Komunikacija upotrebom steganografije i kriptiranja

Hajduković, Hrvoje

Master's thesis / Diplomski rad

2017

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:592142

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-03-28**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

zir.nsk.hr

Image not found or type unknown

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Diplomski studij

KOMUNIKACIJA UPOTREBOM STEGANOGRAFIJE I KRIPTIRANJA

Diplomski rad

Hrvoje Hajduković

Osijek, 2017.

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

IZJAVA

Ja, Hrvoje Hajduković, OIB: 17765200733, student/ica na studiju: Diplomski sveučilišni studij Računarstvo, dajem suglasnost Fakultetu elektrotehnike, računarstva i informacijskih tehnologija Osijek da pohrani i javno objavi moj **diplomski rad**:

Komunikacija upotrebom steganografije i kriptiranja

u javno dostupnom fakultetskom, sveučilišnom i nacionalnom repozitoriju.

Osijek, 29.09.2017.

_____
potpis

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac D1: Obrazac za imenovanje Povjerenstva za obranu diplomskog rada

Osijek, 22.09.2017.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za obranu diplomskog rada

Ime i prezime studenta:	Hrvoje Hajduković
Studij, smjer:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D 799 R, 12.10.2015.
OIB studenta:	17765200733
Mentor:	Doc.dr.sc. Ivica Lukić
Sumentor:	Doc.dr.sc. Mirko Köhler
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	Doc.dr.sc. Zdravko Krpić
Član Povjerenstva:	Doc.dr.sc. Mirko Köhler
Naslov diplomskog rada:	Komunikacija upotrebom steganografije i kriptiranja
Znanstvena grana rada:	**Informacijski sustavi (zn. polje računarstvo)**
Zadatak diplomskog rada:	Ostavriti sigurnu komunikaciju putem chat-a između dva ili više korisnika upotrebom steganografije i kriptiranja. Cilj je upotrebom steganografije sakriti samo postojanje informacije, a kriptiranjem ju dodatno zaštitiu u slučaju otkrivanja.
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 2 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 2 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	22.09.2017.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 29.09.2017.

Ime i prezime studenta:	Hrvoje Hajduković
Studij:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D 799 R, 12.10.2015.
Ephorus podudaranje [%]:	10%

Ovom izjavom izjavljujem da je rad pod nazivom**: Komunikacija upotrebom steganografije i kriptiranja**

izrađen pod vodstvom mentora Doc.dr.sc. Ivica Lukić

i sumentora Doc.dr.sc. Mirko Köhler

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. UVOD 1
1.1. Zadatak diplomskog rada 1
2. KRIPTOLOGIJA 2
2.1. Kriptografija 3
2.1.1. Vrste kriptosustava 5
2.2. Kriptoanaliza 6
2.2.1. Vrste kriptoanalitičkih napada 6
3. STEGANOGRAFIJA 8
4. WEB TEHNOLOGIJE 12
5. WEB APLIKACIJA 16
5.1. Izgled aplikacije 19
5.2. Prijenos poruka 23
6. ZAKLJUČAK 27
LITERATURA ..... **Error! Bookmark not defined.**
SAŽETAK 29
ABSTRACT 30
ŽIVOTOPIS ..... **Error! Bookmark not defined.**
PRILOZI 32
Izvorni kodovi 32

1. UVOD

Čovjek je društveno biće. Kao takvo teži odnosu s drugim ljudima, prvenstveno kroz komunikaciju putem zajedničkog jezika. Komunikacija je moguća kao trenutna, licem u lice između dvije osobe ali i kao poruka koja se prenosi određenim medijem od pošiljatelja do primatelja. Posljednje spomenuta, udaljena komunikacija je tema ovog diplomskog rada.

Udaljena komunikacija je od davnina prisutna u ljudskim životima. Bilo da je riječ o porukama između poznanika, prijatelja, ljubavnika, kraljeva, vođa, političara ili čak ukućana, svaka od navednih komunikacija teži sigurnom transportu i zaštiti sadržaja.

Tema ovog diplomskog rada je sigurna komunikacija upotrebom kriptografije i steganografije. Kriptografija je već dugo vremena pristuna i vitalni dio komunikacijskog svijeta. Steganografija postaje sve više zanimljiva širokoj masi dok se tehnologije koje omugućavaju komunikaciju konstantno smjenjuju. Svrha diplomskog rada je objediniti kriptografiju i steganografiju u jednu web aplikaciju za grupnu komunikaciju. Trenutno popularne i tražene tehnologije poput Javascripta i NodeJS će biti upotrebljene prilikom izrade programskog rješenja.

Tijekom rada bit će objašnjena sama konstrukcija web aplikacije koja podrazumijeva web chat korištenjem HTML, CSS, Javascript i NodeJS tehnologija prilikom pisanja klijentskog i serverskog rješenja. Kao baza podataka korišten je MongoDB, točnije njegov modul Mongoose. Također će biti detaljno opisane kriptografske i steganografske metode kojim se nastoji izmjeniti i sakriti sadržaj poruke između korisnika aplikacije.

1.1. Zadatak diplomskog rada

Ostvariti sigurnu komunikaciju putem chat-a izmedu dva ili više korisnika upotrebom steganografije i kriptiranja. Cilj je upotrebom steganografije sakriti samo postojanje informacije, a kriptiranjem ju dodatno zaštiti u slučaju otkrivanja.

2. KRIPTOLOGIJA

Kriptologija znanost koja se bavi definiranjem i korištenjem metoda za zaštitu informacija dok se drugim dijelom bavi definiranjem i korištenjem metoda za otkrivanje šifriranih informacija [7]. Objekti izučavanja kriptologije su pisane, govorne, vizualne i druge poruke. Kao takva, kriptologija se koristi prvenstveno u vojne i diplomatske svrhe dok je razvojom računarstva i komunikacija prisutna gotovo svugdje.

Kriptografija je znanstevena disciplina za izučavanje i definiranje metoda zaštitu informacija.

Kriptoanaliza je znanstvena disciplina za izučavanje i definiranje metoda otkrivanja šifriranih informacija bez poznavanja ključa.

Probajmo oslikati slijedeću situaciju. Osoba A (u literaturi nazivna Alice) šalje poruku osobi B (u literaturi nazivna Bob Poruka se šalje preko nesigurnog medija, odnosno fizičkog prostora između lokacija Alice i Boba. Bilo da je riječ o prijenosu koji izvršava čovjek namjenjen za transport stvari, automatizirano vozilo, radio veza, telefonska veza ili računalna mreža svi spomenuti načini transporta su ranjivi i podležu mogućnosti pasivnog i aktivnog djelovanja na poruku. Nužno je pretpostaviti da postoji treća osoba E (u literaturi nazvana Eve) koja ima intenciju pročitati poruku ili izmjeniti njen sadržaj [1]. Ovdje je potrebno obratiti pažnju na dva neophodna svojstva transporta svake poruke.

Svojstvo povjerljivosti nalaže kako je sadržaj prenesene poruka zaštićen i neotkriven trećoj strani u komunikacijskom procesu.

Svojstvo integriteta nalaže kako je sadržaj prenesene poruke nepromjenjen u procesu slanja ali još važnije, kako je osoba koja šalje poruku ustinu ta osoba za koju se predstavlja.

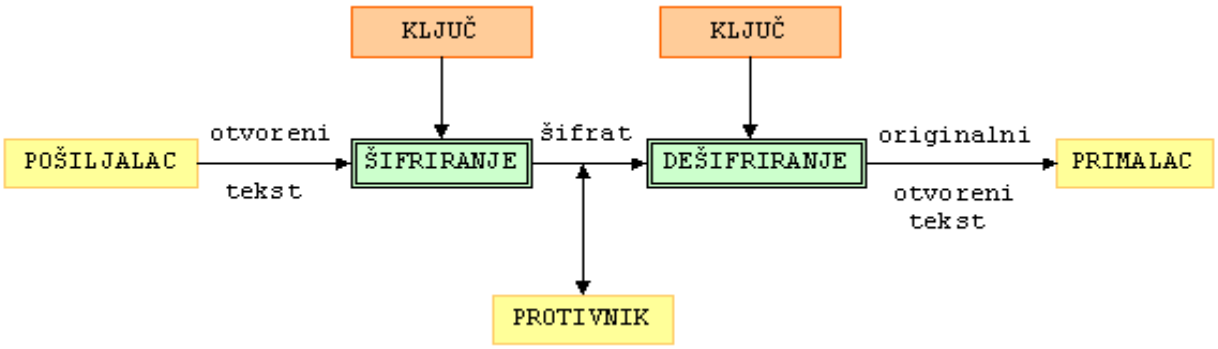
Svojstvo integriteta postiže se dodavanjem digitalnog pečata koristeći jednu od hash metoda koja za određeni tekst poruke stvara specifični digitalni pečat te ukoliko se poruka promjeni samo u jednom znaku, hash metoda kreira potpuno drugačiji digitalni pečat. Napoznatije hash metode su SHA-1, MD5 i SHA-256.

Svojstvo povjerljivosti postiže se primjenom različitih kriptograskih metoda. Kriptografske metode su srž poglavlja i njihov razvoj i metode će biti pobliže objašnjeni u narednim potpoglavljima.

2.1. Kriptografija

Kriptografija je grana kriptologije. Kriptografija ili kriptozaštita je znanstvena discipina koja se bavi izučavanjem i definiranjem metoda za zaštitu informacija u smislu da ih samo onaj kome su informacije namjenjene može pročitati i/ili koristiti [1]. Kriptografija se može prevesti kao tajnopis.

Kako bi pobliže definirali metode kriptografije vratit ćemo se na našu sliku tri osobe u procesu komunikacije. Svrha kriptografije je zaštiti sadržaj poruke od čitanja ili mjenjanja sadržaja poruke od strane treće osobe. Pa prijeđimo na spomenuti scenarij. Alice šalje poruku Bobu. Ako pretpostavimo kako su Alice i Bob uvaženi ratni vođe koji su trenutno u ključnoj fazi ratnih operacija koji će odrediti konačni tijek rata, proces sigurnog prenosa informacija dobiva na važnosti. Pretpostavimo da je Alice špijunažom otkrila tajni plan zajedničkog neprijatelja. Kako bi preduhitrili neprijatelja i okončali rat Alice i Bob moraju djelovati sinkronizirano. Alice odlučuju poslati Bobu poruku sa detaljima tajnog plana i metodom zajedničkog djelovanju u cilju presretanja neprijatelja i zadobivanja pobjede. Alice započinje proces korištenja kriptografije prilikom udaljene komunikacije. Alice ispisuje sadržaj poruke (u literaturi otvoreni tekst). Nakon toga otvoreni tekst dogovorenom metodom kriptiranja uz korištenje zajedničkog privatnog ključa prebacuje u novi tekst (u literaturi šifrat). Nakon toga šifrat napušta siguran prostor pošiljatelja i kreće na nesiguran i opasan put do primatelja. Na putu od pošiljatelja do primatelja postoji opravdana opasnost kako će treća osoba, u ovom slučaju zajednički ratni neprijatelj, sačekati poruke te probati iščitati ili promjeniti sadržaj poruke. Ovdje dolazi do izražaja svrha krptografije kao metode zaštite sadržaja poruke ali i svrha druge grane kriptologije a to je kriptoanaliza. Zajednički neprijatelj nastoji ne dešifrirati, jer za to je potreban privatni ključ koji posjeduju samo Alice i Bob, nego raznim drugim kriptoanalitičkim metodama doći do običnog teksta. Neovisno o uspješnosti razbijanja šifriranog teksta koje je proveo Eve, šifrat dolazi do Boba. Bob šifrat zaprima znajući kako je prošao kroz nesiguran medij. Nad porukom tada izvršava proces dešifriranja pomoću zajedničkog privatnog ključa koji posjeduju samo Alice i Bob. Nakon toga Bob iščitava sadržaj poruke, otvorenog teksta.



Sl. 2.1. Prikaz elemenata kriptografije

Dakako, problem koji se ovdje javlja je pitanje privatnog ključa. Navedena metoda spada u kategoriju simetričnih ključeva što znači da za šifriranje i dešifriranje koristi identičan privatni ključ. Važnije od toga je činjenica kako taj ključ mora biti ranije dogovoren na siguran način i krajnje je neodgovorno i nemudro privatne ključeve slati nesigurnim medijem. Iako je tako, metoda zajedničkog, odnosno dijeljenog privatnog ključa bila je u upotrebi skroz do sredine 20. stoljeća [7]. Tada je implementirana ideja javnog i tajnog ključa koji je vrlo učinkovito rješio problem tzv. „sigurne sobe“ za dogovor i razmjenu privatnih ključava ali i općenito problem sigurnosti poruke koja prolazi kroz nesiguran medij na vrlo elegantan način.

Drugi problem je pitanje integriteta sadržaja poruka [7]. S obzirom na korištene primitivne metode šifriranja i objašnjenje kriptografije s strane povjerljivosti, pitanje integriteta nije uključeno. Integritet ćemo razmotriti kasnije kada bude više riječi o naprednijim metodama šifriranja, prvenstveno se misli na one koje uključuju računala.

2.1.1. Vrste kriptosustava

Kriptosustave možemo klasificirati s obzirom na sljedeća tri kriterija [1]:

- Operacija prilikom šifriranja

Operacije prilikom šifriranja su supstitucijskae šifre i transpozicijske šifre. Supstitucijskim šiframa se svaki znak otvorenog teksta (bit, slovo, grupa bitova ili slova) zamjenjuje nekim drugim znakom. Transpozicijskim šiframa se znakovi otvorenog teksta permutiraju kako bi nastao novi tekst, odnosno šifrat. Traspozicijskim šiframa znakovi otvorenog teksta ostaju isti ali su razmješteni unutra poruke. Postoje također i kriptosustavi koji kombiniraju ove dvije metode.

- Operacije obrade teksta

Operacije obrade teksta su blokovne šifre i protočne šifre. Blokovne šifre obrađuju jedan po jedan blok znak otvorenog teksta. U tu svrhu koriste samo jedan ključ K. Protočne šifre obrađuju znakove otvorenog teksta također jedan po jedan s tim da pritiom koriste niz ključeva koji se pralelno generiraju.

- Svojstvo ključa

Ključevi se djele na javne i tajne, odnosno simetrične kriptosustave s tajnim ključem i asimtrične kriptosustave s javnim ključem. Simetrični ili konvencionalni kriptosustavi koriste identičan ključ za procese kriptiranja i dekriptiranja, što znači da se poznavajući jedan od ključeva, može u potpunosti kriptoanalizirati sustav. Sigurnost simetričnih kriptosustava leži u tajnosti ključa koji je ranije dogovoren na siguran način. Asimetrični kriptosustavi koriste javni ključ prilikom postupaka kriptografije. Gotovo je nemoguće u razumnom vremenu izračunati ključ za dekrptiranje iz ključa za kriptiranje, koji se na javni ključ. Svatko može šifrirati poruku pomoću javnog ključa, ali je nužno da osoba ima odgovaraju ključ za dešifriranje kako bi dešifrirala poruku. Whitfield Diffie i Martin Hellman su 1976. godine iznjeli ideju javnog ključa. Ideja je bila dio rješenja za problem razmjene ključeva za simetrične kriptosustave putem nesigurnog medija

2.2. Kriptoanaliza

Kriptoanaliza ili dekriptiranje je znanstvena disciplina koja se bavi definiranjem i izučavanjem postupaka za čitanje skrivenih poruka bez poznavanja ključa [1].

Osnovna pretpostavka kriptoanalize je da kriptoanalitičar zna koji se kriptosustav koristi. To se zove Kerckhoffsovo načelo, po Nizozemcu Augustu Kerckhoffsu (1835-1903), autoru važne knjige "La Cryptographie militaire" (Vojna kriptografija). Naravno, ova pretpostavka u konkretnom slučaju ne mora biti točna, ali mi ne želimo da nam sigurnost kriptosustava leži na nesigurnoj pretpostavci da naš protivnik ne zna koji kriptosustav koristimo. Čak ukoliko kriptoanalitičar treba provjeriti nekoliko mogućih kriptosustava, time se kompleksnost procedure bitno ne mijenja. Dakle, mi pretpostavljamo da tajnost šifre u potpunosti leži u ključu.

2.2.1. Vrste kriptoanalitičkih napada

Kriptoanalitičkih napade možemo podijeliti u četiri razine [1]:

- Poznavanje šifrata

Kriptoanalitičar posjeduje samo šifrat od nekoliko poruka šifriranih pomoću istog algoritma. Njegov je zadatak otkriti otvoreni tekst od što više poruka ili u najboljem slučaju otkriti ključ kojim su poruke šifrirane.

- Poznavanje otvorenog teksta

Kriptoanalitičar posjeduje šifrat neke poruke, ali i njemu odgovarajući otvoreni tekst. Njegov zadatak je otkriti ključ ili neki algoritam za dešifriranje poruka šifriranih s tim ključem.

- Poznavanje odabranog otvorenog tekst

Kriptoanalitičar ima mogućnost odabira teksta koji će biti šifriran, te može dobiti njegov šifrat. Ovaj napad je jači od prethodnoga, ali je manje realističan.

- Poznavanje odabranog šifrata

Kriptoanalitičar je dobio pristup alatu za dešifriranje, pa može odabrati šifrat, te dobiti odgovarajući otvoreni tekst. Ovaj napad je tipičan kod kriptosustava s javnim ključem. Tu je zadatak kriptoanalitičara otkriti ključ za dešifriranje (tajni ključ).

- Potkupljivanje, ucjena, krađa i slično

Ovaj napad ne spada doslovno u kriptoanalizu, ali je vrlo efikasan i često primjenjivan u kombinaciji s "pravim" kriptoanalitičkim napadima.

3. STEGANOGRAFIJA

Steganografija je znanstvena disciplina koja se bavi skrivanjem informacija [1]. Za razliku od kritpografije čiji je cilj sakriti sadržaj poruke, steganografija koristi tehnike za skrivanje samog postojanja poruke. Staganografskim tehnikama tajna poruka se skriva unutar teksta, zvuka ili najčešće slike tako da se postajanje poruke ni ne može uočiti.

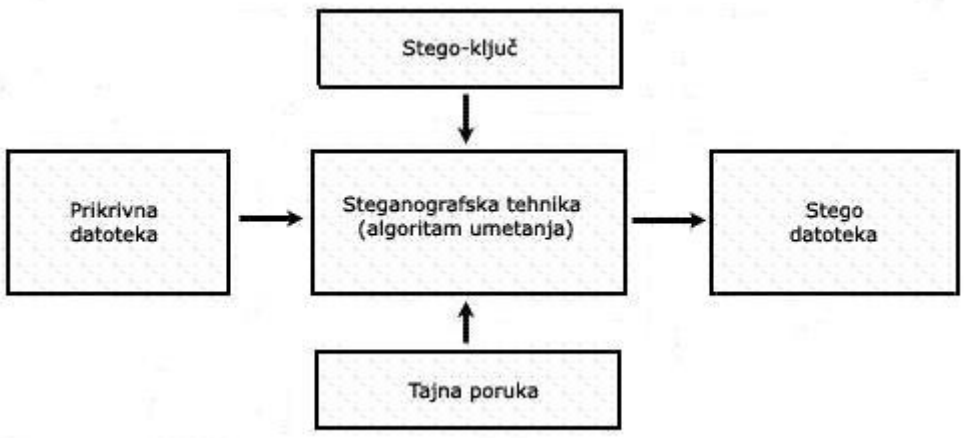
Povijest korištenja steganografije seže u vrijeme od 400 godina prije Krista. Steganografska metoda koja je tad bila korištena uključivala je brijanje glave nositelja poruke a zatim tetoviranje poruke koju bi s vremenom prikrila kosa. Dotični primatelj poruke je tada jednostavno obrijao glavu nositelja kako bi došao do sadržaja poruke. Navodno su se istom tehnikom koristili i njemački špijuni početkom 20.stoljeća. Steganografska metoda korištena u periodima svjetskih ratova je i nevidljiva tinta. Također, svijet knjiga poznaje steganografsku tehniku akrostih kod koje prva slova svakog stiha kriju određenu poruku. To su samo neki od primjeri nedigitalnih metoda steganografije.

Moderna steganografija uključuje digitalne sustave i odnosi se na skrivanje tajnih poruka u redundantnim dijelovima datoteke kao nositelja poruke. Digitalni svijet daje pregršt mogučnosti za primjenu steganografije. Najpoznatija i najraširenija steganografska metoda je skrivanje informacija u digitalnoj slici.

S obzirom kako su digitalne datoteke u računalo pohranjene kao nizovi bitova od kojih se neki više a neki manje bitni, ta činjenica se može iskoristiti i upotrijebiti prilikom primjene steganografije. Manje bitni bitovi, redundatni bitovi ne nose informacije važne za datoteku te upravo njih možemo upotrijebiti i zamijeniti bitovima informacije čije postojanje želimo sakriti. Ovdje je bitno spomenuti kako postoje datoteke koje su pogodone za stagnografske metode dok druge nisu. Primjer su svakako izvršne datoteke programa gdje je svaki bit važan i promjena samo jednog bita može uzrokovati neispravan rad programa.

Algoritam za primjenu steganografije je sljedeći. Vratimo se na popularni primjer komunikacije Alice i Boba uz Eve kao prisutnu treću stranu sa namjerom presretanja i djelovanja na poruku, bilo pasivno ili aktivno. Pošiljatelj Alice ispisuje poruku. Nakon toga izabere prikrivenu datototeku, najčešće sliku i u nju pohranjuje poruku jednom od steganografskih metoda. Slika koja posjeduje skrivenu poruku naziva se stegoobjekt i kao takva odlazi prema primatelju preko nesigurnog medija. Eve promatra promet i na prvi pogled uočava bezazlenu sliku i ne percipira ju kao krucijalnu i izvor informacija između Alice i Boba [7].

U idealnom slučaju niti čovjek niti računalo ne bi smjeli uočiti razliku između obične slike i stegoobjekta, slike koja sadrži tajnu poruku unutar sebe. Kako bi se to postiglo slika mora sadržavati dovoljno redundantnih bitova koji mogu biti zamjenjeni bitovima skrivene poruke. Također, poželjno je tajnu poruku šifrirati jednom od metoda šifriranja prije stvaranja stegoobjekta.



Sl. 3.1. Princip steganografskog procesa

Prilikom kreiranja stegoobjekta ili stegodatoteke poželjno je obratiti pozornost na format slike u koju skrivamo poruku kao i pozitivne i negativne strane steganografske metode koju koristimo.

S obzirom na navedeno, format slike koji je najčešći i najpogodniji za stvaranje stegoobjekata je BMP format. BMP format egzistira kao 16, 24 ili 32-bitni slikovni format. Slikovni formati koji su također prisutni u steganografiji su GIF i JPEG. BMP je nekompresirani slikovni format datoteke u kojemu je slika razložena na sitne kvadratiće, tzv. pixele. Prilikom korištenja steganografskih tehnika najčešće se koristi 24-bitni BMP format iz razloga što takav format osigurava mnogo prostora za skrivanje podataka. Boje su predstavljene kao kombinacija crvene,

zelene i plave gdje svaka od boja predstavalja jedan bajt podataka. Sumirano, kombinacija ove tri boje daje tri bajta, odnosno 24 bita koji predstavaljaju jedan piksel. Svaki piksel može poprimiti jednu od 2^{24} (oko 16.7 milijuna) nijansi. Nedostatak ovog formata je svakako veličina datoteke gdje jedna slika standardne veličine 1024x768 pixela zauozima oko 2.25MB.

Kada govorimo o tehnikama koje se koriste u steganografiji jedna od najčešće korištenih tehnika je svakako LSB (eng. least significant bit) supstitucija. LSB supstitucijom najmanje značajne bitove odabrane datoteke zamjenjujemo bitovima tajne poruke. Spomenuta tehnika se lako primjenjeuje na slikovne datoteke te se velika količina informacije može sakriti minornim ili potpuno neuočljivim utjecajem na datoteku koja je nositelj skrivenih informacija. Ukoliko želimo sakriti poruku u svaki bajt 24-bitne slike BMP formata, možemo spremiti 3 bita tajne poruke u svaki piksel. Tako čineći, u sliku 1024x768 piksela može sakriti $1024 \cdot 768 \cdot 3 = 2359296$ bitova informacija, što je ekvivalent 294 912 bajta ili 288 kilobajta informacija. Iako je riječ o relativno velikoj količini podataka stegoobjekt ili stegoslika ljudskom oku izgleda identično kao i orginal.

Ukoliko želimo demonstrirati tehniku pretpostavimo da želimo sakriti slovo A. ASCII ekvivalent slova A iznosi $65_{(10)}$, što je binarno $01000001_{(2)}$. S obzirom da je riječ o 8 bitova podatka za skrivanje će nam biti potrebno 8 bitova slika, što je izvedivo ako upotrijebimo 3 piksela slike. Sjetimo se kako svaki piksel slike sadržava 3 bajta, svaki za jednu od temeljnih boja : crvena, zelena i plava. Ukoliko su prvi odabrani pikseli slike :

(00100111,	11101001,	11001000)
(00100111,	11001000,	11101001)
(11001000,	00100111,	11101001)

Upotrebom ove stegograske metode i ubacivanjem binarne vrijednsti slova A u sliku, točnije u 3 piksela slike rezultiralo bi sljedećim promjenama :

(0010011**0**,	11101001,	11001000)
(0010011**0**,	11001000,	1110100**0**)
(11001000,	00100111,	11101001)

Primjetimo kako su samo 3 bita od 9 (koliko tvore 3 piksela) promjenili vrijednost kako bi slovo A postalo dio slike.

Statistički gledano, primjenom ove steganografske metode promijeni se polovica najmanje značajnih bitova. Piksel koji se najviše promjeni, promjeni se za samo dvije nijanse boje. Ako to usporedimo s činjenicom postojanja 17.6 milijuna nijansi očito je promjena beznačajna po ljudsko oko. Čak i kada bismo tajnu poruku skrivalu u 2,3 ili 4 najmanje značajna bita prikrivene datoteke, ljudsko oko i dalje bi teško moglo primjetiti razliku stegoslike u odnosu na orginal.

Dakle, ako želimo sakriti tajnu poruku unutar slikovne datoteke koristeći LSB steganografsku metodu, prvi korak je odabiranje prikrivene datoteke. U prvom koraku je važno napomenuti kako slike koje imaju velike površine istih nijansi boja su loš izbor za ulogu podloge i stegoslike. Očito, svaka promjena na takvoj slici će biti lakše uočljiva nego na slikama koje su po svome sadržaju dinamičnije i kvantitativnije. Točnije, najveće promjene mogu se opaziti na jednobojnim slikama. S druge strane, slike razobojnog krajolika može poslužiti kao dobar izbor za podlogu i stegosliku. Prvi korak odabira slike koja zadovoljava navedene uvjete vrlo je bitna stvar pri kreiranju stegoslike. Drugi korak sadržava izbor podskupa najmanje značajnih bitova prikrivene datoteke. Broj bitova u odabranom skupu mora odgovarati broju bitova tajne poruke, osim u slučaju dodatnih zaštitnih mehanizama. Tada ranije definiranom algoritmom od strane pošiljatelja i primatelja, svaki bit stegoslike zamjenjujemo bitom tajne poruke. Najjednostavnije slučaj LSB suptitucije implicira pohranu tajnih bitova u najmanje značajni bit svakog piksela, redom. Treći korak se odvija na strani primatelja koji rekonstruira poruku. Kako bi rekonstruirao poruku, primatelj mora znati podskup najmanje značajnih bitova u kojima se tajna poruka nalazi. Koristeći tehnike izvlačenja poruke iz najmanje značajnih bitova te poznavanjem određenih algoritama slaganja, dolazi do poruke.

Problem koji se javlja kod LSB metode vezan je za pojam kompresije. Ukoliko koristimo bilo koju „lossy“ kompresiju, primjerice JPEG, najmanje značajni bitovi slike bit će modificirani. Drugim rječima, ukoliko stegosliku pretvorimo u JPEG sliku i iz te verzije pokušamo genreririati tajnu poruku, nećemo uspjeti jer je došlo do gubitka podataka. Također, nužno je na slici ne primjenjivati nikakve tehnike obrade.

4. WEB TEHNOLOGIJE

HTML (eng. HyperText Markup Language) je opisni jezik za izradu web stranica. Kao njegov kreator spominje se Tim Berners Lee, nastao negdje 1991. godine. HTML opisni jezik koristi se za određivanje strukture HTML dokumenta. Svrha HTML jezika jest uputiti web preglednik u prikaz hipertext dokumenta. Težnja je ostvariti isti izgled dokumenta nevoisno o web pregledniku, računalu i operacijskom sustavu koji se koristi. HTML je moguće definirati dvama ekstenzijama a to su .html ili .htm.

HTML dokument se sastoji od osnovnih građevnih blokova. Njih nazivamo HTML elemenatima. Svaki HTML element se sastoji od para HTML oznaki (engl. tags). Svaki element može imati atribute. Svrha atribtua je definiranje svojstva elementa.

HTML dokuement uvijek prati određenu strukturu. Na HTML dokumenta preporučljivo je postaviti <!DOCTYPE> element. Spomenuti element označava DTD (engl. Document Type Declaration) koji definira inačica standarda HTML dokumenta. Nakon inačice HTML dokumenta slijedi html element. Njime se označava početak HTML dokumenta. Unutar html elementa nalaze se head i body element. Unutar head elementa definria se zaglavlje HTML dokumenta u kojemu se specificiraju jezične značajke HTML dokumenta naslov stranice, meta podaci i razni linkovi na datoteke potrebne u projektu. Unutar body elementa kreira se sadržaj HTML dokumenta, struktura stranice.

HTML oznake počinje znakom < a završava znakom > . Zatvarajuća HTML oznaka kreira se na isti način kao i otvarajuća s tim da prije završnog znaka >, sadrži kosu crtu. Osim standardnih HTML elemenata, HTML definira i samozatvarajuće elementi. Samozatvarajući elemenati nemaju zatvarajuće oznake.

HTMl omogućava korištenje linkova koji su osovina rada web stranica. Moguće je dodati i razne multimedijske sadržaje poput slika, audio i video elemanta. Moguće je koristiti liste, tablice, forme i druge osnove elemente HTML koji definiraju strukturu stranice.

Komentari su dio HTML dokuemta koje moguće pisati bilo gdje untar HTML dokumenta bez da utječu na prikaz stranice. Najčešće služe kao upute programerima [8].

HTML nije jedini način strukturiranja dokumenta. Sve pristuniji su view engine, čija svrha je jednostavnija sintaksa, ubacivanje podataka u inače statične strukture te korištenje Javascripta unutar HTML dokumenta.

Neki od najpozntaijih view enigne su EJS (eng. EmbbededJS), Mustache (Handlebars) i Pug (Jade). Kroz ovaj rad korišten je EJS view engine prilikom renderiranja par pogleda, uglavnom sporednih. Riječ je o pogledima prilikom logiranja, registracije, pogrešaka i slično. U takvim situacijama se podaci iz rute odnosno forme, prenose u .ejs dokument i prikazuju. Prilikom korištenje view engine potrebno ih je instalirati korištenjem NPM-a te zatražiti u server skripti pa postaviti kao korišteni view engine u datom projektu. Prilikom korištenja Javascripta unutar HTML u .ejs datoteci potrebno je svaku naredbu okružiti tagovima za oznaku da se radi o Javascript kodu unutra HTML. U EJS se to osigurava pomoću <% i %> oznaka. Ukoliko želimo ispisati nekakvu vrijednost potrebno je korstiti <%= person.name %> sintaksu. Svaki .ejs pogled potrebno je locirati u view direktoriju aplikacije. Znači, svrha EJS je korištenje podataka i ugrađenog Javascript prilikom gnereiranja HTML. Ovo svojstvo omgućuje web stranicama da bude kreirane kao statične a da se renderiraiju dinamički. Kad koristimo .ejs poglede unutar Express aplikacije, potrebno je istu renderirati unutar rute narednom render kao odgovor na zahtjev [4].

CSS (eng. Cascading Style Sheets) je jezik za oblikovanje HTML dokumenta nastao 1996. godine. Prvi standrad je objavljen sredinom 1998.godine i omogućuje odvajanje strukture i pojave (izgleda) dokumenata. CSS razlikuje tri načina korištenja koda za stiliziranje. Razlikujemo linijski, unutranji i vanjski način uključivanja CSS koda u HTML dokuement. Najveći prioritet posjeduje linijski obrazac, zatim unutarnji pa tek onda vanjski obrazac. U slučaju korištenja linijskog CSS-a, određena svojstva se pripisuju svakom elementu zasebno. Ovaj način zapisa ima njaviši prioritet kod stiliziranja HTML dokumenta ali se ne koristi zbgo svoje ne praktičnosti koja se očituje u većim projektima. Koristi se eventualno fazama razvoja i testiranja. Korištenjem unutarnjeg CSS odvaja se HTML od CSS koda. Mana je neomgućnost korištenja koda na drugim stranicama. Ovdje do izražaja dolazi vanjsko uključivanje CSS kao link. Izgled dokumenta se kreira unutar jedne datoteke i onda se ista uključuje u sve HTML datoteke što doprinosi modulranosti i lakšem održvanju.

CSS je i danas u potpunosti u upotrebi. Ono što se mjenja su metode i paradigme pisanja. Također, sve je veći broj preprocesora. Preprocesor (eng. preprocessor) je program koja obrađuje ulazne podatke, na osnovu kojih generira izlazne podatke koji služe kao ulazni podaci drugog programa. SASS/SCSS i LESS su samo neki od preprocesora. Preprocesori sa sobom nose veliku modularnost koda, nova pravila pisanja i sintaksu te neka svojstva programksih jezika, poput miksina koji su identični funkcijama u programksim jezicima [8].

Javascript je skriptni jezik. Uz HTML i CSS služi kao osnova frontend tehnolgija. Javascript je prvensteveno jezik preglednika iako dolaskom NodeJS postaje prisutan i na serveru. Sa svojim svojstvima je sličan drugim objektno orijentiranim jezicima, uz posebna svojstva specifična za web. Čisti Javascript je uistinu rijedak. Postoji velik broj Javascript biblioteka. Svakako najzanimljivije su jQuery za poprilično jednostavniji način rada s DOM (eng. document object model), Angular i slično [4].

Svrha servera je da osluškuje zahtjeve klijenata, obrađuje iste parsirajući zahtjeve i locirajuće resurse te zatim šalje odgovor klijentima sa traženim resursom. NodeJS je multiplatformalno Javascript okruženje otvorenog koda za izvršavanje Javascript koda na strani servera. Povijesno gledajući, Javascript je primarno korišten kao jezik preglednika, odnosno klijentske strane. Javascrpt skripte su ondje pisane i uključivane u HTML web stranice ili aplikacije kako bi bile korištene putem Javascript engine [4].

NodeJS omogućava korištenje Javascript skriptnog jezika prilikom kodiranja na strani servera i pokreće skripte na strani servera s ciljem kreiranje dinamičkog sadržaja web stranice prije nego se stranica učita na klijentskoj strani. S obzirom na navedeno, NodeJS je postao jedan od glavnih elemenata paradigme koja pretendira korištenje Javacript jezika svugdje. Ova činjenica omogućava razvijanje web aplikacije korištenjem samo jednog programskog jezika, radije nego oslanjanje na određeni broj različitih programskih jezia prilikom pisanja skipti na serverskoj strani.

NodeJS je kreiran od stane Ryna Dahl 2009.godine, gotovo 13 godina nakon predstavljanja prvog Javascript serverskog okruženja, Netscape's LiveWire Pro Web. Prvotna verzija bila je pogodna samo za Linux i Mac. Početkom 2010. godine predstavljen je NPM (eng. Node package manager). NPM je olakšao programerima objavu i razmjenu koda NodeJS biblioteka. Kao takav kreiran da pojednostavi instalaciju, nadogradnju i brisanje biblioteka. NodeJS omogućava kreiranje web servera i mrežnih alata korištenjem Javascripta i kolekcija modula koji uspješno upravljaju različitinim osnovnim funckionalnostima. Postoje moduli za datotečni I/O sustav, mrežu (DNS; HTTP, TCP, TLS/SSL ili UDP), binarne podatke, kriptografske funckije, tokove podataka i mnoge druge funckionalnosti. NodeJS moduli koriste API (eng Apllication programming interface) kreiran kako bi smanjio kompleksnost prilikom pisanja serverskih aplikacija. 2011. godine Microsoft implemenitra Windows verziju NodeJS.

Kada govorimo o svojstvima NodeJS bitno je spomenuti kako NodeJS posjeduje arhitekturu pokretanu događajima koja je sposobna za asinkorni rad. Cilj navednog je optimizacija i skalabilnost web aplikacija korištenem puno ulazno/izalaznih operacija, omogućavajući rad aplikacija u stvarnom vremenu. Neke od tvrtki koje koriste NodeJS su : GoDaddy, groupon, IBM, LinkedIn, Microsoft, Netflix, PayPal, Yahoo i drugi.

NodeJS može se pokretati na Linux, Mac, Windows, NonStip i Unix serverima. Alternativa prilikom pisanja server skripti u NodeJS vezano uz sintaksu može biti CoffeScript, TypeScript ili bilo koja druga sintaksa koja se može prevesti u Javascript. NodeJS se primarno koristi prilikom kreiranje mrežnih programa poput web servera. Najveća razlika između NodeJS i drugi serverskih jezika poput PHP je da većina funckija u PHP je blokirana dok se ne izvrše. U slučaju NodeJS, funckije se kreirane kako bi bile neblokirajće, izvršavaju se paralelno.

ExpressJS je web okvir (eng. framework) koji funckionira na osnovu NodeJS-a a kreiran je na temelju Connect HTTP server okvira. Express omogućava modularnost web aplikacija definiranjem vlastitih i odvojenih modula sa raznim zadaćama. Moduli se mogu koristiti kao globalni ili lokalni. Svakako, bit će dodani na listu ovisnosti (eng. dependency) u package.json file. ExpressJS omogućava rukovanje raznima zahtjevama koristeći rute (eng. routes) i rutere (eng. router). Pruža mogućnost raspodjele svih potrebnih koraka obrade zahtjeva u međuslojeve (eng. Middleware) te mogućnost korištenja view enigne [6].

MongoDB je nerelacijska baza podataka koja ne koristi SQL, tradicionalnu metoda interakcije s bazom i podacima. Kod relacijskih baza podataka, podaci su spremljenu unutar tablica i redova. Svrha NoSQL baze podataka je zaobilaženje nekih mana relacijskih baza podataka i organiziranje pohrane podataka drugačije. MongoDB je jedna od nerelacijskih baza podataka koja je napravljena za korištenje s Javascript aplikacijama. MongoDB pohranjuje podatke kao kolekcije dokumenata a ne kao tablice i retke. Sam izgled je identičan izgledu Javascript objekta i to je prava svrha takve baze podataka. MongoDB je moguće koristi online ili instalirati lokalno [9].

Node modul za lakše korištenje baze je Mongoose. Potrebno ga je instalirati i uvesti u server skriptu. Mongoose koristi Scheme prilikom interakcije s bazom podataka. Schema zarpavo detaljno definira način pohrane podataka i model baze. Schema može biti više i mogu bit jedna unutar druge. Baš poput objekata [10].

5. WEB APLIKACIJA

Kroz sljedeće poglavlje biti će objašnjeno programsko rješenje diplomskog rada. Safechat je naziv web aplikacije nastale kao produkt diplomskog rada. Ponovimo kako se za izradu grupne chat aplikacije koriste HTML, CSS i Javascript kao frontend tehnologije. Backend tehnologije su NodeJS i njegov modul ExpressJS, koji se koristi oslonjen na NodeJS. Baza podataka je MongoDB i njen modul Mongoose. Riječ je o nerelacijskoj bazi podataka koja se u pravilu vrlo često koristi u kombinaciji s NodeJS tehnologijom [11].

Ideja web aplikacije je omogućiti sigurnu komunikaciju. Točnije, omogućiti sigurnu komunikaciju upotrebom sigurnosnih metoda kriptiranja i steganografije. Cilj je generalno sakriti postojanje bilo kakve poruke metodom steganografije. U praksi to znači kako će internet vezom između servera i klijenata putovati slika. Maligni promatrač podatkovnog prometa to će svakako primjetiti. Ukoliko shvati da je riječ o slikama koje se konstanto transportiraju i to sa servera koji je namjenjen komunikaciji, logično je za pretpostaviti kako će primjeniti određene steganografske metode kako bi izvukao informacije iz slike. Ovdje u prvi plan dolaze metode kriptografije. Poruka koju klijent šalje u Safechat aplikaciji se osim steganografski, osigurava i kriptografski. Sadržaj poruke podliježe metodi kriptografije kako bi se dodatno zaštitio. S obzirom na već spomenute metode kriptoanalitičkih napada, nije baš jednostavno doći do izvornog sadržaja. Naravno, razina sigurnosti poruka unutar Safechat aplikacije leži u tim dvjema metodama i što su one razvijene i modernije sigurnost je veća. Safechat aplikacija koristi stegaografsku metodu LSB (eng. least significat bit) koja bitove poruke pohranjuje u najmanje značajne bitove slike i time proizvodi minoran učinka na izgled slike. Implementiranjem ove metode može se zaključiti kako nije presudno korištenje ove metode osim u slučaju da se izvorna slika otkrije trećoj osobi. Ukoliko je to slučaj, maligni promatrač može usporediti izvornu sliku i stegoobjekt i pronaći razlike u bitovima. Nakon toga preostaje još dekriptirati šifrat u otvoreni tekst. Kada to nije slučaj, promatraču ostaje zadaća otkrivanja lokacije otovorenog teksta ili šifrata unutar stegoobjekta. Korištenje metoda kriptografije i steganografije bit će detaljnije objašnjeno u nastavku teksta.

Prilikom izrade ovakve vrste aplikacije nužno je kreirati server. NodeJS je podloga na kojoj je server konsturiran. NodeJS je potrebno skinuti i instalirati na računalo. Isto je nužno učiniti i sa MonogDB. Nakon instalacije potrebno je projekt inicirati u određeni direktorij. To se izvodi narednom „npm init“ kroz termina, unutar odabranog direktorija koji je lokacija projekta. Pokretanjem spomenute naredbe otvara se opis projekta unutar terminala. Ondje je nužno

definirati neka svojstva poput nazvia projekta, opisa, server skripte, naprednih funckija ali je i moguće postaviti sve potreben module, odnosno ovisnosti (eng. dependencies). Nakon toga potrebno je pokrenuti naredbu „npm install“ koja će instalirati sve upisane module.

Izvanredno svojstvo NodeJS-a kao tehnologije je mogućnost korištenja modula. Prvenstveno je ovdje naglasak na korištenje eksternih node modula, iako je moguće kreirati i vlastite. Par modula je gotovo neophodno prilikom kreiranja servera i oni drastično smanjuju razinu kompleksnosti. Module je moguće dohvatiti globalne ili lokalne, koristeći naredbu „npm install –save express -g“. Opcija „-g“ je dodatna i označva globalnu instalaciju modula, dok opcija „—save“ sprema modul na listu modula u package.json, datoteku koja je nastala iniciranjem projekta.

Među potrebnim modulima, prvi je svakako ExpressJS. Nad tim modulm se zatim instancira sama Express aplikacija nad kojem se onda vrši kreiranje servera i puštanje servera u rad na nekom od portova. Safechat koristi port 3000 na kojem osluškuje zahtjeve klijenata. Sljedeći korak je svakako kreirati prvu rutu unutar aplikacije. Ruta se kreira nad instancom Express aplikacije kao i većina drugih funkcionalnosti. Ruta se definira kao „GET“, „POST“ ili „USE“ zavisno o samoj prirodi rute. Je li njena svrha generalna, prima li podatke pomoću zahtjeva ili je rezultat operacije nad formom specifične namjene. Prva i osnovna ruta je ruta „/“ koja označava index.html, odnosno osnovni zahtjev klijenta prema serveru. S obzirom na privilegiju korištenja Javascripta na server strani koju omogućava NodeJS, funkcije povratnog poziva su više nego zastupljene. Funkcija povratnog poziva je dio svake rute. Unutar osnove rute koja predaje osnovni odgovor, čeka se na zahtjev klijenta prema serveru. Riječ je o asinkronoj funckiji što znači da ista ne ometa rad programa nego reagira tek kad do nje pristigne nekakav događaj. Zatim reagira i program opet nastavlja svojim tokom.

HTTP (eng. HiperText Transfer Protocol) je dominantni protokol na internetu. Njegova svrha je osiguravanje komunikacije između dva entiteta na internetu. Bilo da je riječ o računalima, mobitelima, programima međusobno ili nekoj drugoj vrsti komunikacije. HTTP je koncipiran na način da ostvaruje komunikaciju putem dvaju objekata. Riječ je o objektima „request“ i „response“, odnosno zahtjev i odgovor. Klijent šalje zahtjev prema serveru s ciljem dobivanja nekakvog resursa. Prilikom slanja do servera putuje ime protokola, verzija protokola, način dohvata kao i razni drugi meta podaci. Server tada zaprima zahtjev, preusmjerava ga na određenu rutu u kojoj se zahjev obrađuje i server reagira s obzirom na postavke zahtjeva. Ukoliko je sve u redu reagira vraćanjem meta podataka i traženim resursom. Ukoliko je došlo do

nekakve pogreške, pokreću se mehanizmi djelovanja koji obrađuju svaku novonastalu situaciju. Ili bi barem trebali. Inače je ova paradigma poznata kao komunikacija klijent-server a može se i drugačije definirati zavisno od potrebe. Vjerojatno najvažniji aspekt komunikacije po server su POST forme od strane klijenata. Pomoću njih server dobiva važne informacije koje su u prinicipu tajne i kojima je mjesto često u bazi podataka. Prilikom rukovanja POST formama sa strane servera potreban je novi Node modul naziva „body-parser“. Podaci iz POST formi dolaze do servera kao dio tijela poruke. Kako bi se izvukao sadržaj iz tijela, koje je inače komplicaran, moguće je upotrjebiti navedeni Node modul i olakšati posao. S obziom da su zahtjev i odgovor objekti, nazivu korisnika iz POST forme moguće je pristupiti narednom req.body.name.

HTTP je protokol koji funckionra na način da klijent traži određeni resurs, server zaprima zahtjev i šalje odgovor koji sadržava dotični resurs [7]. Nakon toga se veza ili komunikacija prekida do trenutka kada klijent ponovo šalje zahtjev. Postoje određene metode za drugačije djelovanje. AJAX (eng. Asynchronous Javascript and XML) je metoda koja omogućava asinkronu komunikaciju sa serverom. AJAX omogućava asinkrono dohvaćanje resursa sa servera bez potrebe da se stranica ponovno učitava. Ova funkcionalnost je veliki napredak u svakom pogledu. Međutim, mana je i dalje prekid komunikacije. Nakon što klijent dobije resurs ili dio resursa, komunikacije se prekida.

Kada govorimo o svojstvima nužnima za kreiranje jedne chat aplikacije, osnovo svojstvo je konstantna komunikacija. Ovdje do izražaja dolazi još jedan Node modul a riječ je o web socketu „socket.io“. Socket.io spada u grupu TCP protokola. TCP (eng. Transmission Control Protocol) kreira virtualnu vezu prema drugom objektu komunikacije i zatim prenosi podatke. Ovo svojstvo ga čini dijelom grupe protokola koji se nazivaju spojni protokoli. TCP osigrava pouzdanu isporuku podataka do odredišta u kontroliranom redoslijedu od pošiljatelja prema primatelju. Također, omogućava višestruka povezivanja prema jednoj aplikaciji od strane većeg broja klijenata. Najvažnije svojstvo je priroda uspostavljene veze. Naime, riječ je o dvosmjernoj vezi koja konstantno ostaje otvorena. Za razliku od HTTP, veza ostaje otvoreno dokle god se ne prekine uslijed programske rutine ili od strane korisnika.Nakon dodavanja Node modula za socket komunikaciju, kreira se socket objekt koji se povezuje sa serverom kako bi osluškivao zahtjeve.S obzirom na implementaciju aplikacije, detalji socket komunikacije bit će opisani u nastavku.

Nakon kreiranja osnovnih elemenata ExpressJS servera poput same instance servera, ruta i načina komunikacije, potrebno je definirati i sam izgled aplikacije koji će biti prezentiran

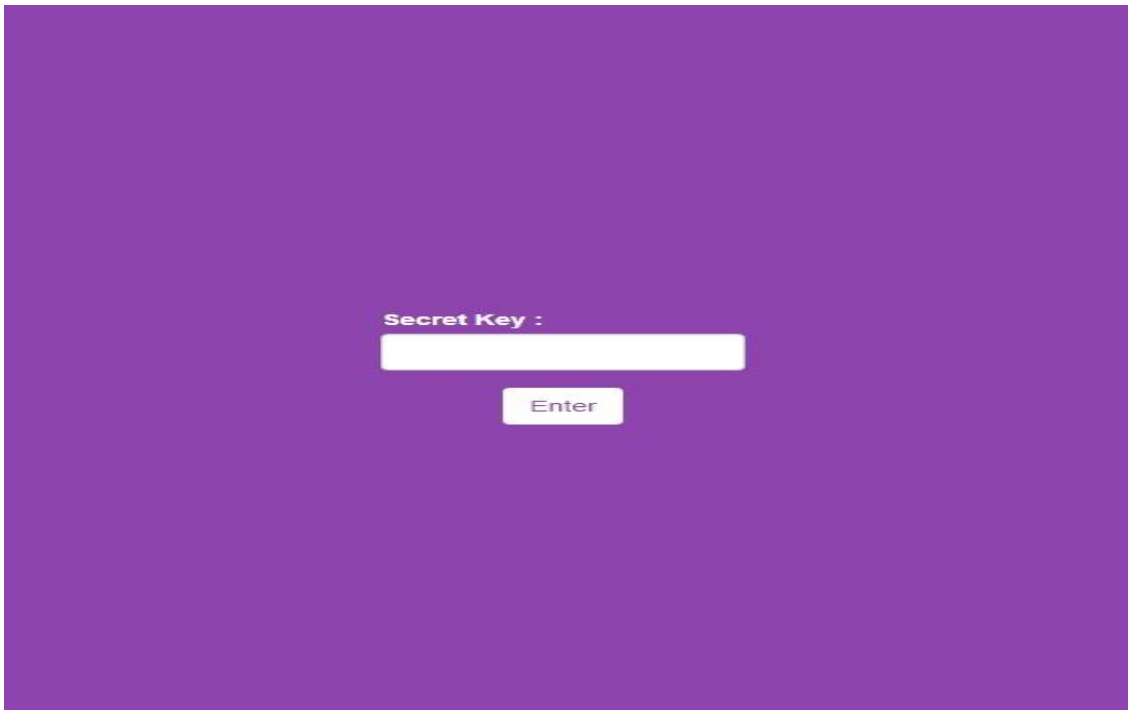
zavisno od stanja. Unutar rute se definiraju odgovori koji prvotno obrađuju negativne rutine poput raznih errora. Nakon toga se obrađuje rutina pozitivnog odgovora. Postoje razne mogućnosti odgovora no kroz ovaj rad bit će prezentirano renderiranje HTML stranice i pojedinih EJS (eng. Embedded Javascript) stranica kao odgovora na zahtjev klijenta. EJS je jedan od view enigne koji prvenstveno dozvoljavaju pisanje Javascript koda unutar HTML-a. Prilikom definiranju odgovora klijentu unutar rute, moguće je proslijediti objekt .ejs dokumentu koji sadrži Javascript elemente. Ovo svojstvo omogućava dinamičko djelovanje i prikaz, inače statičnih elemenata.

Međuslojevi (eng. middleware) su funkcije koje se koriste nakon primanja zahtjeva a prije proslijeđivanja odgovara klijentu. Njihova funckija je obrada i djelovanja nad informacijama od strane klijenata kao i još neke druge funckije. Jedna od njih i postavljanje direktorija kao primarnog za čitanje statičnih datoteka, koje uključuju slike, CSS, HTML i Jacascript datoteke.

5.1. Izgled aplikacije

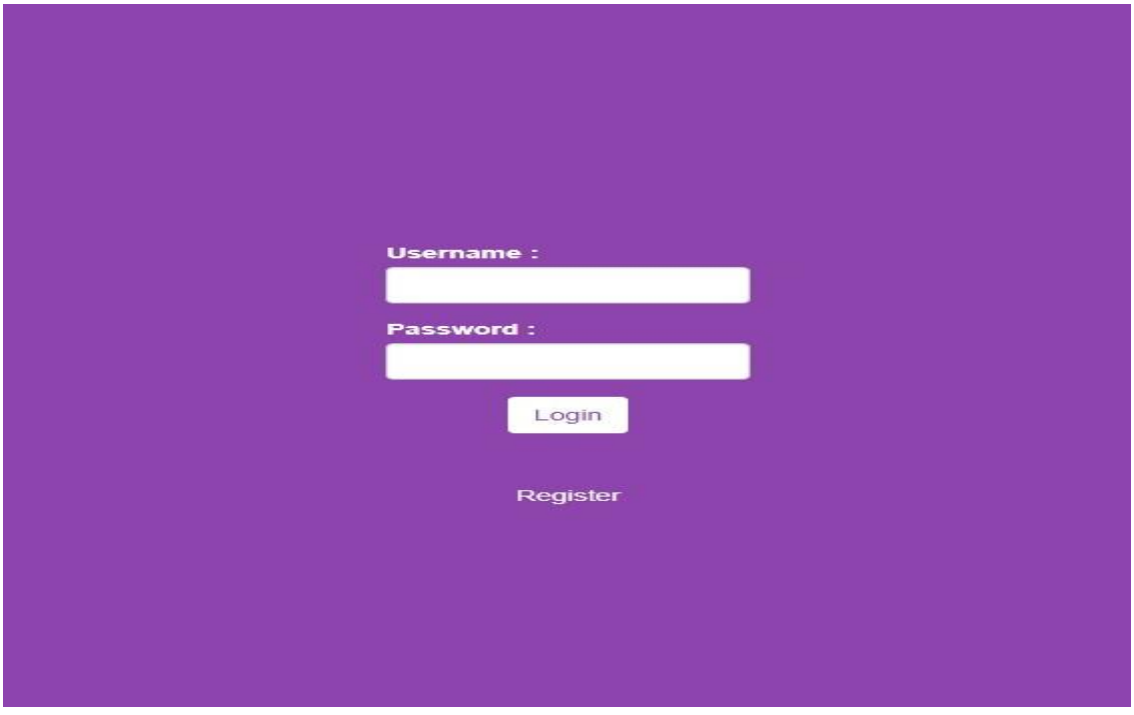
Safechat aplikacije je rađena prema paradigmi „mobile first“ sa idejom mobilnog razvoja i responzivnosti same aplikacije. Dizajn aplikacije je sadržan u style.css datoteci. Datoteka se nalazi unutar stylesheet direktorija koji je opet dio public direktorija uz images i javascript direktorije. Public direktorij je postavljan kao podrazumijevani prilikom korištenja statičnih datoteka pomoći middleware-a.

Zahtjev klijenta prema podrazumijevanoj rutu „/“ vraća cipher.html. Riječ je o HTML datoteci koja je osnova i ulaz u aplikaciju. Izgled pogleda je forma koja zahtjeva unos tajne riječ. Ovdje se želi staviti naglasak na simetrične metode kritpografije koje su uvijek koristile tajni ključ koji je morao biti dogovoren ranije na siguran način. Ovo je prvi korak zaštite i to od neovlaštenog ulaza u aplikaciju. Forma se verificira na klijent strani i ukoliko korisnik ne unese tajnu riječ „enigma“ neće moći pristupiti aplikaciji. Iako bi puno mudrije bilo provjeriti unos na server strani, verifikacije se radi na klijent strani kako ne bi doslo do napada na server u vidu visokog broja zahtjeva prema serveru. Ovdje se prvi put primjenjuje i Javascript skripta koja obrađuje unos u formu. Ukoliko korisnik upiše traženu riječ, pokreće se metoda GET prema login.html datoteci.



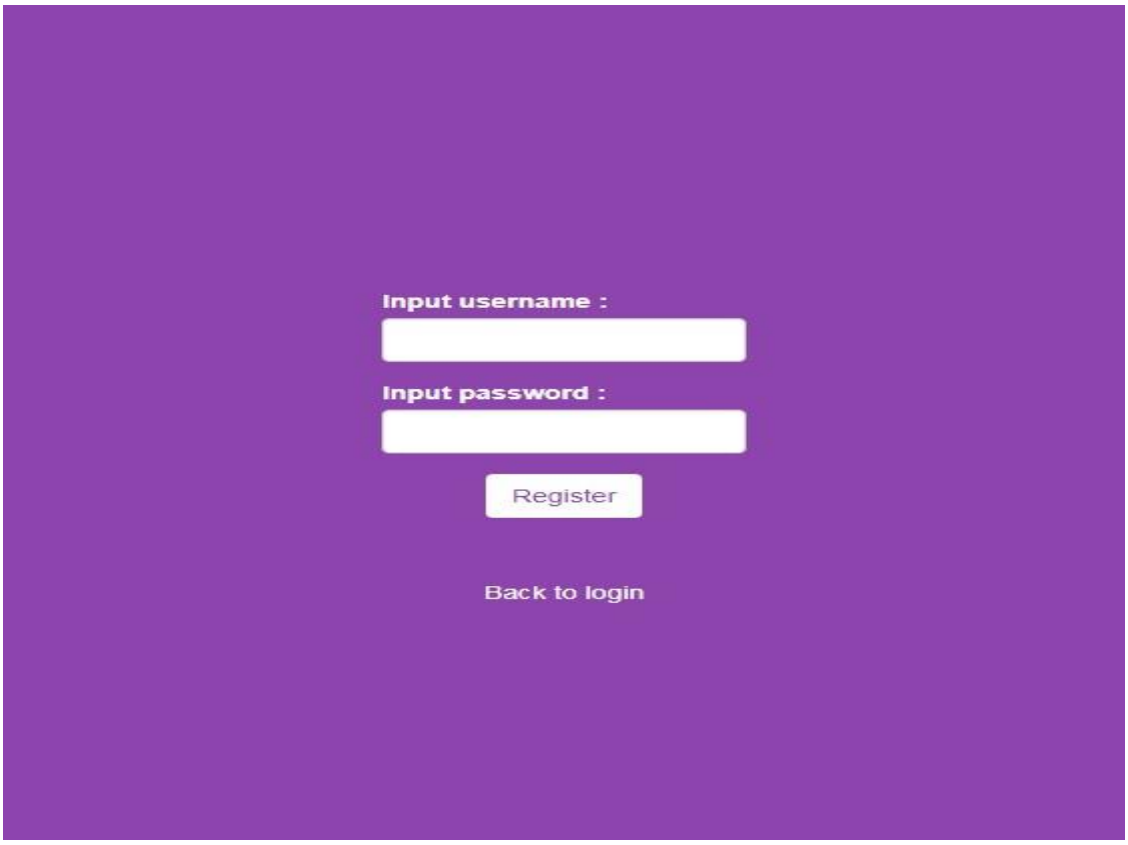
Sl. 5.1. Ulaz u aplikaciju

Kada se korisnik uspješno ulogira u aplikaciju dolazi u login.html. Izgled pogleda sastoji se od forme za unos korisničkog imena i prikladne lozinke. Logično je pretpostaviti kako je prije logiranja nužno obaviti registraciju. Link naziva „Register“ vodi do register.html datoteke.



Sl. 5.2. Logiranje korisnika

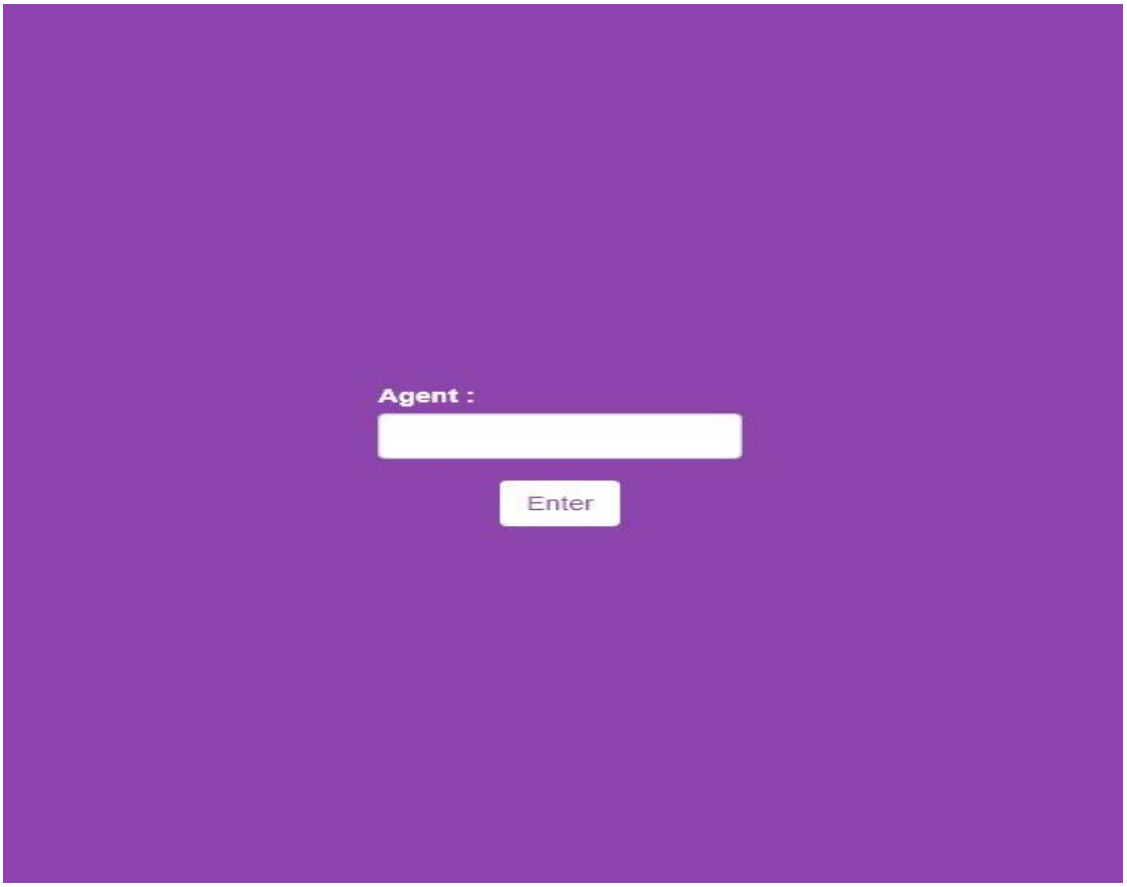
Forma za registraciju zahtjeva unos korisničkog imena i lozinke. U ovom koraku se demonstira korištenje baze podataka. Riječ je o nerelacijskoj bazi podataka MongoDB koja podatke sprema kao kolekcije dokumenta, odnosno objekte. Kolekcije objekata. Node modul Mongoose olakšava interakciju s bazom podataka. Unutar direktorija „models“ kreira se skripta „users.js“. Njena svrha je kreiranje modela baze. Koristi se spomenuti Node modul Manogoose koji se povezuje s bazom. Kao parametar predajemo port baze i ime baze. Port baze se iščitava nakon što se MongoDB instalira na računalo, kreira direktorij za pohranu podataka i pokrenu „monogod.exe“ i „monogo.exe“ terminali. Unutar prvog je moguće pogledati port a unutar drugog je moguća interakcija s bazom podataka. Nakon proslijeđivanja porta, proslijeđuje se naziv baze. Baza se može ranije kreirati kroz terminal. Ako to nije slučaj, aplikacija će sama kreirati bazu sa proslijeđenim imenom. Nakon toga se definira shema. Shema je svojstvo Mongoose modula koja omogućuje stvaranja modela, odnosno objekta u našoj bazi. Safechat definira shemu naziva „personShema“ koja posjeduje username kao obavezni, jedinstveni podatak tipa string te password kao obavezni podatak tipa string. Nakon toga se shema exporta, iznosi van skripte narednom „exports.module“ i dodjeljuje joj se naziv Person. Export module je još jedno bitno svojstvo NodeJS koji omogućava iznos određenih varijabli, funkcija ili čitavih skripti u globalni doseg gdje ih onda drugi dijelovi aplikacije mogu koristiti ukoliko ih uvezu (eng. import) unutar skripte. Izvoz (eng. export) omogućava kreiranje vlastitih modula. Nakon što je model u bazi stvoren, moguće je dodavati podatke u bazu. Forma za registraciju korisnika je potpuno slobodna i jedini uvjet je da korisničko ime mora biti jedinstveno. Nakon ispunjvanja forme za registraciju pokreće se POST metoda prema ruti „/registered“ koja zatim upisuje korisnika u bazu i vraća pogled s informacijama o registraciji. Prikaz je ostvaren korištenjem EJS gdje je dinamički element ime korisnika registriranog u bazu. Nakon uspješnog registriranja i poruke o istom, linkom se korisnik vraća na login dio aplikacije gdje sad može unjeti svoje podatke.



Sl. 5.3. Registracija korisnika

Korisnik unosi podatke u login formi koja metodom POST poziva rutu „/chat“. Unutar rute se parsiraju podaci forme pomoću body-parsera i iz zahjeva forme se iščitavaju korisničko ime i lozinka. Nakon toga se unutar rute vrši pretraživanje baze s ciljem pronalaska korisnika unutar nje. U slučaju negativnog odgovora, korisnik se o njemu informira i ponovno vraća na login formu. Ukoliko se korisnika pronađe unutar baze s unešenim podacima, proslijeđuje se na novi i posljednio dio aplikacije – chat.

Kada korisnik dođe u chat dio aplikacije prvo ga očekuje forma koja traži unos imena agenta. Ovaj dio također pridonosi općenitoj sigurnosti i omogućuje korištenje tajnog imena prilikom komunikacije. Nakon što korisnik unese proizvoljno ime, prikazuje se i sam chat pogled.



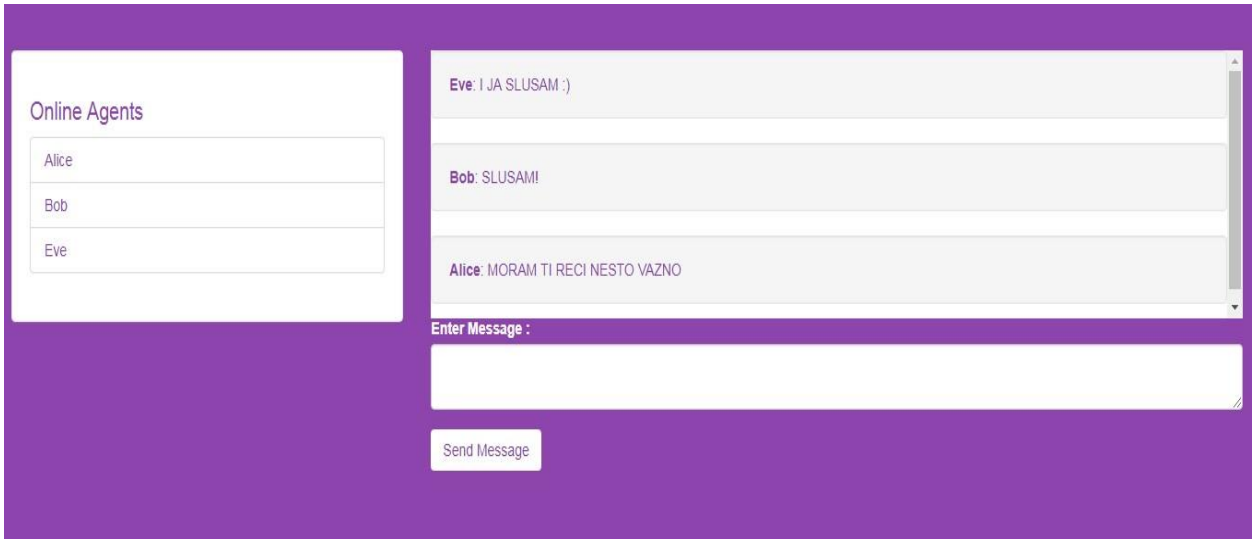
Sl. 5.4. Unos tajnog imena

5.2. Prijenos poruka

Chat pogled se sastoji od elementa za prikaz agenata pri vrhu. Unutar njega se redoslijedom spajanja na aplikaciju dodaju imena novih agenata, jedan ispod drugog. Ovdje do izražaja dolazi socket komunikacija. Nakon što se pokretanjem Safechat aplikacije otvori socket komunikacija, ona čeka na događaje. Događaje se obrađuje korištenjem dvaju elemenata „on“ i „emit“. Koristeći aplikaciju, prvo na red doalzi obrada imena agenata. Nakon što korisnik unese svoj tajni naziv u formu, forma se predaje. Prilikom predaje forme pokreće se prvi socket događaj. Unutar preglednika na klijent strani emitira se događaj „new agent“ koji je zapravo funkcija povratnog poziva i koja čeka na unos imena agenta. Kad se unese ime agenta, ono se emitira, šalje do servera. Nadalje, server posjeduje funckiju pozivajuće poziva „on“. Spomenuta funckija kao parametar prima događaj „new agent“, odnosno čeka na njega. Kada se događaj ostvari s njime dolazi i naziv agenta. Naziv agenta se sprema u polje s nazivima svih trenutno aktivnih

agenata u aplikaciji. Server tada poziva funckiju za ažuriranje popisa agenata u aplikaciji. To čini emitirajući događaj „get agents“ uz koji šalje polje s nazivima agenata. S druge strane, klijent koristeći funkciju povratnog poziva čeka na događaj „get agents“. S tim događajem dolazi i polje s svim imenima agenata koji se zatim ispisuju na pregledniku kao elemetni liste koristeći blagodati Javascripta. Ovim postupkom se konstantno ažurira lista aktivnih agenata pomoću socket funkcija „on“ i „emit“ koje emitiraju, odnosno čekaju na određeni događaj.

Nakon što su se korisnici Safechat aplikacije uključili na chat s tajnim imenom, počinje grupna chat komunikacija. Centralni element chat aplikacije je element za prikaz poruka. Isti je definiran kao DIV sa maksmialnom visinom od 220px nakon čega mu se dodaje klasa za scroll overflow. Kada se nakupi velik broj poruka do njih se može pristupiti skrolanjem prema dolje. Poruke se dodaju kao prvo dijete liste, što znači da svaka nova poruka postaje gornja na listi. Poruke se unose u formu koja se sastoji od HTML elementa textarea i gumb za slanje poruke. Poruka se sastoji od imena agenta koja je vlasnik poruke i same poruke.



Sl. 5.5. Izgled chat aplikacije

Najznačajniji dio same Safechat aplikacije je slanje poruka. Ranije je naznačeno kako se prilikom slanja i primanja poruka koriste metode steganografije i kriptiranja. Prvo, prilikom slanja i primanja poruka se koriste iste metode kao i kod unosa imena agenata. Koristi se socket modul sa funkcijama „on“ i „emit“. Klijent unosi poruku u formu i poziva akciju za slanje klikom na gumb. jQuery metodom se sadržaj poruke uzima iz textarea elementa i isti se čisti za unos nove poruke. Poruka je tada definirana kao otvoreni tekst koji je korisnik unio. Nakon toga, poruka se kriptira kriptografskom metodom naziva „Ceaser cipher“ odnosno, Cezarov šifrat. Metoda uzima otvoreni tekst i svaki znak pomjera za 13 mjesta, što je napoznatiji oblik ove

metode. Ovo pravilo vrijedi za sve znakove osim interpunkcijskih, zbog jednostavnosti. Curenje podataka je svojstveno ovoj metodi i ono se svakako negdje mora dogoditi tako da ova činjenica ne predstavlja velik problem. Nakon što se poruka iz otvorenog teksta prebacila u šifrat, ista se sprema u sliku steganografskom metodom. Za steganografsku metodu se koristi Javascript biblioteka „Steganography.js“.

Vrlo bitno za napomenuti je kako Node posjeduje velik broj modula. Od kojih postoji desetak modula za steganografiju. Modulima poput „Browserify“ moguće je Node module pretvoriti u Javascript budle koji je nakon toga moguće koristiti na klijent strani. Ovo je revolucionarna misao ali posjeduje jednu manu. Samo i isključivo moduli koji u sebi ne sadržavaju druge module mogu biti dio takvog bundle-a. Većina steganografskih modula koristi Node modul za datotečni sustav, što ih čini neupotrebljivim na strani klijenta.

S obzirom na samu konstrukciju aplikacije ova činjenica nije bila fatalna. Ideja je osigurati sigurnu komunikaciju poruka cijelim putem u nesigurnom mediju, od klijenta do servera pa opet do svih klijenata aktivih u Safechat aplikaciji. Stoga, metode kriptiranja i steganografije se koriste isključivo na klijent strani. Razlog leži u tome da se poruke osiguravaju kod klijenta, šalju nesigurnim medijem do servera. Ondje se zaprimaju kao stegoobjekti i emitiraju kao stegoobjekti svim klijetima. Tako čitavm putem poruka biva sigurna. Kada dospije do svih klijenata ondje se razabire unutar slike obrnutom steganografskom metodom i dekripiira u otvoreni tekst. Poboljšanje cijelog postupka bi bilo u mjenjanju metoda kriptografije i slika korištenih za steganografiju prilikom određenog broja slanja ili konstantno na serveru.

Chat pogled sadrži element „main“ čije je svojstvo prikaza postavljano na nevidljivo. Unutar njega se nalazi slika imena „test.jpg“ čija lokacije je u /public/images. Slika je dio klijent strane i koristi se za steganografiju. Slika se obrađuje steganografskom metodom i sprema u drugi HTML element na chat prikazu identičnih dimenzija i svojstava koji je također nevidljiv u Safechat aplikaciji. Ta druga, obrađena slika se zatim emitira do servera pomoću socketa kao „send message“. Server zaprima stegoobjekt unutar događaja „send message“ i proslijeđuje ga svima korisnicima aplikacije kao događaj „new message“.

Klijent strana koristeći socket komunikaciju čeka događaj „send message“. Kad preglednik zaprimi događaj „new message“ stegoobjekt se sprema u drugi nevidljivi HTML element i nad njim se pokreću metode obrnute steganografije a zatim i dekriptiranja dobivene poruke. Nakon toga se poruka ispisuje na aplikaciji.

Socket komunikacija je način komuniciranja svakom pojedinog klijenta sa serverom. Kada klijent napusti aplikaciju poziva se metoda za ažuriranje liste korisnika. Sve poruke korisnika ostaje napisane u aplikaciji dok postoji barem jedan korisnik. Kada posljedni korisnik napusti aplikaciju sve poruke nestaju. Također, valja napomenuti i kako svi korisnici trebaju biti logirani u isto vrijeme kako bi vidjeli sve poruke. Ukoliko netko zakasni neće vidjeti starije poruke. Moguće je poruke spremati u bazu i vidjeti ih kao stalne ali s obzirom na svrhu aplikaciju ne bi imalo smisla jer je ideja potpuna aninimost i zaštita.

6. ZAKLJUČAK

Safechat je web chat aplikacija. Njena svrha je osigurati sigurnu komunikaciju upotrebom kriptiranja i steganografije. Aplikacija se koristi za grupni chat, razmjenu poruka među više korisnika na jednom mjestu. Komunikacija putem socketa je izvrsna za komunikaciju u stvarnom vremenu i pokazala se kao odlična. Izgled aplikacije je rađen s obzirom na „mobile first“ paradigmu i vrlo je jednostavan. View engine je EJS koji je korišten na potrebim mjestima. Možda bi Pug bio bolje rješenje s obziom na redundanciju programskog koda. S obzirom na kompleknsost aplikacije to nije ni nužno.

Ideja Safechat aplikacije je potpuna anonimost i tajnovitost. Današnje metode kriptografije su nevjerojatne i koriste napredne matematičke funckije. Kroz ovaj diplomski rad je pokazana jedna od naprimitivnijih ali i dugo korištenih, Cezarove šifre. Napredak je moguć korištenjem naprednijih metoda, kako simetričnih tako i asimetričnih. Steganografija je sve zanimljivija širokoj masi ljudi i ovdje je prezentirana njena osnovna i najučestalija izvedba pomoću najmanje značajnih bitova slike. Poboljšanje je svakako moguće u smislu promjena slika, dimenzija i metoda pohrane unutar slike. Generalno, obje sigurnosne metode je moguće ekstremno unaprijediti ali za svrhu ovog rada su prikladne.

Raduje korištenje Javascripta na klijent i server strani. Node modula je sve više i sve su napredniji. Svakako je potrebno omogućiti optimalnije korištenje node modula i na klijent strani. S obzirom na dosadašnju praksu korištenja različitih jezika prilikom razvoja na klijent i server strani, razumljivo je da ova domena još nije razvijena ali vjerujam da će ubrzo biti.

Safechat aplikacija osigurava sigurnu komunikaciju, posjeduje sve elemente chat aplikaciju, upotrebljava interakciju s bazom podataka i koristi view engine. Po svojim specifikacijama i sposobnostima zadovoljava postavljeni cilj uspostavljanja sigurne komunikacije između dva ili više korisnika pomoću metoda steganografije i kriptiranja.

LITERATURA

[1] „Internet history, technologies and security“, Coursera, 2017 [online]. Dostupno na: https://www.coursera.org/learn/internet-history/home/welcome [10.travanj 2017]

[2] I. Lukić: NodeJS, Napredno web programiranje – predavanja, FERIT Osijek, 2017. [12.rujan 2017]

[3] I. Lukić: NodeJS, Napredno web programiranje – Laboratorijska vježba 4, FERIT Osijek, 2017. [11.rujan 2017]

[4] I. Lukić: ExpressJS, Napredno web programiranje – predavanja, FERIT Osijek, 2017. [15.rujan 2017]

[5] I. Lukić: ExpressJS, Napredno web programiranje – Laboratorijska vježba 6, FERIT Osijek, 2017. [16.rujan 2017]

[6] „Programming for the web with javascript“, ED-X, 2017 [online]. Dostupno na: https://courses.edx.org/courses/course-v1:PennX+SD4x+2T2017/course/ [21.rujan 2017]

[7] „Introduction to computer science“, ED-X, 2017 [online]. Dostupno na: https://courses.edx.org/courses/course-v1:HarvardX+CS50+X/course/ [10.rujan 2017]

[8] „HTML, CSS and Javascript for web developers“, Coursera, 2017 [online]. Dostupno na: https://www.coursera.org/learn/html-css-javascript-for-web-developers/home [05.lipanj 2017]

[9] „NodeJS“, W3Schools, 2007 [online]. Dostupno na: https://www.w3schools.com/nodejs/ [21.rujan 2017]

[10] „MongoDB“, W3Schools, 2007 [online]. Dostupno na: https://www.w3schools.com/ nodejs_mongodb.asp / [21.rujan 2017]

[11] Mike Wilson, Node Application with MongoDB and Backbone, O'Really Media, Gravenstein Highway North, Sebastopol, CA 95472.2012

SAŽETAK

Sigurnu komunikaciju moguće je uspostaviti na razne načine. U radu se prezentira zaštita upotrebom kriptiranja i steganografije. Kriptografija koristi razne metode zaštite otvorenog teksta. Kroz rad je prikazan simetrični kriptosustav sa tajnim ključem, naziva Cezarov šifrat. Steganografska metoda skrivanja poruke u sliku naziva se LSB (eng. least significat bit). Web aplikacija je ostvareno upotrebom HTML, CSS, Javascript i NodeJS web tehnologija.

Aplikacija je realizirana korištenjem NodeJS i ExpressJS na server strani uz razne pomoćne module koji osiguravaju perzistentan rad servera. Spajanje na MongoDB bazu podataka ostavaruje se korištenjem Node modula Mongoose. Struktura aplikacije je odrađena u HTML i EJS tehnologijama, dok je izgled napravljen u CSS. Javascript je korišten za pisanje koda klijent i server strane.

Kroz rad je vidljivo kako obje metode zaštite pružaju nevjerojatan osjećaj sigurnosti. Obje metode, iako su primitivne pružaju dobre rezultate. Cezarova metoda kriptiranja stvara šifrat koji nažalost posjeduje ugrađenu manu curenja podatka te je zbog te činjenice podložan kriptoanalitičkom napadu. LSB steganografska metoda je po svojoj prirodi vrlo efektivna ukoliko promatrač ne posjeduje izvornu sliku. Mana je korištenje navedene metode kod slika sa kompresijom.

Ključne riječi : Kriptografija, steganografija, Javascript, NodeJS

ABSTRACT

COMMUNICATION USING STEGANOGRAPHY AND CRYPTOGRAPHY

Multiple methods for establishing secure communication exits. This dissertation/paper discusses cryptography and steganography as two such methods. Cryptography utilizes various methods of encrypting plaintext. This work presents the Caesar cypher - a symmetric cryptosystem based on a secret private key.

A steganographic method of hiding a message in a picture is called LSB (least significant bit). A web application is developed by utilizing HTML, CSS, Javascript and NodeJS on the server-side combined with MongoDB and Mongoose. Important to note is that Javascript is used extensively on both the client and the server-side.

This work presents the conclusion that both presented methods provide a comfortable level of security. In a sense, both methods, although rudimentary, give decent results. The Caesar cypher is known to be susceptible to cryptoanalytic attack. The LSB stenographic method is very effective unless the attacker has access to the original photo. A known drawback is applying the method to compressed pictures.

Keywords: cryptography, steganography, Javascript, NodeJS

ŽIVOTOPIS

Hrvoje Hajduković rođen je 09. travnja 1993. godine u Osijeku. Osnovnu školu završava u Dalju 2007. godine i upisuje Prirodoslovno-matematičku gimnaziju u Osijeku. Prirodoslovno-matematičku gimnaziju završava 2011. godine i upisuje prediplomski studij računarstva na Elektrotehničkom fakultetu u Osijeku. 2015. godine završava preddiplomski studij i upisuje blok informacijskih i podatkovnih znanosti u vidu diplomskog studija računarstva. Tijekom završne godine diplomskog studija radi u tvrtki Bamboo Lab.

Hrvoje Hajduković

PRILOZI

Izvorni kodovi

Prilog 1 Ceaser cipher

```
/*  ---------------------------------------------- *\
*         ENCRYPTING METHOD
\*  ---------------------------------------------- */
function Encrypt(plainText) {
        var array    = [],
        newArray = [];
        var uppercasePlainText = plainText.toUpperCase();
        array = uppercasePlainText.split("");
        var arraylength = 0;
        for (var i = 0; i < array.length; i++) {
                if((array[i-1]==' ')&&(array[i]==' )) {
                        newArray.pop();
                        break;
                }
                else {
                        arraylength++;
                        newArray.push(array[i]);
                }
        }
        for (var j = 0; j < arraylength; j++) {
                var oldCharacter = String.fromCharCode(array[j].charCodeAt());
                if (oldCharacter.charCodeAt() > 64) {
                        var newCharacter = String.fromCharCode(oldCharacter.charCodeAt() +
13);
                        if(newCharacter.charCodeAt() > 90)
                        {
```

```
                        newCharacter = String.fromCharCode(newCharacter.charCodeAt()
- 26);
                }
                newArray[j] = newCharacter;
            }
            else {
                newArray[j] = oldCharacter;
            }
        }
        cryptText = newArray.toString();
        cryptText = cryptText.replace(/,/g, "");
        return cryptText;
}
```

```
/*  ---------------------------------------------- *\
*         DECRYPTING METHOD
\*  ---------------------------------------------- */
function Decrypt(cryptText) {
        var array    = [],
        newArray = [];
        var uppercaseCryptText = cryptText.toUpperCase();
        array = uppercaseCryptText.split("");
        var arraylength = 0;
        for (var i = 0; i < array.length; i++) {
                if( (array[i-1]==' ') && (array[i]==' ') ) {
                        console.log("Enter end!");
                        arraylength--;
                        newArray.pop();
                        break;
                }
                else{
                        arraylength++;
                        newArray.push(array[i]);
                }
        }
        for (var j = 0; j < (arraylength); j++) {
                var oldCharacter = String.fromCharCode(array[j].charCodeAt());
                if (oldCharacter.charCodeAt() > 64) {
                        var newCharacter = String.fromCharCode(oldCharacter.charCodeAt() -
13);
                        if(newCharacter.charCodeAt() < 65)
                        {
                                newCharacter = String.fromCharCode(newCharacter.charCodeAt()
+ 26);
```

```
                }
                newArray[j] = newCharacter;
            }
            else {
                newArray[j] = oldCharacter;
            }
        }
        var plainText = newArray.toString();
        plainText = plainText.replace(/,/g, "");
        return plainText;
}
```

Prilog 2 Steganography.js

```
/*  ---------------------------------------------- *\
*         STEGANOGRAPHY
\*  ---------------------------------------------- */
;(function (name, context, factory) {
  if (typeof module !== "undefined" && module.exports) {
    module.exports = factory();
  }
  else if (typeof define === "function" && define.amd) {
    define(factory);
  }
  else {
    context[name] = factory();
  }
})("steg", this, function () {
  var Cover = function Cover() {};
  var util = {
    "isPrime" : function(n) {
      if (isNaN(n) || !isFinite(n) || n%1 || n<2) return false;
      if (n%2===0) return (n===2);
      if (n%3===0) return (n===3);
      var m=Math.sqrt(n);
      for (var i=5;i<=m;i+=6) {
        if (n%i===0) return false;
        if (n%(i+2)===0) return false;
      }
      return true;
    },
    "findNextPrime" : function(n) {
```

```
  for(var i=n; true; i+=1)
    if(util.isPrime(i)) return i;
},
"sum" : function(func, end, options) {
  var sum = 0;
  options = options || {};
  for(var i = options.start || 0; i < end; i+=(options.inc||1))
    sum += func(i) || 0;
  return (sum === 0 && options.defValue ? options.defValue : sum);
},
"product" : function(func, end, options) {
  var prod = 1;
  options = options || {};
  for(var i = options.start || 0; i < end; i+=(options.inc||1))
    prod *= func(i) || 1;
  return (prod === 1 && options.defValue ? options.defValue : prod);
},
"createArrayFromArgs" : function(args,index,threshold) {
  var ret = new Array(threshold-1);
  for(var i = 0; i < threshold; i+=1)
    ret[i] = args(i >= index ? i+1:i);
  return ret;
},
"loadImg": function(url) {
  var image = new Image();
  image.src = url;
  while(image.hasOwnProperty('complete') && !image.complete) {}
  return image;
}
```

```
};
Cover.prototype.config = {
  "t": 3,
  "threshold": 1,
  "codeUnitSize": 16,
  "args": function(i) { return i+1; },
  "messageDelimiter": function(modMessage,threshold) {
          var delimiter = new Array(threshold*3);
          for(var i = 0; i < delimiter.length; i+=1)
            delimiter[i] = 255;
          return delimiter;
        },
  "messageCompleted": function(data, i, threshold) {
        var done = true;
         for(var j = 0; j < 16 && done; j+=1) {
           done = (done) && (data[i+j*4] == 255);
          }
          return done;
        }
};
Cover.prototype.getHidingCapacity = function(image, options) {
  options = options || {};
  var config = this.config;
  var width = options.width || image.width,
     height = options.height || image.height,
     t = options.t || config.t,
     codeUnitSize = options.codeUnitSize || config.codeUnitSize;
     return t*width*height/codeUnitSize >> 0;
};
```

```
/*  ---------------------------------------------- *\
*   ENCODE
\*  ---------------------------------------------- */
Cover.prototype.encode = function(message, image, options) {
  if(image.length) {
    image = util.loadImg(image);
  }
  options = options || {};
  var config = this.config;
  var t = options.t || config.t,
    threshold = options.threshold || config.threshold,
    codeUnitSize = options.codeUnitSize || config.codeUnitSize,
    prime = util.findNextPrime(Math.pow(2,t)),
    args = options.args || config.args,
    messageDelimiter = options.messageDelimiter || config.messageDelimiter;
  if(!t || t < 1 || t > 7) throw "Error: Parameter t = " + t + " is not valid: 0 < t < 8";
  var shadowCanvas = document.createElement('canvas'),
      shadowCtx = shadowCanvas.getContext('2d');
  shadowCanvas.style.display = 'none';
  shadowCanvas.width = options.width || image.width;
  shadowCanvas.height = options.height || image.height;
  if(options.height && options.width) {
    shadowCtx.drawImage(image, 0, 0, options.width, options.height );
  }
  else {
    shadowCtx.drawImage(image, 0, 0);
  }
  var imageData = shadowCtx.getImageData(0, 0, shadowCanvas.width, shadowCanvas.height),
      data = imageData.data;
```

```
var bundlesPerChar = codeUnitSize/t >> 0,
  overlapping = codeUnitSize%t,
  modMessage = [],
  decM, oldDec, oldMask, left, right,
  dec, curOverlapping, mask;
var i, j;
for(i=0; i <= message.length; i+=1) {
  dec = message.charCodeAt(i) || 0;
  curOverlapping = (overlapping*i)%t;
  if(curOverlapping > 0 && oldDec) {
    mask = Math.pow(2,t-curOverlapping) - 1;
    oldMask = Math.pow(2, codeUnitSize) * (1 - Math.pow(2, -curOverlapping));
    left = (dec & mask) << curOverlapping;
    right = (oldDec & oldMask) >> (codeUnitSize - curOverlapping);
    modMessage.push(left+right);
    if(i<message.length) {
      mask = Math.pow(2,2*t-curOverlapping) * (1 - Math.pow(2, -t));
      for(j=1; j<bundlesPerChar; j+=1) {
        decM = dec & mask;
        modMessage.push(decM >> (((j-1)*t)+(t-curOverlapping)));
        mask <<= t;
      }
      if((overlapping*(i+1))%t === 0) {
        mask = Math.pow(2, codeUnitSize) * (1 - Math.pow(2,-t));
        decM = dec & mask;
        modMessage.push(decM >> (codeUnitSize-t));
      }
      else if(((((overlapping*(i+1))%t) + (t-curOverlapping)) <= t)) {
        decM = dec & mask;
```

```
          modMessage.push(decM >> (((bundlesPerChar-1)*t)+(t-curOverlapping)));
        }
      }
    }
    else if(i<message.length) {
      mask = Math.pow(2,t) - 1;
      for(j=0; j<bundlesPerChar; j+=1) {
        decM = dec & mask;
        modMessage.push(decM >> (j*t));
        mask <<= t;
      }
    }
    oldDec = dec;
  }
  var offset, index, subOffset, delimiter = messageDelimiter(modMessage,threshold),
    q, qS;
  for(offset = 0; (offset+threshold)*4 <= data.length && (offset+threshold) <=
modMessage.length; offset += threshold) {
    qS=[];
    for(i=0; i<threshold && i+offset < modMessage.length; i+=1) {
      q = 0;
      for(j=offset; j<threshold+offset && j<modMessage.length; j+=1)
        q+=modMessage[j]*Math.pow(args(i),j-offset);
      qS[i] = (255-prime+1)+(q%prime);
    }
    for(i=offset*4; i<(offset+qS.length)*4 && i<data.length; i+=4)
      data[i+3] = qS[(i/4)%threshold];

    subOffset = qS.length;
  }
```

```
    // Write message-delimiter
    for(index = (offset+subOffset); index-(offset+subOffset)<delimiter.length &&
(offset+delimiter.length)*4<data.length; index+=1)
      data[(index*4)+3]=delimiter[index-(offset+subOffset)];
    // Clear remaining data
    for(i=((index+1)*4)+3; i<data.length; i+=4) data[i] = 255;
    imageData.data = data;
    shadowCtx.putImageData(imageData, 0, 0);
    return shadowCanvas.toDataURL();
  };
```

```
/*  ---------------------------------------------- *\
*   DECODE
\*  ---------------------------------------------- */
  Cover.prototype.decode = function(image, options) {
    if(image.length) {
      image = util.loadImg(image);
    }
    options = options || {};
    var config = this.config;
    var t = options.t || config.t,
      threshold = options.threshold || config.threshold,
      codeUnitSize = options.codeUnitSize || config.codeUnitSize,
      prime = util.findNextPrime(Math.pow(2, t)),
      args = options.args || config.args,
      messageCompleted = options.messageCompleted || config.messageCompleted;
    if(!t || t < 1 || t > 7) throw "Error: Parameter t = " + t + " is not valid: 0 < t < 8";
    var shadowCanvas = document.createElement('canvas'),
        shadowCtx = shadowCanvas.getContext('2d');
    shadowCanvas.style.display = 'none';
    shadowCanvas.width = options.width || image.width;
    shadowCanvas.height = options.width || image.height;
    if(options.height && options.width) {
      shadowCtx.drawImage(image, 0, 0, options.width, options.height );
    }
    else {
      shadowCtx.drawImage(image, 0, 0);
    }
    var imageData = shadowCtx.getImageData(0, 0, shadowCanvas.width, shadowCanvas.height),
        data = imageData.data,
```

```
    modMessage = [],
    q;
  var i, k, done;
  if (threshold == 1) {
    // Check for message end
    for(i=3, done=false; !done && i<data.length && !done; i+=4) { // true = !done(false)
      done = messageCompleted(data, i, threshold);
      if(!done) {
        modMessage.push(data[i]-(255-prime+1));
      }
    }
  }
  var message = "",
      charCode = 0,
      bitCount = 0,
      mask = Math.pow(2, codeUnitSize)-1;
  for(i = 0; i < modMessage.length; i+=1) {
    charCode += modMessage[i] << bitCount;
    bitCount += t;
    if(bitCount >= codeUnitSize) {
      message += String.fromCharCode(charCode & mask);
      bitCount %= codeUnitSize;
      charCode = modMessage[i] >> (t-bitCount);
    }
  }
  if(charCode !== 0) message += String.fromCharCode(charCode & mask);
  return message;
};
return new Cover();
```

```
});
```