Univerzitet u Beogradu

Matematički fakultet

Realizacija servisa za izdavanje digitalnih sertifikata i protokola za proveru validnosti izdatih sertifikata

Master rad

Student

Jelena Todorović

Mentor

prof. dr Vladimir Filipović

Beograd, Septembar 2017

Sadržaj

1. Uvod

Razvoj informacionih tehologija ima za posledicu problem zloupotrebe podataka na internetu. S obzirom na stepen korišćenja interneta u današnjem vremenu, problem bezbednosti postaje izuzetno značajan. Sloj sigurnih soketa (eng. *Secure Socket Layer* – SSL) je sigurnosni protokl za komunikaciju na internetu. Većina veb servera i veb pregledača podržava protokol SSL kao standardni način za bezbednu komunikaciju između klijenta i servera. Za uspešno funkcionisanje protokola SSL neophodno je da uspešno funkcioniše infrastruktura sistema sa javnim ključevima (eng. Public Key Infrastructure – PKI).

Da bi se uspešno koristila Infrastruktura sa javnim ključem, bitan je entitet autorizovan za izdavanje digitalih sertifikata ili sertifikaciono telo (eng. Certificate Authority - CA), čija je glavna uloga da digitalno potpisuje i izdaje javne ključeve vezane za određenog korisnika. Jedna od najvažnijih operacija koju treba da realizuje CA, pored izdavanja sertifikata, je povlačenje sertifikata koje je dati CA potpisao. Povlačenje sertifikata podrazumeva proglašavanje sertifikata nevalidnim, u situaciji kada je korišćenje datog sertifikata nepouzdano. Sertifikaciono telo ažurira i distribuira informacije o validnosti sertifikata koje je potpisao. Digitalni sertifikat, izdati od strane sertifikacionog tela, koristi se za sigurnu komunikaciju i dokaz identiteta.

U ovom radu dat je pregled PKI sistema, sertifikacionog tela i njegovog načina funkcionisanja, zatim kriptografske metode za šifrovanje, očuvanje integriteta podataka i proveru identiteta. Za realizaciju sertifikacionog tela korišćen je OpenSSL alat komandne linije, koji omogućava izvršavanje kriptografskih funkcija. Razvijen je softver koji implementira proces izdavanja, povlačenja i provere validnosti digitalnih sertifikata. Vođenje evidencije o statusima izdatih sertifikata predstavlja vrlo bitnu stavku, kada je reč o sigurnosti na mreži. Zbog toga je razvijeno upravljanje digitalnim sertifikatima u smislu povlačenja sertifikata i mogućnosti provere njihove validnosti. Povlačenjem sertifikata njegov status se menja iz „valid“ u „revoked“, čime se proglašava nepouzdanim za korišćenje. Trenutno postoji implementacija PKI sistema i sertifikacionog tela realizovana od

strane Microsoft-a. Microsoft-ova implementacija pruža različite opcije korisniku, kao što su zaštita privatnog ključ sertifikacionog tela, integracija sa Active Diretory-ijem i slično. U tom smislu ovo rešenje je daleko naprednije od ponuđenog rešenja u ovom radu. Medjutim, cilj ovog rada jeste simplifikacija problema radi istraživanja načina funkcionisanja sertifikacionog tela u PKI sistemu, kao i istraživanje protokola SSL i alata OpenSSL koji ga implementira.

2. Kriptografija

Glavni cilj kriptografije jeste da zaštiti bitne podatke koji se šalju medijumom koji je nezaštićen sam po sebi, u ovom slučaju to je računarska mreža [1]. Postoje različiti kriptografski algoritmi koji pružaju mogućnosti zaštite podataka, a koja se odnosi na:

Zaštita tajnosti – omogućava da podatak ili sadržaj poruke bude dostupan samo onome kome je namenjena, što se postiže šifrovanjem.

Autentifikacija - process u kome se dokazuje identitet krajnjih učesnika u komunikaciji.

Integritet podataka – garantuje da nije došlo do izmene podataka ili sadržaja poruke na njenom putu od izvora do odredišta, najčešće tako što se generiše heš poruke (niz bitova fiksne dužine). Na prijemu se istim algoritmom korišćenim pri slanju poruke formira heš i poredi sa hešom koji je primljen uz poruku. Ako se te dve vrednosti ne poklapaju, znači da je došlo do izmene originalnog saržaja.

Neporecivost transakcije – onemogućava da onaj ko je poslao poruku kasnije tvrdi da je nije poslao. Kad pošiljalac potpiše poruku svojim digitalnim potpisom, zna se da je on koristio svoj privatni ključ, kako bi formirao digitalni potpis. Kako samo pošiljalac ima pristup svom privatnom ključu, to znači da je samo on mogao da pošalje odgovarajuću digitalno potpisanu poruku.

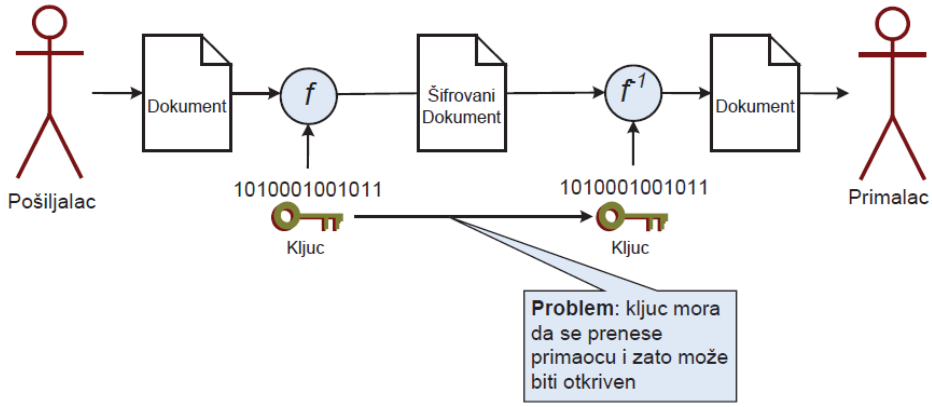
Protokoli kao sto su HTTP, SMTP, FTP, NTP i TelNet ne pružaju zaštitu od nepoželjnih napada. U začecima interneta sigurnost na mreži nije bila velika briga, imajući u vidu da je internet u početku najviše korišćen u akademskim mrežama. Protokoli su koristili običnu autentifikaciju zasnovanu na lozinkama, bez zaštite poverljivosti i integriteta [1].

Protokol SSL (eng. Secure Socket Layer) pruža poboljšanje standardnog TCP/IP modela na taj način što obezbeđuje sigurnost na mreži. Sigurnost se postiže šifrovanjem poruka koje se razmenjuju. Protokol SSL je dodat između transportnog sloja i aplikativnog sloja TCP/IP modela (detaljnije u poglavlju 2.5. Protokol SSL).

Protokol SSL se pokazao kao dobro rešenje jer se lako dodaje u već korišćene protokole komunikacije i na taj način nadograđuje već postojeće funkcionalnosti. Osim mogućnosti šifrovanja, protokol SSL omogućava i autentifikaciju i provera identiteta.

2.1. Simetrički šifarski sistemi

U simetričnom šifarskom sistemu koristi se isti ključ za šifrovanje i za dešifrovanje, samo se za dešifrovanje koristi inverzna funkcija od funkcije šifrovanja. Algoritmi iz ove grupe se nazivaju algoritmi sa tajnim ključem, jer je tajnost ključa koji se koristi za šifrovanje i za dešifrovanje, najbitnija za bezbednost poruke u sistemu. Algoritmi koji koriste simetrični ključ za šifrovanje odlikuju se visokom efikasnošću, što se ogleda u kratkom vremenu šifrovanja poruka. Razlog efikasnosti uobičajenih algoritama za šifrovanje je upotreba kratkih ključeva. Iz tog razloga se ova vrsta algoritama koristi za šifrovanje/dešifrovanje poruka velike dužine. Primarni nedostatak simetričnih kriptografskih algoritama je to što ključ mora ostati tajni. Obe strane u komunikaciji (osoba A i osoba B) moraju posedovati jedinstveni simetrični ključ, zbog čega se javlja problem distribucije ključeva. Naime, korisnici koji žele da razmene poruku prethodno moraju da se dogovore o ključu. Na slici 2.1.1. prikazano je slanje poruke korišćenjem simetričnog šifarskog sistema za šifrovanje date poruke.



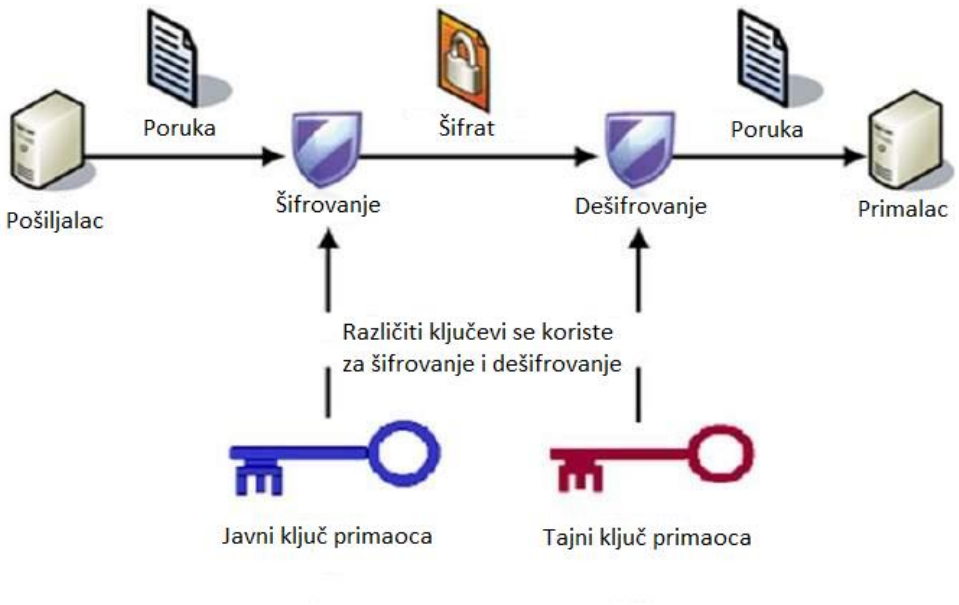
Slika 2.1.1. Slanje poruke korišćenjem simetričnog šifarskog sistema

Jedini pouzdan način je da se oba korisnika fizički sretnu i izvrše razmenu ključa. Međutim, često su korisnici fizički razdvojeni i ne mogu da dođu u neposredni kontakt, zato moraju da koriste neki zaštićen kanal da bi sigurno razmenili

ključeve. Jedno rešenje za problem razmene ključa je kriptografski protokol za razmenu ključeva. OpenSSL pruža Diffie-Hellman protokol za ovu svrhu, koji omogućava dogovor o korišćenju ključa bez njegovog prenošenja kroz mrežu. Medjutim, Diffie-Hellman ne garantuje identitet druge strane sa kojom se razmenjuje ključ. Potreban je dodatni mehanizam koji proverava autentičnost druge strane, kako ne bi došlo do razmene ključa sa napadačem. Najpoznatiji i najviše korišćeni simetrični kriptografski algoritmi su trostruki DES (eng. Data Encryption Standard), AES (eng. Advance Encrption Standrad), RC4. Sigurnost je vezana za dužinu ključa, duži ključevi su bolji. Ne bi trebalo da se koristi ključ koji je kraći od 80 bita. AES podržava samo dužinu od 128 bita i više, dok trostruki DES ima fiksnu dužinu ključa od 112 bita. Veća dužina je nepotrebna, dok se manja smatra slabom [1].

2.2. Asimetrični šifarski sistemi

Asimetrični šifarski sistem, ili sistem sa javnim ključem, ima rešenje za problem simetričnih šifarskih sistema. U asimetričnim šifarskim sistemima svaka strana ima dva ključa, jedan koji mora biti tajni i jedan koji je javni i može biti slobodno distribuiran. Ova dva ključa imaju posebnu matematičku vezu. Da bi osoba A poslala poruku osobi B, koristeći sistem sa javnim ključem, osoba A mora da poseduje javni ključ osobe B. Tada osoba A šifruje svoju poruku koristeći javni ključ osobe B, i tako je šalje. Tako šifrovanu poruku može uspešno da dešifruje samo neko ko poseduje tajni ključ osobe B, trebalo bi da je to samo osoba B.



Slika 2.2.1. Slanje poruke korišćenjem asimetričnog šifarskog sistema

Kriptografija sa javnim ključem može da se koristi za sporazum oko korišćenja ključa simetričnog šifarskog sistema, digitalni potpis i šifrovanje. Tri najčešće korišćena algoritma kriptografije se javnim ključem su Diffie-Hellman (DH), algoritam digitalnog potpisa (eng. Digital Signature Algorithm) i RSA. RSA je najpopularniji algoritam iz ove grupe i dobio je naziv po prezimenima pronalazača Ron Rivest, Adi Shamir i Leonard Adleman. Sistemom sa javnim ključem rešava se problem distribucije ključeva simetričnog šifarskog sistema, pod pretpostavkom da postoji način da javni ključ osobe B stvarno pripada osobi B. U praksi, javni ključevi se distribuiraju u sertifikatima, pri čemu su sertifikati proverni preko treće strane, koja je od poverenja. Treća strana, koja je od poverenja, je organizacija koja proverava entitete koji žele da dobiju svoj sertifikat, zatim izdaje traženi sertifikat i obezbeđuje proveru validnosti istih. Više o tome u poglavlju Infrastruktura sa javnim ključem.

Problem sistema sa javnim ključem je sporost šifrovanja dugačkih poruka, dok su simetrični algoritmi dosta brži pri šifrovanju i dešifrovanju poruka u mreži. Iz ovih razloga asimetrični kriptografski sistemi se najčesce koriste u implementaciji protokola za razmenu ključeva simetričnih šifarskih sistema. Diffie-Hellman protokol za razmenu ključeva je zasnovan na tehnologiji javnog ključa i može se koristiti za razmenu simetričnog ključa.

U RSA algoritmu, ključevi su veliki brojevi sa određenim osobinama. Njihova dužina bi trebala biti veća od 1024 bita da bi se obezbedila sigurnost. Ključevi od 512 bita u ovom slučaju bi bili slabi. Ako bi ključ bio duži od 2048 bita algoritam bi bio previše spor, a ne bi se postigla bitno veća sigurnost. Postojale su sumnje da je dužina ključa od 1024 bita previše slaba, ali u raspoloživoj literaturi ne postoji dokaz za to. Ipak, preporučuje se da 1024 bita bude minimum koji treba da se koristi za dužinu ključa u asimetričnim algoritmima [1].

2.3. Heš funkcije i kodovi za autentifikaciju

Kriptografske heš funkcije vrše transformaciju poruke proizvoljne dužine u nisku bitova fiksne dužine. Vrednost koja se dobije kao izlazna naziva se sažetak poruke (eng. *message digest*).

Prosleđivanje istih ulaznih vrednosti u heš funkciju uvek će rezultirati istom izlaznom vrednošću. Dobijeni rezultat ne odaje nikakve informacije iz podataka koji su prosleđeni na ulazu. Još jedna osobina heš funkcija jeste to da je izuzetno teško naći različite dve ulazne vrednosti koje proizvode isti sažetak. Ovakve funkcije se nazivaju još jednosmerne funkcije. Nadalje, ne bi trebalo da postoji mogućnost da se iz izlazne vrednosti rekonstruiše ulazni podatak. MD5 i SHA1 su najpopularnije jednosmerne heš funkcije koje se koriste. MD5 proizvodi sažetak poruke dužine 128 bita, dok je SHA1 sažetak dužine 160 bita. Preporučuje se korišćenje kriptografskih heš funkcija koje proizvode izlaznu vrednost 160 bita ili više [1].

Kriptografske heš funkcije se koriste u više slučajeva. Na primer, one se često koriste za čuvanje lozinki. Kad se korisnik loguje na sistem, proverava se njegova autentičnost tako što se pokretanjem heš funkcije, koja za ulaznu vrednost ima lozinku i još neke podatke, generiše heš vrednost koja se zatim poredi sa čuvanom vrednošću, gde čuvana vrednost predstavlja takođe heš vrednost lozinke korisnika. Na taj način server ne mora da čuva pravu lozinku, što pruža dodatnu sigurnost ako dođe do napada na bazu gde se čuvaju lozinke. Pored toga, heš funkcije se koriste za očuvanje integriteta poruka koje se šalju u mreži. U takvom scenariju, generiše se sažetak poruke pre slanja i šalje se zajedno sa porukom. Pri prijemu poruka se opet provlači kroz heš funkiciju i poredi sa sažetkom koji je stigao uz

poruku. Ako postoji razlika, tada se može sa sigurnošću tvrditi da je sadržaj poruke promenjen u toku slanja.

Pošto su heš hunkcije javno dostupne i pošto ne sadrže nikakav tajni ključ, u određenim situacijama može doći do zloupotrebe. Da bi se to sprečilo heš funkcije se takođe koriste u kombinaciji sa tajnim ključem i takve funkcije se zovu kodovi za autentifikaciju (eng. *Message Authentication Codes* - MACs). Ovakve funkcije se najčešće koriste u slučaju očuvanja integriteta podataka, bez obzira da li su podaci šifrovani ili ne. Najčešće korišćena MAC funkcija, koju takođe podržava OpenSSL, je HMAC. HMAC može da se koristi u kombinaciji sa bilo kojim heš algoritmom [1].

2.4. Digitalni potpis

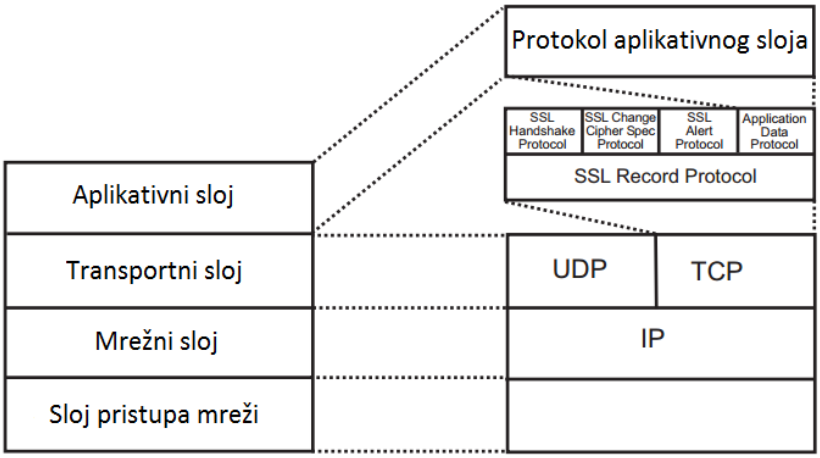
U većini slučajeva MAC-ovi nisu baš korisni zato što zahtevaju dogovor oko tajnog ključa. Bilo bi dobro da poruke mogu da se autentifikuju bez korišćenja tajnih podataka. Kriptografija sa javnim ključem rešava ovaj problem. Ako osoba A potpiše poruku svojim tajnim ključem, onda svako može upotrebiti njen javni ključ kako bi verifikovao da je osoba A zaista potpisala poruku. Algoritam RSA može da se koristi za digitalne potpise, a pored njega koristi se i algoritam za digitalno potpisivanje (eng. *Digital Signature Algorithm – DSA*), koji je podržan protokolom SSL i bibliotekom OpenSSL. Mana digitalnih potpisa jeste to što su spori [1]. Iz tog razloga digitalni potpis se vrši nad heširanom vrednošću poruke. Digitalni potpisi se koriste za potpisivanje digitalnih sertifikata. Entitet A potpisuje sertifikat entiteta B svojim privatnim ključem, tada B daje svoj potpisani sertifikat entitetu C sa kojom želi da uspostavi sigurnu komunikaciju. Da bi C proverio da li dati sertifikat zaista pripada B, obratiće se entitetu A kao strani kojoj veruje. Korišćenjem javnog ključa entiteta A, proverava se potpis datog sertifikata. Nakon provere entitet C može sigurno da komunicira sa B. Učestvovanjem treće strane u komunikaciji omogućava se provera identiteta korisnika u mreži.

2.5. Protokol SSL

Protokol SSL je sigurnosni protokol koji se koristi na internetu. Protokol HTTPS je protokol HTTP koji koristi usluge protokola TLS/SSL na nižem sloju. Protokol SSL

predstavlja sloj između transportnog i aplikativnog sloja i ima dve osnovne funkcionalnosti:

- Uspostavljanje sigurne i autentifikovane veze
- Korišćenje sigurne veze za prenos podataka od pošiljaoca ka primaocu. Fragmenti podataka koji se prenose u mreži zovu se SSL slogovi. Svaki SSL slog, pre slanja, je šifrovan i autentifikovan korišćenjem MAC funkcije. Kad poruka stigne primaocu, ona mora biti dešifrovana i mora biti provereno da sadržaj poruke nije promenjen, pre nego što poruka bude prosleđena aplikativnom sloju.



Slika 2.5.1 TCP/IP model sa protokolom SSL

2.5.1 Protokol SSL sloga

Protokol SSL sloga se koristi za enkapsulaciju podatka iz viših slojeva u fragmente (tj. slogove), zatim se fragmenti na odgovarajući način obrade i potom prosleđuju kroz mrežu [2]. Obrada fragmenta podrazumeva:

- Komprimovanje
- Kriptografsku zaštitu podataka
- Dodavanje SSL zaglavlja

Komprimovanje fragmenta je opciona i retko se koristi u praksi, ali ipak mogućnost komprimovanja pre šifrovanja je bitna zbog toga što podaci nakon šifrovanja ne mogu biti znatno komprimovani. Kriptografska zaštita podataka podrazumeva autentifikovanje poruke (u smislu očuvanja integriteta poruke) i šifrovanje poruke [2]. Postoji tri načina da se to realizuje:

1. Autentifikuje se poruka MAC funkcijom, sažetak poruke se doda na poruku i šifruju su zajedno, zatim se pošalje dobijeni šifrat (koji uključuje i MAC) primaocu. Ovaj pristup se zove autentifikuj-zatim-šifruj (eng. *authenticate-then-encrypt,* AtE) i koristi se u protokolu SSL/TLS.
2. Šifruje se poruka, zatim se dobijeni šifrat autentifikuje MAC funkcijom, zatim se šifrat šalje zajedno sa sažetkom. Ovaj pristup se zove šifruj-zatim-autentifikuj (eng. *encrypt-then-authenticate*, EtA) i koristi se u IPsec protokolu.
3. Poruka se autentifikuje i poruka se šifruje, zatim se sažetak dodaje na šifrat i šalje se. Ovaj pristup se zove šifruj-i-autentifikuj (eng. *encrypt-and-authenticate*, E&A) i koristi se u protokolu SSH.

Na kraju, dodaje se SSL zaglavlje koje sadrži podatak koji opisuje tipu sadržaja (eng. content-type), gde tip sadržaja predstavlja SSL protokole na višem nivou (*SSL Handshake Protocol, SSL Change Cipher Spec Protokol, SSL Alert Protocol, SSL Application Data Protocol*), zatim verziju SSL-a i dužinu poruke sa višeg sloja čiji fragment se šalje kroz mrežu.

2.5.2. Protokol SSL rukovanja

Za HTTPS komunikaciju, veb server treba biti konfigurisan da sluša na portu 443 i da koristi serverski SSL sertifikat. Serverski sertifikat sadrži informacije koje omogućavaju klijentu da potvrdi identitet servera, pre slanja poverljivih informacija. SSL rukovanje omogućava serveru i klijentu da se međusobno autentifikuju i da se dogovore oko upotrebe simetričnog ključa, kao i metodama kompresije. Uspostava HTTPS veze sastoji se iz sledećih koraka:

- Klijent šalje zahtev da pristupi veb strani
- Veb server u svom odogovoru šalje sertifikat koji u sebi sadrži javni ključ servera i sve potrebne informacije za dalju komunikaciju sa klijentom
- Klijent zatim šalje poruku šifrovanu serverskim javnim ključem
- Da bi server dokazao svoj identitet on mora koristiti svoj privatni ključ da bi dešifrovao poruku, klijent nakon toga mora da primi ispravnu dešifrovanu poruku kako bi komunikacija mogla da se nastavi.

U ovom slučaju, napadač uvek može da presretne serversku poruku i da se predstavi kao server koristeći svoj sertifikat. Tada napadač može da uspostavi vezu ka serveru i na taj način prisluškuje komunikaciju između servera i klijenta. Ako se radi samo o prisluškivanju komunikacije, onda govorimo o pasivnom napadu, ali ako napadač modifikuje poruke onda govorimo o aktivnom mrežnom napadu. U oba slučaju, ovakva vrsta napada se zove čovek-u-sredini (eng. man-in-the-middle).

Da bi se sprečila ovakva vrsta napada, klijent mora da validira serverski sertifikat. Rešenje ovog problem je uvođenje treće strane od poverenja koja bi čuvala sve validne sertifikate. Treća strana od poverenja se zove sertifikaciono telo. Sertifikaciono telo potpisuje javni ključ entiteta svojim privatnim ključem. Svojim potpisom sertifikaciono telo potvrđuje da je proveren identitet vlasnika sertifikata. Klijent može da proveri potpis sertifikacionog tela pod pretpostavkom da klijent poseduje javni ključ sertifikacionog tela.

Protokol SSL funkcioniše u praksi iako ima nekoliko problema, a to su:

- Sporost koja je prouzrokovana uvođenjem adekvatne sigurnosti
- Problem čuvanja privatnog ključa
- Loši serverski akreditivi
- Problem provere validnosti sertifikata.

3. Infrastruktura sa javnim ključem

Infrastrukturu sistema sa javnim ključevima (eng. *Public Key Infrastructure*) čine hardver, softver, polise i procedure koje su neophodne za upravljanje, generisanje, skladištenje i distribuciju kriptografskih ključeva i digitalnih sertifikata. PKI sistemi se baziraju na digitalnim identitetima, poznatim pod nazivom digitalni sertifikati koji igraju ulogu svojevrsnih "digitalnih pasoša" i koji povezuju ime vlasnika datog sertifikata sa njegovim javnim ključem u asimetričnom kriptografskom sistemu. Na taj način rešava se problem lažnog predstavljanja u komunikaciji i pasivnog ili aktivnog prisluškivanja.

3.1. Sertifikaciono telo

Sertifikaciono telo je najvažnija komponenta i osnova poverenja PKI sistema. Sertifikaciono telo (eng. Certificate Authority - CA) je organizacija koja izdaje digitalne sertifikate. Sertifikaciono telo ima veliku odgovornost da osigura validnost sertifikata, odnosno CA proverava da dati javni ključ pripada definisanom korisniku i svojim potpisom garantuje da je to istina. Postoje dva osnovna tipa sertifikacionog tela i to su privatni i javni. Privatni CA je odgovoran za izdavanje sertifikata u okviru neke organizacije, dok javni CA izdaje sertifikate za bilo koji javni entitet, što znači da javni CA mora biti javna strana od poverenja [1]. Dakle, osnove funkcije sertifikacionog tela su:

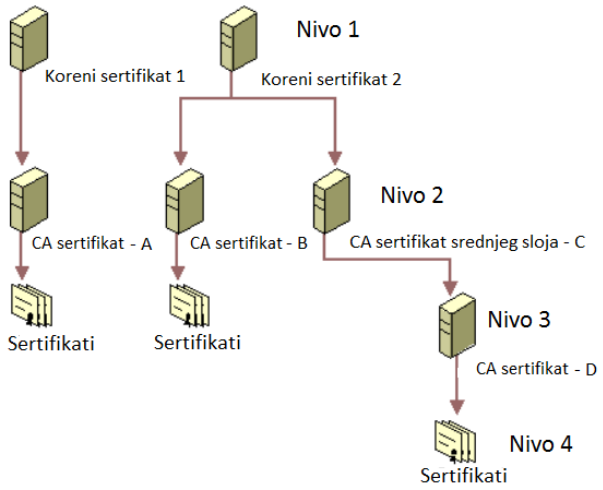
- Generisanje digitalnih sertifikata, tako što povezuje identifikacione podatke određenog korisnika sa njegovim javnim ključem asimetričnog kriptografskog sistema i sve to potvrđuje svojim digitalnim potpisom
- Upravlje rokom važnosti izdatih digitalnih sertifikata
- Povlačenje izdatih digitalnih sertifikata u slučajevima kada za to postoje uslovi, i u tom smislu, objavljivanje liste povučenih sertifikata (eng. Certificate Revocation List – CRL).

3.2. Registraciono telo

Registraciono telo (eng. Registration Authority – RA) proverava korisničke zahteve za izdavaje ili povlačenje digitalnih sertifikata i ako korisnik prođe proveru, registraciono telo prosleđuje zahtev sertifikacionom telu koje generše ili povlači sertifikat. U tom smislu registraciono telo može biti klijent-server aplikacija koja ima svoju lokalnu bazu i koja sa jedne strane komunicira sa krajnjim korisnicima, a sa druge strane sa sertifikacionim telom. [3]

3.3. Mreža povernja

Sertifikat koji je izdalo sertifikaciono telo može da se koristi za potpisivanje drugih sertifikata, koji dalje mogu da potpisuju nove sertifikata. Na taj način sertifikati mogu da budu ulančani. Na vrhu lanca nalazi se koreni (eng. root) sertifikat koji je samopotpisan (eng. *self-signed*). Ne postoji mogućnost da se verifikuje samopotpisani sertifikat zato što su izdavač i subjekat isti. Da bi se verifikovala autentičnost i validnost određenog sertifikata, svaki sertifikat u lancu mora biti takođe verifikovan od sertifikata koji se proverava i sve do korenog sertifikata. Ako bilo koji sertifikat u lancu nije validan, svaki sertifikat ispod njega takođe nije validan. Dakle, struktura mreže poverenja ima strukturu drveta. Listovi – krajnji sertifikati se mogu verifikovati praćenjem unazad, npr. na slici 3.3.1. – sertifikati na nivou 4 se validiraju počevši od CA sertifikata D koga je izdao sertifikat srednjeg sloja C, zatim se proverava sertifikat C korišćenjem korenog sertifikata 2. Ovako predstavljena hijerarhija sertifikata još se zove i mreža poverenja.



Slika 3.3.1. Mreža poverenja

3.4. Digitalni sertifikati

Digitalni sertifikati omogućavaju potvrdu identiteta učesnika u elektronskoj komunikaciji tako što povezuje javni ključ sa istaknutim imenom (eng. *distinguished name*). Istaknuto ime je ime osobe ili entiteta koji poseduje javni ključ za koji je vezan. Dakle, potvrđuje se da određeni javni ključ pripada određenom krajnjem entitetu. Na taj način se sprečava zloupotreba ključeva i mogućnost da se neko neovlašćeno predstavlja tuđim identitetom. Digitalni sertifikat je validan tokom određenog vremenskog perioda, a nakon što istekne potrebno je generisati novi i stari sertifikat proglasiti nevažećim. Sertifikat je potpisan privatnim ključem organizacije koja ga izdaje i sadrži dodatne informacije o entitetu kome pripada sertifikat i o organizaciji koja ga je izdala.

3.4.1. Struktura digitalnih sertifikata

Digitalni sertifikat u sebi sadrži sledeće delove:

- Verzija formata sertifikata – predstavlja oznaku strukture digitalnog sertifikata koja je definisana u standardu X.509. Moguće vrednosti su v1, v2, v3. Predložena vrednost je v3.
- Serijski broj sertifikata - predstavlja redni broj izdatog sertifikata. Način dodeljivanja serijskih brojeva mora biti jedinstven. Serijski broj sertifikata je vrednost koju dodeljuje sertifikaciono telo u trenutku kreiranja digitalnog sertifikata.
- Identifikator algoritma kojim se vrši digitalni potpis – identifikator algoritma digitalnog potpisa (eng. Signature Algorithm) je u stvari oznaka asimetričnog kriptografskog algoritma (RSA, DSA).
- Naziv sertfikacionog tela koje je izdalo sertifikat - struktura koja identifikuje Sertifikaciono telo (CA) koje je generisalo dati sertifikat i koja se sastoji iz sledećih elemenata: ime izdavača sertifikata (eng. commonName), odeljenje u organizaciji (eng. organizationalUnitName), organizacija (eng. organization), mesto (eng. localityName), elektronska adresa (eng. emailAddress), region ili republika u okviru države (eng. stateOrProvinceName), oznaka države (eng. countryName). U sklopu ove strukture se minimalno mora nalaziti naziv države.
- Rok važenja sertifikata - ovaj period je definisan sa dva datuma, a to su datum početka važenja sertifikata (eng. Valid From) i datum prestanka

važenja sertifikata (eng. Valid To). Predložena vrednost datuma početka važenja sertifikata je datum izdavanja sertifikata, a datum prestanka važenja sertifikata će biti izračunat na osnovu definisanog vremena validnosti sertifikata.

- Naziv vlasnika sertifikata - ovo polje identifikuje entitet asociran sa javnim ključem koji se nalazi u polju "javni ključ vlasnika sertifikata". Za kvalifikovani sertifikat, ovo polje mora imati vrednost prepoznatljivog (eng. distinguished) imena subjekta. U sklopu imena se mora nalaziti odgovarajući podskup atributa koji jedinstveno određuju subjekat.
- Javni ključ vlasnika sertifikata – informacija o javnom ključu vlasnika sadrži numeričku reprezentaciju javnog ključa i identifikator asimetričnog algoritma (RSA, DSA) sa kojim se dati ključ primenjuje.
- Specifični podaci koji se odnose na uslove korišćenja sertifikata – sadrži skup polja koja nose dodatne informacije. Neke od ovih informacija mogu biti namena javnog ključa koji vlasnik sertifikata poseduje, opis uslova pod kojim je sertifikata napravljen i za šta se može koristiti.
- Digitalni potpis sertifikata tajnim ključem sertifikacionog tela.

Najčešće korišćen format digitalnog sertifikata je definisan X.509 standardom, verzija 3.

3.4.2. Povlačenje digitalnih sertifikata

Korisnički sertifikati se mogu povući (opozvati) iz dva razloga:

- Došlo je do promena informacija o vlasniku sertifikata koje se nalaze u njegovom sertifikata
- Došlo je do gubitka asimetrinog privatnog ključa korisnika ili je iz bilo kog razloga došlo do kompromitovanja ključa datog korisnika.

U tom smislu, povlačenje sertifikata se odnosi na akciju koja podrazumeva proglašavanja nevažećim javni ključ korisnika, čime se automatski i njegov tajni ključ proglašava nevažećim i datom korisniku se tako onemogućava validno digitalno potpisivanje poruka. Od suštinske je važnosti da informacija o povučenosti datog sertifikata bude što je pre moguće javno objavljena i dostpuna svim učesnicima u sistemu. Rešenje je korišćenje liste povučenih sertifikata (eng.

Certificate Revocation List – CRL). Lista povučenih sertifikata sadrži sve sertifikate koji su povučeni od strane sertifikacionog tela. Distribucija CRL listi je veliki problem. Klijenti moraju da imaju tekuću verziju liste povučenih sertifikata da bi mogli da validiraju nečiji sertifikat. U idealnoj situaciji, klijent bi dobio ažurirane (eng. *up-to-date*) informacije o sertifikatima u trenutku kad CA dobije informaciju o povučenim sertifikatima. Ali, CRL su obično velike liste. Skidanje velike liste pre validiranja svakog sertifikata, može dovesti do velikog kašnjenja i velikog opterećenja servera kada veliki broj klijenata zatraži CRL listu. Zbog tog, CA se trudi da ažurira listu redovno, ali ne odmah nakon dobijene informacije o povučenim sertifikatima. U CRL je upisan i datum kada će ponovo biti ažurirana lista.

Online Certificate Status Protocol (OCSP) predstavlja alternativu za proveru validnosti sertifikata. Primarna svrha ovog protokola bila je rešavanje problema distribucije CRL listi. Protokol OCSP funkcioniše tako što se šalje zahtev ka servisu OCSP (eng. OCSP responder) koji u odgovoru vraća status sertifikata za koji je poslat upit. Status može da bude: dobar (eng. good), povučen (eng. revoked) ili nepoznat (eng. unknown). Status *dobar* ukazuje na pozitivan odgovor zahteva. Pozitivan odgovor znači da sertifikat sa datim serijskim brojem nije povučen, da je validan i da može da se koristi. Status *povučen* ukazuje na to da je traženi sertifikat povučen privremeno ili trajno. Status *nepoznat* ukazuje na to da servis OCSP ne prepoznaje da je dati sertifikate ikad zahtevan od strane CA. Status *povučen* znači da dati sertifikate ne treba da se koristi, dok *nepoznat* status ukazuje da na to da klijent treba da odluči da li želi da potraži informacije negde drugde.

4. Realizacija sertifikacionog tela

Prilikom implementacije veb servisa koji realizuje dve osnovne funkcionalnosti serifikacionog tela (izdavanje digitalnih sertifikata i provera validnosti istih) korišćen je programski jezik Java u kombinaciji sa okvirom za razvoj Spring i OpenSSL alatom komandne linije. Veb servis se startuje na Linux operativnom sistemu, distribucija CentOS 7, gde je instalirana OpenSSL verzija 1.0.1e-fips.

U datoj realizaciji sertifikaciono telo ujedno je i registraciono telo. U EJBCA projektu otvorenog koda (eng. *open source*) PKI arhitektura može imati registracino telo kao odvojen entitet. Odvajanjem funkcija koje vrši registraciono telo od sertifikacionog tela omogućava se registracija različitih korisnika, u tom slučaju je obično potrebno imati različita registraciona tela za različite potrebe. Pored toga, sertifikaciono telo se razrešava dužnosti provere korisnika koji šalju zahtev za izdavanje sertifikata. U datoj realizaciji sertifikaciono telo čuva statuse izdatih sertifikata u SQL bazi i na taj način čitanje i upisivanje statusa izdatih sertifikata ne zavisi od CRL listi, slično kao i u EJBCA implementaciji. [4]

4.1. Model-Pogled-Kontroler arhitektura

Model-Pogled-Kontroler (eng. *Model-View-Controller* – MVC) je često korišćen uzorak u implementaciji veb servisa. Glavni princip MVC uzorka ja da definiše arhitekturu sa jasnim odgovornostima komponenti. U MVC arhitekturi postoji tri sloja:

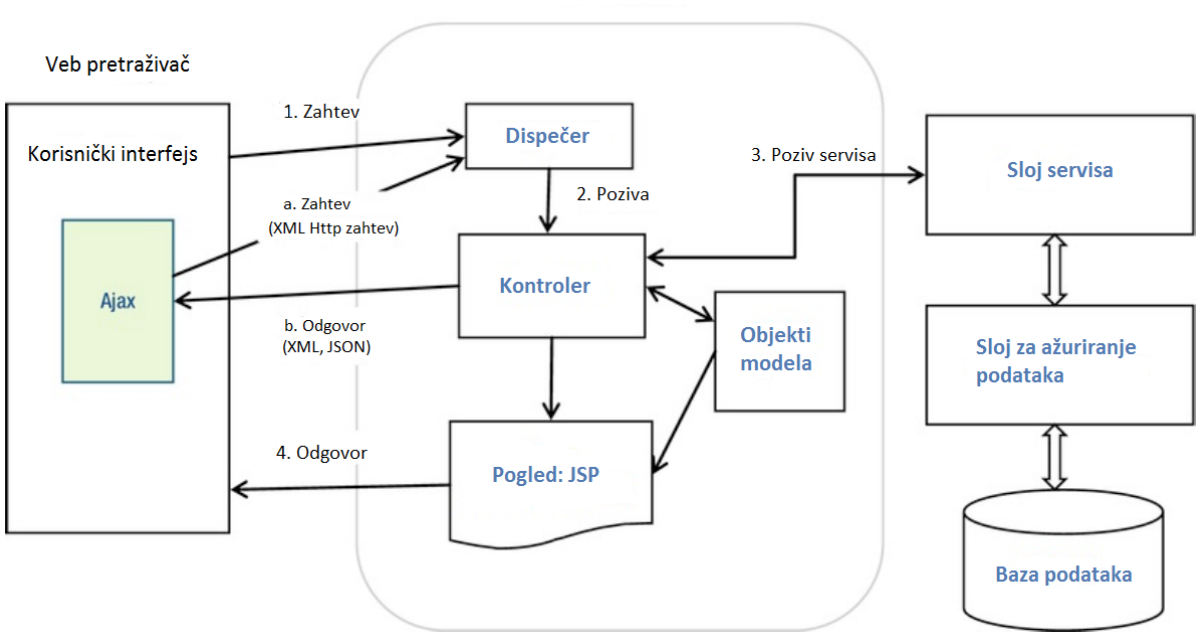
Model - predstavlja poslovne podatke, kao i stanje aplikacije u kontekstu korisnika. Na primer, informacije sa korisničkog profila,

Pogled - predstavlja podatke korisniku u dizajniranom formatu, pruža podršku interakcije korisnika i aplikacije,

Kontroler - upravlja zahtevima koji stižu od korisnika, komunicira sa servisnim slojem, ažurira model, i vodi korisnika na odgovarajući pogled, koji zavisi od rezultata izvršavanje posla.

Slika 4.1.1 prikazuje MVC arhitekturu veb aplikacije. Korisnički zahtev se izvršava na sledeći način:

1. Zahtev ja poslat serveru
2. Dispečer (eng. Dispatcher Servlet) obarađuje zahtev i u zavisnoti od informacija u HTTP zahtevu, prosleđuje ga odgovarajućem kontroleru
3. Kontroler komunicira sa servisnim slojem
4. Kontroler ažurira model, sa podacima koji su dobijeni kao rezultat izvšavanja korisničkog zahteva i vraća odgovarajući pogled korisniku.



Slika 4.1.1. MVC arhitektura [5]

Ako postoje Ajax funkcije u pogledu, tada se ne otvara cela stranica nego samo delovi stranice pogleda. Razlika pri obradi zahteva, koji se proseđuje kroz ajax, je u tome što tada HTTP odgovor nije vezan za pogled, već se uzimaju podaci iz HTTP odgovora i ažuriraće se postojeći pogled.

4.1.1. Spring MVC konfiguracija

Za podešavanje Spring MVC veb aplikacije potreba je inicijalna konfiguracija koja se podešava u web.xml datoteci. Sadržaj ove datoteke se, pored ostalog, odnosi na konfigurisanje *WebApplicationContext-a* i na konfigurisanje dispečera.

Prilikom implementacije sertifikacionog tela korišćena je sledeća konfiguracija:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app                        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" id="WebApp_ID" version="2.5">
       <display-name>CA</display-name>
       <context-param>
       <param-name>contextConfigLocation</param-name>
              <param-value>classpath:spring/application-config.xml</param-value>
       </context-param>
       <!-- Spring MVC filters -->
       <filter>
              <filter-name>CharacterEncodingFilter</filter-name>
              <filter-class>
                     org.springframework.web.filter.CharacterEncodingFilter
              </filter-class>
              <init-param>
                     <param-name>encoding</param-name>
                     <param-value>UTF-8</param-value>
              </init-param>
              <init-param>
                     <param-name>forceEncoding</param-name>
                     <param-value>true</param-value>
              </init-param>
       </filter>
       <listener>
              <listener-
class>org.springframework.web.context.ContextLoaderListener</listener-class>
       </listener>
       <servlet>
       <servlet-name>dispatcherServlet</servlet-name>
       <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-
class>
       <init-param>
              <param-name>contextConfigLocation</param-name>
              <param-value>/WEB-INF/mvc-config.xml</param-value>
       </init-param>
       <load-on-startup>1</load-on-startup>
       </servlet>
       <servlet-mapping>
       <servlet-name>dispatcherServlet</servlet-name>
       <url-pattern>/</url-pattern>
       </servlet-mapping>
</web-app>
```

U <context-param> tagu se *contextConfigLocation* šalje kao parametar i njime se definiše lokacija korene konfiguracione datoteke veb aplikacije. U ovom slučaju to je application-config.xml datoteka, koji kreira java zrna (eng. java beans) vezane za implementaciju servisnog sloja i za objekte za pristup podacima (eng. Data Access Objects - DAO) kao i za konfiguraciju povezivanja na bazu. Dok, druga

konfiguraciona datoteka definisana unutar <servlet> taga, konfiguriše dispečer servlet i sadrži <context:component-scan …/> tag koji skenira pakete u potrazi za java zrnima deklarisanim korišćenjem anotacija @Controller, @Service. Zatim konfiguriše java zrna vezana za *viewResolver* (način na koji se prikazuje pogled, u ovom slučaju to je *Tiles*), *ajaxResolver*, *multipartResolver*.

4.2. Struktura baze podataka

Realizacija servisa za izdavanje sertifikata zahteva definisanje strukture baze podataka. Model baze podataka sadrži sledeće tabele:

- CertSigningRequest
- RootCert
- Certificate
- OCSPCert
- User

Tabela *CertSigningRequest* sadrži informacije o zahtevima za izdavanje sertifikata. *DnsInfo* atribut prestavlja domensko ime mašine za koju se vezuje sertifikat i upisuje se u OpenSSL konfiguracionu datoteku u okvru ekstenzije *server_cert*.

Naziv atributa	Opis	Napomena
id	Primarni ključ	Auto increment, not null
userId	Id korisnika koji upload-uje csr	Strani ključ iz tabele User
csrFile	Lokacije na server gde se čuva zahtev za izdavanje sertifikata (csr)	Varchar(100)
dateCreated	Datum kad je upload-ovan csr	Date
dnsInfo	subjectAltInfo sadrži dns informacije	Varchar(300)

Tabela *RootCert* sadrži informacije o sertifikatima sertifikacionog tela (korenim sertifikatima i sertifikatima srednjeg sloja).

Naziv atributa	Opis	Napomena
Id	Primarni ključ	Auto increment, not null
rootKey	Lokacija tajnog ključa na serveru	Varchar(200)
rootCert	Lokacija ca sertifikata na serveru	Varchar(200)
rootName	Naziv ca sertifikata	Varchar(80)

Tabela *Certificate* sadrži informacije o sertifikatima koje je generisalo sertifikaciono telo na zahtev korisnika.

Naziv atributa	Opis	Napomena
Id	Primarni ključ	Auto increment, not null
userid	Id korisnika koji je generisao sertifikat	Strain ključ iz tabele User
certFile	Lokacija sertifikata na serveru	Varchar(200)
certFileName	Naziv sertifikata	Varchar(80)
dateCreated	Datum kada je generisan sertifikat	Date
signedWithCert	Id CA sertifikata iz RootCert tabele kojim je potpisan	Strani ključ

Tabela *OCSPCert* sadrži informacije o kriptografskim parovima protokola OCSP, kao i atribut *signedWithCert* koji predstavlja id CA sertifikat sa kojim je potpisan.

Naziv atributa	Opis	Napomena
Id	Primarni ključ	Auto increment, not null
certFile	Lokacija ocsp sertifikata na serveru	Varchar(200)
KeyFile	Lokacija ocsp ključa na server	Varchar(200)
signedWithCert	Id CA sertifikata iz RootCert tabele kojim je potpisan	Strani ključ

Tabela *User* sadrži informacije o korisnicima sistema.

Naziv atributa	Opis	Napomena
id	Primarni ključ	Auto increment, not null
username	Korisničko ime	Varchar(20)
password	Lozinka korisnika	Varchar(20)

4.3. Upotreba biblioteke OpenSSL

OpenSSL je sofverska biblioteka koja pruža alat za implementaciju sigurnosnog transportnog sloja (eng. Transport Layer Security - TLS) i sloja sigurnih soketa (eng. Secure Sockets Layer - SSL). Takođe se koristi i kao kriptografska biblioteka. OpenSSL može da se koristi i kao alat iz komandne linije, za izvršavanje osnovnih funkcija. U daljem tekstu biće opisane osnovne komande koje postoje u OpenSSL-u, a koje su korišćene pri realizacije sertifikacionog tela.

4.3.1. Priprema okruženja i openssl konfiguracione datoteke

Potrebno je napraviti lokaciju na serveru gde će se čuvati sve datoteke vezani za aplikaciju koja implementira sertifikaciono telo. Za glavni direktorijum korišćena je lokacija */root/ca/*. Glavni direktorijum u sebi sadrži poddirektorijume *private* i *certs*. U poddirektorijumu *private/* nalaze se tajni ključevi sertifikacionog tela. Poddirektorijum *certs/* sadrži sertifikate koje je izdalo sertifikaciono telo. Takođe potrebno je na istoj */root/ca/* lokaciji čuvati dve datoteke *serial* i *index.txt*. Datoteka serial služi da bi se pratio poslednji serijski broj sertifikata koji je izdat. Od velike je važnosti da se ne izdaju dva sertifikata sa istim serijskim broje od strane istog sertifikacionog tela. Datoteka index.txt predstavlja bazu podataka u koju se upisuju informacije o izdatom sertifikatu.

OpenSSL alat komandne linije pruža mnogo opcija za svaku komandu. Upravljanje opcijama je dosta pojednostavljeno uz pomoć konfiguracione datoteke. OpenSSL komande, koje koriste konfiguracionu datoteku su: ca, req i x509. Konfiguraciona datoteka je organizovana po sekcijama. Svaka sekcija sadrži skup ključeva i njihovih vrednosti. Ključevi su razdvojeni od njihovih vrednosti znakom jednakosti. Komentari su linije koje počinju znakom #.

Primer konfiguracione datoteke:

```
default_ca = ca
[ ca ]
dir =             /root/ca
new_certs_dir =   $dir/certs
serial =          $dir/serial
database =        $dir/index.txt

default_crl_days = 7
default_days =    365
default_md =      md5

policy = ca_policy
x509_extensions = certificate_extensions

#root key and root cert
certificate =     $dir/cacert.pem
private_key =     $dir/private/cakey.pem

[ ca_policy ]
commonName =            supplied
stateOrProvinceName =   supplied
countryName =           supplied
emailAddress =          supplied
organizationName =      supplied
organizationalUnitName = optional

[ certificate_extensions ]
basicConstraints = CA:false
```

4.3.1.1. Sertifikatska proširenja

Kao što je već rečeno, najšire prihvaćen format sertifikata je X.509 format verzija 3. Najznačajnija funkcijonalnost u verziji 3 jeste podrška za proširenja (eng. extensions). Proširenja dozvoljavaju da sertifikat sadrži dodatna polja kao što su *basicConstraints* ili *keyUsage* polja. Različite sertifikatska proširenja se definišu u konfiguracionoj datoteci u vidu sekcije i koriste se za generisanje različitih tipova sertifikata. Postoji 14 proširenja definisanih u X.509v3 standradu, ali samo četiri od njih su dobro podržana i često korišćena [6].

Proširenje *basicConstraint* može da sadrži *CA* i *pathlen* vrednosti. CA vrednost može biti *true* ili *false* čime se ukazuje na to da li sertifikat može biti korišćen kao CA sertifikat. Ako CA vrednost ne postoji u *basicConstraint* proširenju, openssl proverava *keyUsage* proširenje. Ako *keyUsage* sadrži vrednost *keyCertSign,* onda sertifikat može bit korišćen kao CA sertifikat, u suprotnom ne može. Vrednost *pathLen* je broj koji označava maksimum broj sertifikata u lancu koji može

postojati ispod datog sertifikata. Ako je vrednost manja od broja sertifikata u lancu koji su već validirani, onda izdavanje datog sertifikata mora biti odbijeno.

Proširenje *keyUsage* definiše na koji način sertifikat može biti upotrebljen. Ako proširenje postoji u sertifikatu, trebalo bi da bude obeleženo sa *critical* parametrom. Ako postoji vrednost *critical,* onda će ovo proširenje uvek biti korišćeno za definisanje upotrebe sertifikata. Ako *keyUsage* proširenje ne postoji ili je obeleženo sa *noncritical*, smatraće se da su sve vrednosti u sertifikatu postavljene. U tabeli je prikazano koje vrednosti treba da budu postavljene za koju upotrebu sertifikata.

Upotreba sertifikata	Postavljena vrednost
CA sertifikat	keyCertSign, cRLSign
Potpisivanje sertifikata	keyCertSign
Potpisivanje objekata	digitalSignature
S-Mime šifrovanje	keyEncipherment
S-Mime potpisivanje	digitalSignature
SSL klijent	digitalSignature
SSL server	keyEncipherment

Proširenje *extKeyUsage* predstavlja niz identifikatora objekata (eng. Object Identifiers - OIDs) koji dalje definišu koja upotreba setifikata je dopustiva i može, a i ne mora, biti definisana kako *critical*. Svrha ključa može biti definisana identifikatorom objekta od strane bilo koje organizacije kojoj je to potrebno. Identifikatori objekata se koriste da bi mogla da se identifikuje svrha ključa, zbog toga OID-i moraju da budu dodeljeni u okviru ITU-T X.660[1] ili Internet Assigned Numbers Authority (IANA)[2] organizacije. Ako sertifikat sadrži obe ekstenzije *keyUsage* i *extKeyUsage,* obe ekstenzije moraju biti procesirane nezavisno i sertifikat mora biti korišćen za svrhu koja je konzistentna u obe ekstenzije, u suprotnom sertifikat ne sme biti korišćen ni u jednu svrhu [7].

[1] ITU-T X.660 | ISO/IEC 9834-1 definiše strukturu drveta koja podržava internacionalne indentifikatore objekata (OIDs). http://www.itu.int/itu-t/recommendations/rec.aspx?rec=X.660

[2] IANA je organizacija koja ima odgovornost za globalno upravljanje DNS root, IP adresama, i drugim internet protokol resursima. http://www.iana.org/

Proširenje *cRLDistributionPoints* se sastoji od informacija o mestu distribuiranja CRL liste, razloga za distribuciju i naziva organizacije koja izdaje CRL listu.

4.3.2. Kreiranje korenih ključeva i sertifikata

Da bi moglo da izdaje potpisane sertifikate sertifikaciono telo mora da ima svoje vlastite sertifikate kojima bi potpisivalo druge sertifikate. Sertifikati sertifikacionog tela ili koreni sertifikati (koji se nalaze na vrhu lanca poverenja) su samopotpisani sertifikati.

Komanda **genrsa** OpenSSL-a se koristi za generisanje novog tajnog ključa RSA algoritma. Dužina ključa za RSA je obično 1024 bita, ali se za ključeve sertifikacionog tela preporučuje dužina od 2048 bita. Korišćenjem ove komande RSA tajni ključ nije šifrovan, ali postoji mogućnost da se šifruje korišćenjem DES ili 3DES algoritma.
Komanda **rsa** koristi se za manipulisanje RSA ključevima. Ova komanda može da dodaje, modifikuje i ukljanja šifarsku zaštitu nad ključem. Takođe može da se koristi za dobijanje javnog ključa iz tajnog ključa.
Komanda **req** koristi se za generisanje samopotpisanih sertifikata, ali i za generisanje zahteva za izdavanje sertifikata.
Dodatna konfiguracija potrebna za generisanje samopotpisanih sertifikata:

```
[ req ]
default_bits = 2048
default_md = md5
prompt = no
distinguished_name = root_ca_distinguished_name

# ekstenzija koja se koristi uz –x509 opciju
x509_extensions = v3_ca

[ root_ca_distinguished_name ]
commonName =            Matf
stateOrProvinceName =   Belgrade
countryName =           RS
emailAddress =          ca@matf.org
organizationName =      Root Certification Authority

[ v3_ca ]
basicConstraints =      CA:true
```

Primer 1.
openssl genrsa -out rsaprivatekey.pem -passout pass:matf -des3 2048

Navedena komanda generiše 2048-bitni ključ, koji je šifrovan trostrukim DES algoritmom sa lozinkom “matf”. Tajni ključ će biti smešten u rsaprivatekey.pem datoteku.

Primer 2.

```
# openssl rsa -in rsaprivatekey.pem -passin pass:matf –pubout –out rsa
publickey.pem
```

Navedena komada čita tajni ključ iz rsaprivatekey.pem, dešifruje ga korišćenjem lozinke “matf” i ispisuje odgovarajući javni ključ u rsapublickey.pem datoteku.

Primer 3.

```
# openssl req -x509 -newkey rsa:2048 -out cacert.pem -outform PEM
```

Navedena komanda generiše tajni ključ dužine 2048 bita i na osnovu njega generiše samopotpisani setifikat koji će biti sačuvan u cacert.pem datoteku.

4.3.3. Kreiranje ključeva i sertifikata srednjeg sloja

Koreni sertifikat potpisuje sertifikat srednjeg sloja (eng. *intermediate certificate*) i na taj način se formira mreža poverenja. Primarna svrha korišćenja sertifikata srednjeg sloja je sigurnost. Koreni sertifikat može da se čuva van mreže i može da se koristi koliko god je puta neophodno za potpisivanje sertifikata srednjeg sloja. Ako je sertifikat srednjeg sloja kompromitovan, biće povučen od strane korenog sertifikata i kreiran novi kriptografski par. Sertifikat srednjeg sloja se koristi kao sertifikat sertifikacionog tela, odnosno koristi se za potpisivanje korisničkih/serverskih sertifikata. Proces kreiranja sertifikata srednjeg sloja je sličan procesu potpisivanja serverskih sertifikata koji je opisan u sledećem poglavlju, jedina razlika je u korišćenju proširenja.

4.3.4. Generisanje serverskih sertifikata

Za generisanje serverskih sertifikata potrebno je prvo generisati zahtev za izdavanje sertifikata (eng. Certificate Signing Request – CSR). Za kreiranje zahteva za izdavanje sertifikata koristi se **req** komanda OpenSSL-a.

```
# openssl req -newkey rsa:1024 -keyout testkey.pem -keyform PEM –out
testreq.pem
```

```
Generating a 1024 bit RSA private key
.........++++++
```

```
.........++++++
writing new private key to 'testkey.pem'
Enter PEM pass phrase:
Verifying password - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be
incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or
a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:RS
State or Province Name (full name) [Some-State]:Belgrade
Locality Name (eg, city) []:BG
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Test
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname)
[]:www.ca.org
Email Address []:ca@test.org

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:ca
```

Pri pravljenju CSR datoteke komanda će tražiti da se unesu informacije o entitetu koji želi da dobije potpisani sertifikat.

Rezultat ove komande su dve datoteke: testreq.pem i testkey.pem, gde testreq.pem sadrži zahtev za izdavanje sertifikata, dok testkey.pem sadrži tajni ključ koji odgovara javnom ključu iz testreq.pem datoteke. Odnosno, ovom komandom napravljen je par ključeva (tajni i javni ključ). Prilikom pokretanja komande potrebno je uneti pristupnu frazu (eng. *passphrase*) koja se koristi za šifrovanje tajnog ključa.

Sledeća komanda će dati prikaz zahteva za izdavanje sertifikata:

openssl req -in testreq.pem -text –noout

```
Certificate Request:
    Data:
        Version: 0 (0x0)
        Subject: C=RS, ST=Beograd, L=BG, O=A,
CN=A/emailAddress=a@nesto.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (1024 bit)
```

```
                Modulus:
                    00:d7:7b:da:ac:7f:ed:bc:50:f5:f7:7f:00:7f:40:
                    ed:5d:23:c1:1b:8e:e6:a1:81:1c:c6:b0:8d:be:5a:
                    f9:52:0d:31:23:d1:e1:9d:83:67:b7:5c:16:d1:a6:
                    0b:d4:0a:a9:06:69:9c:72:8c:0e:03:8f:d3:d2:ae:
                    3c:cc:6d:63:75:c1:23:64:b5:ad:7d:30:8f:f7:3f:
                    19:8a:9f:78:77:ee:6b:23:79:ed:f7:b2:da:df:63:
                    34:10:1d:0e:46:b5:e3:c9:73:58:ab:e5:93:3f:f3:
                    cb:0a:a1:3e:c5:0b:4f:23:13:89:1c:d1:84:b0:7e:
                    2f:3a:e6:3b:17:1b:85:49:47
                Exponent: 65537 (0x10001)
        Attributes:
            challengePassword        :unable to print attribute
            unstructuredName         :unable to print attribute
    Signature Algorithm: sha256WithRSAEncryption
         d1:6b:79:f5:d5:d1:b3:4b:24:98:9d:59:fc:fc:76:3e:d1:2b:
         aa:fb:4c:df:55:f8:2e:ae:1b:6c:82:36:d9:7e:c2:11:50:55:
         b0:a2:38:31:3a:8d:61:ba:d2:77:4a:18:b7:6d:63:b8:4a:04:
         73:48:f0:5e:3e:6a:3a:4c:93:ae:af:5a:0e:a9:b5:c9:22:f8:
         04:34:0a:c2:64:14:d3:37:56:d8:c3:1d:84:69:cb:01:45:a3:
         c5:a4:b9:ee:97:a9:22:b1:f3:8e:e4:5c:f8:0e:19:26:2c:15:
         3c:cd:c6:38:52:50:ec:71:b9:26:2e:7d:5b:a0:39:d6:26:57:
         05:69
```

Zatim, na osnovu zahteva za izdavanje sertifikata može se upotrebom sertifikata sertifikacionog tela izdati novi serverski sertifikat. Novi sertifikat biće potpisan tajnim ključem CA sertifikata.

openssl ca –passin pass:matf –extensions server_cert –notext –md sha256 –in testreq.pem –out testcert.pem

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 4102 (0x1006)
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=RS, ST=Serbia, L=Belgrade, O=PSTech, OU=DevTest,
CN=Ivan Mihailovic RootCA
        Validity
            Not Before: Aug 30 08:50:21 2016 GMT
            Not After : Sep  9 08:50:21 2017 GMT
        Subject: C=RS, ST=Beograd, O=A, CN=A/emailAddress=a@nesto.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (1024 bit)
                Modulus:
                    00:d7:7b:da:ac:7f:ed:bc:50:f5:f7:7f:00:7f:40:
                    ed:5d:23:c1:1b:8e:e6:a1:81:1c:c6:b0:8d:be:5a:
                    f9:52:0d:31:23:d1:e1:9d:83:67:b7:5c:16:d1:a6:
                    0b:d4:0a:a9:06:69:9c:72:8c:0e:03:8f:d3:d2:ae:
```

```
                    3c:cc:6d:63:75:c1:23:64:b5:ad:7d:30:8f:f7:3f:
                    19:8a:9f:78:77:ee:6b:23:79:ed:f7:b2:da:df:63:
                    34:10:1d:0e:46:b5:e3:c9:73:58:ab:e5:93:3f:f3:
                    cb:0a:a1:3e:c5:0b:4f:23:13:89:1c:d1:84:b0:7e:
                    2f:3a:e6:3b:17:1b:85:49:47
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Cert Type:
                SSL Server
            Netscape Comment:
                OpenSSL Generated Server Certificate
            X509v3 Subject Key Identifier:

C6:5D:29:8A:4E:C4:42:DD:2B:27:84:A3:C1:F5:43:56:14:07:E6:AF
            X509v3 Authority Key Identifier:

keyid:38:3D:C6:C8:BF:A4:85:8F:C5:A8:2F:86:B4:15:0F:CF:EB:C5:7C:E2

DirName:/C=RS/ST=Serbia/L=Belgrade/O=PSTech/OU=DevTest/CN=Ivan
Mihailovic RootCA
                serial:D7:C5:94:11:30:C4:56:22

            X509v3 Key Usage: critical
                Digital Signature, Key Encipherment
            X509v3 Extended Key Usage:
                TLS Web Server Authentication
            Authority Information Access:
                OCSP - URI:http://certisign.lab.pst:8080

            X509v3 Subject Alternative Name:
                DNS:ta-orion4-mtg.lab.pst,  DNS:ta-orion4-admin.lab.pst,
DNS:ta-orion4-ha.lab.pst, DNS:ta-orion4-vm.lab.pst
    Signature Algorithm: sha256WithRSAEncryption
         67:b7:5f:e4:25:1f:b7:b4:0c:c8:35:fd:23:74:82:8b:d7:a1:
         46:e3:61:bc:cc:69:aa:1c:c1:7c:a1:02:dd:6a:ae:cc:3c:2b:
         97:6c:f5:ab:d2:b2:9b:c2:26:75:c7:89:20:ca:ff:e3:7f:86:
         f5:26:87:85:40:c6:f3:fd:ed:27:2f:1d:4e:1c:5d:1f:83:24:
         d1:53:b1:38:27:b3:13:09:50:43:23:9f:0e:10:dd:30:e4:3f:
         23:a4:a0:b3:53:e1:e2:b1:52:d1:ba:6f:ab:06:a2:c3:d9:bd:
         37:8c:63:31:ff:8c:23:24:84:e8:7f:1f:16:ac:2f:d6:69:5f:
         04:5a:a9:11:5c:3a:e9:10:b5:ff:29:b3:e8:c8:bc:83:72:61:
         c9:9d:b9:af:67:85:f3:41:a5:af:c9:84:82:50:32:0e:82:28:
         91:46:eb:21:c5:c4:a0:f9:b1:59:88:a6:a1:84:32:b7:d8:c0:
         62:8f:a5:03:96:93:a2:6e:d5:8e:f6:e5:62:c8:aa:66:d9:48:
         8a:4a:52:28:b1:a9:5f:7f:dd:b2:90:2a:e0:23:5c:77:72:4f:
         5d:ba:86:9a:11:a2:f7:b3:0a:3c:c6:4b:e0:87:e3:94:ea:3b:
         77:83:92:16:20:73:61:a0:08:41:fc:ee:a3:3f:02:d9:a8:a9:
         cd:65:44:66:2f:99:44:f4:66:06:3f:44:f6:aa:8d:98:f9:aa:
         96:24:28:8c:47:31:2f:3f:e1:8d:eb:5b:94:2f:30:b1:a8:e7:
         4f:97:20:99:26:30:d2:f3:1a:33:a4:00:9c:5b:3e:ba:e1:75:
```

```
d5:ba:7b:3f:8e:2c:02:d2:56:36:7a:a4:60:cb:8a:f4:3f:7f:
16:4d:05:5e:02:15:b9:4e:2b:94:ad:ba:10:64:2d:4a:50:a3:
d2:15:0f:51:03:be:f4:21:c2:59:73:7a:24:bf:16:8a:6f:a4:
69:ed:f1:ad:aa:ba:f5:21:1f:94:bd:6d:29:ab:dd:0a:88:d3:
5e:df:ee:37:70:1a:fc:d0:60:05:df:c9:97:10:c8:e4:b3:ee:
a8:e6:1a:c0:42:e3:00:ce:43:ec:83:54:0c:9e:f1:28:ce:a4:
02:7e:40:f0:7b:fb:e3:1c:3c:ba:93:6e:32:6a:11:ad:30:aa:
89:ce:90:eb:b4:cc:a6:a9:43:32:b8:80:65:aa:77:0d:4e:aa:
d7:5b:0c:1e:0b:db:f1:08:08:b5:30:71:3f:0c:d5:db:b4:f9:
8f:47:d2:42:36:69:17:93:25:34:21:d1:2e:db:93:45:51:75:
55:4f:37:98:26:32:b2:9a:71:c5:0d:bb:e4:7f:d0:ac:61:c4:
28:ae:8b:2c:fd:2c:b8:58
```

Ovako napravljen sertifikat u sebi sadrži ekstenziju *Subject Alternative Name*. Svrha ove ekstenzije jeste da sertifikat sadrži informacije koju su jedinstveno vezane za mašinu, tj. server koji će koristiti izdati sertifikat. Informacija koja je jedinstveno vezana za mašinu je njegovo domensko ime (eng. *domain name*) ili FQDN (eng. *Fully Qualified Domain Name*). Korišćenjem ove ekstenzije može da se proveri sam sertifikat, pošto provera lanca poverenja sama po sebi nije dovoljna. U x.509 v3 koristi se subjectAltName ekstenzija i polje dnsName. Domensko ime se čita iz zahteva za izdavanje sertifikata, a zatim dodaje kao proširenje u sertifikat. U realizaciji sertifikacionog tela dnsInfo se čita iz CSR datoteke, a zatim zajedno sa CSR datotekom čuva u bazi. Zatim, pre nego što se pokrene OpenSSL komanda za generisanje sertifikata ovaj podatak se ažurira u konfiguracionoj datoteci OpenSSL-a u sekciji *server_cert*, koja se koristi kao proširenje.

4.3.5. Povlačenje sertifikata

Za povlačenje sertifikata koristi se komanda **ca –revoke**. Ova komanda tražiće da se unese pristupna fraza (eng. *passphrase*) radi zaštite privatnog ključa CA sertifikat.

```
# openssl ca -revoke testcert.pem
```

Output:

```
Using configuration from openssl.cnf
Enter pass phrase for /root/ca/private/rootcakey.pem:
Revoking Certificate 1006.
Data Base Updated
```

Sertifikat koji je povučen ostaće nepromenjen. Jedina izmena dogodiće se u bazi sertifikacionog tela da bi se ažurirala informacija o statusu sertifikata. Status u tom slučaju postaje „R“, što je oznaka da je sertifikat povučen.

4.3.6. Online Certificate Status Protocol (OCSP)

Protokol OCSP omogućava prijem i obradu zahteva u cilju dobijanja odgovora o statusu određenog sertifikata. Servis OCSP osluškuje da li ima upita prosleđenih njemu i ako primi upit vratiće odgovor koji sadrži status datog sertifikata.

Potrebno je dodati proširenje *authorityInfoAccess* u konfiguracionu datoteku. Ovo proširenje koristi se za pristup informacijama sertifikacionog tela i ima sintaksu accessOID;lokacija. AccesOID može biti bilo koji OID koji ima smisla, na primer OCSP i caIssuer.

```
[ server_cert ]
# ... snipped ...
authorityInfoAccess = OCSP;URI:http://ocsp.example.com
```

Zatim treba napraviti par ključeva OCSP protokola. Ovaj par kriptografskih ključeva koristiće se za potpisivanje odgovora koji salje servis OCSP. Sertifikat OCSP mora biti potpisan istim privatnim ključem kojim je potpisan sertifikat čiji status se proverava.

Servis OCSP se pokreće **ocsp** komandom OpenSSL alata. Informacije o statusima izdatih sertifikata se čitaju iz index datoteke. Odgovor OCSP je potpisan sa OCSP kriptografskim parom korišćenjem –rkey i –rsigner opcija.

```
# openssl ocsp -port 127.0.0.1:2560 -text -sha256
      -index intermediate/index.txt
      -CA intermediate/certs/cacert.pem
      -rkey intermediate/private/ocspkey.pem
      -rsigner intermediate/certs/ocspcert.pem
      -nrequest 1
```

Upit se formira isto korišćenjem **ocsp** komande, gde se –cert opcijom prosleđuje sertifikat čiji status se želi proveriti.

```
# openssl ocsp -CAfile intermediate/certs/ca-chain.cert.pem
      -url http://127.0.0.1:2560 -resp_text
      -issuer intermediate/certs/intermediate.cert.pem
      -cert intermediate/certs/test.example.com.cert.pem
```

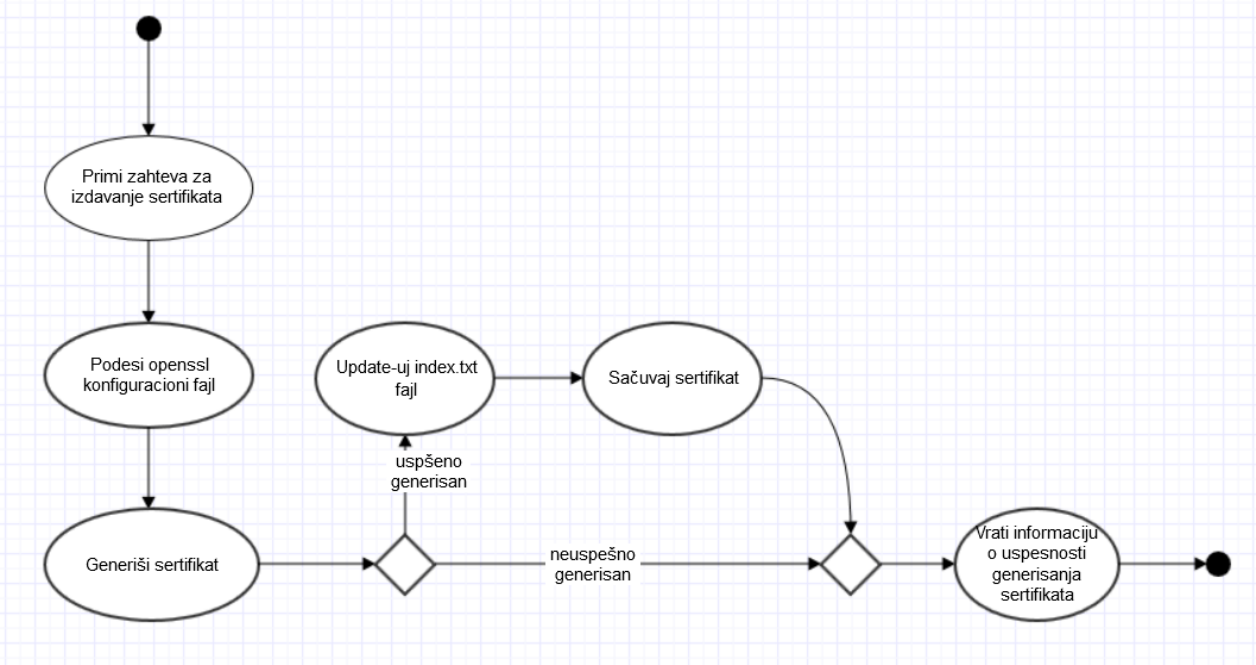
Odgovor u sebi sadrži:

- Da li je uspešno primljen odgovor protokola OCSP (eng. OCSP Response Status)
- Identitet OCSP servera (eng. Responder Id)
- Status sertfikata (eng. Cert Status)

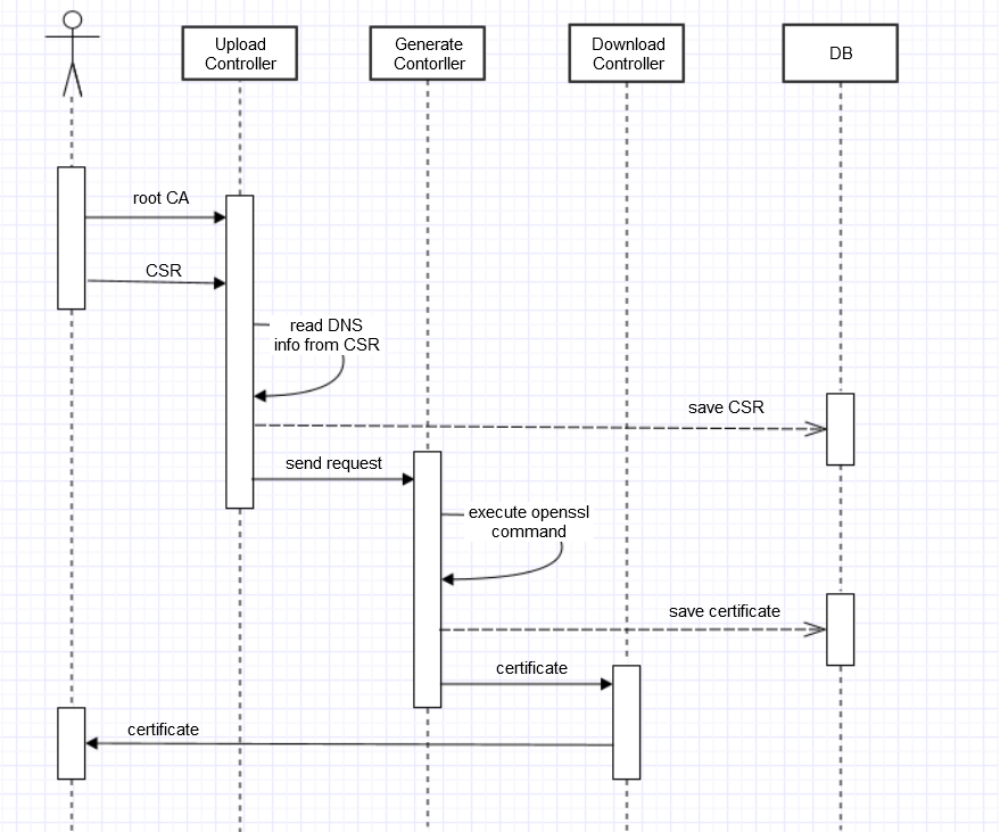
```
OCSP Response Data:
    OCSP Response Status: successful (0x0)
    Response Type: Basic OCSP Response
    Version: 1 (0x0)
    Responder Id: ... CN = ocsp.example.com
    Produced At: Apr 11 12:59:51 2015 GMT
    Responses:
    Certificate ID:
      Hash Algorithm: sha1
      Issuer Name Hash: E35979B6D0A973EBE8AEDED75D8C27D67D2A0334
      Issuer Key Hash: 69E8EC547F252360E5B6E77261F1D4B921D445E9
      Serial Number: 1003
    Cert Status: good
    This Update: Apr 11 12:59:51 2015 GMT
```

4.4. Realizacija izdavanja digitalnih sertifikata

Izdavanje digitalnih sertifikata, odnosno potpisivanje digitalnih sertifikata, realizuje se korišćenjem OpenSSL komandi, kao što je objašnjeno u pogavlju Upotreba openssl-a. Na slici 4.4.1. prikazan je dijagram aktivnosti izdavanja digitalnih sertifikata. Prvi korak jeste slanje zahteva za izdavanje sertifikata, koji treba da sadrži CSR datoteku, zatim informaciju o tome sa kojim sertifikatom sertifikacionog tela će biti potpisan i *dnsInfo* koji se čita iz same CSR datoteke. CSR datoteka zajedno sa tajnim ključem je generisan na serveru na kom će posle potpisani sertifikat biti korišćen. U slučaju kada je OpenSSL komanda uspešno izvršena, kreiraće se nova .pem datoteka i index.txt datoteka će biti dopunjen informacijama o novonastalom sertifikatu koji će potom biti sačuvan u bazi u tabeli *Certificate*. U slučaju da je OpenSSL komanda neuspešno izvršena, što se može dogoditi kad u index.txt datoteci postoji sertifikat za taj veb server, a nije povučen, korisnik će dobiti poruku o grešci. Ako je sertifikat za dati server povučen, moći će da se generiše novi.



Slika 4.4.1. Dijagram aktivnosti izdavanja digitalnih sertifikata



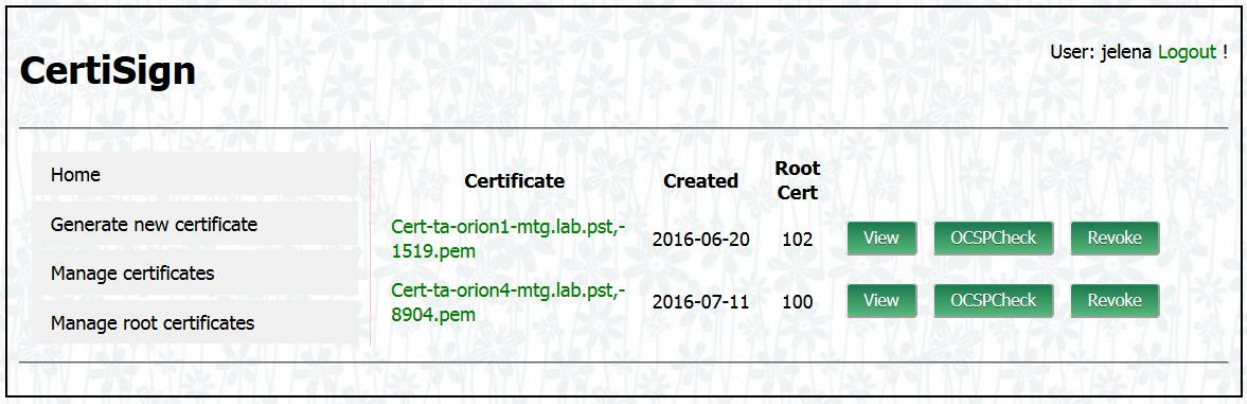
Slika 4.4.2. Dijagram sekvenci izdavanja digitalnih sertifikata

Implementacija; Metod za generisanje sertifikata poziva OpenSSL komandu, i ukoliko je uspešno generisan vraća sertifikat ako nije vraća *null* vrednost. Metoda "createCertificate" je deo poslovne logike aplikacije, a poziva se iz kontrolera "generateCert". Kontroler proverava da li je sertifikat uspešno generisan, i u slučaju kad jeste, poziva servis koji ima zadatak da sačuva sertifikat u bazi, u suprotnom kontroler šalje poruku o grešci u model. U oba slučaja informacija o uspešnosti generisanja sertifikata biće prikazana na pogledu "generate".

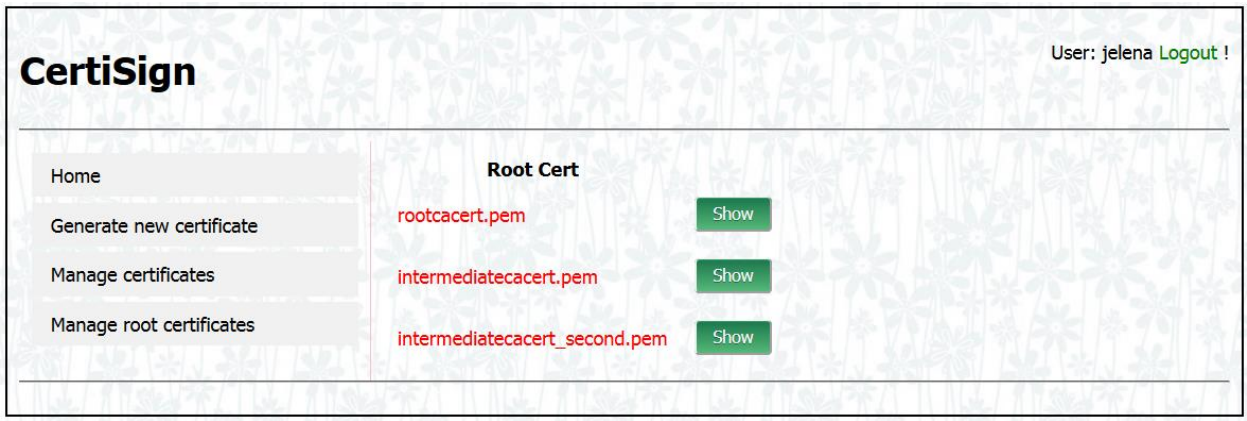
```
public synchronized Certificate createCertificate(CertSigningRequest csr, RootCert
rootCA, String pass) throws SQLException {
	Certificate cert = null;
	LoggerOpenSSL.logInfo("========== Create Certificate method==========");

	int rootCertId = rootCA.getId();
	String rootCertPath = rootCA.getRootCertFilePath();
	Logger.logInfo("Root cert: " + rootCertPath);

	String location = rootCertPath
			.substring(0,rootCertPath.lastIndexOf("certs/"));
	Logger.logInfo("Location of a root cert: "+ location);
	String certLocation = location +"certs/";


	String configFile = location + "openssl.cnf";
	Logger.logInfo("Config " + configFile);

	// set dns info in openssl.cnf file
	String dns = csr.getDnsInfo();
	String dnsInfoSplit [] = null;
	setDnsInfoInOpenSSLConf(dns, configFile);
	if(dns.contains("DNS:")) {
		dnsInfoSplit = dns.split("DNS:");
	}
	else {
		Logger.logError("DNS info malformed!");
		return null;
	}
	// set ca key and ca cert in cnf
	setCACertAndKeyInOpenSSLConf(rootCertPath, rootCA.getRootKeyFilePath(),
							configFile);
	// create filename for new cert
	String fileName = "Cert-" +dnsInfoSplit[1] + "-"+ UtilMethods.getRandomInt(4)
+ ".pem";

	String newCertLocation = certLocation + fileName;
	String command = "openssl ca -batch -passin pass:" + pass + " -config "
			+ configFile + " -extensions server_cert "
			+ "-notext -md sha256 -in " + csr.getCsrFilePath() + " -out "
			+ newCertLocation;
	Logger.logInfo(command);
	String outputOpensslCa = commandExecutioner.exec(command);
```

```
Logger.logInfo(outputOpensslCa);

// check if cert successfully created
String [] fileCheck = {"/bin/bash",
"/home/admin/CA/scripts/checkIfFileExistAndNotEmpty.sh", newCertLocation};

String out = commandExecutioner.execProcessBuilder(fileCheck);
Logger.logInfo("File check: " + out);

if(out.contains("notempty")) {
	Logger.logInfo("New cert has been created: "+ newCertLocation);
	cert = new Certificate(newCertLocation, fileName, rootCertId);
}
else if(out.contains("empty")) {
	if(outputOpensslCa.contains("failed to update database")) {
		Logger.logInfo("Failed to update database");
	}
	Logger.logInfo("File is empty and will be deleted.");
	if(!newCertLocation.equals("/")) {
		Logger.logInfo("File : " + newCertLocation);
		command = "rm -rf " +newCertLocation;
		out = commandExecutioner.exec(command);
		if(out.equals(""))
			Logger.logInfo("Empty cert is deleted.");
	}
	//return error
	return null;
}
else {
	//return error
	return null;
}

LoggerOpenSSL.logInfo("========== create cert method ends =======");
return cert;

}
```

Pored potpisivanja sertifikata, realizovano je i upravljanje sertifikatima i skidanje sertifikata sertifikacionog tela. Upravljanje sertifikatima je deo aplikacije gde se izlistavaju sertifikati koje je generisao trenutno logovan korisnik. Korisnik ima mogućnost da prikaže sertifikat, da ga skine, da proveri njegov status i da ga povuče. Na slici 4.4.3 prikazan je deo aplikacija za upravljenje sertifikatima.



Slika 4.4.3. Stranica za upravljanje izdatim sertifikatima



Slika 4.4.4. Stranica za upravljanje sertifikatima sertifikacionog tela

4.5. Realizacija protokola OCSP za proveru validnosti digitalnih sertifikata

Protokol OCSP za proveru validnosti sertifikata se realizuje u vidu startovanja servisa OCSP (eng. *OCSP responder*) koji funkcioniše po principu zahtev/odgovor. Servis OCSP čita vrednosti iz index.txt datoteke koji u sebi sadrži informacije o svim generisanim sertifikatima.

Index datoteka sadrži sledeće informacije:

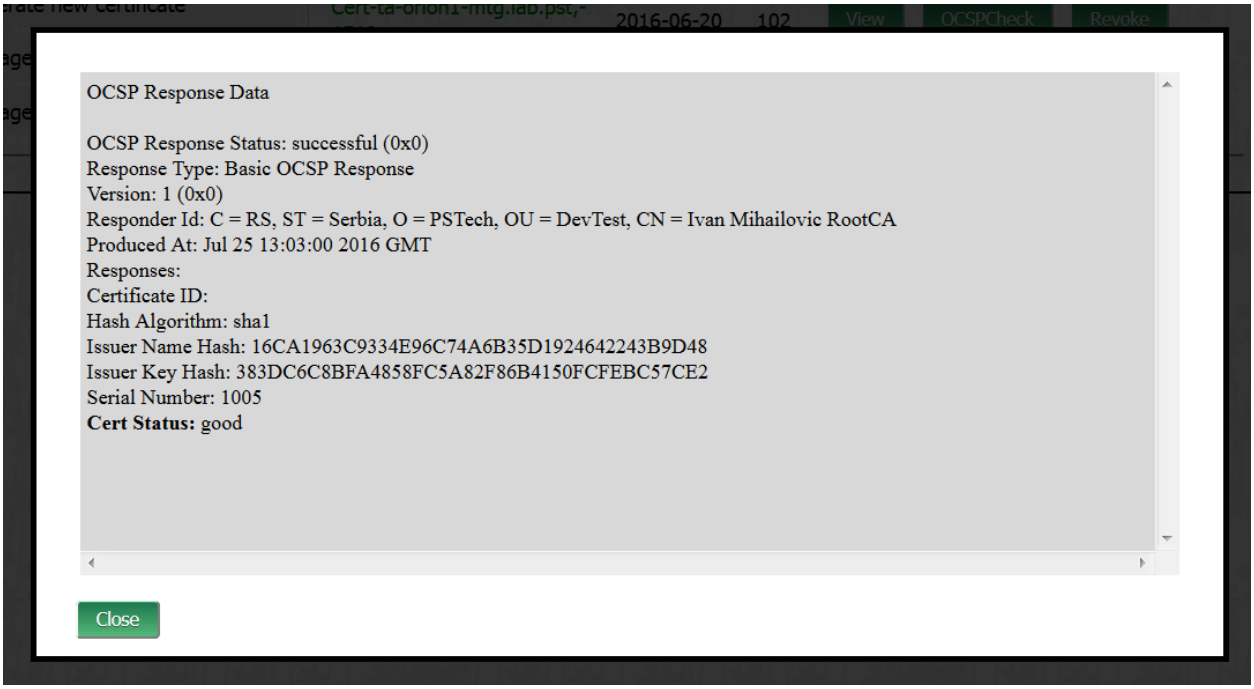
```
R    170630115832Z    160620120929Z    101B    unknown  /C=US/CN=ta-orion1-admin.lab.pst

V    170630120602Z                     101C    unknown  /C=US/CN=ta-orion4-admin.lab.pst

R    170630120947Z    160623094733Z    101D    unknown  /C=US/CN=ta-orion1-admin.lab.pst
```

Oznaka “R” znači da je sertifikat povučen, dok oznaka “V” znači da je sertifikat validan. Kolona četiri sadrži serijske brojeve sertifikata.

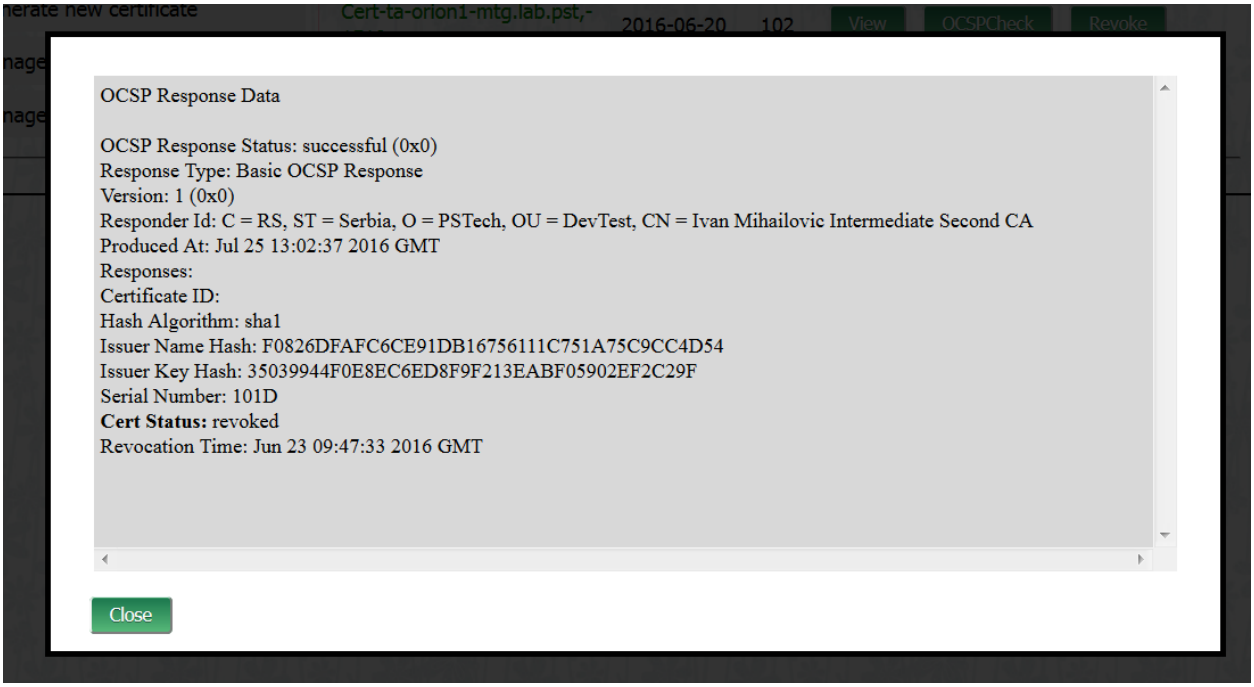
Implementacija; Pokretanje servisa OCSP i slanje zahteva OCSP:

```
public synchronized String ocspChecker(final RootCert ca, final OCSPCert ocsp, final
String index, Certificate cert) {

		Runnable runnable = new Runnable() {

			@Override
			public void run() {
				// TODO Auto-generated method stub
				Logger.logInfo("Starting ocsp responder....");
				String command ="openssl ocsp"
						+ " -port 127.0.0.1:2560 -text -sha256"
						+ " -index " + index
						+ " -CA " + ca.getRootCertFilePath()
						+ " -rkey " + ocsp.getOcspKey()
						+ " -rsigner " + ocsp.getOcspCert()
						+ " -nrequest 1";
				Logger.logInfo(command);
				String out = commandExe.exec(command);
				Logger.logInfo("Out: "+out);
			}
		};
		Thread thread = new Thread(runnable);
		thread.start();
		//this.runOCSPResponder(ca, ocsp, index);

		String response = sendOCSPQuery(ca, cert);

		// parse response
		return parseOCSPResponse(response);

	}

	private String sendOCSPQuery(RootCert ca, Certificate cert) {
		Logger.logInfo("Send query to ocsp responder...");
		String command = "openssl ocsp"
				+ " -url http://127.0.0.1:2560 -resp_text"
				+ " -issuer " + ca.getRootCertFilePath()
				+ " -cert " + cert.getCertFilePath();

		Logger.logInfo(command);
		String out = commandExe.exec(command);
		Logger.logInfo("Request Out: "+out);
		return out;
	}
```

Na slikama 4.5.1 i 4.5.2. prikazano je kako izgleda odgovor servisa OCSP kad je sertifikat validan i kad je povučen. U slučaju kada je sertifikat povučen potrebno je generisati novi.



Slika 4.5.1. Odgovor servisa OCSP u slučaju kada je sertifikat validan



Slika 4.5.2. Odgovor servisa OCSP u slučaju kada je sertifikat povučen

5. Primena digitalnih sertifikata

Upotreba kriptografije i digitalnih sertifikata u digitalnom svetu sve više raste. Veliki broj preduzeća koristi internet kao poslovni alat, na primer e-trgovina, e-poslovanje, kao i bankarsko poslovanje. Zahtevi zaštite komunikacije i podataka rastu uporedo sa upotrebom interneta i razvojem tehnologije, ali i razvojem sofisticiranijih napada na informacione sisteme. Aktuelan napad na sistem je onemogućavanje usluga (eng. Denial-of-service attack - DDoS). DDoS napadom napadač pokušava da sistem učini nedostupinim za korisnike koji ga regularno koriste. Ovo se postiže slanjem prekomernih zahteva ka targetiranom sistemu u cilju preopterećenja sistema, čime se sprečava da pojedini ili svi zahtevi regulranih koristika budu ispunjeni.

5.1. E-trgovina

Internet nastavlja da pravi dramatične izmene u načinu na koji potrošači kupuju proizvode tako što menja i širi distribucione kanale što omogućava da proizvodi budu virtuelno dostupni bilo kad i bio gde [8]. Dalji napredak u korišćenu interneta kao poslovnog alata znatno će uticati na interakciju preduzeća i potrošača. Osiguranje identiteta i poverenja je šema koju je razvila organizacija Comodo i ona predstavlja način kako zadobiti potrebno poverenje od korisnika, ali da poverenje bude zasnovano na jakim osnovama i sigurnosnim tehnikama i alatima. Osiguranje identiteta i poverenja treba da obezbedi:

1. Potrošači treba da imaju preventivne alate kako bi bili sigurni da mogu da veruju svom online trgovcu (eng. verification engine)
2. Trgovcima su potrebni alati kojim bi pomogli potrošačima da verifikuju sertifikatske akreditive
3. Trgovcima je potrebno da svaki put ubede potrošače da njihovi osetljivi podatci neće biti ukradeni
4. Sve prethodno navedene tačke dovode do povećanja sveukupnog poverenja. [9]

Comodo alati, koji se koriste za navedene situacije, su visoko pouzdani sertifikati SSL-a za šifrovanje i autentifikaciju preduzeća, osiguranje identiteta i sertifikati za verifikaciju sadržaja.

Osiguranje identiteta (eng. *identity assurance)* je servis koji u realnom vremenu pruža informacije o identitetima veb servisa koji su vezani za sertifikate SSL-a i predstavlja deo sertfikacionog tela.

Sertifikat za verifikaciju sadržaja (eng. *Content Verification Certificates* - CVC) vezuje veb sadržaj za digitalni sertifikat i može biti verifikovan od strane kupca. Sertifikat za verifikaciju sadržaja omogućava potrošačima da npr. autentifikuju logo plaćanja (Visa, Mastercard..). CVC sertifikati se izdaju od strane Comodo organizacije koja proverava da trgovac ima odobrenje za prihvatanje online uplata; nakon provere kupac može da dobije svoje CVC sertifikat.

5.2. E-poslovanje

Inicijalna upotreba interneta kao poslovnog alata je bila pretežno za potrebe reklamiranja u formi veb sajtova koji sadrže javne informacije. Današnje kompanije na svojim veb sajtovima imaju podršku za proizvode, podršku za potrošače, maloprodaju i kanale za isporuku elektroskih proizvoda i usluga. Elektronsko poslovanje uključuje IT infrastrukturu, mrežu i aplikacije za kontinualno poboljšanje vrednosti. Kada firma odluči da implementira elektronsko poslovanje, mora da zna da to podrazumeva detaljanu analizu poslovnih procesa. Ova strategija ima za cilj pravljenje okruženja koje je orijentisano ka kupcu, koje omogućava brzo snabdevanje i pronalaženje najboljih načina za stvaranje prihoda.

Autentifikacija- provera akreditiva da bi se utrvrdio identitet korisnika ili entiteta.

Autorizacija - process koji se odlučuje kojim podacima korisnik može da pristupi i šta može da radi sa datim podacima.

Dakle, pitanje bezbednosti poslovnog sistema je definisano pitanjima ko može da pristupi sistemu i ako pristupi sistemu šta može na njemu da uradi.

Autorizacija može da se uspostavi dozvolama, tj. pravima pristupa (čitanje,pisanje, ažuriranje), tako da pravila pristupa budu pridružena korisniku. To znači da od korisničkog profila zavisi koje privilegije će dati korisnik imati na sistemu.

Za autentifikaciju koriste se digitalni sertifikati. Digitalni sertifikati, napravljeni od strane sertifikacionog tela, omogućavaju uspostaljanje sigurnih elektroniskih veza, zbog čega je PKI veoma važan sistem za uspostavljanje e-poslovanja. Izbor PKI organizacije, proizvoda ili pristupa, treba da bude u skladu sa celim sigurnosnim planom jedne organizacije. Plan treba da uzme u obzir rezultate analize rizika i da uključi formalno napisanu sigurnosnu polisu. Svrha PKI sistema je da omogući poslovanju sigurno i pouzdano ponašanje na mreži. Zahtevi posla će odrediti ključne faktore za performase operacija. Generalni poslovni zahtevi, koji treba da budu razmatrani kad se vrši izbor PKI sistema, su:

- Potreba da sertifikaciono telo bude dostupno na internetu, intranetu ili ekstranetu
- Potreba da se prikaže u sertifikatu mera do koje CA verifikuje identitet korisnika sertifikata
- Učestalost dodavanja sertifikata
- Učestalost povlačenja sertifikata
- Učestalost privremenog povlačenja sertifikata (što je u praksi retko, ali ako postoji onda je značajno)
- Kako drugi PKI korisnici nadgledaju uticaj na server i telekomunikacione veze u zavisnosti od toga koliko često se proveravaju sertifikati
- Legalna odgovornost vezana za sertifikate
- Cena integrisanja PKI tehnologija u poslovne aplikacije.

5.3. Internet stvari (Internet of Things – IoT)

Internet stvari povezuje pametne uređaje, uključujući automobile, robotizovanu proizvodnju, pametnu medicinsku opremu, pametne mreže i kontrolne sisteme u industriji. Međutim, rast umrežavaja različitih uređaja dovodi i do povećana sigurnosnih rizika. Internet stvari (*Internet of Things* - IoT) predstavlja vrlo kompleksan sistem, koji zahteva sigurnosna rešenja na krajnjim uređajima mreže (eng. *end-to-end)*. Ozbiljni rizici uključuju fizičku štetu, dugačak zastoj, oštećenje opreme kao što su cevi, visoke peći i postrojenja za proizvodnju energije. IoT sigurnosno rešenje može biti pokriveno sa četiri temelja: zaštita komunikacije, zaštita uređaja, upravljanje uređajima i razumevanje sistema [10].

Zaštita komunikacije

Zaštita komunikacije zahteva šifrovanje i autentifikaciju uređaja. Vrlo je uobičajena zabrinutost oko troškova kriptografskih operacija. Kritografija zasnovana na eliptičkim krivama omogućuva dosta brže operacije šifrovanja. Kriptografija eliptičkih kriva predstavlja alternativu za RSA algoritam sa javnim ključem [10].

Zaštita uređaja

Zaštita uređaja zahteva potpisivanje koda, da bi se obezbedilo da je ceo kôd autorizovan za pokretanje. Takođe, potrebna je i zaštita u toku izvršavanja koda (eng. run-time), da bi se obezbedila zaštita od zlonamernih napada koji žele da promene kôd nakon pokretanja. Kriptografsko potpisivanje kôda obezbeđuje da kôd nije promenjen i da može biti pokrenut na uređaju. Ipak, uređaj mora biti zaštićen i posle pokretanja kôda. Zaštita samog uređaja (eng. host-based protection) je od koristi u ovakvom scenariju, a to podrazumeva korišćenje antivirusa, zaštitnih zidova (eng. firewalls) i slično.

Upravljanje uređajima

Na žalost, ranjivost će ipak u jednom trenutku biti otkrivena na vrednim uređajima i moraće da se preuzmu mere za popravku. Problemi u kodu biće otkriveni i biće potrebno ažuriranje softvera. Preduzeće ima koristi ako može daljinski da upravlja uređajima, tj. u slučaju problema radnik ne mora fizički da obilazi mašine.

Razumevanje sistema

Ipak, bez obzira koliko se osigura sistem i koliko god se dobro upravlja njime, neki rizici mogu pobediti sve mere koje su preuzete prilikom obezbeđivanja sistema. Iz tog razloga, jako je bitno da postoji IoT bezbednosne analitike koje pomažu u najboljem razumevanju mreže, što pomaže da se obeleže anomalije koje mogu biti opasne.

6. Zaključak

Infrastruktura sa javnim ključem ima veliki značaj pri čuvanju poverljivih informacija, kao i obezbeđivanje svih potrebnih alata pri zaštiti elektronskog poslovanja. Samim tim upotreba treće strane od poverenja je od krucijalne važnosti za uspešnost funkcionisanja elektronskog poslovanja.

Biblioteka OpenSSL implementira kriptografske algoritme koji se koriste u obezbeđivanju sigurnosti na mreži. Biblioteka OpenSSL ne zahteva znanje o samom procesu generisanja ključeva i ostalim funkcijama i time dodatno olakšava upotrebu istih. Bitna činjenica jeste da je potrebno sveobuhvatno poznavanje sistema kako bi se obezbedila njegova sigurnost. Takođe, potrebno je kombinovanjem različitih algoritama za zaštitu napraviti najefikasnije i najpouzdanije rešenje.

Uvođenje protokola SSL i TLS u standardne protokole komunikacije ima za cenu usporenje sistema, ali ta cena se treba platiti u smislu da je korišćenje protokola SSL i TLS neophodno da bi podaci, serveri i korisnici bili zaštićeni na mreži. Postoje načini da se ubrza sistem određenim tehnikama. Tehnike koje će biti korišćene za poboljšanje efikasnosti sistema zavise od samih poslovnih procesa i potreba organizacije koje uvode sigurnose alate. Javni PKI sistemi, kao što su VeriSign, Comodo, Symantec, pružaju veliki broj alata za zaštitu sistema na mreži svim javnim entitetima. Cilj ovih organizacija je da očuvaju svoj kredibilitet i poverenje klijenata i obezbede najbolje moguće alate, kao i da budu upoznati sa svim mogućim napadima koji prete da ugroze funkcionisanje elektronske komunikacije.

Pri razvoju projekta, koji realizuje veb servis za obavljanje operacija sertifikacionog tela, prvi korak je bio modelovanje baze podataka, čija struktura podržava zahteve same aplikacije. Zatim se vrši analiza aktivnosti pri procesu izdavanja sertifikata na osnovu koje je kasnije sam proces implementiran. Implementacija sertifikacionog tela se radi tako da se mogu dadavati nove funkcionalnosti npr. izdavanje korisničkih sertifikata kao i uvođenje privilegija korisnicima ovog veb servisa. Ovako implementiran CA podržava operaciju izdavanja serverskih digitalnih sertifikata, što znači da dobijeni sertifikati mogu da

se koriste na veb serverima preko kojih se ostvaruje sigurna komunikacija sa klijentima. Da bi generisani sertifikat bio verifikovan, na klijentskoj mašini mora da bude instaliran sertifikat (ili lanac sertifikata) sertifikacionog tela. Sertifikati sertifikacionog tela su dostupni na implementiranom veb servisu i mogu se lako preuzeti. Za implementaciju provere validnosti sertifikata korišćen je protokol OCSP. Servis OCSP pruža informacije o statusu sertifikata u realnom vremenu. Realizacija sertifikacionog tela, prezentovana u ovom radu, može da se koristi za potrebe jedne organizacije ili kompanije, kao i u svrhe testiranja.

7. Literatura

[1] P. Chandra, M. Messier and J. Viega, Network Security with OpenSSL, O'Reilly, 2002.

[2] R. Oppliger, SSL and TLS theory an practice, Artech House, 2009.

[3] V. Team, „Trust Network Certificate Policies,“ VeriSign, 2002.

[4] PrimeKey, „EJBCA documentation,“ 2017. [Na mreži]. Available: https://www.ejbca.org/docs/index.html.

[5] C. Ho / R. Harrop, Pro Spring 3 (Expert's Voice in Spring), Apress, 2012.

[6] I. Ristić, OpenSSL Cookbook, Feisty Duck, 2014.

[7] R. L. W. P. R. Housley, „Internet Engineering Task Force (IETF),“ [Na mreži]. Available: https://www.ietf.org/rfc/rfc3280.txt.

[8] „The Evolution of e-Business Security Requirements,“ 2001. [Na mreži]. Available: https://www.msctrustgate.com/pdf/eBusinessSecurity.pdf.

[9] „Identity & Trust Assurance,“ 2012. [Na mreži]. Available: https://www.comodo.com/pdf/ita_whitepaper.pdf.

[10] „An Internet of Things Reference Architecture,“ 2016. [Na mreži]. Available: https://www.symantec.com/content/dam/symantec/docs/white-papers/iot-security-reference-architecture-en.pdf.

Spisak skraćenica

PKI	*Public Key Infrastructure*
CA	*Certificate Authority*
SSL	*Secure Socket Layer*
TLS	*Transport Layer Security*
HTTP	*Hypertext Transfer Protocol*
SMTP	*Simple Mail Transfer Protocol*
FTP	*File Transfer Protocol*
NTP	*Network Time Protocol*
DES	*Data Encryption Standard*
AES	*Advanced Encryption Standard*
DSA	*Digital Signature Algorithm*
RA	*Registration Authority*
RSA	*Rivest-Shamir-Adleman*
CRL	*Certificate Revocation List*
OCSP	*Online Certificate Status Protocol*
MVC	*Model-View-Controller*
CSR	*Certificate Signing Request*
DDoS	*Denial-of-service attack*
CVC	*Content Verification Certificates*
IoT	*Internet of Things*
OIDs	*Object Identifiers*
FQDN	*Fully Qualified Domain Name*
MACs	*Message Authentication Codes*

Spisak slika