UNIVERZITET U NIŠU

ELEKTRONSKI FAKULTET

# Steganografija

**Implementacija PVD algoritma koriscenjem Pillow biblioteke u Python-u**

Seminarski rad

Studijski program: Računarstvo i informatika

Modul: Softversko inženjerstvo


Mentor:

Prof. dr Bratislav Predić

Student:

Danijel Cakić 1770


Niš, Septembar 2024.

# Uvod

U savremenom digitalnom svetu, sigurnost i privatnost informacija su od suštinskog značaja. Sa razvojem tehnologije i rastom količine digitalnih podataka, zaštita ovih podataka od neovlašćenog pristupa i manipulacije postaje sve važnija. U tom kontekstu, tehnologije za skrivanje podataka, kao što je steganografija, igraju ključnu ulogu u očuvanju privatnosti.

Jedan od popularnih metoda steganografije je primena PVD (Pixel Value Differencing) algoritma, koji omogućava skrivanje informacija u slikama na način koji minimalno menja vizualni izgled slike. Ovaj algoritam pruža efikasan način za ubacivanje tajnih podataka u digitalne slike, bez značajnog narušavanja kvaliteta slike. PVD tehnika koristi razliku u vrednostima piksela kako bi skrivene informacije bile inkorporirane u sliku na način koji je teško primetiti golim okom.

Python, kao jedan od najpopularnijih programskih jezika, nudi bogat ekosistem biblioteka koje olakšavaju implementaciju različitih tehnika obrade slika. Pillow, jedna od najkorišćenijih biblioteka za obradu slika u Python-u, pruža sve potrebne alate za rad sa slikama i može se koristiti za implementaciju PVD algoritma. Ova biblioteka omogućava jednostavno manipulisanje slikama, što je ključno za primenu steganografskih tehnika.

U ovom seminarskom radu, istražićemo implementaciju PVD algoritma koristeći Pillow biblioteku u Python-u. Fokusiraćemo se na osnovne principe algoritma, njegove prednosti u skrivanju podataka, kao i praktičnu primenu kroz kodiranje u Python-u. Osvrnućemo se na izazove u implementaciji i pružiti praktične primere koji ilustruju efikasnost PVD algoritma u kontekstu zaštite i skrivanja podataka.

# Steganografija

Steganografija je sofisticirana tehnika skrivanja informacija unutar drugih poruka, medija ili fizičkih objekata s ciljem da se prikrije postojanje tih informacija od neovlašćenih posmatrača. U osnovi, steganografija omogućava skrivanje gotovo bilo koje vrste digitalnog sadržaja, uključujući tekst, slike, video zapise i audio materijale, unutar drugih datoteka ili medija. Ovi skriveni podaci se zatim mogu ekstraktovati na odredištu, gde se dekodiraju i koriste. Ova metoda je posebno korisna kada je važno ne samo zaštititi sadržaj, već i zavarati potencijalne posmatrače da ne posumnjaju u prisustvo skrivenih podataka.

Prvi zabeleženi primer upotrebe termina steganografija pojavljuje se 1499. godine u delu Johanesa Tritemijusa, koji je u svojoj knjizi "Steganographia" izložio principe kriptografije i steganografije pod maskom traktata o magiji. Ipak, istorija steganografije je mnogo starija, a njen najraniji primer može se pronaći u drevnoj Grčkoj, oko 440. godine pre nove ere. Istoričar Herodot u svojim "Istorijama" opisuje kako je Histieus poslao poruku svom vazalu, Aristagori, tako što je obrijao glavu svog najpoverljivijeg sluge, urezao poruku na njegovu kožu, a zatim ga poslao na put nakon što mu je kosa ponovo izrasla. Ovaj primer pokazuje koliko je steganografija bila ingeniozna i praktična čak i u vreme pre postojanja savremene tehnologije. Još jedan primer iz iste epohe je Demaratova upotreba voštanih tablica, gde je poruku pisao direktno na drvenu podlogu table, koja bi potom bila prekrivena slojem voska, čime bi se prikrila stvarna poruka.

Pored istorijskih primera, u savremenom digitalnom svetu steganografija se primenjuje u elektronskim komunikacijama, gde se skriveni podaci mogu nalaziti unutar različitih formata datoteka poput dokumenata, slika, video zapisa ili čak mrežnih protokola. Medijski fajlovi su posebno pogodni za steganografiju zbog svoje velike veličine, što omogućava sakrivanje ogromnih količina informacija uz minimalne promene u vidljivom sadržaju. Na primer, pošiljalac može uzeti običnu slikovnu datoteku i prilagoditi boju svakog stotog piksela tako da odgovara određenom slovu abecede. Ove promene su toliko suptilne da ih osoba koja nije posebno usmerena na traženje takvih skrivenih podataka teško može primetiti.

Prednost steganografije u odnosu na samu kriptografiju je u tome što skrivena poruka ne privlači pažnju na sebe kao objekat analize. Dok kriptografski šifrovane poruke, koliko god neprobojne bile, mogu privući interesovanje i same po sebi mogu biti inkriminišuće u nekim zemljama gde je enkripcija nezakonita, steganografija omogućava ne samo zaštitu sadržaja, već i maskiranje činjenice da se uopšte šalje neka tajna poruka. Za razliku od kriptografije, koja se bavi zaštitom sadržaja poruke šifrovanjem, steganografija teži prikrivanju i postojanja same poruke, čime pruža dvostruki sloj zaštite.

Upotreba steganografije u savremenom kontekstu postala je posebno relevantna u sajber bezbednosti, gde ransomware grupe i drugi zlonamerni akteri koriste ovu tehniku kako bi sakrili informacije tokom napada. Na primer, napadači mogu sakriti zlonamerni softver, instrukcije za komandno-kontrolne servere ili čak osetljive podatke unutar naizgled bezopasnih fajlova, kao što su slike, video zapisi ili tekstualne datoteke. Na ovaj način,

steganografija postaje moćno sredstvo za prenošenje tajnih informacija u okruženju u kojem bi očigledno šifrovanje bilo detektovano ili blokirano.

Steganografija, kao disciplina sa dugom istorijom i sve širim savremenim primenama, predstavlja ključni alat za prikrivanje i zaštitu informacija. Njena primena u digitalnom svetu, uz tehnološke inovacije, otvara nove mogućnosti za prikrivene komunikacije, ali i izazove za otkrivanje i suzbijanje zlonamernih aktivnosti.

## Kako radi steganografija

Steganografija funkcioniše tako što sakriva informacije na način koji ne izaziva sumnju. Jedna od najrasprostranjenijih tehnika je metoda skrivenih bitova, poznata kao "least significant bit" (LSB) steganografija. Ova metoda podrazumeva ugrađivanje tajnih informacija u najmanje značajne bitove medijskih fajlova. Na primer:

Kod slikovnih datoteka, svaki piksel se sastoji od tri bajta podataka koji predstavljaju crvenu, zelenu i plavu boju. Neki formati slike dodaju i četvrti bajt za transparentnost, poznat kao 'alfa' kanal. LSB steganografija menja poslednji bit svakog od ovih bajtova kako bi sakrila po jedan bit informacija. Tako, za sakrivanje jednog megabajta podataka ovom metodom, potrebna je slika veličine osam megabajta. Promena poslednjeg bita vrednosti piksela ne rezultira vidljivom promenom slike, što znači da osoba koja gleda originalnu i modifikovanu sliku neće moći da primeti razliku.

Ovaj metod može se primeniti i na druge digitalne medije, poput audio i video zapisa, gde se podaci sakrivaju u delovima datoteka koji izazivaju minimalnu promenu u zvučnom ili vizuelnom izlazu.

Pored LSB steganografije, postoji i tehnika zamene reči ili slova. Ova metoda podrazumeva da pošiljalac skriva tajni tekst unutar većeg teksta, raspoređujući reči na specifičnim intervalima. Iako je ova metoda jednostavna za primenu, može učiniti da tekst deluje čudno ili neprirodno, pošto tajne reči možda ne odgovaraju kontekstu rečenica u kojima se nalaze.

Druge tehnike steganografije uključuju sakrivanje čitavih particija na hard disku ili ugradnju podataka u zaglavlja fajlova i mrežnih paketa. Efikasnost ovih metoda zavisi od količine podataka koje mogu sakriti i lakoće njihovog otkrivanja.

Softver za steganografiju obavlja različite funkcije, uključujući:

- Sakrivanje podataka, uz prethodnu enkodaciju podataka kako bi se pripremili za skrivanje unutar drugog fajla.
- Praćenje bitova u osnovnoj datoteci koja sadrži skrivene informacije.
- Enkripciju podataka pre nego što se sakriju.
- Ekstrakciju skrivenih podataka od strane predviđenog primaoca.

## Razlika izmedju steganografije i kriptografije

Steganografija i kriptografija su tehnike koje se koriste za zaštitu informacija, ali koriste različite pristupe da postignu ovaj cilj. Kriptografija se fokusira na promenu strukture same poruke kako bi je učinila nerazumljivom svima osim primaocu koji ima ključ za dešifrovanje. Kada je poruka enkriptovana, ona se pretvara u niz šifrovanih podataka (ciphertext) koji su očigledno zaštićeni, ali sam čin šifrovanja može privući pažnju. Čak i ako sadržaj nije čitljiv, prisustvo šifrovane poruke može izazvati sumnju i istraživanje.

S druge strane, steganografija se bavi prikrivanjem same činjenice da postoji neka tajna komunikacija. Umesto da menja sadržaj poruke, steganografija skriva poruku unutar nečeg drugog, kao što su slike, zvučni ili video fajlovi, ili čak običan tekst. Primer ovoga je skrivanje podataka u najmanje značajnim bitovima slike (LSB steganografija), gde promene u slici nisu vidljive ljudskom oku. Na taj način, osoba koja pregleda takav fajl ne bi ni posumnjala da unutar njega postoji skrivena poruka.

Ključna razlika je u tome što kriptografija može obezbediti integritet, autentifikaciju i neospornost, dok steganografija prvenstveno pruža poverljivost bez izazivanja sumnje. Kriptografske metode, kao što su blok šifre i transpozicione šifre, oslanjaju se na složene matematičke procese da zaštite podatke, dok steganografija uglavnom ne menja podatke značajno i često se oslanja na tehniku skrivanja informacija u medijskim fajlovima, kao što su slike ili zvuk.

Kombinacija steganografije i kriptografije može pružiti dodatni nivo sigurnosti. Kada se šifrovana poruka sakrije steganografski, čak i ako neko otkrije da fajl sadrži skriveni sadržaj, on će i dalje biti zaštićen kriptografskim metodama. Ova kombinacija smanjuje šanse da će neko uopšte pokušati da dešifruje poruku, jer neće biti sumnjivo da poruka postoji. Na ovaj način, steganografija sakriva komunikaciju, dok kriptografija osigurava njenu sadržinu.

## Tipovi steganografije

Postoji pet glavnih tipova steganografije, svaki se fokusira na različitu vrstu medija za skrivanje informacija:

1. **Tekstualna steganografija**

   Podrazumeva skrivanje informacija unutar tekstualnih fajlova. Ova tehnika može uključivati promenu formata postojećeg teksta, zamenu reči unutar teksta, upotrebu gramatika nezavisnih od konteksta za generisanje čitljivih tekstova, ili čak generisanje nasumičnih sekvenci karaktera. Jedan od najjednostavnijih načina jeste da se koristi prvi karakter svake rečenice kako bi se formirala skrivena poruka, ali postoje i složenije metode koje koriste interpunkciju ili suptilne tipografske greške.

2. **Steganografija slika**

   Slike su veoma popularan medij za skrivanje podataka zbog velikog broja elemenata (piksela) koji se mogu koristiti. Ova tehnika često koristi metod najmanje značajnog bita (LSB) za skrivanje informacija u digitalnim slikama. U RGB sistemu boja, pikseli slike se prikazuju kombinacijom crvene, zelene i plave boje. Na primer, menjanje poslednjeg bita svake boje na slici omogućava skrivanje informacija, a te promene su neprimetne ljudskom oku.

3. **Audio steganografija**

   Uključuje skrivanje tajnih poruka unutar audio fajlova tako što se menja binarni zapis zvučnog signala. Kao i kod slika, male promene u zvuku su teško primetne za ljudsko uho. Jedan od najpoznatijih primera je "backmasking", gde se tajne poruke postavljaju tako da postaju razumljive kada se zvučni zapis pusti unazad. Pored toga, postoji metoda eho-skrivenja, gde se nakon zvučnog signala dodaje kratak, nečujni eho koji nosi informacije, ili korišćenje intervala tišine za kodiranje podataka.

4. **Video steganografija**

   Ova tehnika omogućava skrivene podatke unutar digitalnih video formata. Video zapisi su zapravo sekvence slika, pa svaka slika može sadržati skrivene podatke korišćenjem metoda steganografije slika. Ova metoda omogućava skrivanje velike količine podataka kroz više kadrova u videu. Dva osnovna pristupa su sakrivanje podataka u nekomprimovanom video zapisu koji se kasnije komprimuje ili direktno sakrivanje u komprimovani video zapis.

5. **Mrežna steganografija**

   Mrežna ili protokolska steganografija koristi protokole za prenos podataka, poput TCP/IP, UDP ili ICMP, za sakrivanje informacija. Skriveni podaci se mogu uključiti u zaglavlja mrežnih paketa ili u razmaku između slanja različitih paketa. Ova tehnika je posebno interesantna za stručnjake iz oblasti mrežne sigurnosti, ali može biti iskorišćena i od strane malicioznih aktera za skrivanje komunikacija.

# Steganografija slika

Steganografija slika je sofisticirana metoda skrivanja informacija unutar slika, koristeći razne tehnike kako bi se podaci sakrili, a slika ostala naizgled nepromenjena. Za razliku od kriptografije, koja se bavi šifrovanjem sadržaja i sprečavanjem neovlašćenog pristupa, steganografija ima za cilj da sakrije postojanje tih podataka, čineći ih neprimetnim za posmatrače.

Glavna ideja steganografije je da se podaci sakriju unutar medija na način da ne utiču na njegovu upotrebu ili izgled. U slučaju slika, medij može biti bilo koja digitalna slika, poput JPEG, BMP, PNG, ili GIF formata. Ove slike se sastoje od piksela, a svaki piksel je predstavljen binarnim vrednostima koje opisuju njegovu boju (u RGB formatu, na primer).

Na primer, u slici od 24 bita (8 bita po svakoj RGB komponenti), promena najmanje značajnog bita u bilo kojoj od ovih komponenti će izazvati tako malu promenu u boji da ljudsko oko neće biti u stanju da je primeti. Ovo omogućava steganografiji da naizgled neprimetno sakrije podatke unutar slike.

1. **Zamena najmanje značajnog bita (LSB - Least Significant Bit)**

   Zamena najmanje značajnog bita je jedna od najjednostavnijih i najčešće korišćenih tehnika steganografije. Ova metoda funkcioniše tako što se zameni poslednji bit svakog bajta koji predstavlja boju piksela sa bitom tajnog podatka.

   Na primer, u 24-bitnom sistemu za boje (8 bita po svakoj od crvene, zelene i plave komponente boje), zamena jednog bita u svakoj od tih komponenti će izazvati minimalnu promenu u boji piksela, koju ljudsko oko ne može da primeti. S obzirom na to da se promena vrši samo na poslednjem bitu, ukupan vizuelni izgled slike ostaje skoro nepromenjen. Međutim, ova tehnika je veoma ranjiva na napade, jer ako se slika kompresuje (kao što je slučaj sa JPEG formatom), sakriveni podaci mogu biti izgubljeni.

   Primer: Originalni piksel: `(11001010, 10111001, 01101100)`

   Tajni bitovi: `101`

   Piksel nakon zamene LSB-a: `(11001011, 10111000, 01101101)`

2. **Tehnike prostorne domene**

   U tehnikama prostorne domene, podaci se direktno ugrađuju u piksele slike. Ove tehnike uključuju:

   - **Šumna tehnika**: Tajni podaci se dodaju na način da se promeni intenzitet piksela, što izaziva minimalne promene koje su neprimetne za ljudsko oko.

- **Šablonska tehnika**: Pikseli slike se reorganizuju ili preslože prema unapred definisanom šablonu kako bi se sakrili podaci. Ova metoda je otpornija na kompresiju od zamene najmanje značajnog bita, ali je često složenija za implementaciju.

3. **Frekvencijska domena**

   Frekvencijska domena koristi matematičke transformacije za sakrivanje podataka u „frekvencijama“ slike, a ne u pikselima. Ova tehnika je otpornija na kompresiju i transformacije slike. Dve najčešće korišćene transformacije su:

   - **Diskretna kosinusna transformacija (DCT)**: Ova metoda deli sliku na više frekvencijskih komponenti. Tajni podaci se sakrivaju u frekvencijama koje su manje značajne za ljudsko oko. DCT se često koristi u JPEG kompresiji, pa je ova metoda idealna za slike u ovom formatu.
   - **Diskretna Furijeova transformacija (DFT)**: Ova metoda pretvara sliku u frekvencijski domen, gde se tajni podaci ugrađuju u visoke frekvencije slike, koje su manje primetne ljudskom oku. Ova tehnika je efikasna, ali zahteva kompleksnije algoritme za enkodovanje i dekodovanje podataka.

4. **Tehnika širenja spektra**

   Tehnika širenja spektra, inspirisana metodama korišćenim u komunikacijama putem radio talasa, širi tajne podatke kroz više piksela u slici. Na ovaj način, podaci postaju teži za otkrivanje jer su „rasuti“ kroz sliku u obliku manjih promena u boji. Ovo povećava otpornost na detekciju, ali takođe komplikuje vađenje podataka.

5. **Adaptivna steganografija**

   Ova tehnika se prilagođava sadržaju slike kako bi optimalno sakrila podatke. Na primer, podaci se mogu sakriti u oblastima slike koje su kompleksnije ili imaju više detalja, jer promene u takvim oblastima teže privlače pažnju. Adaptivna steganografija koristi napredne algoritme kako bi odabrala koje piksele ili frekvencije će se promeniti.

6. **Fraktalna steganografija**

   Koristeći fraktalne obrasce unutar slika, podaci se ugrađuju na način koji je praktično neprimetan. Fraktali omogućavaju visoku kompresiju i otpornost na gubitke tokom prenosa podataka, što je čini korisnom za složenije aplikacije.

Steganografija slika ima široku primenu u raznim industrijama i sektorima:

- **Digitalni vodeni žigovi**: Ova tehnika se koristi za zaštitu intelektualne svojine, gde su informacije o autoru ili vlasniku ugrađene u sliku, audio ili video datoteku kao „vodeni žig“.
- **Zaštićena komunikacija**: Vlade, vojne agencije i obaveštajne službe koriste steganografiju kako bi omogućile tajnu komunikaciju između agenata ili zvaničnika, smanjujući rizik od presretanja.
- **Skrivena razmena podataka u softverima**: Steganografija se koristi u softverima za zaštitu podataka i obezbeđivanje sigurnog prenosa informacija preko nesigurnih kanala.
- **Digitalna forenzika**: Stručnjaci za forenziku koriste steganografiju za ugrađivanje tragova u slike ili dokumente tokom ispitivanja digitalnih dokaza.

Iako je ova metoda našla široku primenu u današnjem digitalnom svetu, postoje izazovi na koje se mora obratiti posebna pažnja pri implementaciji ove metode:

- **Veličina podataka**: U većini slučajeva, količina podataka koja može biti sakrivena unutar slike je ograničena, naročito kada se koriste jednostavne metode poput zamene LSB-a.
- **Kompresija slike**: Tehnike koje se oslanjaju na direktnu manipulaciju piksela često su ranjive na gubitak podataka prilikom kompresije, naročito u lossy formatima poput JPEG-a.
- **Otkrivanje steganalizom**: Steganaliza je proces detekcije skrivenih podataka u medijima. Napredni algoritmi za steganalizu mogu detektovati promene u pikselima ili frekvencijama koje ukazuju na prisustvo skrivenih podataka.

# PVD (Pixel Value Differencing)

PVD (Pixel Value Differencing) je jedan od popularnih algoritama steganografije, posebno u domenu slika, koji omogućava sakrivanje tajnih informacija unutar slika na način koji je teško primetan ljudskom oku. Ovaj algoritam koristi razliku između susednih piksela da odredi koliko bita informacija može da se sakrije u svaku regiju slike, pri čemu se u ivicama i šumovitim delovima slike sakriva više podataka nego u glatkim delovima. Na taj način se koristi ljudska percepcija koja je manje osetljiva na promene u ivicama slike nego u njenim glatkim regijama.

## Kodiranje poruke

PVD algoritam započinje deljenjem slike u male, nepokrivajuće blokove od po dva susedna piksela. Na primer, neka su pikseli unutar i-tog bloka označeni kao $P_i$ i $P_{i+1}$. Razlika između ova dva piksela, označena kao $d_i$, se računa po formuli:

$$d_i = |P_{i+1} - P_i|$$

Vrednost $d_i$ pruža informacije o tome koliko su pikseli slični. Ako je razlika mala, to znači da se nalaze u glatkoj regiji slike, dok velika razlika ukazuje na ivice ili šumovite delove. Algoritam koristi ovu razliku da odluči koliko tajnih bitova može da se sakrije u svaki blok piksela. Na primer, u glatkim regijama će se sakriti manje bitova, dok će se u ivicama, gde je vizuelna percepcija manje osetljiva, sakriti veći broj bitova.
Vrednost $d_i$ se deli na nekoliko regija (raspona), pri čemu svaki raspon definiše koliko bita može da se sakrije. Broj bitova koji se može sakriti, označen kao ttt, računa se pomoću logaritamske funkcije:

$$t=\log_2(\text{gornjagranica} - \text{donjagranica} + 1)$$

Na primer, ako je did_idi u rasponu između 32 i 63, može se sakriti 5 bita informacije ($\log_2(63-32+1)=5$).

Kada se utvrdi broj bitova t, iz tajne poruke se uzima odgovarajući broj bita, zatim se nova vrednost razlike d′ računa po sledećoj formuli:

$$d'=t_d+\text{donjagranica}$$

Ova nova vrednost d′ se zatim koristi da bi se izračunale nove vrednosti za piksele $P_i$ i $P_{i+1}$, što rezultira izmenjenom slikom koja sada sadrži tajne informacije.

## Dekodiranje poruke

Dekodiranje poruke je obrnuti proces. Slika se ponovo deli na blokove od po dva susedna piksela. Za svaki par piksela, razlika $d_i$ između njih se ponovo računa, a zatim se koristi za izdvajanje broja skrivenih bitova u svakom bloku. Na osnovu razlike, može se utvrditi da li je skriveni bit 1 ili 0. Tajna poruka se dekodira ponavljanjem ovog procesa za svaki blok piksela dok se cela poruka ne povrati.

## Primer

Pretpostavimo da imamo sliku i da želimo da sakrijemo binarnu tajnu poruku "1101". Algoritam PVD bi mogao raditi na sledeći način:

1. Slika se deli na blokove od po dva susedna piksela. Neka su vrednosti prvog bloka $P_1=120$ i $P_2=125$.
2. Razlika između ova dva piksela je $d_1=|125-120|=5$. Prema prethodno definisanom rasponu, možemo sakriti 3 bita u ovom bloku.
3. Iz tajne poruke uzimamo prva 3 bita "110". Zatim računamo novu vrednost razlike $d_1'$, koja će sakriti ove bitove. Ako je raspon za razliku $d_1$ od 0 do 7, nova razlika $d_1'$ bi bila $d_1'=6$.

4. Na osnovu nove vrednosti razlike, prilagođavamo piksele $P_1$ i $P_2$ tako da odgovaraju novoj razlici $d_1'$.
5. Ovaj proces se ponavlja za sve piksel blokove dok se cela tajna poruka ne ugradi u sliku.

## Prednosti i izazovi PVD algoritma

Jedna od glavnih prednosti PVD metode je njena sposobnost da sakrije relativno veliki broj podataka bez značajnog narušavanja kvaliteta slike, jer koristi glatke i ivaste regione slike različito. Veće razlike u pikselima omogućavaju više podataka da se sakrije, dok manji regioni ostaju netaknuti kako bi se sačuvala vizuelna vernost slike.

Međutim, PVD metoda može izazvati određenu degradaciju kvaliteta slike, posebno u regijama gde se sakriva veći broj bitova. Zbog toga su mnogi istraživači radili na poboljšanju ovog algoritma. Na primer, neki modifikovani PVD algoritmi kombinuju PVD sa tehnikom zamene najmanje značajnog bita (LSB) kako bi se dodatno poboljšala sigurnost i kvalitet slike.

# Aplikacija

Implementirao sam aplikaciju koja koristi PVD algoritam za sakrivanje i ekstrakciju teksta iz slika, pružajući korisniku mogućnost da izabere između dve glavne funkcionalnosti – **kriptovanje teksta u sliku** i **dekriptovanje teksta iz slike**. Aplikacija nudi intuitivan korisnički interfejs koji omogućava jednostavno rukovanje slikama i tekstualnim fajlovima.

Funkcionalnosti implementirane aplikacije su:

- Ubacivanje teksta u sliku (Kriptovanje)
    - Korisnik najpre bira opciju "Ubacivanje teksta".
    - Nakon toga, bira sliku koja će služiti kao nosilac tajnog teksta.
    - Sledeći korak je izbor tekstualnog dokumenta (.txt fajl) koji sadrži tekst koji će biti skriven u slici.
    - Aplikacija potom koristi PVD algoritam da ugradi tekst u sliku.
    - Na kraju, rezultat je nova slika sa skrivenim tekstom, koja se može sačuvati na željenu lokaciju.
- Vadjenje teksta iz slike (Dekriptovanje)
    - Ako korisnik odabere ovu opciju, prvo bira sliku koja sadrži sakriveni tekst.
    - Aplikacija potom koristi PVD algoritam za ekstrakciju tajnog teksta iz slike.
    - Rezultat je tekstualni dokument koji sadrži izvučen tekst, spreman za pregled ili dalje korišćenje.

Ovaj proces omogućava jednostavno sakrivanje poruka unutar slika i povratno izvlačenje, a aplikacija pruža jasan i jednostavan interfejs za korisnike.

## Opis tehnologija

**Python**

> Python je interpretirani, visoko-nivo programski jezik poznat po svojoj jednostavnoj sintaksi i čitljivosti, što ga čini pogodnim za širok spektar primena. Zbog dinamičkog tipiziranja i podrške za više paradigmi programiranja, Python je vrlo fleksibilan i efikasan alat. Odlučio sam se za Python zbog njegovog bogatog skupa biblioteka koje značajno pojednostavljuju razvoj aplikacija. Posebno su korisne biblioteke poput Pillow za rad sa slikama, kao i Tkinter za izgradnju grafičkog korisničkog interfejsa, što je olakšalo implementaciju ključnih funkcionalnosti u mojoj aplikaciji.

**Tkinter**

> Tkinter je standardna Python biblioteka za izradu grafičkog korisničkog interfejsa (GUI). Omogućava jednostavno kreiranje prozora, dugmadi, polja za unos i drugih

interaktivnih elemenata aplikacije. Tkinter se ističe svojom jednostavnošću i integracijom sa Python-om, što ga čini idealnim za brzi razvoj desktop aplikacija. U mojoj aplikaciji, koristio sam Tkinter kako bih korisnicima omogućio jednostavan interfejs za biranje slike i tekstualnog fajla, kao i za prikaz poruka o uspešnosti procesa. Ova biblioteka je ključna za pružanje intuitivnog iskustva korisnicima aplikacije.

**NumPy**

NumPy je ključna biblioteka u Python-u koja se koristi za rad sa nizovima (arrays) i pruža podršku za velike, multidimenzionalne matrice i nizove, zajedno sa velikim brojem funkcija visokih performansi za matematičke operacije. NumPy je posebno koristan u nauci o podacima, mašinskom učenju i obradi slika, jer omogućava efikasno rukovanje velikim količinama numeričkih podataka. U mojoj aplikaciji koristio sam NumPy za obradu slike u obliku numeričkog niza, što je omogućilo manipulaciju vrednostima piksela prilikom implementacije PVD algoritma.

**Pillow**

Pillow je popularna Python biblioteka za obradu i manipulaciju slika, koja nudi širok spektar funkcionalnosti za rad sa različitim formatima slika kao što su PNG, JPEG, GIF, BMP, i mnogi drugi. Ona omogućava jednostavno otvaranje, menjanje, obradu i čuvanje slika uz minimalan kod. Pillow podržava osnovne operacije kao što su promena veličine slike, sečenje, rotiranje, kao i naprednije operacije poput filtriranja i modifikacije piksela. Zahvaljujući ovim mogućnostima, Pillow je idealna biblioteka za različite projekte koji zahtevaju rad sa slikama u Pythonu.

Posebno je korisna za primene u oblasti steganografije, kao što je implementacija PVD algoritma (Pixel Value Differencing). Pillow omogućava jednostavan pristup pikselima slike, što je ključno za PVD algoritam, koji se zasniva na analizi razlika u vrednostima susednih piksela kako bi se sakrili podaci unutar slike bez značajnog narušavanja njenog kvaliteta. Ova biblioteka olakšava manipulaciju slikama i čini implementaciju složenih algoritama kao što je PVD znatno jednostavnijom i efikasnijom.

# Opis aplikacije

U nastavku ce biti objašene funkcionalnosti aplikacije.

## Odabir operacije

U prvom delu aplikacije, funkcija *choose_operation()* igra ključnu ulogu u odabiru operacije koju korisnik želi da izvrši. Kada se funkcija pokrene, koristi Tkinter biblioteku za kreiranje grafičkog korisničkog interfejsa (GUI). Prvo se kreira glavni prozor, koji dobija naslov "PVD

Algorithm - Choose Operation" i dimenzije od 400x250 piksela, a zatim se centriraju na ekranu korišćenjem metode *eval()*.

Nakon inicijalizacije prozora, definišu se stilovi fonta za naslov i dugmiće kako bi interfejs bio vizuelno privlačniji. U okviru prozora se kreira okvir (frame) sa dodatnim paddingom, u kojem se postavlja oznaka (label) koja korisnika obaveštava o izboru operacije. Oznaka sadrži tekst "Choose what you want to do:", koristeći podešeni stil fonta.

Nakon toga, dodaju se dva dugmeta. Prvo dugme, "Hide Text in Image",povezuje se sa funkcijom *upload_photo_and_text()*, koja će se pozvati kada korisnik odabere ovu opciju. Ovo dugme zatvara trenutni prozor pre nego što pokrene funkciju za učitavanje slike i teksta. Drugo dugme, "Extract Text from Image", povezuje se sa funkcijom *extract_text_from_image()*, koja se aktivira kada korisnik želi da izvuče tekst iz slike.

Na kraju, poziva se *root.mainloop()*, što omogućava aplikaciji da ostane aktivna i reaguje na korisničke akcije dok se prozor ne zatvori. Ovaj deo aplikacije je osnova interakcije, omogućavajući korisnicima da lako biraju između različitih funkcionalnosti i pružajući im jednostavan i intuitivan način za rad sa PVD algoritmom.

```
def choose_operation():
    root = tk.Tk()
    root.title("PVD Algorithm - Choose Operation")
    root.geometry("400x250")
    root.eval('tk::PlaceWindow . center')

    title_font = tkFont.Font(family="Helvetica", size=16, weight="bold")
    button_font = tkFont.Font(family="Helvetica", size=12)

    frame = tk.Frame(root, padx=20, pady=20)
    frame.pack(expand=True)

    label = tk.Label(frame, text="Choose what you want to do:", font=title_font)
    label.pack(pady=10)

    encode_button = tk.Button(frame, text="Hide Text in Image", font=button_font, bg="#4CAF50", fg="black", padx=10, pady=5,
                              command=lambda: [root.destroy(), upload_photo_and_text()])
    encode_button.pack(pady=5, fill=tk.X)

    decode_button = tk.Button(frame, text="Extract Text from Image", font=button_font, bg="#2196F3", fg="black", padx=10, pady=5,
                              command=lambda: [root.destroy(), extract_text_from_image()])
    decode_button.pack(pady=5, fill=tk.X)

    root.mainloop()
```

## Učitavanje fajlova

U delu aplikacije koji se bavi učitavanjem fajlova, kod za opciju "Hide Text in Image" omogućava korisniku da odabere sliku u koju želi da ubaci tekst, kao i tekstualni dokument čiji sadržaj će biti umetnut u sliku. Kada korisnik pokrene aplikaciju i odabere ovu opciju, prvo se stvara skriveni prozor Tkinter-a pomoću *root.withdraw()*, čime se sprečava da se

glavni prozor prikazuje dok se ne završi izbor fajlova. Zatim se otvara dijalog prozor za odabir slike, koristeći *filedialog.askopenfilename()*. Ovaj prozor omogućava korisniku da izabere sliku sa ekstenzijama .jpg, .jpeg, .png ili .gif. Ako korisnik ne izabere nijednu sliku, program ispisuje poruku "No photo selected" i izlazi iz funkcije, čime se sprečava dalja obrada. Ako je slika uspešno odabrana, njen put se štampa u konzoli, omogućavajući korisniku da potvrdi svoj izbor. Nakon odabira slike, korisnik zatim dobija mogućnost da izabere tekstualni dokument na sličan način. Ovaj dijalog takođe koristi *askopenfilename()*, ali je filtriran da dozvoli samo .txt fajlove. Ako korisnik ne izabere dokument, ponovo se ispisuje poruka i izlazi iz funkcije. Kada je dokument uspešno odabran, njegov put se takođe štampa, a sadržaj se čita putem funkcije *read_text_file()*, čime se priprema za umetanje u sliku.

```
root = tk.Tk()
root.withdraw()

photo_path = filedialog.askopenfilename(
    title="Select a Photo",
    filetypes=[("Image Files", "*.jpg *.jpeg *.png *.gif")]
)

if not photo_path:
    print("No photo selected")
    return

print(f"Selected photo: {photo_path}")

text_path = filedialog.askopenfilename(
    title="Select a Text Document",
    filetypes=[("Text Files", "*.txt")]
)

if not text_path:
    print("No text document selected")
    return

print(f"Selected text document: {text_path}")

text = read_text_file(text_path)
```

Funkcija *read_text_file(text_path)* ima za cilj da učita sadržaj tekstualnog fajla čija putanja se prosleđuje kao argument. Kada se funkcija pozove, koristi se kontekstni menadžer with, što

obezbeđuje da se fajl automatski zatvori nakon što se završi sa njegovim korišćenjem, čime se izbegava mogućnost curenja resursa.

Unutar *with()* bloka, fajl se otvara u režimu čitanja ('r'), što znači da se može pristupiti samo njegovom sadržaju. Nakon otvaranja, funkcija koristi *file.read()* da pročita ceo sadržaj fajla i vrati ga kao string. Ovo omogućava da se sadržaj fajla lako koristi u ostatku aplikacije, na primer, za umetanje u sliku prilikom korišćenja PVD algoritma.

Ova funkcija je jednostavna, ali efikasna, jer omogućava lako i sigurno učitavanje tekstualnih podataka bez potrebe za dodatnim koracima za upravljanje fajlovima. U kontekstu aplikacije, ona igra ključnu ulogu, jer omogućava korisnicima da unesu tekst koji će biti skriven u slikama, čime se omogućava glatka interakcija i funkcionalnost.

```
def read_text_file(text_path):
    with open(text_path, 'r') as file:
        return file.read()
```

S druge strane, kod za opciju "Extract Text from Image" omogućava korisniku da izabere sliku iz koje će se izvući skriveni tekst. Kada korisnik odabere ovu opciju, takođe se pokreće skriveni prozor Tkinter-a, a zatim se otvara dijalog prozor za odabir slike. Ovaj prozor pruža iste mogućnosti odabira kao prethodni, filtrirajući slike sa ekstenzijama .jpg, .jpeg, .png ili .gif. Ako korisnik ne izabere sliku, ispisuje se poruka "No photo selected" i funkcija se završava, sprečavajući dalju obradu. Ukoliko je slika uspešno odabrana, njen put se prikazuje u konzoli. Nakon toga, poziva se funkcija *decode_text_from_image()*, koja preuzima odabranu sliku i izvodi proces dekodiranja kako bi se dobio skriveni tekst. Ovaj deo aplikacije omogućava korisnicima intuitivan i jednostavan način za interakciju sa slikama, olakšavajući obavljanje željenih operacija sa minimalnim naporom i rizikom od greške. Celi flow je osmišljen tako da korisnicima pruži jasne upute i povratne informacije, čime se povećava efikasnost i zadovoljstvo prilikom korišćenja aplikacije.

```
root = tk.Tk()
root.withdraw()

photo_path = filedialog.askopenfilename(
    title="Select a Photo with Hidden Text",
    filetypes=[("Image Files", "*.jpg *.jpeg *.png *.gif")]
)

if not photo_path:
    print("No photo selected")
    return

print(f"Selected photo: {photo_path}")
```

## Ubacivanje teksta u sliku

Funkcija *encode_text_in_image(image_path, text, output_path)* implementira proces umetanja teksta u sliku koristeći PVD (Pixel Value Differencing) algoritam. Ovo se postiže modifikacijom najsitnijih vrednosti piksela slike kako bi se prikrila binarna reprezentacija teksta.

Prvo, funkcija otvara sliku koristeći *Image.open(image_path)*. Ova funkcija iz Pillow biblioteke učitava sliku iz datog putanje i vraća objekat slike. Ovaj objekat omogućava pristup različitim metodama i atributima slike, kao što su veličina, formati i pikseli. U ovom slučaju, cilj je da se slika konvertuje u niz piksela kako bismo mogli da manipulišemo vrednostima boja.

Nakon što je slika otvorena, konvertuje se u numpy niz koristeći *np.array(image, dtype=np.uint8)*. Ova konverzija transformiše sliku u trodimenzionalni niz gde su dimenzije slike predstavljene redovima i kolonama piksela, a svaki piksel se sastoji od tri komponente (R, G, B). Tip *np.uint8* označava da su vrednosti piksela celobrojne i da se kreću od 0 do 255.

```
def encode_text_in_image(image_path, text, output_path):
    image = Image.open(image_path)
    pixels = np.array(image, dtype=np.uint8)

    print("Pixels:", pixels)
```

Pixels: [[[255 255 255]

[255 255 255]

[255 255 255]

...

Sledeći korak je konverzija teksta u njegovu binarnu formu. Ovaj proces se ostvaruje kroz listu razumevanja: *binary_text = ''.join(format(ord(char), '08b') for char in text) + '00000000'*. Ovde, *ord(char)* vraća ASCII vrednost karaktera, koja se zatim pretvara u binarni oblik sa *format(..., '08b')*, generišući osmobitnu reprezentaciju. Na kraju se dodaje oznaka *00000000*, koja označava kraj poruke.

Nakon što je binarni tekst pripremljen, funkcija proverava koliko piksela je dostupno za umetanje teksta. Ovo se izračunava sa *num_pixels = pixels.size // 3*, gde *pixels.size* daje ukupan broj elemenata u nizu. Deljenjem sa 3 se dobija broj piksela, jer svaki piksel sadrži tri vrednosti (R, G, B). Ako je dužina binarnog teksta veća od dostupnog broja piksela, funkcija baca grešku, čime se osigurava da se tekst ne može umetnuti.

```
binary_text = ''.join(format(ord(char), '08b') for char in text) + '00000000'

num_pixels = pixels.size // 3
if len(binary_text) > num_pixels * 3:
    raise ValueError("Text is too long to hide in the image.")
```

Kada su provere završene, funkcija koristi tri ugnježdene *for* petlje kako bi pristupila svakom pikselu slike. Prva petlja prolazi kroz redove piksela, druga kroz kolone, a treća kroz komponente boje piksela. Unutar treće petlje, ako je *idx* veći od dužine binarnog teksta, petlja se prekida.

Za svaki piksel, originalna vrednost se čuva u *pixel_val*, a bit koji se treba umetnuti se dobija iz *binary_text[idx]*. Zatim se koristi izraz *(int(pixel_val) & ~1) | bit* za modifikaciju vrednosti piksela. Ovaj izraz funkcioniše tako što prvo postavlja poslednji bit *pixel_val* na 0 koristeći *& ~1*, a zatim dodaje bit koji želimo da umetnemo. Ovo smanjuje vizuelni uticaj na sliku, jer se menja samo poslednji bit.

Ako nova vrednost piksela nije validna (manja od 0 ili veća od 255), ispisuje se upozorenje. Nakon umetanja, modifikovani niz piksela se konvertuje nazad u sliku koristeći *Image.fromarray(pixels)* i čuva se na zadatoj putanji *output_path* sa *encoded_image.save(output_path)*.

```
idx = 0
for i in range(pixels.shape[0]):
    for j in range(pixels.shape[1]):
        if idx >= len(binary_text):
            break
        for k in range(3):
            if idx >= len(binary_text):
                break
            pixel_val = pixels[i, j, k]
            bit = int(binary_text[idx])

            print(f"Original pixel value: {pixel_val}, Bit to encode: {bit}")

            print('--------',pixel_val, bit)
            new_pixel_val = (int(pixel_val) & ~1) | bit
            if new_pixel_val < 0 or new_pixel_val > 255:
                print(f"Invalid pixel value: {new_pixel_val}")

            pixels[i, j, k] = new_pixel_val
            idx += 1
```

Original pixel value: 255, Bit to encode: 0

-------- 255 0

Original pixel value: 255, Bit to encode: 1

-------- 255 1

Original pixel value: 255, Bit to encode: 0

-------- 255 0

Original pixel value: 255, Bit to encode: 0

-------- 255 0

…

Na kraju, *encoded_image = Image.fromarray(pixels)* se koristi za kreiranje nove slike iz promenjenog niza piksela pixels. Funkcija *Image.fromarray()* iz Pillow biblioteke uzima numpy niz i konvertuje ga u sliku.

Nakon što je slika kreirana, linija *encoded_image.save(output_path)* se koristi za čuvanje nove slike na disku. Metoda *save()* uzima kao argument putanju *output_path*, gde će se slika sa skrivenim tekstom sačuvati. Ova funkcija automatski određuje format slike na osnovu ekstenzije datoteke u putanji (npr. .png, .jpg), što omogućava lako čuvanje slika u različitim formatima.

```
encoded_image = Image.fromarray(pixels)
encoded_image.save(output_path)
```

## Izvlačenje teksta iz slike

U funkciji *decode_text_from_image(image_path)*, proces takodje počinje linijama kao u funkciji *encode_text_in_image(image_path, text, output_path)*. Započinje se otvaranjem slike pomoću Image.open(image_path), a zatim se koristi *np.array(image, dtype=np.uint8)* za konvertovanje slike u numpy niz.

```
def decode_text_from_image(image_path):
    image = Image.open(image_path)
    pixels = np.array(image, dtype=np.uint8)
```

Sledeći korak je inicijalizacija praznog stringa *binary_text*, u koji će se smeštati binarni podaci skrivene poruke. Koristeći dvostruke petlje, funkcija iterira kroz sve piksele slike. Prva petlja prolazi kroz redove piksela, dok druga prolazi kroz kolone. Unutar ovih petlji, treća petlja iterira kroz tri komponente boje (RGB) svakog piksela. Za svaki piksel, funkcija koristi *pixel_val & 1* da dobije poslednji bit (najmanje značajan bit) RGB vrednosti piksela, koji nosi deo skrivene informacije. Ovaj bit se dodaje *binary_text*, formirajući tako niz koji predstavlja skrivenu poruku.

```
binary_text = ''
for i in range(pixels.shape[0]):
    for j in range(pixels.shape[1]):
        for k in range(3):
            pixel_val = pixels[i, j, k]
            binary_text += str(pixel_val & 1)
```

Kada su svi pikseli obrađeni, funkcija nastavlja sa kreiranjem liste karaktera *chars*. Svakih osam bitova iz *binary_text* se grupiše i konvertuje u ASCII karakter. Ovaj proces se vrši u petlji koja iterira kroz *binary_text* u koracima od osam. Ako se naiđe na niz od osam nula (*'00000000'*), što označava kraj skrivene poruke, proces se prekida, jer je to signal da više nema podataka za dekodiranje.

```
chars = []
for i in range(0, len(binary_text), 8):
    byte = binary_text[i:i+8]
    if byte == '00000000':
        break
    chars.append(chr(int(byte, 2)))
```

Nakon što su svi karakteri ekstrahovani, poziva se funkcija *is_ascii_list_valid(chars)*, koja proverava da li su svi karakteri validni ASCII znakovi. Ova funkcija iterira kroz listu karaktera i osigurava da svaki karakter bude tipa string, dužine 1 i da njegova ASCII vrednost ne prelazi 127. Ako bilo koji od ovih uslova nije ispunjen, funkcija vraća *False*.

```
def is_ascii_list_valid(char_list):
    for char in char_list:
        if not isinstance(char, str) or len(char) != 1 or ord(char) > 127:
            return False
    return True
```

Ako se utvrdi da neki karakter nije validan, korisniku se prikazuje upozorenje putem *messagebox.showwarning*, informišući ga da slika ne sadrži validan skriveni tekst. U ovom slučaju, funkcija se završava bez vraćanja rezultata. Ukoliko su svi karakteri validni, funkcija konačno vraća spojeni string skrivenog teksta, omogućavajući korisnicima da dobiju informacije koje su prvobitno skrivene unutar slike.

```
isValidText = is_ascii_list_valid(chars)

if not isValidText:
    messagebox.showwarning("Invalid Text", "The image does not contain valid hidden text.")
    return

return ''.join(chars)
```

# Zaključak

U ovom seminarskom radu detaljno sam obradio različite aspekte PVD (Pixel Value Differencing) algoritma, koji se koristi za skrivanje i izvlačenje teksta iz slika. Posebnu pažnju posvetio sam praktičnoj implementaciji ovog algoritma koristeći Python i Pillow biblioteku, što omogućava lako rukovanje slikama i manipulaciju pixelima.

Prvo sam se fokusirao na teorijske osnove algoritma, objašnjavajući kako se podaci mogu kodirati unutar slika bez značajnog narušavanja kvaliteta slike. Onda sam prešao na praktične aspekte, opisujući proces odabira slika i tekstualnih dokumenata, kao i kako se vrši kodiranje i dekodiranje informacija. U ovom okviru, posebno sam istakao važnost pravilnog rukovanja pixel vrednostima, što je ključno za očuvanje integriteta originalne slike.

Praktična primena PVD algoritma otvara vrata za širok spektar mogućnosti. Na primer, u oblasti digitalne forenzike, ovaj metod može poslužiti za skrivene poruke ili zaštitu autorskih prava, omogućavajući autorima da zaštite svoje radove na diskretan način. U umetnosti, umetnici mogu koristiti ovaj pristup da unesu dodatne poruke ili značaje u svoja dela, čime se dodaje složenost i dubina njihovim radovima.

Takođe, realno korišćenje ovog algoritma može se primeniti u raznim industrijama, uključujući bezbednost informacija i marketing. Mogućnost skrivenog kodiranja podataka unutar slika može biti od koristi za zaštitu poverljivih informacija ili čak za izgradnju interaktivnih marketinških kampanja, gde potrošači mogu otkrivati skrivene poruke ili nagrade unutar vizuelnog sadržaja.

# Literatura

https://www.kaspersky.com/resource-center/definitions/what-is-steganography

https://www.simplilearn.com/what-is-steganography-article

https://www.researchgate.net/publication/357299988_Steganography_and_Cryptography_A_Systematic_Review

https://www.ijrar.org/papers/IJRAR23B2960.pdf

https://www.researchgate.net/publication/345146031_An_Improved_Image_Steganography_Algorithm_Based_on_PVD