Steganografija i steganaliza

Klasan, Kristijan

Sveučilište Jurja Dobrile u Puli
Fakultet informatike u Puli

Kristijan Klasan

STEGANOGRAFIJA I STEGANALIZA

Završni rad

Pula, rujan, 2019. godine

Sveučilište Jurja Dobrile u Puli
Fakultet informatike u Puli

Kristijan Klasan

STEGANOGRAFIJA I STEGANALIZA

Završni rad

JMBAG: 0303069664, redoviti student
Studijski smjer: Informatika

Predmet: Napredne tehnike programiranja

Znanstveno područje: Društvene znanosti

Znanstveno polje: Informacijske i komunikacijske znanosti

Znanstvena grana: Informacijski sustavi i informatologija

Mentor: doc. dr. sc. Tihomir Orehovački

Pula, rujan, 2019. godine

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisan Kristijan Klasan, kandidat za prvostupnika Informatike ovime izjavljujem da je ovaj Završni rad rezultat isključivo mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Završnog rada nije napisan na nedozvoljen način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

Student

_____

U Puli,_____,_____. godine.

IZJAVA
o korištenju autorskog djela

Ja, Kristijan Klasan dajem odobrenje Sveučilištu Jurja Dobrile u Puli, kao nositelju prava iskorištavanja, da moj završni rad pod nazivom Steganografija i steganaliza koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama.
Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

U Puli,_____.

Potpis

_____

Sadržaj

1. Uvod

Svakoga dana se razvijaju nove tehnologije koje omogućuju ljudima jednostavniju komunikaciju između sebe. Zbog toga sve više ljudi poseže za Internetom kao primarnim medijem za prijenos podataka s jedne strane svijeta pa sve do drugog kraja. Danas postoji mnogo načina za prijenos podataka putem Interneta kao što su: e-mail, chat, itd. Međutim, postoji i nedostatak koji se iskazuje kroz pojam sigurnosne prijetnje tijekom prijenosa podataka. Sigurnosna prijetnja predstavlja pojam koji obuhvaća krađu osobnih ili povjerljivih podataka putem Interneta, a manifestira se kroz: viruse, hakerske napade te ostale Internet prijetnje. Zbog toga je vrlo važno obratiti pozornost na sigurnost podataka kako bi osigurali zaštitu podataka od neovlaštenih korisnika i hakerskih napada kako ne bi došlo do krađe ili mijenjanja podataka. Tijekom godina ovo područje je dobilo više pozornost zbog masovnog prijenosa podataka putem Interneta i njegove same brzine prijenosa podataka. Kako bi se poboljšale sigurnosne značajke, razvile su se mnoge tehnologije kao što su: kriptografija, steganografija i digitalni vodeni pečat. Kako je informacija temelj svega, tijekom prijenosa, usmene informacije su bile podložne promjenama. Da bi se zaštitili podaci, razvijao se pisani prijenos informacija kroz poruke. Sadržaj poruka mogao je biti javan i tajan. Tajanstvenost informacija postizala se šifriranjem sadržaja poruka na različite načine. Takve poruke su bile čitljive samo onim osobama kojima su bile namijenjene, ostalima u prvom trenutku ne. Poruke su bile dostupne i lako ih je bilo pronaći ako sumnjate u njihovu postojanost. Načinima šifriranja i dešifriranja sadržaja poruka bavi se znanstvena disciplina kriptografija.

Za razliku od tajanstvenih sadržaja poruka, prenosile su se i čitljive, jasne informacije. Nositelji poruka su ih skrivali u različite objekte i u postojeće informacije s ciljem da se poruka dostavi tajno i nepromijenjenog sadržaja. Disciplina koja se bavi analizom skrivanja informacija naziva se **steganografija**.

„Steganografija je umjetnost i znanost nevidljive komunikacije“ [1].

Razina vidljivosti je smanjena pomoću mnogih tehnika skrivanja podataka, primjeri: LSB (eng. Least Significant Bit) i „Maskiranje i filtriranje“ (eng. Masking and filtering). Navedene tehnike se izvode različitim steganografiskim algoritmima: LSB, F5, Jsteg itd. Otkrivanje informacija skrivenih kroz algoritme proučava disciplina koja se naziva

steganaliza. Steganaliza (eng. Steganalysis) je nastavak steganografije. To je znanost koja se bavi detekcijom skrivenih poruka. Nakon otkrivanja poruke, steganalizom se može odrediti veličina skrivene poruke, te način izdvajanja iz postojećeg objekta. Razvojem tehnike i Interneta, kriptografske metode lagano se zamjenjuju novim digitalnim metodama gdje je manji rizik otkrivanja informacija tijekom prijenosa. Skrivanje informacija u slikovnoj datoteci sve je raširenija metoda jer je manje sumnjiva i veliki broj fotografija konstantno se prenosi.

Iako su kriptografija i steganografija dvije različite discipline, njih dvije su usko povezane. Skrivene ili ugrađene slike, video i audio datoteke mogu služiti za prijenos privatnih poruka koje će sadržavati ugrađene datoteke. Gledamo li sa stajališta sigurnosti, svaka datoteka može sadržavati maliciozne sadržaje unutar sebe. Primjer: unutar slike možemo ugraditi maliciozni sadržaj koji će predstavljati ozbiljnu prijetnju korisniku te će imati tu datoteku na svojem računalu. Upravo zbog toga se steganografija može koristiti za zlonamjerne radnje koje mogu ugroziti sigurnost na Internetu. Nadalje, steganografske tehnike se mogu primijeniti na različite formate datoteka. Najčešće su u fokusu i dalje slike kao glavni izvor datoteka, te njihovi formati .jpg,.png,.gif. Također, uz navedene discipline, postoji još i digitalni vodeni pečat koji je neophodan dio steganografije. Njegova uloga je zaštititi autorska prava tako što ugrađuje simbol ili logotip u obliku vodenog pečata koji se ne može mijenjati tijekom prijenosa datoteke Internetom. Svaki vodeni pečat je dodatno zaštićen posebnom lozinkom koju zna samo distributer datoteke.

Cilj rada je približiti i pojasniti prednosti uporabe steganografskih metoda. U ovom radu osvrnut ću se na povijesni razvoj skrivanja poruka te njenu podjelu kroz glavne dijelove. Objasnit ću dvije tehnike skrivanja informacija unutar datoteke: metodu najmanje značajnog bita i metodu slika u slici. U radu ću navesti osnovne pojmove vezane uz detekciju informacija, primjenu skrivanja i otkrivanja podataka danas. Osim navedenih pojmova, tu je i primjena programa OpenPuff. Aplikacijski dio temelji se na uporabi i primjeni LSB metode kroz supstituciju bita najmanje važnosti. Programski dio rađen je uporabom programa C#.

2. Steganografija

2.1 Uvod u steganografiju

Pojam steganografija (eng. Steganography) dolazi od grčke riječi čije se ime izvodi od riječi „steganos„ – pokriven i „graphein“ – pisati što u cijelosti znači „skriveno pisanje“. Steganografija je znanost koja se bavi skrivanjem informacija u druge izvore informacije kao što su: slika, tekst, audio i video datoteke. Najčešće se radi o datotekama bezazlenog sadržaja za čiji sadržaj ne sumnjamo da sadrži ugrađene datoteke. Steganografija i kriptografija koriste se zbog sigurnosnog slanja podataka od pošiljatelja do primatelja poruke bez da treća osoba otkrije ili posumnja u njeno postojanje. U steganografiji se poruka čuva u tajnosti bez promjene dok se u kriptografiji izvorni sadržaj promijeni tijekom faza kao što su šifriranje i dešifriranje. Drugim riječima, steganografija skriva informacije kako treća osoba ne bi otkrila njezino postojanje dok je cilj kriptografije prikaz šifrirane poruke s namjerom da spriječi trećoj osobi čitanje njezinog sadržaja. Na primjer, zamislite da osoba pošalje tajnu poruku drugoj osobi tako što zaokruži slova u knjizi koristeći nevidljivu tintu. Kada primatelj primi knjigu sa skrivenom porukom, on može pod posebnim svjetlom pročitati slova koja je zaokružio pošiljatelj. Svatko tko gleda u knjigu ne bi mogao razbiti šifru jer ne bi niti znao da se u njoj nalazi skrivena poruka. Navedeni primjer je klasični primjer steganografije. Međutim, steganografija se razvija svakoga dana pa se kroz povijest njezina postojanja može jasno vidjeti taj razvoj.

2.2 Razvoj steganografije kroz povijest

Zamislimo na trenutak svijet bez: električne energije, telefona, osobnog računala, Interneta. Polazeći od tog trenutka podijelio sam povijesni razvoj skrivanja poruka na dva dijela:

1. Povijesni razvoj steganografije do izuma računala, 1985.godine
2. Suvremeni razvoj steganografije, od računala do danas

S obzirom na korištene tehnike skrivanja podataka, postoje tri osnovna tipa steganografije:

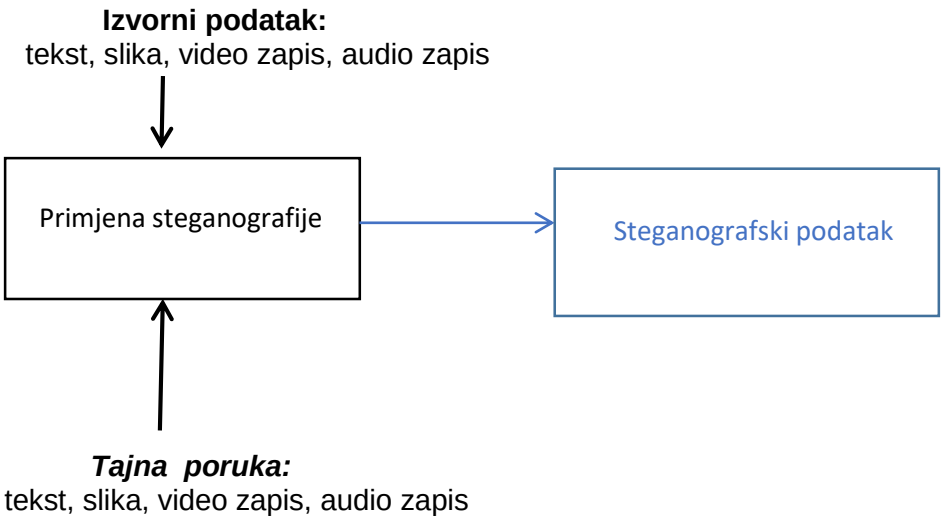
- Tehnička steganografija: obuhvaća one metode koje tajnu poruku skrivaju korištenjem alata, uređaja ili kemikalija i spada u povijesni razvoj do izuma računala.

- Lingvistička steganografija: koristi metode skrivanja tajne poruke u nevažne informacije. Informacije su jezičnog sadržaja. Dijeli se u dvije skupine: semagrami i otvoreni kodovi.
- Digitalna steganografija: predstavlja skrivanje poruka kroz bitove, u digitalnim medijima: slici, audio zapisu ili video zapisu.

Još u Drugom svjetskom ratu, špijuni i borci otpora su pisali poruke nevidljivom tintom, te objavljivali poruke grijanjem dokumenata. Pojava računala, a posebice razvoj Interneta, preselio je steganografiju u digitalno područje. Sve do danas, područje digitalne steganografije se intenzivno razvija te mnogi današnji autori smatraju da je ona budućnost steganografije [2]. Radi lakšeg upoznavanja i preglednosti, tehnike su tablično prikazane.

Tablica 1. Podjela steganografije

Tehnička steganografija	Lingvistička steganografija		Digitalna steganografija
	Semagrami	Otvoreni kodovi	
. skrovita mjesta . urezivanje na tijelo . voštane ploče . voštane kuglice . tinta u jajetu . nevidljiva tinta . mikrofotografija . Morseova abeceda . scitala . bonovi za preživljavanje . rebusi . Tezej i Minotaur u labirintu	. vizualni semagrami . tekstualni semagrami	. žargonski kod i znakovni kod . skrivene šifre: a) rešetkaste šifre b)nulte šifre	. tehnika umetanja . tehnika izmjene najmanje značajnih bitova . tehnika proširenog spektra . tehnika perceptivnog maskiranja . tehnika stvaranja nositelja . tehnika izobličavanja . statistička metoda



Slika 1. Odnos podataka i primjene

Steganografski podaci su se prenosili oduvijek, različitim i maštovitim načinima. Neke od mogućnosti prijenosa detaljnije ćemo upoznati kroz povijesni razvoj. Bez obzira o kojem je tipu steganografije riječ, zajednička im je primjena. Svaki steganografski prijenos mora imati izvorni podatak u koji se skriva tajna poruka. Rezultat steganografskog prijenosa je steganografski podatak.

2.2.1 Tehnička steganografija kroz povijest

Iako pojam steganografije datira tek od 15. stoljeća, primjena skrivanja podataka poznata je još prije Krista.

- **Skrovita mjesta**: prvi načini skrivanja informacija su u skrovitim mjestima. Neka od takvih mjesta su: dvostruki džepovi na odjevnim predmetima, pete obuće i slična mjesta. Dvostruka dna torbi, drvenih bačava i ostalih uporabnih predmeta dugo su vremena bila idealna mjesta za skrivanje. Informacije su se pohranjivale i u zubima čovjeka. Nije nepoznata bila ni metoda gutanja poruke privezane za konac.

- **Urezivanje na tijelo**: tajne poruke su se i urezivale na tijelo čovjeka - glasnika. Glasniku bi obrijali glavu, urezali poruku, čekali da kosa poraste pa tek onda poslali glasnika. Da bi mogli pročitati poruku, morali su glasniku ponovo obrijati glavu. Ovakav način slanja skrivenih poruka koristio se u Grčko - Perzijskim ratovima. Zapis o ovoj tehnici skrivanja poruke može se naći u priči o Histajeju, koji je Aristagoru Miletskog želio nagovoriti na bunu protiv perzijskog kralja.

- **Voštane ploče:** jedan od načina pisanja u staroj Grčkoj je pisanje po voštanim pločama. Drvene ploče bi se prelile voskom i na ohlađenim podlogama pisali čitljiv i vidljiv tekst. Zapis o uporabi drvenih voštanih ploča za tajno pisanje potječe od Herodota u djelu Historije. U svome djelu, opisuje sukobe između Grčke i Perzije u 5. stoljeću prije Krista. Kserkso, despotski vođa Perzijanaca, odlučio je napasti Grčku jer Atena i Sparta mu nisu slali darove i danak. Pet godina je sakupljao vojsku. Naoružavanje Perzijanaca opazio je Demarat, prognani Grk. Odlučio je upozoriti Grke. Herodot je zapisao:

  „*Kako je opasnost od otkrivanja bila velika, poruku je mogao poslati samo na jedan način, i to ovako: s para je sklopivih drvenih pisaćih tablica sastrugao vosak, na drvu napisao što Kserkso kani učiniti, pa poruku prekrio rastaljenim voskom. Tako te pločice, naizgled prazne, nisu mogle izazvati sumnju stražara na putu. Kad je poruka napokon stigla na odredište, nitko, koliko mi je poznato, nije mogao razriješiti tajnu, sve dok se tome nije domislila Gorgona, kći Kleomanta i žena Leonidina pa rekla ostalima neka ostružu vosak i vidjet će da na drvu nešto piše. Tako i učine pa otkriju i pročitaju poruku, te je napokon proslijede i ostalim Grcima“* [3].

  Grci su se počeli pripremati i 23.rujna 480. godine prije Krista došlo je do ratnog sukoba, ali faktora iznenađenja nije bilo. Grci su namamili velike perzijanske brodove u uski zaljev i za jedan dan odnijeli pobjedu nad Perzijancima.

- **Voštane kuglice:** Kinezi su na komadiću svile pisali poruke, svilu presavili i umetnuli u male voštane kuglice ili bi tanku svilu savili u kuglicu i uronili u tekući vosak, te ohladili. Kuglice su često nosili na vidljivim dijelovima odjeće, kao završetke marama, ukrasnih vrpci i slično. Što su bile vidljivije, to su bile manje sumnjive. Ima i primjera da su punjene voštane kuglice glasnici gutali i na taj način prenosili važne poruke.

- **Mikrofotografija i mikrotekst:** tekst poruke ili fotografije smanjio bi se na veličinu točke na papiru, veličine pola milimetra. Točka bi se lako zalijepila na papir, najčešće na slova: i,j. Točke su pisane posebnom vrstom tinte. Glavni nedostatak ove metode je reflektiranje točke pod određenim kutom. Poruka tada postaje uočljiva.

- **Tinta** 1535. godine talijanski znanstvenik Giovanni Battista della Porta je otkrio metodu skrivanja poruke unutar jajeta. Jaje bi skuhao i na ljusku kuhanog jajeta napisao poruku smjesom kalijeva sulfata u octu. Difuzija smjese bi se odvijala polako i nakon izvjesnog vremena na ljusci jajeta nije bilo ništa vidljivo. Poruka se mogla pročitati tek kada se jaje oguli. Poruka je bila vidljiva na krutom kuhanom bjelanjku.

- **Nevidljiva tinta:** koristila se još u 1. stoljeću naše ere. Tako je Plinije Stariji objasnio kako se pomoću „mlijeka biljke mlječike“ dobiva nevidljiva tinta. Sok navedene biljke u suhom stanju je proziran, ali nakon blagog zagrijavanja postaje smeđe boje. Tijekom prvog i drugog svjetskog rata ova metoda se često koristila pri prenošenju ratnih informacija. Pošiljatelj bi napisao pismo čiji tekst ne može privući ničiju pozornost. Između redova teksta ili u praznom dijelu papira napisali bi tajni tekst specijalnom tintom. Takva tinta pripremala se od sljedećih tvari: mlijeka, octa, voćnih sokova, limuna ili urina. Osušeni tekst takve poruke bio bi nevidljiv. Zagrijavanjem papira, dijelovi pisani specijalnom tintom bi potamnili i tekst bi postao vidljiv. Ova metoda se i danas često koristi.

- **Scitala:** prvo kriptografsko pomagalo, poznato u 5.stoljeću prije Krista. To je drvena palica određenog promjera oko kojeg se namata kožna vrpca. Na omotanu vrpcu oko scitale ispisao bi se određeni tekst, a kad se vrpca odmota ostaje niz slova koja nemaju nikakvo značenje. Poruka postaje jasna, tek kad se vrpca ponovo omota oko drvene palice istog promjera. Sama vrpca bi se nosila kao remen ispisana slovima s unutrašnje strane. Ovaj uređaj je kombinacija kriptografije i steganografije. Kriptografski dio odnosi se na pisanje informacije, a steganografski na skrivanje poruke ispod remena.

- „**Bonovi za preživljavanje**“: tijekom I. i II. svjetskog rata pojavili su se letci na kojima je mjesnim jezikom bila ispisana molba za pomoć onima koji ih donesu. Pomoć se sastojala u osiguranju zaklona, hrane i medicinske pomoći. Letke su dobivali zrakoplovci.

- **Morseova abeceda:** poruke pisane Morseovom abecedom skrivale su se na poleđini poštanskih maraka ili u drugim različitim objektima.

- **Tezej i Minotaur u labirintu:** Jedna od metoda umijeća prikrivanja, za razliku od navedenih, ima korijenje u grčkoj mitologiji. Paul Lunde piše o Tezeju i Minotauru:

  „*Legendarno čudovište, napola čovjek, napola bik, potomak kralja Minosa iz Knosa na Kreti, kojem je redovito trebalo žrtvovati djevice s Peleponeza, bio je zatočen i skriven u labirintu ispod palače. U potrazi za čudovištem u namjeri da ga ubije, Tezej dobro pazi da obilježi put, kako bi se vratio, odmotavajući klupko i tako traži pravi put u mračnim tunelima, pa se tako označenim putem nakon uspješne misije vraća na svjetlo dana*“ [4].

  Mitološka priča popularna je na mozaicima i zidnim slikama. Tijekom 19.stoljeća otkriveni su ostaci palače u Knosu i ispod nje opisani labirint.

2.2.2 Lingvistička steganografija

I. Semagrami : u ovoj tehnici skrivanja poruke premještaju se simboli, znakovi ili objekti. Promjene simbola i znakova nose određenu informaciju i ne smiju biti uočljivi.

a) Vizualni semagrami: na svakom radnom stolu nalaze se uobičajeni predmeti/objekti: računalo, bilježnica, olovka, kalkulator, telefon i slično. Svi objekti raspoređeni su na uobičajeni način,tako da budu što dostupniji tijekom rada. Promjena rasporeda objekata na stolu, a da ne bude uočljiva već specifična, temelji su ove tehnike.

b) Tekstualni semagrami: navedena tehnika predstavlja način skrivanja informacija promjenom veličine slova i brojeva, promjenom fonta teksta, uporabom drugog stila pisma, korištenjem jednostrukog i višestrukog razmaka i slično.

II. Otvoreni kodovi

1. Žargonski kod: ova poruka nije skrivena, ona je dostupna svim čitateljima. Problem ove steganografske tehnike je uporaba žargona, jezika kojeg razumije mala skupina ljudi.

a) Znakovni kod: imenovani kod predstavlja podskup žargonskog koda. U ovom kodu određene fraze predstavljaju određene pojmove. Fraze su ranije dogovorene.

2. Skrivene šifre: sama riječ šifre ukazuje da ovu metodu primjenjuju samo oni koji znaju točno dešifrirati poruku. Šifrirana poruka u poruci je temelj ove metode. Postoje dvije vrste skrivene šifre.

a) Rešetkasta šifra: metoda predstavlja niz predložaka koji u svojim otvorima skrivaju poruku.

b) Nulta šifra: tehnika skrivanja podataka je jednostavna i predstavlja određeni slijed, niz pravila, npr. čitanje svakog petog slova. Istaknuta pročitana slova predstavljaju tekst skrivene i važne poruke. Primjer: Velvalee Dickinson, japanska špijunka i preprodavačica lutaka, poznatija je pod imenom Doll Woman (žena lutka). U svojim narudžbenicama lutaka, iz New Yorka u Južnu Ameriku, slala bi skrivene podatke o kretanju brodova.

2.2.3 Digitalna steganografija

Digitalna steganografija prikriva informacije unutar računalnih datoteka. Kao transportni sloj mogu poslužiti: dokumenti, slikovne datoteke, audio datoteke, programi ili protokoli. Veličine multimedijskih datoteka su idealne za steganografski prijenos. Dokumenti spadaju u tekstualne datoteke. Steganografija teksta rjeđe se koristi jer ova vrsta datoteka nema puno suvišnih podataka i teže je skriti poruku. Skrivanje podataka u audio datotekama postiže se metodom maskiranja. Ova metoda koristi ljudsko uho za skrivanje podataka, zvuk u zvuku. Protokoli su oni koji predstavljaju steganografski prijenos informacija unutar poruka i mreže.

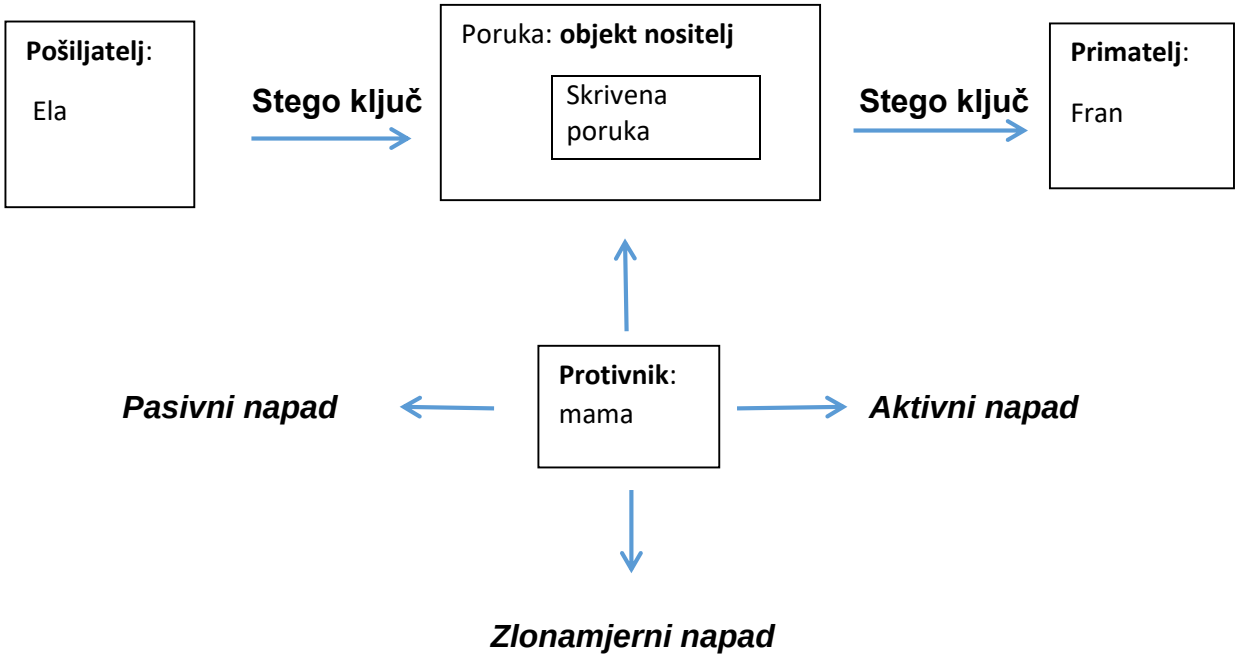
Narednih godina, digitalna steganografija se počela primjenjivati i u sustavima internetske telefonije, kao što je Skype. Istraživanje Sveučilišta Perdue pronašlo je dokaze o zločincima koji koriste alate steganografije uglavnom u dječjoj pornografiji i raznim financijskim prijevarama [2]. Također, steganografija se može manifestirati kroz spyware program koji prima naredbe koristeći slike. Zlonamjerni softver bi komunicirao s rukovoditeljima cyber kriminala koristeći meme (eng. memes) na Twitteru. Nakon što bi došao na žrtvino računalo, zlonamjerni softver bi otvorio relevantni tweet i izvukao upute iz neke slike. Među naredbama su bile: snimite snimku zaslona radne površine, prikupiti sve informacije o pokrenutim procesima, kopirati podatake iz međuspremnika, napisati imena datoteka iz navedene mape [5].

2.3 Načela steganografije

U literaturi, problem zatvorenika, predstavlja primjer steganografskog sustava ili osnova načela steganografije.

Moj primjer steganografskog sustava je mobilno dopisivanje dvoje mladih. Sven i Ela su brat i sestra, a Fran najbolji Svenov prijatelj. Ela i Fran su u vezi. Saznavši za sastanke mladih, mama zabranjuje Eli svaki kontakt s Franom. Sestra moli brata da joj pomogne, na način da poruke dečku šalje preko njegovog mobitela. Sven pristaje, ali pod uvjetom da sestra mjesto i vrijeme sastanka u poruci sakrije u tekstu. Neko vrijeme je komuniciranje i sastajanje bilo neprimjetno. Mama je sumnjala i odlučila provesti istragu. Dok je Sven spavao, tiho je uzela njegov mobitel, te provjerila pozive i poruke. Nije našla ništa sumnjivo, komunikacija dvaju prijatelja. Vratila je mobitel i otišla spavati. U jednom trenutku se prisjetila sadržaja poruka i rečenice koja se ponavljala. Malo čudno. Provjerit će to sutra. Nakon provjere, dosjetila se da malo izmjeni poruku i pošalje. Rezultatima nije bila zadovoljna. Odlučila je riskirati i poslati novu izmijenjenu poruku. Mjesto sastanka je u kući. Sina je poslala u knjižnicu, a za Elu je bila u višesatnoj kupovini. Pozvonivši na vrata, Ela je otvorila Franu, iznenadila se misleći da je brat poslao poruku. Mama je skrivena u kući pratila „sastanak“, te otklonila sve sumnje i realizirala svoj cilj.

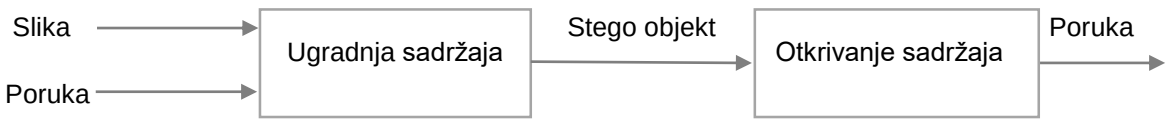
Navedeni primjer komuniciranja je primjer steganografije. Ela i Fran predstavljaju osobe koje komuniciraju na opisani način: Ela je pošiljatelj, a Fran primatelj. Mama je „protivnik“ veze. Sama poruka sadrži skrivenu tekstualnu poruku. Poruka, u kojoj će biti skrivena druga poruka, predstavlja objekt nositelja. Uporabom stego ključa kao dodatne zaštite, dodaje se skrivena informacija u postojeću poruku, ali i izdvaja iz poruke. Zbroj navedena sva tri elementa, objekt nositelj+skrivena poruka+stego ključ, predstavlja steganografski objekt ili skraćeno stego objekt. Analizirajući model steganografskog sustava, primjer stego objekta predstavlja problem komuniciranja. Problem se rješava između pošiljatelja i primatelja umetanjem informacije u poruku, tj. skrivanje poruke u poruci. Mama, kao protivnik veze, sumnja u slanje informacija, te pokušava otkriti. Prvi pokušaj predstavlja slučajno čitanje mobilnih poruka, pasivni napad, bez vidljivih rezultata. U drugom pokušaju mijenja dio skrivene poruke, koja ne donosi očekivana rješenja problema i to je aktivni napad. Treći pokušaj je mijenjanje sadržaja skrivene poruke, koju prilagođava sebi i time dolazi do cilja, dok sam pokušaj predstavlja zlonamjerni napad.



Slika 2. Model steganografskog sustava

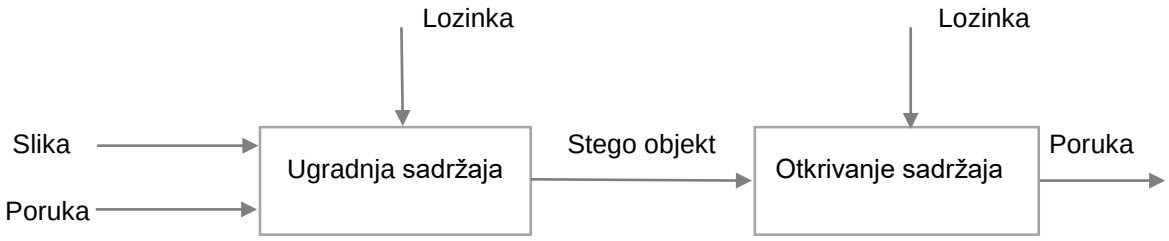
Iz trećeg pokušaja je vidljivo probijanje steganografskog sustava koji se sastoji od tri elementa: otkrivanje skrivene poruke, njeno izdvajanje i mijenjanje sadržaja. Pomoću opisanog primjera se može zaključiti da postupak skrivanja odnosno otkrivanja poruke može biti javan ili tajan. Prema tome, steganografiju možemo koristiti i razvrstati na tri tipa s obzirom na sigurnost tajne poruke:

1. Čista steganografija (eng. Pure steganography): predstavlja proces ugradnje podataka u objekt bez korištenja privatnih ključeva zaštićenih lozinkom. Ova vrsta koristi naslovnu sliku (eng. Cover image) na koju se ugrađuje tajna poruka ili druga datoteka određenog formata te algoritmi za ugradnju same poruke u sliku. Čista steganografija se u potpunosti oslanja na tajnost što znači da ukoliko korisnik posumnja u njen sadržaj, na vrlo jednostavan način može doći do tajnog sadržaja kojeg ona posjeduje. Glavni nedostatak koji odlikuje ovu vrste je upravo lagan dolazak do skrivenog sadržaja.



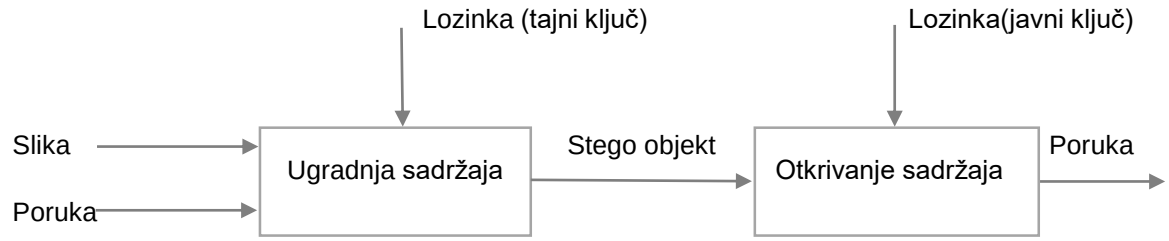
Slika 3. Proces čiste steganografije [6]

2. Steganografija tajnog ključa (eng. Secret key steganography): predstavlja proces koji je načelno isti kao i proces čiste steganografije, samo što ovdje postoji i steganografski ključ. Steganografski ključ je lozinka koja može biti izvedena kao datoteka ili kao parametar koji korisnik mora unijeti ukoliko želi sakriti/otkriti sadržaj podataka. Najčešće se izvodi kao tekst kojega korisnik treba unijeti kako bi mogao koristiti steganografsku tehniku. Lozinku postavlja pošiljatelj poruke. Ponekad se lozinka zaštićuje kriptografskim metodama kako bi se osigurala neprobojnost same lozinke. Nerijetko se postavlja i više lozinki za zaštitu tajnog sadržaja, a baziraju se tako da svaka lozinka mora biti drugačija od one prethodne. Otkrivanje slike se vrši pomoću istog ključa s kojim smo ugradili tajni sadržaj unutar datoteke. Nedostatak ovog procesa jest dijeljenje tajnog ključa, a da ga potencijalni napadač ne pronađe.



Slika 4. Proces steganografije tajnog ključa [6]

3. Steganografija javnog ključa (eng. Public key steganography): predstavlja proces gdje je postupak sličan steganografiji tajnog ključa. U ovom slučaju postoji tajni ključ koji služi za šifriranje poruke i javni ključ koji služi za njeno dešifriranje. Javni ključ se pohranjuje u javnu bazu podataka. Steganografija javnog ključa se koristi kada trebamo prenijeti sadržaj koji će biti dostupan više osoba i koji nije od velike važnosti za napadača. Većina znanstvenika ne prihvaća ovakav proces jer postoji samo u teoriji.



Slika 5. Proces steganografije javnog ključa [6]

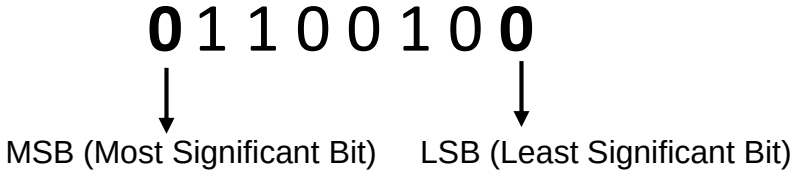
3. Način rada steganografije

3.1. Osnovni pojmovi vezani uz način rada

Steganografija se danas sve više koristi u multimedijalnim datotekama kako bi sakrili tajni sadržaj. U većini slučajeva se koriste slike kao mogući medij za prijenos ugrađenog tajnog sadržaja. U digitalnom svijetu boje se prikazuju kao kombinacije crvene, plave i zelene vrijednosti (tzv. RGB boje). Svakoj od navedenih boja je dodijeljena vrijednost od 0 do maksimuma čija je granica 255. U slučaju da se koristi 8-bitni prikaz jedne primarne boje, tada se može predstaviti 256 nijansi primarne boje. Međutim, budući da imamo tri boje, možemo predstaviti 16 milijuna boja s 24 bita. U multimediji postoje dvije glavne boje, a to su crna i bijela. Crna boja ima RGB vrijednost [0, 0, 0], dok je bijela predočena s RGB vrijednosti [255, 255, 255]. Primjer: ako crvena boja ima vrijednost 237, ona naizgled može izgledati isto kao i boja čija vrijednost može biti između 230-239. Zbog toga, vrijednost određene boje nije toliko bitna za cjelokupni izgled slike, te brojke postaju idealno mjesto za sakrivanje tajnih poruka. Posljednja znamenka je poput jedne u nizu zaokruženih slova u knjizi koji smo sakrili nevidljivom tintom. Ukoliko promijenimo bitove najmanje važnosti neke RGB boje, slika neće izgubiti puno na kontrastu. Promjena bitova se radi kako bi se ugradio tajni sadržaj unutar slike, a da pri tome slika ne promijeni puno u odnosu na originalnu sliku.

Svaka slika se sastoji od piksela koji predstavlja temeljnu jedinicu digitalne slike. Drugim riječima, to je najmanja točka čija se boja može kontrolirati. U digitalnom fotoaparatu, jedan receptorski piksel osjeća boju svjetla koja ga udara i bilježi tu vrijednost kako bi načinila sliku. Na monitoru računala piksel radi obrnuto i emitira boju svjetla. Svaki piksel radi na principu da je postavljen u mrežu i da mu se dodaje određena boja. Svaki kanal R, G i B u RGB prostornom prostoru je predstavljen brojem, a taj je broj predstavljen brojem bitova. U LSB steganografiji se postupak vrši tako da se manipuliraju bitovi najmanje važnosti [2].

Pretvorba broja iz decimalnog u binarni zapis: broj $100_{(10)} = 1100100_{(2)}$



3.2 Elementi steganografije

Osnovni elementi steganografije su oni elementi koji se koriste kako bi unutar njih mogao ugraditi jedan ili više tajnih sadržaja. Neki od osnovnih elemenata su: slika, tekst, audio i video datoteka. Kada korisnik želi primijeniti steganografiju, treba odabrati jedan od prijenosnih medija kako bi unutar njih sakrio tajni sadržaj. U većini slučajeva se koriste slike za skrivanje sadržaja. Slika, kao prijenosni medij, sadrži dva osnovna elementa: naslovnu sliku (eng. Cover image) i tajni sadržaj (npr. neka druga slika koja ima ulogu tajnog sadržaja odnosno predstavlja tajni podatak). Naslovna slika treba imati veliku raznolikost boja kako se ne bi potakla sumnja na postojanje tajnog sadržaja. Slike s malom raznolikošću boja učinit će ugradnju tajnog sadržaja lakšom za otkrivanje. Također, još jedan od važnijih elemenata steganografije su svakako tajni podaci. Podaci koje želimo ugraditi unutar neke datoteke moraju biti manji od datoteke u koju se želi ugraditi. Drugim riječima, podaci moraju biti manji od omota datoteka u koje žele ugraditi sadržaj. Ukoliko želimo ugraditi veću sliku od one naslovne slike, tada će doći do značajne razlike u slikama. Takav će sadržaj, zbog različitosti u nijansama boja i veličini same datoteke, sigurno probuditi radoznalost potencijalnog napadača. Velike datoteke su brzo uočljive i teško se prenose, pa se moraju sažeti. Postupak sažimanja veličina slikovnih datoteka naziva se kompresija. Kompresijom se stvara prostor za pohranu podataka, uporabom matematičkih formula.

Postoje dvije vrste kompresije :

- Kompresija s gubitkom (eng. Lossy): stvara datoteke manje veličine tako da odbacuje suvišne podatke iz izvorne datoteke. Nedostatak ove metode je mogućnost gubljenja dijelova poruke. Primjer navedene kompresije je JPEG format.
- Kompresija bez gubitka (eng. Lossless): ne uklanja podatke iz originala, već predstavlja podatke matematičkim formulama. Najpoznatiji primjeri kompresije bez gubitaka je GIF i 8- bitni BMP.

Jednom, kada se ugradi tajni sadržaj unutar originalnog sadržaja (npr. naslovna slika) nastaje stego objekt. Stego objekt tada sadrži podatke s naslovne slike i podatke sa skrivenog sadržaja. Dodatno, stego objekt može biti ojačan sa sigurnosnom lozinkom, kako bi se spriječio napad na objekt.

3.3 Steganografija teksta

Steganografija teksta se zasniva na skrivanju teksta unutar digitalnog medija. Vjeruje se kako je steganografija teksta najteža od svih ostalih metoda zbog toga što nema suvišnih podataka koje su prisutne u ostalim digitalnim metodama. Pohranjivanje tekstualne datoteke zahtjeva manje prostora. Steganografija teksta sadrži 3 različite metode na kojima se temelji, a to su:

Slika 6. Prikaz podjele steganografije teksta po metodama

1. Metode na temelju formata: uključuju fizičku promjenu formata teksta kako bi se sakrile informacije. Međutim, ova metoda ima i svoje nedostatke. Ukoliko stego datoteku otvorimo s programom za obradu teksta, biti će otkrivene pogrešno napisane riječi i dodatni bijeli prostor. Promijenjene veličine fontova mogu izazvati sumnju kod čitatelja. Ako je izvorni tekst dostupan, uspoređivanjem čistog teksta sa steganografskim tekstom, manipulirani dijelovi teksta biti će svima vidljivi [7].

2. Slučajna i statistička generacija： metoda se temelji na nizovima znakova i riječi. Skrivanje informacija unutar nizova znakova ugrađuje informacije koje se pojavljuju u slučajnom nizu znakova. Taj slijed mora biti slučajan svima koji presreću poruku. Drugi pristup generiranju karaktera je uzimanje statističkih svojstava učestalosti riječi i duljine slova kako bi se stvorile “riječi” (bez leksičke vrijednosti) koje će imati iste statističke osobine kao stvarne riječi u danom jeziku [8].

3. Lingvistička metoda: razmatra jezična svojstva generiranog i modificiranog teksta, često koristi lingvističku strukturu kao mjesto za skrivene poruke. Steganografski podaci mogu se sakriti unutar same sintaktičke strukture [8]. Mana ove metode leži u činjenici da će manji oblik gramatike ponekad dovesti do ponavljanja teksta. Ukoliko se pojavi nedostatak semantičke strukture, pojaviti će se niz rečenica koje međusobno nisu povezane.

3.4 Steganografija u audio zapisu

Metoda skrivanja podataka u audio zapisu koristi se najviše zbog njenog velikog raspoloživog prostora unutar kojeg se može sakriti tajni sadržaj. Ovu metodu koriste istraživači kako bi sakrili velike podatke koristeći audio signale kao nositelje tajne poruke. Drugim riječima, ova metoda se oslanja na umetanje tajne poruke u zvučni signal (audio signal) na kojem se vrši mijenjanje binarnog slijeda originalne zvukovne datoteke. Audio datoteke su danas dostupne u različitim formatima, poznatiji formati su: WAV i MP3 format. U audio steganografiji, naslovnica je audio datoteka, a tajna informacija može biti tekstualna datoteka, slika ili zvuk [9].

Najpoznatije metode koje koriste metodu steganografije u audio zapisu su:

1. LSB metoda: originalni audio zapis se pretvara u binarni zapis, te se nad njime vrši supstitucija najmanje značajnog bita. Na mjesto najmanje značajnog bita zvuka, ugrađuje se bit tajne poruke. Postupak se izvodi sve dok se svi bitovi ne ugrade pod audio signal. Ovakav pristup će omogućiti da je bit koji se zamjenjuje jednak onome s kojim ga želimo zamijeniti. Na taj način, pola vremena se bit ne mijenja što ne dovodi do smanjenja degradacije kvalitete zvuka.

2.Raspršeni spektar: ova metoda pokušava širiti tajne informacije preko frekvencijskog spektra audio signala. Cilj metode je iskoristiti slabosti slušnih organa čovjeka kako bi sakrili tajni sadržaj u zvuk. Ovakav sustav je sličan sustavu koji koristi LSB implementaciju koja širi poruke bitova preko cijele zvučne datoteke. Razlika u odnosu na LSB metodu je ta što se tajni podaci prenose pomoću koda koji je neovisan o stvarnom signalu. Jedna od najčešće korištenih formata je WAV audio datoteka. Ona se sastoji od velikog broja audio uzoraka koji su slični pikselima na slici. Poruka se lako može sakriti u WAV datoteci mijenjanjem LSB bitova. Kao rezultat, konačni signal zauzima pojas koji je veći od prijenosnog. Problematika ove metode odgovara rezultatu koji koristi niske frekvencije koje su primjetne ljudskom uhu [10].

Jedna od najpoznatijih besplatnih aplikacija za skrivanje tajnog sadržaja unutar zvuka je DeepSound. Imenovana aplikacija obuhvaća skrivanje i otkrivanje tajnog sadržaja unutar zvuka. DeepSound podržava šifriranje tajnih dokumenata s AES-256 algoritmom kako bi zaštitio autorska prava. Također, omogućuje skrivanje tajnog sadržaja u različitim formatima zvuka. Neki od njih su: FLAC, MP3, WMA, WAV, APE.

3.5 Slikovna steganografija

Kao što je već navedeno, slike predstavljaju danas najraširenij medij steganografskog prijenosa informacija. Kompleksnost slikovne steganografije uvjetuje podjelu na metode:

- metoda prostorne domene
- metoda frekvencijske domene
- HTML metoda

Tehnike steganografskog zapisa, u kojem je glavni nositelj slika, dijele se na dvije skupine: u domeni slike i domeni transformacije, kao što je prikazano u tablici.

Tablica 2. Domene slike i transformacije

Domena slike	**Domena transformacije**
- ugrađuje poruke direktno u intenzitet piksela	- slike se prvo transformiraju
- metoda umetanja bita	- uključuje uporabu algoritama
- jednostavni sustavi	- skrivanje poruka na dijelovima naslovnica
- kompresija bez gubitaka	- neovisne o formatu slika
- tehnike ovise o formatu slike	

3.5.1 Metode prostorne domene

Metode prostorne domene temelje se na modifikaciji svih elemenata, metodom supstitucije ili zamjene. Slike, kao nositelj datoteke i umetnute informacije, podložne su RGB sustavu koji je opisan u poglavlju “Digitalna steganografija“.

3.5.2 Supstitucija bita najmanje važnosti

Jedna od jednostavnijih metoda skrivanja informacija u slike je supstitucija bita najmanje važnosti ili tzv. LSB metoda. Bit najmanje važnosti vezan je uz numeričku vrijednost bitova u oktetu. Tako, bit najveće važnosti predstavlja najveću aritmetičku vrijednost, a bit najmanje važnosti najmanju aritmetičku vrijednost. Sve promjene koje se događaju unutar okteta, odvijaju se na zamjeni bita najmanje aritmetičke vrijednosti

jer neće imati velike posljedice na slikovnu datoteku. Promjene koje se događaju, čovjek često puta ne može percipirati svojim osjetilom vida, ali u nekima ipak može. Na primjer, ugrađivanje u najmanje značajan bit mijenja vrijednost boje za jedan. Ugrađivanje u dva bita, može uzrokovati promjenu vrijednosti boje za dva, pa tako ponovno dobivamo drugačiju vrijednost. Steganografija izbjegava uvođenje što više varijacija, kako bi se smanjila vjerojatnost otkrivanja. Ukoliko se koristi LSB metoda, tada se gube neki podaci iz naslovne slike. Kako bi se to postiglo, potrebno je odbaciti bitove najmanje važnosti te umjesto njih ugraditi bitove skrivene poruke unutar naslovne slike. Na taj način stego slika će promijeniti par nijansi boja u odnosu na originalnu naslovnu sliku, te neće biti vidljiva ljudskom oku. Rastavljanje informacije na bitove ne možemo postići bez poznavanja ASCII koda.

Primjer LSB metode: skrivanje slova „K“ u niz zadanog okteta:

10010101 00001101 11001001 10010110

00001111 11001011 10011111 00010000

Binarni zapis slova „K“ **: 01001011**.

Ovih osam bitova upisuje se na mjesto najmanje važnosti, te dobivamo:

1001010**0** 0000110**1** 1100100**0** 1001010**0**

0000111**1** 1100101**0** 1001111**1** 0001000**1**

Navedeni primjer djelomično je supstituiran.

Skrivanje podataka u slikovnim datotekama metodom LSB najčešće se koriste 24-bitne BMP slike. Slikovni element je predstavljen kombinacijom crvene, plave i zelene boje. Svaka od navedenih boja ima kanal veličine osam bita. Kod jednog slikovnog elementa moguće je sakriti tri bita, tako da čovjek ne primijeti. Zbog svoje veličine i česte primjene koriste se i 8-bitna BMP slika ili format GIF, u kojem je jedan piksel prezentiran jednim bajtom. Promijenimo li format slici u kojoj je skrivena poruka, može doći do oštećenja slike ili potpunog gubljenja tajne informacije. Spy Pix je iPhone aplikacija koja koristi steganografiju za slanje tajnih poruka. Prihvaća dvije slike iz iPhone kamere i ugrađuje ih jednu u drugu te nudi klizač u kojem se mogu odabrati varijante s bitovima pomoću kojih ćemo ugraditi sliku i dobiti na kvaliteti slike [11].

3.5.3 HTML metoda

HTML steganografija je dio teksta steganografije koji koristi HTML web dokument kao nositelj skrivene poruke. Korištenje HTML-a ima neke prednosti kao što je velika količina dokumenata dostupna za skrivanje podataka i time je dekodiranje tih podataka od strane neovlaštenog korisnika vrlo teško [12]. Danas postoji više metoda koje koriste HTML metodu, a neke od njih su:

1. Promjena redoslijeda atributa

Metoda se bazira na slijedu atributa (eng. bgcolor,background), rezultat je bit 0. Ukoliko se promijeni redoslijed (eng. background,bgcolor) tada je rezultat bit 1. Mana ove metode je što nije dovoljno sigurna ako se gleda kroz aspekt sigurnosti.

<body bgcolor = "slika.png" background=" #000000"> —Rezultat je→ 0

<body background = "#000000" bgcolor = "slika.png" > —Rezultat je→ 1

Daljnjim navođenjem boja, korisnik ovom tehnikom samo treba primijeniti gornji ključ kako bi pročitao poruku koja je skrivena u HTML dokumentu.

2. Prazni prostor (eng. White Space)

Metoda se bazira na ostavljanju jednog praznog prostora nakon što se napiše neka određena naredba u HTML-u. Kako bi koristili ovu metodu potreban je stego ključ.

<tag>.</tag>tekst<tag/> —Rezultat je→ 0

<tag >.</tag>tekst<tag/ > —Rezultat je→ 1

} Stego ključ

<kupac ><ime>Karlo</ime ><id >1234</id></kupac> Tekst → 101100 rezultat

3. Izmjena pisanog stanja slova

Metoda prati binarni zapis koji želimo sakriti u HTML. Vrijednost bita 1 označava veliko slovo, a bit 0 kao rezultat ostaje isti tj. malo slovo. Primjer. 01011010

```
<html>                                  <hTmL>
  <head>                                  <HeAd>
    <title> Izmjena </title>   Rezultat →    <title> Izmjena </title>
  </head>                                 </head>
</html>                                 </html>
```

3.5.4 Skrivanje slike u slici

Ova metoda temelji se na LSB metodi. Najčešće se koristi metoda supstitucije 4 najmanje značajna bita slikovnog elementa nositelja slike, te se ugrađuju 4 najvažnija bita tajne slike. Međutim, ponekad je korisniku ponuđeno da sam odabere koliko bitova želi koristiti za skrivanje. Što je bit važniji, tajna slika će biti manje uočljiva. U sliku nositelja se umeće tajna slika i tako se dobiva stego slika. Ovom tehnikom djelomično se gubi dio tajne slike. Glavne dvije boje u steganografiji su crna i bijela boja. Ako uzmemo u obzir bijelu boju i njen zapis u binarnom obliku, možemo predočiti vrijednost svakog bita čija je vrijednost potrebna za skrivanje podataka unutar slike.

Tablica 3. Prikaz vrijednosti pojedinog bita

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
1	**1**	**1**	**1**	**1**	**1**	**1**	**1**

Ukoliko koristimo LSB metodu s 4 najmanje značajna bita, tada binarni zapis dijelimo na pola. Prva polovica će biti predstavljena kao MSB, a druga kao LSB. U nastavku, druga polovica će biti zamijenjena s binarnim zapisom tajnog sadržaja. U kodiranju se isti postupak vrši za svaki oktet RGB boje, ali u ovom slučaju uzet ćemo u obzir samo jedan slučaj boje. Zamislimo da imamo slučaj binarnog zapisa naslovne slike i binarni zapis tajne slike koju želimo ugraditi u naslovnu sliku.

Primjer: na raspolaganju imamo dvije slike, originalnu i tajnu sliku. Korisnik aplikacije želi ugraditi tajnu sliku pod originalnu koristeći metodu slika u slici LSB metodom s 4 bita. Dani su binarni zapisi dviju slika, potrebno je izvršiti zahtjev korisnika.

Binarni zapis originalne (tzv. naslovne) slike: $01110101_{(2)}$

Binarni zapis tajne slike: $10110010_{(2)}$

Rezultat:

0 1 1 1 1 0 1 1

1. korak: podijeliti binarni zapis originalne i tajne slike na dva dijela, te odrediti MSB i LSB bitove, ukoliko je riječ o LSB-u s četiri bita.

Naslovna slika (originalna slika):

0 1 1 1 | 0 1 0 1

MSB LSB

Slika koju želimo sakriti (tajna slika):

1 0 1 1 | 0 0 1 0

MSB LSB

Svaki binarni zapis sada je podijeljen na dva dijela: MSB i LSB bitove. MSB bitovi kod originalne slike trebaju ostati, dok se LSB bitovi originalne slike odbacuju kako bi na njihovo mjesto mogli doći MSB bitovi slike koju želimo sakriti pod originalnu sliku. LSB bitovi tajne slike se također odbacuju.

2. korak: odbacivanje nepotrebnih bitova i dodavanje MSB bitova tajne slike. Kako je riječ o metodi s 4 bita, potrebno je odbaciti 4 bita najmanje važnosti originalne slike te dodati nova 4 bita koja će zamijeniti odbačene bitove. Za odbacivanje bitova koristi se matematička operacija dijeljenja i vrijednost bita s kojim se dijeli binarni zapis. U ovom slučaju, kako bi odbacili 4 bita najmanje važnosti, podijeliti ćemo binarni zapis sa 16 zbog toga što vrijednost 16 obuhvaća bitove koje želimo odbaciti. Primjer: ako želimo odbaciti 2 bita iz originalne slike, tada dijelimo binarni zapis brojem 4. U slučaju da želimo odbaciti 3 bita, dijelimo brojem 8, itd.

Naslovna slika: 01110101 / 16 = 0111 ⟶ 0 1 0 1 ostatak se odbacuje

Tajna slika: 10110010 / 16 = 1011 ⟶ 0 0 1 0 ostatak se odbacuje

Nakon što odbacimo nepotrebne bitove, potrebno je oktet nadopuniti novim bitovima. U ovom slučaju, koristiti ćemo 0 kao bitove s kojima popunjavamo prazninu. Kako bi to postigli, rezultat dijeljenja kod originalne slike množimo sa 16 kako bi dobili nova 4 bita vrijednosti 0. Naime, množenje s rezultatom dijeljenja se koristi samo kod originalne slike kako bi dobili novu sliku koja će biti slična originalnoj slici. Tajnu sliku ne množimo s rezultatom dijeljenja jer bi tada dobili sliku gotovo jednaku tajnoj slici. Kontrast iste slike se ne bi puno promijenio, te bi tako ugrađena slika u originalnoj slici bila odmah percipirana za potencijalnog napadača. Kako se to ne bi dogodilo, tajna slika zadržava svoj rezultat dijeljenja i naknadno se dodjeljuje binarnom zapisu originalne slike. Binarni zapis se množi sa 16 kako naslovna slika ne bi izgubila puno na kontrastu. Množenje binarnog zapisa naslovne slike sa 16:

0111 * 16 = 01110000 => nova slika slična originalnoj, razlika je u nijansama boja.

Rezultat dijeljenja tajne slike ostaje isti nakon dijeljenja. Drugim riječima, stvara se nova slika koja će biti zatamnjena u odnosu na originalnu tajnu sliku. Postupak se radi kako bi se tajna slika mogla neprimjetno ugraditi u originalnu sliku. Dobivanje stego slike se postiže zbrajanjem binarnih zapisa tajne i originalne slike. Konačni rezultat: 01110000 + 1011 = 01111011 -> binarni zapis stego slike.

Tablica 4. Prikaz slika koje prolaze kroz metodu skrivanja slike u slici s 4 bita

Naslovna slika	**Tajna slika**
	
Slika 2a. Dijeljenje i množenje sa 16	**Slika 2b. Rezultat dijeljenja sa 16**
	
Stego slika	**Objašnjenje**
	**= Stego slike** Stego slika je rezultat zbrajanja binarnog zapisa naslovne slike i binarnog zapisa tajne slike (slika2a + slika2b). Npr. 01110101 / 16 = 0111 10110010 / 16 = 1011 0111 * 16 + 1011 = 01110000 + 1011 Rezultat = 01111011 -> Stego slika

4. Steganaliza

4.1 Uvod u steganalizu

Sve što je skriveno nema nikakvog značaja, ako i ne otkrijemo. Znanost koja se bavi otkrivanjem steganografski skrivenih poruka naziva se steganaliza (eng. Steganalysis). Ova vještina temelji se na usporedbi uzoraka bitova, najčešće velikih datoteka. Cilj steganalize nije samo otkriti sumnjive podatke uspoređujući bitove podataka, već i procijeniti veličine skrivenih podataka. Tijekom umetanja podataka, da bi postupak bio kompleksniji, često se ubacuju podaci nebitnog sadržaja. Ti podaci bi trebali steganalitičaru oduzeti dosta vremena i znanja uz računalnu tehniku, pri otkrivanju sadržaja poruke. U slučaju da je poruka još i kriptirana, proces će se dodatno zakomplicirati. Uz sve to, treba obratiti pozornost na mogućnost postojanja šuma u datoteci. Nakon svih tih postupaka, novi korak je izdvajanje skrivene poruke. Samim izdvajanjem poruke, proces steganalize nekada nije završen. Izdvojena poruka može biti temeljena na kriptografiji i u tom slučaju se pristupa postupku dešifriranja. Ukoliko nije moguće pročitati rezultat steganalize može se izvršiti proces uništavanja datoteke. Osim namjernog uništavanja podataka, može doći i do nenamjerne promjene sadržaja datoteka koje se mogu očitovati promjenom određenih svojstava uporabivog medija. Koji će od načina detektiranja skrivene poruke i njenog izdvajanja iz datoteke uporabiti mora odlučiti steganalitičar. Tijekom odlučivanja, veliku ulogu imaju dostupne informacije. Djelovanje steganalitičara opisuje se kao vrsta napada na konkretnu datoteku. U literaturi postoji nekoliko oblika napada.

Oblici napada:

- Poruka (eng. Chosen-message attack) - poznata nam je tajna poruka, te steganografska datoteka (eng. Steganography-only attack) nad kojom će se provoditi različite analize.

- Poznati nositelj (eng. Known-carrier attack) - raspoloživi su i steganografska datoteka i steganografski nositelj, tj. izvorna datoteka unutar koje je skrivena tajna poruka.

- Poznata poruka (eng. Known-message attack) - dostupna nam je tajna poruka.

- Odrabrana steganografska tehnika (eng. Chosen-steganography attack) - poznata je i steganografska datoteka i steganografski alat, odnosno algoritam koji je korišten za umetanje tajne poruke.

- Odabrani steganografski alat, odnosno algoritam korišten za kreiranje steganografske datoteke. Svrha ovog napada jest utvrditi odgovarajuće uzorke u steganografskoj datoteci koji nam mogu ukazati na korištenje određenog steganografskog alata i algoritma [13].

Poznatije tehnike steganalize su metoda neobičnih uzoraka i vizualna detekcija.

Neobični uzorci: unutar steganografskih datoteka impliciraju na potencijalno skrivenu poruku unutar istih. Upotrebom različitih alata i tehnika, moguće je identificirati te uzorke. Npr. alatima za analizu diska moguće je filtriranjem pronaći skrivene informacije u nekorištenim dijelovima. Različiti filtri mogu poslužiti za identificiranje TCP/IP paketa koji sadrže skrivene ili neispravne podatke unutar svog zaglavlja. Pregledom teksta unutar nekog tekstualnog procesora moguće je pronaći male nepravilnosti kod razmještaja riječi i redaka ili suvišne razmake koji impliciraju na postojanje skrivene poruke. Slike mogu sadržavati izobličenja te varijacije u boji koje nakon identificiranja određenim alatom upućuje na prisutne skrivene informacije [14].

Vizualna detekcija: jedna od poznatijih tehnika steganografije je i vizualna detekcija. Sam naziv ukazuje da se ova tehnika temelji na vizualnim metodama. Iskustvo ima veliku ulogu u ovom dijelu. Uspoređuju se podaci, najčešće slikovni medij. Analizira se slika i prate nijanse promjene boja. Tehnika je najjednostavnija za uporabu ako postoji glavni nositelj ili original slika. U navedenom slučaju, uspoređujući slike, mogu se uočiti nepravilnosti u bojama ili kvaliteti slike. Iskusniji steganalitičari lako mogu otkriti steganografski alat, kojim je informacija skrivena, pa je problem riješen. U mnogim slučajevima je problem kompleksniji jer nemamo originalnu sliku. Kod takvih primjera obraća se pozornost na veličinu slike. Detaljno se analizira da li je: izrezana, skraćena ili nadopunjena slika. Osim veličine, treba obratiti pozornost i na paletu boja unutar slike. Često puta su slike prepune boja koje ne odgovaraju ili su tonovi jednolični. I u jednom i udrugom slučaju vizualna detekcija paleta boja ukazuje na mogućnost steganografskog podatka. Vizualna detekcija je stara tehnika, koja ne zastarjeva.

4.2 Otkrivanje teksta iz slike

Jedan od glavnih čimbenika steganalize je otkrivanje teksta iz slike. Ukoliko korisnik sumnja na postojanje skrivenog teksta, potrebno je sliku podvrgnuti programima koji se baziraju na steganalizi odnosno otkrivanju teksta. Prilikom otkrivanja teksta mogu se koristiti različiti postupci otkrivanja, kao što su: DCT, LSB, EOF tehnike. U ovom dijelu rada ću objasniti dvije poznatije tehnike za otkrivanje teksta, a to su: LSB tehnika i tehnika zamjena boje.

1. Tehnika se oslanja na otkrivanje teksta koristeći LSB metodu. Cilj ove metode je izdvojiti zadnje bitove svakog piksela za koje se smatra da su nositelji teksta. Takvi bitovi se potom pretvaraju u bajt, a bajt u slovo prema ASCII tablici. Prednost ove tehnike se ogleda u usporedbi stego i originalne slike. Naime, ljudsko oko često ne može detektirati razliku između tih dvaju slika. U nekim slučajevima, mogu se izdvojiti dva ili više bita iz piksela, ali se njihov sadržaj može puno lakše detektirati. Najčešće se danas izdvaja jedan bit jer neće uzrokovati oštećenje slike i time ugroziti sigurnost sadržaja od potencijalnog napadača. Ponekad je postupak otkrivanja otežan jer postoji prisutnost kriptografije koja se manifestira kroz lozinku koja zaštićuje tajni sadržaj. Upravo zbog toga, danas još uvijek ne postoje programi koji uspješno rješavaju sve oblike steganografije.

2. Tehnika se odnosi na zamjenu jedne RGB boje piksela s vrijednosti jednog slova. U ovoj tehnici skrivanja teksta je potrebno odrediti veličinu teksta kojeg želimo ugraditi u sliku. Veličina teksta označava broj piksela koji je potreban kako bi se poruka uspješno sakrila. Kako bi ugradili jedno slovo, potrebno je vrijednost slova zamijeniti s vrijednosti neke od RGB boje piksela. Postupak se izvodi iterativno, sve dok se sva slova ne sakriju pod sliku. Međutim, otkrivanje teksta je nešto otežano jer steganalitičar ne zna koliko je slova skriveno pod sliku i ne zna pod kojom bojom se slovo nalazi. Rješenje za skrivanje veličine teksta može biti tako što se veličina sakrije u prvi ili predzadnji piksel slike, ali to ne mora biti uvijek slučaj. U tom slučaju, potrebno je usporediti svaku vrijednost boje s vrijednostima u ASCII tablici. Na kraju se dobivena slova prikazuju korisniku kao rješenje tajne poruke. Glavni nedostatak ove metode je stego slika koja može potaknuti sumnju ukoliko je unutar nje sakriveno više od 255 znakova. Drugim riječima, slika gubi na kontrastu, te su vidljive promjene u odnosu na originalnu sliku.

4.3 Otkrivanje slike iz slike

Nakon uspješnog skrivanja tajne slike unutar naslovne slike, sljedeći je korak otkrivanje stego slike i njenih sadržaja koje ona posjeduje. Za potrebe otkrivanja treba ponovno koristiti nešto matematike kako bi se takav problem riješio na brz i efikasan način. U ovom slučaju iskoristit ću primjer iz prethodnog zadatka. Prije rješavanja konkretnog problema, treba se podsjetiti da binarni zapis stego slike iznosi $01111101_{(2)}$. Iz navedenog binarnog zapisa, moguće je predvidjeti bitove koji predstavljaju naslovnu sliku i one bitove koji predstavljaju tajnu sliku. Napomena, svi primjeri slika se temelje na pikselima odnosno RGB bojama. Postupak se tada izvodi za sve tri boje: R,G i B. Kao primjer se navodi samo jedan slučaj.

Binarni zapis stego slike: $01111101_{(2)}$ = $\underbrace{0\,1\,1\,1}$ | $\underbrace{1\,1\,0\,1}$

Originalni bitovi Tajni bitovi

U primjeru skrivanja slike u slici, korištena je tehnika LSB metode s 4 bita. Primjer otkrivanja se temelji na istim načelima kao i u skrivanju slike. Problem otkrivanja slike rješavamo tako da se binarni zapis stego slike podijeli s onoliko bitova koliko je ugrađeno u tajni sadržaj. Stoga prvi korak obuhvaća dijeljenje binarnog zapisa sa 16 (zbog LSB 4 bita) i dobivanja bitova originalne slike: 01111101 / 16 = 0111 Nakon dijeljenja se množi sa 16, kako bi imali ponovno oktet: 0111 * 16 = 01110000 Na taj način, rezultatu dijeljenja se pridodaju 0, koje će nadopuniti bitove tajne slike koji su odbačeni tijekom dijeljenja.

Međutim, da bi se dobila slika slična originalu, potrebno je i dobiveni rezultat dijeljenja podijeliti sa 16, kako bi dobili ostatak okteta. Drugi korak se temelji na ostatku dijeljenja binarnog zapisa, te se u matematici koristi posebna funkcija pod nazivom „mod“ (oznaka %). Mod radi na istom principu kao i klasično dijeljenje, samo što se u ovom slučaju gleda isključivo ostatak dijeljenja. Dijeljenje binarnog zapisa s modom (%) da bi dobili ostatak: 01111101 % 16 = 1101. Ostatak dijeljenja se također množi sa 16, kako bi ponovno imali oktet, odnosno dobili tajni sadržaj slike.

Množenje dobivenog ostatka dijeljenja sa 16: 1101 * 16 = 11010000

Na kraju smo iz naslovne slike dobili dvije nove slike: originalnu sliku i tajnu sliku.

Tablica 5. Primjer rezultata otkrivanja tajne slike iz naslovne slike

Stego slika	Originalna slika	Tajna slika
		

Iz prikazane tablice se može zaključiti da je stego slika različita u odnosu na originalnu sliku. Razlika je u kontrastu, originalna slika ima nešto dublje boje u odnosu na stego sliku. Takav kontrast je dobiven dijeljenjem binarnog zapisa stego slike, te kasnijim množenjem sa 16. Možemo primijetiti da stego i originalna slika nisu pretjerano različite. Međutim, ako malo bolje promotrimo slike, na stego slici se mogu vidjeti obrisi tajne slike. Takav rezultat je postignut zbog toga što smo u ovom primjeru koristili LSB s 4 bita. Zaključak nas dovodi do razmišljanja da LSB metoda koja koristi 4 najmanje značajna bita i nije dobra u svim slučajevima. Drugim riječima, slika koja koristi LSB s 4 bita u nekim slučajevima će dati zadovoljavajuće rezultate, te se obrisi tajnog sadržaja uopće neće vidjeti. Svaki slučaj zavisi o vrsti slike i njezinom kontrastu. Ako je tajna slika dosta svjetlija u odnosu na naslovnu, tada će rezultat sadržavati obrise tajne slike koji će biti vidljivi ljudskom oku. Koristeći LSB metodu s manje značajnih bitova, postići ćemo precizniji rezultat, odnosno tajna slika se neće toliko vidjeti na stego slici. Najbolji slučaj je korištenje samo jednog bita za skrivanje slike. Rezultat će biti takav da se stego slika gotovo neće primijetiti u odnosu na originalnu sliku. U slučaju da koristimo slučaj LSB sa 7 bitova, tada bi dobili stego sliku na kojoj se odmah može uočiti da sadrži skriveni sadržaj u sebi. Danas postoje mnogo bolje, ali i daleko složenije metode od LSB metode koje su realizirane kroz algoritme kao što su: F5, JSTEG, OutGuess algoritam.

5. Primjena steganografije

5.1. Vodeni pečat

Vodeni žig ili pečat je jedna od legalnih primjena steganografije. Temelji se na dodavanju informacija u glavnu datoteku, bez velikih promjena nad izvornim skupom podataka. Najčešće se skriva potpis koji označava podrijetlo ili vlasništvo. Time se štite autorska prava. Mogućnost kopiranja i krađe svedena je samim time na minimum. Skrivena informacija ponekad može biti javna i vidljiva. Primjenom ove metode važno je naglasiti pozitivnu stranu: pečat se ne može neautorizirano ukloniti. Siguran je od mnogih signalnih modifikacija ili lažiranja pečata i mora se moći detektirati.

Postoje dvije vrste pečata: klasični i digitalni vodeni pečat.

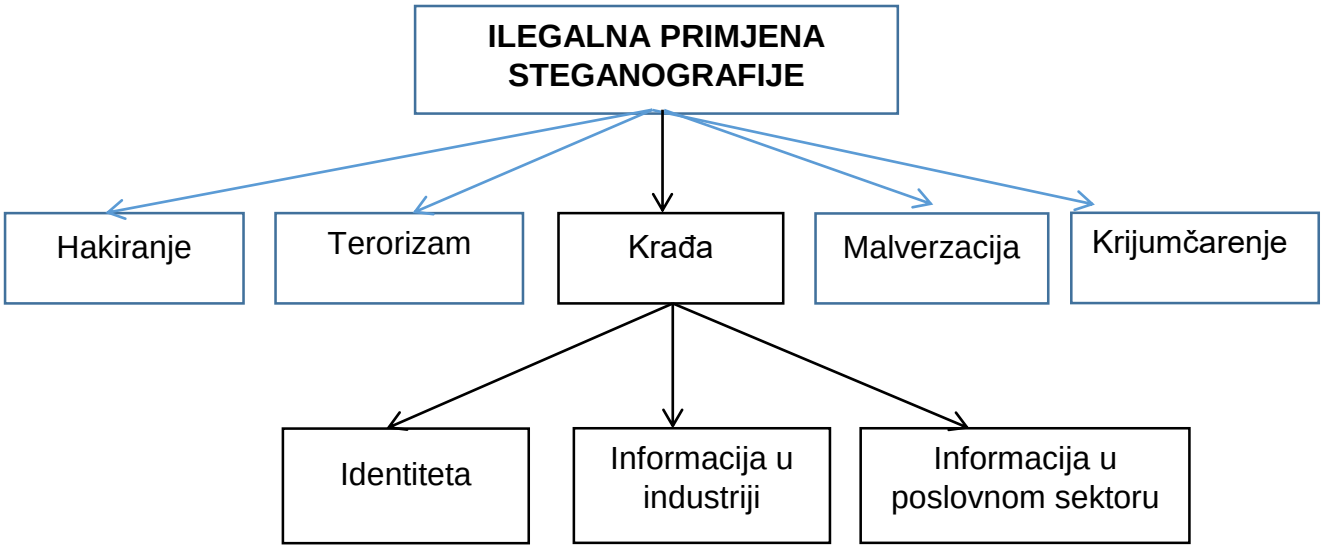
- Klasični vodeni pečat oznaka je utisnuta na papiru dizajnirana tako da se vidi samo pod određenim kutom gledanja. Koristi se u tiskanju novčanica, putovnicama, kao i u tiskanju poštanskih marki kako bi se tiskovine zaštitile od mogućeg krivotvorenja.
- Digitalni vodeni pečat je informacija koja se određenim metodama sakriva u originalni signal (multimedijalni zapis). Dodana informacija je nevidljiva, tj. signal dodavanjem vodenog pečata ne smije biti značajnije promijenjen. Digitalni vodeni pečat može sadržavati sve multimedijalne sadržaje: slike, glazbu i video, no najčešća im je primjena u zaštiti sadržaja fotografije.

5.2. Otisak prsta

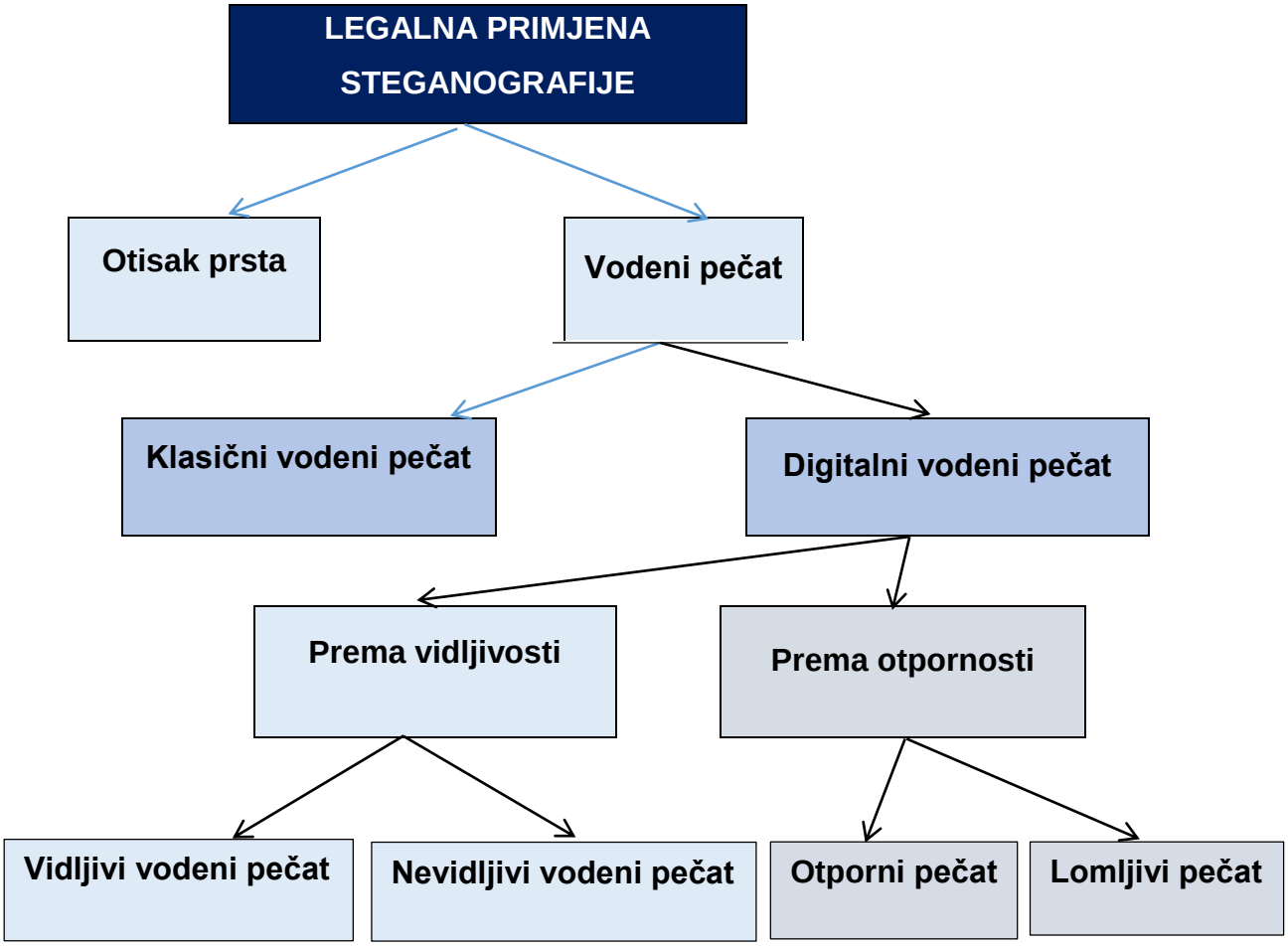
Otisak prsta ugrađuje se u različite dokumente koji se mogu dostavljati mnogim korisnicima. Time se štiti intelektualno vlasništvo, nemogućnost krađe ili davanja nekretnina trećim osobama. Ljudski otisci su jedinstveni, trajni tijekom života pojedinaca te su prikladni markerima ljudskog identiteta. Osim što se koriste u policiji za identificiranje potencijalnih subjekata, otisci prstiju pripadaju jednim od poznatijih tehnologija, smatraju se legitimnim dokazima na sudovima diljem svijeta. Sastoje se od četiri komponente: korisničko sučelje, baza podataka, modul za opis i modul za provjeru. Korisničko sučelje obuhvaća mehanizme za unos otisaka prstiju u sustav. Baza podataka sastoji se od zbirke zapisa s provedenim otiscima. Modul za upis vrši upis otisaka u bazu. Modul provjere provjerava podudaranje otisaka s traženim primjerom [15].

5.3 Primjena steganografije u praksi

Steganografija se koristi u različitim područjima. Primjena može biti legalna i ilegalna.



Slika 7. Pregled ilegalne primjene steganografije



Slika 8. Pregled legalne primjene steganografije

5.4 Primjena digitalnog vodenog pečata

Sve veći broj istraživanja o vodenom pečatu u proteklom desetljeću bio je često potaknut važnim aplikacijama koje se bave autorskim pravima i njihovom zaštitom. Neki od primjera uporabe digitalnog vodenog pečata su:

1. Vidljivi vodeni pečat: jednostavnom metodom zbrajanja dodaje se na originalnu sliku. Nedostaci vidljivih pečata su degradacija kvalitete slike.

a) Aplikacija za identifikaciju vlasnika: nekad je teško prepoznati vlasnika određenog digitalnog zapisa poput videozapisa ili slike. Umjesto da se uključi obavijest autorskog prava za svaku sliku ili neki drugi digitalni sadržaj, koristi se aplikacija temeljena na vodenom pečatu koji koristi tehniku ugrađivanja autorskog prava u sliku [16].

b) Koristi se za zaštitu autorskog prava. Primjer: fakultet izdaje knjige na kojima se nalazi njegov logo. Knjige se slobodno koriste za osobne potrebe, ali se ne mogu iskoristiti u komercijalne svrhe [15].

2. Nevidljivi vodeni pečat: sakriveni su unutar sadržaja slike, nevidljivi ljudskom oku, te ih je moguće detektirati programima s tom namjenom. Neki od primjera su:

a) Velike reklamne agencije su počele koristiti aplikacije kako bi osigurali da se njihove reklame izvode određeni broj puta na TV postajama. Informacije koje služe za identifikaciju pojedinačnih videozapisa, ugrađuju se u sami videozapis radi praćenja emitiranja[16].

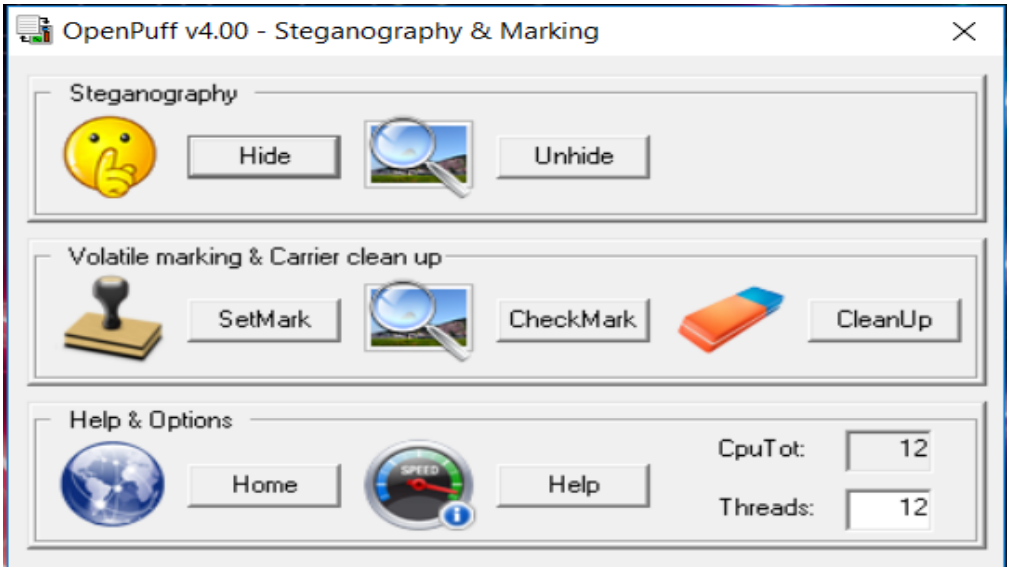
b) Aplikacija za praćenje transakcija. Primjer: koristi se za bilježenje primatelja svake zakonske kopije filma ili drugog medija ugrađivanjem vodenog pečata u svaku kopiju. U slučaju da je film bio postavljen na Internet, filmski producenti mogli su identificirati koji je primatelj filma bio izvor postavljanja medijskog sadržaja na Internet [16].

3. Lomljivi nevidljivi vodeni pečat:

a) Primjer za ovu vrstu pečata je baza podataka ljudskih otisaka prstiju u digitalnoj knjižnici. Prilikom snimanja u bazu na sliku se stavlja pečat. Svaka promjena otiska može se detektirati jer je pečat uništen i otisak nije valjan [17].

5.5 Primjena steganografije korištenjem programa OpenPuff

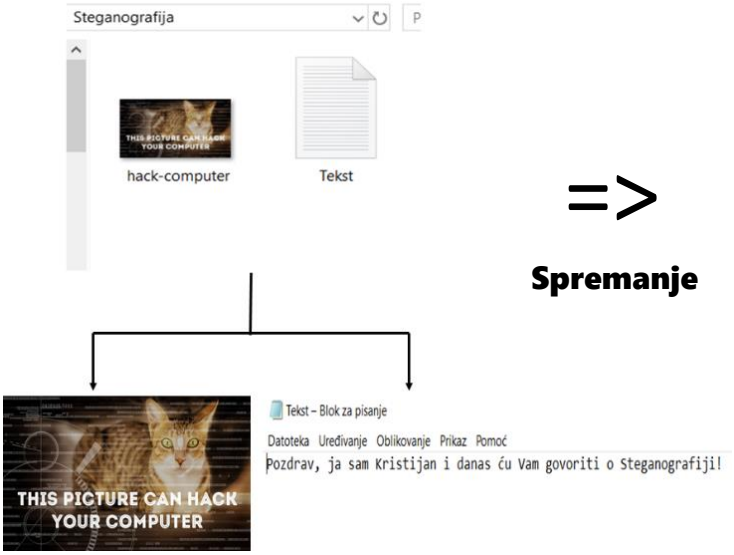
Kako bi najbolje prikazao korištenje steganografije s vodenim pečatom u praksi, koristiti ću besplatni softverski alat OpenPuff. OpenPuff se temelji na supstituciji bita najmanje važnosti te na taj način skriva podatke unutar slike ili nekog drugog multimedijskog sadržaja. Prilikom pokretanja programa učitava se izbornik unutar kojeg imamo nekoliko opcija: skrivanje poruka u slike, otkrivanje poruka iz slika, skrivanje određenih riječi pod sliku i ponovno otkrivanje istih. Sve navede opcije su vrlo učinkovite s gledišta steganografije.



Slika 9. Prikaz izbornika u programu OpenPuff

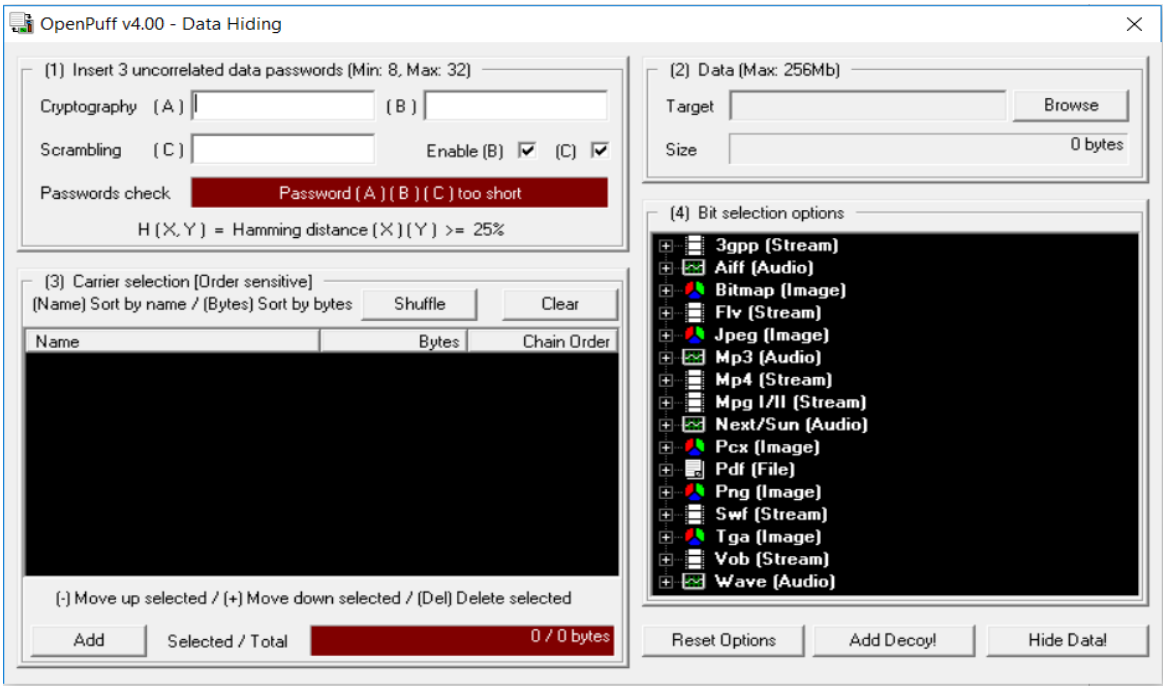
U ovom primjeru koristiti ću prvu opciju iz izbornika, a to je *spremanje poruke pod sliku*. Poruka u ovom slučaju ne mora nužno biti tekst, već može biti iz neke druge domene multimedija. Drugim riječima, cilj ovoga primjera je skrivanje dvaju sadržaja pod originalnu sliku. Radi se o skrivanju teksta i slike. Postupak je sljedeći: prvo odaberemo opciju *sakrij* (eng. Hide) iz izbornika. Nakon toga se otvara novi prozor koji će nam omogućiti unošenje tri jedinstvene lozinke. Lozinke su potrebne kako bi osigurali da sadržaj naše steganografske slike nitko ne bi mogao otkriti jer se one baziraju na kriptografiji. Sadržaj i broj znamenaka svake od lozinki mora biti različit. Ukoliko se nešto od toga približno podudara, program neće dozvoliti daljnje korištenje. Ukoliko unesemo ispravne lozinke, tada možemo pristupiti odabiru slike pod koju želimo spremiti sadržaj. Nakon odabira slike, biramo sadržaj koji želimo sakriti pod nju. Skrivam jedan tekst i sliku. Nakon toga, odabirem kategoriju za skrivanje sadržaja. Odabir kategorije je važan za kvalitetu naše buduće steganografske slike. Drugim

riječima, kategoriju odabiremo na način da pogledamo veličinu našeg sadržaja kojeg želimo sakriti. Ako sadržaj iznosi par MB tada se on može sakriti pod sliku. Međutim, u slučaju da je sadržaj izrazito velik, tada ga nema smisla skrivati pod sliku, jer bi onda ukupni iznos slike bio prevelik, što bi poticalo sumnju kod ljudi. Zbog toga se za velike sadržaje koristi video u MP4 formatu, kako bi ukupan iznos bio što realniji za određeni multimedijski sadržaj. Na raspolaganju imamo još opcije kao što su: MP4, Jpeg, Flv, Bitmap, 3gpp, Png, Pcx, Pdf itd.



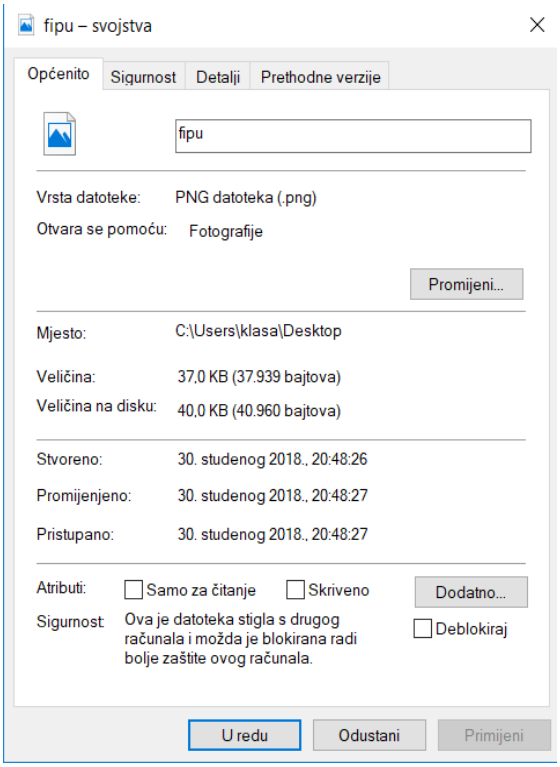
Slika 10. Podaci za skrivanje

Slika 11. Originalna slika

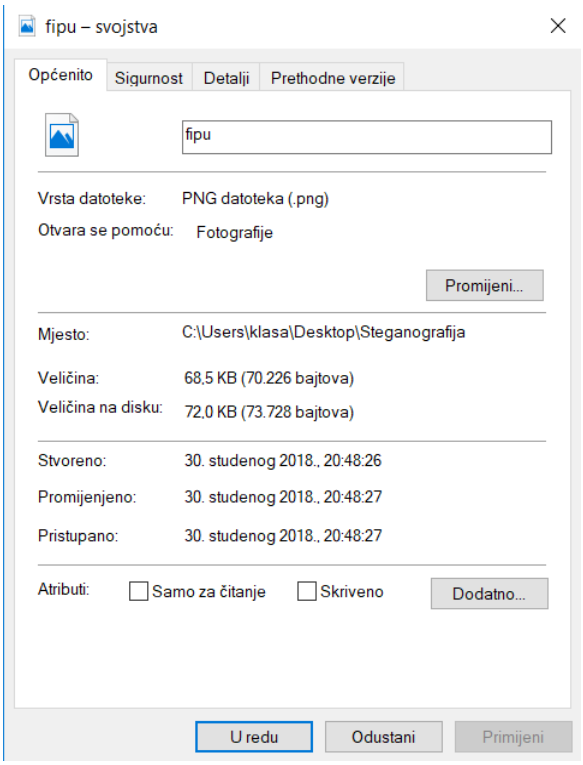


Slika 12. Prikaz sučelja za skrivanje podataka

U ovom slučaju sam koristio sliku formata Jpeg, što je znači da moj sadržaj za skrivanje iznosi svega par MB-a. Nakon odabira *formata*, na raspolaganju imam još opcije: *poništavanja* sve što sam odabrao (eng. Reset), dodavanje tajnog objekta i skrivanje podataka pod sliku. Nakon ispravnih dodavanja svih sadržaja, odabrao sam opciju *skrivanje podataka pod sliku* (eng. Hide Data). Odabirom navedene opcije, aplikacija vraća povratnu informaciju o uspješnosti stegiranja podataka, odnosno o uspješnosti skrivanja pod sliku. Nakon skrivanja podataka, slijedi provjera veličine stego slike i originalne slike. Možemo vidjeti da veličina originalne slike iznosi 37 KB, a stego slike 68,5 KB. Iako je u ovome primjeru stego slika veća od originalne slike, većina stručnjaka nikada nema obje slike u posjedu, što znači da nemaju mogućnost uspoređivanja veličina slika. Možemo zaključiti da se u ovome primjeru teško može razaznati da li se radi o stego slici jer je upravo njena veličina u okvirima normalnih veličina slika, te je steganografija prihvatljiva.



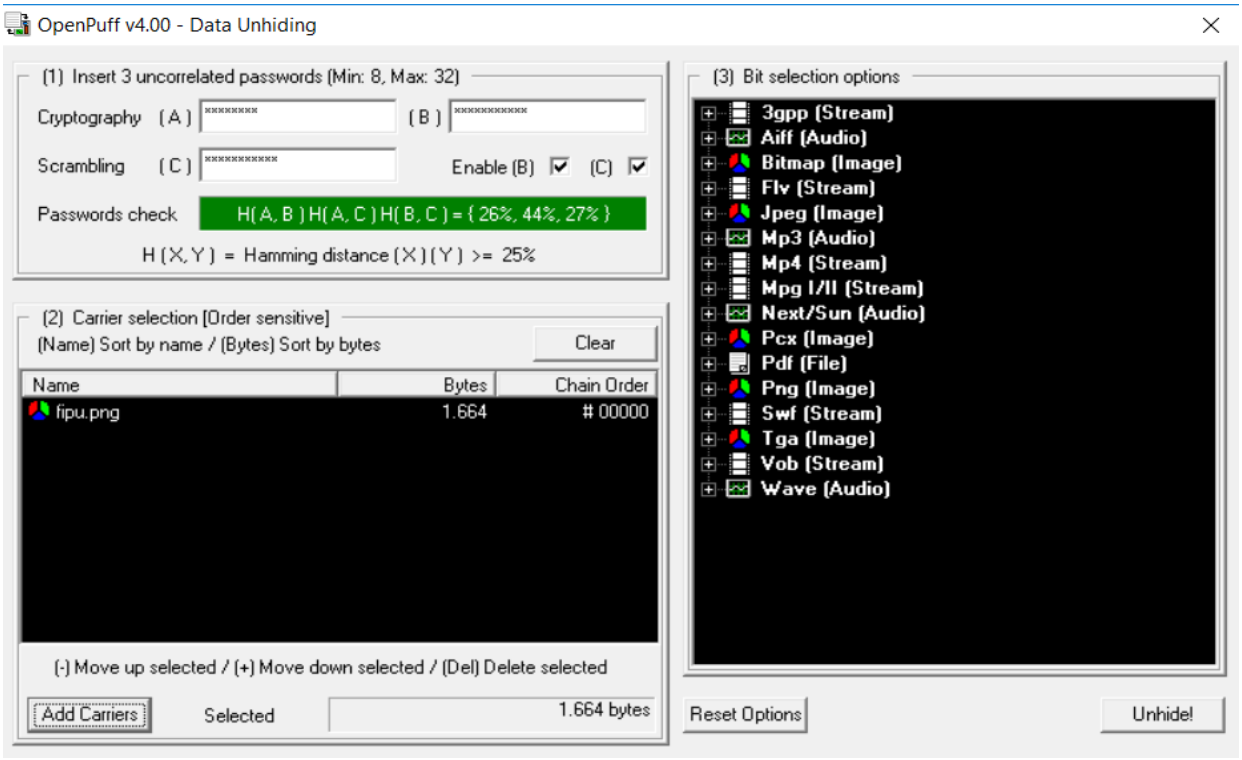
Slika 13. Prikaz veličine originalne slike



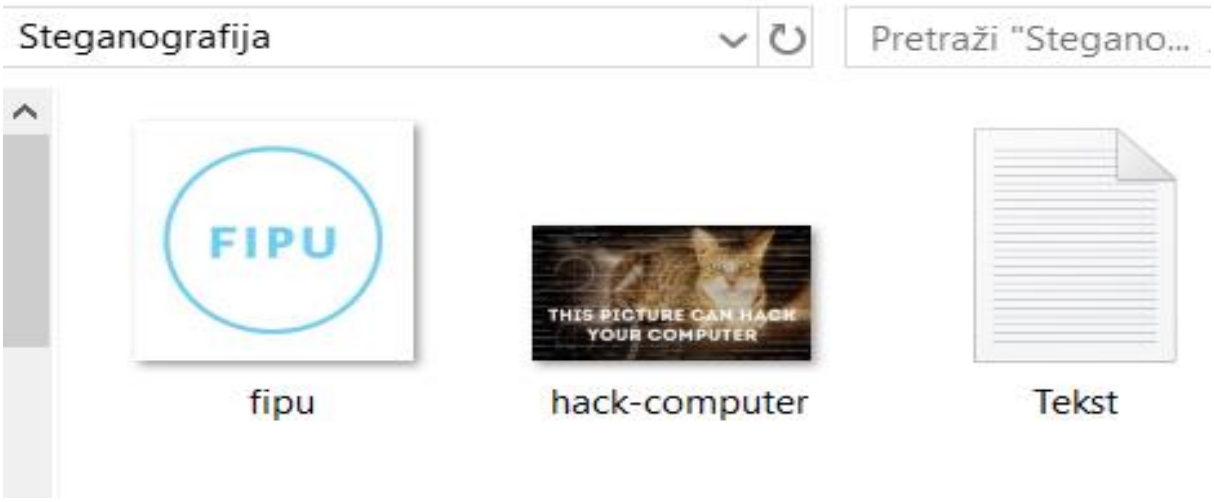
Slika 14. Prikaz veličine stego slike

Dodatno ću objasnit postupak otkrivanja podataka iz stego slike. Koristiti ću primjer iz prethodnog slučaja. Prvi korak je ponovno pokretanje programa i odabir opcije *otkrij* (eng. Unhide) iz izbornika. Odabirom opcije otkrij, otvara se isti prozor kao i u prošlom slučaju. Prvo odaberemo sliku iz koje želimo izdvojiti sadržaj. Sljedeći korak je ponovno unošenje iste lozinke koje smo u prethodnom slučaju unijeli za tu sliku. Ako

su lozinke točne, možemo izvršiti postupak izdvajanja sadržaja iz slike. Program za izdvajanje sadržaja u ovom slučaju daje dvije mogućnosti, a to su: naredba za *resetiranje* koja služi za poništavanje slike i lozinki, te opcija *otkrij* za otkrivanje sadržaja iz slike. Odabrat ću opciju otkrij. Program mi nakon izvođenja operacije vraća povratnu informaciju o uspješno izdvojenome sadržaju iz slika. Potom otvara mapu sa svim izdvojenim sadržajima, a to su: tekst, slika te originalna slika pod kojom sam skrivao podatke.



Slika 15. Prikaz postupka otkrivanja podataka iz slike



Slika 16. Prikaz sadržaja iz stego slike

6. Implementacija programa StegoProgram izrađenog u C#

U ovom dijelu objasniti ću rad svoje aplikacije StegoProgram i opisati programsko rješenje koje se bazira na osnovama steganografije. Na samom početku potrebno je objasniti na koji način su prozori povezani, te njihovu vezu s kodom. Također, navesti ću i pojasniti sve opcije koje program posjeduje kao i njihovu implementaciju u C#.

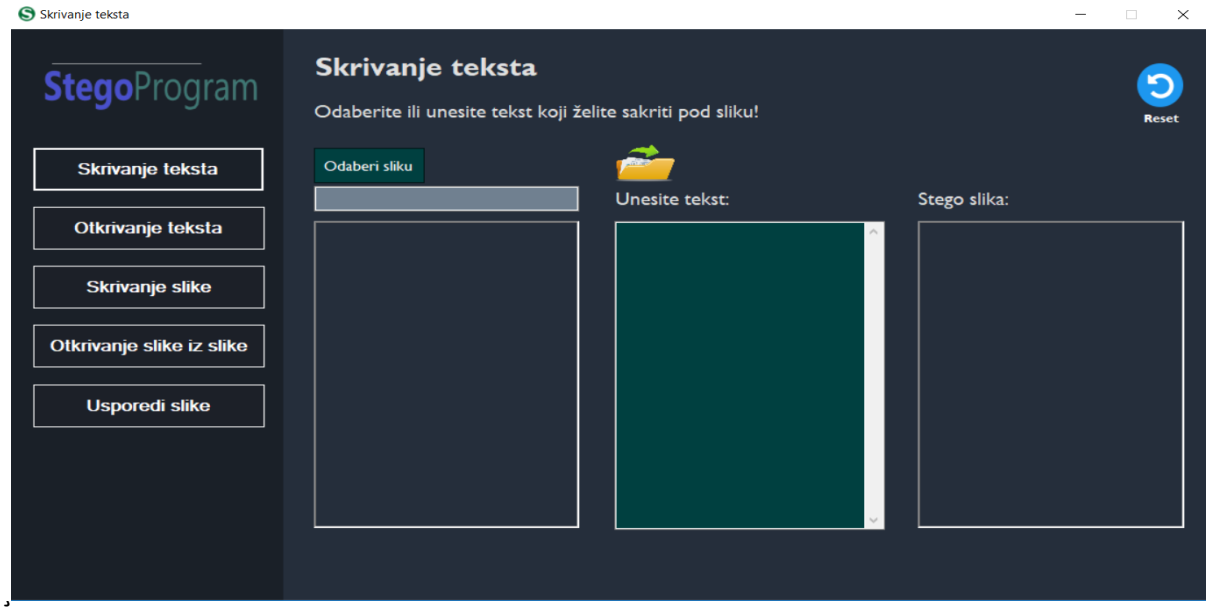
6.1 Općenito o programskom jeziku C#

Ovaj projekt je rađen u programskom jeziku C# za kreiranje aplikacije koja radi na sustavu Windows. C# je objektno orijentirani programski jezik koji je razvijen od strane Microsofta. Možemo reći da je C# idealan za programere početnike jer većinu stvari nasljeđuje iz drugih programskih jezika. Također, podržava ispravljanje pogrešaka na izvornoj i strojnoj razini [18]. Ovaj programski jezik je poznat po svojoj jednostavnosti i fleksibilnosti, najviše je napravljen kako bi .NET platforma dobila programski jezik. Visual C# jedna je od ponuđenih komponenti u Visual Studiu, a radi na principu Visual Basica, te je vrlo slična Javi i C++. Visual Studio .NET je integrirano razvojno okruženje koje se koristi za razvoj konzola, grafičkih korisničkih sučelja, web aplikacija i web usluga. Unutar sebe sadrži objekte koji predstavljaju strukture koje imaju metode, podatkovne elemente i ostale interakcije. Kod aplikacija predviđenih za Windows sustav, kreiramo obrasce koji koriste sučelje (eng. Windows forms). Kreiranje se temelji na dovlačenju elemenata koji se nalaze u grupi *alati* (eng. Toolbox).

Korisničko sučelje gradimo tako što povlačimo dizajnerske elemenate na prozor aplikacije. Nakon odabira konkretnih elemenata, potrebno ih je povezati s određenim funkcijama. Dvostrukim klikom na odabrani element dizajnerskog alata, otvara se površina za unos programskog koda kojega će taj element izvršavati. Program koji se piše u programu C# se sprema s ekstenzijom .cs, a pokreće se naredbom *otklanjanja neispravnosti* (eng. Debug). Aplikacija StegoProgram se sastoji od prozora koji se nasumično otvaraju i zatvaraju. Svi prozori su naslagani tako da se mogu pozvati u bilo kojem trenutku. Kada korisnik pritisne neki gumb, postojeći prozor se zatvara, te se otvara novi prozor sa sadržajem koji se nalazi na njemu. Otvaranje i zatvaranje prozora je realizirano pomoću naredbi *this.Hide* i *form.Show*. Konkretno this.Hide se koristi za skrivanje postojećeg prozora. Kako bi mogli otvoriti novi prozor, potrebna je instanca klase prozora kojeg želimo otvoriti. Svaka klasa sadrži konstruktor koji služi za učitavanje tog dijela aplikacije.

6.2 Skrivanje teksta unutar slike

Prilikom pokretanja programa StegoProgram, na raspolaganju imamo nekoliko opcija: skrivanje i otkrivanje teksta, skrivanje i otkrivanje tajnog sadržaja slike te usporedba slika. Prva opcija programa se odnosi na skrivanje teksta unutar slike. Odabirom navedene opcije, otvara se novi prozor koji će omogućiti unos tajne poruke i odabir naslovne slike unutar koje možemo sakriti napisanu poruku.

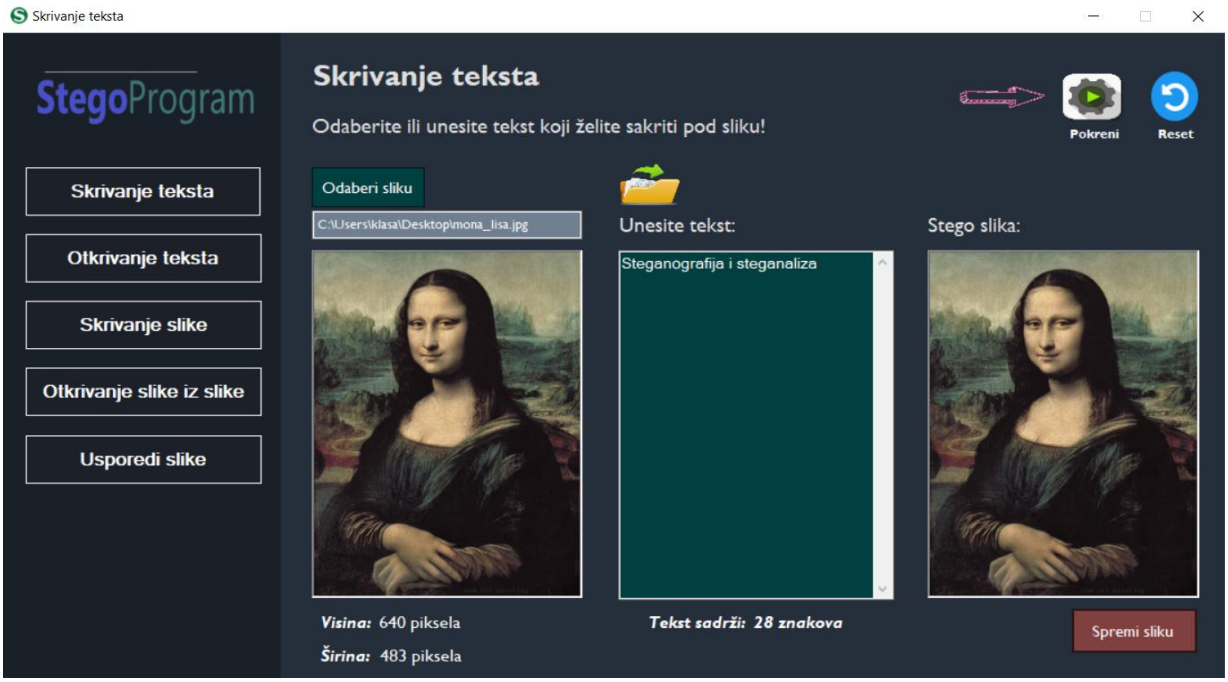


Slika 17. Korištenje opcije „Skrivanje teksta“ u programu StegoProgram

Svi prikazani elementi imaju svoju ulogu u programu, pa su tako na slici prikazani: *gumbi* (eng. Buttons), *okviri za slike* (eng. PictureBox), *prostor za unos tajne poruke* i *puta slike* (eng. TextBox). Pritiskom na gumb *odaberi sliku*, otvara se novi prozor koji će omogućiti odabir slike koju želimo koristiti kao naslovnu sliku. Prilikom odabira slike, možemo koristiti najpoznatije formate za slike .jpg ili .png. Odabirom slike, ista slika se učitava u prostor za sliku, a put do slike se upisuje u poseban tekstualni okvir kako bi korisnik znao u svakom trenutku put slike. Međutim, u programu postoje i skrivene operacije koje se izvode tek kada se izvrši određena faza operacije. U ovom slučaju, odabirom slike, u prozoru se prikazuju dimenzije istoimene slike izražene u pikselima. Na zaslonu će biti vidljiv gumb koji omogućava pokretanje procesa skrivanja teksta unutar slike. Korisnik unosi tajnu poruku koju želi sakriti pod sliku ili može odabrati tekstualnu datoteku s računala. Klikom na gumb *pokreni*, otvaraju se dodatni prozori u obliku dijaloga koji provjeravaju želi li korisnik zaista pokrenuti proces. Ukoliko korisnik odgovori s potvrdnim odgovorom, prima obavijest da će proces potrajati, te se pokreće

proces skrivanja teksta pod naslovnu sliku. Ukoliko korisnik ne želi pokrenuti proces, omogućena mu je naknadna izmjena sadržaja. Na kraju, korisnik može spremiti dobivenu stego sliku klikom na gumb *spremi sliku*. Međutim, ako korisnik želi obrisati sav sadržaj koji je prethodno odabrao, to može jednostavno napraviti klikom na gumb *resetiraj* (eng. Reset) označenog plavom strelicom.

Primjer: u ovom dijelu sam koristio sliku Mona Lise kao naslovnu sliku, .jpg formata slike. Tekst kojeg skrivam je naziv mog završnog rada: „Steganografija i steganaliza“. Pokretanjem procesa, dobivam povratnu informaciju da je slika veličine 640 x 483 piksela i da tekst iznosi 28 znakova. Nakon svršetka procesa, dobiveni rezultat je predočen sljedećom slikom:



Slika 18. Rezultat pokretanja procesa za skrivanje teksta unutar slike

Najvažniji dio ovog procesa je uspoređivanje originalne slike sa stego slikom. Drugim riječima, tražimo razlike između te dvije slike i da li su one vidljive ljudskim okom. Opažanje ljudskim okom još uvijek nije zabilježeno u steganografiji, upravo zbog toga što tekst sadrži vrlo malo prostora pa ne ostavlja duboke tragove na slikama koje su tada izražene dubljim kontrastom. Korištenjem programa koji se temelji na steganalizi, moguće je otkriti njihovu različitost ili pomoću navedenog programa koji sadrži opciju usporedi slike, a biti će opisana nešto kasnije. Skrivanje teksta unutar slike je najčešće postignuta s LSB metodom zamijene najmanje važnog bita, ali postoje i druge opcije kojima se može postići isti efekt.

Na slici 19. prikazana je slika Mona Lise pod kojom skrivam poruku veličine 28 znakova. Slika je JPG formata dimenzija 483 x 640. Na slici 20. je prikazana stego slika koja je rezultat skrivanja poruke. Promjene na fotografijama nije moguće uočiti.

Slika 19. Originalna slika Mona Lise

Slika 20. Stego slika Mona Lise

Na slici 21. prikazana je slika Mjeseca koja služi za skrivanje tajne poruke. Slika je JPG formata dimenzija 259 x 194. Unutar originalne slike je skriveno 10396 znakova. Na slici 22. je prikazana stego slika koja predstavlja rezultat skrivanja tajne poruke. Promjene na fotografijama ponovno nije moguće uočiti okom.

Slika 21. Originalna slika Mjeseca

Slika 22. Stego slika Mjeseca

Nakon prikazanih primjera skrivanja teksta, moguće je zaključiti kako ova metoda uspješno skriva tekst unutar slike. Za otkrivanje teksta se koristi grana steganalize.

6.3 Implementacija skrivanja teksta unutar slike

Prvi dio programskog rješenja se odnosi na izradu korisničkog sučelja. Svi elementi se dodaju iz *alata* (eng.Toolbox) jednostavnim povlačenjem na zaslon aplikacije (eng. Drag & Drop). Nakon postavljanja elemenata, slijedi dio programa preko kojega možemo odabrati naslovnu sliku. Ukoliko se klikne na gumb *odaberi sliku*, program će omogućiti odabir naslovne slike. Otvaranjem *dijaloškog okvira* (eng. OpenFIleDialog) moguć je pregled i odabir slike određene mape koje posjedujemo na računalu. Pomoću filtera *formata* definiram format slike koji se može odabrati za naslovnu sliku. Odabirom *slike*, ispunjava se uvjet koji daje prikaz slike te se put slike upisuje u tekstualni okvir. Na kraju se izvodi metoda *provjeri_velicinu()* koja ispisuje dimenzije odabrane slike i postavlja vidljivost skrivenih gumbiju na *true*.

```
private void btnOdaberi_Click(object sender, EventArgs e)
      {
         OpenFileDialog dialog = new OpenFileDialog();
         dialog.Filter = "png files(*.png)|*.png|jpg. files(*.jpg)|*.jpg|All files(*.*)|*.*";
         if (dialog.ShowDialog() == DialogResult.OK)
         {
            lokacija_slike = dialog.FileName.ToString();
            textBox1.Text = lokacija_slike;
            pictureBox1.ImageLocation = lokacija_slike;
         }
         provjeri_velicinu();
      }
```

Slika 23. Prikaz programskog koda za pregledavanje i odabir slike

Proporcionalno otvaranju slike, programski kod za spremanje je sličan programu za otvaranje. Razlikuju se u naredbama za spremanje slike tj. koristi se *dijaloški okvir za spremanje slike* (eng. SaveFileDialog) te se iz bitmape uzima slika koja se sprema.

```
private void btnSpremi_Click(object sender, EventArgs e)
      {
         SaveFileDialog spremi = new SaveFileDialog();
         spremi.Filter = "png files(*.png)|*.png|jpg files(*.jpg)|*.jpg|All files(*.*)|*.*";
         Bitmap slika2 = new Bitmap(pictureBox2.Image);
         if (spremi.ShowDialog() == DialogResult.OK)
         {
            lokacija_slike2 = spremi.FileName.ToString();

            slika2.Save(lokacija_slike2);
         }
      }
```

Slika 24. Prikaz programskog koda za spremanje slike

Bitmape se definiraju kao mreža ćelija zvanih pikseli,a svaki piksel sadrži vrijednost boje. Karakteriziraju ih broj piksela i dubina boje po pikselu. U metodi *sakrij_tekst()* prvo se kreira bitmapa i u nju se dodaje odabrana slika. U varijablu *velicina_teksta* pohranjuje se veličina znakova unesenog teksta. Veličina teksta je potrebna kako bi program znao koliko puta treba izvoditi postupak skrivanja teksta. Glavni dio programa se odnosi na ugnježđenu for petlju koja se kreće po pikselima slike, a izvodi se do kraja širine i visine slike. Zatim se dohvaćaju vrijednosti svakog piksela iz bitmape. Kako bi program mogao ispravno raditi, potrebno je podijeliti glavni dio programa na dva dijela. Prvi dio se odnosi na ugradnju veličine teksta u predzadnji bit, da bi program kod otkrivanja teksta mogao ispravno pretvoriti točan broj znakova u tekst. Ugradnja veličine teksta je postignuta tako što se kod svake boje predzadnjeg bita ugrađuju bitovi veličine teksta. Kako bi se moglo sakriti nekoliko tisuća znakova unutar slike, potrebno je koristiti 16-bitni zapis (2 bajta). Međutim, da bi boje predzadnjeg piksela donekle bile iste onim originalnim, potrebno je 16 bitova preraspodijeliti tako da crvena i zelena boja primaju po 6 bitova, a plava boja 4 bita iz broja veličine teksta. Postupak je takav da se uvode 3 varijable koje spremaju vrijednosti razlike originalne vrijednosti i binarnog zapisa iz kojega se izdvajaju 6 najmanje važnih bitova, odnosno u slučaju plave boje 4 bita. U varijablu *velicina_teksta1* se sprema prvih 6 bitova iz *velicina_teksta*, a to je postignuto tako što se veličina teksta podijeli s 1024 jer je potrebno izdvojiti prvih 6 bitova iz veličine teksta. Postupak se dalje ponavlja samo što se u drugom slučaju uzima ostatak dijeljenja od 1024 kako bi se uzeo drugi set od 6 bitova. Na kraju se uzima ostatak dijeljenja sa 16 kako bi se dohvatila zadnja 4 bita. Nakon postupka izdvajanja bitova iz veličine teksta, potrebno je svaku vrijednost boje zbrojiti s izdvojenim bitovima i ugraditi u bitove. Na ovaj način su svi bitovi veličine teksta ugrađeni u predzadnji bit slike te prilikom otkrivanja teksta program jednostavno može pročitati maksimalni broj znakova i otkriti sakrivene znakove. Nakon ugradnje veličine teksta potrebno je izdvojiti zadnji bit svake RGB boje zadanog piksela. Svaki piksel sadrži 3 boje i pri tome možemo kod jednog piksela izmijeniti 3 zadnja bita kod svake boje. Jedan znak koji se ugrađuje ima 8 bitova. Prema ovoj teoriji, potrebna su 3 piksela kako bi se ugradio jedan znak. Zbog toga sam u programu uveo brojač koji broji piksele u rasponu od 0 do 3. Ukoliko brojač sadrži vrijednost 0, tada se izvodi blok koji dohvaća te pretvara vrijednost u znak prema ASCII tablici. Kada brojač izbroji do 3, tada se brojač ponovno postavlja na 0 i uzima novo slovo koje je potrebno ugraditi. Varijabla *brojac2* služi za dohvaćanje novog slova, sve dok ne odbroje sva slova koje

korisnik želi sakriti. Drugi dio se odnosi na skrivanje teksta unutar slike sve do posljednjeg znaka. Nakon dobivenog znaka, potrebno je izdvojiti iz svakog piksela vrijednost najmanje značajnog bita. Zbog toga je potrebno kod svake RGB boje podijeliti njihovu vrijednost s 2 i sačuvati ostatak dijeljenja. Dobiveni ostatak oduzimam od vrijednosti RGB boje kako bi dobio vrijednost nove boje Primjer: ugradnja slova „a“ pod sliku, te promatranje prvog piksela crvene RGB boje.

1) $135 = 10000111_{(2)} \Rightarrow 135 \% 2 = 1000011\mathbf{0}_{(2)}$ ⟶ 0 je ostatak (zadnji bit)

2) Novi pikselR = $135 - 1 = 10000111_{(2)} - 00000001_{(2)} = 134 \Rightarrow 10000110_{(2)}$

```
public void sakrij_tekst() {

    Bitmap slika = new Bitmap(lokacija_slike);

    if (upitnik == DialogResult.OK)
    {
      var velicina_teksta = textBox2.TextLength;
      for (int i = 0; i < slika.Width; i++)
      {
        for (int j = 0; j < slika.Height; j++)
        {
          Color pikseli = slika.GetPixel(i, j);
          if (i == slika.Width - 1 && j == slika.Height - 1)
          {
              var pikselR = pikseli.R - pikseli.R % 64;
              var pikselG = pikseli.G - pikseli.G % 64;
              var pikselB = pikseli.B - pikseli.B % 16;

              var velicina_teksta1 = velicina_teksta / 1024;
              var velicina = velicina_teksta % 1024;
              var velicina_teksta2 = velicina / 16;
              var velicina_teksta3 = velicina % 16;

              pikselR = pikselR + velicina_teksta1;
              pikselG = pikselG + velicina_teksta2;
              pikselB = pikselB + velicina_teksta3;

              slika.SetPixel(i, j, Color.FromArgb(pikselR, pikselG, pikselB));
          }
          try
          {
              if (brojac == 0)
              {
                  if (brojac2 < velicina_teksta)
                  {
                      slova = Convert.ToChar(textBox2.Text.Substring(brojac2, 1));
                      vrij = Convert.ToInt32(slova);
                  }
              }
              if (brojac2 < velicina_teksta)
              {
                  int pikselR = pikseli.R - pikseli.R % 2;
                  int pikselG = pikseli.G - pikseli.G % 2;
                  int pikselB = pikseli.B - pikseli.B % 2;
                  brojac++;
```

Slika 25. Programski kod koji skriva veličinu teksta u predzadnji piksel

Grananje služi kako bi se raspodijelio posao ugradnje bitova unutar piksela. Prvi slučaj služi kako bi dohvatili prva 3 bita najvažnije vrijednosti (tzv. MSB bitovi). Drugim riječima, postupak izdvajanja bitova iz okteta se izvodi slijedno od najvažnijih bitova prema onim najmanje važnima. Za dobivanje prvog MSB-a, potrebno je podijeliti vrijednost s 128 te zbrojiti vrijednosti crvene boje koja je dobivena oduzimanjem. Drugi bit se dobiva dijeljenjem sa 64, a treći s 32. Nakon dijeljenja se uvijek uzima ostatak pri dijeljenju s 2 i zbraja s vrijednosti odgovarajuće boje. Nakon zbrajanja, nove vrijednosti boja se postavljaju u bitmapu (eng. SetPixel). Isti princip se odvija u drugom i trećem slučaju. U drugom slučaju se dohvaća vrijednost 4. bita tako što se podijeli sa 16 i uzme ostatak nakon novog dijeljenja s 2. Postupak dohvaćanja 3. bita se postiže tako da podijelimo s 8 i ponovno uzmemo ostatak dijeljenja s 2. Također vrijednost 2. bita se dobiva tako da se podijeli s 4 i uzme ostatak dijeljenja. U trećem slučaju potrebno je izdvojiti posljednja dva bita. Prvi bit se dobiva dijeljenjem s 2 i njegovim ostatkom. Zadnji bit se uzima tako što se koristi samo ostatak dijeljenja s 2. Također, u trećem slučaju se varijabla brojac postavlja ponovno na 0, što znači da program u sljedećem koraku uzima novo slovo koje se skriva pod sliku. *Varijabla brojac2* se uvećava za 1, te se na taj način dohvaća novo slovo. U slučaju da se brojač ne uveća za 1, program bi ponovno uzeo u obzir isto slovo koje je već sakrio pod sliku. Nakon izvođenja svih slučajeva, jedan ciklus skrivanja jednog slova je uspješno završen te je slovo sakriveno unutar slike. Za skrivanje preostalih slova, postupak je identičan. Ukoliko slika nije dovoljno velika za količinu skrivanja teksta, javiti će se iznimka koja će korisniku dati do znanja da odabere veću sliku.

Tablica 6. Prikaz bitova raspoređenih po bojama i pripadnosti switch grananju

1	0	0	0	0	1	1	0
Crvena	Zelena	Plava	Crvena	Zelena	Plava	Crvena	Zelena
Case 1	Case 1	Case 1	Case2	Case 2	Case 2	Case 3	Case 3

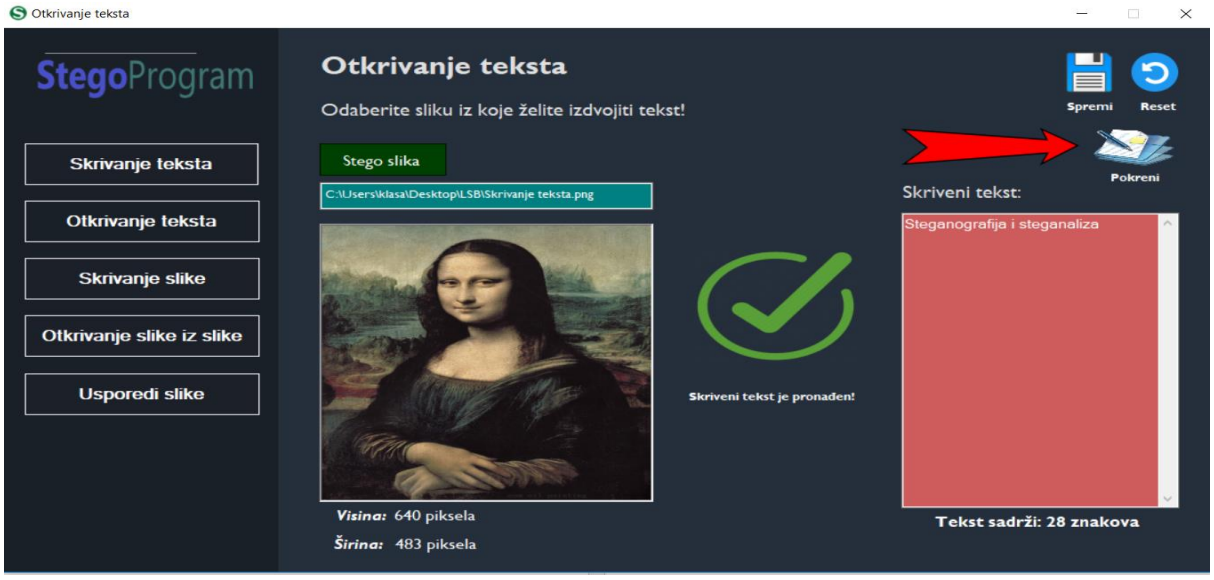
Iz tablice se može vidjeti koji se bitovi ugrađuju pod koje vrijednosti boja. U prvom primjeru, bit 1 je prvi koji se ugrađuje pod crvenu boju prvog piksela. Zatim se ugrađuje bit 0 na zadnji bit zelene boje prvog piksela. Postupak se izvodi slijedno sve dok se svi bitovi ne sakriju pod vrijednosti boja. Zamjena zadnjeg bita (LSB metoda) kod RGB boja određenog piksela je najbolja opcija skrivanja teksta pod sliku jer se razlika između originalne i stego slike ne može vidjeti ljudskim okom.

```
                switch (brojac){
                    case 1:
                    {
                        int r = vrij / 128;
                        pikselR = pikselR + r;

                        int g1 = vrij / 64;
                        int g = g1 % 2;
                        pikselG = pikselG + g;

                       int b1 = vrij / 32;
                       int b = b1 % 2;
                       pikselB = pikselB + b;

                        slika.SetPixel(i, j, Color.FromArgb(pikselR, pikselG, pikselB));
                  }
                    break;
                   case 2:
                   {
                       int r2 = vrij / 16;
                       int r = r2 % 2;
                       pikselR = pikselR + r;

                      int g2 = vrij / 8;
                      int g = g2 % 2;
                      pikselG = pikselG + g;

                     int b2 = vrij / 4;
                     int b = b2 % 2;
                     pikselB = pikselB + b;

                      slika.SetPixel(i, j, Color.FromArgb(pikselR, pikselG, pikselB));
                }
                 break;
                 case 3:
                {
                    int r3 = vrij / 2;
                    int r = r3 % 2;
                    pikselR = pikselR + r;

                    int g3 = vrij % 2 ;
                    pikselG = pikselG + g3;

                    slika.SetPixel(i, j, Color.FromArgb(pikselR, pikselG, pikselB));

                    brojac = 0;
                    brojac2++;
                }
                  break;
              }
              label9.Visible = true;
              label10.Visible = true;
          }
      }
      catch (Exception e)
      {
          string poruka = "Odaberite veću sliku za unos teksta!";
          string naslov = "Greška!";
          MessageBox.Show(poruka, naslov, MessageBoxButtons.OK, MessageBoxIcon.Error);
      }
    }
  }
  pictureBox2.Image= slika;
}
```

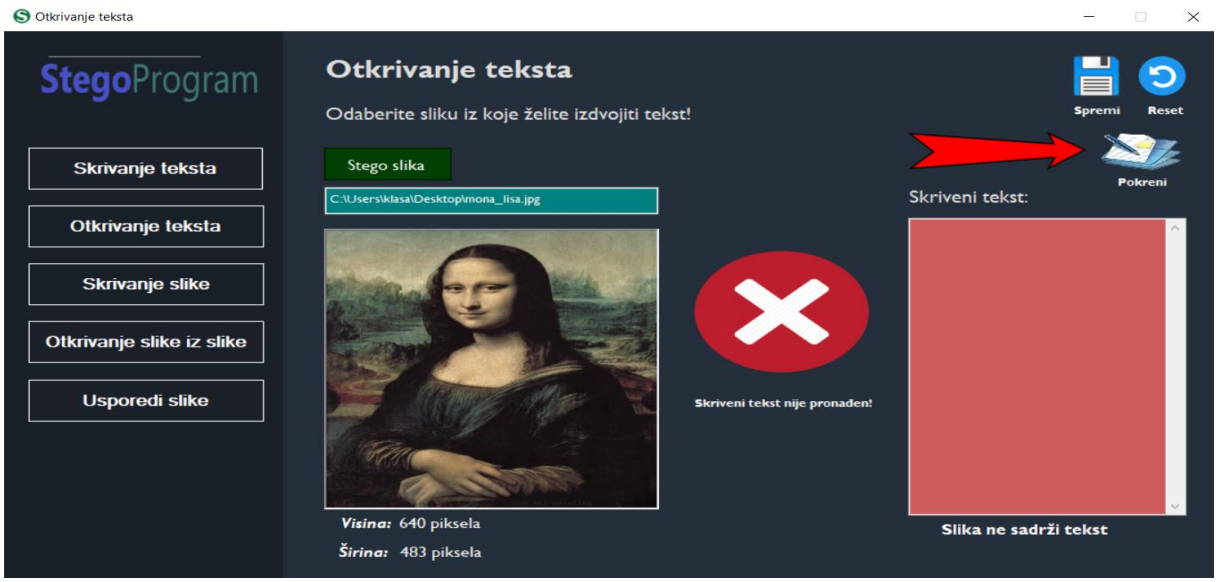
Slika 26. Programski kod koji skriva slova koristeći LSB metodu

6.4 Otkrivanje skrivenog teksta iz slike

Prvi korak u otkrivanju teksta je odabir slike za koju smatramo da sadrži tajnu poruku. Klikom na gumb *odaberi sliku*, otvara se prozor za pregled na kojem odabiremo stego sliku koju smo maloprije napravili. Odabirom slike, izračunavaju se dimenzije slike te se pojavljuje gumb za pokretanje procesa otkrivanja. Pritiskom na gumb *otkrij tekst*, otvara se dijaloški okvir koji nas pita želimo li pokrenuti proces otkrivanja. Potvrdnim odgovorom pokreće se proces otkrivanja sadržaja iz slike. Ukoliko slika sadrži tajnu poruku, tada će na zaslonu pisati da je skriveni tekst uspješno pronađen, te će se tekst pojaviti u tekstualnom okviru. Ukoliko želi, korisnik može spremiti dobiveni tekst na računalo. U drugom primjeru je prikazana slika koja ne sadrži tajnu poruku.



Slika 27. Prikaz uspješnog rezultata programa za otkrivanje teksta iz slike



Slika 28. Prikaz neuspješnog rezultata programa za otkrivanje teksta iz slike

6.5 Implementacija otkrivanja teksta iz slike

Prvi korak je kreiranje bitmape u koju se postavlja slika. Zatim se ponovno dohvaća vrijednost RGB boja predzadnjeg piksela slike unutar kojeg je ugrađena veličina teksta. Takvi pikseli su potrebni kako bi program znao koliko znakova treba otkriti u programu. Kako bi dobili veličinu teksta odnosno zadnjih 6 bitova, potrebno je podijeliti i uzeti ostatak dijeljenja sa 64 kod vrijednost crvene i zelene boje. Nadalje, potrebno je podijeliti vrijednost plave boje kako bi se dobilo zadnja 4 bita. Na ovaj način, dobivamo svih 16 bitova veličine teksta koji su skriveni u zadnjem pikselu. Kako bi mogli objediniti svih 16 bitova, potrebno je koristiti klasu *niza bitova* (eng. BitArray). Naredba upravlja nizom vrijednosti bita koje su predstavljene kao bool i koristi se kada pohranjujemo bitove. Kako bi se bitovi unijeli po redoslijedu, upotrijebio sam for petlju koja se sastoji od uvjeta. Prvi uvjet služi za unos prva 4 bita koja su dobivena prilikom otkrivanja bitova iz plave boje. Naredba za unos bitova zahtjeva indeks pohrane bita i logičke vrijednost bita. Drugi i treći uvjeti se koriste kada pohranjujemo 6 bitova iz zelene odnosno crvene boje. Na kraju je potrebno proslijediti vrijednosti bitova iz niza u metodu PretvoriInt koja vraća decimalni broj odnosno veličinu teksta.

```
public void otkrij_tekst()
{
      Bitmap otkrij = new Bitmap(textBox1.Text);
      Color zadnji = otkrij.GetPixel(otkrij.Width-1, otkrij.Height-1);
      BitArray bit = new BitArray(16);
       int vrij1 = zadnji.R % 64;
       int vrij2 = zadnji.G % 64;
       int vrij3 = zadnji.B % 16;
       int ostatak,ostatak2,ostatak3=0;

          for(int bb=0; bb<bit.Count; bb++)
          {
             if (bb < 4){
                ostatak = vrij3 % 2;
                var ostatak1 = Convert.ToBoolean(ostatak);
                vrij3 /= 2;
                bit.Set(bb, ostatak1);  }

             if (bb>3 && bb < 10){
                ostatak2 = vrij2 % 2;
                var ostatak22 = Convert.ToBoolean(ostatak2);
                vrij2 /= 2;
                bit.Set(bb, ostatak22); }

             if (bb>10 && bb < 16){
                ostatak3 = vrij1 % 2;
                var ostatak33 = Convert.ToBoolean(ostatak3);
                vrij1 /= 2;
                bit.Set(bb, ostatak33);
            }
       }
          int min_vrijednost = pretvoriInt(bit);
```

Slika 29. Programski kod koji otkriva vrijednost veličine teksta

```
int pretvoriInt(BitArray bitArray)
{
     int[] polje = new int[1];
     bitArray.CopyTo(polje, 0);
     return polje[0];
 }
```

Slika 30. Programski kod koji pretvara int vrijednost u bajt

Prilikom otkrivanja teksta potrebno je koristiti polje koje služi za punjenje bitova u niz (eng. BitArray). Ugnježđena for petlja se koristi za dohvaćanje svakog piksela unutar slike. Varijabla *brojac2* ima ulogu brojača koja broji slova do maksimalne veličine teksta. Ukoliko je vrijednost brojača manja od veličine teksta, odvija se grananje. Drugim riječima, kada je vrijednost brojača manja od veličine teksta, označava postojanje slova koja se mogu otkriti. Grananje služi kako bi se u 3 koraka otkrila vrijednost jednog slova. Kod svih slučajeva, potrebno je uzeti zadnji bit svake RGB boje. Međutim, kako je svako slovo skriveno u 3 različita piksela, tako se i ovdje mora otkrivanje slova izvršiti u 3 navrata odnosno 3 slučaja. Prvi slučaj se koristi kod prvog piksela, te se kod njega uzima zadnji bit. Dijeljenjem s 2 i uzimanjem ostatka, dobiva se zadnji bit koji predstavlja jedan bit u oktetu binarnog zapisa sakrivenog slova.Isti bit se uzima i pretvara u bool vrijednost koristeći metodu *Convert.ToBoolean*. Rezultat pretvaranja se pohranjuje u polje. Drugi slučaj se odnosi na drugi piksel. Treći slučaj se odnosi na treći piksel, te se ujedno tamo obavlja punjenje u niz bitova iz polja. Za potrebe punjenja koristi se for petlja koja se kreće od vrijednosti 7 do 0. Jednom napunjenji niz potrebno je pretvoriti u bajt koristeći novu metodu *PretvoriByte* koja obavlja pretvorbu bitova u bajt. Dobiveni bajt se metodom *Convert.ToChar* pretvara u slovo prema ASCII tablici. Na kraju se brojac postavlja na vrijednost 0, te brojac2 se uvećava za 1 kako bi sljedeći put program dohvatio novo slovo. U tekstualnom okviru se postavljaju slova koja su otkrivena u programu. Ukoliko slika ne sadrži tajnu poruku, javit će se poruka da tekst nije pronađen. Zadnja opcija programa će omogućiti spremanje dobivene tajne poruke u tekstualnu datoteku koju korisnik može pohraniti na svoje računalo.

```
byte PretvoriByte(BitArray bitovi)
{
    byte[] bajt = new byte[1];
    bitovi.CopyTo(bajt,0);
    return bajt[0];
 }
```

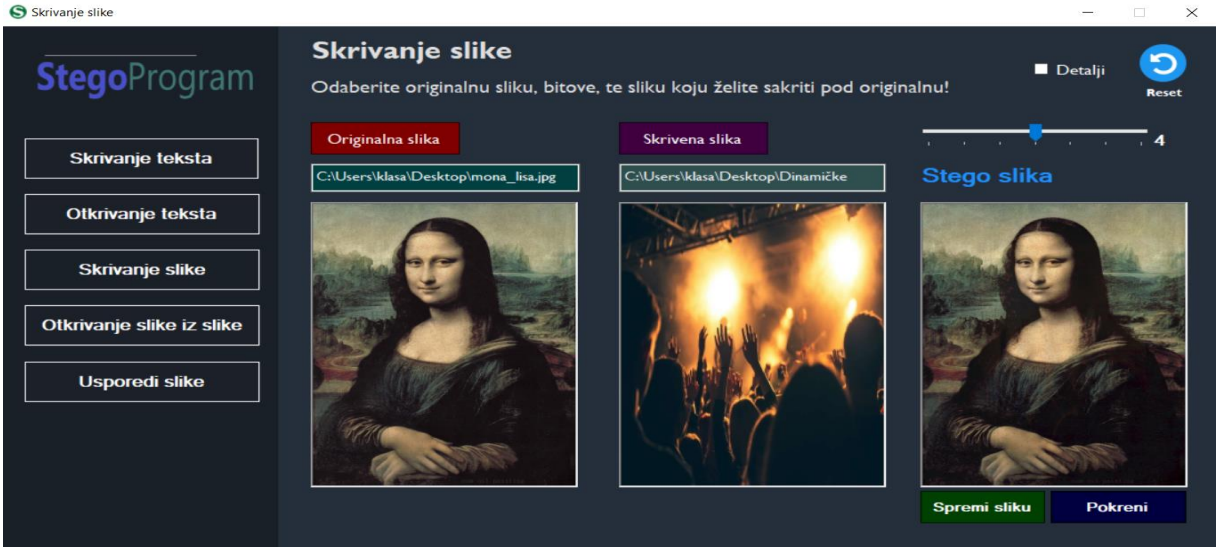
Slika 31. Programski kod koji pretvara bitove u bajt

```
 if (brojac2 < min_vrijednost)
{
    brojac++;
    switch (brojac)
    {
        case 1:
        {
            int pikseliR = pikseli.R % 2;
            bool pikselR = Convert.ToBoolean(pikseliR);
            polje.SetValue(pikseliR, 7);

            int pikseliG = pikseli.G % 2;
            bool pikselG = Convert.ToBoolean(pikseliG);
            polje.SetValue(pikseliG, 6);

            int pikseliB = pikseli.B % 2;
            bool pikselB = Convert.ToBoolean(pikseliB);
            polje.SetValue(pikseliB, 5);
        }
          break;
       case 2:
       {
            int pikseliR = pikseli.R % 2;
            bool pikselR = Convert.ToBoolean(pikseliR);
            polje.SetValue(pikseliR, 4);

            int pikseliG = pikseli.G % 2;
            bool pikselG = Convert.ToBoolean(pikseliG);
            polje.SetValue(pikseliG, 3);

            int pikseliB = pikseli.B % 2;
            bool pikselB = Convert.ToBoolean(pikseliB);
            polje.SetValue(pikseliB, 2);
     }
        break;
      case 3:
      {
           int pikseliR = pikseli.R % 2;
           bool pikselR = Convert.ToBoolean(pikseliR);
           polje.SetValue(pikseliR, 1);

           int pikseliG = pikseli.G % 2;
           bool pikselG = Convert.ToBoolean(pikseli.G);
           polje.SetValue(pikseliG, 0);

           BitArray b1 = new BitArray(8);

           for (int n = 7; n >= 0; n--)
           {
               bool bitovi = Convert.ToBoolean(polje.GetValue(n));
               b1.Set(n, bitovi);
           }

           byte bajt = new byte();
           bajt = ConvertToByte(b1);
           char slovo = Convert.ToChar(bajt);
           string tekst = System.Text.Encoding.ASCII.GetString(new byte[] { Convert.ToByte(slovo)});

           cijeli_tekst = cijeli_tekst + tekst;
           brojac = 0;
           brojac2++;
     }
       break;
}
```

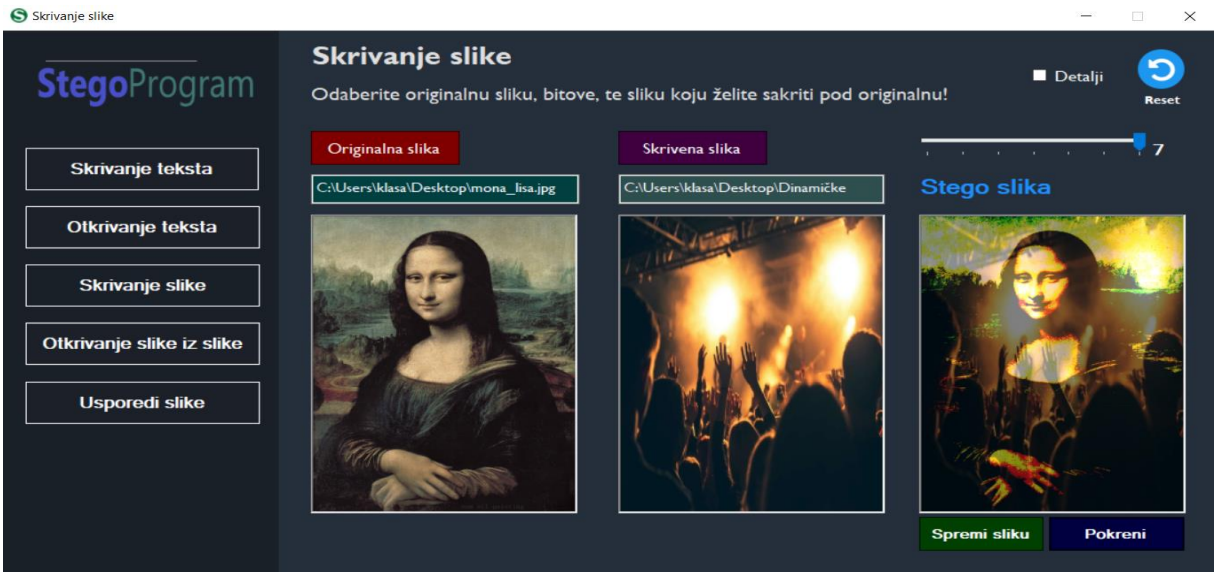
Slika 32. Programski kod koji vrši otkrivanje slova LSB metodom

6.6 Skrivanje slike unutar slike

Ova opcija se zasniva na korištenju LSB metode, te odabirom bita pod kojima želimo sakriti sadržaj. Cijela metoda je opisana u poglavlju Skrivanje slike u slici. U ovom primjeru ću koristiti ovu metodu s 4 najmanje značajna bita. Prvi korak je odabir originalne slike, koja će imati ulogu naslovne slike za sakrivanje sadržaja. Nakon toga je potrebno odabrati sliku koju želimo sakriti pod originalnu sliku. U svakom trenutku korisnik može poništiti odabrane podatke tako da pritisne gumb *resetiranje*. Klikom na gumb *pokreni*, otvara se dijaloški okvir koji ponovno pita korisnika za pokretanje procesa. Potvrdnim odgovorom, proces se pokreće te nakon dužeg vremena daje rezultate u obliku stego slike. Originalna i stego slika se razlikuju u kontrastu, te se ovdje može primjetiti da stego slika ima vidljive obrise skrivene slike.



Slika 33. Postupak skrivanja slike unutar slike koristeći LSB metodu 4 bita



Slika 34. Postupak skrivanja slike unutar slike koristeći LSB metodu 7 bitova

6.7 Implementacija skrivanja slike unutar slike

Odabirom opcije *skrivanja slike unutar slike*, otvara se novi prozor koji daje mogućnost uporabe procesa skrivanja slike. Prvi korak je odabir naslovne i tajne slike, za to je potreban kod koji odgovara istom kodu koji je naveden kod implementacije skrivanja teskta unutar slike. Odabirom slika, potrebno je slike izjednačiti po dužini i širini. Drugim riječima, slike moraju biti istih dimenzija. U varijable *original* i *skrivena* stavljamo lokacije slika koje smo odabrali. Kako bi mogli mogli manipulirati s pikselima odabranih slika, potrebno je kreirati dvije nove bitmape. U nove varijable *originalSirina* i *skrivenaSirina* postavljamo vrijednost širine originalne odnosno skrivene slike. IF uvjetom ispitujemo je li širina originalne slike veća od skrivene. Ukoliko je, treba u varijablu originalSirina postaviti vrijednost širine skrivene slike. Postupak se radi i za obrnuti slučaj, kada skrivena slika ima veću širinu od originalne. Isti princip je i za visinu slike. Nakon podešavanja dimenzija slika, potrebno je uvesti dvije nove bitmape sa novim dimenzijama. Primjer: *new Bitmap*(slika, new Size(novaSirina, novaVisina));

```
var original = lokacija_slike1;
var skrivena = lokacija_slike2;

Bitmap original1 = (Bitmap)Image.FromFile(lokacija_slike1);
Bitmap skrivena1 = (Bitmap)Image.FromFile(lokacija_slike2);

var originalSirina = original1.Width;
var skrivenaSirina = original1.Width;

if (skrivenaSirina < originalSirina)
{
    originalSirina = skrivenaSirina;
}
else
{
 skrivenaSirina = originalSirina;
 }

 var originalVisina = original1.Height;
 var skrivenaVisina = skrivena1.Height;

if(skrivenaVisina < originalVisina)
{
    originalVisina = skrivenaVisina;
}
else
{
    skrivenaVisina = originalVisina;
}
Bitmap original_slika = new Bitmap(original1, new Size(originalSirina, originalVisina));
Bitmap skrivena_slika = new Bitmap(skrivena1, new Size(skrivenaSirina, skrivenaVisina));
Bitmap stego_slika = new Bitmap(original_slika);
```

Slika 35. Prikaz programskog koda za podešavanje dimenzija slika

U programu korisnik bira bitove pod kojima želi sakriti sliku. Za ovu potrebu koristi se grananje koje će omogućiti skrivanje tajne slike pod 7 različitih bitova. Svaki bit odgovara jednom slučaju. Metoda *proces_skrivanja* je razložena na 3 slike koje su priložene na sljedećim stranicama. Opis rada programa se navodi u daljnjem tekstu.

```
            switch (bit)
            {
               case 1:
                  {
                     original = 2;
                     skrivena = 128;
                     proces_skrivanja(original, skrivena);
                  }
                  break;
               case 2:
                  {
                     original = 4;
                     skrivena = 64;
                     proces_skrivanja(original, skrivena);
                  }
                  break;
               case 3:
                  {
                     original = 8;
                     skrivena = 32;
                     proces_skrivanja(original, skrivena);
                  }
                  break;
               case 4:
                  {
                     original = 16;
                     skrivena = 16;
                     proces_skrivanja(original, skrivena);
                  }
                  break;
               case 5:
                  {
                     original = 32;
                     skrivena = 8;
                     proces_skrivanja(original, skrivena);
                  }
                  break;
               case 6:
                  {
                     original = 64;
                     skrivena = 4;
                     proces_skrivanja(original, skrivena);
                  }
                  break;
               case 7:
                  {
                     original = 128;
                     skrivena = 2;
                     proces_skrivanja(original, skrivena);
                  }
                  break;
            }
         }
```

Slika 36. Prikaz prosljeđivanja parametara bita za skrivanje slike

Primjer: LSB metoda s 4 bita, potrebno je odbaciti posljednja 4 bita originalne slike, te ugraditi 4 nova za bitove skrivene slike. Najjednostavniji način je korištenje ugnježđene for petlje koja će prolaziti kroz piksele i dijeliti vrijednosti njihovih boja sa 16 (jer se radi o 4 bita). Prva petlja ide po širini, a druga po visini originalne slike. Zatim slijedi dohvaćanje vrijednosti piksela. Svaku vrijednost RGB boje dijelimo sa 16 kako bi odabacili 4 bita. U istoj liniji koda možemo dodati * 16, koja ima zadatak nadopuniti binarni zapis s bitovima koji su odbačeni. Riječ je o bitovima 0000. Na kraju potrebno je omogućiti spremanje izmijenjenih piksela pod sliku. Postupak je isti i za skrivenu sliku, samo što nije potrebno množiti sa 16, jer je cilj dobiti zatamnjenu sliku koja će se ugrađivati pod originalnu.

```
for (int i = 0; i < original_slika.Width; i++)
{
    for (int j = 0; j < original_slika.Height; j++)
    {
        Color piksel = original_slika.GetPixel(i, j);
        double rj1 = (piksel.R / javno) * javno;
        int rezultat1 = (int)rj1;

        double rj2 = (piksel.G / javno) * javno;
        int rezultat2 = (int)rj2;

        double rj3 = (piksel.B / javno) * javno;
        int rezultat3 = (int)rj3;
        original_slika.SetPixel(i, j, Color.FromArgb(rezultat1, rezultat2, rezultat3));
        pictureBox2.Image = original_slika;
    }
}
```

Slika 37. Prikaz programskog koda koji služi za izmjenu originalne slike

```
for (int x = 0; x < skrivena_slika.Width; x++)
{
   for (int y = 0; y < skrivena_slika.Height; y++)
   {
       Color pikseli = skrivena_slika.GetPixel(x, y);

        double rez1 = (pikseli.R / tajno);
        int rjesenje1 = (int)rez1;

        double rez2 = (pikseli.G / tajno);
        int rjesenje2 = (int)rez2;

        double rez3 = (pikseli.B / tajno);
        int rjesenje3 = (int)rez3;

        skrivena_slika.SetPixel(x, y, Color.FromArgb(rjesenje1, rjesenje2, rjesenje3));
    }
}
```

Slika 38. Prikaz programskog koda koji služi za izmjenu skrivene slike

Nakon uspješne promjene bitova, potrebno je napisati program koji će objediniti bitove originalne i skrivene slike u stego sliku. Postupak se odvija tako da se ponovno koristi for petlja koja će ići po pikselima originalne slike (stego_slika = originalna_slika) jer unutar nje želimo sakriti sadržaj. Zatim slijedi dohvaćanje trenutnih piksela originalne i skrivene slike (eng. GetPixel). Kako bi dobili stego sliku, potrebno je zbrojiti bitove originalne i stego slike. Postepenim zbrajanjem vrijednosti RGB boja, dobiva se stego slika. Na kraju, postavljamo zbrojene piksele unutar bitmape koju zatim stavljamo u okvir za stego slike.

```
for(int u=0; u<stego_slika.Width; u++)
{
  for(int z=0; z<stego_slika.Height; z++)
  {
    Color piksel_original = original_slika.GetPixel(u, z);
    Color piksel_skrivena = skrivena_slika.GetPixel(u, z);

    var glavni1 = piksel_original.R + piksel_skrivena.R;
    var glavni2 = piksel_original.G + piksel_skrivena.G;
    var glavni3 = piksel_original.B + piksel_skrivena.B;

    stego_slika.SetPixel(u, z, Color.FromArgb(glavni1, glavni2, glavni3));
    pictureBox4.Image = stego_slika;
  }
}
```

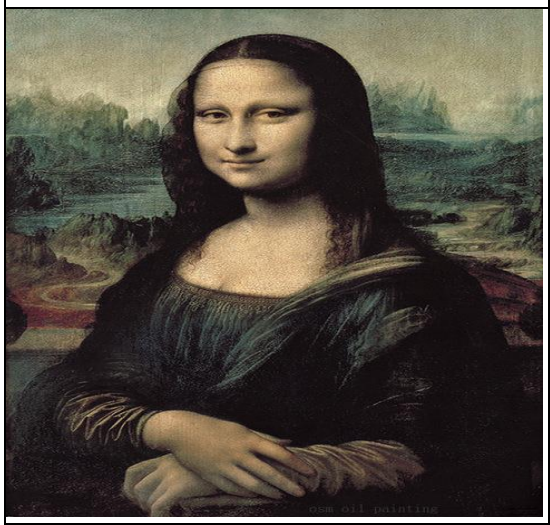
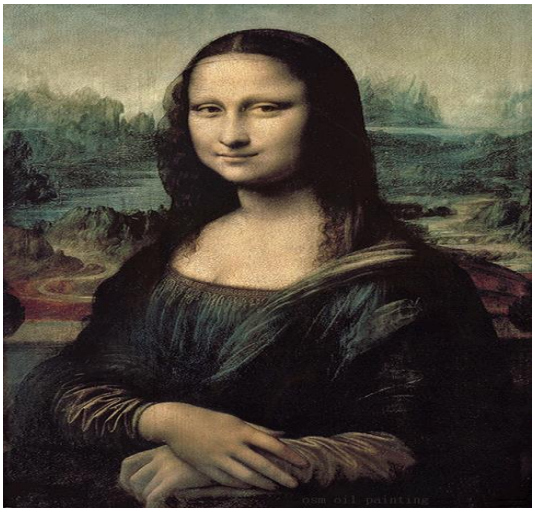
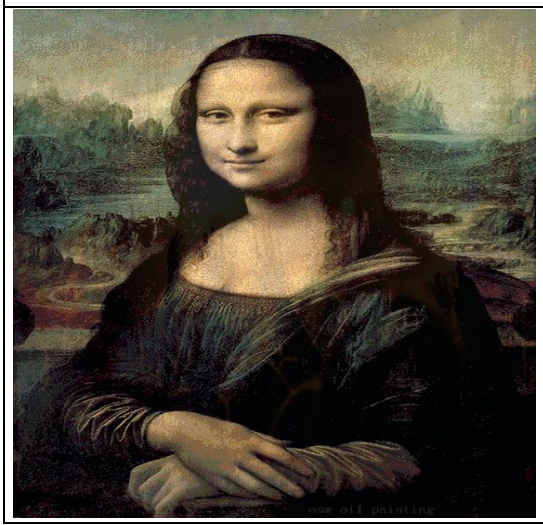
Slika 39. Prikaz programskog koda koji služi za stvaranje stego slike

Nakon izvršavanja svakog dijela ovoga koda, dobiva se konačno rješenje u obliku stego slike. Stego slika se može spremiti tako da koristimo standardni kod za spremanje datoteka na računalo. Opisani postupak spremanja datoteke je opisan u implementaciji skrivanja teksta pod sliku. U odnosu na preostale primjere spremanja, ovdje se koristi nova bitmapa koja sadrži stego sliku. Standardnom naredbom *spremi*, spremamo stego sliku na željeno mjesto na računalu.

```
public void spremi_sliku()
{
  SaveFileDialog spremi = new SaveFileDialog();
  spremi.Filter = "png files(*.png)|*.png|jpg files(*.jpg)|*.jpg|All files(*.*)|*.*";
  string nova_lokacija = "";

  if (spremi.ShowDialog() == DialogResult.OK)
  {
      nova_lokacija = spremi.FileName.ToString();
      Bitmap slika_spremi = new Bitmap(pictureBox4.Image);
      slika_spremi.Save(nova_lokacija);
   }
}
```

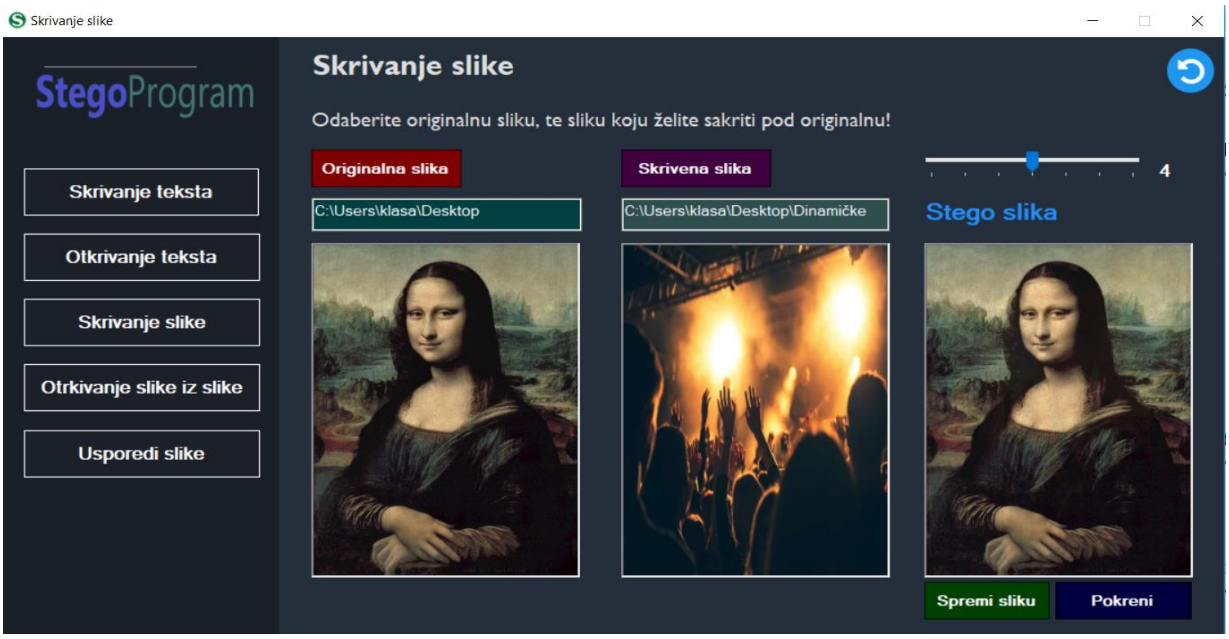
Slika 40. Programski kod za spremanje stego slike

Tablica 7. Prikaz rezultata skrivanja slike koristeći različite bitove

Originalna slika		Slika koju želimo sakriti
		
Stego slika s LSB-om 1 bitom		Stego slika s LSB-om 3 bita
		
Stego slika s LSB-om 5 bitova		Stego slika s LSB-om 7 bitova
		

6.8 Otkrivanje slike u slici

Biranjem opcije *otkrivanje slike* iz programa, otvara se novi prozor koji otkriva sliku u slici. Klikom na gumb *odaberi sliku*, otvara se prozor za pregledavanje i odabir slike iz koje želimo otkriti sadržaj. Učitavanjem slike, pokreće se metoda koja ispisuje dimenzije slike, te se dodatno prikazuje gumb za pokretanje procesa otkrivanja slike iz slike. Pomoću klizača korisnik bira pod koliko bitova želi otkriti sadržaj. Pritiskom na gumb *otkrij sliku*, otvara se dijaloški okvir koji provjerava želi li korisnik pokrenuti proces. Potvrđivanjem odgovora, pokreće se proces otkrivanja. Izvršavanjem procesa, prikazuje se tajna slika iz originalne slike.



Slika 41. Prikaz postupka otkrivanja slike iz slike

Svaka slika se nakon procesa otkrivanja može spremiti na računalo. U slučaju da korisnik želi izbrisati slike, to se može napraviti klikom na gumb *resetiranje* (koji je označen strelicom). Metoda se zasniva na LSB metodi zamijene najmanje važnih bitova iz okteta. Činjenica je da se ponovno moraju izvlačiti bitovi iz okteta kako bi pročitali njen sadržaj. Potrebno je podjeliti svaku vrijednost RGB boje piksela s bitom koji odaberemo. Primjer: ako se radi o LSB metodi s 4 bita, potrebno je izvući 4 tajna bita iz originalne slike, a ostala 4 pripadaju originalnoj slici. Izvlačenje tajnih bitova iz originalne slike se postiže ostatkom dijeljenja tzv. modom. Nakon dijeljenja oktet se nadopunjava s 4 bita nule, kako bi se nadopunio oktet. Isti princip se koristi i kod originalne slike, samo što u tom slučaju ne treba ostatak dijeljenja već rezultat dijeljenja. Postupak dijeljenja je različit za preostale bitove.

6.9 Implementacija otkrivanja slike u slici

Prvi korak je stvaranje bitmape koja sadrži lokaciju stego slike. Za potrebe otkrivanja tajne slike u slici, koristi se ugnježđena petlja koja prolazi kroz širinu i visinu slike. Zatim se dohvaćaju trenutni pikseli for petlje koji su potrebni za otkrivanje tajne slike odnosno originalne slike. Svaki piksel dijelimo s bitom koji smo odabrali u postavkama. Ako je riječ o otkrivanju originalne slike, tada vrijednosti RGB boje pomnožimo s istim bitom, kako bi nadomjestili prazninu u oktetu binarnog zapisa. Međutim, ukoliko se radi o skrivenoj slici, tada se ostatak dijeljenja množi s bitom koji je identičan za sve tri boje u pikselu. Nakon izračuna vrijednosti boja, potrebno je u bitmapu postaviti nove vrijednosti piksela koji će predstavljati tajnu sliku.

```
public void otkrivanje_skrivene_slike(int odabrani_bit,int skriveni)
{
    Bitmap slika1 = new Bitmap(lokacija_slike);

     for (int i=0; i<slika1.Width; i++)
     {
         for(int j=0; j<slika1.Height; j++)
         {
             Color piksel = slika1.GetPixel(i, j);
              int vrijednost1 =(piksel.R % odabrani_bit)* skriveni;
              int vrijednost2 =(piksel.G % odabrani_bit)* skriveni;
              int vrijednost3 = (piksel.B % odabrani_bit)* skriveni;

              slika1.SetPixel(i, j, Color.FromArgb(vrijednost1, vrijednost2, vrijednost3));
              pictureBox2.Image = slika1;
              skrivena = true;
          }
     }
}
```

Slika 42. Prikaz programskog koda koji služi za otkrivanje tajne slike

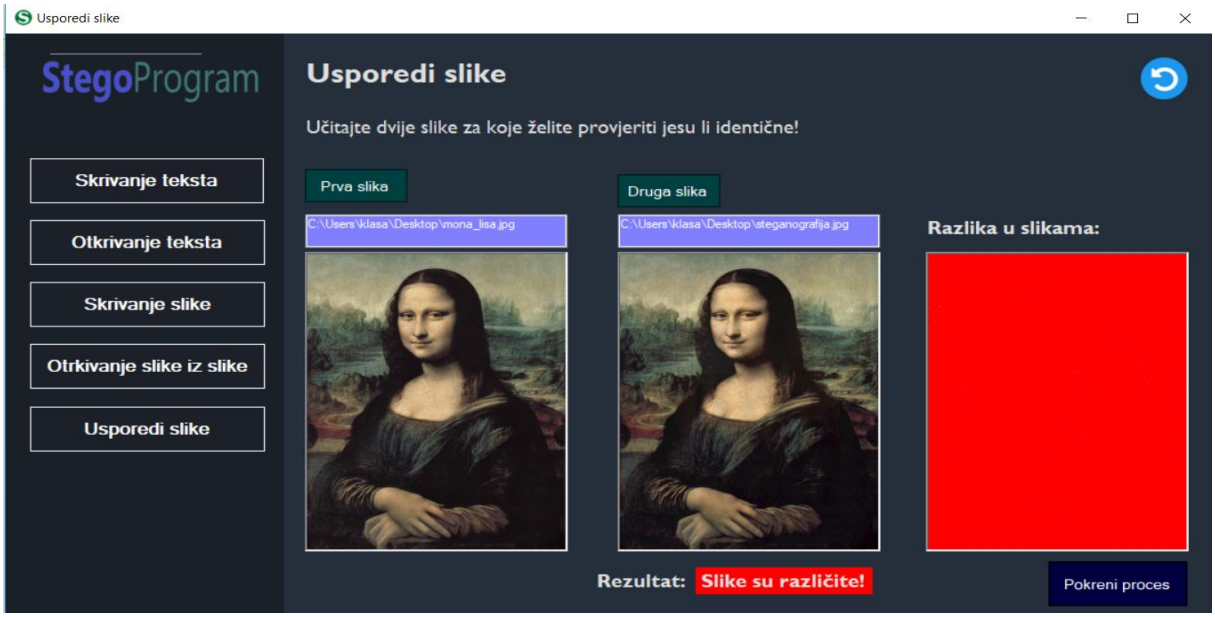
```
public void otkrivanje_original_slike(int odabrani_bit)
{
    Bitmap original_slika = new Bitmap(lokacija_slike);

    for(int i=0; i<original_slika.Width; i++)
    {
        for(int j=0; j<original_slika.Height; j++)
        {
            Color pikseli = original_slika.GetPixel(i, j);
            int vrijednost21 = (pikseli.R / odabrani_bit) * odabrani_bit;
            int vrijednost22 = (pikseli.G / odabrani_bit) * odabrani_bit;
            int vrijednost23 = (pikseli.B / odabrani_bit) * odabrani_bit;

            original_slika.SetPixel(i, j, Color.FromArgb(vrijednost21, vrijednost22, vrijednost23));
            pictureBox3.Image = original_slika;
            original = true;
         }
    }
}
```

Slika 43. Prikaz programskog koda koji služi za otkrivanje originalne slike

6.10 Uspoređivanje slika

Zadnja opcija u aplikaciji se odnosi na usporedbu dviju slika odnosno na lakši primjer steganalize. Glavni zadatak programa je ta da kada korisnik odabere dvije slike, program mu vraća povratnu informaciju jesu li slike iste ili nisu. Dodatak povratnoj informaciji je slika koja predstavlja prikaz piksela koji su različiti iz tih odabranih slika. Crvenom bojom su označeni pikseli koji su različiti, dok su bijelom bojom označeni oni koji su isti u obadvije slike. Prvi korak se odnosi na odabir slika koje želimo usporediti. Klikom na gumb *pokreni proces* pokreće se proces usporedbe slika. Nakon nekoliko sekundi program vraća povratnu informaciji o podudaranosti.

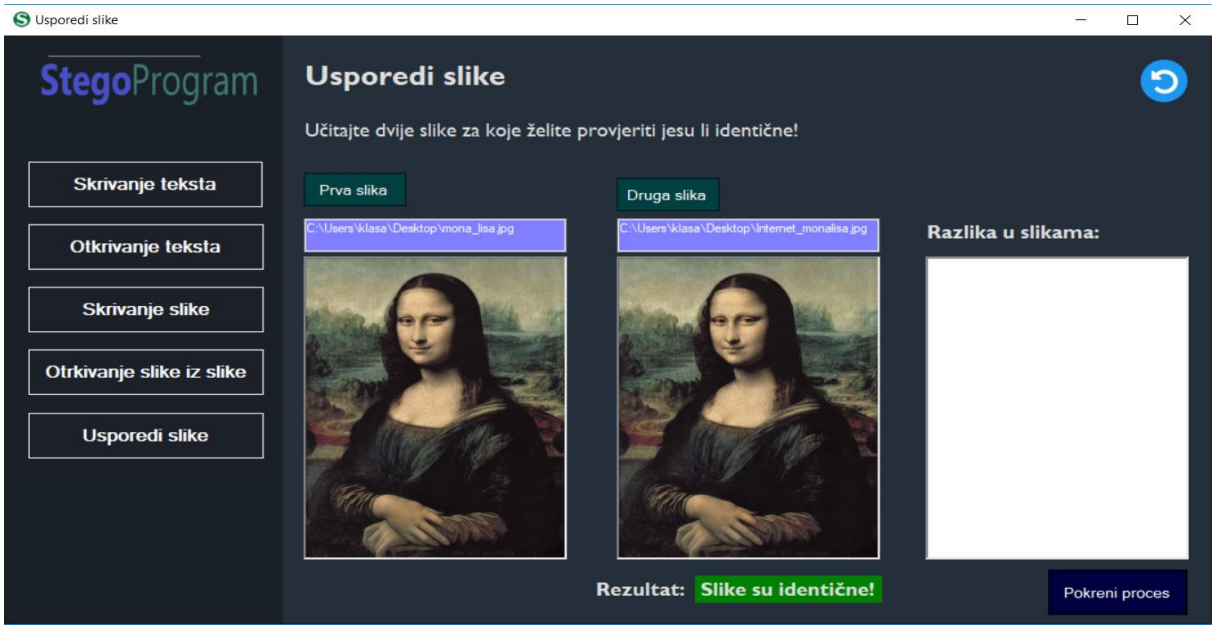
1. primjer: u ovom primjeru koristiti ću originalnu sliku Mona Lise koju sam koristio u prethodnim opcijama, a za drugu sliku ću odabrati stego sliku koju sam dobio prilikom skrivanja slike. Pokretanjem procesa usporedbe, program mi vraća informaciju da su slike različite, te sam kao prilog dobio sliku koja je cijela crvene boje. U ovom slučaju crvena boja predstavlja različitost slika, što znači da je doslovno svaki piksel različit u odnosu na originalnu sliku. Međutim, kada razmislimo o rezultatu, to je logično rješenje jer smo tamo kod svakog piksela mijenjali vrijednosti RGB boje.



Slika 44. Rezultat postupka uspoređivanja originalne i stego slike

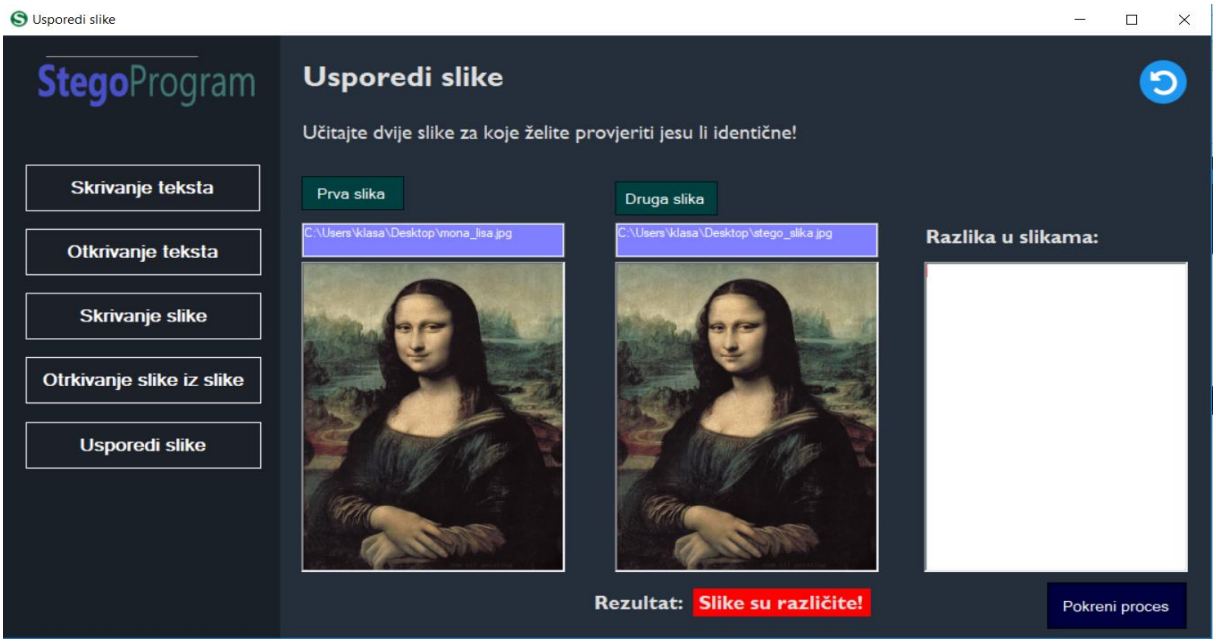
2. primjer: u drugom primjeru ću pokazati da su dvije slike identične. Za potrebe ovog uspoređivanja, učitati ću sliku Mona Lise sa računala. Drugu sliku tj. istu takvu ću preuzeti s Interneta kako bi vidio je li točan podatak da slika gubi na kontrastu tijekom preuzimanja. Pokretanjem procesa dobiven je rezultat koji upućuje da su slike

identične. Osim povratne informacije, dobivena je bijela slika, što znači da se vrijednosti svih piksela podudaraju.



Slika 45. Rezultat postupka uspoređivanja dviju istih slika

3. primjer: ovaj primjer obuhvaća odabir dviju slika koje su ljudskom oku identične. Pokretanjem procesa, dobiva se rezultat o različitosti slika, a dobivena slika je naizgled cijela bijela. Detaljnijim promatranjem lijevog gornjeg kuta, možemo vidjeti sitne crvene točke (piksele). Točke predstavljaju tekst koji smo sakrili pod sliku u prvoj opciji. Zbog takvih „sitnijih stvari“ je potreban ovakav program.



Slika 46. Rezultat postupka uspoređivanja naizgled dviju istih slika

6.11 Implementacija uspoređivanja slika

Učitavanjem slika u program, korisnik u slučaju pogreške prilikom odabira može poništiti slike tako što odabere gumb *resetiranje*. Program koji uljučuje ovu naredbu se sastoji od for petlje koja služi za brisanje slika. Brisanje se postiže tako da se vrijednost slika postavi na null. Sama for petlja se izvodi onoliko puta koliko smo kreirali okvira za slike. Glavni dio programa za uspoređivanje slika je sadržan u metodi pod nazivom: *razlika_slika().* Kako bi mogli općenito raditi sa slikama, potrebno je izraditi dvije bitmape koje će sadržavati slike koje smo odabrali za uspoređivanje. Nakon toga, potrebno je odrediti najmanju visinu i širinu, kako bi mogli izraditi sliku koja će prikazivati različitost piksela. Za određivanje najmanje vrijednost širine i visine, koristi se naredba koja uspoređuje širine odnosno visine odabranih slika (eng. Math.Min). Određivanjem dimenzija, izrađuje se nova bitmapa za treću sliku koja je zadužena za prikaz piksela koji su različiti. Drugi korak se odnosi na uvođenje novih varijabli koji će imati ulogu brojača. Upravo te varijable će sadržavati broj koliko ima istih ili različitih piksela. Kako bi mogli razlikovati različite piksele od onih istih, potrebno je dodijeliti boje. Crvena boja prikazuje različite piksele, dok bijela prezentira iste piksele. Varijable *poz_odgovor* i *neg_odgovor* služe kako bi igrale ulogu RGB boja kod postavljanja piksela na sliku. Ugnježđenom for petljom dolazimo do svakog piksela slike, te se dohvaćaju trenutni pikseli odabranih slika. Zatim slijedi glavni dio programa koji se odnosi na uvjet koji provjerava jesu li pikseli druge slike sadržani unutar piksela prve slike. Ukoliko jesu, uvjetom se uspoređuje vrijednost svake boje određenog piksela. Logika nalaže da se sve RGB boje prve slike moraju podudarati sa RGB bojama druge slike određenog piksela. Ukoliko se pronađe različitost samo jedne boje, program odmah zabilježi da je slika različita tako što na sliku ucrta crveni piksel. Međutim, ako su sve RGB boje jednake, tada se bilježi piksel bijele boje. Postupak uspoređivanja se izvršava sve dok for petlja ne prođe sve piksele slike po visini i širini. Na kraju se gledaju varijable koje su u gornjem dijelu koda definirane za ulogu brojača. Tada drugim uvjetom provjeravamo je li *brojac_neg* sadrži vrijednost 0 ili neku drugu vrijednost. Ukoliko varijabla sadrži nulu, tada se ispisuje poruka da su slike identične. U slučaju da varijabla sadrži neku drugu vrijednost osim nule, tada će program ispisati da su slike različite. Konačnu sliku koja prikazuje različitost piksela možemo prikazati u okviru za slike namijenjenog za tu namjenu.

```
public void razlika_slika()
 {
     Bitmap prva_slika = new Bitmap(lokacija_slike1);
     Bitmap druga_slika = new Bitmap(lokacija_slike2);

     int sirina = Math.Min(prva_slika.Width, druga_slika.Width);
     int visina = Math.Min(prva_slika.Height, druga_slika.Height);

     Bitmap treca_slika = new Bitmap(sirina, visina);

     int brojac_poz = 0;
     int brojac_neg = 0;

     Color poz_odgovor = Color.White;
     Color neg_odgovor = Color.Red;

     for (int i = 0; i < treca_slika.Width; i++)
     {
        for (int j = 0; j < treca_slika.Height; j++)
        {
            Color piksel = prva_slika.GetPixel(i, j);
            Color piksel2 = druga_slika.GetPixel(i, j);

            if (piksel.Equals(piksel2))
           {
               if (piksel.R == piksel2.R && piksel.G == piksel2.G && piksel.B == piksel2.B)
              {
                  treca_slika.SetPixel(i, j, poz_odgovor);
                  brojac_poz = brojac_poz + 1;
              }
           }
           else
           {
              treca_slika.SetPixel(i, j, neg_odgovor);
              brojac_neg = brojac_neg + 1;
           }
        }
     }
    if (brojac_neg == 0)
    {
       labelRezultat.Text = "Slike su identične!";
       labelRezultat.BackColor = Color.Green;
    }
    else
    {
       labelRezultat.Text = "Slike su različite!";
       labelRezultat.BackColor = Color.Red;
    }
    pictureBox3.Image = treca_slika;
}
```

Slika 47.Prikaz programskog koda koji služi za uspoređivanje slika

```
private void btnPokreni_Click(object sender, EventArgs e)
{
   razlika_slika();
}
```

Slika 48. Prikaz programskog koda koji poziva metodu za usporedbu slika

7. Zaključak

Potreba za skrivanjem podataka javila se još u 5.stoljeću prije Krista. Iako je većina metoda širem pučanstvu poznata i danas se koriste, uz male preinake. Velike promjene nastupile su nastankom digitalne tehnologije, pojavom Interneta i njegovih mogućnosti. Sam način skrivanja informacija dobio je novi oblik i dimenziju. Prvi načini skrivanja odnosili su se na pisanu formu, tekst. Daljim usavršavanjem čovjeka, steganografija se proširila na slici. Skrivanje informacija u slici najzastupljeniji je način.

Navest ću neke od pozitivnih razloga:

- Sama slika sastoji se od velikog broja bitova, pa je unutar mreže digitalnih podataka lako ubaciti ili zamijeniti niz podataka.
- Posljedice su nevidljive golim okom promatraču. U sliku se može umetnuti tekstualni podatak , slikovna informacija ili kombinacija.
- Kod umetanja podataka u slikovni medij treba voditi računa o kvaliteti slike. Slika treba biti visoke rezolucije i kontrasta.
- Umetnuti dio ne smije biti prevelik, da ne bi došlo do deformacije nositelja.
- Iako je ovaj način skrivanja podataka poznat, veliki broj slika nalazi se na internetskom portalu, pa je nemoguće sve slike kontrolirati i detaljno pregledati.
- Informacija je dobro skrivena i zaštićena. U slučaju detektiranja informacije, treba pronaći način izdvajanja i dekodiranja.
- Steganografske metode mogu biti u kombinaciji sa kriptografskim metodama i poruke su dodatno zaštićene i nerazumljive onima kojima nisu namijenjene.

Uz navedene prednosti, postoje i nedostaci kod skrivanja i prijenosa informacija:

- Postoji veliki broj steganografskih metoda i sama tehnika skrivanja informacija je u stalnom razvoju, čovjek treba konstantno pratiti i usavršavati se, svaka nova metoda kompliciranija je i treba uložiti više vremena i rada.
- Slikovne datoteke velikog su formata i način prijenosa je složeniji i dugotrajniji, često se moraju komprimirati.
- Uporabom LSB metode, kod slike podvrgnute kompresiji, doći će do djelomičnog gubitka informacija.
- Rotacijom slike gube se podaci.

- Način skrivanja i detektiranja podataka je idealan za ilegalne i kriminalne radnje.

Navedene prednosti i nedostatke skrivanja podataka, potaknut će još brži razvoj novih steganografskih metoda s ciljem bržeg i sigurnijeg protoka informacija. Razvoj čovječanstva i prijenosa informacija uvjetuju steganografsku budućnost. Još više treba uložiti truda i znanja u proces steganalize, jer je to proces u razvoju. Proces steganografije treba biti paralelan s procesom steganalize.

8. Popis slika

9. Popis preuzetih slika

• Mona Lisa .jpg format, Internet stranica: https://ae01.alicdn.com/kf/HTB1LcJ8OFXXXXc3XXXXq6xXFXXXM/Spray-Painting-Wall-Artwork-Decorative-Portrait-Painting-Canvas-Printings-Beautiful-Figure-Mona-Lisa Picture-for-Home.jpg_640x640.jpg, pristupano: 10.7.2019.

• Mjesec .jpg format, Internet stranica: https://timedotcom.files.wordpress.com/2017/02/full-moon.jpg, pristupano: 15.8.2019.

• FIPU.png format, Internet stranica: https://fipu.unipu.hr/_news/icons/e545ab4ae6fb10cc7fb973484d6f433e8559_icon.png, pristupano: 25.11.2018.

• Night Out .jpg format, Internet stranica: https://www.rentinc.co.uk/blog/wp-content/uploads/best-nights-out-in-leeds-key-club.jpg, pristupano 28.6.2019.

• Mačka .png format, Internet stranica: https://1.bp.blogspot.com/kS9BSyMtxu0/VWxDlaEtsAI/AAAAAAAAjHU/snjjsmab7vQ/s728-e100/hack-computer.jpg, pristupano: 25.11.2018.

10. Popis tablica

11. Literatura

[1] Dujella, A.; Maretić M.: Kriptografija, Zagreb, 2007.

[2] Warkentin M., Pope M., Bekkering E., Schmidt M.,*Digital Steganography-An Introduction toTechniques and Tools*,Article 22, Volume 30, 2012., Internet stranica:https://aisel.aisnet.org/cgi/viewcontent.cgi?referer=https://www.google.com/&httpsredir=1&article=3664&context=cais, pristupano: 30.6.2019.

[3] Singh, S., *Šifre*, Mozaik knjiga, Zagreb, 2003.

[4] Lunde P., *Tajne kodova*, Izdavač Novi Liber, 2010.

[5] Kuksov I., *What is digital steganography?*, Kaspersky daily: Technology, 2019, Internet stranica: https://www.kaspersky.com/blog/digital-steganography/27474/, pristupano 30.6.2019.

[6] Zaidoon Kh, A. Zaidan,A.A. Zaidan,B.B & Alanazi.H.O., 2010., *Overview: main fundamentals for steganography*, Journal of Computing, 2(3), pp.40-43.

[7] Agarwal M., *Text steganograpic approaches: a comparison*, International Journal of Network Security & Its Applications (IJNSA), siječanj 2013, Internet stranica: https://arxiv.org/ftp/arxiv/papers/1302/1302.2718.pdf, pristupano 30.6.2019.

[8] Banerjee I., Bhattacharyya S., Sanyal G., *Novel Text Steganography through Special Code Generation*, International Conference on Systemics, 2011, Internet stranica:https://pdfs.semanticscholar.org/7476/865d8824b81640639aadd6ac737c47b8c5a3.pdf, pristupano: 30.6.2019.

[9] Shodganga Inflibnet, *Audio steganography tehniques*, Chapter 5, Internet stranica:https://shodhganga.inflibnet.ac.in/bitstream/10603/147552/14/14_chapter%205.pdf, pristupano: 30.6.2019.

[10] Reddy Sharath K., *Audio steganography: The art of hiding secret message*, ECE Department, Internet stranica: https://www.slideshare.net/sharathkanjula1/audio-steganography-sharath, pristupano: 10.7.2019.

[11] JuicyBits, *Spy Pix software*, Apple Computer Inc., 2015, Internet stranica: https://www.juicybitssoftware.com/spypix/, pristupano 10.7.2019.

[12] Dhanani C., Panchal K., *HTML Steganography using Relative links & Multi web-page Embedment*, Volume 2, Issue 2, ISSN: 2321-9939, 2014, Internet stranica: https://www.ijedr.org/papers/IJEDR1402108.pdf, pristupano 10.7.2019.

[13] Kovač R., *Steganografija: analiza mehanizma rada i praktična programska demonstracija*, Otvoreni sustavi i sigurnost, siječanj 2016, Internet stranica: https://security.foi.hr/wiki/index.php/Steganografija_%E2%80%93_tehni%C4%8Dka_analiza_mehanizma_rada_i_prakti%C4%8Dna_programska_demonstracija..html, pristupano: 10.7.2019.

[14] CARNET, *Steganografija,* CCERT-PUBDOC, travanj 2006, Internet stranica: https://www.cis.hr/www.edicija/LinkedDocuments/CCERT-PUBDOC-2006-04-154.pdf, pristupano: 18.7.2019

[15] Jain A., Pankanti S., *The Essential Guide to Image Processing: Fingerprint Recognition (Second Edition)*, Chapter 23, Academic Press, 2009, Internet stranice:https://www.sciencedirect.com/science/article/pii/B9780123744579000238, pristupano: 15.7. 2019.

[16] Averkiou M., *Digital Watermarking*, Tutorials, srpanj 2015, Internet stranica: http://www.imageprocessingplace.com/downloads_V3/root_downloads/tutorials/Digital%20Watermarking.pdf, pristupano: 15.7.2019.

[17] CARNET, *Digitalni vodeni žigovi*, NCERT-PUBDOC-2010-08-310, 2010, Internet stranica:https://www.cert.hr/wp-content/uploads/2019/04/NCERT-PUBDOC-2010-08-310.pdf, pristupano: 15.7.2019.

[18] Techopedia, Microsoft *Visual Studio .NET*, Home/Dictionar/Tags/Development, Internet stranica: https://www.techopedia.com/definition/15740/visual-studio-net, pristupano: 15.7.2019.

SAŽETAK

Naziv: Steganografija i steganaliza

Suvremena tehnologija implicira i suvremeni način komuniciranja, koji može biti javni i tajni. Sve što je tajanstveno zainteresira čovjeka, počevši od same teme rada. Obzirom da je ovo disciplina koja nije široj javnosti poznata i u razvoju je, u prvom dijelu rada opisan je povijesni razvoj steganografije. Na temelju povijesnog razvoja, slijedi podjela i osnovna načela steganografije. Upoznat ćemo se s pojmovima steganalize i vodenog pečata, te njegove primjene. Jednostavniji primjeri metode prostorne domene, supstitucija bita najmanje važnosti i skrivanje slike u slici, zastupljeni su u radu. Uz prilog pisanom radu, priložiti ću implementacijski dio koji će obuhvaćati skrivanje/otkrivanje teksta i skrivanje/otkrivanje slike. Vršit će se i usporedba slika kako bi saznali jesu li slike identične ili različite. Svaki navedeni dio programa se zasniva na LSB metodi, te je cijeli program izrađen koristeći programski jezik C#.

KLJUČNI POJMOVI: steganografija, steganaliza, vodeni pečat, LSB metoda

ABSTRACT

Name: Steganography and steganalysis

Modern technology implies a modern way of communication, which can be public and secret. Everything that is mysteriously interested in man, starting from the very subject of work. Since this discipline is not known to the wider public and is in development, the first part of the paper describes the historical development of steganography. Based on historical development, the division and basic principles of steganography follow. We will get acquainted with the concepts of steganalysis and water seal, and its application. Simpler examples of the spatial domain method, the substitution of the least important bit and the hiding of the image in the picture, are represented in the paper. With the support in writing, I will attach an implementation part that will include hiding / detecting text and hiding / image detection. A comparison of images will also be made to find out whether the images are identical or different. Each given part of the program is based on the LSB method, and the entire program is created using the C # programming language.

KEY POINTS: steganography, steganalysis, watermark, LSB method