



UNIVERZITET SINGIDUNUM
DEPARTMAN ZA POSLEDIPLOMSKE STUDIJE I MEĐUNARODNU SARADNJU

- MASTER STUDIJSKI PROGRAM -
SAVREMENE INFORMACIONE TEHNOLOGIJE

Bezbednosni mehanizam za prenos i skladištenje podataka u oblaku

- Master rad -

Mentor:

Prof. dr. Mladen Veinović

Student:

Marija Vujošević

Br. indeksa: 2013/410159



Beograd, 2015.



Bezbednosni mehanizam za prenos i skladištenje podataka u oblaku

Sažetak: U ovom radu proučava se oblast zaštite podataka na udaljenim serverima u oblaku i tokom prenosa komunikacionim kanalima. Vršiti se pregled najkvalitetnijih komercijalnih rešenja iz ove oblasti. Pažnja će biti posvećena onim rešenjima koja imaju implementiranu *end-to-end* zaštitu. Nakon toga biće predstavljen pregled teorijskih osnova istraživanja. Biće objašnjeni simetrični šifarski algoritmi (DES i AES), režimi rada blokovskih šifara, zatim asimetrični šifarski algoritmi, tačnije RSA, pravi i pseudoslučajni generatori brojeva. Kako će se rešenje koje će biti razvijeno baviti bezbednim prenosom podataka između klijenta i servera, potrebno je objasniti pojmove autentifikacije i SSL. Na kraju fokus će biti stavljen na razvoj sopstvenog rešenja u kome će biti implementiran predstavljeni bezbednosni mehanizam uz modul za generisanje kriptoloških ključeva.

Ključne reči: *end-to-end* zaštita, skladištenje podataka u oblaku, zaštita podataka, simetrični i asimetrični šifarski sistemi, generatori slučajnih brojeva, autentifikacija

Security mechanism for cloud end-to-end encryption

Abstract: In this paper we study the Cloud computing data security. A focus is set on some of the most quality commercial solutions that have implemented end-to-end encryption. Then a theoretical basis of research overview will be presented which include symmetric key cryptography (DES and AES), block ciphers, public key cryptography, namely RSA and true and pseudorandom number generator. Since the solution that will be developed will deal with secure data transfer between client and server, it is necessary to explain the concepts of authentication and SSL. In the end, the focus will be set on own solutions development in which this security mechanism will be implemented including a symmetric key generation module.

Keywords: end-to-end encryption, Cloud computing, data security, symmetric and public-key cryptography, random number generator, authentication

SADRŽAJ

1. Uvod	4
2. Metodologija naučnog istraživanja	5
2.1 Predmet istraživanja	5
2.2 Ciljevi i zadaci istraživanja	5
2.3 Istraživačke hipoteze	5
2.4 Metodi istraživanja i tok istraživačkog procesa	6
3. Pregled u oblasti istraživanja	7
3.1 Tresorit	8
3.2 SpiderOak	10
3.3 TeamDrive.....	12
3.4 MEGA.....	15
3.5 Dropbox i Boxcryptor.....	18
3.6 Glavne karakteristike rešenja	20
4. Teorijske osnove istraživanja	21
4.1 Simetrični šifarski sistemi	21
4.1.1 DES.....	22
4.1.2 AES.....	24
4.2 Režimi rada blokovskih šifara	27
4.2.1 Elektronska kodna knjiga (ECB)	27
4.2.2 Ulančavanje blokova šifrata (CBC).....	28
4.2.3 Inicijalni vektor	29
4.2.4 Ispunjavanje.....	29
4.3 Asimetrični šifarski sistemi	29
4.3.1 RSA.....	30
4.4 Heš funkcija	31
4.4.1 MD5	31

4.4.2	SHA	31
4.4.3	Kriptografska so.....	32
4.5	Generatori slučajnih brojeva	33
4.5.1	Pravi generator slučajnih brojeva (TRNG)	33
4.5.2	Pseudoslučajni generator brojeva (PRNG)	34
4.5.3	Entropija	34
4.6	Autentifikacija.....	35
4.7	SSL.....	36
4.8	NIST testovi.....	37
4.8.1	Ispitivanje učestalosti u nizu.....	37
4.8.2	Ispitivanje učestalosti u bloku	38
4.8.3	Ispitivanje uzastopnih ponavljanja istih bitova u nizu	38
4.8.4	Ispitivanje najdužeg uzastopnog ponavljanja jedinica u bloku	39
4.8.5	Ispitivanje stanja binarne matrice	40
4.8.6	Ispitivanje diskretne Furijerove transformacije.....	41
4.8.7	Ispitivanje nepreklapajućih uzoraka	42
4.8.8	Ispitivanje preklapajućih uzoraka	42
4.8.9	Maurerov univerzalni statistički test	43
4.8.10	Ispitivanje linearne kompleksnosti	44
4.8.11	Serijski test.....	45
4.8.12	Ispitivanje približne entropije	46
4.8.13	Ispitivanje slučajnih zbirova.....	46
4.8.14	Ispitivanje slučajne digresije.....	47
4.8.15	Ispitivanje slučajne promenljive digresije.....	48
4.9	.NET Framework 4.5	49
4.10	Visual C#	50
4.11	SQL	51
4.11.1	Tipovi podataka	51
4.11.2	Relacioni model	52
4.11.3	SQL Server.....	52

4.11.4 Oblak.....	53
4.11.5 Bezbednosna poboljšanja.....	53
4.12 WCF.....	54
5. Pregled predloženog rešenja.....	57
5.1 Generička šema predloženog rešenja	57
5.2 Postavka i objašnjenje eksperimentalnog okruženja	60
5.3 Performanse predloženog okruženja sa prikazom eksperimentalnih rezultata	71
5.4 Oblast primene predloženog rešenja	73
6. Zaključak	74
6.1 Sumarni ciljevi rada i ostvareni rezultati	74
6.2 Predlog za budući rad	74
Literatura.....	75
Spisak slika	76
Spisak tabela.....	78
Spisak grafikona.....	78

1. Uvod

Informacije predstavljaju važan resurs u savremenom poslovanju. Bez obzira u kom se obliku čuvaju, moraju biti adekvatno zaštićene. Iz tog razloga zaštita informacija, očuvanje njene poverljivosti, integriteta i celovitosti je od presudne važnosti.

U poslednjih nekoliko godina, javlja se globalni trend skladištenja podataka na udaljenim serverima (u oblaku). Korisnicima se nudi sve veći broj rešenja za migraciju podataka sa lokanih računara na te servere.

U mnoštvu ponuđenih komercijalnih rešenja potrebno je pronaći ona koja će zadovoljiti najstrože sigurnosne kriterijume. Akcenat se stavlja kako na sigurno skladištenje podataka tako i na njihov siguran prenos komunikacionim kanalima. Istraživanjem dosadašnjeg stanja u ovoj oblasti, izdvojila su se neka rešenja koja će u nastavku rada biti predstavljena. U prvom delu rada biće opisan način na koji funkcionišu, koje šifarske algoritme (simetrične i asimetrične) koriste kao i platforme koje podržavaju. Zatim će sve najvažnije karakteristike biti sumirane u uporednom tabelarnom prikazu. Takođe, pored svih rešenja koja koriste *end-to-end* zaštitu, biće predstavljen i Dropbox, kao jedno od najpopularnijih rešenja za skladištenje u oblaku u svetu, koje ne koristi ovaj nivo zaštite i biće objašnjen način kako se taj problem može prevazići.

U drugom delu ovog rada biće predstavljena znanja iz oblasti kriptologije i objašnjeni pojmovi i način rada simetričnih i asimetričnih šifarskih sistema kao i režimi rada blokovskih šifara. Nakon toga će biti objašnjen pojam heš funkcije i standardi koji se koriste u njenoj primeni, a biće obrađeni pojmovi pravog i pseudoslučajnog generatora slučajnih brojeva. Sa stanovišta zaštite podataka u njihovom prenosu i pristupu neophodno je razumevanje pojmova autentifikacije i SSL-a.

Nastavak rada posvećen je NIST-ovom paketu testova razvijenih za testiranje slučajnosti binarne sekvence koje proizvode hardver ili softver na bazi kriptografski slučajnih ili pseudoslučajnih brojeva.

Završni deo rada predstavlja opis razvijenog sopstvenog rešenja u kome će biti implementiran bezbednosni mehanizam za šifrovanje/dešifrovanje i skladištenje tih fajlova na udaljenom serveru upotrebom *end-to-end* zaštite.

2. Metodologija naučnog istraživanja

2.1 Predmet istraživanja

U današnje vreme, kada većina korisnika brine o bezbednosti svojih podataka, potrebno je naći metode i rešenja koja će omogućiti visok stepen zaštite. Iz tog razloga predmet istraživanja u ovom radu predstavlja zaštitu podataka pre njihovog smeštanja na udaljeni server (u oblaku), ali i zaštitu komunikacionih kanala.

Metoda slanja podataka na server u otvorenom tekstu, a zatim šifrovanje i smeštanja istih, ostavlja mogućnost serveru da pročita podatke pre šifrovanja. Na ovaj način nivo bezbednosti opada. Iz tog razloga uvodi se novi princip kojim se to sprečava. Reč je o *end-to-end* zaštiti koja podrazumeva šifrovanje fajlova na računaru korisnika pre slanja na server. Ovim se postiže viši nivo sigurnosti time što server ne može doći do sadržaja fajlova koji se na njemu čuvaju.

Tokom istraživanja vršiće se pregled nekih od najboljih komercijalnih rešenja i sagledavanje njihovih karakteristika kako bi se uočile njihove prednosti i nedostaci. Svi ti podaci biće uzeti u obzir prilikom razvoja sopstvenog rešenja koje će biti predstavljeno u master radu.

2.2 Ciljevi i zadaci istraživanja

Ovaj rad namenjen je svima koji brinu o bezbednosti svojih podataka i podjednako se odnosi na privatne i poslovne korisnike.

Naučni cilj ovog rada je istraživanje rešenja koja nude skladištenje korisničkih podataka u oblak vršeći njihovu sinhronizaciju sa lokalnih računara. Fokus je stavljen na ona rešenja koja omogućavaju njihovo šifrovanje pre slanja putem komunikacionog kanala.

Glavni cilj ovog rada jeste izrada sopstvenog rešenja koja će korisnicima najpre omogućiti generisanje sopstvenih ključeva kojima će potom šifrovati željene fajlove i nakon toga ih smestiti u bazu podataka na udaljenom serveru.

2.3 Istraživačke hipoteze

Opšta hipoteza

Informacije koje se prenose preko komunikacionih kanala, a zatim smeštaju na servere, izložene su riziku da budu kompromitovane ili ukradene. Sve više vlada potreba za zaštitom tih podataka i pouzdanim rešenjima koja će je pružiti. Iz tog razloga potrebno je, od mnoštva komercijalnih rešenja, odabrati ono koje će zadovoljiti potrebe i visoke zahteve korisnika. Da bi se to učinilo, važno je dobro poznavanje oblasti zaštite

informacija, a zatim je neophodno izvršiti analizu performansi i pregled karakteristika koje ta rešenja pružaju.

Radna hipoteza

Nakon pregleda specifikacija rešenja, odabira se nekoliko najkvalitetnijih. Kreira se eksperimentalno okruženje u kome se ona testiraju i vrši njihova analiza. Svrha analize je u tome da se sagledaju performanse rešenja kako bi se uvidele njihove prednosti i nedostaci i na taj način utvrdilo koje kriptografske tehnike i sistem zaštite je potrebno implementirati u razvoju sopstvenog rešenja.

2.4 Metodi istraživanja i tok istraživačkog procesa

Metodologija istraživanja svodi se na upotrebu materijala prikupljenog sa izvora na Internetu kao i stručne literature iz oblasti zaštite informacija. U eksperimentalnom okruženju, metodom analize prikupljaju se informacije o najvažnijim karakteristikama odabranih rešenja za skladištenje podataka.

Tabelarnim prikazom i metodom komparacije analiziranih podataka dolazi se do odabira kriptografskih tehnika koje će biti korišćene prilikom razvoja sopstvenog rešenja.

3. Pregled u oblasti istraživanja

U današnje vreme postoji širok spektar uređaja na kojima je moguće skladištiti podatke. Većina njih je mobilnog tipa, a podaci koji se na njima čuvaju imaju svoju vrednost i mogu biti više ili manje osetljivi. U slučaju gubitka uređaja, svi podaci dospevaju u ruke nekoga ko bi ih potencijalno mogao zloupotребiti. Iz tog razloga je potrebno čuvati ih na sigurnom mestu.

Poslednjih godina javlja se trend skladištenja podataka na udaljenim serverima koji omogućavaju njihovu sinhronizaciju na više uređaja. Ova *on line* rešenja pružaju korisnicima lokaciju na kojoj mogu da kreiraju rezervne kopije svojih podataka.



Slika 3.1 – Prikaz stanja fajla u fazama prenosa (na računaru korisnika, tokom prenosa i na serveru), prenos otvorenog teksta ili šifrovanog fajla

Šifrovanje predstavlja verovatno najbolji način da se obezbedi sigurnost podataka. Rešenja koja pružaju *end-to-end* šifrovanje podataka koriste najjače šifarske algoritme kako za šifrovanje podataka pre smeštanja na server, tako i prilikom komunikacije sa serverom. Na slici 3.2 prikazana je arhitektura ovakvog rešenja. U nastavku će biti predstavljena neka od trenutno najboljih komercijalnih rešenja.



Slika 3.2 – Arhitektura šifarskih *on line* rešenja

3.1 Tresorit

Tresorit je *on line* servis za skladištenje podataka u oblaku sa sedištem u Mađarskoj i Švajcarskoj što obezbeđuje poboljšanu bezbednost i šifrovanje podataka iz razloga što ne podleže sudskim odredbama NSA (engl. *National Security Agency*).

Kao dodatna bezbednosna mera, Tresorit servis je dostupan samo preko klijentske desktop aplikacije, veb fajl upload-er ne postoji. Softver je dostupan za Microsoft Windows i Mac OS X operative sisteme kao i za Android, iOS, Windows Phone mobilne platforme.

Tresorit koristi *end-to-end* šifrovanje što znači da se šifrovanje i dešifrovanje vrši na strani klijenta. Pristup fajlovima ima jedino korisnik koji ih je kreirao i korisnici ovlašćeni od strane vlasnika. Ovaj način šifrovanja omogućava da se svi fajlovi koje skladište u oblaku budu sigurni isto koliko bi bili sigurni da se nalaze na lokalnom računaru korisnika.

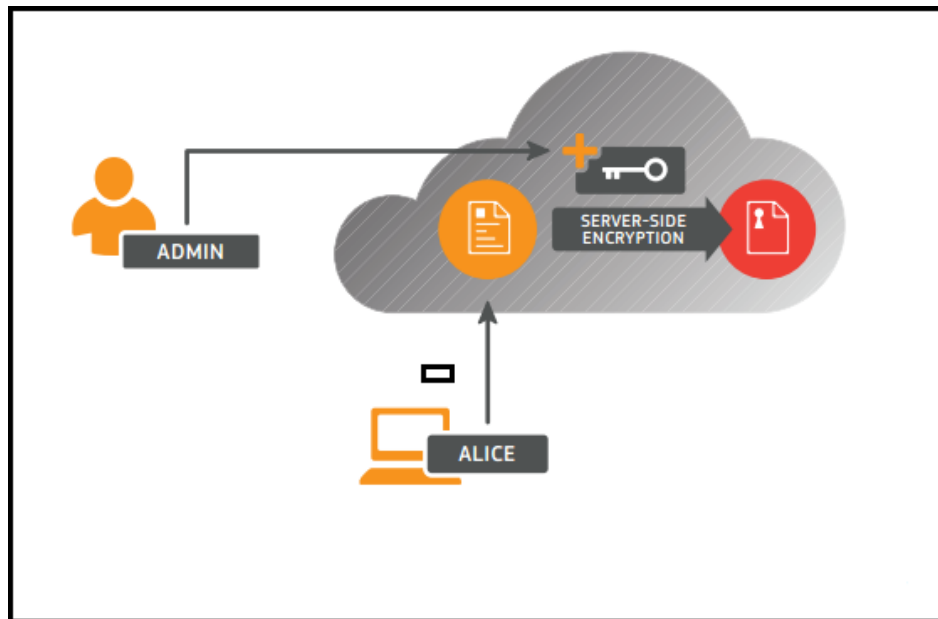
Usklađen je sa najstrožim bezbednosnim propisima i koristi isključivo industrijski standardizovane algoritme. Osnovna razlika između Tresorit i njegovih konkurenata je ta da Tresorit primenjuje AES-256 šifrovanje fajlova na strani klijenta dok se oni još uvek nalaze na lokalnom računaru i tek nakon toga ih šalje u oblak. Na ovaj način se onemogućava pristup podacima od strane servera. Transakcije (npr. postavljanje fajlova i modifikacije) se autentifikuju pomoću RSA-4096. Fajlovi su takođe zaštićeni HMAC (engl. *Hash-based message authentication code*) autentifikacijom koja se primenjuje na SHA-512 heševe.

Šifrovani fajlovi se smeštaju u oblak preko TLS zaštićenih komunikacionih kanala. Na ovaj način se garantuje poverljivost podataka.

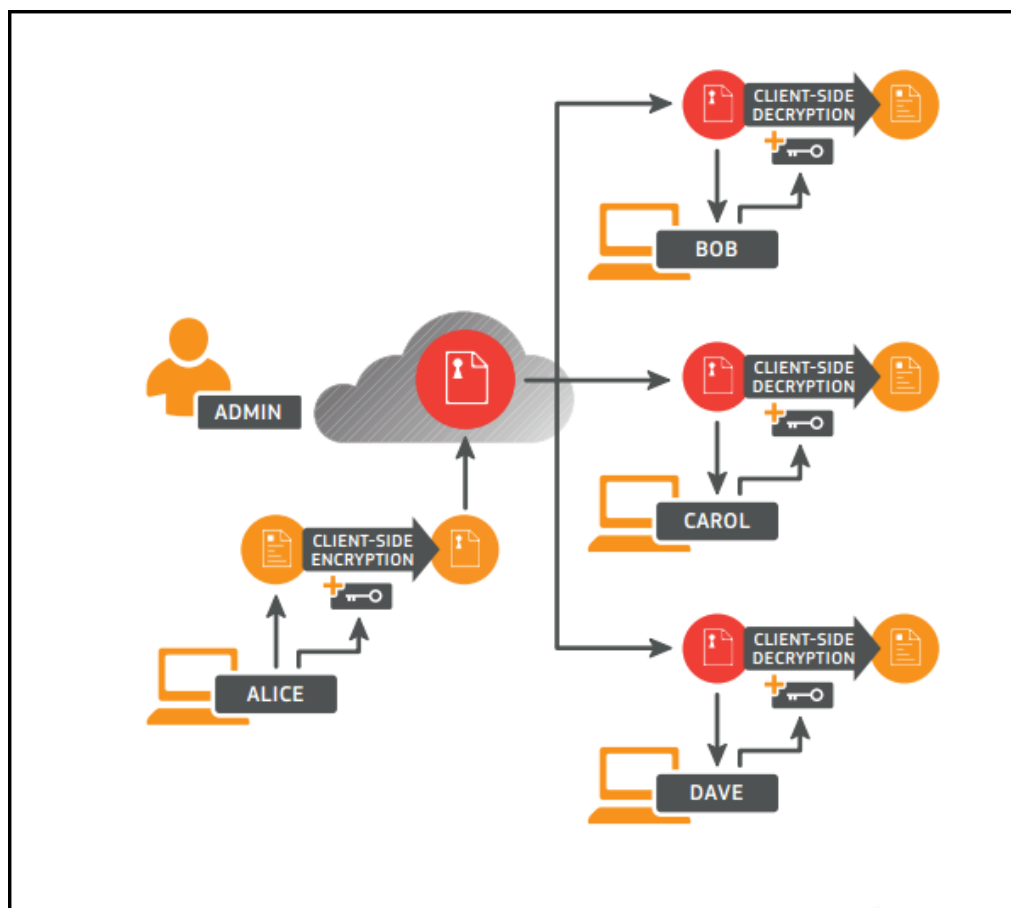
Sistem za upravljanje ključevima je kreiran tako da dokumentima može pristupiti samo onaj kome je taj dokument namenjen. Kada korisnik modifikuje dokument on se automatski ponovo šifruje novim ključem. Pored toga, stari ključ se menja i folder se tada ponovo šifruje novim ključem. Kada se jednom fajl ili folder ponovo šifruje novim ključem, sledeći korisnik koji ga koristi, koristi taj novo-kreirani ključ. Važna karakteristika je ta da se ključ kojim se šifruju podaci nalazi jedino kod korisnika aplikacije.

Tresorit omogućava da se bilo koji folder na računaru pretvori u deljeni folder.

U nastavku je predstavljen uporedni prikaz klasičnog načina skladištenja podataka u oblaku u kome su dokumenta vidljiva u otvorenom tekstu (slika 3.3) i *end-to-end* šifrovanja i dešifrovanja fajlova koji primenjuje Tresorit (slika 3.4) [21] [22].



Slika 3.3 – Klasičan način skladištenja podataka u oblaku



Slika 3.4 – Tresorit primenjuje end-to-end šifrovanje

3.2 SpiderOak

SpiderOak predstavlja *on line* alat za skladištenje podataka, njihovo deljenje, sinhronizaciju i pristup, sa sedištem u SAD. Razvijeno rešenje omogućava rad na Windows, Mac i Linux operativnim sistemima, a takođe postoje i verzije za Android i iOS mobilne platforme.

Korisnikovi SpiderOak podaci su čitljivi samo njemu. Većina *on line* sistema za skladištenje šifruje podatke samo tokom njihovog prenosa, ali oni ne ostaju šifrovani na serveru. To znači da svako može pristupiti podacima koji se nalaze na serveru ako ima pristup njemu kao što je npr. osoblje kompanije. Ukoliko se i čuvaju šifrovani podaci, obično se lozinka (ili skup ključeva za šifrovanje) često čuva zajedno sa podacima, čime su dostupni svakome ko ima pristup serverima. SpiderOak nikada ne čuva lozinku, što znači da su podaci uvek zaštićeni.

SpiderOak šifrovanje je sveobuhvatno - čak i sa fizičkim pristupom serverima SpiderOak osoblje ne može da zna čak i imena fajlova i foldera. Sve što oni vide jesu šifrovani podaci.

Korisnik svoje podatke štiti lozinkom koja se nikada ne prenosi na server u svom izvornom obliku (otvorenom tekstu). U slučaju da korisnik zaboravi svoju lozinku ne postoji način da se ti podaci povrate tj. da se dešifruju.

Prilikom inicijalnog pokretanja SpiderOak aplikacije, na računaru se generiše serija jakih ključeva. Oni se potom šifruju korisničkom lozinkom i tako šifrovani skladište se na server.

Generisanje ključa se vrši pomoću JavaScript-a. Nakon što se korisnik registruje na sajtu, JavaScript kôd pravi heš vrednost lozinke pomoću *bcrypt* algoritma pre slanja na server. Nakon inicijalnog pokretanja aplikacije od korisnika se traži da unese korisničko ime i lozinku. Lozinka se nikada ne šalje u otvorenom tekstu.

Ukoliko korisnik ne želi da koristi JavaScript, u tom slučaju on ne unosi lozinku već mu se generiše privremeni aktivacioni kôd koji se koristi za autentifikaciju pri prvom pokretanju aplikacije. Nakon toga će od njega biti zatraženo da unese lozinku. U oba slučaja, po inicijalnom pokretanju aplikacije generišu se jaki šifarski ključevi koji se šifruju korisničkom lozinkom i potom čuvaju. Originalna lozinka se ne čuva nigde. Ovo predstavlja neophodne korake ka kreiranju pravog *zero-knowledge* okruženja. Upotrebom *zero-knowledge* sistema privatnosti korisnici ne moraju da brinu o tome da li je kompanija koja čuva njihove podatke od poverenja.

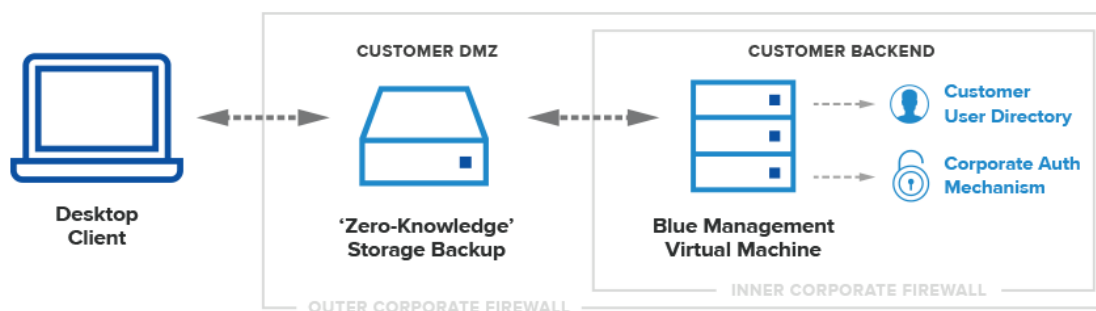
SpiderOak koristi AES-256 u CFB modu i HMAC-SHA256. Koristi se grupa ključeva – za određene akcije se koristi različit ključ (novi ključ za svaki folder, verziju fajla i pojedinačnih blokova podataka od kojih su sastavljene verzije fajlova). Posedovanje ključeva sa ovako ograničenim dometima omogućava dodatnu sigurnost podataka.

Ključevi se šifruju sa AES-256, koristeći PBKDF2 (sa SHA-256) algoritam za derivaciju ključa uz minimum 16384 rundi i 32 bajtova slučajnih podataka (engl. *salt*). Ovaj pristup sprečava *brute force* napade ili napade na bazu podataka.

SpiderOak nalozi takođe uključuju i 3072 bitni RSA par ključeva koji se trenutno ne koristi, ali se generiše zbog daljih planova u razvoju aplikacije u smeru kreiranja deljenih fajlova između korisnika.

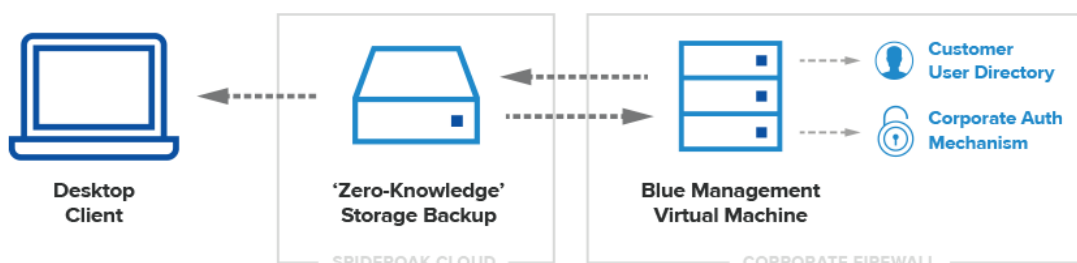
Nažalost, većina *on line* sistema za skladištenje podataka šifruje podatke samo prilikom njihovog prenosa što znači da se oni na serveru nalaze u otvorenom formatu i time su izloženi riziku. SpiderOak koristi *end-to-end* šifrovanje čime su podaci potpuno zaštićeni i jedino im može pristupiti njihov vlasnik korišćenjem svoje lozinke.

Sa svojim „*Private Cloud*“ rešenjem, SpiderOak postavlja svoje kompletno *end-to-end* rešenje iza korisničkog *firewall*-a – sve od autentifikacije, skladištenja do svih komunikacija koje se odvijaju (slika 3.5Slika 3.5).

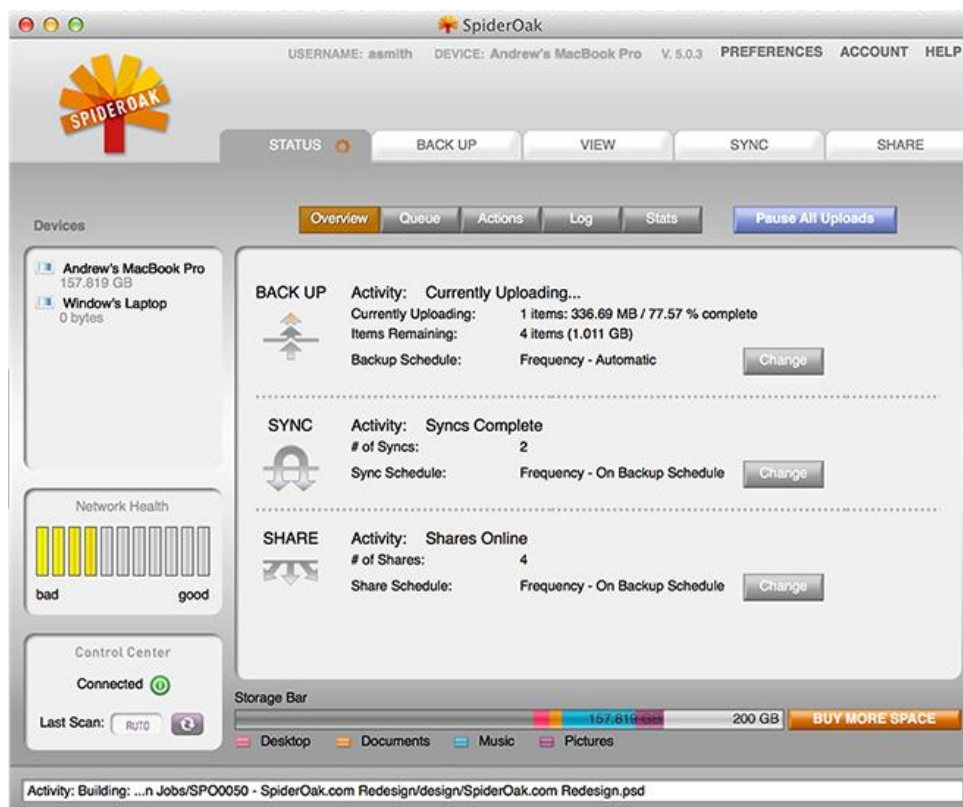


Slika 3.5 – On-premise privatni oblak

Sa svojim „*Hosted Storage*“ rešenjem, SpiderOak skladišti korisničke podatke u SpiderOak Enterprise-class data centrima. Autentifikacija se odvija na strani korisnika, a skalabilnost na strani servera (slika 3.6) [23].



Slika 3.6 – SpiderOak *host*-ovano skladište



Slika 3.7 – SpiderOak aplikacija

3.3 TeamDrive

TeamDrive predstavlja nemački softver i provajdera servisa u oblaku koji omogućava sinhronizaciju, skladištenje, deljenje i bezbedna razmenu podataka kako za privatne tako i za poslovne klijente. Obezbeđuje zaštitu privatnosti deljenja podataka i u isto vreme pomaže da se spreči njihov gubitak putem *on line* izrade rezervnih kopija. TeamDrive 3 omogućava bezbednu sinhronizaciju podataka između desktop računara i mobilnih uređaja (pametnih telefona i tablet računara).

Softver je dostupan za Windows, Mac OS X i Linux operativne sisteme kao i za Android i iOS mobilne platforme za četiri tipa naloga: besplatan, lični, profesionalni i *SecureOffice*.

U TeamDrive, fajlovi se skladište u ono što se naziva Prostor (engl. *Space*). Svaki Prostor ima svoj 256-bitni AES ključ. Ključ, koji se nalazi lokalno na klijentskom računaru i poznat je samo njemu, koristi se da omogući pristup i šifrovanje fajlova u Prostoru pre nego što oni napuste računar krajnjeg korisnika. Svaki Prostor može imati veći broj krajnjih korisnika ili članova koji imaju pristup fajlovima koji se u njemu nalaze. Fajlovi se razmenjuju preko SpaceDepot na TeamDrive hosting-u ili WebDAV serveru.

Tokom instalacije softvera korisnik mora da završi proces registracije. Aplikacija omogućava korisniku da kreira jedan ili više Prostora ili da se pridruži Prostoru u koji je pozvan. Prostor može biti bilo koji direktorijum na fajl sistemu računara krajnjeg

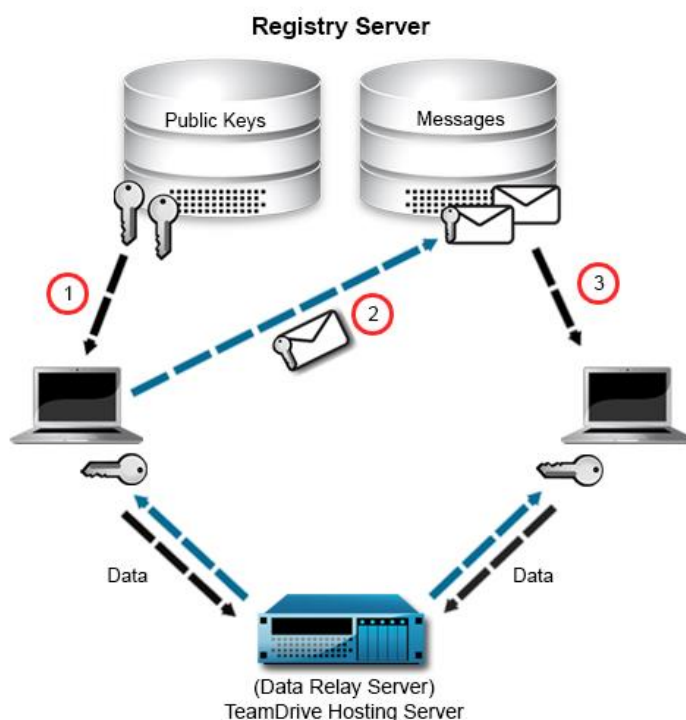
korisnika. Fajlovi u ovom direktorijumu se mogu normalno koristiti od strane ostalih aplikacija, međutim, sve promene napravljene na fajlovima automatski se prenose na sve članove određenog prostora nakon što se fajl zatvori.

TeamDrive server za registraciju skladišti informacije o svim TeamDrive korisnicima. Ovo uključuje podatke koji su uneti tokom instalacije i javni ključ korisnika koji je kreiran tokom instalacije. Svaki korisnik se identifikuje preko e-mail adrese. Ona se verifikuje u toku instalacije upotrebom aktivacionog kôda. Server za registraciju obezbeđuje sigurnu razmenu poruka između TeamDrive klijenata. Mehanizam za pozivanje ostalih korisnika da se pridruže u Prostor takođe koristi ovaj servis poruka.

Server koji nudi SpaceDepot je odgovoran za čuvanje i prenos izmena koje se dešavaju u Prostorima u SpaceDepot. Skladištenje i prenosni mehanizmi omogućavaju klijentima da razmene fajlove čak i ako oba klijenta nisu na mreži u isto vreme. Sve datoteke koje su uskladištene na *Enterprise* hosting serveru šifrovane su pomoću 256-bitnog AES ključa.

Komunikacija između TeamDrive komponenti se zasniva na HTTP protokolu. Standardne metode HTTP POST, PUT i GET se koriste za sve transakcije. Sva komunikacija se pokreće od strane TeamDrive klijenta. Imajući ovo u vidu, TeamDrive klijent se ne razlikuje od običnog Veb pretraživača. Shodno tome, TeamDrive klijent će funkcionisati u svakom mrežnom okruženju koje ima pristup Internetu (uključujući i HTTP *proxy*). Koristeći ovaj pristup, TeamDrive kreira virtuelnu privatnu mrežu bez potrebe za konfiguracijom sistema u mrežnom okruženju (bez podešavanja *firewall*-a osim kada se koristite određeni portovi).

U nastavku je prikazana TeamDrive arhitektura (slika 3.8).



Slika 3.8 – TeamDrive arhitektura

Bezbednost i poverljivost korisničkih podataka predstavlja najviši prioritet. Pre prenosa podatka, oni se šifruju na računaru korisnika sa AES-256 i samo korisnik odlučuje ko ima pristup njegovim podacima.

Šifrovanje u TeamDrive funkcioniše na sledeći način. Nakon instalacije, kreira se par ključeva (Diffie-Hellman u verziji TeamDrive 2 i RSA-3072 u verziji TeamDrive 3). Ovi ključevi se koriste za siguran prenos pozivnica između korisnika za deljenje zajedničkih fajlova. Javni ključ se čuva na centralnom serveru kojim upravlja TeamDrive.

TeamDrive šifruje fajlove na računaru korisnika pre nego što ih pošalje na server, korišćenjem AES-256 algoritama. Za svaki folder kreira se poseban AES-256 ključ. Kada korisnik pozove novog člana grupe, simetrični ključ foldera koji se deli šifruje se javnim ključem člana grupe kome je poziv upućen i potom mu se šalje zahtev. Podaci za pristup disku na serveru TeamDrive se takođe šalju u šifrovanom obliku. Član grupe kome je zahtev upućen dešifruje simetrični ključ svojim privatnim ključem i nakon toga dobija pristup deljenom folderu. Ovaj postupak osigurava da samo ovlašćeni članovi grupe imaju pristup podacima.

Uspostavljanje i upravljanje ključevima i pristup podacima se vrši automatski na TeamDrive-u, bez potrebe korisnika da brine o tome. Ovo obezbeđuje da nijedan podatak u otvorenom tekstu ne napusti korisnikov računar i pojedincima ne dozvoljava ne ovlašćeni pristup.

Svi 256-bitni AES ključevi čuvaju se samo na računaru klijenta. Amazon, NSA i treća lica nemaju nikakvu mogućnost dešifrovanja podataka.

TeamDrive nudi dve mogućnosti za sinhronizaciju i pouzdano čuvanje podataka: korisnik može izabrati između TeamDrive oblaka (korisnik koristi TeamDrive servere) ili može koristiti svoj TeamDrive server.

Sinhronizacija korisničkih Prostora (sinhronizacija foldera) se vrši pomoću TeamDrive klijentske aplikacije korišćenjem TeamDrive servera. Ovo osigurava da svi fajlovi uvek budu pouzdano i bezbedno sinhronizovani.

TeamDrive softver se može koristiti u kombinaciji sa sva tri servera u isto vreme. Mogu se koristiti TeamDrive Amazon Cloud, privatni TeamDrive Personal Server ili WebDAV [24].

3.4 MEGA

MEGA predstavlja servis za skladištenje podataka u oblaku i za fajl *hosting* sa sedištem na Novom Zelandu. Obezbeđuje robustan oblak za skladištenje koji pruža visok nivo privatnosti.

Softver je prvenstveno kreiran za upotrebu iz Veb pretraživača, ali ubrzo je razvijena i aplikacija za Windows operativni sistem (u izradi su aplikacije za Mac i Linux) kao i za Android i iOS mobilne platforme.

MEGA koristi *end-to-end* zaštitu. Podaci se šifruju na klijentskom računaru pre slanja na server i dešifruju nakon smeštanja na klijentski računar. Podaci dobijeni od klijenata se čuvaju i prenose doslovno: server ne vrši dešifrovanje, niti ponovo šifruje niti verifikuje šifrovanje korisničkih fajlova. Sve kriptografske operacije su pod kontrolom krajnjeg korisnika.

Klijentski računar je odgovoran i za generisanje, razmenu i upravljanje ključevima za šifrovanje. Svi ključevi ostaju na klijentskom računaru sa izuzetkom RSA javnih ključeva. Sva komunikacija koja se odvija između klijenta i servera je šifrovana. Korisnik je jedini koji poseduje ključ za šifrovanje i odlučuje ko će imati pristup njegovim podacima. Takođe se od korisnika ne zahteva nikakva instalacija programa s obzirom da se sve dešava u njegovom Veb pretraživaču.

MEGA koristi AES-128 simetrični šifarski algoritam za šifrovanje fajlova dok se za uspostavljanje zajedničke tajne između korisnika koristi RSA-2048 (dužina ključa je izabrana kao srednja vrednost između "previše nesigurno" i "suviše sporo"). Šifrovanje, dešifrovanje i generisanje ključeva je implementirano u JavaScript-u (slika 3.9).

Entropy Collection

This game will collect **entropy** based on your mouse movements. This will strengthen your RSA key.
You can abort the game once the progress indicator is at 100%.

0%



Do you wish to play? If you skip this step, the security of your account may suffer. Please move your mouse extensively and randomly to reduce the risk.

Start Game

Skip

Progress >

Slika 3.9 – MEGA – Početni korak u povećavanju entropije RSA ključa

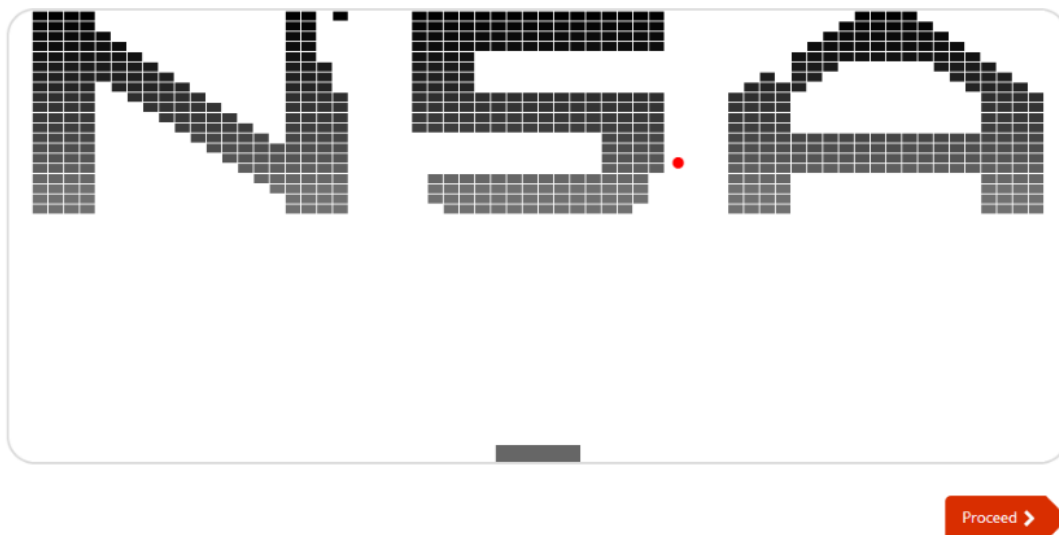
Generator slučajnih brojeva koji je ugrađen u JavaScript dodatno je unapređen modulom za ojačavanje RSA ključa i entropije preko podataka prikupljenih preko miša i tastature (slike 3.10 i 3.11).

Entropy Collection

This game will collect **entropy** based on your mouse movements. This will strengthen your **RSA key**. You can abort the game once the progress indicator is at 100%.

Done! you may now proceed to the next step to complete your key.

100%



Slika 3.10 – MEGA – Procedura prikupljanja entropije na osnovu pokreta miša

Public - Private key creation

Your 2048-bit RSA public / private key pair is now being created. This is a very important step and can take a couple of seconds. Please be patient.



The earliest known use of cryptography relates back to almost 4000 years ago, in ancient Egypt

Complete your account

Slika 3.11 – MEGA – Generisanje RSA para ključeva nakon prikupljenih podataka

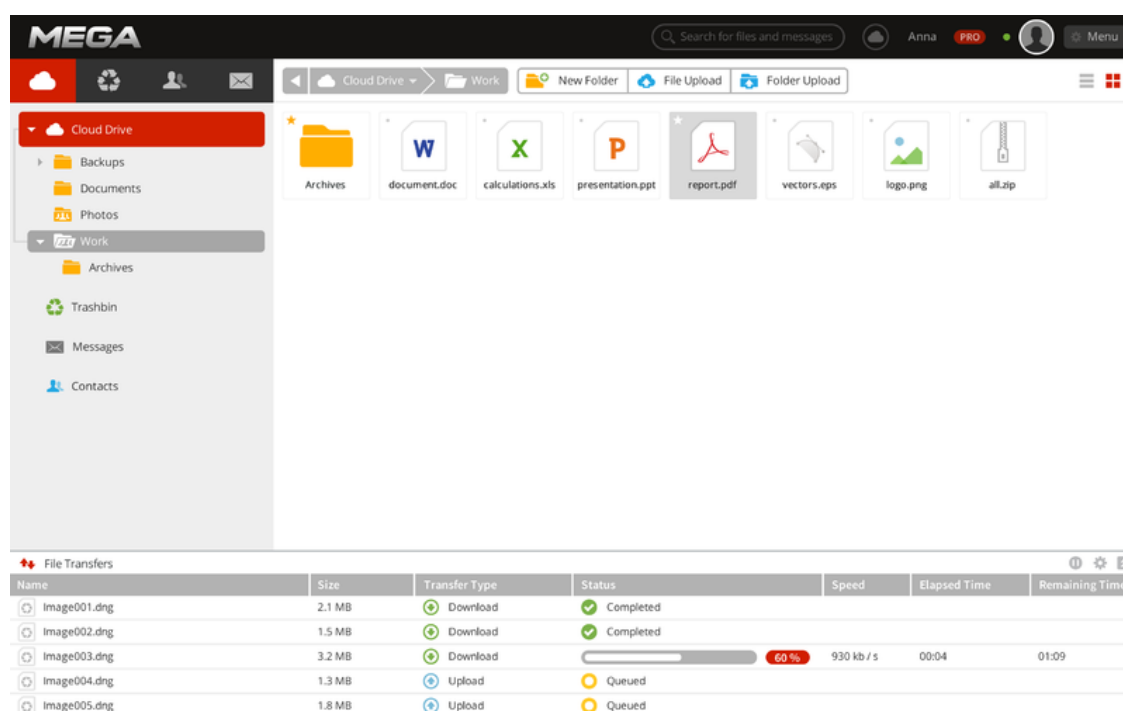
Sve simetrične kriptografske operacije zasnovane su na AES-128 šifarskom algoritmu. On radi u CBC modu za blokove atributa fajlova i foldera i u CTR modu za same podatke. Svaki čvor fajla i foldera koristi sopstveni nasumično generisan 128-bitni ključ. Fajl čvorovi koriste isti ključ za blok atributa i podataka, plus 64-bitnu slučajnu početnu vrednost i 64-bitni MAC za proveru integritet podataka.

Svaki korisnički nalog koristi glavni simetrični ključ za šifrovanje u ECB modu svih ključeva u čvoru koji se nalaze u njegovom stablu. Ovaj glavni ključ se štiti heš vrednošću korisnikove lozinke i tako smešta i čuva na MEGA serverima.

Pored simetričnog ključa, svaki korisnički nalog ima 2048-bitni RSA par ključeva za bezbednu razmenu podataka kao što su deljeni ključevi ili ključevi fajlova i foldera.

Kod deljenih foldera između korisnika, vlasnik foldera je taj koji određuje prava pristupa korisnicima tog foldera. Svi korisnici dobijaju pristup preko zajedničkog deljenog ključa koji se pomoću RSA od vlasnika prosleđuje novim korisnicima.

U nastavku je prikazan izgled MEGA aplikacije (slika 3.12) [25].



Slika 3.12 – MEGA aplikacija

3.5 Dropbox i Boxcryptor

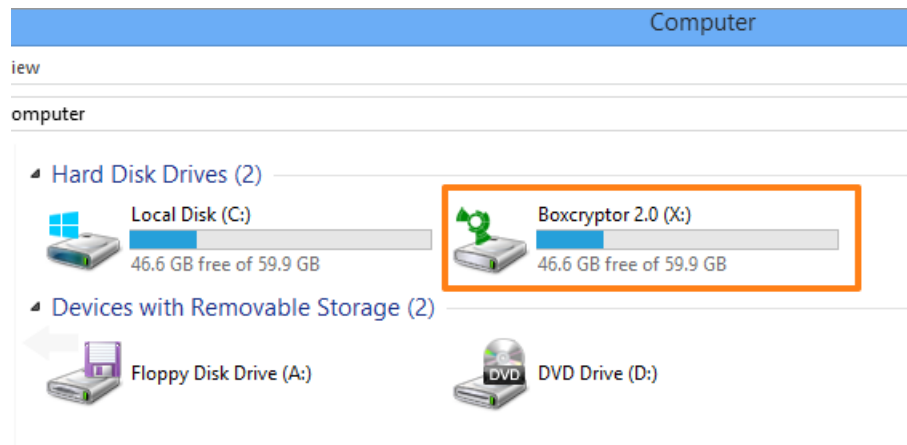
Jedno od najpopularnijih rešenja za skladištenje podataka u oblaku u svetu sigurno je Dropbox (pored Google Drive i Microsoft OneDrive). Za razliku od do sada predstavljenih rešenja, on ne nudi zaštitu podataka na strani korisnika, ali postoji mogućnost da bude razvijena u budućnosti. Što se njegovih sigurnosnih performansi tiče, one se odnose na upotrebu AES-256 šifarskog algoritma i SSL protokola za zaštitu komunikacionih kanala. Takođe, korisnicima se nudi i dodatan nivo sigurnosti upotrebom dvofaktorske autentifikacije koja može biti realizovana upotrebom aplikacije za generisanje kôda na mobilnom telefonu (engl. *Time-Based One-Time Password*) ili slanjem kôda putem tekstualnih poruka (SMS). Pre same uspostave dvofaktorske autentifikacije, korisnik dobija 16-cifarni rezervni kôd koji će moći da iskoristi ukoliko izgubi svoj mobilni telefon.

S obzirom da Dropbox ne omogućava *end-to-end* zaštitu podataka, korisnici i dalje to sebi mogu omogućiti upotrebom komercijalnih aplikacija koje su razvijene u te svrhe. Jedno od takvih rešenja je i Boxcryptor. Podržava sve veće provajdere za skladištenje u oblaku kao što su Dropbox, Google Drive, Microsoft OneDrive, SugarSync, a takođe podržava i rešenja koja koriste WebDAV standard kao što su Cubby, Strato HiDrive, and ownCloud) (slika 3.13) [26] [27].



Slika 3.13 – Boxcryptor, lista provajdera koje podržava

Boxcryptor funkcioniše tako što kreira virtuelni disk na računaru koji omogućava šifrovanje fajlova na računaru korisnika pre njihovog prebacivanja u oblak (slika 3.14). Svaki fajl koji se smesti na Boxcryptor particiju biva automatski šifrovan. Svaki fajl ima svoj jedinstven ključ generisan na slučajan način.



Slika 3.14 – Boxcrypt virtuelni disk

Boxcryptor u procesu šifrovanja koristi kombinaciju simetričnog AES-256 i asimetričnog RSA-4096 šifarskog algoritma. Svaki fajl ima svoj jedinstveni slučajno generisan ključ koji se generiše u trenutku kreiranja fajla. [27] Ključ se koristi za šifrovanje i dešifrovanje fajla na sledeći način:

Šifrovanje

1. Kreira se slučajno generisan ključ
2. Fajl se šifruje ključem
3. Ključ se šifruje korisnikovim javnim ključem
4. Tako šifrovan ključ skladišti se uz šifrovan fajl
5. Ukoliko više korisnika ima pristup fajlu, ključ se šifruje onoliko puta koliko ima korisnika, sa njihovim javnim ključevima i potom skladišti u šifrovani fajl

Dešifrovanje

1. Dešifruje se ključ upotrebom privatnog ključa korisnika
2. Dešifruje se fajl upotrebom ključa

Algoritmi

1. AES-256, CBC i PKCS7
2. RSA-4096 i OAEP

3.6 Glavne karakteristike rešenja

U nastavku će biti prikazan tabelarni pregled (tabela 3-1) glavnih karakteristika prethodno opisanih rešenja. Redosledom kojim su opisana, rešenja su predstavljena u tabeli. Prvo su predstavljeni algoritmi koji se koriste. Sva predstavljena rešenja koriste AES-256 simetričnim šifarskim algoritmom za šifrovanje podataka (sa izuzetkom MEGA koje koristi AES-128), dok se za zaštitu komunikacije upotrebljava asimetrični RSA algoritam. Takođe, tu je i heš u slučaju Tresorit i SpiderOak.

Sva rešenja dostupna su na Windows operativnom sistemu kao i u mobilnim verzijama. Neka od rešenja podržavaju rad na Linux i Mac operativnim sistemima.

Tabela 3-1 – Glavne karakteristike opisanih rešenja

	Algoritam			Veb aplikacija	Operativni sistem			
	AES	RSA	Heš		Windows	Linux	Mac	Mobilna verzija
Tresorit	256	4096	SHA-512	–	√	–	√	√
SpiderOak	256	3072	SHA-256	–	√	–	–	√
TeamDrive	256	3072	–	–	√	√	√	√
MEGA	128	2048	–	√	√	–	–	√
Dropbox	256	–	–	√	√	√	√	√
Boxcryptor	256	4096		–	√	–	√	√

4. Teorijske osnove istraživanja

Svojstvo da se informacije mogu čitati samo od autorizovanih korisnika ili primaoca predstavlja tajnost podataka. Zaštita tajnosti informacija postiže se na taj način što se u procesu šifrovanja otvorena poruka, upotrebom odgovarajućeg ključa, transformiše u novi oblik informacije - šifrat. Dešifrovanje predstavlja reverzni proces u odnosu na šifrovanje - transformiše šifrat u otvoreni tekst. Šifrovanjem se postiže zaštita tajnosti i integritet informacija.

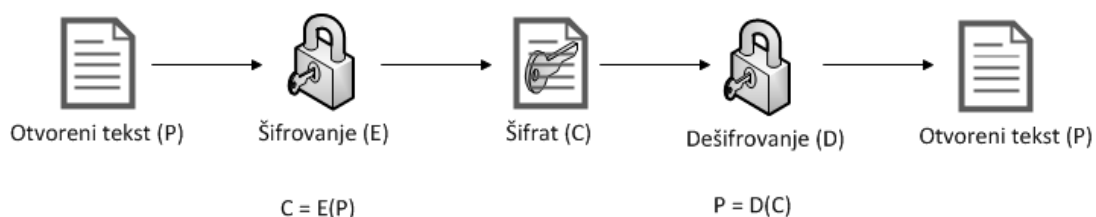
Tajnost poruke zasniva se isključivo na tajnosti ključa s obzirom da je rad svih šifarskih algoritama koji se danas koriste poznat. Ključ predstavlja početnu vrednost algoritma kojim se vrši šifrovanje, a tajnost se ogleda u tome što njega poseduje samo osoba koja vrši šifrovanje.

Tehnike šifrovanja imaju veliki značaj iz razloga što imaju glavnu ulogu u zaštiti od informatičkih prevara, povrede sigurnosti podataka, zaštiti pouzdanosti korespodencije, profesionalne tajne i u elektronskoj trgovini.

4.1 Simetrični šifarski sistemi

Simetrični šifarski sistemi predstavljaju sisteme šifrovanja tajnim ključem pri čemu je ključ za šifrovanje identičan ključu za dešifrovanje. Karakterišu se relativno velikom brzinom rada i jednostavnom implementacijom.

Kod šifrovanja poruke upotrebom simetričnih sistema, tajnost i autentičnost poruke zasnivaju se na autentičnosti ključa, a sistem je bezbedniji ukoliko se ključ generiše na što slučajniji način (slika 4.1).



Slika 4.1 – Proces šifrovanja i dešifrovanja u simetričnim šifarskim sistemima

Simetrični šifarski sistemi se dele na:

- Klasične
- Sekvencijalne
- Blokove

U nastavku će biti opisan rad blokovskih šifara.

Blokovske šifre se mogu posmatrati kao moderni naslednici klasičnih kodnih knjiga gde ključ određuje kodnu knjigu. Unutrašnji rad algoritama savremenih blokovskih šifara je kompleksan. Kod blokovskih šifara, otvoreni tekst i šifrat se dele na blokove fiksne dužine (64, 128, 192, 256) u bitovima, zatim se svaki blok nezavisno šifrjuje. Šifrat se dobija na osnovu otvorenog teksta iterativnom primenom funkcije koja se naziva runda. Ulaz u funkciju runde je ključ i izlaz prethodne runde. Blokovske šifre kombinuju svojstva konfuzije i difuzije i kompletnost. Najčešće se realizuju softverski. [5]

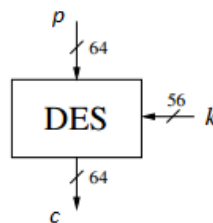
4.1.1 DES

Data Encryption Standard (DES) razvijen je 1970-tih godina. Predstavlja blokovsku šifru sa simetričnim ključem usvojenu od strane NIST-a (engl. *National Institute of Standards and Technology* - američka federalna agencija za standarde i tehnologiju). Pripada grupi modernih šifara.

Cilj dizajna je bio da se uvedu principi *konfuzije* i *difuzije*.

- **Konfuzija** – Šifrat treba da zavisi od otvorenog teksta i ključa na složen način.
- **Difuzija** – Svaki bit šifrata treba da je funkcija svih bitova otvorenog teksta i svih bitova ključa.

DES vrši šifrovanje blokova dužine 64 bita ključem dužine 56 bita (slika 4.2). [3]



Slika 4.2 – DES blokovska šifra

Svaki blok otvorenog teksta se transformiše u 16 rundi koje obavljaju identičnu operaciju. U svakoj rundi se koristi 48 bita ključa (podključ).

DES je Fejstel šifra. Fejstel šifra (nazvana po pioniru blokovskih šifara Horstu Fejstelu) ne predstavlja određenu šifru nego generalni princip dizajniranja šifri. Kod Fejstel šifri otvoreni tekst P se deli na blokove tako što se svaki blok deli na dva dela – levi L i desni R .

$$P = (L_0, R_0)$$

U svakoj rundi i ($i = 1, 2, \dots, n$), leva i desna strana se izračunavaju na sledeći način:

$$\begin{aligned} L_i &= R_{i-1} \\ R_i &= L_{i-1} \oplus F(R_{i-1}, K_i) \end{aligned}$$

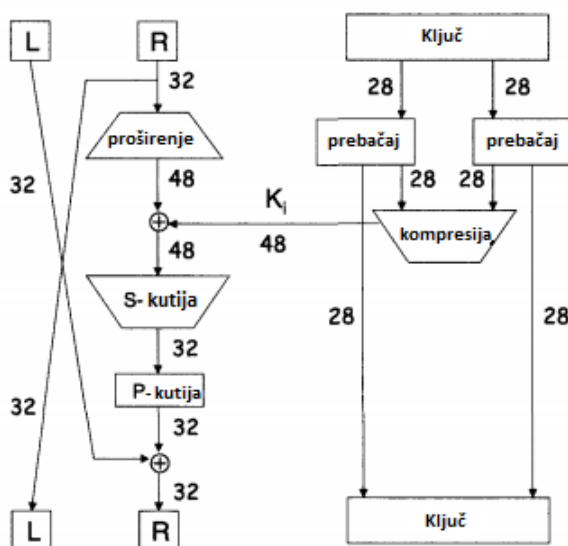
F je funkcija runde.

K_i je podključ za rundu i . Podključ je izveden iz ključa K prema specifičnom algoritmu. Izlaz iz runde je šifrat C .

$$C = (L_n, R_n)$$

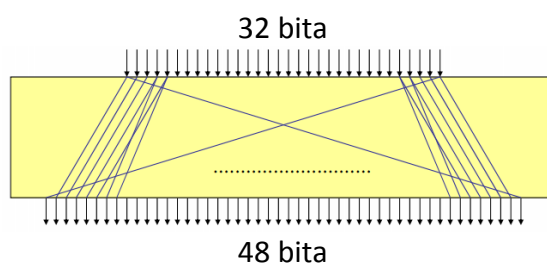
Pošto je DES Feistel šifra, šifrovanje prati pravila prikazana u jednačini:

$$F(D_{i-1}, K_i) = P - kutija (S - kutija(proširenje(D_{i-1} \oplus K_i)))$$

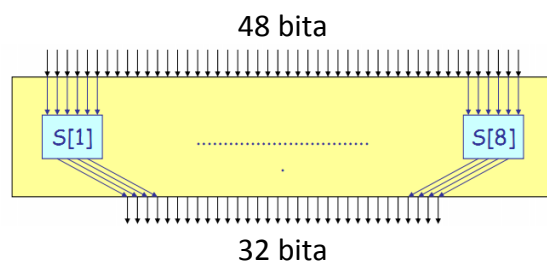


Slika 4.3 – Jedna runda DES algoritma

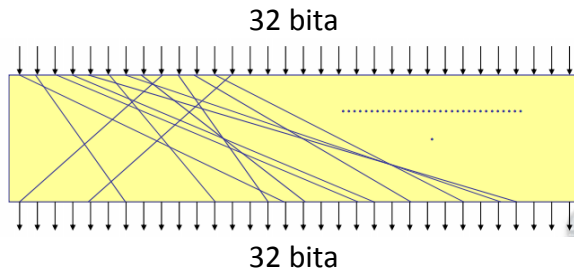
E-kutija (engl. *Expand*) proširuje i permutuje 32 ulazna bita na 48 bita i podključ je XOR-ovan sa rezultatom. Potom S-kutije (engl. *substitution*) kompresuju ta 48 bita na 32 bita pre nego što se rezultat prosledi P-kutiji (engl. *Permutation*). Izlaz iz P-kutije je XOR-ovan sa starom levom polovinom da bi se dobila nova desna polovina. [1]



Slika 4.4 – E-kutija



Slika 4.5 – S-kutija



Slika 4.6 – P-kutija

Trideset godina intenzivne krypto analize nije otkrilo nikakve propuste tipa “*back door*”. Sa aspekta sigurnosti, danas je DES ranjiv na *brute-force* napade samo zbog male dužine ključa, a ne zato što postoji neki ozbiljniji skraćeni napad. Iako su razvijeni neki napadi koji teorijski zahtevaju neznatno manje posla u odnosu na potpunu pretragu ključeva, svi dosad napravljeni programi za razbijanje DES-a jednostavno isprobavaju sve kombinacije ključeva dok ne naiđu na onu pravu, odnosno, primenjuju potpunu pretragu ključeva. [5]

4.1.2 AES

Advanced Encryption Standard (AES) je najrasprostranjenija simetrična šifra. Iako se izraz „*Standard*” odnosi samo na Vladu SAD-a, AES je obavezan za upotrebu i u raznim industrijskim standardima, a takođe se koristi i u komercijalne svrhe. U komercijalne standarde koji koriste AES spadaju standard za Internet sigurnost *IPsec*, *TLC*, standard za šifrovanje *Wi-Fi* komunikacije IEEE 802.11i, *SSH* (engl. *Secure Shell*) mrežni protokol, Internet komunikacija preko *Skype*-a i razni drugi sigurnosni proizvodi. [3]

Razvijen je krajem 1990-tih godina od strane NIST-a i zasnovan na *Rijndael* (izgovara se slično kao engl. *Rain doll*) algoritmu. AES je kao i DES, blokovska šifra sa simetričnim ključem standardizovana od strane NIST-a. Za razliku od DES-a, nije Fejstel šifra. Glavna posledica toga je da AES operacije moraju biti povratne da bi se nešto moglo dešifrovati. Takođe, za razliku od DES-a, AES algoritam ima komplikovanu matematičku strukturu.

Otporan na poznate napade, veoma je brz, moguć je paralelni dizajn, kao i implementacija na mnogim procesorima i pametnim karticama. Do danas nisu uočeni bolji napadi od *brute-force* napada na AES.

Funkcionalni parametri AES-a su:

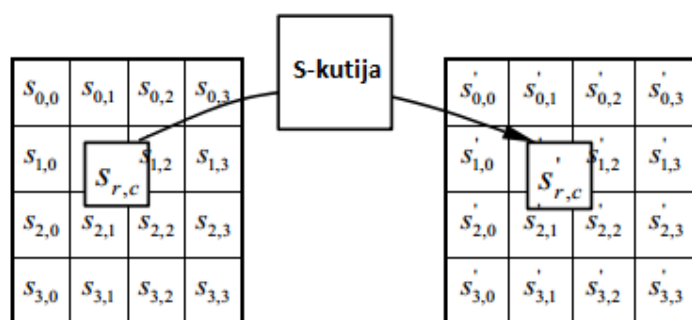
- Dužina bloka otvorenog teksta je 128, 192 i 256 bita
- Dužina ključa je 128, 192 i 256 bita
- Ima ukupno od 10 do 14 rundi, zavisno od dužine ključa
- U svakoj rundi koriste se 4 funkcije
 - Nelinearni sloj (engl. *ByteSub*)

- Sloj linearnog mešanja (engl. *ShiftRow*)
- Nelinearni sloj (engl. *MixColumns*)
- Dodatni sloj ključa (engl. *AddRoundKey*).

Bitna karakteristika AES-a je da je dužina bloka otvorenog teksta u potpunosti nezavisna od dužine ključa. To znači da je moguće šifrovati blok otvorenog teksta dužine 128 bita sa ključem dužine 256 bita.

Sve operacije AES algoritma obavljaju se na dvodimenzionalnom nizu bitova, odnosno matrici stanja. Šifrovanje, odnosno dešifrovanje se vrši tako što se ulazni blok podatka kopira u matricu stanja nad kojom se sprovode razne operacije. Završna vrednost matrice ili stanje, kopira se u izlazni šifarski blok. AES ulazne podatke tretira kao matrice dimenzije 4x4. [1]

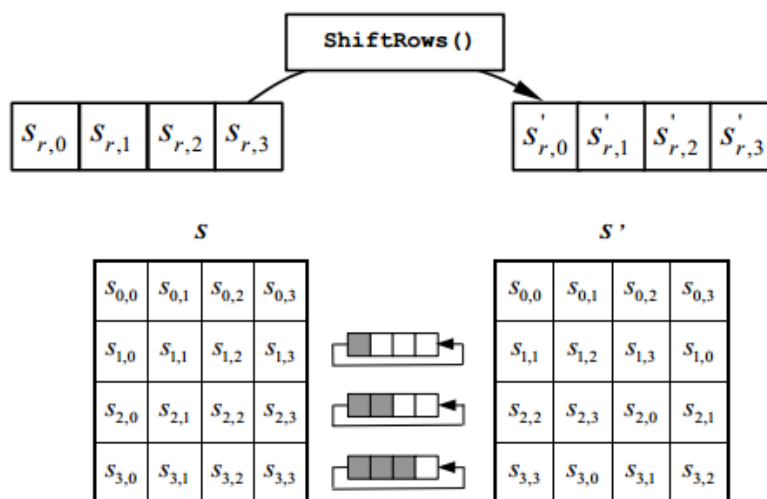
ByteStub operacija se primenjuje na svaki bajt a_{ij} tako da je $b_{ij} = \text{ByteStub}(a_{ij})$. Kao rezultat se dobija niz b_{ij} prikazan na slici 4.7.



Slika 4.7 – ByteStub operacija

ByteStub operacija je za AES isto što i S-kutija za DES. Može se posmatrati kao nelinearna, ali inverzibilna, kompozicija dve matematičke operacije ili kao tabela.

ShiftRow operacija predstavlja ciklično pomeranje bajtova u svakom redu matrice 4x4. U prvom redu ne postoji pomeranje, u drugom se pomera jedan bajt, u trećem dva i u poslednjem redu tri bajta se pomeraju u levo.



Slika 4.8 – ShiftRow operacija

MixColumn operacija se primenjuje na svaku kolonu matrice 4x4 kao što je prikazano na slici 4.9. Ovu operaciju čine operacije pomeranja i XOR-ovanja. [15]

$$\begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix} \quad \text{za } 0 \leq c < Nb.$$

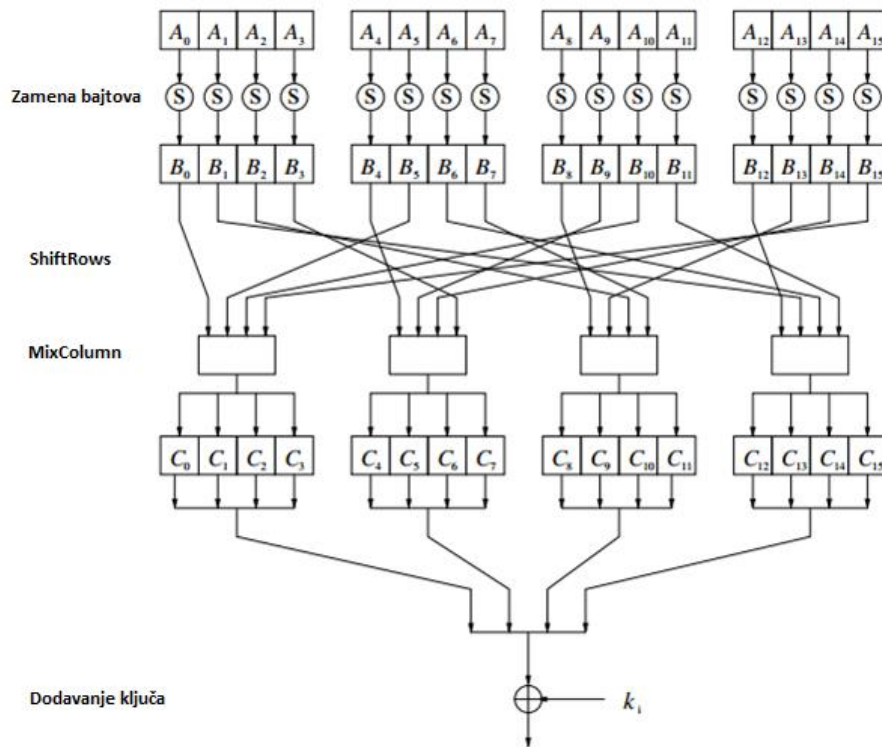
Slika 4.9 – MixColumn operacija

AddRoundKey je jednostavna operacija. Kao kod DES-a, algoritam rasporeda ključa je korišćen da bi se generisao podključ za svaku rundu.

Ako je K_{ij} matrica 4x4 podključa za određenu rundu, onda se podključ XOR-uje sa trenutnom matricom 4x4 a_{ij} koja predstavlja ulaz kao što je prikazano u nastavku: [1]

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} \oplus \begin{bmatrix} k_{00} & k_{01} & k_{02} & k_{03} \\ k_{10} & k_{11} & k_{12} & k_{13} \\ k_{20} & k_{21} & k_{22} & k_{23} \\ k_{30} & k_{31} & k_{32} & k_{33} \end{bmatrix} = \begin{bmatrix} b_{00} & b_{01} & b_{02} & b_{03} \\ b_{10} & b_{11} & b_{12} & b_{13} \\ b_{20} & b_{21} & b_{22} & b_{23} \\ b_{30} & b_{31} & b_{32} & b_{33} \end{bmatrix}$$

Slika 4.10 – AddRoundKey operacija



[3]

Slika 4.11 – AES funkcija runde za runde 1,2,..., n-1

4.2 Režimi rada blokovskih šifara

Upotrebom blokovskih šifara (DES, AES) uvek postoji potreba da se šifruje više od jednog bloka otvorenog teksta. Prilikom šifrovanja odnosno dešifrovanja, svaki otvoreni tekst se deli na blokove veličine n bita. Otvoreni tekst koji prelazi n bita se deli na n -bitne blokove. Ukoliko dužina otvorenog teksta nije deljiva sa n bita, mora biti dopunjena do veličine bloka.

Postoji više načina da se šifruje dugačak otvoreni tekst sa blokovskim šiframa. Najpoznatiji režimi rada su:

- **ECB** – Elektronska kodna knjiga (engl. *Electronic Code Book*)
- **CBC** – Ulančavanje blokova šifrata (engl. *Cipher Block Chaining*)
- **CFB** – Sprega šifrata (engl. *Cipher Feedback*)
- **OFB** – Sprega izlazne vrednosti (engl. *Output Feedback*)
- **CTR** – Brojački (engl. *Counter*)

Svi režimi rada su nezavisni (odnosno primenljivi) na svim blokovskim algoritmima. [3]

4.2.1 Elektronska kodna knjiga (ECB)

Elektronska kodna knjiga (engl. *Electronic codebook* - ECB) predstavlja najjednostavniji režim rada blokovskih šifara. Princip rada ovog režima je u tome da se za isti blok otvorenog teksta dobija isti blok šifrata. U ovom režimu svaki blok šifrata je potpuno nezavisan.

U ECB režimu, šifrovanje se vrši na sledeći način

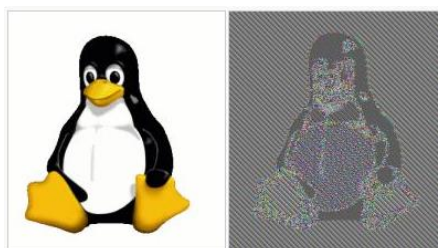
$$C_i = E(P_i, K) \quad \text{za } i = 0, 1, 2$$

a dešifrovanje

$$P_i = D(C_i, K) \quad \text{za } i = 0, 1, 2$$

Problem sa ECB režimom se ogleda u tome što se prilikom kriptanalize poređenjem više otvorenih tekstova i šifrata bez upotrebe ključa može primetiti da se delovi poruke ponavljaju. Napadač može zaključiti da je $C_i = C_j$ i iz toga zaključuje da je $P_i = P_j$. Različite poruke imaju zajedničke bitne sekvence. Redudansa poruka može biti velika. [1]

Ilustracija ECB režima prikazana je na slici 4.12.

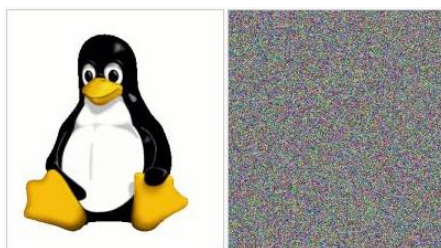


Slika 4.12 – ECB režim rada

4.2.2 Ulančavanje blokova šifrata (CBC)

Režim ulančavanja blokova šifrata (engl. *Cipher-block chaining* - CBC) predstavlja najčešće korišćen režim. Radi na principu ulančavanja blokova šifrata. Šifrat od prethodnog bloka otvorenog teksta koristi se za "zamagljivanje" otvorenog teksta sledećeg bloka, pre nego što se primeni algoritam za šifrovanje.

Ilustracija CBC režima prikazana je na slici 4.13.



Slika 4.13 – CBC režim rada

Formula za šifrovanje u CBC režimu je

$$C_i = E(P_i \oplus C_{i-1}, K) \quad \text{za } i = 0, 1, 2$$

$$C_o = E(IV \oplus P_o, K)$$

$$C_1 = E(C_o \oplus P_1, K)$$

$$C_2 = E(C_1 \oplus P_2, K)$$

a za dešifrovanje

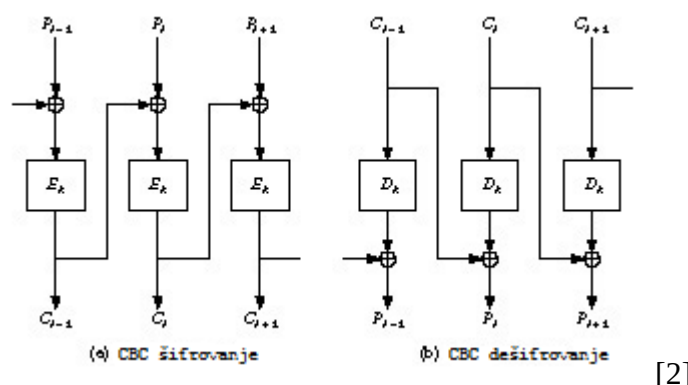
$$P_i = D(C_i, K) \oplus C_{i-1} \quad \text{za } i = 0, 1, 2$$

$$P_i = IV \oplus D(C_o, K)$$

$$P_1 = C_o \oplus D(C_1, K)$$

$$P_2 = C_1 \oplus D(C_2, K)$$

U režimu za ulančavanje blokova šifrata, otvoreni tekst se prvo XOR-uje prethodnim blokom šifrata pre nego što se šifruje. Nakon što se blok otvorenog teksta šifruje, njegov šifrat se smešta u registar povratnih vrednosti. Pre šifrovanja novog bloka otvorenog teksta, XOR-uje se sa povratnom vrednošću da bi se dobio novi ulaz za naredni blok. Prednost *CBC* režima je ta da identičan otvoreni tekst neće identično odgovarati šifratu. [1]



Slika 4.14 – Šematski prikaz rada režima za ulančavanje blokova

4.2.3 Inicijalni vektor

Inicijalni vektor se koristi samo za generisanje prvog bloka šifrata. Poželjno je da bude generisan na čisto slučajan način i nije neophodno da bude tajan. Dužina inicijalnog vektora (IV) je uvek jednaka veličini bloka date šifre. [1]

4.2.4 Ispunjavanje

Većina otvorenog teksta koji je potrebno šifrovati nije deljiv sa veličinom bloka i iz tog razloga poslednji blok nema dovoljan broj bita. Rešenje ovog problema ogleda se u ispunjavanju tog bloka. Ispunjavanje (engl. *Padding*) se može vršiti nulama, jedinicama i nulama. PKCS7 predstavlja jedan od najpopularnijih režima ispunjavanja. [2]

4.3 Asimetrični šifarski sistemi

Asimetrični šifarski sistemi (još se nazivaju i sistemi sa javnim ključevima - engl. *Public Key Cryptography* – PKI) predstavljaju šifarske sisteme koji koriste dva ključa od kojih je jedan javni, a drugi tajni (za razliku od simetričnih šifarskih sistema kod kojih se koristi isti ključ i za šifrovanje i za dešifrovanje).

Iako se razlikuju, ova dva ključa su u jedinstvenoj vezi - jedan ključ koristi se za šifrovanje, a drugi za dešifrovanje. Ni jedan ključ sam po sebi nema mogućnost obavljanja obe funkcije. Dešifrovanje nije moguće istim ključem kojim se vrši šifrovanje. Na ovaj način eliminiše se jedan od najvećih problema u simetričnim šifarskim sistemima koji se odnosi na distribuciju tajnih simetričnih ključeva.

Šifarski algoritmi u asimetričnim sistemima su takvi da se jedino primenom para ključeva može realizovati uspešno šifrovanje i dešifrovanje.

Asimetrični šifarski sistemi koriste se za efikasnu autentifikaciju, integritet poruka, neporecivost i sl., a retko kada se koriste za funkciju poverljivosti. Najčešće se kombinuju sa simetričnim šifarskim sistemima. Na primer, jedna od popularnih primena PKI sistema

je uspostavljanje simetričnog ključa za šifrovanje i dešifrovanje u simetričnom šifarskom sistemu.

Prilikom upoređivanja brzine šifrovanja, sistemi sa simetričnim ključevima su višestruko brži u odnosu na sisteme sa javnim ključevima i iz tog razloga se sistemi sa javnim ključevima ne retko koriste za prenošenje tajnog simetričnog ključa umesto šifrovanja celokupnih poruka i time se postižu odlične performanse sistema. [5]

4.3.1 RSA

RSA algoritam predložili su 1978. god. Ron Rivest, Adi Šamir i Len Adelman po kojima je i dobio ime i predstavlja najpoznatiji algoritam iz asimetrične kriptografije. Zasniva se na matematički složenom postupku faktorizacije (rastavljanja) proizvoda dva velika prosta broja na njegove faktore (činioce).

Za šifrovanje kod asimetričnih sistema se koristi jednosmerna funkcija sa zamkom. Suština je u pronalaženju funkcije E koja menja otvorenu poruku M u šifrat C , tj. $(C = E(M, K_e))$. Parametar K_e se naziva javni ključ i koristi se isključivo za šifrovanje i zbog toga E predstavlja jednosmernu funkciju. U procesu dešifrovanja koristi se vrednost K_d koja predstavlja tajni ključ i primenjuje se na inverznu funkciju $M = D(C, K_d)$. Na taj način se iz šifrata C dobija otvorena poruka M . U ovom slučaju K_d predstavlja zamku. Proces dešifrovanja je jednostavan poznavanjem zamke.

Da bi se generisao javni i tajni par ključeva za RSA, biraju se dva velika prosta broja p i q i formira se njihov proizvod $N = pq$, zatim se izabere broj e , uzajamno prost broj sa proizvodom

$$\varphi(N) = (p-1)(q-1)$$

Za dobijanje e i d potrebno je zadovoljiti jednakost

$$ed = 1 \bmod (p-1)(q-1).$$

Broj N je modul, broj e je eksponent za šifrovanje, a d je eksponent za dešifrovanje.

RSA par ključeva čine:

- Javni ključ (N, e)
- Privatni ključ (d)

Operacije šifrovanja i dešifrovanja vrše se primenom funkcija modularnog eksponenta i to na sledeći način:

- Šifrovanje: $C = M^e \bmod N$
- Dešifrovanje: $M = C^d \bmod N$

Tajnost asimetričnog šifarskog sistema sa RSA algoritmom se zasniva na složenosti faktorizacije broja N . Nakon što bi se izvršila faktorizacija broja N na činioce p i q , bilo bi lako, korišćenjem javno dostupne vrednosti e doći do privatne vrednosti d i iz uslova $ed=1(\bmod \varphi(N))$. Za dovoljno velike brojeve p i q , a samim tim i N , proces

faktorizacije je dugotrajan posao. Uz dostupne računarske resurse može se uporediti sa procesom totalne pretrage po prostoru svih ključeva kod simetričnih šifarskih sistema. Problem kod asimetričnih šifarskih sistema predstavlja ne postojanje čvrstog matematičkog dokaza da je rastavljanje na faktore jedini način za razbijanje RSA.

RSA algoritam je odabran kao metod šifrovanja podataka u drugom predloženom rešenju i odabrane specifikacije su:

- Dužina ključa je 1024 bita
- EkspONENT je 65537

Istraživanja ukazuju na to da najmanja dužina ključeva kod RSA algoritma treba da bude 1024 bita kako bi se obezbedila adekvatnu sigurnost. [1] [4] [5]

4.4 Heš funkcija

S obzirom da šifrovanje ne štiti integritet poruke, za zaštitu integriteta (autentičnosti sadržaja poruke) koristi se heš funkcija (funkcija za sažimanje poruke odnosno za generisanje skraćene verzije poruke). Za otvoreni tekst različite dužine, nakon primene heš funkcije dobija se tekst fiksne dužine. [9]

Glavna karakteristika heš funkcije u kriptografskim aplikacijama je ta da mora biti jednosmerna. [2]

$$y = h(M)$$

h – heš funkcija

M – poruka

Y – rezultat heš funkcije

Najpoznatija grupa heš funkcija je MD4 grupa. MD5, SHA i RIPEMD pripadaju ovoj grupi. [3]

4.4.1 MD5

MD5 (engl. *Message Digest*) predstavlja jednosmernu funkciju koja za ulaz proizvoljne dužine daje izlaz od 128 bita. Razvijena je 1991. godine od strane Rolanda Rivesta. Bila je široko rasprostranjena (za sigurnost Internet protokola, za poređenje sačuvanih lozinki). Poznato je da ima slabosti i da nije otporna na kriptografske napade. I pored ovoga, zbog svoje brzine rada se koristi u proveru integriteta većih fajlova. [3]

4.4.2 SHA

Nakon što su primećene slabosti kod MD5 funkcije, 1993. godine NIST je uveo novi standard - SHA (engl. *Secure Hash Algorithm*). Prvi član ove porodice je nazvan SHA, iako je on danas poznat pod nazivom SHA-0. Modifikovan je 1995. god. i dobijen

je SHA-1. Razlika između SHA-0 i SHA-1 ogleda se u rasporedu funkcija za kompresovanje koji obezbeđuje veću kriptografsku sigurnost.

NIST je 2002. god. uveo novu familiju algoritama (SHA-2) koje se razlikuju po dužini heš vrednosti: SHA-256, SHA-384 i SHA-512. SHA-224 je objavljen 2004. god. Ovi algoritmi su bezbedniji zbog dužine heš vrednosti i određenih promena u algoritmu. Takođe su kompatibilni sa AES blokovskom šifrom.

SHA-1 predstavlja industrijski standard u kriptografiji i pored toga što algoritmi iz SHA-2 porodice pružaju veću sigurnost. SHA-0 i SHA-1 algoritmi se danas ne mogu smatrati bezbednim zato što postoje kriptografski napadi koji su sposobni da pronađu koliziju za relativno kratko vreme. [3]

Tabela 4-1 – Heš algoritmi

Algoritam		Izlaz (u bitovima)	Inicijalna veličina u bitovima	Broj rundi	Kolizije
SHA-0		160			
SHA-1		160	512	80	Da
SHA-2	SHA-224	224	512	64	Za sad ne
	SHA-256	256	512	64	Ne
	SHA-384	384	1024	80	Ne
	SHA-512	512	1024	80	Ne

4.4.3 Kriptografska so

Prilikom logovanja korisnika na sistem, vrši se provera validnosti njegove lozinke. Ukoliko se u bazi čuvaju heš vrednosti lozinki, prilikom autentifikacije računa se heš vrednost unete lozinke od strane korisnika i poredi se sa vrednošću iz baze. Ukoliko se one poklapaju, korisnik je autentifikovan i omogućen mu je pristup sistemu.

S druge strane, ukoliko napadač želi da pristupi, može koristiti napad sa rečnikom (engl. *Dictionary attack*). Da bi mu se otežao posao koristi se kriptografska so (engl. *Salt*) koja predstavlja slučajne vrednosti koje se dodaju lozinci pre upotrebe heš funkcije. Kriptografska so i rezultat heš funkcije se smeštaju u bazu. Ukoliko je broj mogućih „posoljenih“ vrednosti dovoljno veliki, ovim se praktično eliminiše napad sa rečnikom, jer napadač za svaku kriptografsku so mora da generiše heš vrednost.

„Posoljena“ vrednost ne mora da bude tajna. Ona se može posmatrati kao inicijalni vektor (IV) kod blokovskih šifara čime se postiže da se isti blokovi otvorenog teksta šifruju na različit način. Po tom principu, primenom „posoljene“ vrednosti, identične lozinke dobijaju različite heš vrednosti. [1]

$$y = h(p, s)$$

p – lozinka
 s – slučajna vrednost
 h – heš funkcija
 y – rezultat heš funkcije lozinke i slučajne vrednosti

4.5 Generatori slučajnih brojeva

Slučajnost predstavlja suštinu za generisanje ključeva i jednu od glavnih sigurnosnih komponenti za mnoge kriptografske protokole. Ukoliko se za generisanje ključa koristi neodgovarajuća slučajnost, celokupna sigurnost protokola može biti ugrožena. Sa ovog stanovišta, generisanje ovakvih slučajnih podataka je od velike važnosti.

Generatori slučajnih brojeva (engl. *Random Number Generator* - RNG) se koriste u velikom broju aplikacija za simulacije i kriptografiju. Predstavljaju važan deo mnogih kriptografskih sistema, kao što su generisanje ključeva, inicijalizacija vektora, za ispunjavanje poruka i drugo.

Generatori slučajnih brojeva su uređaji koji generišu niz brojeva ili neku vrstu simbola koji se pojavljuju nasumice. Koriste se u razne svrhe, kao što su simulacija, modelovanje složenih fenomena, kriptografija i naravno zauvek popularne igre i kockanje. Postoje dva osnovna pristupa za generisanje slučajnih brojeva: [10]

- Pravi generator slučajnih brojeva (engl. *True Random Number Generator* - TRNG)
- Pseudoslučajni generator brojeva (engl. *Pseudo Random Number Generator* - PRNG)

4.5.1 Pravi generator slučajnih brojeva (TRNG)

Pravi generator slučajnih brojeva (engl. *True Random Number Generator* - TRNG) izvlači slučajnosti iz fizičkih fenomena kao što su atmosferski šum, male varijacije u pokretima miša ili vreme između poteza mišem. Uglavnom se koristi u igrama i kockanju gde je potrebna istinska slučajnost, jer su previše spori za upotrebu u statističkim i kriptografskim aplikacijama. [10]

TRNG karakteriše činjenica da njegov izlaz ne može da se reprodukuje. Na primer, ako se novčić baci 100 puta i snimi dobijena sekvenca od 100 bita, biće gotovo nemoguće da bilo ko generiše istu sekvencu od 100 bita. Šanse za uspeh su $1/2^{100}$, što predstavlja izuzetno malu verovatnoću. TRNG je zasnovan na fizičkim procesima. Neki od primera su bacanje novčića, bacanje kockice, šum poluprovodnika... U kriptografiji, TRNG je često potreban za generisanje sesijskih ključeva, koji se potom distribuiraju korisnicima. [3]

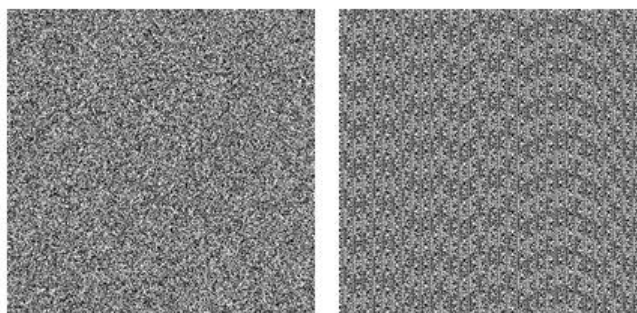
4.5.2 Pseudoslučajni generator brojeva (PRNG)

Pseudoslučajni generator brojeva (engl. *Pseudo Random Number Generator* - PRNG) predstavlja generator za generisanje pseudoslučajnih brojeva na osnovu inicijalnog stanja koje je određeno na slučajan način.

U odnosu na TRNG, PRNG je algoritam koji koristi neku matematičku formulu ili za generisanje niza brojeva koji se pojavljuju nasumice. Niz brojeva proizvedenih nije zaista slučajan. Potpuno je određen proizvoljnim početnim stanjem. PRNG je efikasan, determinističan i periodičan, što ga čini pogodnim za aplikacije gde se zahteva mnogo brojeva i gde je korisno da se isti redosled može lako ponoviti kao što su simulacije i modelovanja. PRNG treba da poseduje neke karakteristike da bi se smatrao dobrim PRNG-om, kao što su uniformnost, nezavisnost, pravilna inicijalizacija, nepredvidivost, efikasnost i prenosivost. Kvalitet slučajnosti PRNG-a se meri izlaganjem skupu statističkih testova koji se nazivaju konzervativni testovi. Svaki kriptografski siguran PRNG treba da zadovolji dva osnovna uslova: test sledećeg bita i stanje kompromisa ekstenzija. [10]

Tabela 4-2 – Karakteristike dva tipa generatora slučajnih brojeva

Karakteristike	PRNG	TRNG
Efikasnost	da	ne
Determinizam	da	ne
Periodičnost	da	ne



[14]

Slika 4.15 – Vizuelna uporedna analiza rezultata TRNG (levo) i PRNG (desno)

4.5.3 Entropija

Teorija informacija se bavi merenjem i prenosom informacija kroz komunikacioni kanal. Šenonova teorija informacija predstavlja osnovni rad u ovoj oblasti, koja pruža mnogo korisnih alata koji se zasnivaju na merenju informacije u smislu bita ili u smislu složenosti struktura potrebnih za kodiranje datih podataka.

U teoriji informacija pod pojmom entropije misli se na Šenonovu entropiju objavljenu u njegovom čuvenom radu iz 1948. godine „Matematička teorija

komunikacija“. Ona predstavlja meru za količinu neodređenosti odnosno prosečnu količinu informacije koju sadrže generisane poruke nekog informacionog izvora. [9]

Količina informacija u poruci M se meri entropijom poruke, označenom sa $H(M)$. Entropija poruke merena u bitovima je $\log_2 n$, u kojima je n broj mogućih značenja. Ovo podrazumeva da je svako značenje podjednako verovatno. [2]

Za poruku P koja predstavlja binarni niz, entropija zavisi od verovatnoće 0 i 1. Entropija binarnih nizova se definiše na sledeći način:

$$H = -p \log_2 p - (1 - p) \log_2 (1 - p)$$

Kod binarnih sekvenci, entropija H predstavlja informaciju po jednom bitu. Poželjna vrednost entropije je $H \approx 1$. Ova vrednost važi za verovatnoće pojavljivanja nula i jedinica $P_{1/0} \{ \sim 0.5, \sim 0.5 \}$.

Entropija se često koristi u vezi sa generatorima slučajnih brojeva. Redosled slučajnih promenljivih se naziva slučajan ako ima maksimalnu entropiju po bitu, što je 1:0.

4.6 Autentifikacija

Termin kontrola pristupa (engl. *Access control*) se koristi za sve bezbednosne probleme koji se odnose na pristup resursima sistema. Postoje dva područja koja su od primarnog značaja, autentifikacija i autorizacija. Autentifikacija je proces utvrđivanja da li korisniku treba dozvoliti pristup sistemu.

Na ovo mestu razmotrićemo metode koje se koriste da bi se izvršila autentifikacija osobe na lokalni računar. Posebna vrsta problema postoji kada informacija o autentifikaciji mora da prođe kroz mrežu. Iako deluje kao da su ove dve vrste problema blisko povezane one su, zapravo, potpuno različite. Kada su računarske mreže u pitanju, autentifikacija je skoro u potpunosti problem vezan za bezbednosne protokole.

Po definiciji, autentifikovanim korisnicima je dozvoljen pristup resursima sistema. Ipak, nemaju svi korisnici pun pristup svim resursima sistema. Na primer, samo privilegovan korisnik, poput administratora, može imati dozvolu da instalira softver.

Autorizacija predstavlja skup postupaka i tehnika za ograničavanje rada autentifikovanih korisnika. Ona zapravo predstavlja binarnu odluku – pristup je ili odobren ili nije, dok se autorizacija odnosi na više pravila o pristupanju različitim sistemskim resursima.

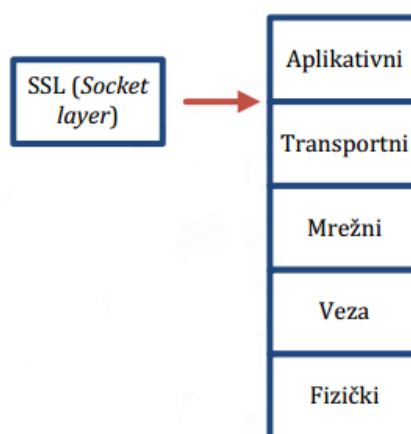
Računar može autentifikovati osobu na osnovu sledećeg:

- nešto što znate
- nešto što imate
- nešto što jeste

Lozinka je primer „nečega što znate“. One su često najslabija karika u mnogim savremenim bezbednosnim sistemima. Primer „nečega što imate“ jeste npr. platna kartica, mobilni telefon i sl. Biometrija predstavlja „nešto što jeste“. Za pojačavanje bezbednosti može se vršiti kombinacija više navedenih tehnika. [1]

4.7 SSL

SSL (engl. *Secure Sockets Layer*) protokol se koristi za većinu bezbednih transakcija preko Interneta. Nalazi se između aplikativnog i transportnog sloja u OSI modelu (slika 4.16), obično je smešten između HTTP i TCP protokola.



Slika 4.16 – SSL u OSI modelu

SSL protokol omogućava Veb pretraživaču i Veb serveru bezbednu komunikaciju, omogućava Veb pretraživaču da izvrši proveru identiteta Veb servera. SSL protokol zahteva od Veb servera posедovanje digitalnog sertifikata kako bi se ostvarila SSL konekcija. SSL koristi javni ključ za šifrovanje podataka koji se prenose preko SSL konekcije.

HTTPS (engl. *Secure Hypertext Transfer Protocol*) je komunikacioni protokol dizajniran za prenos šifrovanih informacija između računara koji komuniciraju preko Interneta. HTTPS je zapravo HTTP koji koristi SSL.

TCP/IP (engl. *Transmission Control Protocol/Internet Protocol*) reguliše prenos i rutiranje podataka preko Interneta. Drugi protokoli, poput HTTP protokola izvršavaju se iznad TCP/IP na način da svi koriste TCP/IP kao podršku za prikaz Veb stranica ili omogućavanje rada *e-mail* servera.

SSL protokol uključuje dve pod kategorije protokola: *SSL Record* protokol i *SSL Handshake* protokol. *SSL Record* protokol definiše format koji se koristi za prenos podataka. *SSL Handshake* protokol podrazumeva korišćenje *SSL Record* protokola za razmenu niza poruka između servera i klijenta koji koriste SSL prilikom prve uspostave SSL veze. Ova razmena poruka je dizajnirana da olakša sledeće akcije:

- Server se autentifikuje na klijentu
- Klijent i server se dogovaraju oko kriptografskih algoritama koje će koristiti

- Opciono – klijent se autentifikuje na serveru
- Koristite se tehnike sa javnim ključem za generisanje deljene tajne
- Uspostavlja se šifrovana SSL veza

SSL *Handshake* protokol služi za dogovor između servera i klijenta oko toga koje će šifarske algoritme koristiti za međusobnu autentifikaciju, za razmenu sertifikata i za uspostavu sesijskih ključeva. [11]

4.8 NIST testovi

NIST-ov paket testova predstavlja statistički paket koji se sastoji od 15 testova koji su razvijeni za testiranje slučajnosti binarne sekvence koje proizvode hardver ili softver na bazi kriptografski slučajnih ili pseudoslučajnih brojeva. Ovi testovi se fokusiraju na različite vrste ne slučajnosti koje bi mogle da postoje u nizu. Neki testovi su podeljeni u više podtestova.

4.8.1 Ispitivanje učestalosti u nizu

U fokusu ovog testa je ispitivanje odnosa jedinica i nula u nizu bitova. Cilj je uočavanje jednakosti pojavljivanja ove dve vrednosti. Potreban je približan broj jedinica i nula u sekvenci. Svi podtestovi koji proizilaze iz ovog testa direktno zavise od njegove uspešnosti.

Test se poziva preko metode **Frequency(n)** gde je n dužina bitnog uzorka. Metoda koristi i dodatni parametar ϵ koji predstavlja niz bitova generisanog preko RNG ili PRNG koji se testira.

Primer:

- Ulazni parametri

$\epsilon = 11001001000011111101101010100010001000010110100011$
 $00001000110100110001001100011001100010100010111000$
 $n = 100$

- Obrada

$S_{100} = -16$
 $S_{obs} = 1.6$

- Izlazni parametar

$P = 0.109599$

- Zaključak

S obzirom da je $P \geq 0.01$, smatra se da je sekvenca slučajna. [12]

4.8.2 Ispitivanje učestalosti u bloku

Test posmatra odnos jedinica i nula u n -bitnim blokovima. Cilj predstavlja uočavanje jednakosti broja jedinica i nula u svakom n -bitnom bloku.

Poziv testa se vrši preko metode **BlockFrequency(M, n)** gde su

M – dužina svakog bloka

n – dužina bitnog uzorka

Kao i kod testa za ispitivanje učestalosti u nizu, koristi se i dodati parametar ε .

Primer:

- Ulazni parametri

$\varepsilon = 11001001000011111101101010100010001000010110100011$

$00001000110100110001001100011001100010100010111000$

$n = 100$

$M = 10$

- Obrada

$N = 10$

$\chi^2 = 7.2$

- Izlazni parametar

$P = 0.706438$

- Zaključak

S obzirom da je $P \geq 0.01$, smatra se da je sekvenca slučajna. [12]

4.8.3 Ispitivanje uzastopnih ponavljanja istih bitova u nizu

Posmatrana karakteristika u ovom testu je ukupan broj uzastopnih ponavljanja jednog broja u nizu. Svrha ovog testa je da se odredi da li broj uzastopnih ponavljanja nula i jedinica odgovara očekivanoj slučajnoj sekvenci.

Poziv testa se vrši preko metode **Runs(n)** gde je n dužina bitnog uzorka. Takođe postoji i dodatni parametar ε .

Primer:- Ulazni parametri

$\varepsilon = 11001001000011111101101010100010001000010110100011$
 $00001000110100110001001100011001100010100010111000$
 $n = 100$
 $\tau = 0.02$

- Obrada

$\pi = 0.42$
 $Vn(obs) = 52$

- Izlazni parametar

$P = 0.500798$

- Zaključak

S obzirom da je $P \geq 0.01$, smatra se da je sekvenca slučajna. [12]

4.8.4 Ispitivanje najdužeg uzastopnog ponavljanja jedinica u bloku

U ovom testu se posmatra najduže uzastopno ponavljanje jedinica u n -bitnim blokovima. Svrha je određivanje da li se dužina najdužeg uzastopnog ponavljanja poklapa sa dužinom koja bi se očekivala u nizu slučajnih brojeva.

Poziv testa vrši se preko metode **LongestRunOfOnes(n)** gde je n dužina bitnog uzorka. Takođe postoji i dodatni parametar ε .

M – dužina svakog bloka. Test je podešen tako da za M koristi tri vrednosti: $M = 8$, $M = 128$ i $M = 104$ u skladu sa dužinom parametra n :

Minimum n	M
128	8
6272	128
750,000	104

Primer:

U slučaju kada je $K = 3$ i $M = 8$:

- Ulazni parametri

$\varepsilon = 11001100000101010110110001001100111000000000001001$
 $00110101010001000100111101011010000000110101111100$
 $1100111001101101100010110010$

$$n = 128$$

- Obrada

Maksimalno ponavljanje podbloka	Maksimalno ponavljanje podbloka
11001100 (2)	00010101 (1)
01101100 (2)	01001100 (2)
11100000 (3)	00000010 (1)
01001101 (2)	01010001 (1)
00010011 (2)	11010110 (2)
10000000 (1)	11010111 (3)
11001100 (2)	11100110 (3)
11011000 (2)	10110010 (2)

$$v_0 = 4; v_1 = 9; v_2 = 3; v_4 = 0; x_2 = 4.882457$$

- Izlazni parametar

$$P\text{-value} = 0.180609$$

- Zaključak

S obzirom da je $P \geq 0.01$, smatra se da je sekvenca slučajna. [12]

4.8.5 Ispitivanje stanja binarne matrice

Svrha ovog testa je provera linearne zavisnosti između podnizova fiksne dužine iz originalnog niza.

Poziv testa vrši se preko metode **Rank(n)** gde je n – dužina bitnog uzorka. Koristi se i dodati parametar ε .

Primer:

- Ulazni parametri

$$\varepsilon = 11001001000011111101101010100010001000010110100011$$

$$00001000110100110001001100011001100010100010111000$$

$$n = 100$$

- Obrada

$$N_1 = 46$$

$$N_0 = 47.5$$

$$d = -1.376494$$

- Izlazni parametar

$$P = 0.168669$$

- Zaključak

S obzirom da je $P \geq 0.01$, smatra se da je sekvenca slučajna. [12]

4.8.6 Ispitivanje diskretne Furijerove transformacije

Fokus ovog testa predstavlja najviše vrednosti diskretne Furijerove transformacije niza. Svrha ovog testa je da otkrije periodične funkcije u testiranom nizu koja bi ukazivala na odstupanje od pretpostavke o slučajnosti. Namera je da se otkrije da li je broj najviših vrednosti koje prelaze 95% značajno različit od preostalih 5%.

Poziv testa vrši se preko metode **DiscreteFourierTransform(n)** gde je n – dužina bitnog uzorka. Koriste se i dodatni parametri:

M – broj redova u matrici

Q – broj kolona u matrici

Primer:

- Ulazni parametri

ε = prvih 1000000 binarnih vrednosti proširenje ε

$$n = 100000$$

$$M = Q = 32$$

- Obrada

$$N = 97$$

$$F_M = 23, F_{M-1} = 60, N - F_M - F_{M-1} = 14$$

$$\chi^2 = 1.2619656$$

- Izlazni parametar

$$P = 0.532069$$

- Zaključak

S obzirom da je $P \geq 0.01$, smatra se da je sekvenca slučajna. [12]

4.8.7 Ispitivanje nepreklapajućih uzoraka

Karakteristika koja se posmatra u ovom testu je učestalost pojave svih mogućih n -bitnih uzoraka gde ne dolazi do preklapanja u celom ispitivanom nizu. Test bi trebalo da otkrije da li je broj pojava uzoraka koji se ne preklapaju približno jednak broju koji se očekuje za niz slučajnih brojeva.

Poziv testa vrši se preko metode **NonOverlappingTemplateMatching(m,n)** gde su

m – dužina bita u šablonu

n – dužina bitnog uzorka

Primer:

- Ulazni parametri

$\varepsilon = 2^{20}$ bitova dobijenih preko G-SHA-1 generatora

$n = 2^{20}$

$B = 000000001$

- Obrada

$\mu = 255.984375$

$\sigma^2 = 247.499999$

$W1 = 259; W2 = 229; W3 = 271; W4 = 245; W5 = 272; W6 = 262;$

$W7 = 259; W8 = 246$

$\chi^2(\text{obs}) = 5.999377$

- Izlazni parametar

$P = 0.110434$

- Zaključak

S obzirom da je $P \geq 0.01$, smatra se da je sekvenca slučajna. [12]

4.8.8 Ispitivanje preklapajućih uzoraka

Karakteristika koja se posmatra u ovom testu je učestalost pojave svih mogućih n -bitnih uzoraka gde dolazi do preklapanja u celom ispitivanom nizu. Test bi trebalo da otkrije da li je broj pojava uzoraka koji se preklapaju približno jednak broju koji se očekuje za niz slučajnih brojeva.

Poziv testa vrši se preko metode **OverlappingTemplateMatching(m,n)** gde su

m – dužina bita u šablonu

n – dužina bitnog uzorka

Primer:

- Ulazni parametri

ε = binarno proširenje ε do 1000000 bita

$n = 1000000$

$B = 111111111$

- Obrada

$v_0 = 329$; $v_1 = 164$; $v_2 = 150$; $v_3 = 111$; $v_4 = 78$; and $v_5 = 136$

χ^2 (obs) = 8.965859

- Izlazni parametar

$P = 0.110434$

- Zaključak

S obzirom da je $P \geq 0.01$, smatra se da je sekvenca slučajna. [12]

4.8.9 Maurerov univerzalni statistički test

Ovim testom se ispituju mogućnosti kompresije niza bez gubitka informacije. Za niz koji se može značajno kompresovati smatra se da nije niz slučajnih bitova zbog činjenice da je kompresija niza efikasnija ukoliko niz pokazuje periodična svojstva.

Poziv testa vrši se preko metode **Universal(L, Q, n)** gde su

L – dužina svakog bloka

Q – broj blokova u inicijalnoj sekvenci

n – dužina bitnog uzorka

Takođe postoji i dodatni parametar ε .

Primer:

- Ulazni parametri

ε = binarno proširenje ε do 1000000 bita

$n = 1000000$

$B = 111111111$

- Obrada

$v_0 = 329$; $v_1 = 164$; $v_2 = 150$; $v_3 = 111$; $v_4 = 78$; and $v_5 = 136$

$$\chi^2 (\text{obs}) = 8.965859$$

- Izlazni parametar

$$P = 0.110434$$

- Zaključak

S obzirom da je $P \geq 0.01$, smatra se da je sekvenca slučajna. [12]

4.8.10 Ispitivanje linearne kompleksnosti

Ovaj test se koristi za dobijanje dužine linearnog pomeračkog registra (engl. *Linear feedback shift register* - LFSR). Svrha ovog testa je utvrđivanje da li je sekvenca dovoljno kompleksa da se može smatrati slučajnom.

Poziv testa vrši se preko metode **LinearComplexity(M, n)** gde su

M – dužina svakog bloka

n – dužina bitnog uzorka

Koristi se i dodati parametar ε kao i K – broj stepeni slobode

Primer:

- Ulazni parametri

ε = prvih 1000000 binarnih vrednosti proširenja ε

$$n = 1000000 = 10^6$$

$$M = 1000$$

- Obrada

$$v_0 = 11; v_1 = 31; v_2 = 116; v_3 = 501; v_4 = 258; v_5 = 57; v_6 = 26$$

$$\chi^2(\text{obs}) = 2.700348$$

- Izlazni parametar

$$P = 0.845406$$

- Zaključak

S obzirom da je $P \geq 0.01$, smatra se da je sekvenca slučajna. [12]

4.8.11 Serijski test

Test se koristi za utvrđivanje učestalosti svih mogućih preklapanja n -bitnog niza u čitavoj sekvenci.

Poziv testa vrši se preko metode **Serial(m, n)** gde su

m – dužina svakog bloka

n – dužina bitnog uzorka

Koristi se i dodati parametar ε .

Primer:

- Ulazni parametri

$\varepsilon = 1000000$ bita binarnog proširenja ε

$m = 2$

$n = 1000000 = 10^6$

- Obrada

#0s = 499971

#1s = 500029

#00s = 250116

#01s = #10s = 249855

#11s = 250174

$\psi^2_2 = 0.343128$; $\psi^2_1 = 0.003364$; $\psi^2_0 = 0.000000$

- Izlazni parametar

$\nabla\psi^2_2 = 0.339764$

$\nabla^2\psi^2_2 = 0.336400$

$P_1 = 0.843764$

$P_2 = 0.561915$

- Zaključak

S obzirom da su P_1 i $P_2 \geq 0.01$, smatra se da je sekvenca slučajna. [12]

4.8.12 Ispitivanje približne entropije

U ovom testu posmatra se takođe učestalost pojave svih mogućih preklapajućih n -bitnih uzoraka u nizu. Cilj je poređenje učestalosti preklapajućih blokova sa očekivanim rezultatima.

Poziv testa vrši se preko metode **ApproximateEntropy(m, n)** gde su

m – dužina svakog bloka

n – dužina bitnog uzorka

Koristi se i dodati parametar ε .

Primer:

- Ulazni parametri

$\varepsilon = 11001001000011111101101010100010001000010110100011$

$00001000110100110001001100011001100010100010111000$

$m = 2$

$n = 100$

- Obrada

$ApEn(m) = 0.665393$

$\chi^2(obs) = 5.550792$

- Izlazni parametar

$P = 0.235301$

- Zaključak

S obzirom da je $P \geq 0.01$, smatra se da je sekvenca slučajna. [12]

4.8.13 Ispitivanje slučajnih zbirova

Svrha ovog testa je potreba da se utvrdi da li je kumulativni zbir parcijalnih sekvenci koje se javljaju u nizu koji se testira suviše veliki ili suviše mali u odnosu na očekivano ponašanje tog kumulativnog zbira slučajnih sekvenci.

Poziv testa vrši se preko metode **CumulativeSums(mode,n)** gde su

$mode$ – $mode = 0$ ili $mode = 1$

n – dužina bitnog uzorka

Koristi se i dodati parametar ε .

Primer:- Ulazni parametri

$$\varepsilon = 11001001000011111101101010100010001000010110100011$$

$$00001000110100110001001100011001100010100010111000$$

$$n = 100$$

$$mode = 0 \mid \mid mode = 1$$
- Obrada

$$z = 1.6 \mid \mid z = 1.9$$
- Izlazni parametar

$$P = 0.219194 \mid \mid P = 0.114866$$
- Zaključak

S obzirom da je $P \geq 0.01$, smatra se da je sekvenca slučajna. [12]

4.8.14 Ispitivanje slučajne digresije

Svrha ovog testa je da se utvrdi da li broj određenih stanja u okviru ciklusa odstupa od onoga što bi se očekivalo od slučajne sekvence. Ovaj test je zapravo niz od osam testova (i zaključaka), jedan test i zaključak za svako od stanja: -4, -3, -2, -1 i +1, +2, +3, +4.

Poziv testa vrši se preko metode **RandomExcursions(n)** gde je n – dužina bitnog uzorka. Koristi se i dodati parametar ε .

Primer:- Ulazni parametri

$$\varepsilon = 1000000 \text{ bita binarnog proširenja } \varepsilon$$

$$n = 1000000 = 10^6$$
- Obrada

$$J = 1490$$

Stanje = x	x^2	P	Zaključak
-4	3.835698	0.573306	slučajan
-3	7.318707	0.197996	slučajan
-2	7.861927	0.164011	slučajan
-1	15.692617	0.007779	nije slučajan
+1	2.485906	0.778616	slučajan
+2	5.429381	0.365752	slučajan
+3	2.404171	0.790853	slučajan
+4	2.393928	0.792378	slučajan

- Zaključak

Za x, u sedam stanja važi da je $P \geq 0.01$, što dovodi do zaključka da je sekvenca slučajna. [12]

4.8.15 Ispitivanje slučajne promenljive digresije

Svrha testa je da se utvrdi da li ukupan broj stanja u kojima se niz nalazi, između [-9, -1] i [1, 9] premašuje očekivane vrednosti čisto slučajne sekvence. Ovaj test zapravo predstavlja grupu od 18 testova (i zaključaka), svaki test i zaključak za svako stanje: -9, -8,..., -1 i +1, +2,..., +9.

Poziv testa vrši se preko metode **RandomExcursionsVariant(n)** gde je n – dužina bitnog uzorka. Koristi se i dodati parametar ϵ .

Primer:

- Ulazni parametri

$\epsilon = 1000000$ bita binarnog proširenja ϵ

$n = 1000000 = 10^6$

- Obrada

J = 1490

Stanje = x	x^2	P	Zaključak
-4	3.835698	0.573306	slučajan
-3	7.318707	0.197996	slučajan
-2	7.861927	0.164011	slučajan
-1	15.692617	0.007779	nije slučajan
+1	2.485906	0.778616	slučajan
+2	5.429381	0.365752	slučajan
+3	2.404171	0.790853	slučajan
+4	2.393928	0.792378	slučajan

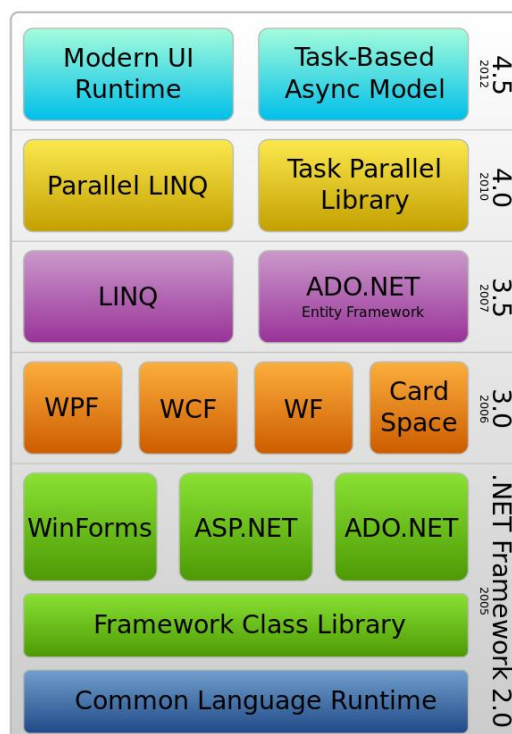
- Zaključak

S obzirom da je $P \geq 0.01$ za svako od 18 stanja za x , smatra se da je sekvenca slučajna. [12]

4.9 .NET Framework 4.5

.NET *Framework* predstavlja razvojnu platformu za izradu aplikacija za Windows, Windows Phone, Windows Server i Microsoft Azure. Sastoji se od *Common Language Runtime* (CLR) i klasnih biblioteka koje uključuje klase, interfejsse i vrednosne tipove koji podržavaju širok spektar tehnologija. .NET *Framework* omogućava pojednostavljen razvoj i primenu, kao i integraciju sa različitim programskim jezicima, uključujući Visual Basic i Visual C#. [16]

Na slici 4.17 prikazan je razvoj .NET *Framework*-a kroz verzije. Najnovija verzija je 4.5.1 koja ide uz Visual Studio 2013.



Slika 4.17 – Razvoj .NET Framework-a

4.10 Visual C#

Visual C# predstavlja moderan, viši programski jezik. Nastao je 2002. god. Kao sastavni deo *.NET Framework*-a 1.0. Predstavlja objektno-orijentisani programski jezik za izradu aplikacija na *.NET Framework* platformi u razvojnom okruženju *Visual Studio*. C# ima svoje korene u C porodici jezika i samim tim omogućava familijarnost C, C++ i Java programerima. [17]

Omogućava programerima izradu različitih sigurnih i robusnih aplikacija koje rade na *.NET Framework*. C# se može koristiti za kreiranje Windows klijentskih aplikacija, XML Veb servisa, distribuiranih komponenti, klijent-serverskih aplikacija, aplikacije za baze podataka i još mnogo drugih. Visual C# pruža napredni editor kôda, razvoj korisničkog interfejsa, integrisani program za otklanjanje grešaka, kao i mnoge druge alatke koje omogućavaju lakše razvijanje aplikacija zasnovanih na C# jeziku i *.NET Framework*-u. [18]

Na slici 4.18 prikazan je tok razvoja programskog jezika C#.

C#	1.0	1.2	2.0	3.0	3.0	4.0	5.0
C# Features			1. Generics 2. Partial types 3. Anonymous methods 4. Iterators 5. Nullable types 6. Private setters (properties) 7. Method group conversions (delegates)	1. Implicitly typed local variables 2. Object and collection initializers 3. Auto-Implemented properties 4. Anonymous types 5. Extension methods 6. Query expressions 7. Lambda expressions 8. Expression trees 9. Partial Methods		1. Dynamic binding 2. Named and optional arguments 3. Generic co- and contravariance 4. Embedded interop types ("NoPIA")	1. Asynchronous methods 2. Caller info attributes
.net framework	1.0	1.1	2.0	3.0	3.5	4.0	4.5
Framework Year	13-feb-2002	24-apr-2003	7-nov-2005	6-nov-2006	19-nov-2007	12-apr-2010	15-aug-2012
CLR	1.0	1.1	2.0	2.0	2.0	4.0	4.5
CLR New libraries	Original version	First update	Rewrite of Framework	WCF,WPF,WF	LINQ	Parallel Extensions	Asynchronous programming model
Development IDEs /Tools	VS .NET	VS .NET 2003	VS 2005	Expression Blend	VS 2008	VS 2010	VS 2012
bundled with OS	N/A	Win Server 2003	Windows Server 2003 R2	Windows Vista, Windows Server 2008	Windows 7, Win Server 2008 R2	N/A	Windows 8 ,Windows Server 2012

[19]

Slika 4.18 – Razvoj programskog jezika C#

4.11 SQL

SQL (engl. *Structured Query Language*) je standardni jezik, usklađen sa ANSI i ISO standardima, dizajniran za pristup i manipulisanje podacima u sistemima za upravljanje relacionim bazama podataka (RDBMS - engl. *Relational Database Management Systems*). RDBMS je sistem za upravljanje bazama podataka na osnovu relacionog modela (semantički model za predstavljanje podataka).

SQL predstavlja deklarisan jezik jer se njegovim rečenicama iskazuje šta se želi učiniti, dok RDBMS određuje način na koji će se to izvesti.

SQL čine nekoliko podjezika:

- Jezik za definisanje podataka (DDL – *Data Definition Language*)
- Jezik za rukovanje podacima (DML - *Data Manipulation Language*)
- Jezik za kontrolu pristupa podacima (DCL - *Data Control Language*)

DDL radi sa definicijama objekata koristeći naredbe kao što su CREATE, ALTER i DROP.

DML omogućava pristup i izmenu podataka upotrebom naredbi kao što su SELECT, INSERT, UPDATE, DELETE, TRUNCATE i MERGE.

DCL upravlja pravima pristupa pojedinačnih korisnika ili grupa, koristeći naredbe GRANT i REVOKE. [6]

4.11.1 Tipovi podataka

SQL podržava veći broj tipova podataka podeljenih u sledeće kategorije:

- Numerički tipovi podataka
- Vremenski tipovi podataka
- Tekstualni i Binarni tipovi podataka

Tabela 4-3 – Numerički tipovi podataka

Tip	Minimalna vrednost	Maksimalna vrednost	Bajtovi
BIGINT	-9,223,372,036,854,775,808	9,223,372,036,854,775,807	8
INT	-2,147,483,648	2,147,483,647	4
SMALLINT	-32,768	32,767	2
TINYINT	0	255	1

Tabela 4-4 – Vremenski tipovi podataka

Tip	Raspon	Bajtovi	Format
DATETIME	January 1, 1753, - December 31, 9999	8	'YYYYMMDD hh:mm:ss.nnn' '20090212 12:30:15.123'
SMALLDATETIME	January 1, 1900, - June 6, 2079	4	'YYYYMMDD hh:mm' '20090212 12:30'
DATE	January 1, 0001, - December 31, 9999	3	'YYYY-MM-DD' '2009-02-12'
TIME	N/A	3-5	'hh:mm:ss.nnnnnnn' '12:30:15.1234567'
DATETIME2	January 1, 0001, - December 31, 9999	6-8	'YYYY-MM-DD hh:mm:ss.nnnnnnn' '2009-02-12 12:30:15.1234567'
DATETIMEOFFSET	January 1, 0001, - December 31, 9999	8-10	'YYYY-MM-DD hh:mm:ss.nnnnnnn [+ -] hh:mm' '2009-02-12 12:30:15.1234567 +02:00'

Pored ovih osnovnih tipova, SQL podržava i druge tipove podataka kao što su npr. Blob, Spartial i druge. [6]

4.11.2 Relacioni model

Relacioni model predstavlja semantički model za upravljanje podacima. Cilj relacionog modela je da omogući konzistentno predstavljanje podataka sa minimalnom redundansom ili uopšte bez nje, kao i da definiše integritet podataka kao deo modela. RDBMS treba da sprovede relacioni model i obezbedi sredstva za skladištenje, upravljanje, proveru integriteta i izvršava upite nad podacima. [6]

4.11.3 SQL Server

SQL Server 2012 predstavlja Microsoft-ovu platformu koja podržava Oblak (engl. *Cloud*). Kompanije mogu koristiti SQL Server 2012 kako bi efikasno zaštitile svoje podatke i kako bi mogle efikasno da ih koriste na različitim platformama – preko desktopa, mobilnih uređaja, data centara do bilo privatnog ili javnog oblaka.

Nakon uspeha SQL Server 2008 R2, SQL Server 2012 je napravio snažan uticaj na organizacije širom sveta sa svojim značajnim mogućnostima. On omogućava

organizacijama odlične performanse i dostupnost. Takođe, SQL Server 2012 donosi niz hibridnih rešenja. Na primer, organizacija može razviti i postaviti aplikacije i baze podataka na tradicionalnim ne virtuelizovanim okruženjima, na privatni ili javni oblak. Štaviše, ova rešenja mogu se lako integrisati jedna sa drugim, nudeći potpuno integrisano hibridno rešenje. [7]

4.11.4 Oblak

Microsoft podržava dve vrste SQL Server oblaka: privatni i javni.

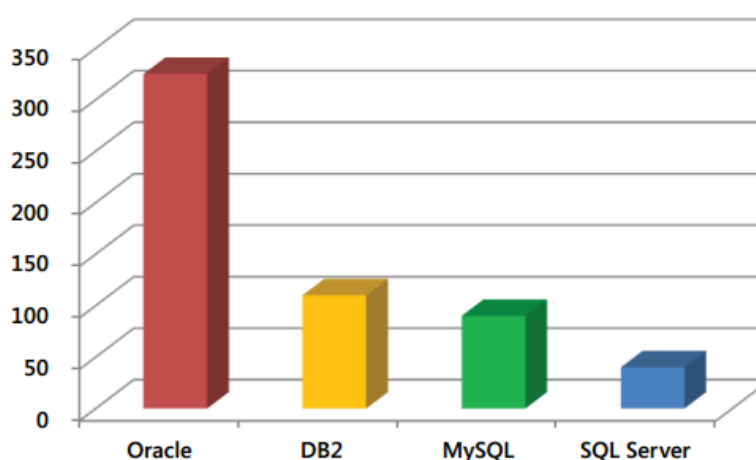
Upotreba termina privatnog oblaka može biti pomalo zbunjujuća, zbog činjenice da se skladišti lokalno, ali se u tom slučaju koristi virtualizacija. Njegovo funkcionisanje je ograničeno virtuelnom tehnologijom kao što je broj podržanih.

Javni oblak naziva se Windows Azure SQL baza (ranije nazvan SQL Azure). On se nalazi u Microsoft-ovima data centrima. Hardver, održavanje i sve nadogradnje su odgovornost Microsoft-a. Korisnik je ipak i dalje odgovoran za reindeksiranje podataka i optimizaciju upita. [7]

4.11.5 Bezbednosna poboljšanja

Prošlo je oko 10 godina od kako je Microsoft pokrenuo inicijativu za očuvanje pouzdanosti podataka. Od tada, SQL Server beleži najbolje rezultate po pitanju ranjivosti i izloženosti riziku od njegovih direktnih konkurenata na tržištu.

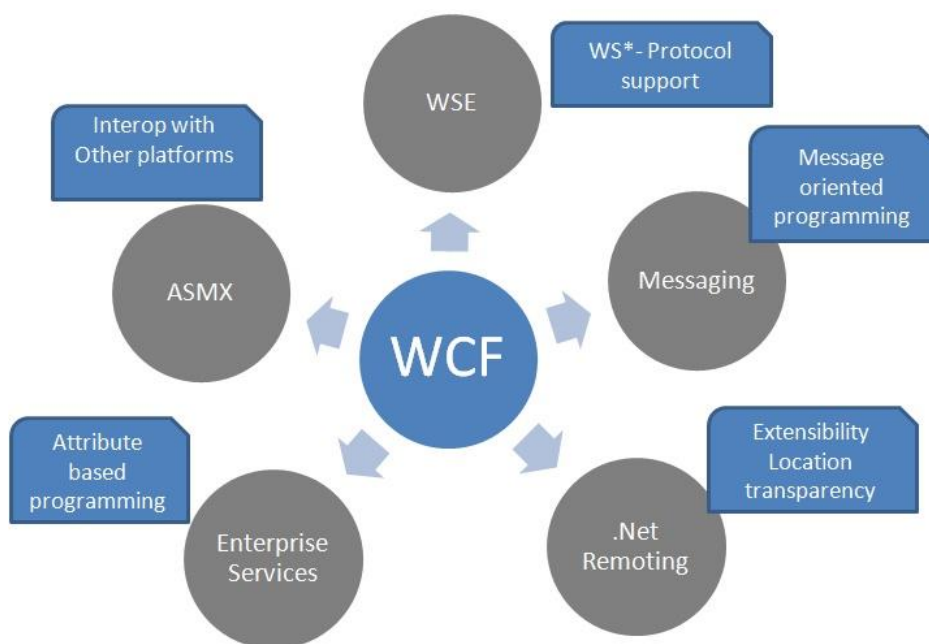
Grafikon 4-1 koji je predstavio Nacionalni institut za standarde i tehnologiju (engl. *National Institute of Standards and Technology* - NIST) prikazuje ranjivost i izloženost riziku u periodu od januara 2002 do juna 2010, četiri najveća i najpoznatija rešenja za upravljanje bazama podataka na kojem se jasno vidi da SQL Server beleži najbolje rezultate. [7]



Grafikon 4-1 – Ranjivosti i izloženosti riziku predstavljene od strane NIST-a za period od januara 2002 do januara 2010

4.12 WCF

WCF (engl. *Windows Communication Foundation*) predstavlja programsku platformu i sistem za razvoj, konfigurisanje i primenu mrežnih distribuiranih servisa. Interoperabilnost je fundamentalna karakteristika WCF-a. Predstavlja jedinstven programski model na .NET *Framework*-u od verzije 3.0. WCF sadrži kombinaciju karakteristika Veb servisa, *Remoting*, MSMQ i COM + (slika 4.19). Pruža zajedničku platformu za sve tipove komunikacija u .NET-u. [20]



Slika 4.19 – Različite tehnologije korišćene za kreiranje WCF servisa

Adrese

U WCF-u, svaki servis sadrži jedinstvenu adresu (engl. *Addresses*). Ona obezbeđuje dva važna elementa: lokaciju servisa i transportni protokol ili transportnu šemu koja se koristi za komunikaciju sa servisom. Lokacija ukazuje na ime mašine, sajta, mreže ili komunikacionog porta sa kojim servis komunicira. [8]

WCF podržava sledeće transportne šeme:

- HTTP/HTTPS
- TCP
- IPC
- *Peer network*
- MSMQ
- *Service bus*

Ugovori

U WCF-u, svi servisi izlažu ugovore (engl. *Contracts*). Ugovor je nezavisan od platforme i predstavlja standardni način opisivanja rada servisa. [8]

Postoje četiri vrste ugovora:

- **Ugovori o pružanju usluga (engl. *Service contracts*)**
Opisuju koje operacije klijent može obavljati na servisu.
- **Ugovori podataka (engl. *Data contracts*)**
Definišu koje vrste podataka se prosleđuju ka i iz servisa.
- **Ugovori o greškama (engl. *Fault contracts*)**
Definišu koje greške servis prikazuje i kako se one propagiraju do klijenata.
- **Ugovori poruka (engl. *Message contracts*)**
Omogućavaju servisu direktnu interakciju sa porukama.

Vezivanja

Postoji više aspekata komunikacije sa bilo kojim servisom, a postoji mnogo mogućih komunikacionih šablona. Poruke se mogu slati sinhrono ili asinhrono, mogu biti dostavljene odmah ili biti u redu za čekanje. [8]

WCF definiše pet često korišćenih vezivanja (engl. *Bindings*):

- **Osnovno vezivanje (engl. *Basic binding*)**
Osnovno vezivanje je dizajnirano tako da omogući novim WCF klijentima korišćenje starih ASMX servisa.
- **TCP vezivanje (engl. *TCP binding*)**
Koristi unakrsnu komunikaciju na Intranetu. Podržava različite funkcije, uključujući pouzdanost, transakcije i bezbednost, a prilagođeno je WCF ka WCF komunikaciji. Zahteva da i klijent i servis koriste WCF.
- **IPC vezivanje (engl. *IPC binding*)**
Predstavlja najsigurnije vezivanje iz razloga što ne prihvata pozive izvan mašine na kojoj se nalazi.
- **Veb servis vezivanje (engl. *Web Service (WS) binding*)**
WS vezivanje koristi HTTP ili HTTPS za transport i nudi niz karakteristika kao što su pouzdanost, transakcije i bezbednost.
- **MSMQ povezivanje (engl. *MSMQ binding*)**
MSMQ vezivanje koristi MSMQ za prenos i nudi podršku za nepovezane pozive na čekanju.

Krajnje tačke

Svaki servis povezan je sa adresom koja definiše gde se servis nalazi, vezivanje koje definiše kako se odvija komunikacija sa servisom i ugovor koji definiše šta servis radi. WCF formalizuje ovaj odnos u obliku krajnje tačke (engl. *Endpoints*). Krajnja tačka predstavlja fuziju adrese, ugovora i vezivanja (slika 4.20). [8]



Slika 4.20 – Ključna tačka

Prednosti WCF-a:

1. Može da komunicira sa drugim servisima u odnosu na .NET *Remoting* gde klijent i servis moraju da budu na .NET platformi
2. WCF servisi pružaju veću pouzdanost i sigurnost u odnosu na ASMX Veb servise
3. Ne zahteva mnogo izmena u kôdu za implementaciju bezbednosnih mehanizama. Potrebno je izvršiti promene samo u konfiguracionom fajlu
4. Sadrži integrisan sistem za logovanje koji se podešava u konfiguracionom fajlu [20]

5. Pregled predloženog rešenja

U ovom delu rada biće predstavljen kompletan proces razvoja sopstvenog rešenja za bezbedan prenos i skladištenje podataka u oblaku. Najpre će generičke šeme biti upotrebljene u svrhu objašnjenja ideje i načina njenog funkcionisanja, a zatim i njegova konkretna primena. Na kraju će, upotrebom nekoliko testova, biti prikazane ostvarene performanse rešenja.

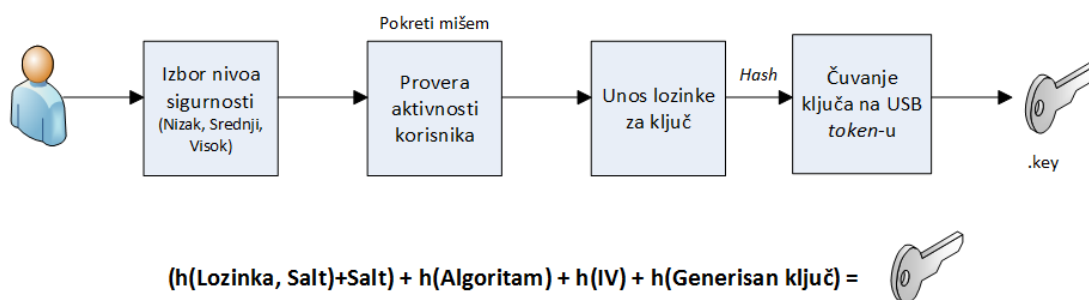
5.1 Generička šema predloženog rešenja

Osnovna ideja pri osmišljavanju ovog rešenja, a zatim i njegovoj realizaciji, bila je kreiranje rešenja koje će korisnicima omogućiti bezbedan, a s druge strane lak i intuitivan način skladištenja svojih podataka na udaljeni server (u oblaku). Za razumevanje njegovog funkcionisanja neophodno je napraviti generičku šemu.

Rešenje čine tri faze: generisanje ključa, šifrovanje i dešifrovanje. Parametri za generisanje ključa zavise od nivoa sigurnosti odabranog od strane korisnika.

Proces generisanja ključa započinje odabirom nivoa sigurnosti od koga zavisi vreme koje će biti potrebno da se ključ generiše. Nakon toga, u panelu predviđenom za to, prikupljaju se parametri tj. koordinate miša koji se koriste kao materijal za ključ. Po završetku generisanja, korisnik unosi lozinku kojom se on štiti. Materijal za ključ čine lozinka, algoritam za šifrovanje (izabran na osnovu nivoa sigurnosti) i parametri dobijeni preko miša. Jedan deo tih parametara se koristi za inicijalni vektor, a drugi za sam ključ. [13]

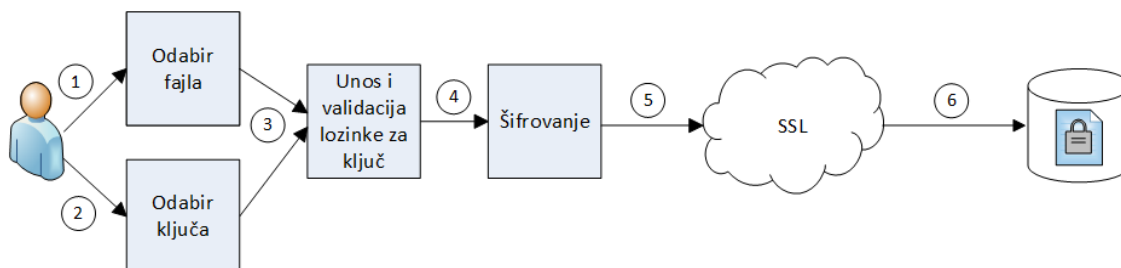
Korisnikova lozinka se čuva kao „posoljena“, pa hešovana vrednost. Zatim se vrši konkatenacija heš vrednosti algoritma, inicijalnog vektora i ključa. Nakon toga, kreira se fajl sa ekstenzijom **.key** (slika 5.1) i čuva na izabranoj lokaciji od strane korisnika (na USB *tokenu*). [13]



Slika 5.1 – Šema aplikacije - Generisanje ključa

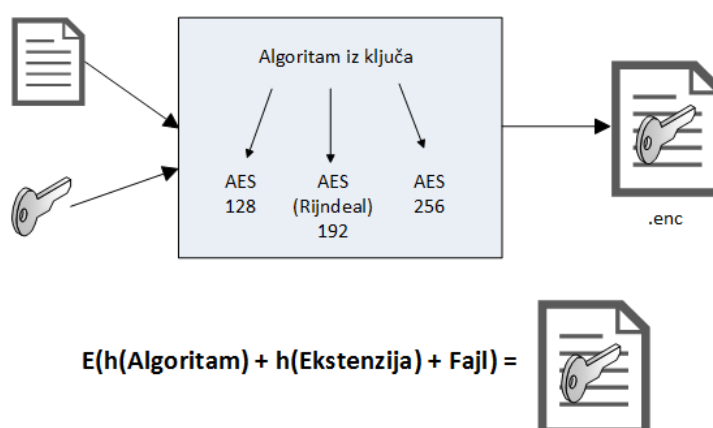
Nakon što je generisan (barem jedan) ključ, započinje proces šifrovanja (slika 5.2). Po odabiru fajla koji će se šifrovati i ključa, najpre se vrši validacija lozinke kojom je zaštićen, kako bi se onemogućila njegova upotreba od strane neovlašćenih korisnika.

Nakon uspešne validacije započinje proces šifrovanja (detaljno će biti opisan u nastavku). Tako šifrovan fajl se, preko zaštićene SSL konekcije, šalje i smešta u bazu podataka na udaljenom serveru. Ključ se ni u jednom trenutku ne šalje na server, uvek ostaje kod korisnika, smešten na njegovom USB *tokenu*.



Slika 5.2 – Šema aplikacije – Proces šifrovanja i bezbednog skladištenja fajla na udaljeni server (u oblak)

U procesu šifrovanja nakon uspešne validacije unete lozinke od strane korisnika, iz ključa se uzima algoritam kojim se fajl šifruje. Šifrovani fajl sadrži heš vrednosti algoritma i njegove originalne ekstenzije i naravno, sam fajl (slika 5.3).

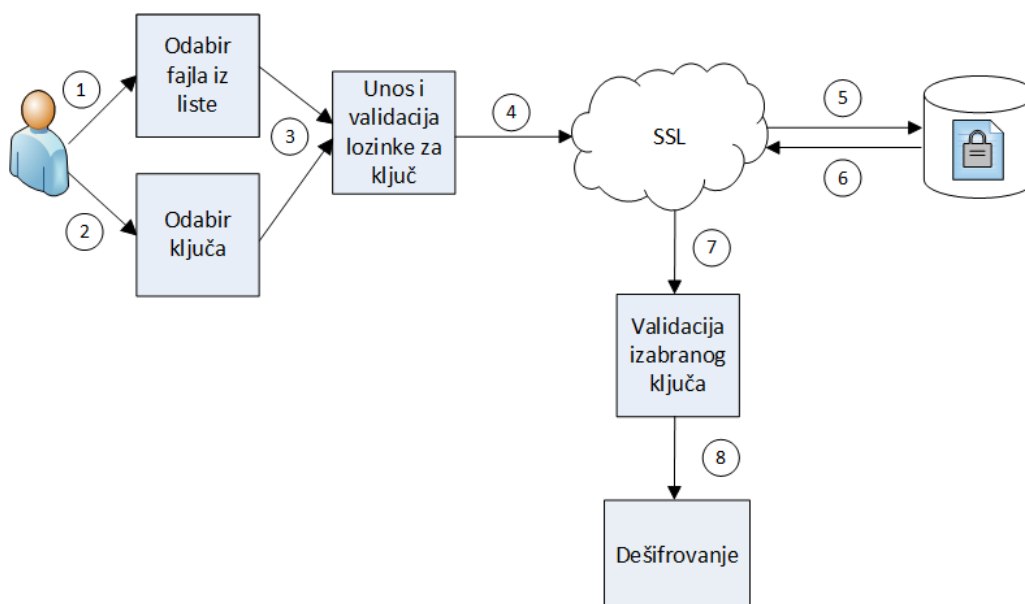


$$E(h(\text{Algoritam}) + h(\text{Ekstenzija}) + \text{Fajl}) =$$



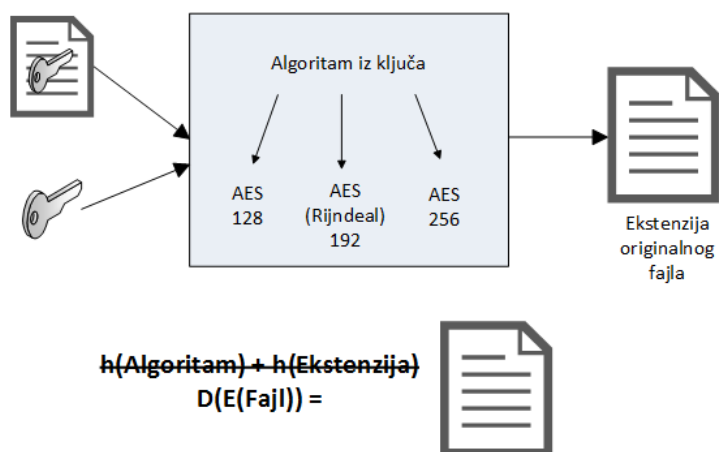
Slika 5.3 – Šema aplikacije – Šifrovanje

U fazi dešifrovanja potrebno je izabrati iz liste fajl sa servera koji se želi dešifrovati, kao i odgovarajući ključ. Kao što je i ranije objašnjeno, funkcija *end-to-end* zaštite je ta da se fajlovi dešifruju na lokalnom računaru korisnika, ne na serveru. Iz tog razloga, odabrani šifrovani fajl se preko zaštićene SSL veze prenosi na računar korisnika, gde se potom vrši validacija izabranog ključa i nakon toga se izabrani fajl dešifruje (slika 5.4).



Slika 5.4 – Šema aplikacije – Proces preuzimanja šifrovanog fajla sa udaljenog računara (iz oblaka) i njegovo dešifrovanje

Prilikom dešifrovanja, kao i prilikom šifrovanja, od korisnika se zahteva da, nakon što izabere fajl koji želi da dešifruje i odgovarajući ključ, unese lozinku. Ukoliko je ona ispravna, proverava se da li algoritam iz ključa odgovara onom iz šifrovanog fajla. Potom se iz fajla uklanjaju podaci o algoritmu i ekstenziji i on se dešifruje. Podatak o ekstenziji se koristi za vraćanje originalne ekstenzije fajla (slika 5.5).



$$h(\text{Algoritam}) + h(\text{Ekstenzija}) \\ D(E(\text{Fajl})) =$$

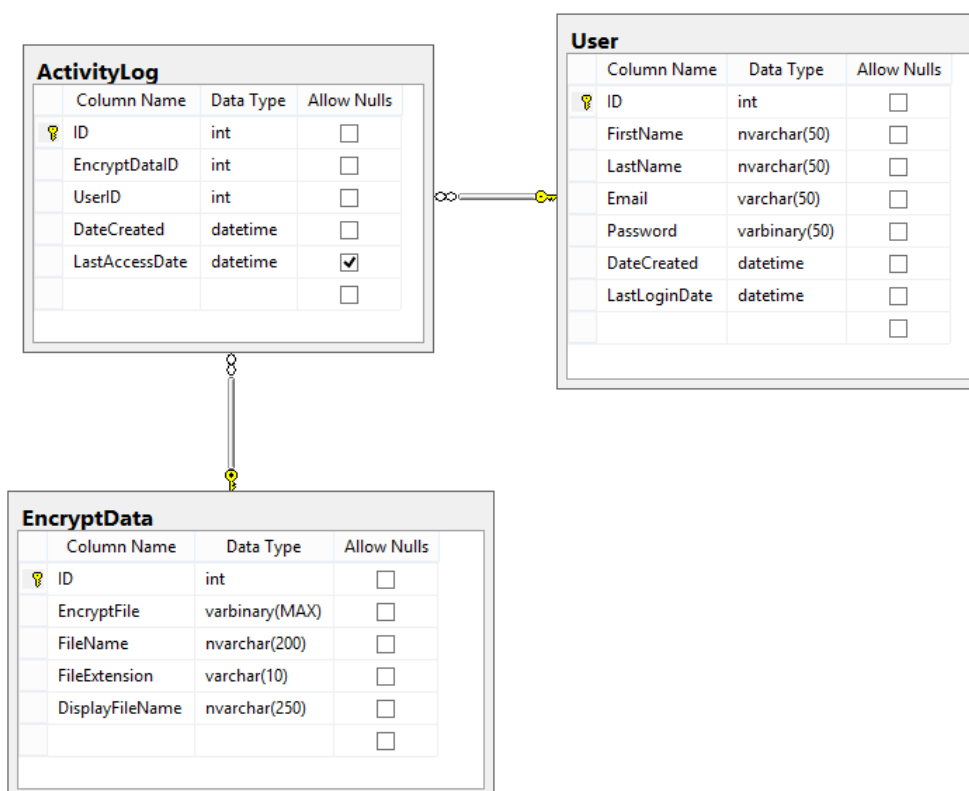
Slika 5.5 – Šema aplikacije - Dešifrovanje

Microsoft SQL Server 2012 korišćen je kao server baze podataka. Bazu čine tri tabele: *User*, *EncryptData* i *ActivityLog* (slika 5.6).

Tabela *User* čuva podatke o registrovanim korisnicima. Pored osnovnih podataka, čuvaju se datum i vreme registrovanja korisnika u sistem, kao i datum i vreme njegovog poslednjeg prijavljivanja.

Šifrovani fajlovi skladište se u *EncryptData* tabeli u binarnom formatu. U njoj se još čuvaju i podaci o nazivu fajla, njegovoj ekstenziji kao i naziv fajla koji će biti prikazan korisniku u fazi dešifrovanja.

ActivityLog tabela čuva podatke o aktivnosti korisnika u sistemu, odnosno o tome kada je korisnik kreirao određeni fajl i kada mu je poslednji put pristupio tj. dešifrovao ga.



Slika 5.6 – Dijagram baze podataka

5.2 Postavka i objašnjenje eksperimentalnog okruženja

Razvijeno rešenje čini klijent-server arhitektura. Celokupno rešenje razvijeno je na Microsoft platformi. Na serverskoj strani, za skladištenje šifrovanih podataka, koristi se SQL Server 2012 baza podataka, dok je za komunikaciju sa klijentom koristi implementiran WCF (engl. *Windows Communication Foundation*) servis. Klijenta aplikacija razvijena je u programskom jeziku C# (Microsoft .NET Framework 4.0, Visual Studio 2013). U testnom okruženju korišćena je aplikacija namenjena za te svrhe, dok je Matlab R2012b kao okruženje za grafički prikaz dobijenih rezultata.

Rešenje je razvijeno sa ciljem da korisniku omogući siguran i lak način za skladištenje svojih podataka na udaljenom serveru. To je postignuto implementacijom *end-to-end* zaštite koja podrazumeva šifrovanje fajlova na strani korisnika i njegovo skladištenje u bazi podataka na serveru. Šifrovani fajlovi se smeštaju u oblak preko SSL zaštićenog komunikacionog kanala. Na ovaj način se garantuje poverljivost podataka.

Šifrovanje svih vrsta fajlova vrši se simetričnim šifarskim algoritmima korišćenjem ključa koji je generisan na slučajan način pokretima miša. Dobijeni ključ predstavlja vrstu TRNG i podvrgnut je teorijsko-informacionoj analizi.

Kako bi se ostvarila sigurna komunikacija, neophodno je kreiranje sertifikata (serverskog i klijentskog). Za potrebe razvoja i testiranja aplikacije, generisani su *self-signed* sertifikati (nisu potpisani od strane vrhovnog CA tela).

Generisanje je vršeno u *Command Prompt*-u *Visual Studio*-a korišćenjem alatke **makecert.exe** putem sledećih naredbi (slika 5.7):

- **serverski** sertifikat

```
makecert -pe -ss My -sr LocalMachine -a sha1 -sky exchange -n  
CN=ServerCertificateMasterRad
```

- **klijentski** sertifikat

```
makecert -pe -ss MY -sr LocalMachine -a sha1 -sky exchange -n  
CN=ClientCertificateMasterRad
```

-pe – omogućava izvoz privatnog ključa, omogućava da bude izvezen zajedno sa sertifikatom

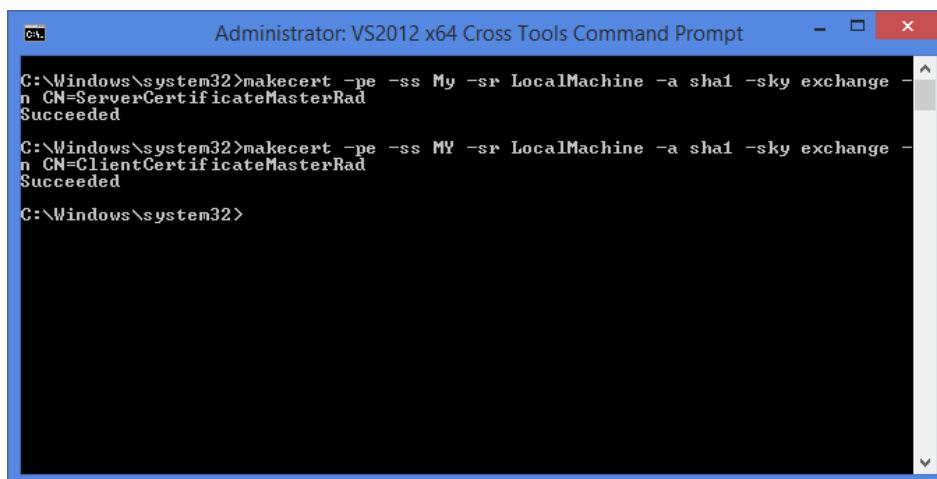
-ss – označava naziv lokacije na kojoj će sertifikat biti smešten

-sr – označava lokaciju sertifikata

-a – označava algoritam sertifikata (može biti md5, sha1, sha256, sha384 ili sha512)

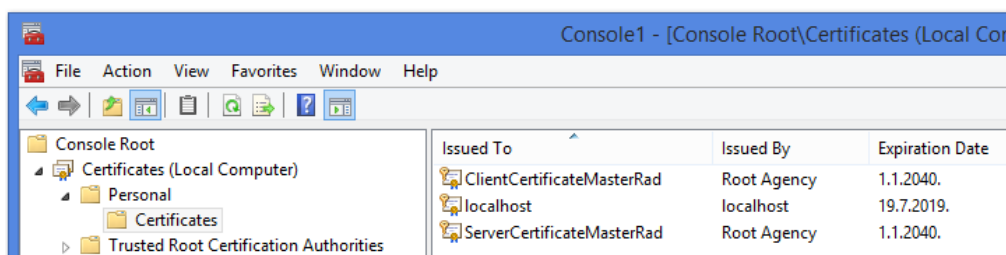
-sky – označava tip ključa - može biti *signature* ili *exchange* (ključ se koristi za šifrovanje i razmenu ključa)

-n – označava naziv sertifikata



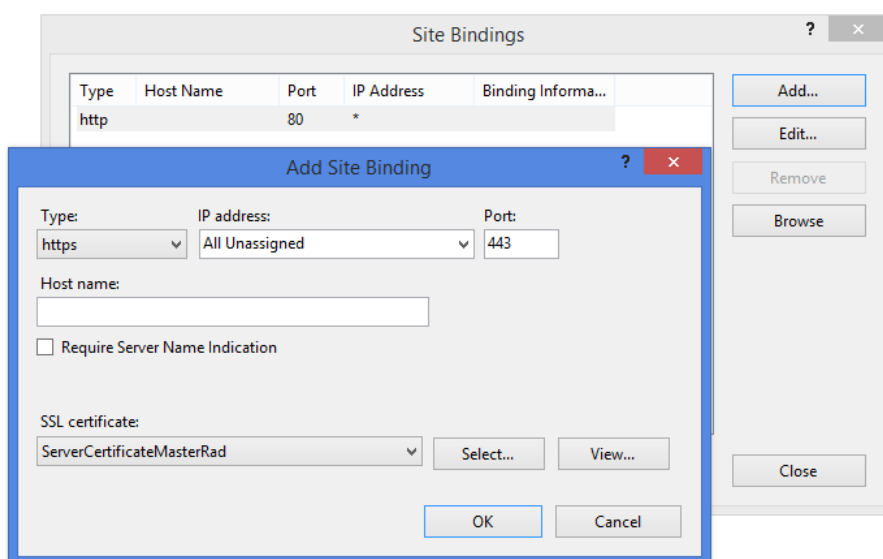
Slika 5.7 – Generisanje serverskog i klijentskog sertifikata

Generisani sertifikati se mogu videti preko *Microsoft Management Console*, navigacijom Console Root – Certificates (Local Computer) – Personal – Certificates (slika 5.8).



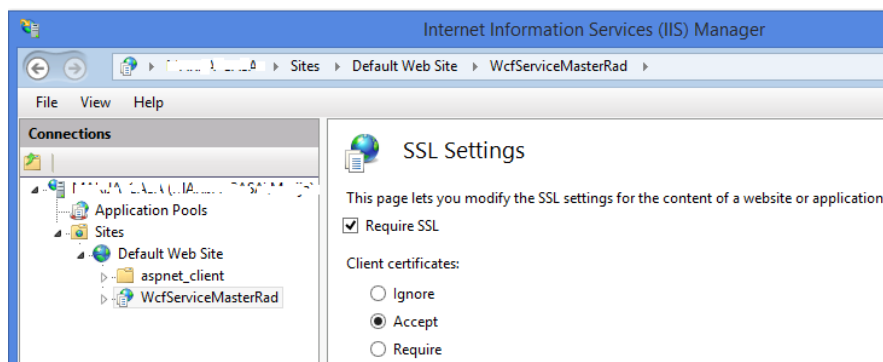
Slika 5.8 – Lokacija generisanih sertifikata

Razvijeni WCF servis postavljen je na IIS (engl. *Internet Information Service*). Da bi servis koristio sertifikat neophodno je na *Default Web Site* dodati vezu za HTTPS (slika 5.9).



Slika 5.9 – Podešavanje HTTPS na IIS

Na samom servisu, u opciji *SSL Settings* potrebno je izabrati opciju *Require SSL* (slika 5.10).



Slika 5.10 – IIS – WCF servis zahteva SSL

Na slikama 5.11 i 5.12 prikazana je konfiguracija WCF servisa i klijentske aplikacije.

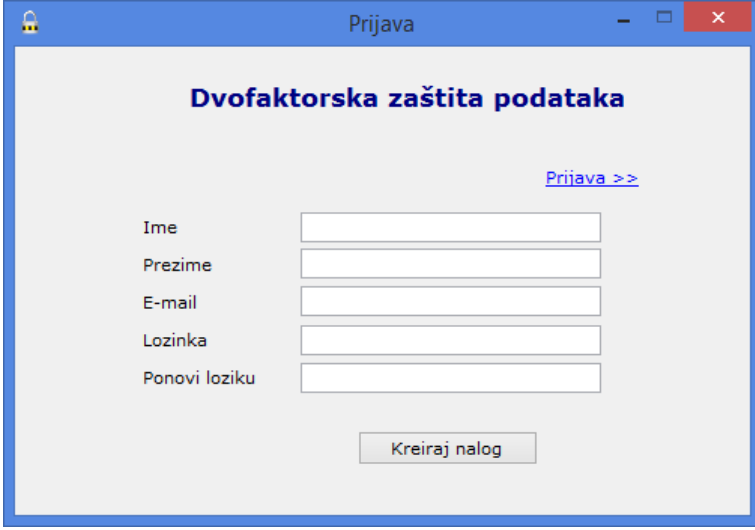
```
<system.serviceModel>
  <services>
    <service name="WcfServiceMasterRad.ServiceMasterRad" behaviorConfiguration="MasterRadCustomBinding">
      <endpoint address="" binding="customBinding" contract="WcfServiceMasterRad.IServiceMasterRad" bindingConfiguration="MasterRadCustomBinding"/>
    </service>
  </services>
  <behaviors>
    <serviceBehaviors>
      <behavior name="MasterRadCustomBinding">
        <serviceMetadata httpGetEnabled="true" httpsGetEnabled="true" />
        <serviceDebug includeExceptionDetailInFaults="false" />
        <serviceCredentials>
          <clientCertificate>
            <authentication certificateValidationMode="PeerTrust" trustedStoreLocation="LocalMachine" />
          </clientCertificate>
          <serviceCertificate findValue="ServerCertificateMasterRad" x509FindType="FindBySubjectName" storeLocation="LocalMachine" storeName="My"/>
        </serviceCredentials>
      </behavior>
    </serviceBehaviors>
  </behaviors>
  <bindings>
    <customBinding>
      <binding name="MasterRadCustomBinding" closeTimeout="00:20:00" openTimeout="00:20:00" receiveTimeout="00:20:00" sendTimeout="00:20:00">
        <security authenticationMode="CertificateOverTransport" requireSecurityContextCancellation="true"/>
        <httpsTransport />
      </binding>
    </customBinding>
  </bindings>
  <serviceHostingEnvironment aspNetCompatibilityEnabled="true" multipleSiteBindingsEnabled="true" />
</system.serviceModel>
```

Slika 5.11 – Konfiguracija WCF servisa

```
<system.serviceModel>
  <bindings>
    <customBinding>
      <binding name="MasterRadCustomBinding">
        <security authenticationMode="CertificateOverTransport"/>
        <httpsTransport />
      </binding>
    </customBinding>
  </bindings>
  <behaviors>
    <endpointBehaviors>
      <behavior name="MasterRadCustomBinding">
        <clientCredentials>
          <clientCertificate findValue="ClientCertificateMasterRad" storeLocation="LocalMachine" storeName="My" x509FindType="FindBySubjectName" />
          <serviceCertificate>
            <authentication certificateValidationMode="PeerTrust"/>
          </serviceCertificate>
        </clientCredentials>
      </behavior>
    </endpointBehaviors>
  </behaviors>
  <client>
    <endpoint address="http://localhost/ServiceMasterRad.svc"
      binding="customBinding" bindingConfiguration="MasterRadCustomBinding"
      contract="ServiceReferenceMasterRad.IServiceMasterRad" behaviorConfiguration="MasterRadCustomBinding" name="ServiceMasterRadEndpoint">
      <identity>
        <dns value="ServerCertificate"/>
      </identity>
    </endpoint>
  </client>
</system.serviceModel>
```

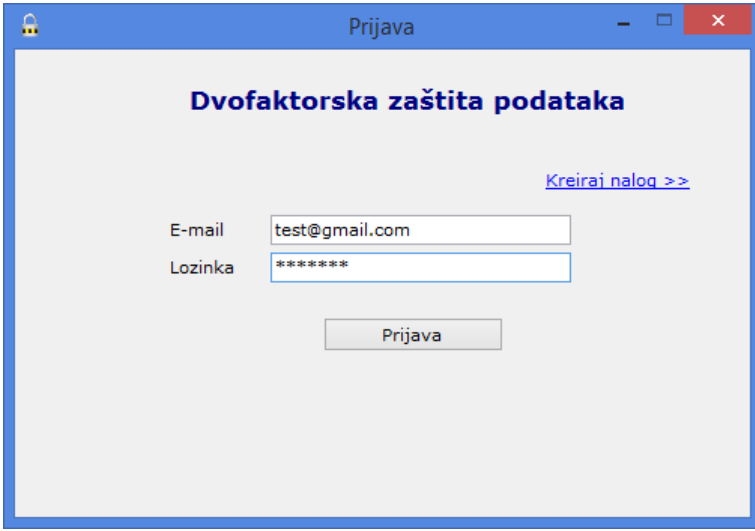
Slika 5.12 – Konfiguracija klijentske aplikacije

Prvi korak nakon instalacije klijentske aplikacije jeste registrovanje korisnika na sistem (slika 5.13). Od njega se traži da unese svoje podatke i lozinku. Oni se čuvaju u bazi podataka, a lozinka se skladišti kao hešovana vrednost (SHA-256). Minimalna dužina lozinke je 6 karaktera.

The screenshot shows a web application window titled 'Prijava'. The main heading is 'Dvofaktorska zaštita podataka'. There is a link 'Prijava >>' in the top right. The form contains five input fields: 'Ime', 'Prezime', 'E-mail', 'Lozinka', and 'Ponovi loziku'. Below these fields is a button labeled 'Kreiraj nalog'.

Slika 5.13 – Kreiranje korisničkog naloga

Nakon što se korisnik jedanput registrovao, svakim narednim pokretanjem aplikacije, od njega će biti zatraženo prijavljivanje na sistem unošenjem *e-mail*-a i lozinke kako bi se izvršila autentifikacija korisnika (vrši se validacija heša lozinke) (slika 5.14).

The screenshot shows the same 'Prijava' window. The 'E-mail' field is filled with 'test@gmail.com' and the 'Lozinka' field is filled with '*****'. The 'Kreiraj nalog >>' link is now 'Kreiraj nalog >>'. The 'Prijava' button is now visible at the bottom.

Slika 5.14 – Prijavljivanje korisnika na sistem

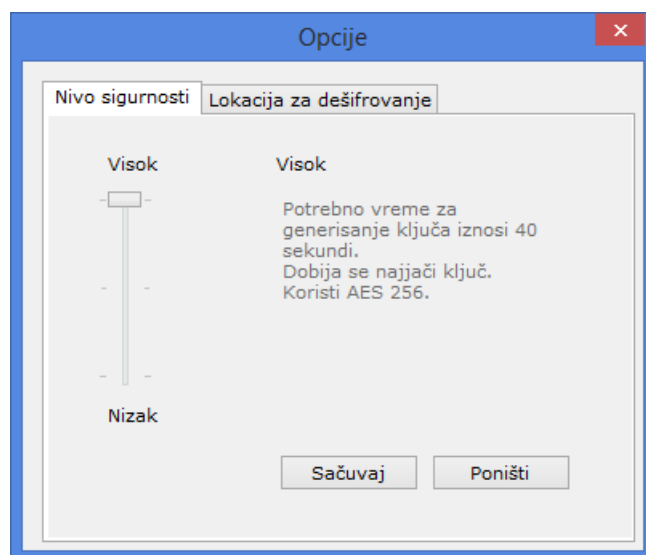
Da bi upotreba aplikacije bila moguća, neophodno je generisanje barem jednog ključa kojim se može vršiti šifrovanje fajlova. Pre samog generisanja istog, potrebno je iz menija *Alatke* odabrati opciju *Opcije* koja omogućava odabir nivoa sigurnosti koji će se koristiti kako za generisanje ključeva (vreme potrebno za njegovo generisanje) tako i za samo šifrovanje fajlova.

Definisana su tri nivoa sigurnosti:

- Nizak
- Srednji
- Visok

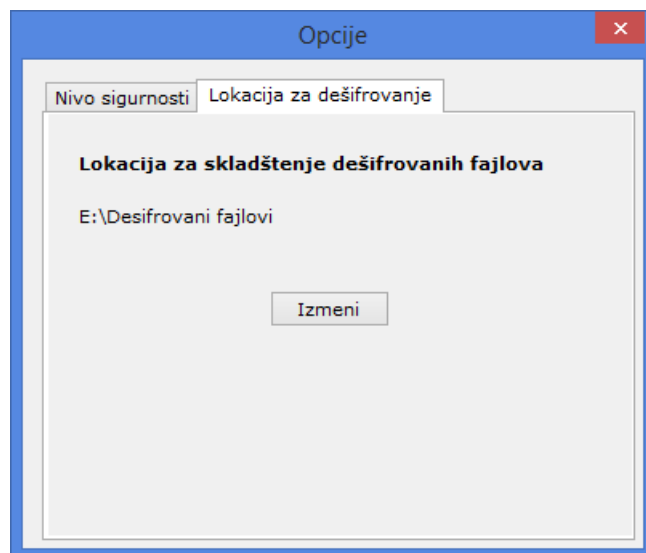
Nizak nivo koristi AES 128 (dužina bloka 128 bita, dužina ključa 128 bita) šifarski algoritam, srednji AES 192 (*Rijndael*) (dužina bloka 192 bita, dužina ključa 192 bita), dok visoki podrazumeva upotrebu AES 256 (dužina bloka 256 bita, dužina ključa 256 bita). [13]

Prilikom odabira određenog nivoa, korisnik dobija i kraće objašnjenje šta koji od nivoa znači kako bi mu bio olakšan odabir (slika 5.15). Heš funkcija SHA-256 se koristi za sve nivoie sigurnosti. [13]



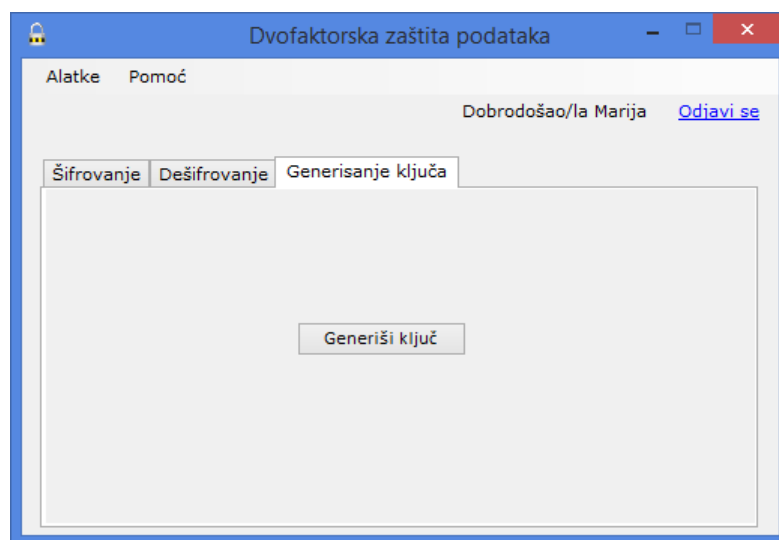
Slika 5.15 – Odabir nivoa sigurnosti

Pored ovog podešavanja, takođe je potrebno izabrati (u meniju *Alatke*, opcija *Opcije*) i lokaciju na koju će biti smešteni fajlovi nakon preuzimanja sa servera i dešifrovanja (slika 5.16).



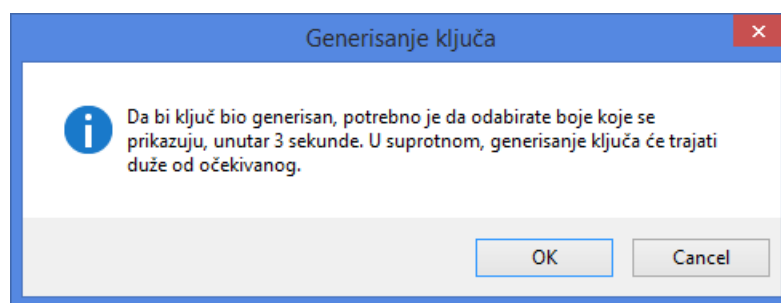
Slika 5.16 – Odabir lokacije za dešifrovanje

Glavni panel razvijenog rešenja podeljen je u tri sekcije: generisanje ključa, šifrovanje i dešifrovanje. Iz sekcije *Generisanje ključa*, odabirom dugmeta *Generiši ključ* započinje proces generisanje (slika 5.17).



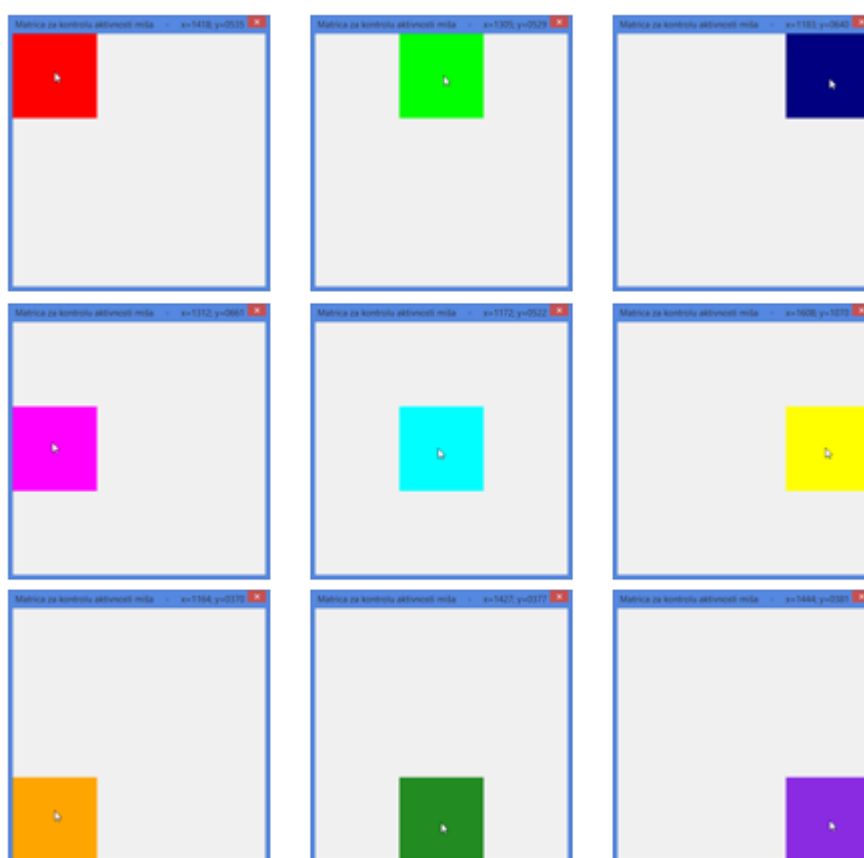
Slika 5.17 – Faza generisanja ključa

Pre samog početka, korisnik se obaveštava o akcijama koje je potrebno da preduzme tokom trajanja generisanja (slika 5.18).



Slika 5.18 – Obaveštenje pre početka generisanja ključa

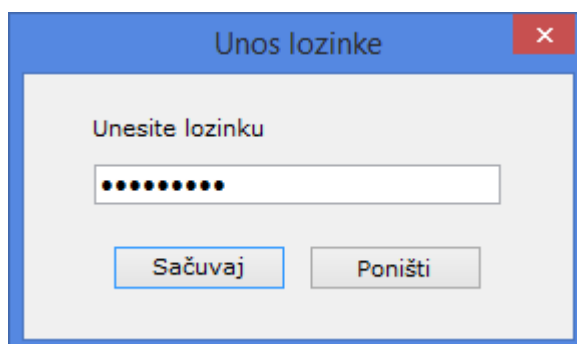
Nakon svih neophodnih podešavanja započinje proces generisanja. Od korisnika se zahteva da klikom mišem odabere označeni kvadrat. Nakon svakog klika, prikazuje se sledeći slučajno izabran kvadrat. Tokom ovog perioda, skupljaju se koordinate miša prilikom klika (slika 5.19). Ukoliko korisnik nije bio aktivan barem tri sekunde, ukupno vreme potrebno za generisanje se za toliko uvećava. Tokom generisanja, materijal za ključ se čuva u memoriji. [13]



Slika 5.19 – Proces generisanja ključa

U narednom koraku zahteva se unos lozinke kojom će se štititi taj ključ (slika 5.20). Minimalna dužina mora biti 6 karaktera. Na ovaj način u procesu šifrovanja ili dešifrovanja, korisnik će morati da koristi dvofaktorsku autentifikaciju koristeći nešto što ima (ključ) i nešto što zna (lozinka). Sledeći korak je čuvanje ključa na odabranom

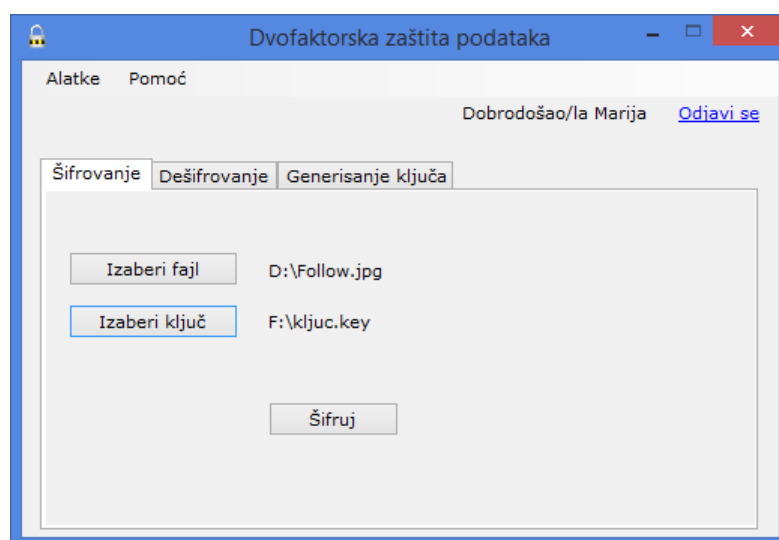
harverskom *tokenu* (USB *token*). Čuvanje na čvrstom disku nije moguće. Ključ se nalazi isključivo kod klijenta i ni u kom slučaju se ne šalje na server. Rešenje omogućava generisanje neograničenog broja ključeva za sva tri nivoa sigurnosti. [13]



Slika 5.20 – Zaštita ključa lozinkom

Kada korisnik poseduje (barem jedan) ključ, može koristiti rešenje u svrhu šifrovanja, odnosno dešifrovanja svojih fajlova.

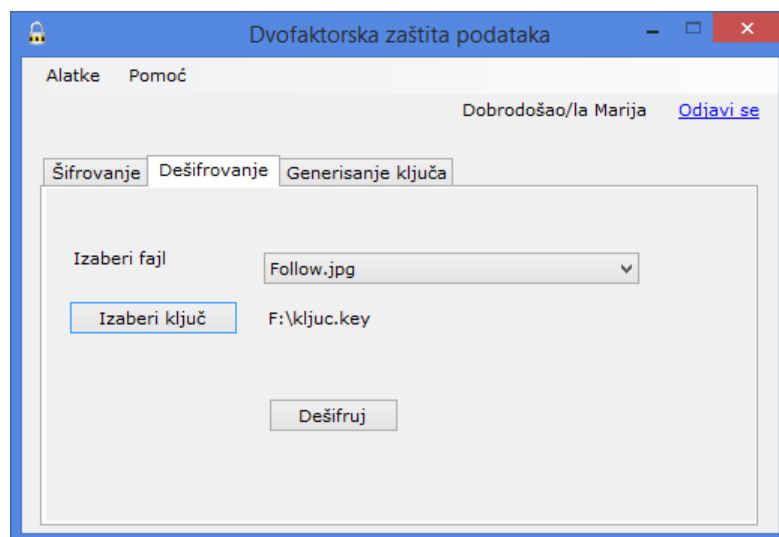
U procesu šifrovanja potrebno je izabrati fajl koji se želi šifrovati i generisani ključ kojim će se on šifrovati. Fajl koji se šifrue može se nalaziti na bilo kojoj lokaciji na klijentskom računaru osim one na kojoj se nalazi ključ (slika 5.21). Pre samog šifrovanja vrši se verifikacija ključa putem lozinke koja je unesena prilikom njegovog generisanja. [13]



Slika 5.21 – Faza šifrovanja

Tako šifrovani fajl šalje se na server. Pre samog snimanja, vrši se provera da li fajl od ranije postoji u bazi. Ukoliko postoji, korisnik se o tome obaveštava i skladištenje nije moguće. Ukoliko je sve u redu, sam fajl se smešta u *EncryptData* tabelu. Radi praćenja istorije fajla, u tabeli *ActivityLog*, beleži se aktivnost korisnika, odnosno vreme kada je i koji korisnik uskladištio fajl.

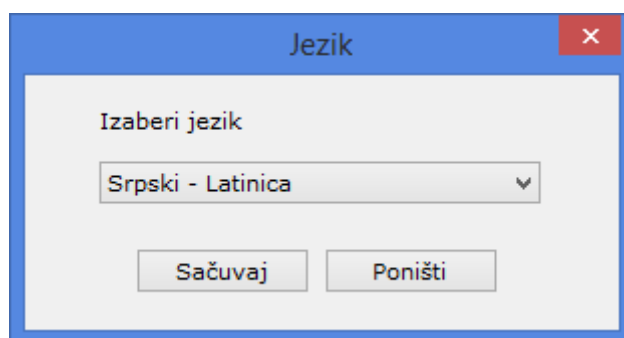
Nakon skladištenja u bazu, originalni fajl se briše sa klijentskog računara.



Slika 5.22 – Faza dešifrovanja

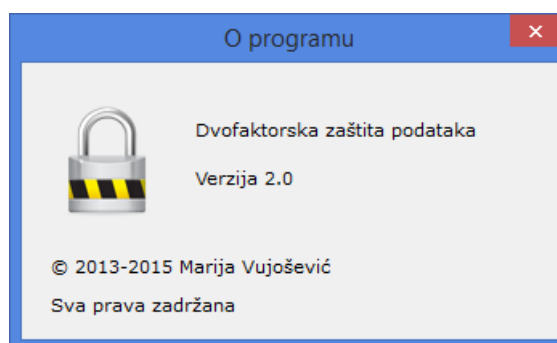
U sekciji Dešifrovanje (slika 5.22), korisniku se prikazuje lista fajlova koje je prethodno smestio u oblak. On bira onaj koji želi da dešifruje i učitava ključ (sa USB *tokena*) i potom unosi lozinku za validacija ključa. Ukoliko je validacija uspešna, šifrovani fajl se preuzima sa servera (u memoriju klijentskog računara), a zatim proverava da li je izabrani ključ odgovarajući (da li je taj korišćen i prilikom šifrovanja). Ukoliko jeste, fajl se dešifruje i skladišti na lokaciji koja je prethodno izabrana u podešavanjima aplikacije. U suprotnom, šifrovani fajl se briše iz memorije.

Sa osnovnog panela aplikacije, iz menija *Alati – Jezik*, korisnik ima mogućnost izbora jezika aplikacije. Lokalizacija je izvršena na srpski (latinica i ćirilica) i engleski jezik (slika 5.23).



Slika 5.23 – Odabir jezika aplikacije

Iz menija *O programu*, prikazuju se informacije o aplikaciji (njenoj verziji i autoru) (slika 5.24).



Slika 5.24 – O programu

❖ Specifikacija hardverske platforme

Ceo proces razvoja rešenja, kao i njegovo testiranje u eksperimentalnom okruženju sa klijentske strane sprovedeno je na hardverskoj platformi sa sledećim karakteristikama:

Tabela 5-1 – Hardverske karakteristike razvojnog i eksperimentalnog okruženja (klijentska strana)

Komponente	Opis
Procesor	Intel i5 – 3570 3.4GHz
Memorija	8GB RAM
Hard disk	1TB
Rezolucija ekrana	1280x1024
Operativni sistem	Windows 8.1

Za simulaciju okruženja u oblaku, serverska strana smeštena je na virtuelnoj mašini sa sledećim karakteristikama:

Tabela 5-2 – Hardverske karakteristike eksperimentalnog okruženja (serverska strana)

Komponente	Opis
Procesor	Intel i5 – 3570 3.4GHz
Memorija	2GB RAM
Hard disk	30GB
Rezolucija ekrana	1280x1024
Operativni sistem	Windows 8.1

5.3 Performanse predloženog okruženja sa prikazom eksperimentalnih rezultata

Jaka informaciona analiza generisanog ključa je od velike važnosti iz razloga što postavlja teorijske okvire za utvrđivanje jačine dobijenog kriptološkog ključa. Korišćenjem Šenonove entropije dolazi se do prosečne količine informacija koje su sadržane u dobijenom ključu. [13]

Nad dobijenim ključevima sprovedeni su testovi za procenu informacionog sadržaja (objašnjeni u poglavlju 4.8). Od opisanih testova, za ovo rešenje najvažniji su serijski test i ispitivanje entropije preklapajućih i nepreklapajućih uzoraka. Dobijeni rezultati predstavljeni su u uporednom prikazu sa uzorkom preuzetim sa www.random.org. Rezultati testova prikazani su u tabelama 5-3, 5-4 i 5-5. [13]

Tabela 5-3 – Rezultati serijskog testa

Serijski test		
Bigrami		
	random.org	generisan ključ
00	7773	7471
01	7828	8299
10	7827	8300
11	7821	9294
Trigrami		
	random.org	generisan ključ
000	3818	3537
001	3955	3935
010	3863	4172
011	3965	4127
100	3955	3936
101	3873	4364
110	3965	4127
111	3856	5167

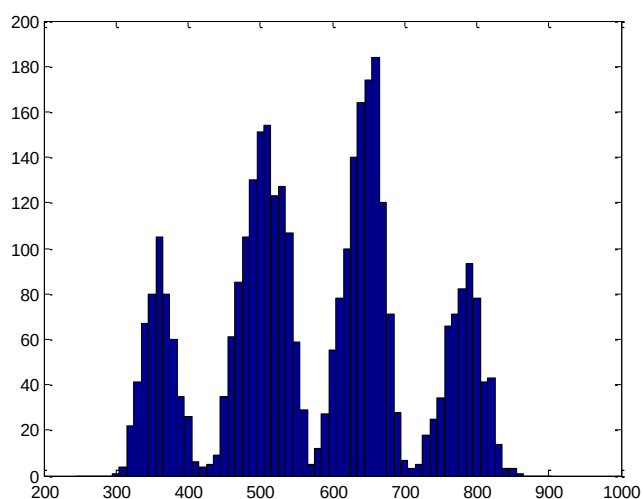
Tabela 5-4 – Rezultati ispitivanja entropije sa preklapanjem

Entropija sa preklapanjem		
	random.org	generisan ključ
Monobit	0.9999982981270112	0.9978454731456681
Bigram	0.9999969195844207	0.9978439065269784
Trigram	0.9999512260990479	0.9974066948986483
Matrica 4x4	0.9999630196341835	0.997744280937179

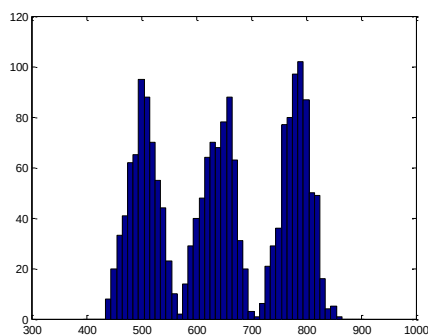
Tabela 5-5 – Rezultati ispitivanja entropije bez preklapanja

Entropija bez preklapanja		
	random.org	generisan ključ
Monobit	0.9999982981270112	0.9978454731456681
Bigram	0.9999443575006272	0.9978439065269784
Trigram	0.9999343024266536	0.9974066948986483
Matrica 4x4	0.9999630196341835	0.997744280937179

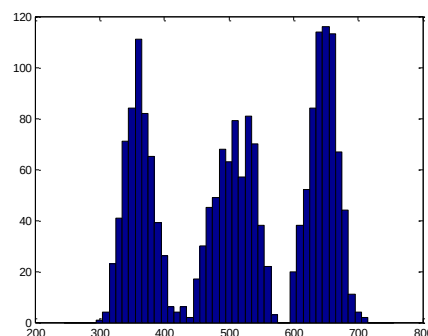
Na slikama 5.25, 5.26 i 5.27 predstavljeni su dobijeni rezultati preko histograma u Matlab-u. [13]



Slika 5.25 – Prikaz dobijenog ključa preko histograma

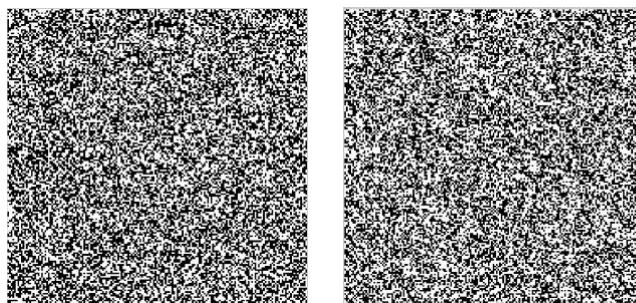


Slika 5.26 – Prikaz vrednosti dobijenih preko X ose



Slika 5.27 – Prikaz vrednosti dobijenih preko Y ose

Generisani ključ predstavljen je preko bitmape i upoređen je sa uzorkom sa random.org gde se jasno vidi da je dobijen TRNG (slika 5.28). [13]



Slika 5.28 – Vizuelna uporedna analiza rezultata sa random.org (levo) i generisanog ključa (desno)

5.4 Oblast primene predloženog rešenja

Razvijeno rešenje namenjeno je svima onima koji žele da svojim podacima obezbede visok stepen zaštite, a istovremeno omogućava lako i intuitivno korišćenje. U slučaju eventualnog otkaza računara, podaci ostaju bezbedni na serveru, a kako se generisani ključevi čuvaju na USB *tokenu*, ti isti podaci mogu biti preuzeti na bilo koji drugi računar i dešifrovani. Klijentska aplikacija zauzima vrlo malo prostora na disku.

Može se koristiti kako u privatne tako i u poslovne svrhe. Poslovni korisnici aplikaciju mogu koristiti za npr. zaštitu strogo poverljive dokumentacije i drugih fajlova koji su od vitalne važnosti za kompaniju. Sa druge strane, privatni korisnici takođe imaju potrebu za zaštitom određenih podataka, naročito ako se oni nalaze na računarima koje dele sa drugim korisnicima (članovima porodice, prijateljima...).

6. Zaključak

6.1 Sumarni ciljevi rada i ostvareni rezultati

U prvom delu ovog rada predstavljena su neka od najkvalitetnijih rešenja na tržištu iz oblasti skladištenja podataka u oblaku. Svako od njih detaljno je predstavljeno da bi na kraju bila napravljena komparativna analiza njihovih karakteristika kao što su šifarski algoritam, heš funkcija i operativni sistemi koji su podržani. Na ovaj način sagledane su njihove prednosti i nedostaci i na osnovu toga odlučeno je šta će biti implementirano u rešenju koje je predstavljeno u ovom radu.

U drugom delu predstavljene su teorijske osnove ovog istraživanja. Čitaoci ovog rada upoznati su sa simetričnim i asimetričnim šifarskim sistemima, režimima rada blokovskih šifara, heš funkcijom, autentifikacijom i razvojnim okruženjem koje je korišćeno, što im omogućava potpuno shvatanje opisa razvijenog rešenja.

Pregled predloženog i razvijenog rešenja čini završni deo rada. Ono predstavlja sistem za bezbedan prenos i skladištenje (šifrovanih) podataka u oblaku. Cilj je bio da se obezbedi *end-to-end* zaštita, odnosno prenos unapred šifrovanih fajlova na server. Autentifikacija korisnika na sistem kao i sam prenos fajlova odvija se preko SSL zaštićenog kanala čime se garantuje poverljivost podataka.

Rešenje obezbeđuje kriptografsku zaštitu svih vrsta fajlova uz implementirani modul za generisanje kriptološkog ključa preko pokreta miša koji su uzeti kao materijal za ključ zbog potrebe da se omogući što veća slučajnost, a samim tim i entropija. Drugi cilj ovog rada bio je da se postigne pravi generator slučajnih brojeva (TRNG). Takođe je omogućeno generisanje neograničenog broja ključeva uz odabir određenog nivoa sigurnosti. Dobijeni ključevi se skladište isključivo na hardverskom *tokenu* (USB *token*). Dosta pažnje je posvećeno i ergonomiji aplikacije kako bi korisnici sa lakoćom mogli da je koriste.

Generisani ključevi su podvrgnuti teorijsko-informacionoj analizi u eksperimentalnom okruženju kojom je potvrđeno da je postignuta željena slučajnost. Kao reprezentativni uzorak sa kojim su upoređivane dobijene vrednosti korišćene su vrednosti preuzete sa sajta random.org.

Da bi se sprečila zloupotreba ključeva od strane drugih korisnika ili u slučaju krađe istih, uvedena je dvofaktorska autentifikacija prilikom šifrovanja odnosno dešifrovanja fajlova. Generisani ključevi se dodatno štite lozinkom.

6.2 Predlog za budući rad

Razvijena aplikacija predstavlja verziju 2.0. Što se tiče daljeg razvoja, u planu je razvoj univerzalne aplikacije za mobilne telefone i Windows 8.1 operativni sistem (engl. *Universal App*). S obzirom da je Veb servis već razvijen, biće neophodan samo razvoj klijentskih aplikacija.

Literatura

- [1] Stamp, M., **Information security: principles and practice**, drugo izdanje, John Wiley & Sons, New Jersey, 2011.
- [2] Schneier, B., **Applied Cryptography: Protocols, Algorithms, and Source Code in C**, drugo izdanje, John Wiley & Sons, New Jersey, 1996.
- [3] Paar, C., Pelzl, J., **Understanding cryptography**, Springer, Verlag, Berlin, Heidelberg, 2010.
- [4] Adams, C., Lloyd, S., **Understanding PKI: Concepts, Standards and Deployment Considerations**, drugo izdanje, Addison Wesley, 2002.
- [5] Veinović, M., Adamović, S., **Kriptologija 1**, Univerzitet Singidunum, Beograd, 2013.
- [6] Ben-Gan, I., **Microsoft SQL Server 2012 T-SQL Fundamentals**, O'Reilly Media, Sebastopol, California, 2012.
- [7] Mistry, R., Misner, S., **Introducing Microsoft SQL Server 2012**, Microsoft Press, Redmond, Washington, 2012.
- [8] Löwy, J., **Programming WCF Services, 3rd Edition**, O'Reilly Media, Sebastopol, CA, USA, 2010.
- [9] Srivastava, G., „Pseudorandom number generator using multiple sources of entropy,” University of Victoria, 2006.
- [10] Jagannatham, A., „Mersenne Twister – A Pseudo Random Number Generator and its Variants,” George Mason University, Department of Electrical and Computer Engineering, 2008.
- [11] Bhiogade, S., „Secure Socket Layer,” u *InSITE - “Where Parallels Intersect”*, 2002.
- [12] Rukhin, A., Soto, J., Nechvatal, J., Smid, M., Barker, E., Leigh, S., Levenson, M., Vangel, M., Banks, D., Heckert, A., Dray, J., Vo, S., „A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications,” National Institute of Standards and Technology, Gaithersburg, 2010.
- [13] Vujošević, M., „Razvoj sopstvenog rešenja za kriptografsku zaštitu sa implementiranim modulom za generisanje simetričnog kriptološkog ključa,” na „Prva međunarodna naučna konferencija - Sinteza 2014”, Univerzitet Singidunum, Beograd, april.2014, 900-994, DOI: 10.15308/SINTEZA-2014-990-994
- [14] „Statistical Analysis,” Random.org, <http://www.random.org/analysis> [pregledano septembar 2014].
- [15] National Institute of Standards and Technology, „Announcing the Advanced Encryption Standard (AES),” 2001. <http://csrc.nist.gov/publications/PubsFIPS.html> [pregledano septembar 2014].
- [16] „.NET Framework 4.5,” Microsoft, [http://msdn.microsoft.com/en-us/library/vstudio/w0x726c2\(v=vs.110\)](http://msdn.microsoft.com/en-us/library/vstudio/w0x726c2(v=vs.110)) [pregledano oktobar 2014].
- [17] „Visual C# resources,” Microsoft, <http://msdn.microsoft.com/en-us/vstudio/hh341490.aspx> [pregledano oktobar 2014].
- [18] „Introduction to the C# Language and the .NET Framework”, Microsoft, <http://msdn.microsoft.com/en-us/library/z1zx9t92.aspx> [pregledano oktobar 2014].

- [19] CLR C# and dot net version comparison chart, <http://www.c-sharpcorner.com/Blogs/11177/clr-C-Sharp-and-dot-net-version-comparison-chart.aspx> [pregledano oktobar 2014].
- [20] Introduction to WCF, <http://wcftutorial.net/introduction-to-wcf.aspx> [pregledano mart 2015].
- [21] „Free 50GB Online Encrypted Storage from Tresorit,“ <http://www.sparkindark.com/free-50gb-online-encrypted-storage-tresorit.html> [pregledano april 2014].
- [22] Tresorit, <https://tresorit.com> [pregledano april 2014].
- [23] SpiderOak, <https://spideroak.com> [pregledano april 2014].
- [24] TeamDrive, <http://www.teamdrive.com> [pregledano april 2014].
- [25] MEGA, <https://mega.co.nz> [pregledano maj 2014].
- [26] Dropbox, <https://www.dropbox.com> [pregledano avgust 2014].
- [27] Boxcryptor, <https://www.boxcryptor.com> [pregledano avgust 2014].

Spisak slika

Slika 3.1 – Prikaz stanja fajla u fazama prenosa (na računaru korisnika, tokom prenosa i na serveru), prenos otvorenog teksta ili šifrovanog fajla.....	7
Slika 3.2 – Arhitektura šifarskih <i>on line</i> rešenja.....	7
Slika 3.3 – Klasičan način skladištenja podataka u oblaku	9
Slika 3.4 – Tresorit primenjuje end-to-end šifrovanje.....	9
Slika 3.5 – On-premise privatni oblak.....	11
Slika 3.6 – SpiderOak <i>host</i> -ovano skladište	11
Slika 3.7 – SpiderOak aplikacija	12
Slika 3.8 – TeamDrive arhitektura	13
Slika 3.9 – MEGA – Početni korak u povećavanju entropije RSA ključa	15
Slika 3.10 – MEGA – Procedura prikupljanja entropije na osnovu pokreta miša.....	16
Slika 3.11 – MEGA – Generisanje RSA para ključeva nakon prikupljenih podataka.....	16
Slika 3.12 – MEGA aplikacija.....	17
Slika 3.13 – Boxcryptor, lista provajdera koje podržava	18
Slika 3.14 – Boxcrypt virtuelni disk.....	19
Slika 4.1 – Proces šifrovanja i dešifrovanja u simetričnim šifarskim sistemima.....	21
Slika 4.2 – DES blokovska šifra.....	22
Slika 4.3 – Jedna runda DES algoritma	23
Slika 4.4 – E-kutija.....	23
Slika 4.5 – S-kutija.....	23
Slika 4.6 – P-kutija.....	24
Slika 4.7 – ByteStub operacija	25
Slika 4.8 – ShiftRow operacija.....	25
Slika 4.9 – MixColumn operacija	26
Slika 4.10 – AddRowKey operacija.....	26
Slika 4.11 – AES funkcija runde za runde 1,2,..., n_r-1	26

Slika 4.12 – ECB režim rada	27
Slika 4.13 – CBC režim rada	28
Slika 4.14 – Šematski prikaz rada režima za ulančavanje blokova	29
Slika 4.15 – Vizuelna uporedna analiza rezultata TRNG (levo) i PRNG (desno)	34
Slika 4.16 – SSL u OSI modelu	36
Slika 4.17 – Razvoj .NET Framework-a	49
Slika 4.18 – Razvoj programskog jezika C#	50
Slika 4.19 – Različite tehnologije korišćene za kreiranje WCF servisa	54
Slika 4.20 – Ključna tačka.....	56
Slika 5.1 – Šema aplikacije - Generisanje ključa	57
Slika 5.2 – Šema aplikacije – Proces šifrovanja i bezbednog skladištenja fajla na udaljeni server (u oblak).....	58
Slika 5.3 – Šema aplikacije – Šifrovanje	58
Slika 5.4 – Šema aplikacije – Proces preuzimanja šifrovanog fajla sa udaljenog računara (iz oblaka) i njegovo dešifrovanje.....	59
Slika 5.5 – Šema aplikacije - Dešifrovanje.....	59
Slika 5.6 – Dijagram baze podataka	60
Slika 5.7 – Generisanje serverskog i klijentskog sertifikata	62
Slika 5.8 – Lokacija generisanih sertifikata	62
Slika 5.9 – Podešavanje HTTPS na IIS.....	62
Slika 5.10 – IIS – WCF servis zahteva SSL.....	63
Slika 5.11 – Konfiguracija WCF servisa	63
Slika 5.12 – Konfiguracija klijentske aplikacije	63
Slika 5.13 – Kreiranje korisničkog naloga	64
Slika 5.14 – Prijavljivanje korisnika na sistem	64
Slika 5.15 – Odabir nivoa sigurnosti	65
Slika 5.16 – Odabir lokacije za dešifrovanje	66
Slika 5.17 – Faza generisanja ključa	66
Slika 5.18 – Obaveštenje pre početka generisanja ključa	67
Slika 5.19 – Proces generisanja ključa	67
Slika 5.20 – Zaštita ključa lozinkom	68
Slika 5.21 – Faza šifrovanja	68
Slika 5.22 – Faza dešifrovanja	69
Slika 5.23 – Odabir jezika aplikacije.....	69
Slika 5.24 – O programu	70
Slika 5.25 – Prikaz dobijenog ključa preko histograma	72
Slika 5.26 – Prikaz vrednosti dobijenih preko X ose	73
Slika 5.27 – Prikaz vrednosti dobijenih preko Y ose	73
Slika 5.28 – Vizuelna uporedna analiza rezultata sa random.org (levo) i generisanog ključa (desno).....	73

Spisak tabela

Tabela 3-1 – Glavne karakteristike opisanih rešenja	20
Tabela 4-1 – Heš algoritmi	32
Tabela 4-2 – Karakteristike dva tipa generatora slučajnih brojeva	34
Tabela 4-3 – Numerički tipovi podataka.....	51
Tabela 4-4 – Vremenski tipovi podataka	52
Tabela 5-1 – Hardverske karakteristike razvojnog i eksperimentalnog okruženja (klijentska strana).....	70
Tabela 5-2 – Hardverske karakteristike eksperimentalnog okruženja (serverska strana)..	70
Tabela 5-3 – Rezultati serijskog testa	71
Tabela 5-4 – Rezultati ispitivanja entropije sa preklapanjem	72
Tabela 5-5 – Rezultati ispitivanja entropije bez preklapanja	72

Spisak grafikona

Grafikon 4-1 – Ranjivosti i izloženosti riziku predstavljene od strane NIST-a za period od januara 2002 do januara 2010	53
---	----