za neku konstantu, dok drugome smanji za konstantu. Time se osigurava da je očekivana promjena intenziteta jednaka 0 što uvelike pomaže da postupak bude visoko transparentan. Ono što je dobro kod ovog algoritma je što je otporan na neke transformacije slike kao što je npr rezanje dijelova slike ili rotacije, upravo zato što se ista poruka kodira na više mesta na slici. S druge strane, ta činjenica dovodi do loše strane ovog pristupa a to je mali kapacitet. Zato se ovaj pristup koristi samo kada treba prenijeti male poruke.

Osvrnut ćemo se još malo na format slika *JPEG*. Taj format je jedan od najkorišteñijih formata općenito te su osmišljeni brojni steganografski postupci koji dobro rade upravo za ovaj format. *JPEG* slike koriste lossy kompresiju i to na način da se izvede prvo diskretna kosinusna transformacija (*DCT*) nad slikom te se spremaju koeficijenti koji se dobiju, a nakon toga se Huffmanovim kodiranjem komprimiraju ti koeficijenti. Rekonstrukcijom iz tih koeficijenata se dobiva dobra (ali ipak ne savršena) aproksimacija originalne slike. Zato što je kompresija *lossy*, algoritmi koje smo dosad spomenuli ne mogu se primjeniti direktno na ovom formatu slika, ali je moguće primjeniti *LSB* nad samim koeficijentima prije Huffmanovog kodiranja, jer će slični koeficijenti dati slične slike, a sami Huffmanov postupak je lossless kompresija što znači da će se poruka sačuvati.

Slika 1.2: Transformacija tajne poruke u signal male snage

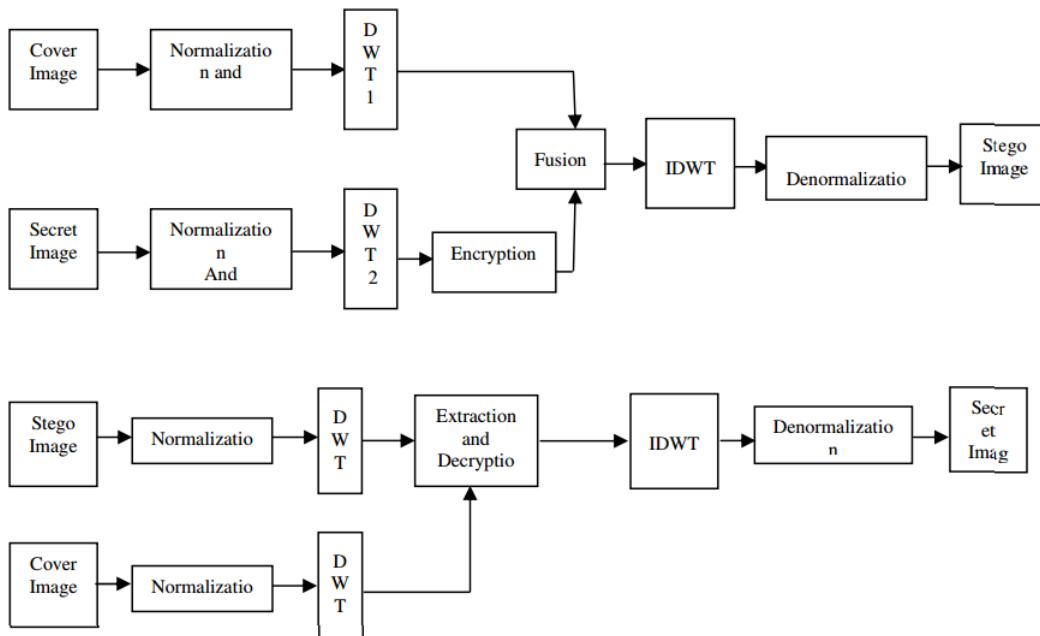


Osim pristupa iz porodice *LSB* algoritama, pogledat ćemo i neke postupke iz transformacijske domene. Postupci iz transformacijske domene su općenito robustniji od onih iz slike domene pa su jako korisni u slučajevima kada je slika podvrgnuta raznim transformacijama jer su relativno otporni na promjene u kodiranoj slici. Također, ovi postupci općenito imaju i veliki kapacitet te je moguće sakriti čak cijelu sliku unutar druge slike.

Jedan od tih postupaka je takozvani *Spread-spectrum* postupak. On spada i u slikovnu domenu i u transformacijsku, ovisno o implementaciji. Originalno se koristi za ubacivanje tajnih zvučnih poruka u zvučne signale, ali se isti princip može primjeniti i na slikama. Algoritam se temelji na tome da se informacija koja se želi sakriti prikaže kao signal definiran na uskom području frekvencija i koji je općenito velike snage. Zatim se taj signal rastegne preko većeg raspona frekvencija, a time se i snaga smanji. Takav signal male snage će zapravo biti šum koji možemo ubaciti u drugi signal. Zbog niske snage, naša poruka će biti dobro skrivena(zagušena) i neće se čuti praktički никакva razlika u odnosu na originalni signal. Kada signal dođe na odredište, poruka se može izvaditi iz signala korištenjem raznih filtera. Slika 1.2 prikazuje transformaciju poruke u signal male snage i velikog raspona frekvencija.

Jedni od najpoznatijih postupaka iz transformacijske domene su postupci koji se temelje na diskretnim transformacijama signala/slike. Konkretno, tu se koriste razne diskrete transformacije: kosinusna, Fourierova, transformacija valića itd. Ti postupci imaju veliki kapacitet i tako su povoljni za skrivanje slika unutar drugih slika. Funkcioniраju tako da se zbroje transformacije od originalne i skrivene slike putem enkodera te se iz tog zbroja izvuče slika koja se šalje. Zatim se na primateljevom kraju ta slika opet kombinira sa originalnom kako bi se mogao izvući onaj dio transformacije koji

**Slika 1.3:** Model steganografskog postupka temeljen na diskretnoj transformaciji valića (DWT)



odgovara skrivenoj poruci. Pomoću inverzne diskretne transformacije se skrivena slika zatim rekonstruira. Slika 1.3 prikazuje model steganografskog postupka koji se temelji na diskretnoj transformaciji valića (*Discrete Wavelet Transform*).

Opisali smo neke od brojnih steganografskih postupaka i pokazali u kojim situacijama se koriste. Svaki od tih postupaka ima svoje dobre strane i mane stoga je potrebno ovisno o problemu izabrati onaj pravi. U ovom projektu se za zadani problem koristi obični *LSB* postupak zbog svoje jednostavnosti, a dovoljno je dobar za praktičnu primjenu, ali uvijek treba imati na umu i ostale postupke koji su nam na raspolaganju.

### 1.3. Konceptualno rješenje zadatka

## **2. Postupak rješavanja zadataka**

## 3. Ispitivanje rješenja

Razvijeni sustav za steganografski postupak, kao i svi steganografski algoritmi, ograničen je količinom podataka koja se može upisati u neku sliku. Ova granica prvenstveno je određena veličinom slike, a određuju je i odabir kanala te najznačajniji bit do kojeg se upisuju podaci. Glavna ideja steganografskog algoritma jest upisati podatak u sliku bez velikog utjecaja na konačni izgled. Drugim riječima, izgled konačne slike uvjetovan je količinom podataka koji se u nju upisuju. Potrebno je pronaći dobre parametre steganografskog algoritma koji nude dobara kapacitet skrivenih podataka, a neznatno žrtvuju kvalitetu izvorne slike.

Parametri algoritma koji su podešavani u ispitivanju su:

- Najznačajniji bit do kojeg se slijedno upisuje podatak
- RGB komponente u koje će se upisivati podaci

Algoritam *Least Significant Bit(LSB)* upisivanje sadržaja započinje s bitovima najnižeg značaja. Razlog tome leži u tome što se izmjenom bita najmanjeg značaja piksel najmanje mijenja. Praktični primjer bio bi kada bismo odlučili mijenjati samo B(*blue*) komponentu i to samo najniži bit svakog bajta. Za svaki piksel slike(3 komponente, svaka po 1 bajt) dobije se jedan bit prostora za skrivanje podataka. Općenito, količina podataka koja se može upisati( $n_{data}$ ), u ovisnosti o broju komponenti za upisivanje  $n_{components}$  i broja najnižih bitova svake odabrane komponente za upisivanje  $n_{bits}$  te veličini(broju piksela)  $n_{pixels}$  slike je:

$$n_{data} = \lfloor \frac{n_{pixels} \cdot n_{components} \cdot n_{bits}}{8} \rfloor \quad [B] \quad (3.1)$$

Udio veličine podataka  $\eta$  koje je za dane parametre moguće upisati u sliku u odnosu na ukupnu veličinu slike je:

$$\eta = \frac{n_{data}}{3 \cdot n_{pixels}} = \frac{n_{components} \cdot n_{bits}}{24} \quad (3.2)$$

Povećanjem  $\eta$  vizualna razlika između izvorne slike i slike obrađene steganografskim algoritmom, u našem slučaju *LSB*-om, povećava se. U nastavku slijede rezultati ispitivanja odnosa izvorne i obrađene slike u ovisnosti o parametrima  $n_{components}$  i  $n_{bits}$ .

### 3.1. Ispitna baza

Ispitivanje algoritma *LSB* napravljeno je nad dvjema različitim slikama. Slika 3.1 korištena je kako bi se ispitala osjetljivost boja na postupak, dok bi slika 3.2 trebala pokazati osjetljivost tekstura. Ispitni primjeri pripremljeni su u ovisnosti o parametrima  $n_{components}$  i  $n_{bits}$ . Za obje izvorne slike algoritam je pokrenut za sve moguće kombinacije parametara. Parametar  $n_{components}$  poprima vrijednosti iz  $[1, 3]$ , pošto svaki piksel ima crvenu, zelenu i plavu komponentu boje. Parametar  $n_{bits}$  poprima vrijednosti iz  $[1, 8]$ , gdje se za svaki bajt koristi  $n_{bits}$  bitova u koje se upisuje podatak. Ukupno postoje 24 moguća para parametara algoritma. Nadalje, za svaki ispitni primjer odabrana je najveća količina podataka moguća za dane parametre, te su generirani nizovi bitova nasumične vrijednosti. U idućem odjeljku slijede rezultati ispitivanja, koji vizualno prikazuju promjenu kvalitete slike u ovisnosti o parametrima algoritma.

**Slika 3.1:** Izvorna slika jednostavnih boja, bez tekture



**Slika 3.2:** Izvorna slika s teksturom



## 3.2. Rezultati ispitivanja

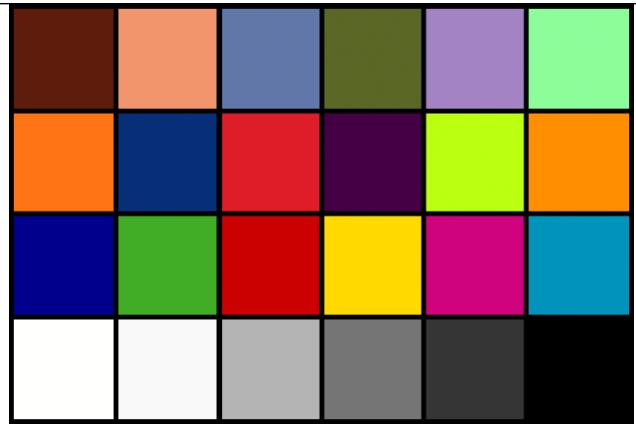
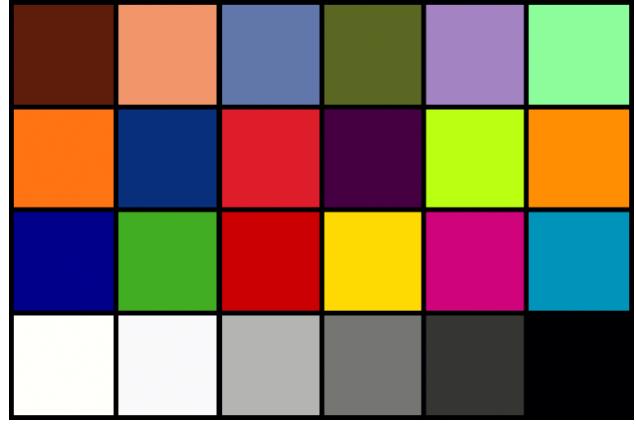
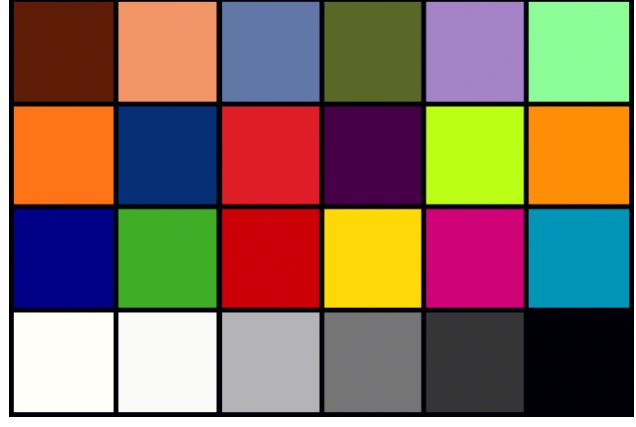
Tablice 3.1 i 3.2 prikazuju obređene slike algoritmom *LSB* za vrijednosti parametara  $n_{components}$  i  $n_{bits}$ . Treći stupac,  $\eta$ , prikazuje udio upisanih podataka u ukupnoj veličini slike.

**Tablica 3.1:** Osjetljivost kvalitete slike jednostavnih boja na *LSB* u ovisnosti o parametrima  $n_{components}$  i  $n_{bits}$

$n_{components}$	$n_{bits}$	$\eta$	Slika
1	1	4%	

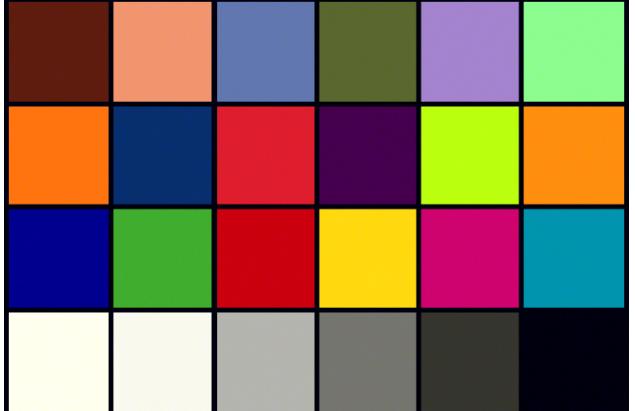
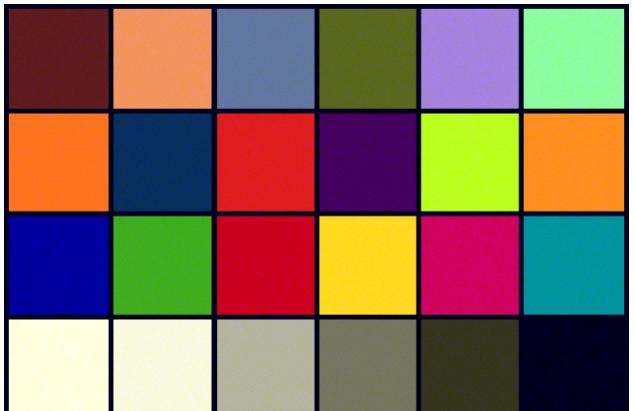
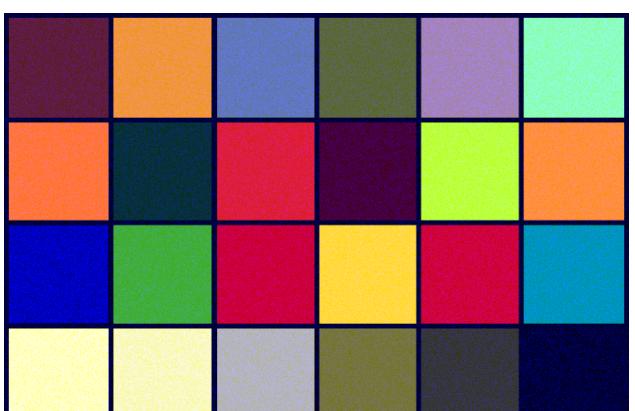
*Nastavlja se na idućoj strani*

Tablica 3.1 – Nastavljen s prethodne strane

$n_{components}$	$n_{bits}$	$\eta$	Slika
1	2	8%	
1	3	12%	
1	4	17%	

Nastavlja se na idućoj strani

Tablica 3.1 – Nastavljeno s prethodne strane

$n_{components}$	$n_{bits}$	$\eta$	Slika
1	5	21%	
1	6	25%	
1	7	29%	

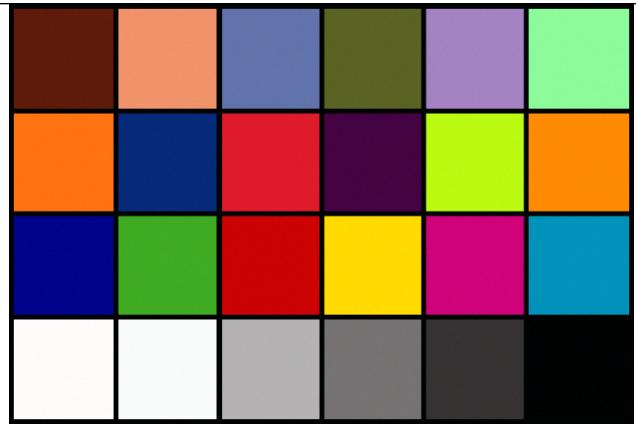
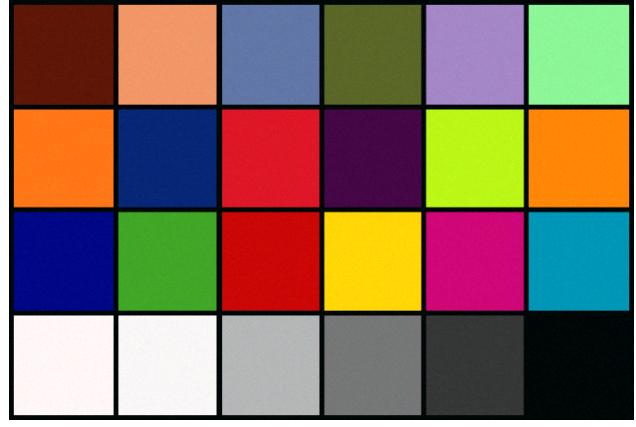
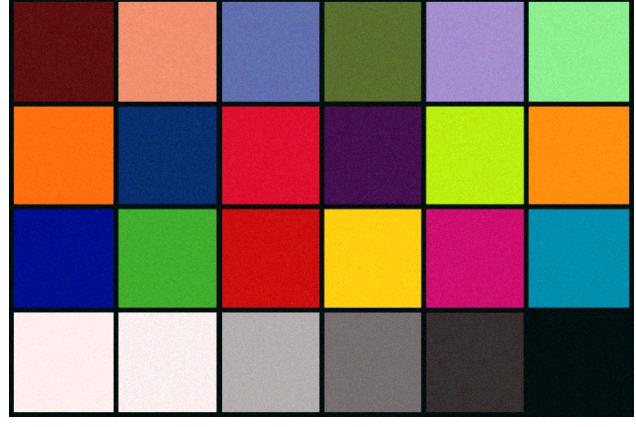
Nastavlja se na idućoj strani

Tablica 3.1 – Nastavljen s prethodne strane

$n_{components}$	$n_{bits}$	$\eta$	Slika
1	8	33%	
2	1	8%	
2	2	12%	

Nastavlja se na idućoj strani

Tablica 3.1 – Nastavljen s prethodne strane

$n_{components}$	$n_{bits}$	$\eta$	Slika
2	3	25%	
2	4	33%	
2	5	42%	

Nastavlja se na idućoj strani

Tablica 3.1 – Nastavljen s prethodne strane

$n_{components}$	$n_{bits}$	$\eta$	Slika
2	6	50%	
2	7	58%	
2	8	67%	

Nastavlja se na idućoj strani

Tablica 3.1 – Nastavljen s prethodne strane

$n_{components}$	$n_{bits}$	$\eta$	Slika
3	1	12%	
	2	25%	
3	3	38%	

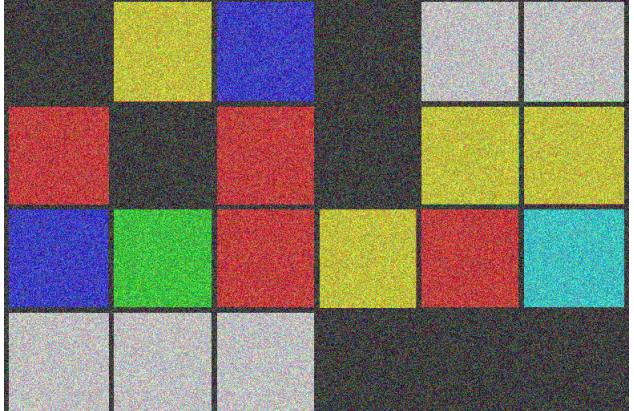
Nastavlja se na idućoj strani

Tablica 3.1 – Nastavljen s prethodne strane

$n_{components}$	$n_{bits}$	$\eta$	Slika
3	4	50%	
3	5	62%	
3	6	75%	

Nastavlja se na idućoj strani

Tablica 3.1 – Nastavljeno s prethodne strane

$n_{components}$	$n_{bits}$	$\eta$	Slika
3	7	88%	
3	8	100%	

Tablica 3.2: Osjetljivost kvalitete slike tekstura na  $LSB$  u ovisnosti o parametrima  $n_{components}$  i  $n_{bits}$

$n_{components}$	$n_{bits}$	$\eta$	Slika
1	1	4%	

Nastavlja se na idućoj strani

Tablica 3.2 – Nastavljen s prethodne strane

$n_{components}$	$n_{bits}$	$\eta$	Slika
1	2	8%	
1	3	12%	
1	4	17%	

Nastavlja se na idućoj strani

Tablica 3.2 – Nastavljen s prethodne strane

$n_{components}$	$n_{bits}$	$\eta$	Slika
1	5	21%	
1	6	25%	
1	7	29%	

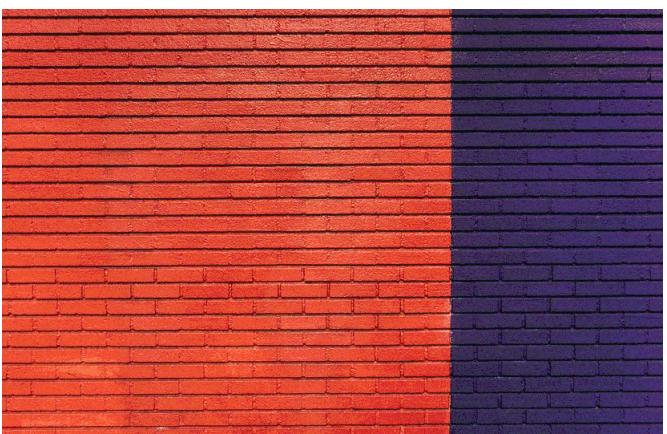
Nastavlja se na idućoj strani

Tablica 3.2 – Nastavljen s prethodne strane

$n_{components}$	$n_{bits}$	$\eta$	Slika
1	8	33%	
2	1	8%	
2	2	12%	

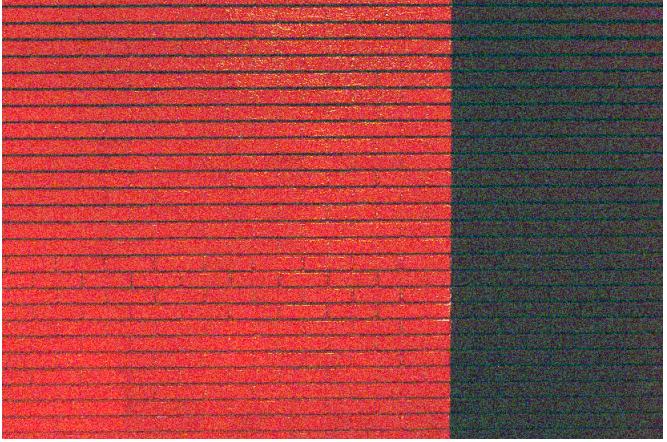
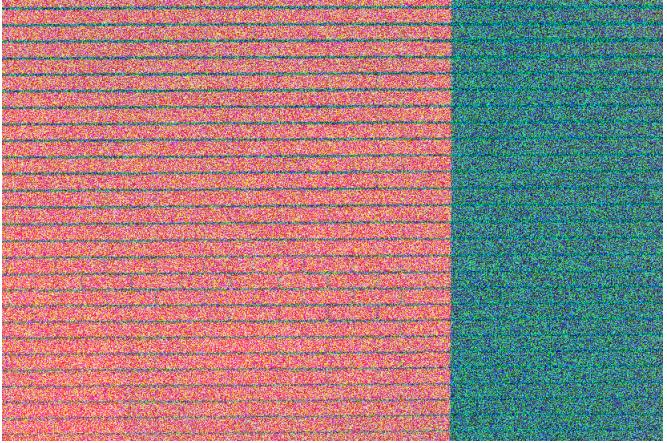
Nastavlja se na idućoj strani

Tablica 3.2 – Nastavljen s prethodne strane

$n_{components}$	$n_{bits}$	$\eta$	Slika
2	3	25%	
2	4	33%	
2	5	42%	

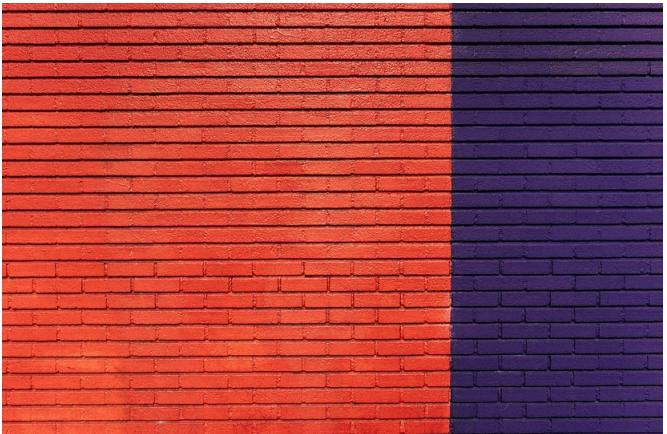
Nastavlja se na idućoj strani

Tablica 3.2 – Nastavljen s prethodne strane

$n_{components}$	$n_{bits}$	$\eta$	Slika
2	6	50%	
2	7	58%	
2	8	67%	

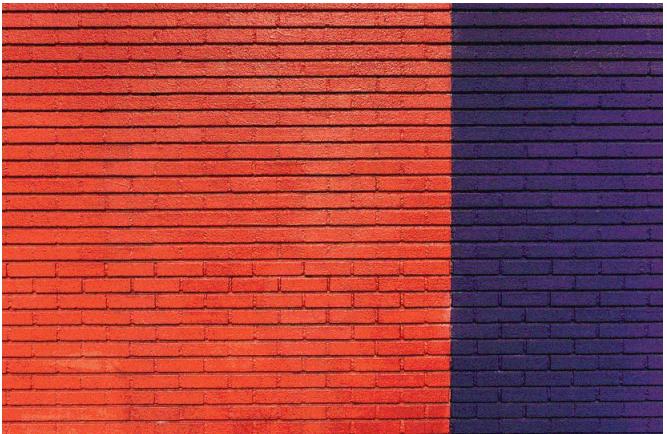
Nastavlja se na idućoj strani

Tablica 3.2 – Nastavljen s prethodne strane

$n_{components}$	$n_{bits}$	$\eta$	Slika
3	1	12%	
3	2	25%	
3	3	38%	

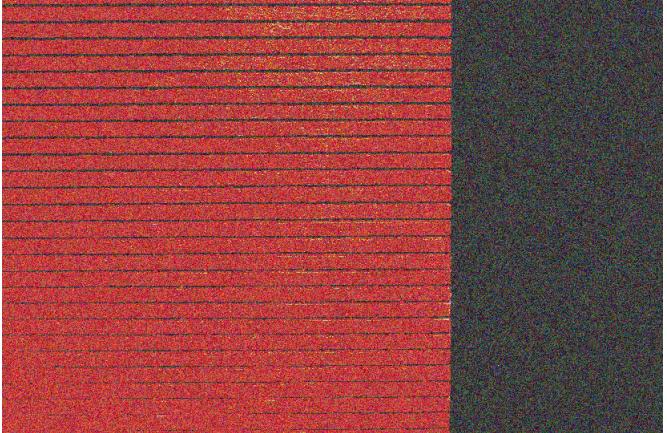
Nastavlja se na idućoj strani

Tablica 3.2 – Nastavljen s prethodne strane

$n_{components}$	$n_{bits}$	$\eta$	Slika
3	4	50%	
3	5	62%	
3	6	75%	

Nastavlja se na idućoj strani

Tablica 3.2 – Nastavljeno s prethodne strane

$n_{components}$	$n_{bits}$	$\eta$	Slika
3	7	88%	
3	8	100%	

### 3.3. Analiza rezultata

Ispitivanje je pokazalo kako upisana količina podataka znatno može utjecati na promjenu izgleda izvorne slike. Bez obzira na prirodu izvorne slike, za  $\eta > 50\%$  obrađena slika sadrži znatnu količinu šuma, i nikako se ne preporučuje korištenje algoritma s parametrima  $n_{bits}$  i  $n_{components}$  za  $\eta > 50\%$ . Nadalje, parametri  $n_{bits}$  i  $n_{components}$  za  $\eta < 25\%$  generiraju slike za koje se ne može razlučiti da je nad njima vršena obrada. Iz ovog područja algoritam se najbolje ponaša za one parametre koji svaki kanal mijenjaju u istoj mjeri iz razloga što je za te parametre izgled kontrasta na obrađenim slikama najprirodniji (svaka boja mijenja se u istoj mjeri). Za  $\eta \in [25\%, 50\%]$  algoritam daje prihvatljiva rješenja, ali samo za one parametre koji svaku komponentu boje mijenjaju u istoj mjeri. Za parametre iz tog intervala koji primjerice mijenjaju 7 bitova samo jedne komponente, slika poprima neprirodan izgled i u velikoj mjeri odskače od

izgleda izvorne slike.

Analiza rezultata potvrđuje izvornu hipotezu da je algoritam osjetljiv na količinu podataka koji se upisuju na sliku. Najbolji rezultati ostvareni su za parametre koji mijenjaju do 2 najmanje značajna bita svake komponente(za  $\eta \in [0\%, 25\%]$ ), što u konačnici znači da se u svaku sliku može upisati količina podataka koja odgovara četvrtini veličine izvorne slike.

# 4. Opis programske implementacije rješenja

Slijedi opis sučelja programske implementacije i način njenog korištenja. Programska implementacija reverzibilne deidentifikacije i steganografskog postupka izvedena je u programskom jeziku *Java*. Implementacija sadrži grafičko korisničko sučelje radi lakšeg i intuitivnijeg korištenja. U nastavku su dani dijagrami razreda i kratak opis programskog koda te način korištenja same aplikacije kroz grafičko korisničko sučelje.

## 4.1. Opis programske implementacije

Programski jezik *Java* odabran je zbog toga što se prevodi u *byte-code* koji se može platformski neovisno izvršavati na svim platformama na kojima je moguće pokrenuti *Java Virtual Machine*, odnosno okolinu za izvršavanje *Java byte-codea*.

Slika 4.1 prikazuje dijagram razreda reverzibilne deidentifikacije i steganografskog postupka. `SteganographyAlgorithm` predstavlja čisto apstraktno sučelje steganografskog algoritma. Implementacije tog sučelja moraju ponuditi 3 metode:

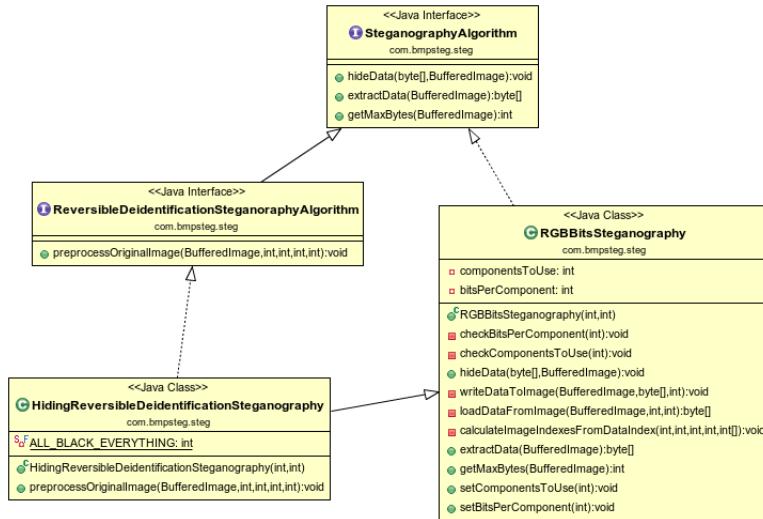
**hideData**) Metoda prima niz bajtova i sliku u koju se ti bajtovi moraju steganografski pohraniti.

**extractData** Metoda prima sliku u kojoj su steganografski pohranjeni podaci i vraća ih u obliku bajtova.

**getMaxBytes** Metoda vraća maksimalni broj bajtova koje korisnik može predati metodi `hideData`.

`ReversibleDeidentificationSteganographyAlgorithm` predstavlja čisto apstraktno sučelje koje nasljeđuje sučelje `SteganographyAlgorithm` i

**Slika 4.1:** Dijagram razreda reverzibilne deidentifikacije i steganografskog postupka



nudi još jednu dodatnu metodu:

**preprocessOriginalImage** Metoda prima sliku u koju se podaci steganografski zapisuju i koordinate pravokutnika koji označava identifikacijsku značajku osobe. Od korisnika koji implementira navedeno sučelje očekuje se da za postupak reverzibilne deidentifikacije u navedenoj metodi "sakrije" identifikacijsku značajku osobe tako da se ona ne može vidjeti na samoj slici, ali će ta informacija biti steganografski zapisana u samu sliku.

**RGBBitsSteganography** predstavlja implementaciju sučelja **SteganographyAlgorithm**. Implementacija je izvedena na način da se podaci koje je potrebno steganografski upisati redom zapisuju u plavu, zelenu i crvenu komponentu od najnižeg bita prema najvišem, ovisno o odabiru parametara koliko komponenti boje i koliko bitova po komponenti se koristi za upis podataka u sliku. Implementacija prije upisa samih podataka prvo upisuje 4 bajta koji označavaju koliko podataka je upisano u sliku tako da pri ekstrakciji podataka iz slike bude dostupna informacija koliko je podataka zapisano u slici. Metoda **getMaxBytes** implementirana je na način da vraća broj bajtova koje je moguće upisati u sliku u ovisnosti o odabranom broju komponenata i broju bitova po komponenti umanjenih za 4 bajta koji se koriste kao informacija o broju upisanih podataka u sliku.

**HidingReversibleDeidentificationSteganography** predstavlja implementaciju sučelja **ReversibleDeidentificationSteganographyAlgorithm** koja nasljeđuje razred **RGBBitsSteganography**. Metoda **preprocessOriginalImage** implementirana je tako da na slici popuni pravokutnik na predanim koordinatama cr-

nom bojom.

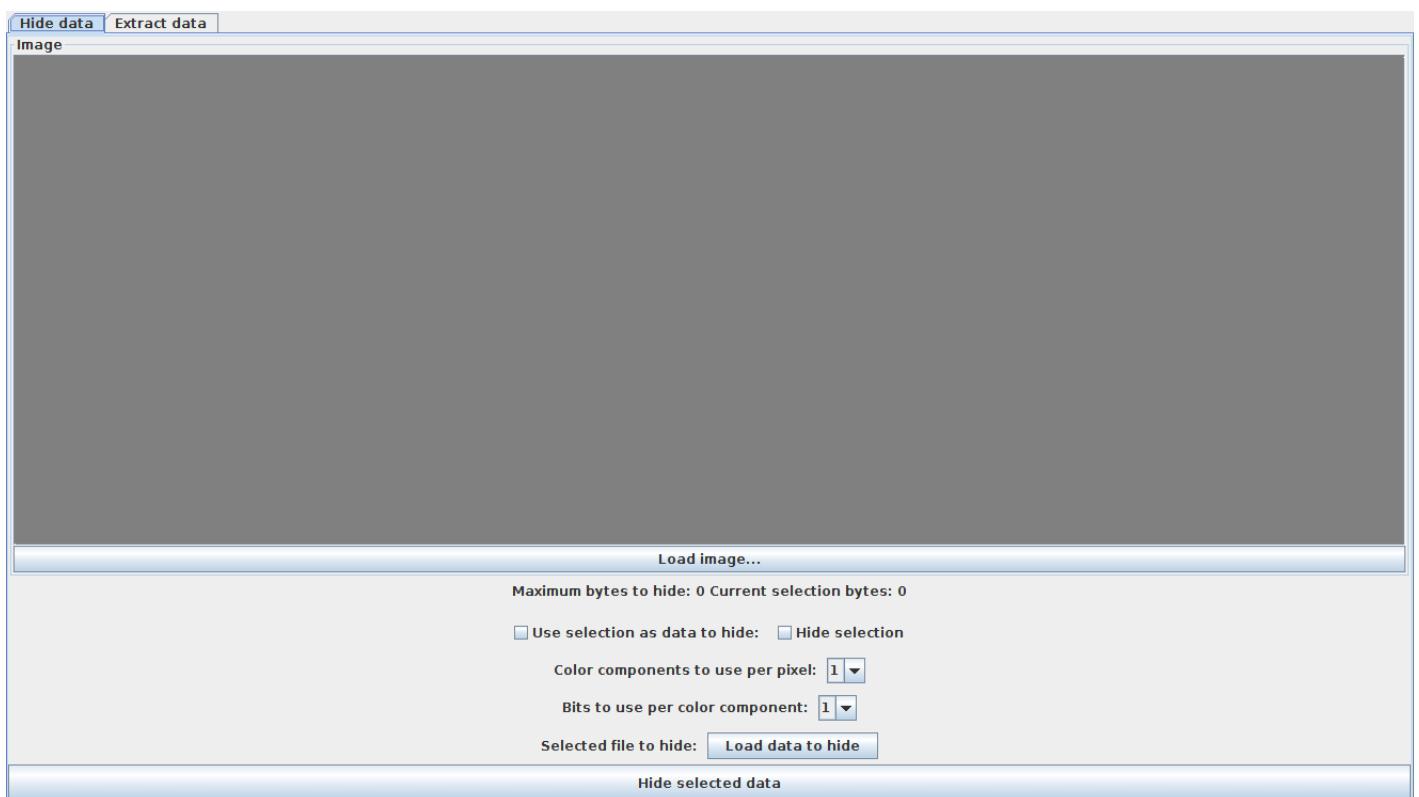
## 4.2. Način korištenja implementacije

Pokretanjem aplikacije vidljivo je kako je kroz grafičko korisničko sučelje moguće izabrati između dvije komponente korištenja:

**sakrivanje podataka** U ovom načinu korištenja moguće je učitati sliku i sakriti podatke u nju.

**ekstrakcija skrivenih podataka** U ovom načinu korištenja moguće je učitavanje slike sa steganografski sakrivenim podacima i njihova ekstrakcija.

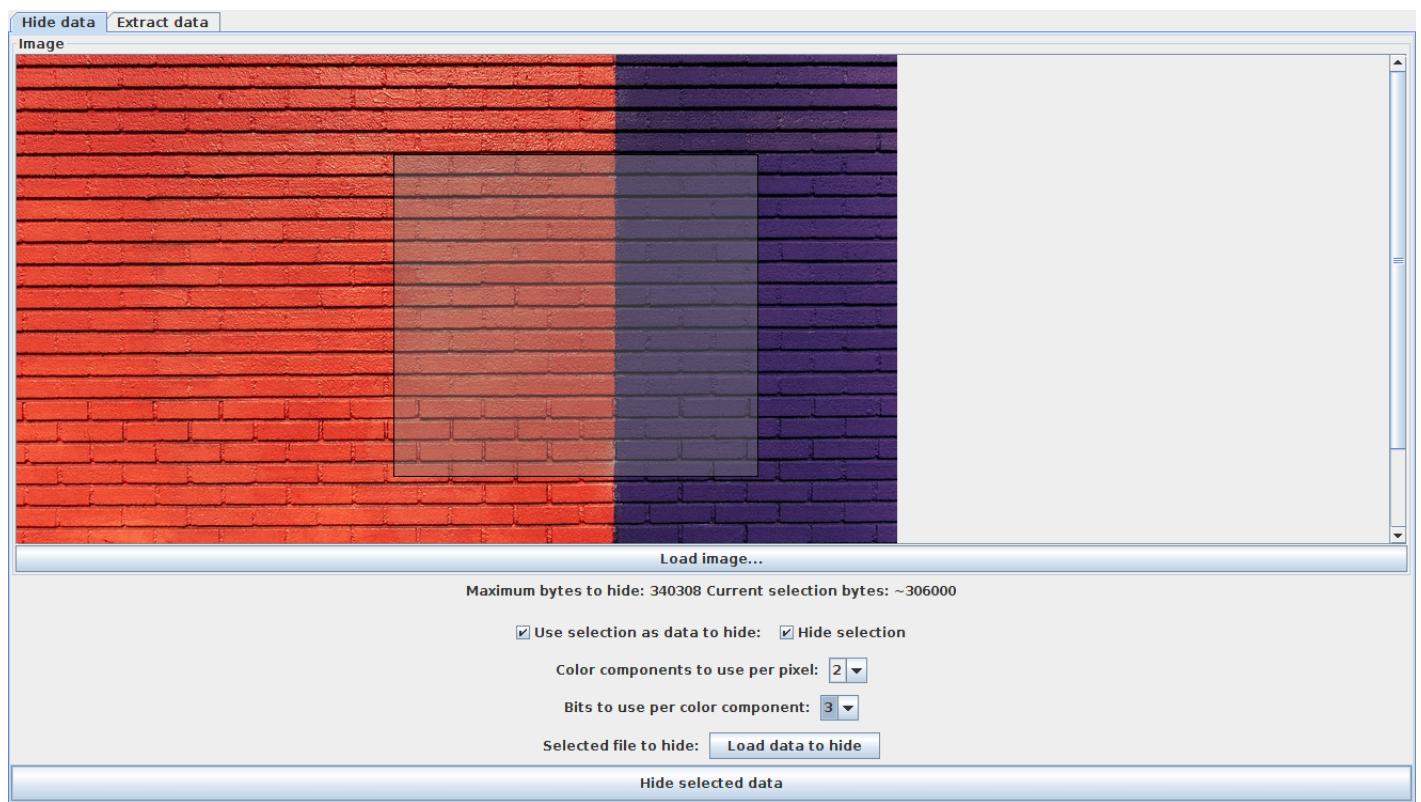
**Slika 4.2:** Početni ekran aplikacije



Slika 4.2 prikazuje početni ekran aplikacije kada je odabran način rada za sakrivanje podataka. Pritiskom na tipku *Load image...* otvara se prozor za odabir slike koja će se koristiti za skrivanje podataka. Ispod tipke za učitavanje slike nalazi se informacija o tome koliko bajtova je maksimalno moguće steganografski zapisati u učitanu sliku koristeći odabrane parametre. Također, ukoliko se koristi označeno područje za sakrivanje podataka, također je prikazana i približna veličina odabranog područja u

bajtovima. Kućica *Use selection as data to hide* predstavlja izbor hoće li se sakrivati podaci koji su označeni na slici ili će se koristiti podaci koje je moguće učitati pritiskom na tipku *Load data to hide*. Kućica *Hide selection* označava izbor hoće li se na slici podaci koji su označeni prekriti crnim pravokutnikom u svrhu skrivanja. Navedene kućice možemo koristiti u svrhu reverzibilne deidentifikacije tako da na slici označimo identifikacijske značajke osobe i označimo obe kućice. Algoritam će koristiti podatke sa slike kao podatke koji se steganografski upisuju i sakrit će označeno područje na originalnoj slici. Iz dobivene slike se naknadno ekstrakcijom mogu izlučiti sakrivene identifikacijske značajke osobe. Padajući izbornik *Color components to use per pixel* omogućuje odabir koliko komponenti boje će se koristiti za sakrivanje podataka. Komponente se koriste sljedećim redoslijedom: prvo plava komponenta, zatim zelena komponenta te naposlijetku crvena komponenta. Padajući izbornik *Bits to use per color component* omogućuje odabir koliko bitova po komponenti boje će se koristiti za steganografsko zapisivanje podataka u sliku.

**Slika 4.3:** Aktivna selekcija dijela slike prilikom sakrivanja podataka



## **5. Zaključak**

## **6. Literatura**