Факултет инжењерских наука Универзитета у Крагујевцу

Назив студијског програма: Рачунарска техника и софтверско инжењерство
Ниво студија: Основне академске студије
Предмет: Заштита података (Технике криптовања)
Број индекса: 576/2016

Никола С. Радовановић

Анализа и имплементација различитих стеганографских алгоритама у дигиталним сликама

Дипломски рад

Комисија за преглед и одбрану:

1. **Доц. Др. Владимир Миловановић - ментор**
2. _____
3. _____

Датум одбране:_____

Оцена:_____

У оквиру овог дипломског рада кандидат треба да анализира различите стеганографске алгоритме, односно алгоритме за скривање произвољних дигиталних података унутар дигиталних слика које су коришћене као носиоци. Биће дат преглед техника које раде на бази метаподатака, рада у просторном домену, спектралном домену као и технике засноване на машинском и дубоком учењу. Главни акценат биће стављен на имплементацију алгоритма у просторном домену који унутар битова најмање тежине кодује скривене информације мењајући слику носиоца што је могуће мање. Метрике за поређење оригиналне и прерађене слике биће изложене, као и употребљене за квантификацију квалитета алгоритма. Анализа утицаја различитих параметара алгоритма као што су број пиксела који се користе, просторна позиција пиксела и број измењених бита најмање тежине ће бити приказани.

Препоручена литература:

[1] A. Cheddad, J. Condell, K. Curran, P. Mc Kevitt: Digital image steganography: Survey and analysis of current methods,Elsevier Signal Processing 90 727–752, 2010.

[2] M. Hussain, A. T.S. Ho, A. Wahid, K. Jung, Image steganography in spatial domain: A survey, Development of IoT Security Middleware Platform for Multimedia Security, 2018.

[3]M. Chaumont. Deep Learning in steganography and steganalysis from 2015 to 2018. M. Hassaballah. Digital Media Steganography: Principles, Algorithms, Advances, Elsevier, 2019.

Крагујевац, 18.6.2020.

Ментор:
Доц. Др. Владимир Миловановић

_____

Садржај

Резиме

Стеганографија је наука и вештина скривања поверљивог садржаја унутар другог садржаја тако да поверљиви садржај буде невидљив свима осим странама које комуницирају. Рад ће анализирати различите стеганографске алгоритме у дигиталним сликама, односно алгоритме за скривање података унутар дигиталних слика које су коришћене као носиоци. Биће дат преглед техника које раде на бази метаподатака, рада у просторном и фреквенцијском домену као и техника заснованих на дубоком учењу. Главни акценат биће стављен на имплементацију алгоритама који раде у просторном домену слике. Алгоритми ће бити детаљно објашњени и упоређени по различитим метрикама.

Кључне речи: стеганографија, слика носилац, стего-слика, просторни домен, фреквенцијски домен, капацитет, сигурност, LSB, PVD, EMD.

Abstract

Steganography is the science and skill of hiding confidential content within other content so that the confidential content stays undetectable to anyone else except the communicating parties. This work will cover various steganographic algorithms in digital images, that is the algorithms for hiding digital data inside digital images that are used as covers. An overview of techniques that are based on working with metadata, working in spatial and frequency domain as well as techniques based on deep learning will be given. The main emphasis will be placed on implementing the algorithms that work in spatial domain of an image. The algorithms will be explained in detail and compared by different metrics.

Keywords: steganography, cover image, stego-image, spatial domain, frequency domain, capacity, security, LSB, PVD, EMD.

1. Увод

Све већим напретком технологије у подручју комуникација и информатике, долази до широке употребе рачунара у свим пољима друштвеног живота. Постоје разне могућности дистрибуције и обраде дигиталног садржаја попут електронске поште, аудио записа, слика и видеа. У оваквим условима, присутан је проблем сигурности информација јер се информације преносе путем разних канала и медијума у дигиталном облику и лако се могу пронаћи, видети и преузети од стране лица којима те информације нису намењене. У сврху заштите података од нелегалног приступа могу се користити криптографски алгоритми који шифрирају податке и стеганографски алгоритми који скривају њихово постојање.

Стеганографија је наука и вештина скривања поверљивог садржаја унутар другог садржаја тако да поверљиви садржај буде невидљив свима осим странама које комуницирају. Једнако као и криптографија, стеганографија се користи да би се сачувала тајност података. Суштинска разлика је у томе што криптографија шифрира тајну поруку и мења је тако да се она не може разумети, док стеганографија скрива њено постојање тако да је у најбољем случају комуникација између две стране невидљива. Стеганографски систем уграђује скривени садржај унутар неупадљивих медијума, како не би изазвао сумњу прислушкивача. Стеганографија потиче из грчких речи *steganos* (скривено, прикривено) и *graphein* (писати) тако да је буквално значење те речи „Скривено писање“.

У раду ће бити описан развој стеганографије кроз историју почевши од античких времена све до савремене стеганографије која данас користи у сврху заштите поверљивих дигиталних података. Рад је фокусиран на стеганографију у дигиталним сликама и биће дат преглед разних техника које раде у просторном и фреквенцијском домену слике носиоца информација, као и техника заснованим на дубоком учењу. Такође, детаљно ће бити објашњен начин имплементације неколико алгоритама који раде у просторном домену у програмском језику Пајтон. Алгоритми ће на крају бити упоређени по различитим метрикама чији ће начин израчунавања такође бити објашњен.

2. Стеганографија кроз историју

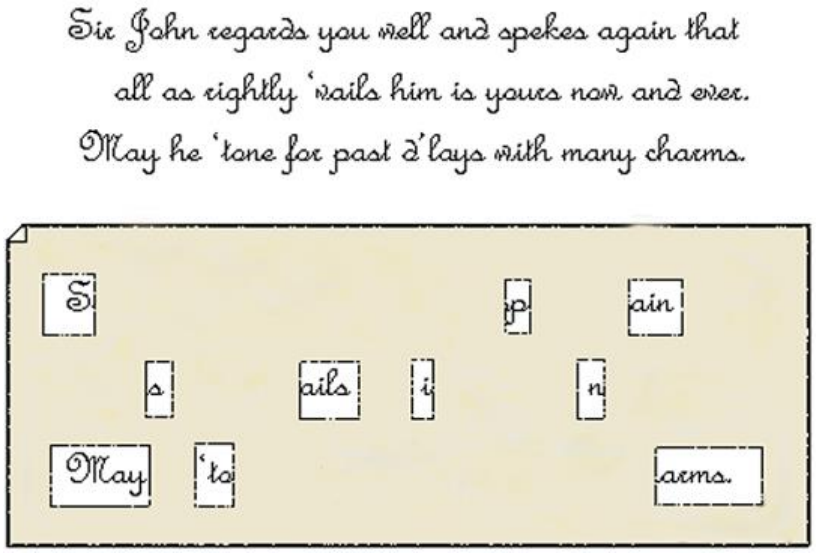
Стеганографија се користила у разним формама чак од 5. века п.н.е [1,3]. Хистеј, који је био тиран Милета, обријао је главу гласнику и на њој написао поруку свом синовцу Аристагори у којој га је охрабрио да се побуни против персијског принца. Када је гласнику порасла коса, он га је послао Аристагори.

У 480. години п.н.е, грк Демарат је послао спартанцима поруку у којој их је упозорио о инвазији Ксеркса. Да би послао поруку, користио је таблице прекривене воском. Скинуо је восак са таблице и написао поруку на дрвету испод, а затим је таблицу поново прекрио воском тако да изгледа ново и неискоришћено.

Наска линије, огромни геолифи на Наска висоравни близу истоименог града у Перуу, такође се могу сматрати обликом стеганографије јер се слике не могу видети осим ако се не посматрају са велике висине.

Године 1499. Немачки опат Јоханес Тритемиус је написао књигу Steganographia и унутар ње описао како сакрити поруку унутар безазленог текста.

Пре 500 година, италијански математичар Ђироламо Кардано изумео је методу тајног писања која се састоји од следећих елемената: две стране у комуникацији имале су папирну маску са рупама, затим је пошиљалац поставио маску на папир и кроз рупе написао тајну поруку на папир, затим се маска склони и остатак папира се допуни текстом тако да се тајна порука сакрије. Ова метода је по Кардану названа Карданова решетка.



Слика 1, пример карданове решетке

Честа метода за прикривање комуникације је била коришћење невидљивог мастила. Ова метода је била највише коришћена током Другог светског рата, где се на наизглед потпуно невиним писмима невидљивим мастилом додавала скривена порука. Осим ове, позната метода скривања порука је била камуфлирање тајне поруке у мноштво наизглед обичног текста где се права порука добијала ако се прочитају само одређена слова. Оваква метода се назива метода нулте шифре и коришћена је највише од стране немачких шпијуна

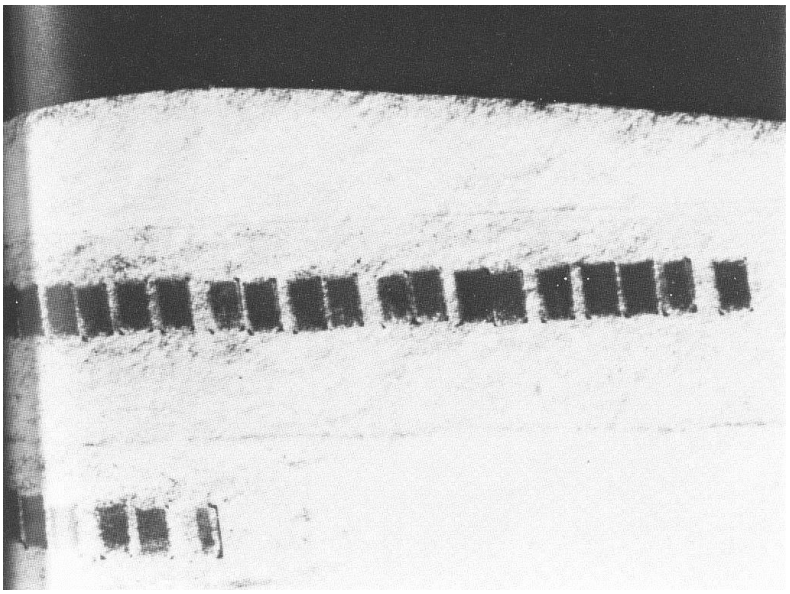
током Другог светског рата. Једна позната порука послата од стране немачког шпијуна изгледала је:

> *Apparently neutral's protest is thoroughly discounted and ignored. Isman hard hit. Blockade issue affects pretext for embargo on by-products, ejecting suets and vegetable oils.*

Читањем другог слова сваке речи добије се порука:

Pershing sails from NY June I

Нацисти су током Другог светског рата развили методу микротачака. Микротачке су фотографије величине тачке на писаћој машини где су се на једној од њих могле наћи странице и странце информација. Поруке су могле бити прочитане само ако се тачка ставила под уређај са великим увеличавањем (обично 200 пута и више).



Слика 2, микротачке залепљене на папир

3. Модерна стеганографија

Тренутно је нагласак стављен на различите облике дигиталне стеганографије. Порастом моћи компјутера и развојем дигиталне обраде сигнала, теорије информација и теорије кодирања, стеганографија се „дигитализовала". Три битне карактеристике сваког система који се бави скривањем података су: капацитет, сигурност и робусност. Капацитет представља количину података која се може сакрити унутар медијума, сигурност представља неспособност прислушкивача да детектује скривену информацију, а робусност

представља количину модификација коју медијум може да истрпи пре него што се оштете скривене информације.

Стеганографија стреми ка високој сигурности и капацитету што узрокује тиме да су скривене информације прилично рањиве, чак и тривијалне модификације на медијуму могу да униште информацију.

Да би стеганографија била успешна, потребно је прикрити само постојање тајне поруке, јер ако прислушкивач зна да се она налази унутар медијума и познаје технику којом је тајна порука убачена може је прочитати. Чак и ако су подаци криптовани, ако неко посумња на коришћење стеганографије може вишеструком конверзијом медијума уништити тајне податке.

Једна варијација стеганографије која се користи у илегалне сврхе је Тројански коњ. Тројански коњи су коришћени од стране нападача да убаце вирусе и друге видове малициозног софтвера унутар неке организације или корисничког рачунара. Функционише тако што се малициозни софтвер сакрије унутар другог софтвера који корисник види тако да док корисник мисли да инсталира неку жељену апликацију, у позадини малициозни софтвер може уништити корисничке датотеке или пружити нападачу приступ свим подацима корисника. Тројански коњи се разликују од традиционалне стеганографије по томе што носећи медијуми могу искључиво бити извршне датотеке које инсталирају малициозни софтвер у кориснички рачунар. Друга разлика је у начину комуникације. Код традиционалне стеганографије две стране учествују у комуникацији и свесне су скривених порука унутар медијума, док код Тројанског коња једино је нападач свестан тајних података, а жртва несвесно инсталира малициозни софтвер на свој рачунар док мисли да инсталира нешто што жели.

4. Стеганографске технике

У стеганографији постоје три основне технике које се могу користити за скривање информација у датотекама и то су: убацивање, замена и генерисање [4].

- Убацивање: техника која омогућава скривање података у деловима датотека који су од најмањег значаја за злонамерног корисника. На пример, већина датотека садржи EOF (end-of-file) ознаку. При преслушавању звучног записа, апликација која репродукује запис престаје када наиђе на EOF. Могуће је убацити податке након EOF ознаке тако да они немају никаквог ефекта на звучни запис. Једини проблем је што се величина датотеке може знатно повећати што може пробудити сумњу прислушкивача.

- Замена: техника која се заснива на замени најмање значајних информација оригиналног носиоца информација и мења их тајним подацима. На пример, код звучних записа, свака јединица звука се састоји од више бајтова. Мењањем најмање значајног бита унутар једног оваквог бајта битовима поруке, звук ће бити модификован али тако да људско ухо неће моћи да препозна разлику.

- Генерисање: техника која на основу тајних података генерише нову датотеку тако да код ове технике уопште није потребна оригинална датотека која носи информације. Предност коришћења ове технике је то што као резултат даје нову датотеку која је имуна на поређење са другим датотекама, што код претходне две технике није случај јер би злонамерни корисник поређењем оригиналне датотеке и датотеке са тајним информацијама открио разлике.

5. Стеганографске методе у дигиталним сликама

Стеганографија користи ограничене способности људског визуелног система. Било која датотека може бити сакривена одређеном методом унутар носиоца податка, а да то не буде детектовано људским оком. Најпознатији формати слика на интернету су GIF (Graphics interchange format), JPEG (Joint Photographic Experts Group) и PNG (Portable Network Graphics). Већина техника развијено је са циљем да се искористе структуре ових формата [2].

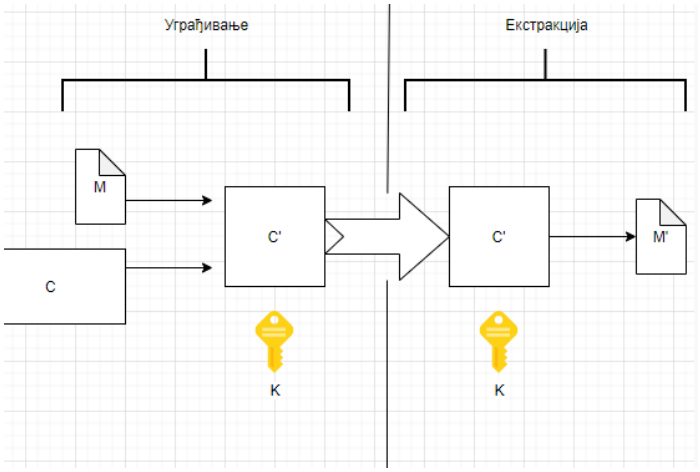
Процес уграђивања се одвија на следећи начин (што је приказано графички на слици број 3):

Нека C означава носиоца информација, на пример слику у коју се уграђују тајни подаци. Нека је K кључ који се користи за опционо криптовање поруке, а M тајна порука, на пример датотека коју треба послати. Дакле процес уграђивања (Em) изгледа:

Em: C⊕K⊕M->C’

Процес екстракције односно добијања података када порука стигне на одредиште приказује се на следећи начин :

Ex(Em(C,K,M))=M’



Слика 3, процес уграђивања и екстракције

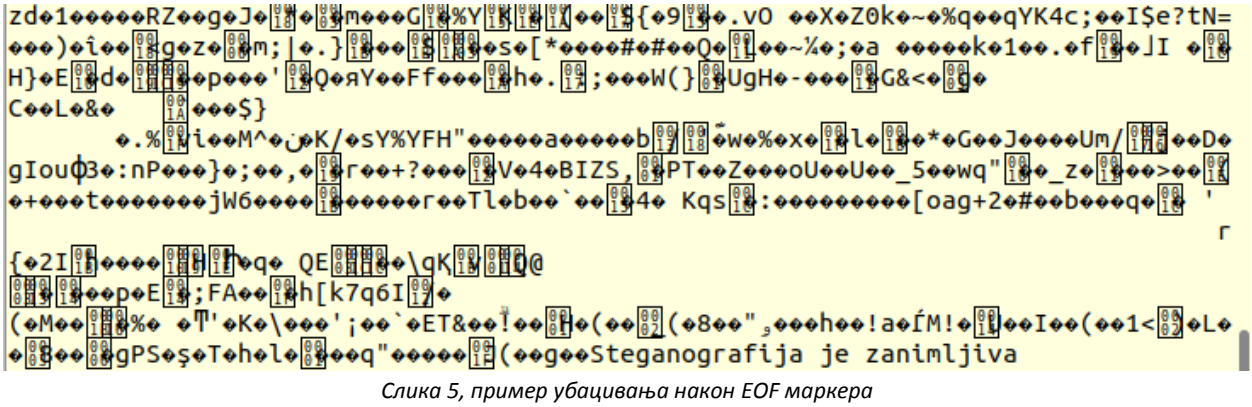
5.1 Стеганографија искоришћавањем формата слика

Коришћењем технике убацивања могуће је сакрити тајну поруку у подручја датотеке која се не мењају у процесима обраде. Готово сваки тип датотеке, укључујући и дигиталне фотографије садржи EOF (end-of-file) маркер. Убацивањем података након овог маркера ефективан садржај оригиналне датотеке се не мења. На Linux оперативном систему ово је могуће постићи једноставном командом у терминалу: cat message.txt >> image.jpg. Ова команда садржај датотеке message.txt додаје у датотеку image.jpg. Команда cat приказује садржај датотеке message.txt, а помоћу >> садржај датотеке се преусмерава на крај датотеке image.jpg, односно иза њеног EOF маркера.

Када се слика отвори помоћу било ког програма за приказивање слика, њен садржај се приказује, а све што је иза EOF маркера бива игнорисано.

Слика 4, слика са садржајем убаченим након EOF маркера

Међутим када се слика отвори помоћу било ког текстуалног едитора, наша тајна порука је приказана након мноштва нечитљивих података који представљају слику (слика 5).



Слика 5, пример убацивања након EOF маркера

Ове промене не могу бити детектоване ни људским визуелним системом ни поређењем оригиналне слике и слике са подацима помоћу хистограма пошто овакав начин уметања поруке не мења ни квалитет ни садржај оригиналне слике.

Још један начин да се тајна порука сакрије унутар слике а да не мења њен садржај и квалитет је убацивање података у заглавље слике, односно простор у коме се налазе разне информације о датотеци као што су марка и модел апарата који је направио слику, датум њеног настанка, величина отвора бленде, експозиција и осетљивост слике (ISO вредност) [2]. Ово су метаподаци, односно подаци о подацима и чувају се у заглављу датотеке.

Чињеницу да су у слици скривени подаци може одати величина датотеке која, у зависности од величине тајне поруке, може знатно да се разликује од величине оригиналне слике.

Слика 6, пример убацивања података у заглавље слике

5.2 Стеганографија у просторном домену слике

Методе које раде у просторном домену заснивају се на модификацији носиоца информација у просторном домену, што укључује енкодирање тајних података унутар најмање значајних битова у оквиру носиоца информација. Ове методе се служе техником замене и мењају мање битне битове унутар датотеке која носи информације. Коришћењем неке од метода које раде у просторном домену увек је потребно наћи компромис између капацитета, односно количине тајних података које је могуће пренети и очувања квалитета слике носиоца.

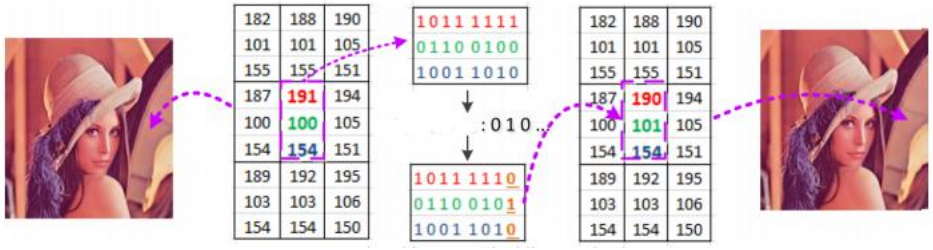
Мењањем што више битова слике носиоца могуће је пренети више информација, али долази до што већег нарушавања квалитета слике и појаве дисторзије која у зависности од броја битова који се користе за енкодирање тајних информација може постати и видљива људском оку.

5.2.1 Методе базиране на LSB

Основна метода која ради у просторном домену је LSB (Least significant bit) метода и заснива се на једноставној замени битова слике битовима поруке. Битови се могу мењати у пикселима слике секвенцијално или може постојати неки алгоритам који ће бирати унутар којих пиксела ће се смештати подаци. Број битова најмање тежине у пикселу који се мењају битовима поруке утицаће на капацитет и квалитет слике па је потребно наћи баланс између ова два параметра [5].

Дигитална слика у боји представљена је дводимензионим низом пиксела. Сваки пиксел је представљен помоћу 3 вредности у опсегу [0, 255] које представљају интензитет

црвене, зелене и плаве боје. Помоћу ове методе, битови тајне поруке се уграђују у све три компоненте пиксела. На пример, ако пиксел у слици која ће носити тајну поруку има вредности интензитета боја [125, 150, 214], што је у бинарном систему: [01111101, 10010110, 11010110], а битови тајног податка који се уграђује 101101 и супституција се врши на 2 бита најмање тежине, добијају се нове вредности интензитета боја пиксела [01111**10**, 10010**11**, 11010**01**] односно [126, 151, 213]. Илустрација примера са другим вредностима и где се уграђивање врши на једном биту најмање тежине дат је на слици 7.



Слика 7, илустрација LSB методе

Током времена појавиле су се разне методе које су настале као варијација LSB методе. Ове методе су настале са циљем да се оптимизује капацитет, а побољша визуелни квалитет слике носиоца информација и смањи могућност да се стеганографија открије поступком стеганализе.

Неке од ових метода су адаптивне LSB методе засноване на ивицама, текстури и осветљењу слике и на основу ових параметара се одређује број битова у пикселу који се користе за чување битова тајног податка.

Код најједноставније LSB методе код сваког пиксела семења само бит најмање тежине једним битом поруке. На овај начин пиксели са парним вредностима су или увећани за 1 или остају исти, док се пиксели са непарним вредностима или смање за 1 или остају исти.

Као резултат, сваке две узастопне вредности на хистограму вредности пиксела конвергираће ка истој вредности што је лако да се открије стеганализом хистограма. Као одговор на очигледне слабости основне методе, појавило се побољшање LSB методе звано LSBM (LSB matching). У овој методи, у циљу превазилажења проблема упарених вредности на хистограму, када се бит тајног податка и бит најмање тежине на слици носиоцу информација не поклапају, вредност пиксела се насумично повећа или смањи за 1. Сваки бит улазног тајног податка може се поклопити са битом најмање тежине слике носиоца са вероватноћом 0.5. Неколико метода је настало са циљем да се ова вероватноћа смањи и то се постиже тако што се пиксели групишу у блокове одређене величине.

Други приступ одолевању стеганализи хистограма је метода LSB+ која ради тако што се у слику носиоца додају још неки додатни битови како би хистограм стего слике изгледао што сличније хистограму оригиналне слике. Пошто ова метода такође утиче на статистичке и перцептивне особине слике, настало је побољшање ове методе где се идентификују осетљиви пиксели и у њих се не убацују додатни битови. Ова метода је названа LSB++ и одолева статистичким нападима а одржава сличност хистограма са оригиналном сликом што је постигнуто LSB+ методом.

Новија метода LSB WH (LSB word-hunt) је метода инспирисана осмосмерком која је настала са циљем да се смањи број модификација по пикселу слике и тако постигне бољи квалитет стего слике. Ова метода број модификација по пикселу смањује на 0.315.

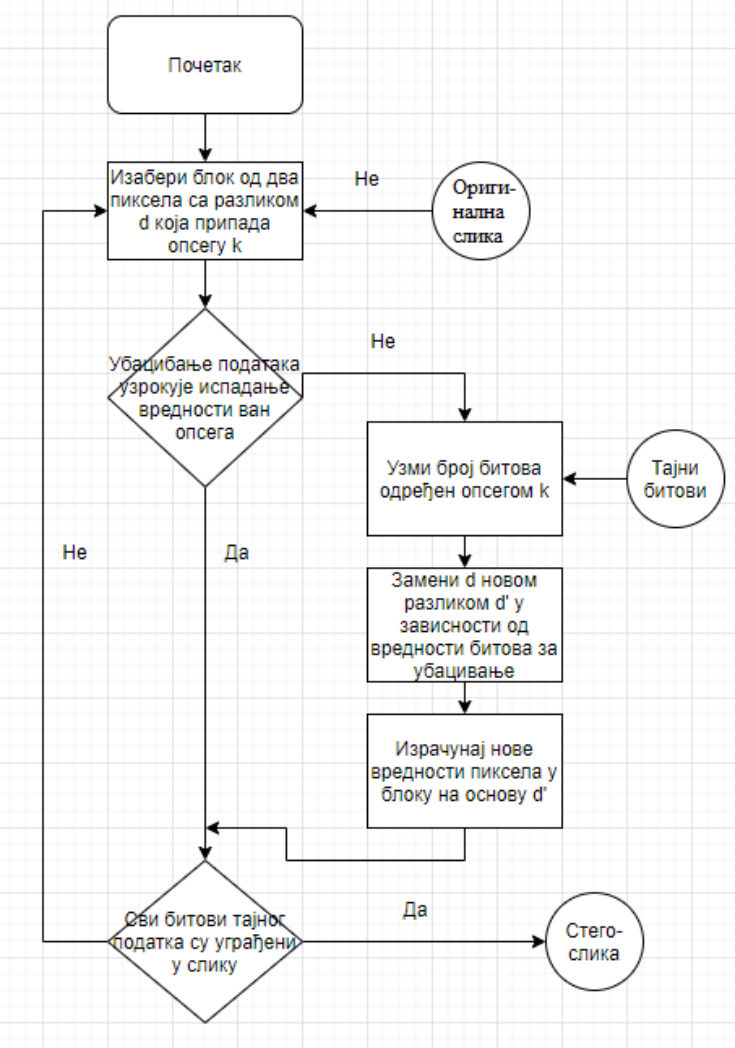
Разне стеганографске методе користе алгоритме енкрипције како би још боље заштитиле тајну информацију пре њеног убацивања у слику. Адаптивна LSB метода, где се користи насумично k као број битова који ће бити убачен у пиксел. Слика у којој ће се сместити тајни подаци се дели на непреклапајуће блокове једнаке величине и енкриптовани тајни подаци се убацују у сваки блок током четири насумичне шетње. Најбоља случајна шетња која обезбеђује најмању деградацију блока унутар слике се памти и фиксира за тај блок. Одлука за сваки блок се чува и све заједно се памте као тајни кључ. За тајне податке и стего кључ могу се користити различити нивои енкрипције.

Остале адаптивне методе укључују детекцију ивица и убацивање више битова тајне поруке унутар ивичних регија на слици. Ове методе обезбеђују већи капацитет, а укупна дисторзија слике се смањује. Могуће је и коришћење детектора комплексних региона на слици у које ће се убацивати више битова тајних података.

5.2.2 Методе базиране на PVD

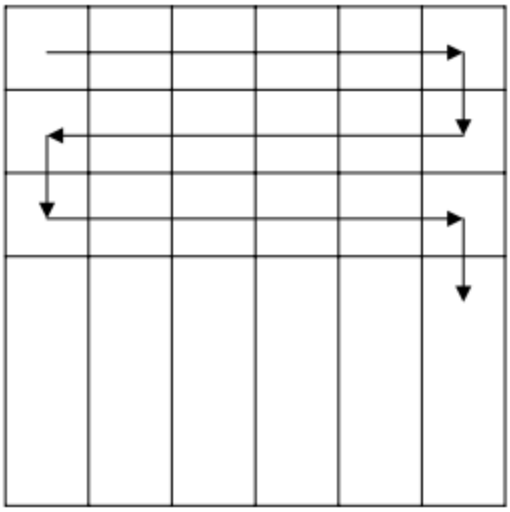
Иако су методе засноване на LSB једноставне за имплементацију и веома флексибилне у виду могућности комбиновања са осталим методама, њихов главни недостатак је директна повезаност капацитета и визуелног квалитета стего-слике. Повећање броја битова најмање тежине унутар пиксела у које се убацују тајни подаци у одређеној мери нарушава квалитет слике.

Као одговор на овај проблем појавиле су се методе које раде на бази разлике вредности пиксела (PVD – Pixel Value Differencing) [6]. У овој методи се користи разлика вредности два суседна пиксела да би се донела одлука о броју битова који ће се користити за убацивање података. Слика се дели на блокове једнаке величине и сваки блок садржи два суседна пиксела. Пиксели се у блокове групишу цик-цак методом обиласка, односно обилазећи слику по редовима. Начин груписања пиксела у блокове приказан је на слици 9. Унутар сваког блока рачуна се разлика између два пиксела и на основу добијене вредности доноси се одлука о броју битова који ће се користити за убацивање података. Мања вредност разлике даје индикацију да је то глатка област, а већа вредност да је то ивична област. Идеја да се више битова убацује у ивичне области потиче од тога да људски визуелни систем мање детектује промене у ивичним областима него у глатким. Вредности разлике се групишу у опсеге и на основу тих опсега прави се табела за селекцију броја битова за убацивање података. Код сивих слика пиксели имају једну вредност, док се код слика у боји овај поступак понавља за све од три боја пиксела. Генерална шема процеса уграђивања тајног податка у слику приказана је на слици 8.



Слика 8, основни PVD алгоритам

За сваки блок суседних пиксела p_i и p_{i+1} рачуна се разлика d. Ако су вредности тих пиксела g_i и g_{i+1}, d се рачуна као $g_{i+1} - g_i$ што се може наћи у опсегу вредности [-255, 255]. Блок чија је вредност d близу нуле сматра се екстремно глатким блоком, а блок чија је вредност d близу вредности 255 или -255 сматра се екстремно ивичним блоком. Због симетрије разматрају се апсолутне вредности разлике d односно у опсегу [0, 255] и вредности се групишу у опсеге који су означени са R_i, где је $i = 1,2 \dots, n$. Ови опсези су означени индексима од 1 до n. Доња и горња граница опсега R_i су редом означене са l_i и u_i, где су $l_1 = 0$ и $u_n = 255$, односно први опсег почиње од вредности 0 а последњи се завршава вредношћу 255. Ширина опсега R_i једнака је $u_i - l_i + 1$. У разним методама базираним на PVD ширине опсега се могу бирати на разне начине, али ће у наставку бити објашњен алгоритам где су ширине опсега степени двојке. Оваква рестрикција олакшава убацивање бинарних података. Интервали се бирају на основу перцепције људског визуелног система. Ширине опсега које представљају вредности глатких блокова су изабране тако да буду мање од ширина опсега које представљају ивичне блокове. На тај начин се постиже мање приметно уграђивање битова.



Слика 9, начин формирања блокова

Све вредности унутар једног опсега се сматрају веома сличним. Када се замени било која вредност разлике неком другом из истог опсега, сматра се да се та промена не може приметити људским оком. Ова метода ради баш тако да када се подаци уметну у пиксел разлика између два пиксела у блоку се мења у другу разлику која припада истом опсегу као разлика пре уметања битова тајног податка. Другим речима, када се битови тајног податка убацују на овај начин у пикселе унутар блока разлика између та два пиксела се мења неприметно људском оку.

Тајни податак односно тајна порука може бити било која датотека представљена низом битова. Циљ је убацити све битове тајног податка унутар блокова пиксела на слици која ће носити скривени податак. Број битова који ће бити уметнут у сваки блок зависи од ширине опсега којој припада разлика између пиксела у блоку.

За неки блок од два пиксела B са индексом k и разликом између пиксела d, број битова који могу бити убачени у блок, означено са n, рачуна се као $n = \log_2(u_k - l_k + 1)$. Пошто је ширина сваког блока степен двојке, n ће бити цео број. Подскуп тајне поруке C од n битова се убацује у блок B. Нова вредност разлике се рачуна на следећи начин:

$$d' = \begin{cases} l_k + b, & d \geq 0 \\ -(l_k + b), & d < 0 \end{cases} \quad (1)$$

Где је b децимална вредност битова у подскупу тајних података C. Вредност b је у опсегу $(0,\ u_k - l_k)$, а самим тим вредност d' је у опсегу $(l_k,\ u_k)$. Мењање d са d' резултује у промену која се не може приметити људским оком. Затим се b убацује у пиксел инверзном калкулацијом тако да се добију нове вредности (g'_i, g'_{i+1}) пиксела у блоку у стего слике (p'_i, p'_{i+1}). Процес се завршава када се сви битови тајног податка угради у слику.

Израчунавање нових вредности (g'_i, g'_{i+1}) се одвија на следећи начин:

$$f(g_i, g_{i+1}) = \begin{cases} (g_i - ceiling_m,\ g_{i+1} + floor_m), & \text{ако је } d \text{ непаран} \\ (g_i - floor_m,\ g_{i+1} + ceiling_m), & \text{ако је } d \text{ паран} \end{cases} \quad (2)$$

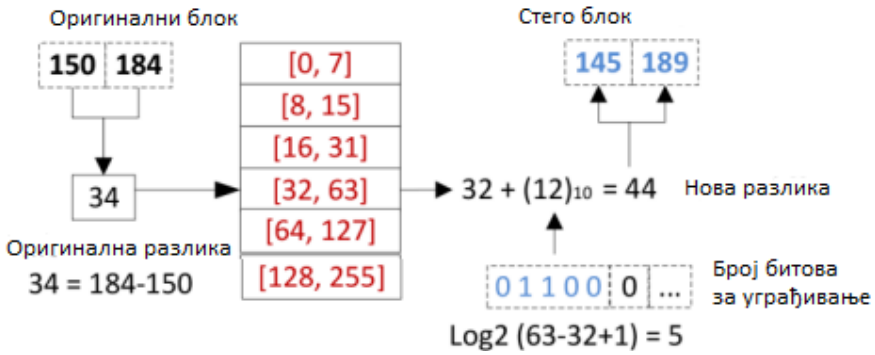
Где је $m = d' - d$, $ceiling_m = \lceil \frac{m}{2} \rceil$, а $floor_m = \lfloor \frac{m}{2} \rfloor$. Једначина изнад задовољава особину да је разлика између g'_{i+1} и g'_i једнака d'. Једначином је такође омогућено да се постигне што мања укупна дисторзија у блоку, па је и самим тим промена мање видљива људском оку.

У једначини изнад, мања вредност d' узрокује мањи интервал између g'_i и g'_{i+1}, док њена већа вредност узрокује већи интервал између ових вредности. Због тога је могуће полазећи од g_i и g_{i+1} добити невалидне вредности за g'_i и g'_{i+1}. Односно, применом једначине је могуће добити вредности за g'_i и g'_{i+1} ван опсега [0, 255].

Иако је могуће додатно подесити вредности тако да припадају опсегу [0, 255] тиме што се вредности фиксирају на неку од граничних вредности, ово може произвести велике дисторзије на слици. Да би се овај проблем решио, потребно је уврстити проверу која ће детековати вредности које узрокују испадање ван опсега и овакве блокове потпуно избећи при процесу уграђивања тајних података. Блокови стего-слике ових прескочених блокова биће једнаки блоковима на оригиналној слици. До ове појаве при примени овог алгоритма долази се веома ретко.

Провера проблематичних вредности почиње израчунавањем вредности ($\hat{g}_i$, $\widehat{g_{i+1}}$) помоћу формуле 2 за $m = u_k$ што обезбеђује да вредности ($\hat{g}_i$, $\widehat{g_{i+1}}$) узрокују највећу разлику у том блоку. Ако било која од ове две вредности испада из опсега [0, 255] блок са пикселима (p_i, p_{i+1}) се прескаче при убацивању тајних података.

Генерална шема уметања тајних битова у пикселе приказана је на слици 10.



Слика 10, уграђивање битова у блок

Нека су оригиналне вредности неке боје пиксела у блоку 150 и 184. Разлика између њих је 34 што припада опсегу [32, 63]. Ширина овог опсега је 63-32+1 = 32 = 2^5, што значи да се у овај блок може убацити 5 битова тајног податка. Нека је 5 тренутних битова тајног податка које је потребно урадити у слику 01100. Децимална вредност ових битова је 12 и додаје се доњој граници опсега да се добије нова вредност разлике 44. Коначно се нове вредности рачунају помоћу формуле 2 и добијају се нове вредности 145 и 189.

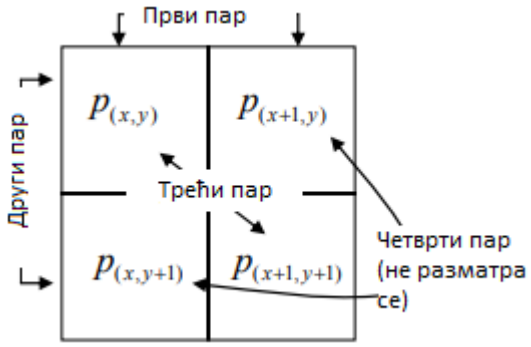
Процес екстракције битова тајног податка из стего-слике се одвија на следећи начин. Разлика између вредности пиксела (g_i^*, g_{i+1}^*) у блоку у стего-слици је $d^* = g_{i+1}^* - g_i^*$. Децимална вредност битова у овом блоку се рачуна на следећи начин:

$$b = \begin{cases} d^* - l_k, & d^* \geq 0 \\ -d^* - l_k, & d^* < 0 \end{cases} \quad (3)$$

Затим се ова вредност претвара у бинарни запис са $\log_2(u_k - l_k + 1)$ битова где су u_k и l_k границе опсега којима припада разлика d^* и додаје се до тог тренутка екстрактованим битовима тајног податка.

Постоје разне модификације које се могу применти на основном алгоритму заснованом на PVD. Разне методе варирају по величини блока пиксела, границама опсега разлика пиксела унутар блока, броју битова који се уграђују у блок у зависности од вредности разлике, начину обиласка слике, као и начину решавања проблема када новодобијене вредности буду ван опсега.

Метода PVD са троструким диференцирањем (Tri-Way PVD) ради са блоковима пиксела величине 2x2. Рачунају се разлике између три пара пиксела као што је приказано на слици 11, па се на основу вредности ових разлика и опсега којима припадају битови тајног податка уграђују у блок.



Слика 11, троструко дифренцирање

Овом методом се постиже већи капацитет слике, као и мања укупна дисторзија слике. Постоје и методе које раде на основу израчунавања функције модула које у потпуности узимају обзир корелацију између вредности црвене, зелене и плаве боје пиксела. Број битова који се уграђују у црвену и плаву вредност пиксела зависи од разлике између вредности зелене боје и медијане вредности зелене боје у свим блоковима.

5.2.3 Методе засноване на EMD

Главна идеја иза EMD (Exploiting modification direction) методе лежи у томе да једну цифру тајног податка у $(2n + 1)$-арном систему нотације носи n пиксела слике и у томе да је вредност било које боје пиксела највише измењена за 1 [7]. За сваких n пиксела дакле постоји $2n$ могућих модификација и још једна додатна када се ниједан пиксел не мења што даје укупно $(2n + 1)$ могућнх вредности тајне цифре која се уграђује у тих n пиксела.

Пре уграђивања потребно је низ битова који представљају тајни податак претворити у цифре са базом $(2n + 1)$. Низ битова тајног податка се дели на сегменте дужине L битова и децимална вредност сваког дела се у $(2n + 1)$-арном систему нотације представља са K цифара где је:

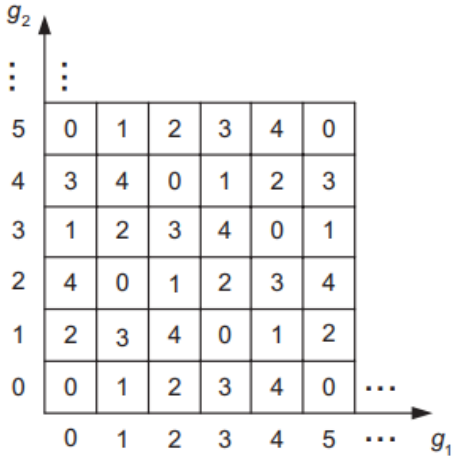
$$L = \lfloor K * \log_2(2n + 1) \rfloor \quad (4)$$

На пример, бинарна секвенца (1101 0110 1001) може бити представљена као (23 11 14) у нотационом систему са базом 5, где су $L = 4$ и $K = 2$. Дакле тајни податак се посматра као секвенца цифара у $(2n + 1)$-арном систему нотације.

Процес уграђивања почиње насумичним пермутовањем свих пиксела у оригиналној слици помоћу неког кључа дељеног између обе стране у комуникацији. Затим се пиксели групишу у групе од по n пиксела. Вредности пиксела у једној групи се означавају са g_1, g_2..., g_n, и за сваки блок се рачуна функција f на следећи начин:

$$f(g_1, g_2, \dots, g_n) = [\sum_{i=1}^{n} (g_i * i)] \, mod \, (2n + 1) \tag{5}$$

Пошто су вредности цели бројеви, вектор $[g_1, g_2, \dots, g_n]$ може бити представљен јединичном хипер-коцком. На слици 12 је приказан најпростији, дводимензиони случај за блокове од два пиксела односно где је $n = 2$. На осама се налазе све могуће вредности пиксела [0, 255], а у квадратима се налази f вредност за било који пар вредности пиксела. Вредности f у било ком квадрату се разликује од вредности $2n$ најближих квадрата и то су бројеви у опсегу $[0, 2n]$. Пример овакве представе f вредности дат је на слици 12.



Слика 12, f вредности за n=2

Затим се цифре тајног потака у $(2n + 1)$-арном систему нотације мапирају групама пиксела. Ако је тајна цифра d једнака функцији f није потребна никакава модификација. За $d \neq f$, рачуна се вредност $s = d - f mod(2n + 1)$. Ако је $s \leq n$, вредност пиксела g_s се повећава за 1, а ако није вредност пиксела g_{2n+1-s} се смањује за 1. На пример, нека је оригинална група пиксела у блоку [137 139 141 140], $n = 4$, $f = 3$ и тајна цифра коју је потребно уградити 4 у систему нотације са базом 9. Пошто је $s = d - f = 4 - 3 = 1$ што је мање од 4, вредност пиксела 1 се повећава за 1 и блок у стего слици ће бити [138 139 141 140]. У процесу екстракције тајног податка за тренутни блок у стего-слици је само потребно израчунати врендност f.

До проблема може доћи при покушају да се вредност пиксела g_s повећа за 1 или вредност пиксела g_{2n+1-s} се смањи за 1 што није дозвољено ако је пиксел засићен. На пример,

блок пиксела у коју је потребно уградити тајне податке има вредности [255 255 255 254]. $n = 4$, $f = 8$ и потребно је уградити у тај блок цифру 0 што даје $s = 1$. Ово захтева да се вредност првог пиксела у блоку увећа за 1, што није могуће јер има вредност 255. У овом случају вредност првог пиксела се смањује за 1 и блок постаје [254 255 255 254] што даје $f = 7$, а одатле $s = 2$, што би значило да је потребно повећати вредност другог пиксела, али то није могуће јер је и други пиксел засићен. Поступак се понови, односно уместо тога вредност другог пиксела се смањи за 1 тако да блок постане [254 254 255 254] и онда је $f = 5$ што даје $s = 4$ што значи да је потребно повећати четврти пиксел у блоку тако да је коначна вредност пиксела у блоку стего-слике [254 254 255 255]. Иако је у овом случају потребно вршити модификацију више пиксела у блоку, оваква појава је веома ретка тако да се генерални квалтет стего-слике не квари.

Разне методе су засноване на EMD и настале су са циљем да се повећа капацитет односно да се обезбеди пренос што више информација као и повећа визуелни квалитет стего-слике. Методе HoEMD (Highlight of exploiting modification direction) и AdEMD (Adaptive exploiting modification direction) користе опреацију модула и узимају у обзир осетљивост људског визуелног система. HoEMD искоришћава правце пиксела. Пиксели са већим променама значе више праваца, а самим тим се у њих може уградити више битова тајног податка. Диференцирање пиксела у оквиру AdEMD методе се користи да би се донела одлука о томе да ли пиксел који се налази у ивичној регији може толерисати веће промене од пиксела која се налази у глаткој регији. Код ове методе се такође користи таблица са опсезима и промењене вредности пиксела се налазе у истом опсегу као и оригиналне вредности пре убацивања тајних битова. Све методе засноване на EMD морају поштовати ово правило да би се тајни податак коректно екстрактовао из стего-слике.

5.2.4 Методе засноване на упаривању пиксела (PPM – Pixel Pair Matching)

Ове методе користе пар пиксела $(p_{i,1}, p_{i,2})$ као референтну координату за налажење другог пара пиксела $(p'_{i,1}, p'_{i,2})$ у предефинсаној области $\phi\,(p_{i,1}, p_{i,2})$ тако да буде задовољено $f(p'_{i,1}, p'_{i,2})$ = SB [8]. Где је f функција екстракције, а SB вредност цифре тајног податка у B-арном систему нотације. Процес уграђивања цифре тајног податка се врши заменом пара $(p_{i,1}, p_{i,2})$ са $(p'_{i,1}, p'_{i,2})$. Приказ овог процеса дат је на слици 13.

База B се добија као $B = 2k^2 + 2k + 1$, где је k цео број већи или једнак јединици. Број k се одређује следећом формулом:

$$\lfloor \frac{M_1 * M_2}{2} \log_2(2k^2 + 2k + 1) \rfloor \geq |S| \tag{6}$$

где $|S|$ представља дужину битова податка S, M_1 и M_2 редом представљају број редова и колона слике у коју ће се уградити тајни податак. Када се одреди параметар k потребно је одредити и област $\phi\,(p_{i,1}, p_{i,2})$ на следећи начин:

$$\phi\,(p_{i,1}, p_{i,2}) = \{(a, b) \,|\, |a - p_{i,1}| + |b - p_{i,2}| \leq k\} \tag{7}$$

Карактеристична вредност за сваки елемент (a, b) у области ϕ $(p_{i,1}, p_{i,2})$ рачуна се на следећи начин:

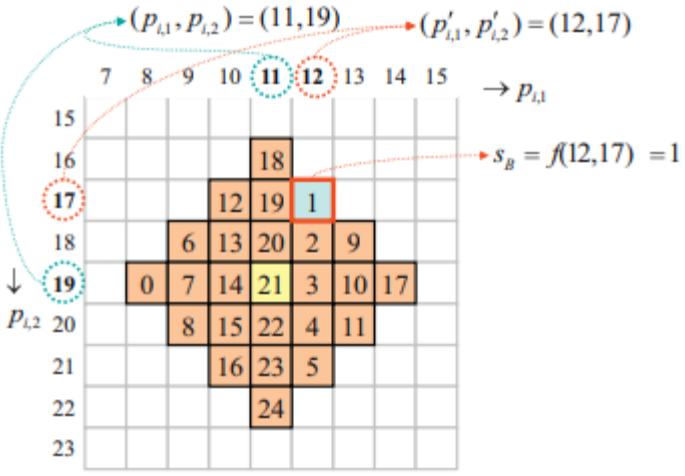
$$f(a,b) = mod((2k+1)a + b, B) \tag{8}$$

где $f(a,b)$ представља функцију екстракције а $mod(x,y)$ је функција која рачуна остатак при дељењу броја x бројем y. Да би се убацила цифра SB у бази B у пикселе са вредностима $p_{i,1}$ и $p_{i,2}$, претражују се вредности у ϕ $(p_{i,1}, p_{i,2})$ како би се пронашле координате $(p'_{i,1}, p'_{i,2})$ такве да је $f(p'_{i,1}, p'_{i,2})$ = SB. Пар пиксела $(p_{i,1}, p_{i,2})$ се мења паром $(p'_{i,1}, p'_{i,2})$. Да би се решио проблем испадања вредности ван опсега [0, 255] модификује се вредност p'_x у p''_x на следећи начин:

$$p''_x = \begin{cases} p'_x - B, & p'_x > 255 \\ p'_x + B, & p'_x < 0 \end{cases} \tag{9}$$

Да би се екстрактовала цифра тајног податка SB уграђена у пар пиксела $(p'_{i,1}, p'_{i,2})$ потребно је само израчунати карактеристичну функцију тог пара, односно SB = $f(p'_{i,1}, p'_{i,2})$.

На пример, $k = 3$ и потребно је уградити цифру 1 у систему нотације са базом 25 у пар пиксела (11, 19). Област ϕ (11, 19) је прказана на слици 13. Пошто је у ϕ (11, 19), *f*(12, 17) = 1, пар пиксела (11, 19) мења се паром пиксела (12, 17). Да би се екскрактовала тајна цифра потребно је само израчунати $f(12,17) = 1_{25}$, дакле добијена тајна цифра је 1_{25}.



Слика 13, област ϕ(11, 19) и карактеристичне вредности

Адаптивна PPM метода заснива се на налажењу области ϕ (x, y) дизајнираној специјално за уграђивање цифре B тако да се у стего-слици обезбеди најмања дисторзија. Метода PPM се може користити и у комбинацији са PVD где се у пар пиксела тајни податак уграђује коришћењем две референтне табеле.

5.3 Стеганографија слика у фреквенцијском домену

Методе које скривају податке у фреквенцијском домену су почеле да се појављују касније у односу на методе које раде у просторном домену. Брзим развојем информационих технологија, настаје све већа потреба за високом сигурношћу. Самим тим што расте компутациона моћ уређаја, пружа се могућност за коришћење комплекснијих метода скривања података које се некада нису користиле због времена потребног да се изврше потребна израчунавања. Иако су се методе које раде у просторном домену развиле до те мере да је постојање тајног податка у слици неприметно људском оку, ове методе не пружају велику сигурност и подложне су детекцији при стеганализи. Као одговор на постојеће проблеме појавиле су се стеганографске методе које раде у фреквенцијском домену слике [3].

5.3.1 Методе засноване на дискретној косинусној формацији (DCT)

Методе које раде у просторном домену нису погодне за рад са форматима слика које користе поступак компресије при чему се трајно оштећују и губе тајни подаци. Поступак DCT трансформације за улазну слику F при коме се добија излазна слика T се врши помоћу следећег обрасца :

$$T_{pq} = \alpha_p \alpha_q \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} F_{mn} cos \frac{\pi(2m+1)p}{2M} cos \frac{\pi(2n+1)q}{2N} \tag{10}$$

где је $0 \leq p \leq M, 0 \leq q \leq N$ и

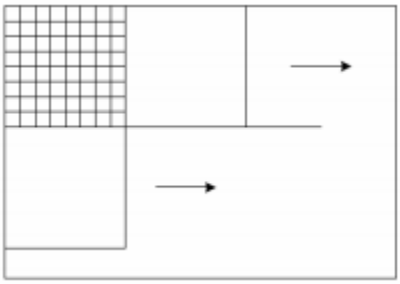
$$\alpha_p = \begin{cases} \frac{1}{\sqrt{M}}, & p = 0 \\ \sqrt{2/M}, & 1 \leq p \leq M-1 \end{cases} \quad \alpha_q = \begin{cases} \frac{1}{\sqrt{N}}, & q = 0 \\ \sqrt{2/N}, & 1 \leq q \leq N-1 \end{cases}$$

M и N су димензије улазне слике, док m и n су променљиве које се крећу у опсегу од 0 до M - 1 и 0 до N – 1 респективно.

DCT своју примену налази највише у JPEG сликама . JPEG компресија примењује дискретну косинусну трансформацију на сваки блок слике величине 8x8. Начин поделе слике на блокове приказан је на слици 14. За сваки блок се израчунава 64 коефицијената. Коефицијенти дискретне косинусне трансформације за блок величине 8x8 се израчунавају заменом вредности у формулу 10. [9]:

$$T_{pq} = \frac{1}{4} C_p C_q \sum_{m=0}^{7} \sum_{n=0}^{7} F_{mn} cos \frac{\pi(2m+1)p}{16} cos \frac{\pi(2n+1)q}{16} \tag{11}$$

где је $C_x = \frac{1}{\sqrt{2}}$ за $x = 0$ и $C_x = 1$ за $1 \leq x \leq 7$.



Слика 14, подела слике на блокове

Израчунати коефицијенти се затим квантизују помоћу одговарајуће матрице квантизације. Матрица квантизације је матрица димензија 8x8 и свака од вредности коефицијената $T_{pq}, 0 \leq p, q \leq 7$ се кванитизује помоћу одговарајуће вредности из матрице.

Вредност $T_{0,0}$ представља једносмерну компоненту и представља средњу вредност свих вредности пиксела. Матрица квантизације приказана је на слици 15.

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Слика 15, матрица квантизације

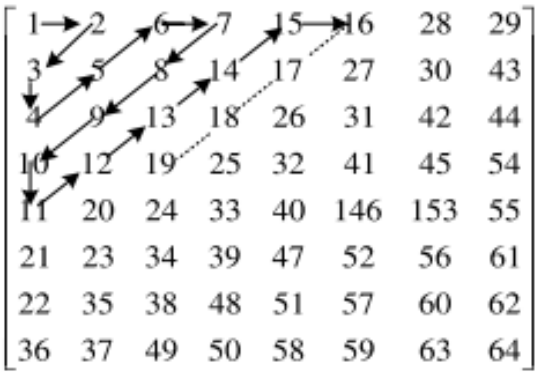
Поступак квантизације се врши дељењем коефицијената одговарајућим вредностима у матрици квантизације, а вредности у матрици су означене са $k_{pq}, 0 \leq p, q \leq 7$. Квантизовани коефицијент је дат следећом формулом:

$$kT_{pq} = \left[\frac{T_{pq}}{k_{pq}}\right] \tag{12}$$

Матрица квантизације није јединствена, али матрица на слици 15 је матрица предложена у JPEG стандарду, међутим може се десити да различити произвођачи фотоапарата могу користити другачије матрице квантизације. Вредности унутар матрице квантизације добијене су екстензивним експериментисањем и изабране тако да се постигне компромис између компресије и квалитета слике [10].

Основна стеганографска метода који ради у фреквенцијском домену и заснована је на DCT трансформацији и као слику за скривање података користи слику JPEG формата је J-Steg метода [11]. Метода почиње партиционисањем слике на блокове величине 8x8 и

затим користи дискретну косинусну трансформацију да трансформише блок у коефицијенте који се скалирају помоћу стандарне JPEG матрице за квантизацију која је показана на слици 15. Затим се битови тајног податка уграђују у квантизоване коефицијенте који су различити од 0, 1 или -1. Редослед за обилазак слике који ова метода користи је цик-цак и приказан је на слици 16.



Слика 16, цик-цак обилазак

На крају се користи нека врста ентропијског кодирања тако да се сваки блок компресује и генерише се JPEG датотека. Најчешћа врста кодирања је Huffman-ово кодирање које за симболе који се чешће појављују кодира са мање битова. Пошто ће након квантизације неки, а често и већина коефицијената постати 0, капацитет ове методе је знатно лимитиран. Још један од недостатака је то што ће се битови тајне поруке најчешће уграђивати у коефицијенте у нискофреквентном делу, што су заправо и најбитнији коефицијенти. У нискофреквентном делу садржана је највећа енергија слике јер су на природним сликама заступљенији благи прелази.

Пример изгледа једног блока пиксела током J-Steg алгоритма који се састоји од примене DCT трансформације на том блоку, квантизације коефицијената а затим и уграђивања битова у коефицијенте приказан је на слици број 17.

$$\begin{bmatrix} 139 & 144 & 149 & 153 & 155 & 155 & 155 & 155 \\ 144 & 151 & 153 & 156 & 159 & 156 & 156 & 156 \\ 150 & 155 & 160 & 163 & 158 & 156 & 156 & 156 \\ 159 & 161 & 162 & 160 & 160 & 159 & 159 & 159 \\ 159 & 160 & 161 & 162 & 162 & 155 & 155 & 155 \\ 161 & 161 & 161 & 161 & 160 & 157 & 157 & 157 \\ 162 & 162 & 161 & 163 & 162 & 157 & 157 & 157 \\ 162 & 162 & 161 & 161 & 163 & 158 & 158 & 158 \end{bmatrix}$$

а)

$$\begin{bmatrix} 1260 & -1 & -12 & -5 & 2 & -2 & -3 & 1 \\ -23 & -17 & -6 & -3 & -3 & 0 & 0 & -1 \\ -11 & -9 & -2 & 2 & 0 & -1 & -1 & 0 \\ -7 & -2 & 0 & 1 & 1 & 0 & 0 & 0 \\ -1 & -1 & 1 & 2 & 0 & -1 & 1 & 1 \\ 2 & 0 & 2 & 0 & -1 & 1 & 1 & -1 \\ -1 & 0 & 0 & -1 & 0 & 2 & 1 & -1 \\ -3 & 2 & -4 & -2 & 2 & 1 & -1 & 0 \end{bmatrix}$$

б)

$$\begin{bmatrix} 79 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

ц)

$$\begin{bmatrix} 78 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -3 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

д)

Слика 17, а) блок пиксела на оригиналној слици, б) коефицијенти DCT трансформације, ц) квантизовани коефицијенти DCT трансформације, д) коефицијенти након уметања података

Главни недостатак ове методе је то што се тајни податак уграђује у коефицијенте ниске фреквенције, где је садржана највећа енергија слике што доводи до смањења квалитета слике када се у њу угради подаци. Једно од решења овог проблема које је настало са циљем да се побољша визуелни квалитет слике је уметање битова тајног податка у коефицијенте средњих фреквенција.

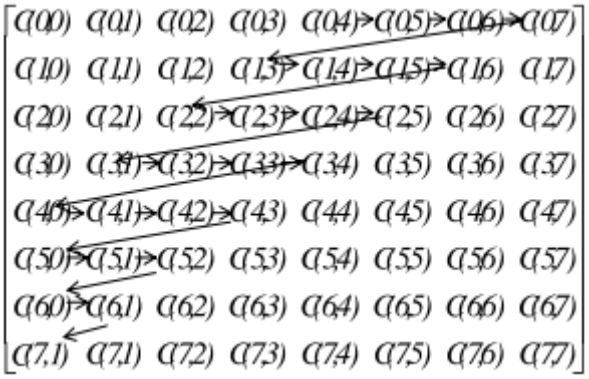
JQTM метода користи другачију табелу од J-Steg методе, као и другачији обилазак табеле квантизације при уграђивању битова [12]. Модификована табела приказана је на слици 18.

$$\begin{bmatrix} 16 & 11 & 10 & 16 & 1 & 1 & 1 & 1 \\ 12 & 12 & 14 & 1 & 1 & 1 & 1 & 55 \\ 14 & 13 & 1 & 1 & 1 & 1 & 69 & 56 \\ 14 & 1 & 1 & 1 & 1 & 87 & 80 & 62 \\ 1 & 1 & 1 & 1 & 68 & 109 & 103 & 77 \\ 1 & 1 & 1 & 64 & 81 & 104 & 113 & 92 \\ 1 & 1 & 78 & 87 & 103 & 121 & 120 & 101 \\ 1 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

Слика 18, JQTM табела квантизације

Вредности коефицијената ниских и високих фреквенција остају исти као у JPEG квантизационој табели док се вредности 26 коефицијената средњих фреквенција постављају на 1.

Као и у J-Steg методи, слика се дели на непреклапајуће блокове пиксела величине 8x8, а затим се за сваки блок рачунају коефицијенти DCT трансформације који се затим квантизују помоћу JQTM табеле. Након тога је потребно уградити битове тајног податка унутар слике. Са $C_i[a, b]$ су означене вредности коефицијента у реду a и у колони b у i-том блоку, а тајни податак је представљен низом битова $S = \{s_1, s_2, \dots, s_m\}$, где је m број битова тајног податка. У два бита најмање тежине коефицијената средње фреквенције у блоку се уграђује два бита поруке. Битови s_1 и s_2 се уграђују у два бита најмање тежине коефицијента $C_1[0, 4]$, битови s_3 и s_4 се уграђују у два бита најмање тежине коефицијента $C_1[0, 5]$ и тако даље. Ред уметања битова тајне поруке у коефицијенте приказан је на слици 19.



Слика 19, редослед уграђивања битова у JQTM методи

Затим се као и у претходној методи врши нека врста ентропијског кодирања и генерише се компресована JPEG датотека која садржи тајни податак.

5.4 Адаптивне методе и стеганографија дубоким учењем

Стеганографија се уопштено може представити као игра 3 играча [13]. Две стране, назване лице А и лице Б, покушавају да размене поруку, а да то не буде детектовано од стране неког трећег лица. Они користе безазлени медијум, у овом случају дигиталну слику, како би сакрили тајну поруку. Треће лице, означено као лице Ц посматра комуникацију између лица А и Б. Лице Ц посматрањем комуникације проверава да ли су дигиталне слике природне, односно не садрже никакве тајне податке или су поруке сакривене у њих, или је то стего-слика која скрива тајну поруку.

Ознака игре између ова три играча потиче из теорије игара. Сваки играч покушава да нађе стратегију која максимизује његове шансе за победу. Овако се овај проблем може представити као min-max проблем који је потребно оптимизовати. Оптимално решење за овај проблем, ако постоји, се назива Нешов еквилибријум. Нешов еквилибријум је концепт решења игре са два или више играча код ког се подразумева да сваки играч зна стратегије еквилибријума осталих играча и да ниједан играч не може профитирати променом своје стратегије. Скуп играча се налази у Нешовом еквилибријуму ако је сваки од њих донео најбољу могућу одлуку узевши у обзир одлуке свих осталих играча.

У стеганографији тешко је математички формализовати овај проблем, па се Нешов еквилибријум одређује симулацијом игре. Лице А игра игру изоловано од осталих играча, односно нема интеракцију са лицем Б и лицем Ц при одређивању најбољег алгоритма за уметање тајног податка у слику. Идеја је да лице А користи 3 алгоритма (или 2 у поједностављеној верзији где се лице Б занемарује) који се зову агенти и који су означени као: агент А, агент Б и агент Ц. Циљ агента А је да тајни податак уметне у слику на такав начин да агент Ц не може то да детектује, а да агент Б може да екстрактује тајни податак из стего-слике. У смислу теорије игара постоје две стране које се такмиче (Лице А и лице В са једне стране и лице Ц са друге). Лице А може да покрене симулацију у којој се агенти „надмећу“. Када агенти достигну Нешов еквилибријум, лице А завршава симулацију и чува агента А као свој стратешки адаптивни алгоритам за уметање, а лицу Б шаље агента Б што представља његов стратешки адаптивни алгоритам за екстракцију података из стего-слике. Тајна комуникација између лица А и Б је сада могућа коришћењем ова два алгоритма.

Претече неуронских мрежа у тражењу оптималног алгоритма за уметање података су приступи MOD (Model Optimized Distortion) и ASO (Adaptive Steganography by Oracle). Оба приступа се заснивају на надметању агента А и агента Ц. Агент Б се не користи јер агент А генерише мапу цене која сваком пикселу у слици придружује вредност цене. Мапа цене се користити за уметање података помоћу алгоритма STC (Syndrome-Trellis Codes) који минимизује дисторзију слике. Лице А генерише мапу за слику у коју ће се уградити подаци, а затим STC алгоритмом скрива податке унутар слике и добија стего-слику. Лице Б прменом STC алгоритма може екстрактовати скривене податке из стего-слике.

Оба приступа пролазе кроз два корака која се понављају док се не достигне критеријум заустављања:

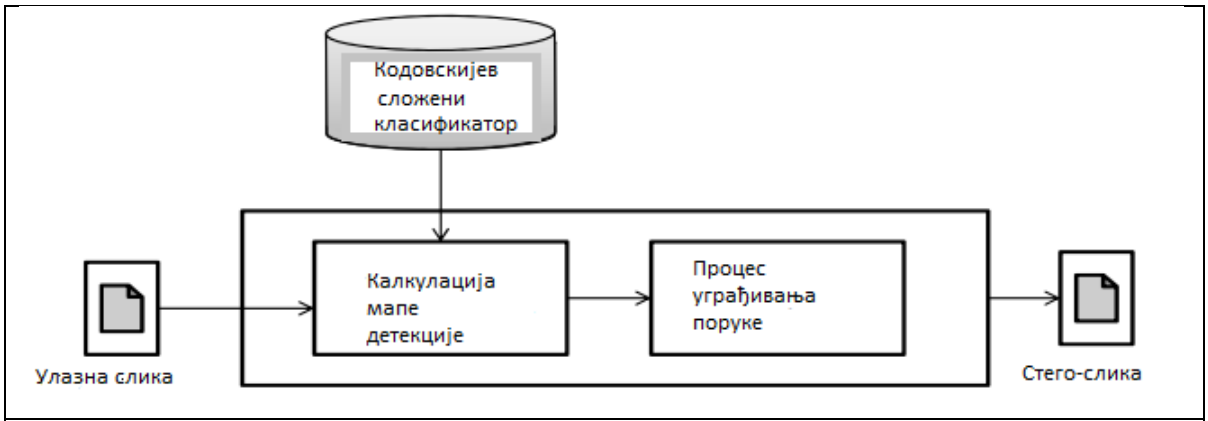
- 1) Агент А ажурира мапу цена уметања тако што пита агента Ц како је најбоље ажурирати сваку цену уметања тако да се шанса за детекцију што

више смањи. У MOD приступу, агент Ц је SVM (Support Vector Machine), а агент А ажурира цене уметања тако што смањује SVM маргину између оригиналне и стего-слике. У ASO приступу, агент Ц је сложени класификатор назван Oracle.

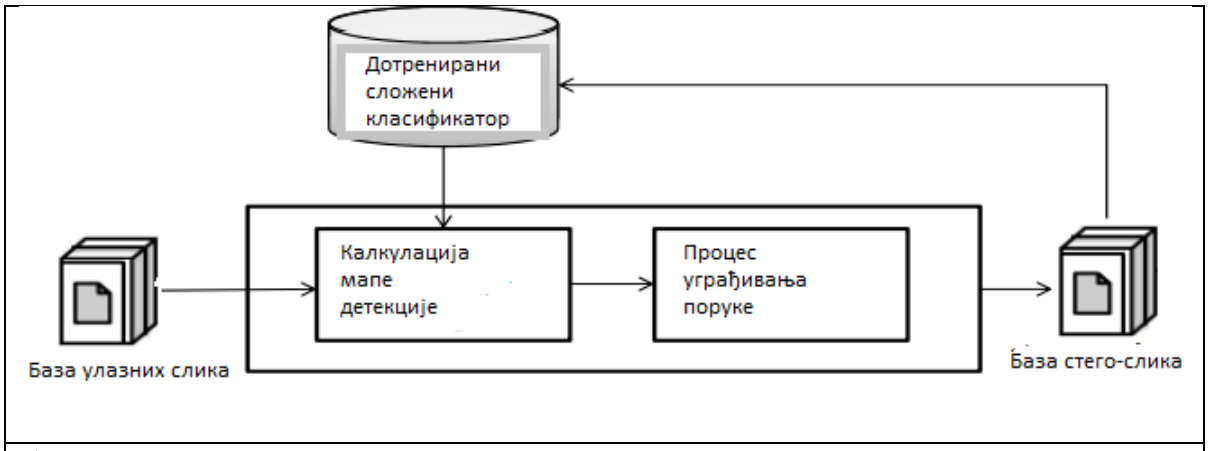
- 2) Агент Ц ажурира свој класификатор учећи како да открије тајне податке у стего-слици

Ако је са $x = (x_1, x_2, \dots, x_n)$ означен скуп свих пиксела оригиналне слике. Циљ је уметнути тајну поруку $m = (m_1, m_2, \dots, m_m)$ у слику тако да се направи стего-слика са пикселима $y = (y_1, y_2, \dots, y_n)$ тако да се минимизује функција дисторзије $D(x, y)$ под условом константног капацитета. Функција дисторзије заснована је на мапи цене која сваком пикселу у оригиналној слици придружује вредност цене која представља утицај на сигурност при модификацији тог пиксела. Код MOD приступа вредност цене је параметризована великим бројем параметара. Тражење најбољих параметара који обезбеђују највећу сигурност се врши симплекс алгоритмом оптимизације који се састоји од итеративног уметања поруке у слику носиоца и модификацију параметара. Критеријум за заустављање је величина SVM маргине.

ASO приступ при рачунању мапе цена не узима у обзир дистрибуцију модела само тренутне слике већ и дистрибуцију базе података пошиљаоца. Овај приступ користи непараметарску методу која користи Кодовскијев сложени класификатор за рачунање мапе цена. На тај начин се у потпуности искоришћавају информације о слици носиоцу информација и целокупна база пошиљаоца и омогућава да се при слању тајних информација одабере најсигурнија слика. Генерална шема ASO приступа приказана је на слици 20.



а)



б)

Слика 20, генерална ASO шема

Први корак приказан на слици 20 а) користи скуп класификатора који су научени да разликују слике без података и стего-слике. Рачуна се мапа цене и она се користи за уметање тајног податка у слику. Други корак је итеративни корак који служи да повећа сигурност и смањи шансу за детекцију тајног податка. Користе се класификатори који су истренирани у претходном кораку и могу да разликују оригиналне слике и стего-слике добијене у претходном кораку. Итеративни процес се понавља све док се не достигне жељена вероватноћа грешке класификатора и на крају овог корака се добија база стего-слика. Након извршења обе фазе ове методе, пошиљалац може да изабере слику која обезбеђује највећу сигурност тајног податка.

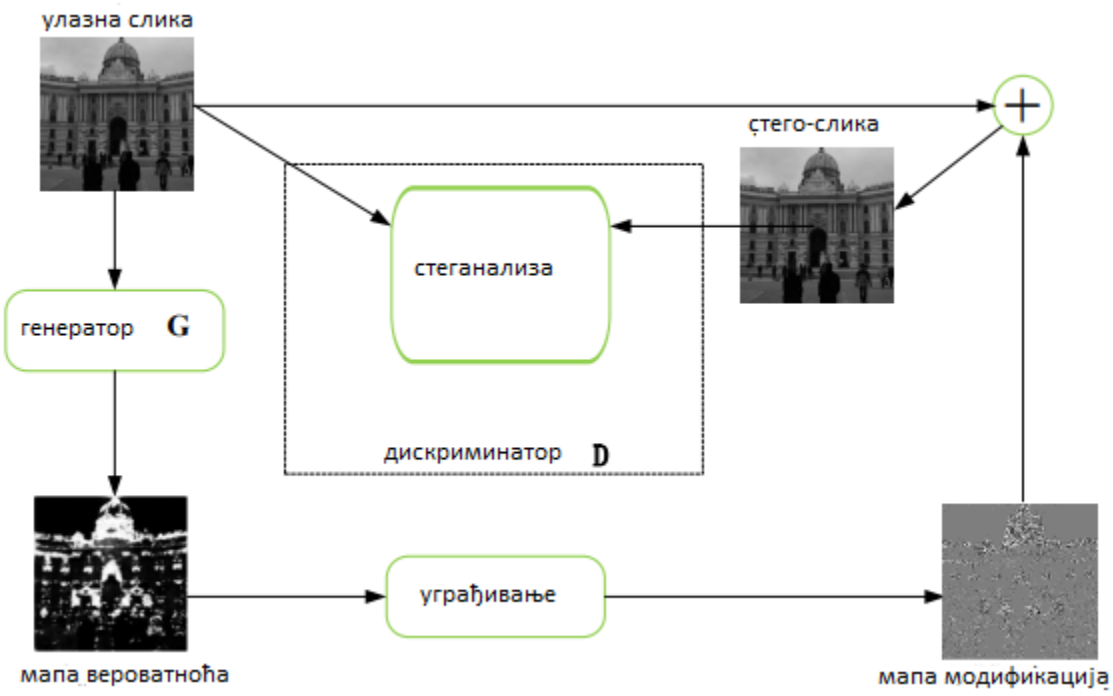
Коришћење неуронских мрежа омогућава још лакшу симулацију помоћу дискриминаторских мрежа чија је улога да донесу одлуку да ли је слика права или синтетичка. Овај приступ се назива GAN (Generative Adversial Networks). Неуронским мрежама је још лакше представити min-max проблем, а оптимизација се спроводи алгоритмом пропагације уназад. Захваљујући разним библиотекама дубоког учења које су се појавиле претходних неколико година, сада је лако направити GAN систем у разним програмским језицима, а растом компутационе моћи симулације су убрзане тако да се мреже брже тренирају.

Први приступ синтези слика помоћу GAN генератора подразумевала је генерисање слика које ће носити тајни податак а затим скривање података унутар те слике њеном модификацијом. Овакав приступ се назива приступ са модификацијом. Идеја је да овакав приступ генерише базу слика носиоца са великом сигурношћу. Међутим овај приступ није толико логичан и сматра се да је уместо генерисања синтетичких слика носиоца боље користити природне слике. Интересантнији приступ је синтеза слика које у себи већ садрже тајне информације односно стего-слике. Овакав приступ се назива приступ без модификације зато што се генерише слика која одмах може бити послата.

Пре свега потребно је креирати неуронску мрежу која је у могућности да генерише слике. У ову сврху користи се DCGAN (Deep Convolutional GAN) неуронска мрежа. Овај генератор може да генерише слике када се на улаз доведе вектор фиксне величине са

вредностима униформно расподељеним између [-1, 1]. Други корак се састоји у учењу друге мреже да екстрактује вектор када јој се на улаз доведе синтетисана слика. Вектор који на излазу даје ова мрежа мора бити једнак вектору који се користио за генерисање слике. Сада лице А мапира поруку помоћу униформно распоређеног вектора фиксне величине и генерише слику. Лице Б поседује мрежу за екстракцију и на основу слике које добије од лица А може да добије тајну поруку.

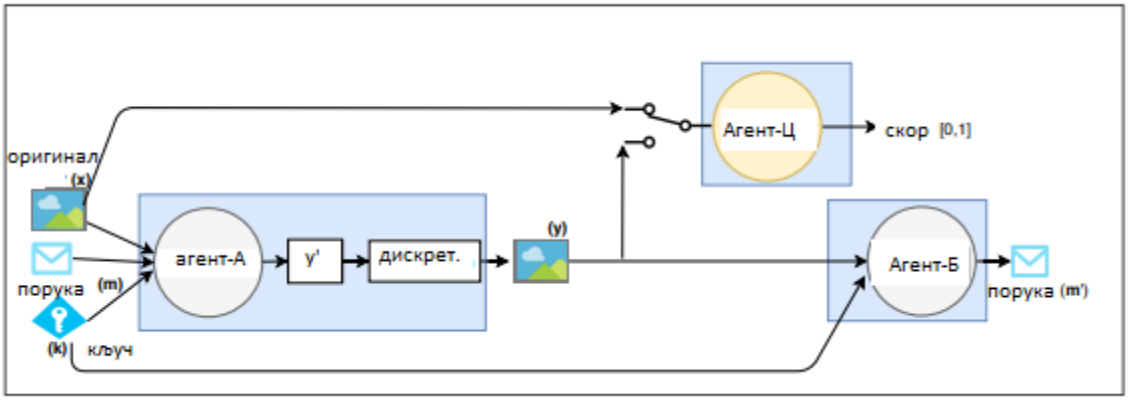
Следећи приступ је приступ заснован ма модификацији мапе вероватноћа. У овом приступу постоје две мреже: генераторска и дискриминаторска мрежа. Генераторска мрежа на основу оригиналне слике генерише мапу која се зове мапа вероватноћа модификације. На основу ове мапе помоћу неке функције за насумично извлачење добија се мапа са вредностима које припадају скупу {-1, 0, +1}. Ова мапа се назива мапа модификације и представља такозвани стего-шум. Дискриминаторска мрежа као улаз добија слику добијену сумирањем (пиксел по пиксел) оригиналне слике и стего-шума добијеног генераторском мрежом. Циљ диксриминаторске мреже је да разликује оригиналу и стего-слику. Циљ генераторске мреже је да генерише такву мапу модификација која ће учинити то да мрежа дискриминатор греши што је више могуће.



Слика 21, приступ заснован на модификацији

Приступ заснован на игри са 3 играча користи концепт симулације игре који је објашњен на почетку овог погавља. Разлика је у томе што су три алгоритма односно агента у овом приступу неуронске мреже. Агент А и агент Б су повезани том смислу да раде са заједничким циљем да обезбеде сигурну комуникацију односно да информације остану скриревене од агента Ц а тајни податак може да се екстрактује без грешака. Принцип ове игре са три играча приказан је на слици 22. Агент А на основу кључа и оригиналне слике

генерише стего-слику. Ову стего-слику користи агент Б како би могао да добије тајни податак, а агент Ц доноси одлуку о томе да ли је слика заправо оригинална слика без тајних података или стего-слика и на основу своје предикције као излаз даје скор. Коришћењем неуронскх мрежа олакшава достизање стратешког еквилибријума јер је проблем представљен као min-max проблем и оптимизација се може вршити пропагацијом уназад.



Слика 22, приступ заснован на игри 3 играча

6. Имплементација алгоритама у програмском језику Пајтон

У овом делу рада биће описан рад и начин програмске имплементације различитих комбинација алгоритама који раде у просторном домену слике са варијацијама величине блока пиксела на који се дели оригинална слика. Слике које ће скривати податке су RGB слике PNG формата. Алгоритми ће такође бити упоређени по односу сигнал/шум, укупном капацитету односно највећој дужини тајног податка који могу сакрити и просечном броју битова који се мењају унутар сваког бајта односно вредности једне боје пиксела. Алгоритми су реализовани у програмском језику Пајтон, а библиотека која се користи рад са сликама, односно за учитавање оригиналне слике, чување стего-слике, рачунање просечног броја измењених битова и односа сигнал шум је OpenCV.

Реализовани програм је апликација командне линије. Слика у коју се уграђују подаци, тајни податак, назив стего слике и одабрани алгоритам бирају се задавањем аргумената у командној линији. Потребно је прво конфигурисати аргументе у коду.

```
parser = argparse.ArgumentParser()
parser.add_argument('--inimage', help='Putanja do slike u koju skrivamo
podatke; podrazumevano cover.png')
```

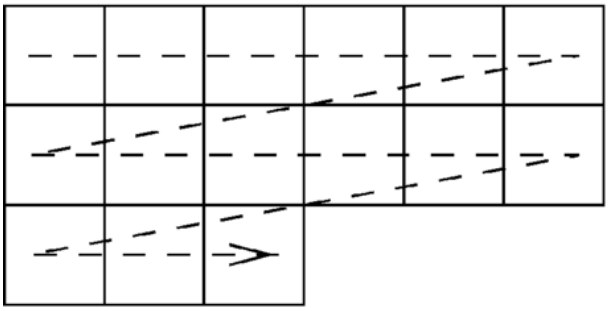
```
parser.add_argument('--alg', help='Algoritam koji se koristi za skrivanje
tajnih podataka')
parser.add_argument('--outimage', help='Putanja do slike u koju ce biti
sakriveni podaci podrazumevano stego.png')
parser.add_argument('--infile', help='putanja do tajne datoteke')
args = parser.parse_args()
```

При екстракцији података потребно је задати име стего-слике, назив датотеке која ће представљати екстрактовани тајни податак и алгоритам који мора одговарати алгоритму коришћеном при уметању података.

```
parser = argparse.ArgumentParser()
parser.add_argument('--inimage', help='Putanja do slike u kojoj su skrivene
informacije')
parser.add_argument('--alg', help='Algoritam koji se koristi za skrivanje
tajnih podataka')
parser.add_argument('--outfile', help='Ime nove datoteke; podrazumevano
izlaz; ekstenziju dobija u zavisnosti od skrivenih podataka')
args = parser.parse_args()
```

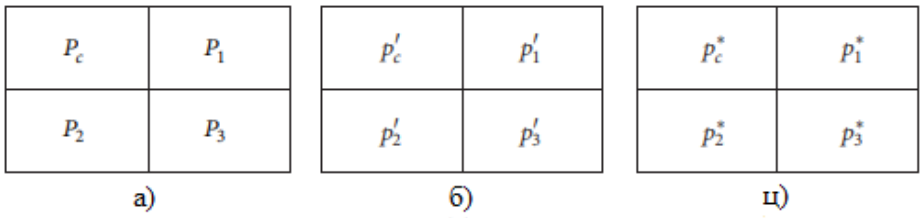
Алгоритми су засновани на LSB, PVD и EMD.

Начин обиласка слике код свих имплементираних алгоритама је исти и приказан је на слици 23. При обиласку за сваки блок алгоритам се примењује за вредности сваке од три боја пиксела редом.

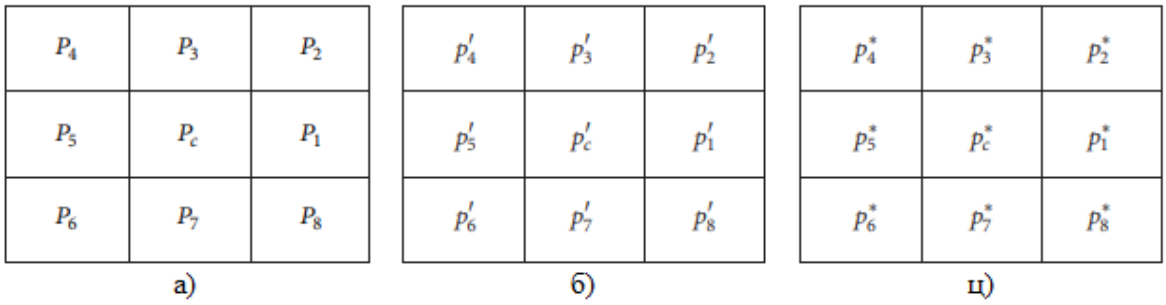


Слика 23, начин обиласка слике

Код метода које раде са величинама блокова 2x2, блок пиксела на оригиналној слици је приказан на слици 24а). Након уметања тајних података у ту слику блок постаје стего-блок и он је означен као на слици 24б). Након слања стего-слике, страна која прима слику може да екстрактује поруку и блок пиксела при екстракцији је приказан на слици 24ц). Слично, за блок величине 3x3 изгледи оригиналног, стего и блока коришћеног за екстракцију су приказани на слици 25 [14].



Слика 24, блокови пиксела величине 2x2



Слика 25, изглед блокова величине 3x3

Тајна порука може бити било која датотека (слика, звучни запис, текстуална датотека, извршна датотека...). На почетку потребно је припремити датотеку како би она могла да се уметне у слику. Тајна датотека се учитава као бинарна и преводи се у дугачак низ битова који се умећу у слику. Код за поступак превођења датотеке у низ битова је дат у наставку.

```
def tobits3(s):
    res = ''
    for c in s:
        bits = bin(c)[2:]
        bits = '00000000'[len(bits):] + bits
        res += bits
    return res
```

6.1 LSB+PVD метода са блоком величине 2x2

Слика се подели на блокове и обилази као што је приказано на слици 23. Затим се за сваки блок врши уметање.

Бит најмање тежине тежине пиксела p_c се мења вредношћу 1 што може да послужи као индикатор како би се убрзала екстракција података. Следећа два бита најмање тежине овог пиксела се мењају помоћу два бита тајног податка и добија се нова вредност пиксела p'_c. За случај да је овом променом дошло до значајније промене вредности пиксела потребно је извршити оптимизацију вредности на следећи начин. Са s_1 је означена децимална вредност 3 бита најмање тежине пиксела p'_c, а децимална вредност 3 бита најмање тежине

пиксела p_c је означена са i_1. Рачуна се разлика између ове две вредности као $df1 = i_1 - s_1$ и вредност p_c' се оптимизује на следећи начин:

$$p_c' = \begin{cases} p_c' + 2^3, \text{ако је } df1 > 2^{3-1}, 0 \leq (p_c' + 2^3) \leq 255 \\ p_c' - 2^3, \text{ако је } df1 < -2^{3-1}, 0 \leq (p_c' - 2^3) \leq 255 \\ p_c', \qquad \text{иначе} \end{cases} \tag{13}$$

Затим се рачунају вредности $d_i = |p_c' - p_i|$ за $i = 1,2,3$, односно за сваки преостали пиксел у блоку. Та вредност се налази у једном од опсега из табеле 1.

Табела 1, табела опсега

Опсег, $\{l_i, u_i\}$	$R_1 = \{0,7\}$	$R_1 = \{8,15\}$	$R_1 = \{16,31\}$	$R_1 = \{32,63\}$	$R_1 = \{64,127\}$	$R_1 = \{128,255\}$
Број битова који се умећу n_i	3	3	3	3	4	4

На основу опсега којем припада d_i из табеле се добија вредност n_i односно број битова тајног податка који ће бити сакривен унутар тог пиксела. Затим се n_i битова тајног податка конвертује у децималну вредност ds_i за $i = 1,2,3$. На основу ове вредности рачуна се нова вредност разлике као $d_i' = l_i + ds_i$ за $i = 1,2,3$. За сваку вредност пиксела у блоку p_i за $i = 1,2,3$ рачунају се две нове вредности $p_i'' = p_c' - d_i'$ и $p_i''' = p_c' + d_i'$. Нова вредност p_i' се бира на следећи начин:

$$p_i' \begin{cases} p_i'', \text{ако је } |p_i - p_i''| < |p_i - p_i'''|, 0 \leq p_i'' \leq 255 \\ p_i''', \qquad \text{иначе} \end{cases} \tag{14}$$

Код задатог алгоритма дат је у наставку и итеративно се позива за сваки блок слике.

```
def pvd2x2(podmatrica, data, dokle):

    ps = int(podmatrica[0][0])
    p1 = int(podmatrica[0][1])
    p2 = int(podmatrica[1][0])
    p3 = int(podmatrica[1][1])
    pk = [p1, p2, p3]
    podmatrica[0][0] = podmatrica[0][0] | 1
    podaci = data[dokle: dokle+2]

    dokle += 2
    podmatrica[0][0] = zamenibitove(podmatrica[0][0], podaci)
    df1 = int(('00000000'+bin(ps)[2:])[-3:],2) -
int(('00000000'+bin(podmatrica[0][0])[2:])[-3:],2)
```

```
    if df1>4 and podmatrica[0][0]+8>0 and podmatrica[0][0] + 8 <= 255:
        podmatrica[0][0] += 8
    elif df1<-4 and podmatrica[0][0]-8>0 and podmatrica[0][0] - 8 <= 255:
        podmatrica[0][0] -= 8

    for i in range(3):
        d=abs(podmatrica[0][0]-pk[i])
        for j in range(len(number_of_bits)):
            if (d>=number_of_bits[j][1] and d<=number_of_bits[j][2]):
                l=number_of_bits[j][1]
                n=number_of_bits[j][0]

        if dokle>len(data)-n:
            return dokle+n
        ds=data[dokle:dokle+n]
        dp=l+int(ds,2)
        pip=int(podmatrica[0][0]-dp)
        pipp=int(podmatrica[0][0]+dp)
        if (pipp>255):
            pipp-=255
        if (pip<0):
            pip+=255
        if (abs(pk[i]-pip)<abs(pk[i]-pipp) and pip>=0 and pip<=255):
            pk[i]=pip
        else:
            pk[i]=pipp
        dokle+=n

    podmatrica[0][1]=pk[0]
    podmatrica[1][0]=pk[1]
    podmatrica[1][1]=pk[2]
```

Оригинална слика, односно слика која ће носити информације је димензија 894x890 и приказана је на слици 26. Тајни податак који ће бити сакривен у слику може бити било шта, али је за пример узета слика димензија 458x408. Тајна слика приакзана је на слици број 27.

Слика 26, оригинална слика

Слика 27, тајни податак

Након покретања алгоритма командом:

```
python3 embed.py --inimage cover.png --infile secret.png --outimage stego.png --alg pvd2
```

добија се стего-слика односно слика у коју је сакривена тајна.

Слика 28, стего-слика добијена PVD+LSB 2x2 алгоритмом

Стего-слика се сада може послати примаоцу. Када прималац прими стего-слику он коришћењем одговарајућег алгоритма може добити тајне податке из слике. Алгоритам за екстракцију података објашњен је у наставку.

Слика се обилази истим редоследом као и при уметању битова, а редослед је приказан на слици 23. Блок при екстракцији података је приказан на слици 24ц). Проверава се бит најмање тежине пиксела p_c^* и ако је 1 значи да се ту налазе подаци и да је потребно извршити екстракцију података из тог блока пиксела. За случај да је бит најмање тежине пиксела p_c^* једнак 0, алгоритам екстракције се завршава и тиме се штеди време јер се не обилази цела слика пошто би се обилазили и делови у којима уопште нису сакривени подаци. Ово је нарочито корисно у случајевима када су тајни подаци веома мале величине у односу на слику која их крије. Затим се екстрактују следећа два бита најмање тежине пиксела p_c^*. Након тога за сваку вредност пиксела у блоку рачунају се вредности $d^* = |p_c^* - p_i^*|$ и $s_i^* = d_i^* - l_i$ за $i = 1,2,3$. Вредност d_i^* припада једном од опсега приказаних на слици 26, а вредност l_i је доња граница тог опсега. Сада се s_i^* конвертује у бинарну вредност са n_i бинарних цифара на основу опсега ком припада d_i^*.

Код за овај алгоритам дат је у наставку.

```
def decpvd2x2(podmatrica):.
    data=''
    p = []
    for i in range (2):
```

```
        for j in range (2):
            p.append(podmatrica[i][j])
        p0_bin = bin(p[0])
    p0_bin = '00000'+bin(p[0])[2:]
    data=data + p0_bin[len(p0_bin)-3: len(p0_bin)-1]
    for i in range(1,4):
        di=abs(int(p[0])-int(p[i]))
        for j in range(len(number_of_bits)):
            if (di>=number_of_bits[j][1] and di<=number_of_bits[j][2]):
                l=number_of_bits[j][1]
                n=number_of_bits[j][0]
        si=di-l
        si_bin='00000'+bin(si)[2:]
        data=data+si_bin[-n:]
    return data
```

Прималац покреће процес екстракције командом:

```
python3 extract.py --inimage stego.png --outifle secret-new --alg pvd2
```

6.2 LSB+EMD метода са блоком величине 2x2

Слика се подели на блокове и обилази као што је приказано на слици 23. Затим се за сваки блок врши уметање.

Бит најмање тежине тежине пиксела p_c се мења вредношћу 0 што може да послужи као индикатор како би се убрзала екстракција података. Следећа два бита најмање тежине овог пиксела се мењају помоћу два бита тајног податка и добија се нова вредност пиксела p'_c. За случај да је овом променом дошло до значајније промене вредности пиксела потребно је извршити оптимизацију вредности на следећи начин. Са s_1 је означена децимална вредност 3 бита најмање тежине пиксела p'_c, а децимална вредност 3 бита најмање тежине пиксела p_c је означена са i_1. Рачуна се разлика између ове две вредности као $df1 = i_1 - s_1$ и вредност p'_c се оптимизује помоћу формуле 13.

Преостали пиксели у блоку (p_1, p_2, p_3) су означени као p_k. У сваку од вредности p_k ће бити уграђено два бита тајног податка. Нека је децимална вредност та два бита тајног податка означена са m_k. Потребно је одредити x из опсега $\{-3, -2, -1, 0\}$ и одредити $p''_k = p_k + x$ тако да је услов $p''_k mod\ 4 = m_k$ задовољен. На сличан начин потребно је одредити x из опсега $\{1,2,3\}$ и одредити $p'''_k = p_k + x$ тако да је услов $p'''_k mod\ 4 = m_k$ задовољен, а ако ни за једну вредност из опсега једнакост није задовољена поставља се $p'''_k = -10$. Нова вредност пиксела p'_k рачуна се на следећи начин:

$$p'_k = \begin{cases} p''_k, \text{ако је } \{(p'''_k < 0 \; ili \; p'''_k > 255), 0 \le p''_k \le 255\} \\ \quad \text{или } \{0 \le (p''_k, p'''_k) \le 255, |p_k - p''_k| \le |p_k - p'''_k|\} \\ p''_k, \text{ако је } \{(p'''_k < 0 \; ili \; p'''_k > 255), 0 \le p''_k \le 255\} \\ \quad \text{или } \{0 \le (p''_k, p'''_k) \le 255, |p_k - p''_k| \le |p_k - p'''_k|\} \end{cases} \quad (15)$$

Код описаног алгоритма написан у програмском језику Пајтон дат је у наставку.

```
def emd2x2(podmatrica, data, dokle):

    ps = int(podmatrica[0][0])
    p1 = int(podmatrica[0][1])
    p2 = int(podmatrica[1][0])
    p3 = int(podmatrica[1][1])
    pk = [p1, p2, p3]
    podmatrica[0][0] = podmatrica[0][0] & 254

    podaci = data[dokle: dokle+2]
    dokle += 2
    podmatrica[0][0] = zamenibitove(podmatrica[0][0], podaci)
    df1 = int(('00000000'+bin(ps)[2:])[-3:],2) -
int(('00000000'+bin(podmatrica[0][0])[2:])[-3:],2)
    if df1>4 and podmatrica[0][0]+8>0 and podmatrica[0][0] + 8 <= 255:
        podmatrica[0][0] += 8
    elif df1<-4 and podmatrica[0][0]-8>0 and podmatrica[0][0] - 8 <= 255:
        podmatrica[0][0] -= 8

    opcije1 = [-3,-2,-1,0]
    opcije2 = [1, 2, 3]
    pip = []

    if dokle > len(data)-2:
        return dokle+2

    for k in pk:
        tr = data[dokle: dokle+2]
        for x in opcije1:
            if (k+x) % 4 == int(tr,2):
                pip.append(k+x if k+x>0 else 4+k+x)
                break
        dokle+=2

    dokle -= 6
    pipp = [-10, -10, -10]

    for k in range(len(pk)):
        for x in opcije2:
            tr = data[dokle: dokle+2]
            if (pk[k]+x) % 4 == int(tr,2):
```

```
                pipp[k] = pk[k]+x
                break
        dokle+=2

    nove = []
    for i in range(3):
        if (pipp[i]<0 or pipp[i] > 255) or (pip[i]>=0 and pip[i]<=255 and
pipp[i]>=0 and pipp[i] <= 255 and abs(p1-pip[i]) <= abs(p1-pipp[i])):
            nove.append(pip[i])
        else:
            nove.append(pipp[i])

    podmatrica[0][1] = nove[0]
    podmatrica[1][0] = nove[1]
    podmatrica[1][1] = nove[2]
```

Као пример користе се оригинална слика и тајна слика као и у претходном алгоритму које су приказане редом на сликама 26 и 27, а стего-слика добијена након убацивања тајне слике у оригиналну приказана је на слици број 29.

Након покретања алгоритма командом:

`python3 embed.py --inimage cover.png --infile secret.png --outimage stego.png --alg emd2`

добија се стего-слика односно слика у коју је сакривена тајна слика.

Слика 29, стего-слика добијена EMD+LSB 2x2 алгоритмом

Када прималац прими стего слику он може из ње екстрактовати тајни податак на начин описан у наставку. Слика се обилази на исти начин као и при убацивању тајног података. Ако је бит најмање тежине пиксела p_c^* једнак 0 то је индикатор да се ту налазе подаци и да је потребно извршити екстракцију. Следећа два бита најмање тежине из пиксела p_c^* се екстрактују. Из осталих пиксела у блоку (p_1^*, p_2^*, p_3^*) децимална вредност уметнутих битова је $m_k = p_k^* \; mod \; 4$ за $k = 1,2,3$ и свака од ових вредности се конвертује у две бинарне цифре и додаје до сада екстрактованим тајним подацима.

Код за поступак екстракције написан у програмском језику Пајтон је дат у наставку.

```
def decemd2x2(podmatrica):
    data = ''
    p = []
    for i in range (2):
        for j in range (2):
            p.append(podmatrica[i][j])
    p0_bin = bin(p[0])
    p0_bin = '00000'+bin(p[0])[2:]
    data=data + p0_bin[len(p0_bin)-3: len(p0_bin)-1]
    mi = []
    for i in range(1,4):
        mi.append(p[i] % 4)

    mi_bin = []
    for i in range(3):
        mi_bin.append('00'+bin(mi[i])[2:])

    for i in range(3):
        data=data + mi_bin[i][-2:]

    return data
```

Прималац покреће процес екстракције командом:

```
python3 extract.py --inimage stego.png --outifle secret-new --alg emd2
```

6.3 LSB+PVD метода са блоком величине 3x3

Оригинална слика се дели на блокове величине 3x3 и обилази се редоследом који је приказан на слици 23. Изглед блока оригиналне слике приказан је на слици 25а).

Три бита најмање тежине централног пиксела p_c се мењају помоћу три бита тајног податка и добија се нова вредност пиксела p_c'. Бит најмање тежине тежине пиксела p_8 се мења вредношћу 1 што може да послужи као индикатор како би се убрзала екстракција података. Следећа два бита најмање тежине овог пиксела се мењају помоћу два бита тајног

података и добија се нова вредност пиксела p'_8. Са s_1 је означена децимална вредност 3 бита најмање тежине пиксела p'_c, а децимална вредност 3 бита најмање тежине пиксела p_c је означена са i_1. Слично са s_2 је означена децимална вредност 3 бита најмање тежине пиксела p'_8, а децимална вредност 3 бита најмање тежине пиксела p_8 је означена са i_2, Рачунају се разлике $df1 = i_1 - s_1$ и $df2 = i_2 - s_2$. Вредност p'_c се оптимизује помоћу формуле 13, а вредност p'_8 помоћу формуле 16.

$$p'_8 = \begin{cases} p'_8 + 2^3, \text{ако је } df2 > 2^{3-1}, 0 \le (p'_8 + 2^3) \le 255 \\ p'_8 - 2^3, \text{ако је } df2 < -2^{3-1}, 0 \le (p'_8 - 2^3) \le 255 \\ p'_8, \qquad \text{иначе} \end{cases} \tag{16}$$

Затим се рачунају вредности $d_i = |p'_c - p_i|$ за $i = 1,2,\dots,7$, односно за сваки преостали пиксел у блоку. Та вредност се налази у једном од опсега из табеле 1. На основу опсега којем припада d_i из табеле се добија вредност n_i односно број битова тајног податка који ће бити сакривен унутар тог пиксела. Затим се n_i битова тајног податка конвертује у децималну вредност ds_i за $i = 1,2,\dots,7$. На основу ове вредности рачуна се нова вредност разлике као $d'_i = l_i + ds_i$ за $i = 1,2,\dots,7$. За сваку вредност пиксела у блоку p_i за $i = 1,2,\dots,7$ рачунају се две нове вредности $p''_i = p'_c - d'_i$ и $p'''_i = p'_c + d'_i$ и нова вредност p'_i се рачуна помоћу формуле 14. Код за реализацију овог алгоритма у програмском језику Пајтон дат је у наставку.

```
def pvd3x3(podmatrica, data, dokle):

    ps=podmatrica[0][0]
    p8=podmatrica[2][2]
    pk=[]
    for i in range(0,3):
        for j in range(0,3):
            if (i==1 and j==1) or (i==2 and j==2):
                pass
            else:
                pk.append(podmatrica[i][j])
    podaci=data[dokle: dokle+3]
    dokle+=3
    podmatrica[1][1] = zamenibitove2(podmatrica[1][1], podaci)
    podmatrica[2][2] = podmatrica[2][2] | 1
    podaci=data[dokle:dokle+2]
    dokle+=2
    podmatrica[2][2] = zamenibitove(podmatrica[2][2], podaci)
    df1 = int(('00000000'+bin(ps)[2:])[-3:],2) -
int(('00000000'+bin(podmatrica[1][1])[2:])[-3:],2)
    if df1>4 and podmatrica[1][1]+8>0 and podmatrica[1][1] + 8 <= 255:
        podmatrica[1][1] += 8
    elif df1<-4 and podmatrica[1][1]-8>0 and podmatrica[1][1] - 8 <= 255:
        podmatrica[1][1] -= 8
```

```
    df2 = int(('00000000'+bin(p8)[2:])[-3:],2) -
int(('00000000'+bin(podmatrica[2][2])[2:])[-3:],2)
    if df2>4 and podmatrica[2][2]+8>0 and podmatrica[2][2] + 8 <= 255:
        podmatrica[2][2] += 8
    elif df2<-4 and podmatrica[2][2]-8>0 and podmatrica[2][2] - 8 <= 255:
        podmatrica[2][2] -= 8
    for i in range(7):
        d=abs(int(podmatrica[1][1])-int(pk[i]))
        for j in range(len(number_of_bits)):
            if (d>=number_of_bits[j][1] and d<=number_of_bits[j][2]):
                l=number_of_bits[j][1]
                n=number_of_bits[j][0]
        if dokle>len(data)-n:
            return dokle+n
        ds=data[dokle:dokle+n]
        dp=l+int(ds,2)
        pip=int(podmatrica[1][1]-dp)
        pipp=int(podmatrica[1][1]+dp)
        if (pipp>255):
            pipp-=255
        if (pip<0):
            pip+=255
        if (abs(pk[i]-pip)<abs(pk[i]-pipp) and pip>=0 and pip<=255):
            pk[i]=pip
        else:
            pk[i]=pipp
        dokle+=n
    podmatrica[0][0]=pk[0]
    podmatrica[0][1]=pk[1]
    podmatrica[0][2]=pk[2]
    podmatrica[1][0]=pk[3]
    podmatrica[1][2]=pk[4]
    podmatrica[2][0]=pk[5]
    podmatrica[2][1]=pk[6]
```

Као пример користе се оригинална слика и тајна слика као и у претходном алгоритму које су приказане редом на сликама 26 и 27, а стего-слика добијена након убацивања тајне слике у оригиналну приказана је на слици број 30.

Након покретања алгоритма командом:

```
python3 embed.py --inimage cover.png --infile secret.png --
outimage stego.png --alg pvd3
```

добија се стего-слика односно слика у коју је сакривена тајна слика.

Слика 30, стего-слика добијена PVD+LSB 3x3 алгоритмом

Стего-слика се сада може послати примаоцу. Када прималац прими стего-слику он коришћењем одговарајућег алгоритма може добити тајне податке из слике. Алгоритам за екстракцију података објашњен је у наставку.

Слика се обилази истим редоследом као и при уметању битова, а редослед је приказан на слици 23. Блок при екстракцији података је приказан на слици 25ц). Проверава се бит најмање тежине пиксела p_8^* и ако је 1 значи да се ту налазе подаци и да је потребно извршити екстракцију података из тог блока пиксела. За случај да је бит најмање тежине пиксела p_8^* једнак 0, алгоритам екстракције се завршава и тиме се штеди време јер се не обилази цела слика пошто би се обилазили и делови у којима уопште нису сакривени подаци. Ово је нарочито корисно у случајевима када су тајни подаци веома мале величине у односу на слику која их крије. Затим се екстрактују следећа два бита најмање тежине пиксела p_8^* и три бита најмање тежине пиксела p_c^*. Након тога за сваку вредност пиксела у блоку рачунају се вредности $d^* = |p_c^* - p_i^*|$ и $s_i^* = d_i^* - l_i$ за $i = 1,2,\dots,7$. Вредност d_i^* припада једном од опсега приказаних на слици 26, а вредност l_i је доња граница тог опсега. Сада се s_i^* конвертује у бинарну вредност са n_i бинарних цифара на основу опсега ком припада d_i^* и додаје се до сада екстрактованим подацима. Код алгоритма екстракције написан у програмском језику Пајтон дат је у наставку.

```
def decpvd3x3(podmatrica):
    data=''
    p=[]
    for i in range(0,3):
```

```
        for j in range(0,3):
            if (i==1 and j==1) or (i==2 and j==2):
                pass
            else:
                p.append(podmatrica[i][j])

    pcbin='00000'+bin(podmatrica[1][1])[2:]
    data=data+pcbin[-3:]
    p8bin='00000'+bin(podmatrica[2][2])[2:]
    data=data+p8bin[len(p8bin)-3: len(p8bin)-1]
    for i in range(7):
        di=abs(int(podmatrica[1][1])-int(p[i]))
        for j in range(len(number_of_bits)):
            if (di>=number_of_bits[j][1] and di<=number_of_bits[j][2]):
                l=number_of_bits[j][1]
                n=number_of_bits[j][0]
        si=di-l
        si_bin='00000'+bin(si)[2:]
        data=data+si_bin[-n:]

    return data
```

Прималац покреће процес екстракције командом:

```
python3 extract.py --inimage stego.png --outifle secret-new --alg pvd3
```

6.4 LSB+EMD метода са блоком величине 3x3

Бит најмање тежине тежине пиксела p_8 се мења вредношћу 0 што може да послужи као индикатор како би се убрзала екстракција података. Следећа два бита најмање тежине овог пиксела се мењају помоћу два бита тајног податка и добија се нова вредност пиксела p_8'. За случај да је овом променом дошло до значајније промене вредности пиксела потребно је извршити оптимизацију вредности на следећи начин. Са s_2 је означена децимална вредност 3 бита најмање тежине пиксела p_8', а децимална вредност 3 бита најмање тежине пиксела p_8 је означена са i_2. Рачуна се разлика између ове две вредности као $df2 = i_2 - s_2$ и вредност p_8' се оптимизује помоћу формуле 16.

Преостали пиксели у блоку $(p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_c)$ су означени као p_k. У сваку од вредности p_k ће бити уграђено два бита тајног податка. Нека је децимална вредност та два бита тајног податка означена са m_k. Потребно је одредити x из опсега $\{-3, -2, -1, 0\}$ и одредити $p_k'' = p_k + x$ тако да је услов $p_k'' mod\ 4 = m_k$ задовољен. На сличан начин потребно је одредити x из опсега $\{1,2,3\}$ и одредити $p_k''' = p_k + x$ тако да је услов $p_k''' mod\ 4 = m_k$ задовољен, а ако ни за једну вредност из опсега једнакост није задовољена поставља се $p_k''' = -10$. Нова вредност пиксела p_k' се рачуна помоћу формуле 15. Код за реализацију описаног алгоритма написан у програмском језику Пајтон дат је у наставку.

```
def emd3x3(podmatrica, data, dokle):
```

```
    p8=podmatrica[2][2]
    pk=[]
    for i in range(0,3):
        for j in range(0,3):
            if i==1 and j==1:
                pass
            else:
                pk.append(podmatrica[i][j])

    podmatrica[2][2] = podmatrica[2][2] & 254
    podaci=data[dokle: dokle+2]
    dokle+=2
    podmatrica[2][2] = zamenibitove(podmatrica[2][2], podaci)

    df2 = int(('00000000'+bin(p8)[2:])[-3:],2) -
int(('00000000'+bin(podmatrica[2][2])[2:])[-3:],2)
    if df2>4 and podmatrica[2][2]+8>0 and podmatrica[2][2] + 8 <= 255:
        podmatrica[2][2] += 8
    elif df2<-4 and podmatrica[2][2]-8>0 and podmatrica[2][2] - 8 <= 255:
        podmatrica[2][2] -= 8

    if dokle > len(data)-2:
        return dokle+2

    opcije1 = [-3,-2,-1, 0]
    opcije2 = [1, 2, 3, 10]

    for i in range(8):
        podaci=data[dokle: dokle+2]
        dokle+=2
        if dokle > len(data)-2:
            return dokle+2
        m=int(podaci, 2)
        for j in opcije1:
            if (pk[i]+j)%4==m:
                pip=pk[i]+j
        pipp=-10
        for j in opcije2:
            if (pk[i]+j)%4==m:
                pipp=pk[i]+j

        if (pipp<0 or pipp > 255) or (pip>=0 and pip<=255 and pipp>=0 and
pipp <= 255 and abs(pk[i]-pip) <= abs(pk[i]-pipp)):
            pk[i]=pip
        else:
            pk[i]=pipp
    podmatrica[0][0]=pk[0]
    podmatrica[0][1]=pk[1]
    podmatrica[0][2]=pk[2]
```

```
    podmatrica[1][0]=pk[3]
    podmatrica[1][1]=pk[4]
    podmatrica[1][2]=pk[5]
    podmatrica[2][0]=pk[6]
    podmatrica[2][1]=pk[7]
```

Као пример користе се оригинална слика и тајна слика као и у претходном алгоритму које су приказане редом на сликама 26 и 27, а стего-слика добијена након убацивања тајне слике у оригиналну приказана је на слици број 31.

Након покретања алгоритма командом:

`python3 embed.py --inimage cover.png --infile secret.png --outimage stego.png --alg emd3`

добија се стего-слика односно слика у коју је сакривена тајна слика.

Слика 31, стего-слика добијена EMD+LSB 3x3 алгоритмом

Када прималац прими стего слику он може из ње екстрактовати тајни податак на начин описан у наставку. Слика се обилази на исти начин као и при убацивању тајног података. Ако је бит најмање тежине пиксела p_8^* једнак 0 то је индикатор да се ту налазе подаци и да је потребно извршити екстракцију. Следећа два бита најмање тежине из пиксела p_8^* се екстрактују. Из осталих пиксела у блоку $(p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_c)$ децимална вредност уметнутих битова је $m_k = p_k^* \, mod \, 4$ за $k = 1,2,3,4,5,6,7, c$ и свака од ових

вредности се конвертује у две бинарне цифре и додаје до сада екстрактованим тајним подацима.

Код за поступак екстракције написан у програмском језику Пајтон је дат у наставку.

```
def decemd3x3(podmatrica):

    data = ''
    p = []
    for i in range (3):
        for j in range (3):
            p.append(podmatrica[i][j])
    p8_bin = bin(p[8])
    p8_bin = '00000'+bin(p[8])[2:]
    data=data + p8_bin[len(p8_bin)-3: len(p8_bin)-1]
    mi = []
    for i in range(8):
        mi.append(p[i] % 4)

    mi_bin = []
    for i in range(8):
        mi_bin.append('00'+bin(mi[i])[2:])

    for i in range(8):
        data=data + mi_bin[i][-2:]

    return data
```

Прималац покреће процес екстракције командом:

```
python3 extract.py --inimage stego.png --outifle secret-new --alg emd3
```

6.5 Поређење имплементираних метода

Основна метрика за поређење представљених алгоритама је однос сигнала и шума (PSNR) који се добија поређењем оригиналне слике и стего-слике. Оригинална слика представља идеалну слику, а све промене на њој узрокују шум који утиче на генерални квалитет слике. PSNR се рачуна на следећи начин:

$$PSNR = 10 * \log_{10}\left(\frac{MAX_i^2}{MSE}\right) \tag{17}$$

где је:

$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n=1}[O(i,j) - S(i,j)]^2$$

Улазна слика и стего слика су димензија mxn, O представља оригиналну слику, а S стего-слику. Вредност MAX_i је максимална вредност пиксела. Пошто алгоритми раде са сликама PNG формата са 8-битним бојама, ова вредност је 255.

Следећа метрика за поређење ових алгоритама је укупан капацитет који је представља максималан број битова који су могли да буду уграђени у слику на одређеном сегменту. Сегмент на коме се рачуна укупни капацитет.

Још једна битна метрика за поређење алгоритама је број промењених битова по бајту оригиналне слике. Рачуна се као просечна вредност броја битова промењених пиксела у стего-слици у односу на оригиналну слику. Мање модификација по пикселу значиће мањи шум и мањи утицај на квалитет оригиналне слике.

Израчунате вредности за задату слику и задат тајни податак који су коришћени у претходним поглављима су дате у следећој табели:

Табела 2, добијене вредности за све алгоритме

	капацитет	PSNR	BPB
LSB+PVD 2x2	5346713	42.58	0.72
LSB+EMD 2x2	4773960	41.66	0.96
LSB+PVD 3x3	5538558	39.35	0.8
LSB+EMD 3x3	4773960	38.16	1.02

Алгоритми засновани на EMD имају фиксан капацитет јер се два бита тајне поруке уграђују у једну боју пиксела. EMD алгоритам са блоком величине 2x2 има бољи однос сигнал/шум од алгоритма PVD са блоком величине 3x3 иако мења просечно више битова по једном бајту, односно по једној боји пиксела. То се дешава јер алгоритми засновани на EMD стриктно мењају два бита најмање тежине у пикселима што производи мањи шум. Алгоритми са мањим блоковима генерално имају бољи однос сигнал/шум и мањи број битова који се мењају по једном бајту оригиналне слике.

Алгоритми засновани на PVD мењају 3 или 4 бита најмање тежине што зависи од тренутног блока пиксела, односно од тога да ли се блок налази у ивичној регији или глаткој регији слике. Такође, PVD приступ има бољу вредност односа сигнал/шум и мање модификација битова по једном бајту када ради са блоком величине 2x2, али алгоритам који ради са блоком величине 3x3 има већи капацитет. Алгоритми засновани на PVD су у овом случају бољи јер пружају знатно већи капацитет на цену малог смањења квалитета стего-слике.

Ове вредности се могу разликовати у зависности од изабране оригиналне слике и тајног податка, тако да пошиљалац може за тренутну слику коју шаље одредити најбоље параметре и тај алгоритам користити за слање.

7. Закључак

Стеганографија располаже мноштвом метода које обезбеђују сигурну комуникацију између пошиљаоца и примаоца. Представља алтернативу криптографији али се такође може користити и у комбинацији са њом. Циљ стеганографије је учинити комуникацију невидљивом нежељеним лицима. Кроз историју се користила у разним формама, а са развојем рачунара стеганографија се дигитализовала и нашла своју примену у разним областима, чак и у илегалним.

Методе које раде у просторном домену су лакше за разумевање и имплементацију, али имају већи неповољан на квалитет слике носиоца и подложније су статистичким нападима од техника које раде у спектралном домену. Неке од најзаступљенијих метода у просторном домену су: LSB, PVD и EMD. LSB као најједноставнија метода је најлакша за имплементацију, али њен главни недостатак је то што се битови носиоца мењају, а у обзир се не узима осетљивост људског визуелног система на те промене. EMD метода мења вредности пиксела у слици што је мање могуће, а PVD узима у обзир осетљивост људског визуелног система на промене и на основу тога бира број битова који ће се мењати у пикселу оригиналне слике. Ове методе је могуће комбиновати.

Методе у спектралном домену засноване су на DCT трансформацији и скривању података у коефицијентима трансформације који се затим квантизују. Основна метода J-Steg за квантизацију коефицијената користи табелу JPEG стандарда и битове тајне поруке крије у областима ниских фреквенција где је садржана највећа енергија слике јер ће већина коефицијената средњих и високих фреквенција бити анулирана. JQTM користи модификовану табелу и битови се скривају унутар коефицијената средње фреквенције.

Коришћењем неуронских мрежа могуће је истренирати мрежу за генерисање стего-слика која обезбеђује највећу сигурност података. За детекцију стеганографије се такође користи неуронска мрежа звана мрежа дискриминатор и циљ генераторске мреже је да произведе што више грешака дискриминатора.

При поређењу имплементираних метода јасно је да се мора наћи компромис између капацитета и одржавања квалитета слике када се у њу уметну тајни подаци. Ако две стране које комуницирају размењују обично податке који су доста мањи од капацитета слике носиоца увек је боље користити алгоритме који пружају бољи однос сигнал/шум јер је тако обезбеђен највећи могући квалитет стего-слике и самим тим смањена шанса за детекцију од стране трећег лица. Алгоритме са већим капацитетом треба користити само када величина тајног податка то захтева.

8. Литература

[1] N. Provos, P. Honeyman: Hide and seek: an introduction to steganography, IEEE Security & Privacy Volume: 1 , Issue: 3, 2003.

[2] A. Cheddad, J. Condell, K. Curran, P. Mc Kevitt: Digital image steganography: Survey and analysis of current methods,Elsevier Signal Processing 90 727–752, 2010.

[3] J. Judge: Steganography: Past, Present, Future, SANS Institute, 2020.

[4] E. Cole: Hiding in Plain Sight: Steganography and the Art of Covert Communication, Wiley Publishing, Inc., Indianapolis, Indiana, 2003.

[5] M. Hussain, A. T.S. Ho, A. Wahid, K. Jung, Image steganography in spatial domain: A survey, Development of IoT Security Middleware Platform for Multimedia Security, 2018.

[6] D. Wu, W. Tsai: A steganographic method for images by pixel-value differencing, Elsevier Pattern Recognition Letters 24 1613–1626, 2003

[7] X. Zhang, S. Wang: Efficient Steganographic Embedding by Exploiting Modification Direction, IEEE COMMUNICATIONS LETTERS, VOL. 10, NO. 11, 2006.

[8] W. Hong, T. Chen: A Novel Data Embedding Method Using Adaptive Pixel Pair Matching, IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, VOL. 7, NO. 1, 2012

[9] R. Ansari, C. Guillemot, N. Memon: Lossy Image Compression: JPEG and JPEG2000 Standards, Elsevier Handbook of Image and Video Compressing, 2005.

[10] H. Bhattacharjee, S. Kumar Bandyopadhyay: Frequency Domain Approach of Image Steganography, International Journal of Innovative Research in Information Security (IJIRIS) , 2016.

[11] X. Li, J. Wang: A steganographic method based upon JPEG and particle swarm optimization algorithm, Elsevier Information Sciences 177 3099–3109, 2007.

[12] C. Chang, T. Chen, L. Chung, A steganographic method based upon JPEG and quantization table modification, Elsevier Information Sciences 141 123–138, 2002.

[13] M. Chaumont. Deep Learning in steganography and steganalysis from 2015 to 2018. M. Hassaballah. Digital Media Steganography: Principles, Algorithms, Advances, Elsevier, 2019.

[14] A. Pradhan, K. Sekhar, G.Swain: Digital Image Steganography Using LSB Substitution, PVD, and EMD, Hindawi Mathematical Problems in Engineering, 2018.