



Elektrotehnički fakultet
BEOGRAD

DIPLOMSKI RAD

**Razvoj konfigurabilnog sistema za generisanje i verifikaciju
aplikacija za rad sa X.509 sertifikatima**

Kandidat:
Bogdana Veselinović 375/12

Profesor:
doc. dr Žarko Stanisavljević

Beograd
Septembar 2016.

Sadržaj

1.	UVOD	3
2.	X.509 SERTIFIKATI.....	6
2.1.	ISTORIJA X.509 STANDARDA.....	6
2.2.	STRUKTURA X.509 STANDARDA	6
2.2.1.	X.509 sertifikati verzije 1 i 2.....	6
2.2.2.	X.509 sertifikati verzije 3.....	7
2.3.	INFRASTRUKTURA JAVNIH KLJUČEVA (PKI)	11
3.	IMPLEMENTACIJA I DIZAJN.....	12
3.1.	KORIŠĆENE TEHNOLOGIJE.....	12
3.2.	OPIS SISTEMA.....	12
3.3.	OPŠTA ARHITEKTURA SISTEMA	13
3.4.	DIZAJN GRAFIČKOG KORISNIČKOG INTERFEJSA	24
3.5.	UOČENI PROBLEMI	29
4.	PRIMER KORIŠĆENJA SISTEMA	31
4.1.	PROFESOR.....	31
4.2.	STUDENT	32
4.3.	KORISNIK.....	33
4.4.	VERIFIKACIJA GENERISANIH SERTIFIKATA	36
5.	ZAKLJUČAK.....	37
	LITERATURA	38

1. UVOD

Potreba za očuvanjem poverljivosti informacija postoji još od samog začetka civilizacije. Uporedo s razvojem različitih načina komunikacije, razvijale su se i metode kojima se postizala zaštita razmenjenih poruka. Najprimitivnijim oblikom kriptografije može se smatrati i najobičnije pisanje sadržaja poruke pomoću olovke i papira, obzirom da većina ljudi u to vreme nije znala da čita. [1] Jedini cilj najranijih vidova kriptografije bio je pretvaranje poruke u nerazumljivi skup znakova za sve, osim za primaoca, kako bi ona bila zaštićena za vreme svog prenosa. Vremenom je, pored osnovne poverljivosti, između ostalog nastala potreba i za osiguravanjem da je poruka stigla do svog primaoca neizmenjena, kao i da je poslata od navedenog pošiljaoca. Pisani potpisi su od svog nastanka pa do danas jedno od osnovnih sredstava za identifikaciju koje se teško falsifikuje. Potpisi na slikama ili drugim umetničkim delima označavaju ime umetnika. U srednjem veku pošiljalac bi sipao vruć vosak preko spoja kojeg formiraju ivice papira preklopljenog pisma i zatim utisnuo jedinstveni otisak pomoću prstena. Nije bilo moguće pročitati sadržaj pisma, a da se pritom ne ošteti pečat. Tako je primalac poruke bio siguran u identitet pošiljaoca, da je poruka stigla neizmenjena do svog cilja, kao i da nije pročitana od strane neke druge osobe za vreme transporta.

Sa pojavom Interneta počinje i elektronska razmena podataka. U moderno doba, razmena informacija putem Interneta predstavlja ekvivalent pisanju poruka i slanju preko pošte. Sve opasnosti koje su postojale u tradicionalnoj razmeni informacija postoje i sada. Kao pandan potpisima i pečatima nastaju digitalni potpisi. Obezbeđuju iste servise, ali ako su pravilno implementirani mnogo ih je teže falsifikovati. Digitalni potpisi se oslanjaju na asimetričnu kriptografiju (šifrovanje javnim ključem) u kojoj se koriste različiti ključevi za šifrovanje i dešifrovanje. Poruka se enkriptuje privatnim ključem pošiljaoca, dok se za dekripciju koristi njegov poznati javni ključ. Tako svi koji prime poruku mogu da utvrde neizmenjenost sadržaja i autentikuju pošiljaoca, jer je samo onaj ko poseduje privatni ključ koji odgovara korišćenom javnom ključu mogao da enkriptuje poruku.

Digitalni potpisi imaju svoju primenu u digitalnim sertifikatima. Kada bi se javni ključevi slali direktno od vlasnika ka drugim učesnicima u komunikaciji postojao bi rizik od presretanja poruke i zamene javnog ključa drugim (*man-in-the-middle attack*). [2] Zato se oslanjamo na sertifikacione autoritete (dalje u tekstu CA), velike organizacije koje izdaju, upravljaju i opozivaju digitalne sertifikate. Sertifikati vezuju informaciju o javnom ključu sa informacijama o njegovom vlasniku i najpribližnije bi bili opisani kao virtuelne lične karte. Nalik njima, sertifikati takođe poseduju informacije o svom vlasniku, datum izdavanja, organizaciju koja ih je izdala i potpis te organizacije, ali za razliku od lične karte, posedovanje nečijeg sertifikata je beskorisno u cilju lažnog predstavljanja bez poznavanja privatnog ključa vlasnika. [3] U većini slučajeva, kada otvorimo neki od poznatijih web sajtova preko jednog

od internet pretraživača, pored adrese sajta pojavi se ikona katanca, što označava da je konekcija privatna i da server poseduje validan i pouzdani sertifikat. Sigurna veza između servera i klijenta, u konkretnom slučaju web sajta i pretraživača, ostvaruje se preko SSL protokola koji koristi digitalne sertifikate da omogući razmenu osetljivih informacija kao što su npr. broj kreditne kartice ili akreditivi za prijavljivanje na web sajt.

U doba masovne komunikacije, društvenih mreža i on-line transakcija, gde je privatnost od velikog značaja, očigledno je da digitalni sertifikati igraju veliku ulogu u zaštiti informacija. Sam proces njihovog kreiranja, potpisivanja i validacije je veoma apstraktan i obrađuje se kao jedna od tema u sklopu predmeta Zaštita podataka na Elektrotehničkom fakultetu u Beogradu. U cilju što boljeg razumevanja strukture digitalnih sertifikata, kao i njihovog generisanja i upotrebe, postoji projektni zadatak koji podrazumeva implementaciju aplikacije sa grafičkim korisničkim interfejsom (dalje u tekstu GUI) u programskom jeziku Java za kreiranje i potpisivanje X.509 digitalnih sertifikata. X.509 predstavlja jedan od najpopularnijih standarda koji određuju format sertifikata sa javnim ključevima kreiran od strane ITU-T. [4] Aplikacija treba da omogući funkcionalnosti kao što su:

- generisanje novog para ključeva,
- uvoz/izvoz postojećeg para ključeva,
- potpisivanje sertifikata i
- izvoz kreiranog sertifikata.

Cilj ovog rada je da se projektuje i implementira zahtevani softverski sistem sa proširenim funkcionalnostima u kom će implementacija GUI-a biti potpuno razdvojena od implementacije dela sistema za upravljanje sertifikatima, što bi kasnije omogućilo studentima da, na osnovu priloženog GUI-a i interfejsa logike rada, sami implementiraju sve karakteristike sistema koje dati GUI podržava kako bi se dobila funkcionalna aplikacija za rad sa digitalnim sertifikatima. Realizacija grafičkog korisničkog interfejsa obuhvata veliki deo projektnog zadatka i oduzima mnogo vremena, koje bi moglo biti iskorišćeno na implementaciju većeg broja funkcionalnosti, a nije presudna za razumevanje suštine upotrebe digitalnih sertifikata. Pored toga, cilj je realizovati i uprošćeni server – klijent sistem za verifikaciju koji koristi SSL konekciju za komunikaciju. Za autentikaciju servera treba koristiti X.509 sertifikat generisan u realizovanom sistemu za upravljanje digitalnim sertifikatima.

Drugo poglavlje ovog rada posvećeno je izgradnji teorijske podloge za razumevanje X.509 standarda za upravljanje digitalnim sertifikatima. Pružen je kratak pregled istorije standarda, a zatim detaljno opisana struktura sertifikata koji je u skladu sa X.509 standardom. Na kraju je objašnjen proces generisanja, izdavanja i verifikacije sertifikata.

Opis sistema koji je realizovan u sklopu praktičnog dela ovog rada predstavljen je u trećem poglavlju. U prvom potpoglavlju su navedene korišćene tehnologije za razvijanje sistema.

Skup funkcionalnosti koje sistem obezbeđuje dat je u drugom potpoglavlju. Detaljno opisana arhitektura celokupnog sistema data je u trećem potpoglavlju uz priložene UML dijagrame paketa i klasa. Posebno, četvrto potpoglavlje, posvećeno je dizajnu konfigurabilnog grafičkog korisničkog interfejsa, obzirom da je njegova implementacija jako složena. U poslednjem potpoglavlju nabrojani su uočeni problemi prilikom razvijanja sistema.

Primer korišćenja celokupnog konfigurabilnog sistema za kreiranje i verifikaciju aplikacija za rad sa X.509 sertifikatima dat je u četvrtom poglavlju, kroz interakciju sa 3 grupe korisnika. Konfiguracija grafičkog korisničkog interfejsa i izvoz izvornog koda aplikacije za generisanje i potpisivanje sertifikata prikazani su u prvom potpoglavlju. U drugom potpoglavlju dat je primer kada studenti na osnovu dobijenih biblioteka iz prvog potpoglavlja treba da implementiraju funkcionalnosti aplikacije. Treće potpoglavlje sadrži ilustraciju primera generisanja jednog lanca X.509 sertifikata. Generisani lanac sertifikata se koristi u primeru iz četvrtog potpoglavlja za uspostavljanje SSL konekcije između servera i klijenta.

Peto poglavlje obuhvata kratak rezime rada uz prikaz mogućih unapređenja.

2. X.509 SERTIFIKATI

U ovom poglavlju je pružena teorijska osnova za razumevanje X.509 standarda za upravljanje digitalnim sertifikatima. Na početku poglavlja se daje kratak istorijat X.509 standarda, a zatim se fokusira na objašnjenje strukture formata X.509 sertifikata, preciznije na opis polja sadržanih u sertifikatu pomenutog formata. Najpre će biti opisani X.509 sertifikati verzija 1 i 2, a zatim verzije 3 koja podržava opcione ekstenzije. Nakon toga, ukratko će biti opisan postupak izdavanja sertifikata od strane CA i njihovog korišćenja u server – klijent komunikaciji.

2.1. ISTORIJAT X.509 STANDARDA

X.509 standard je nastao u direktnoj asocijaciji sa X.500 standardom. Za razumevanje principa X.509 standarda, potrebno je razumeti i njegovu istoriju. X.500 je zamišljen kao globalna, distribuirana baza podataka imenovanih entiteta: ljudi, računara, štampača, itd. Drugim rečima, trebalo je da bude globalni, on-line telefonski imenik. [5] Svaki ulaz "imenika" ima jedinstveno ime (Distinguished Name). Osnovu koncepta čini hijerarhijska organizacija tih ulaza distribuirana preko jednog ili više servera. [6] X.509 sertifikati su prvenstveno definisani da povežu javne ključeve sa X.500 imenima, ali danas se široko koriste van X.500 protokola i strukture. Veliki broj sertifikata je preinstaliran u Web pretraživače, za ostvarivanje bezbedne SSL konekcije.

2.2. STRUKTURA X.509 STANDARDA

X.509 digitalni sertifikati su evoluirali tokom vremena. Verzija 1 objavljena je 1988. godine, a verzija 2, sa malim izmenama u odnosu na prvu verziju, 1993. godine. Najskorije objavljena je verzija 3 koja podržava ekstenzije. Svako može da definiše ekstenziju i pripoji je sertifikatu. Postoji tačno definisana struktura ekstenzije i generalan metod njihovog dodavanja sertifikatu. Za opis strukture sertifikata po standardu se koristi formalni jezik - ASN.1. [7]

2.2.1. X.509 SERTIFIKATI VERZIJE 1 I 2

X.509 digitalni sertifikati verzije 1 i 2 sastoje se od: tbs sertifikata (*tbs certificate*), algoritma potpisa (*signature algorithm*) i vrednosti potpisa (*signature value*). [8]

Tbs certificate sadrži informacije vezane za vlasnika sertifikata i izdavača, javni ključ, rok važenja i slično. Čine ga sledeća polja:

- **Version** – polje koje sadrži informaciju o verziji sertifikata (1, 2 ili 3). Podrazumevana vrednost je 0, što označava sertifikat verzije 1.
- **Serial number** – serijski broj sertifikata. Predstavlja numerički identifikator jedinstven na nivou CA.
- **Signature algorithm identifier** – sadrži identifikator algoritma upotrebljenog od strane CA prilikom potpisivanja sertifikata. Ovo polje specificira korišćene *public-key* i

hashing algoritme.

- **Issuer X.500 name** – identifikuje entitet koji je potpisao i izdao sertifikat. Sadrži X.500 jedinstveno ime CA koji je potpisao sertifikat. X.500 imena su sastavljena od atributa kao npr. *country, state, common name*, itd...
- **Validity period** – predstavlja vremenski interval za vreme kog CA garantuje da će održavati informacije o statusu sertifikata. Čine ga dva datuma, datum kada počinje period važenja (*not before*) i kada se završava (*not after*).
- **Subject X.500 name** – identifikuje entitet povezan sa javnim ključem sačuvanim u polju za informacije o javnom ključu. Sadrži X.500 jedinstveno ime vlasnika sertifikata.
- **Subject public key information** – sadrži dve veoma bitne informacije: vrednost javnog ključa vlasnika sertifikata i identifikator algoritma sa kojim javni ključ može biti korišćen.
- **Issuer unique identifier (Version 2)** – opciono polje koje sadrži informaciju o jedinstvenoj identifikaciji X.500 imena izdavača sertifikata, u slučaju da je tokom vremena isto ime dodeljivano različitim entitetima.
- **Subject unique identifier (Version 2)** - opciono polje koje sadrži informaciju o jedinstvenoj identifikaciji X.500 imena vlasnika sertifikata, u slučaju da je tokom vremena isto ime dodeljivano različitim entitetima.

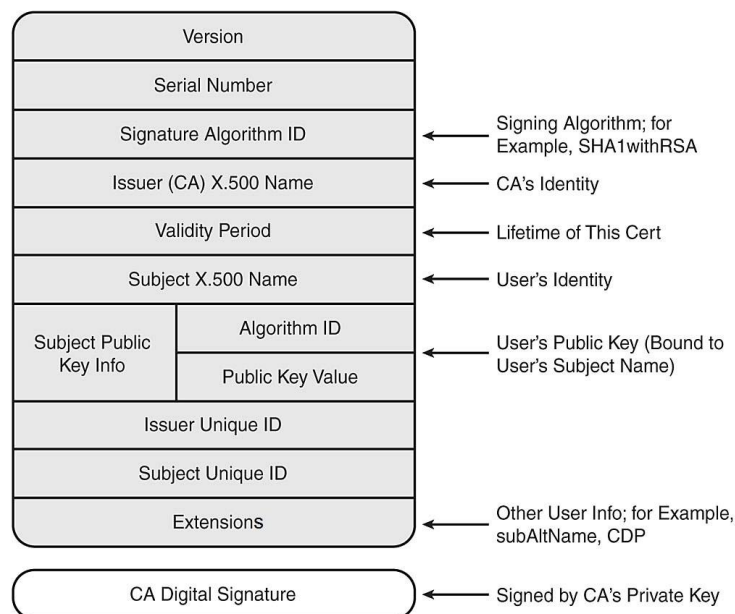
Verzija 2 X.509 sertifikata nije u širokoj upotrebi. Osnovni razlozi za to su što postoje bolje metode za jedinstvenu identifikaciju entiteta, a i standard ne preporučuje ponovnu upotrebu jedinstvenih imena entiteta tokom vremena.

Signature algorithm sadrži informacije o kriptografskom algoritmu upotrebljenog od strane CA prilikom potpisivanja sertifikata. Ovo polje specificira korišćene *public-key* i *hashing* algoritme. Sastoji se od identifikatora i parametara algoritma.

Signature value sadrži digitalni potpis sračunat na osnovu tbs sertifikata koji predstavlja ulaz u funkciju za potpisivanje. Generisanjem ovog potpisa, CA garantuje validnost informacija u tbs sertifikatu i povezuje javni ključ sa vlasnikom tbs sertifikata.

2.2.2. X.509 SERTIFIKATI VERZIJE 3

Sertifikati verzije 3 (slika 1) sadrže sva polja koja sadrže sertifikati verzije 1 i 2 uz proširenje u vidu opcionih ekstenzija. Ekstenzije su moguće samo u sertifikatima verzije 3 i predstavljaju generički mehanizam za povezivanje dodatnih atributa sa javnim ključem i vlasnikom sertifikata, kao i metod za upravljanje vezama između sertifikacionih autoriteta. Pored standardom definisanih ekstenzija moguće je definisati nove ekstenzije. Sertifikat ne sme da sadrži više od jedne instance specifične ekstenzije. Sve ekstenzije se sastoje is 3 osnovna polja: identifikator tipa (*type*), kritičnost (*criticality*) i vrednost (*value*). [9]



(Slika 1) *Struktura X.509 sertifikata verzije 3.* [10]

Type sadrži informaciju o tipu podataka koja se nalaze u polju za vrednost ekstenzije. Istovremeno predstavlja i identifikator ekstenzije (*Object identifier - OID*). Preporučuje se da svi tipovi ekstenzija budu registrovani kod međunarodno priznate organizacije za standarde, npr. ISO.

Criticality je jednobitni flag i na osnovu njega ekstenzija može biti *critical* ili *non-critical*. Ako je ekstenzija označena kao *critical* to znači da ona sadrži neke informacije koje aplikacija koja koristi sertifikat ne sme da zanemari. Ako sistem ne prepozna ekstenziju označenu sa *critical* ili ne može da je obradi, onda on mora da odbaci taj sertifikat. *Non-critical* ekstenzija može biti zanemarena ako sistem nije u stanju da je prepozna, ali u slučaju da je prepozna je mora je obraditi.

Value polje sadrži same podatke ekstenzije koji su *DER* kodirani. [11] Podaci su tipa predstavljenog *type* poljem ekstenzije.

Svi CA koji podležu standardu treba da pruže podršku za sledeće ekstenzije: *key identifiers*, *basic constraints*, *key usage* i *certificate policies*. Podrška za druge ekstenzije je opcionala. Standard jasno definiše i ekstenzije koje bi svi sistemi koji koriste X.509 sertifikate morali da prepoznaju. To su: *key usage*, *certificate policies*, *subject alternative name*, *basic constraints*, *name constraints*, *policy constraints*, *extended key usage* i *inhibit anyPolicy*. Dodatno, trebalo bi da mogu da prepoznaju i *authority* i *subject key identifier* i *policy mappings*. Dalje u tekstu su redom navedene standardne ekstenzije i data njihova objašnjenja.

- **Authority key identifier** – jedinstveni identifikator javnog ključa koji odgovara privatnom ključu korišćenom za potpisivanje sertifikata. Služi u slučaju kada CA za

vreme svog životnog veka koristi više parova ključeva za potpisivanje sertifikata. Vrednost ovog polja bi trebalo da bude generisana iz vrednosti javnog ključa koji se koristi za verifikaciju sertifikata ili pomoću generatora jedinstvenih numeričkih vrednosti.

- **Subject key identifier** – ima sličnu svrhu kao i *authority key identifier*. Služi da identifikuje sertifikat koji sadrži specifičan par ključeva. Svi sertifikati izdati od strane jednog CA moraju u svom polju *authority key identifier* imati vrednost polja *subject key identifier* CA sertifikata.
- **Key usage** – niz bitova koji predstavlja svrhe u koje se ključ sertifikata može upotrebiti (tabela 1). Ako je ova ekstenzija označena kao *critical* onda se par ključeva može upotrebiti samo u navedene svrhe. U suprotnom služi kao pomoć aplikacijama da brzo pronađu odgovarajući par ključeva za određenu upotrebu.

No	bit	upotreba
0	<i>digitalSignature</i>	verifikacija digitalnih potpisa osim potpisa na digitalnim sertifikatima i listama opozivanja sertifikata (<i>CRLs</i>)
1	<i>nonRepudiation</i>	pruža servise neporecivosti porekla
2	<i>keyEncipherment</i>	šifrovanje privatnih i tajnih ključeva, npr. za vreme transporta
3	<i>dataEncipherment</i>	šifrovanje sirovih korisničkih podataka bez upotrebe posredničkih algoritama simetričnog šifrovanja
4	<i>keyAgreement</i>	koristi se za sporazumevanje oko ključeva
5	<i>keyCertSign</i>	verifikacija digitalnih potpisa na digitalnim sertifikatima
6	<i>CRLSign</i>	verifikacija digitalnih potpisa na listama opozivanja sertifikata
7	<i>encipherOnly</i>	šifrovanje podataka prilikom sporazumevanja oko ključeva
8	<i>decipherOnly</i>	dešifrovanje podataka prilikom sporazumevanja oko ključeva

(Tabela 1) Značenje bita u *key usage* ekstenziji.

- **Certificate policies** – sekvenca jedne ili više informacija o politici izdavanja sertifikata korisniku i/ili moguće upotrebe sertifikata. Informaciju o određenoj politici čini specijalno formatiran broj OID-a (*Object Identifier*) registrovan kod međunarodno priznate organizacije za standarde. Postoji specijalna vrednost OID-a *anyPolicy*. [12]
- **Policy mappings** – koristi se samo u CA sertifikatima i to u *cross-certification* procesu kada dva CA izdaju jedan drugom sertifikate. Sastoji se od jednog ili više parova OID-

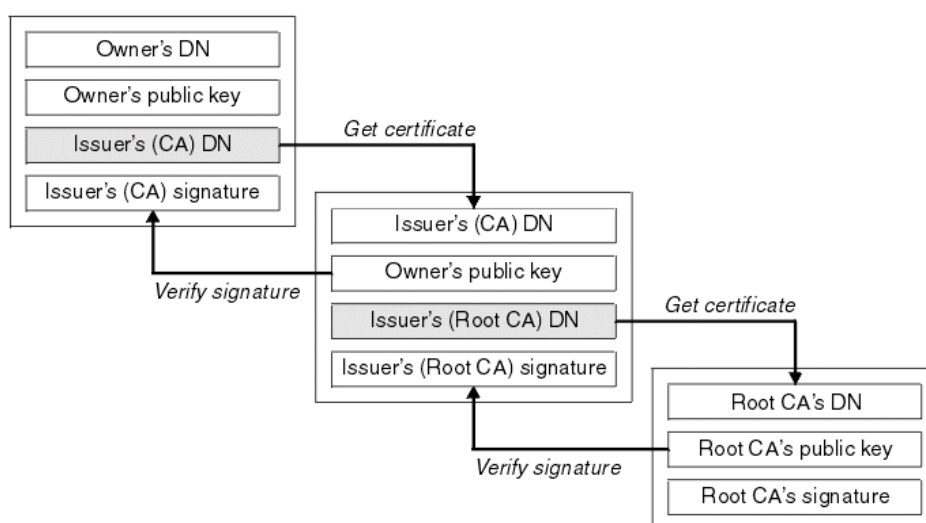
ova. Pruža mehanizam mapiranja politike CA koji potpisuje sertifikat na politiku CA navedenog u cross-sertifikatu.

- ***Subject alternative name*** – određuje jedno ili više jedinstvenih imena za vlasnika sertifikata. Neke od dozvoljenih formi imena su: e-mail adresa, IP adresa, DNS ime,...
- ***Issuer alternative name*** – određuje jedno ili više jedinstvenih imena za CA koji je potpisao sertifikat. Dozvoljene forme imena su iste kao za *subject alternative name*.
- ***Subject directory attributes*** – pruža mogućnost dodavanja dodatnih identifikacionih informacija o vlasniku sertifikata pored onih koje su pružene poljima *Subject X.500 name* i *Subject alternative name*. Moguće je dodati npr. datum rođenja, pol, državljanstvo,...
- ***Basic constraints*** – sastoji se od dva polja. Jedno polje je boolean flag koje označava da li je vlasnik sertifikata CA ili krajnji korisnik. Drugo polje (*pathLenConstraint*) ima svrhu samo ako je vlasnik označen kao CA i ako je vrednost *keyCertSign* bita u *key usage* ekstenziji postavljena na 1. Predstavlja maksimalan broj posredničkih sertifikata koji mogu da prate ovaj sertifikat na validnom sertifikacionom putu.
- ***Name constraints*** – koristi se u cross-sertifikatima. Pruža mogućnost CA koji izdaje cross-sertifikat da definiše domen prihvatljivih imena u lancu sertifikata koji se proteže od tog cross-sertifikata.
- ***Policy constraints*** – koristi se u cross-sertifikatima. Pruža mogućnost CA koji izdaje cross-sertifikat da definiše skup prihvatljivih politika u lancu sertifikata koji se proteže od tog cross-sertifikata.
- ***Extended key usage*** – označava dodatne svrhe u koje se javni ključ može upotrebiti pored onih navedenih u *key usage* ekstenziji. Svaka moguća upotreba označena je jedinstvenim OID-om. Moguće vrednosti su: *any extended key usage*, *server authentication*, *client authentication*, *code signing*, *e-mail protection*, *time stamping* i *OCSP signing*.
- ***CRL distribution points*** – identifikuje način dobijanja informacija o listama opozivanja sertifikata (*Certificate revocation lists*).
- ***Inhibit anyPolicy*** – vrednost označava broj dodatnih ne-samo-izdatih sertifikata koji se mogu pojaviti na validnom putu pre nego što *anyPolicy* više nije dozvoljena.
- ***Freshest CRL*** – identifikuje način dobijanja informacija o *delta CRL*-ama. [13]

2.3. INFRASTRUKTURA JAVNIH KLJUČEVA (PKI)

Public key infrastructure je skup uloga, politika i procedura potrebnih za kreiranje, upravljanje, distribuiranje, upotrebu, čuvanje i opozivanje digitalnih sertifikata i upravljanje enkripcijom javnim ključevima. [14] Kao što je već rečeno, digitalni sertifikati povezuju javne ključeve sa identitetom vlasnika ključa, a to povezivanje se postiže registracijom i izdavanjem digitalnih sertifikata od strane sertifikacionih autoriteta (*certificate authorities*). CA su velike organizacije u koje imamo poverenja, kao što su npr Symantec, Comodo i GoDaddy koji drže većinu svetskog tržišta. Dakle, osnovna uloga CA je potpisivanje javnog ključa vlasnika sertifikata svojim privatnim ključem i njegovo objavljivanje. Time se poverenje u vlasnika sertifikata oslanja na poverenje u CA koji je potpisao taj sertifikat. Često su CA hijerarhijski organizovani. U takvoj organizaciji postoje dva tipa CA: *root* i *intermediate* sertifikacioni autoriteti. Tako sertifikat nakon izdavanja postaje deo lanca sertifikata na čijem vrhu se nalazi sertifikat korenog CA koji je *self-issued* (sam sebe potpisao). Između *root* CA i krajnjeg digitalnog sertifikata može se naći više *intermediate* CA sertifikata.

Taj proces najbolje se može razumeti na primeru. Jedan od primera korišćenja sertifikata je u sigurnoj server – klijent HTTPS konekciji. Biće opisan samo slučaj autentikacije servera. Pri uspostavljanju odnosa server klijentu šalje svoj sertifikat. Da bi SSL konekcija bila uspostavljena klijent mora da proveri da li je sertifikat validan i to radi tako što proverava njegov potpis. Ako klijent veruje sertifikacionom autoritetu koji je izdao ključ i može da verifikuje njegov potpis, onda može da zaključi da je sertifikat validan i uspostavlja sigurnu vezu sa serverom. Potrebno je proveriti sve sertifikate u lancu sertifikata (slika 2).



(Slika 2) Validacija lanca sertifikata. [15]

3. IMPLEMENTACIJA I DIZAJN

Ovo poglavlje posvećeno je opisu dizajna i implementacije samog sistema. U prvom potpoglavlju su nabrojane korišćene tehnologije za implementaciju sistema. U drugom potpoglavlju su opisane opšte karakteristike i funkcionalnosti sistema. Treće potpoglavlje daje detaljan uvid u arhitekturu celokupnog sistema sa izuzetkom dizajna grafičkog korisničkog interfejsa. Dizajn i izgled GUI-a je posebno opisan u četvrtom potpoglavlju. Na kraju, u petom potpoglavlju, nabrojani su uočeni problemi prilikom dizajna i implementacije sistema i načini na koji su prevaziđeni.

3.1. KORIŠĆENE TEHNOLOGIJE

Prilikom razvoja praktičnog dela ovog rada korišćen je programski jezik **Java**, prvenstveno zbog portabilnosti. Prednost je data Javi i zbog već postojećih paketa klasa i *open source* biblioteka koje olakšavaju rad sa digitalnim sertifikatima i kriptografskim algoritmima. Korišćen je *Bouncy Castle* API verzija 1.54 [16]. Za lakši unos i prikaz podataka u formatu datuma na GUI-u korišćena je biblioteka *jdatepicker-1.3.4.jar*. [17] Sistem je razvijan u *Eclipse Luna* [18] integrisanom okruženju pod *Windows 10* operativnim sistemom. Korišćeno izvršno okruženje je *jre1.8.0_77*. [19]

3.2. OPIS SISTEMA

Sistem je podeljen na dva dela. Prvi deo predstavlja aplikacija za generisanje i potpisivanje X.509 digitalnih sertifikata, a drugi deo čine serverska i klijentska aplikacija za verifikaciju digitalnih sertifikata generisanih aplikacijom iz prvog dela ovog sistema.

Prvo će biti opisane karakteristike aplikacije za generisanje i potpisivanje X.509 digitalnih sertifikata. Sistem interaguje sa korisnikom preko grafičkog korisničkog interfejsa. Realizovani GUI je konfigurabilan u smislu verzije sertifikata koje aplikacija generiše, algoritama za kreiranje parova ključeva i ekstenzija verzije 3 X.509 digitalnih sertifikata. Njegov izgled se može menjati na osnovu konfiguracionih parametara koji se učitavaju iz konfiguracionog fajla ili inicijalizacijom u izvornom kodu aplikacije. Funkcionalnosti koje obezbeđuje aplikacija za generisanje i potpisivanje X.509 digitalnih sertifikata su sledeće:

- generisanje novog para ključeva jednim od podržanih algoritama,
- izvoz postojećeg para ključeva u *PKCS#12* format i zaštita fajla *AES* algoritmom,
- uvoz postojećeg para ključeva u *PKCS#12* formatu zaštićenog *AES* algoritmom,
- pregled detalja postojećih parova ključeva za X.509 sertifikat,
- potpisivanje postojećeg X.509 digitalnog sertifikata verzije 1 ili 3 jednim od podržanih algoritama,
- izvoz potpisanog sertifikata u *PEM* ili *DER* kodirani .cer fajl,
- uvoz postojećeg sertifikata iz .cer fajla *PEM* ili *DER* kodiranog.

Podržani algoritmi za generisanje parova privatnih i javnih ključeva su: *DSA*, *RSA* i *ECDSA* (tabela 2). Veličine ključa za *DSA* algoritam kreću se od 512 do 2048 bita. Veličine ključa za *RSA* algoritam kreću se od 512 do 4096 bita. Podržani skupovi eliptičnih kriva za *ECDSA* algoritam su: X9.62, SEC i NIST.

Public key algoritam	Hash algoritam
DSA	SHA1
	MD2
	MD5
RSA	SHA1
	SHA224
	SHA256
	SHA384 *
	SHA512 *
	RIPEMD128
	RIPEMD160
	RIPEMD256
ECDSA	SHA1
	SHA224
	SHA256
	SHA384
	SHA512

* dužina ključa RSA algoritma mora biti minimum 1024 bita

(Tabela 2) Tabela podržanih algoritama za potpisivanje X.509 digitalnih sertifikata.

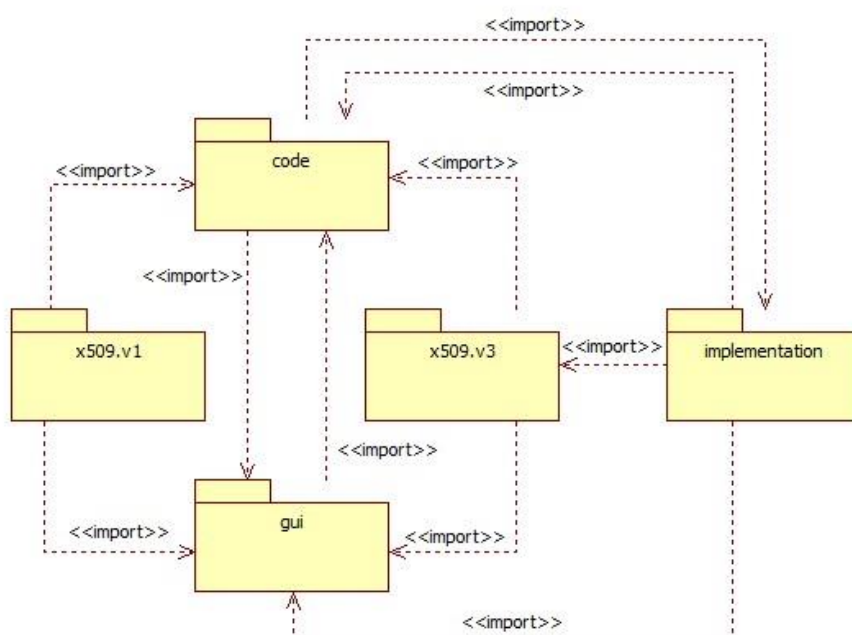
Podržane ekstenzije verzije 3 X.509 digitalnih sertifikata su: *authority* i *subject key identifiers*, *key usage*, *certificate policies*, *subject* i *issuer alternative name*, *subject directory attributes*, *basic constraints*, *extended key usage* i *inhibit anyPolicy*.

Deo sistema za verifikaciju aplikacije za generisanje X.509 digitalnih sertifikata čine jedna serverska i jedna klijentska aplikacija sa kojima korisnik komunicira preko komandne linije i konzole. Osnovna funkcionalnost koju obezbeđuje ovaj deo sistema je ostvarivanje uspešne SSL konekcije između serverske i klijentske aplikacije i slanje znakovne poruke od klijenta ka serveru preko te sigurne veze.

3.3. OPŠTA ARHITEKTURA SISTEMA

Pri projektovanju sistema akcenat je stavljen na tri problema vezana za aplikaciju za generisanje i potpisivanje X.509 sertifikata. Prvi problem bio je napraviti sistem gde postoji jasna granica u izvornom kodu između implementacije GUI-a i logike rada sa digitalnim sertifikatima. Takva arhitektura sistema bi omogućila studentima na predmetu Zaštita

podataka da bolje savladaju gradivo vezano za X.509 sertifikate tako što bi za zadatak imali implementaciju funkcionalnosti aplikacije bez potrebe za realizacijom GUI-a. Drugi problem bio je napraviti konfigurabilan GUI čiji je izgled moguće menjati na osnovu konfiguracionih parametara. Podesiv GUI dovodi do velikog broja različitih kombinacija algoritama za generisanje parova ključeva i ekstenzija verzije 3, i omogućava različitu složenost aplikacije koju studenti treba da implementiraju. Treći problem bio je napraviti sistem koji će biti lako nadogradiv. Trebalo je olakšati kasnije dodavanje novih algoritama za generisanje parova ključeva i novih ekstenzija verzije 3 X.509 digitalnih sertifikata, a eventualno i dodavanje polja u slučaju novih verzija. Ispunjavanjem ta tri zahteva dobijen je sistem sledeće arhitekture (slika 3):



(Slika 3) UML dijagram paketa aplikacije za generisanje i potpisivanje X.509 digitalnih sertifikata.

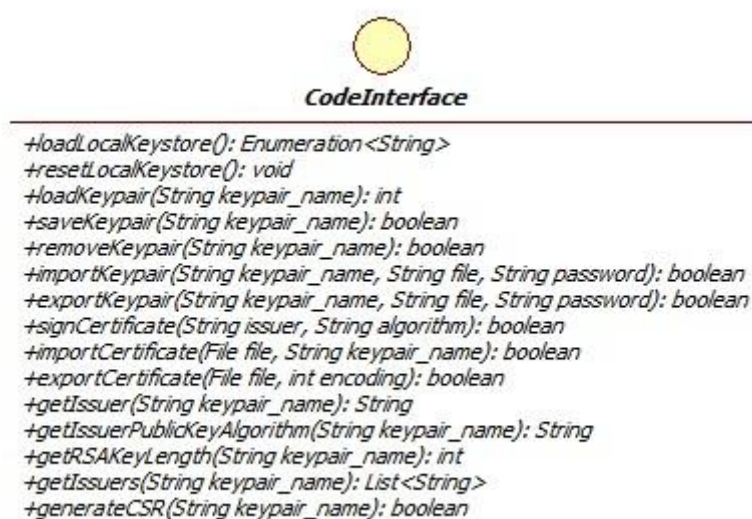
Izvorni kod je raspoređen po paketima na takav način da se odabirom *gui*, *code* i jednog od *x509.v1* ili *x509.v3* paketa lako dobija .jar biblioteka čijim korišćenjem studenti treba da dobiju aplikaciju koja ispunjava zadate zahteve.

Ulazna tačka programa je klasa *X509* i ona se nalazi u paketu *code*. Pokretanjem programa kreira se novi objekat klase *MyCode* čijem konstruktoru se prosleđuju konfiguracioni parametri GUI-a *algorithm_conf* i *extensions_conf*. Niz *boolean* vrednosti *algorithm_conf* je dužine 4 i svaki njegov element odgovara jednom od algoritama za generisanje parova ključeva, redom *DSA*, *RSA*, *GOST* (nije implementiran, ali postoji mogućnost iscrtavanja GUI-a sa tom opcijom, uz sitne dorade) i *EC*. Vrednost njegovih elemenata označava da li na GUI-u treba iscrtati mogućnost za generisanje para ključeva odgovarajućim algoritmom. Niz

boolean vrednosti *extensions_conf* je dužine 15 i svaki njegov element odgovara jednoj od standardnih ekstenzija verzije 3 X.509 digitalnih sertifikata, redom *authority* i *subject key ID*, *key usage*, *certificate policies*, *policy mappings* (nije implementirana, niti je GUI podržava), *subject* i *issuer alternative name*, *subject directory attributes*, *basic constraints*, *name constraints* (nije implementirana, niti je GUI podržava), *policy constraints* (nije implementirana, niti je GUI podržava), *extended key usage*, *CRL distribution points* (nije implementirana, niti je GUI podržava), *inhibit anyPolicy*, *freshes CRL* (nije implementirana, niti je GUI podržava). Vrednost njegovih elemenata označava da li na GUI-u treba iscrtati panel za unos i prikaz podataka odgovarajuće ekstenzije. Vrednosti elemenata ovih konfiguracionih nizova se postavljaju u *main* funkciji klase *X509* pre kreiranja objekta tipa *MyCode* tako što se učitaju iz konfiguracionog fajla čiji se naziv prosleđuje preko argumenata komandne linije. U slučaju da nije naveden konfiguracioni fajl, GUI se konfiguriše da podržava sve implementirane algoritme i ekstenzije.

Paket **gui** sadrži sve klase vezane za implementaciju grafičkog korisničkog interfejsa. Klasa čija instanca predstavlja sam prozor aplikacije je klasa *MainFrame*. Objekti te klase u sebi sadrže reference na sve panele sa dugmičima i poljima za unos i prikaz podataka. Dobijanje podataka sa grafičkog korisničkog interfejsa i prikazivanje na njemu omogućeno je preko javnog interfejsa ka GUI-u u formi apstraktnih klasa *GuiInterfaceV1*, *GuiInterfaceV* i *GuiInterfaceV3* koje imaju polje tipa *MainFrame*. Detaljan dizajn *gui* paketa biće posebno opisan u sledećem potpoglavlju.

U paketu **code** se, između ostalog, nalaze i klase za obradu izuzetaka i uslužna klasa za formatiranje datuma. U taj paket je smešten *CodeInterface*, interfejs sa potpisima svih metoda koje student treba da implementira da bi se dobila funkcionalna aplikacija (slika 4). Dat je UML prikaz interfejsa, a zatim opis funkcionalnosti koje bi svaka metoda trebalo da ispuni.



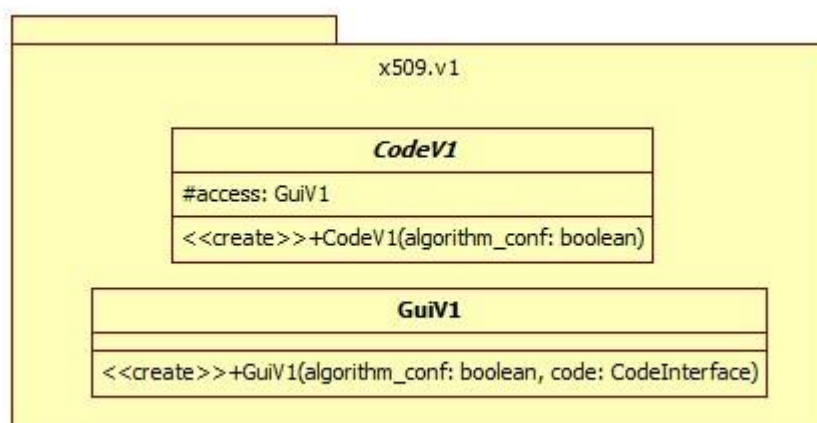
(Slika 4) UML prikaz interfejsa *CodeInterface*.

- Metoda *loadLocalKeystore()* treba da učitava lokalno skladište ključeva i kao povratnu vrednost vrati listu *alias*-a za parove ključeva/sertifikata u *keystore*-u.
- Metoda *resetLocalKeystore()* treba da obriše lokalno skladište ključeva.
- Metoda *loadKeypair(String keypair_name)* treba da učitava podatke o paru ključeva/sertifikatu iz lokalnog *keystore*-a koji je sačuvan pod aliasom *keypair_name* i prikaže ih na grafičkom korisničkom interfejsu. Povratna vrednost metode je celobrojna vrednost koja označava uspešnost operacije. Metoda vraća -1 u slučaju greške, 0 u slučaju da sertifikat sačuvan pod tim aliasom nije potpisan, 1 u slučaju da je potpisan, 2 u slučaju da je u pitanju uvezeni *trusted* sertifikat.
- Metoda *saveKeypair(String keypair_name)* treba da na osnovu podataka sa grafičkog korisničkog interfejsa generiše i sačuva novi par ključeva u lokalnom *keystore*-u pod aliasom sa vrednošću *keypair_name*. Povratna vrednost metode označava uspešnost operacije, *false* u slučaju greške.
- Metoda *removeKeypair(String keypair_name)* treba da iz lokalnog *keystore*-a obriše par ključeva/sertifikat koji je sačuvan pod aliasom *keypair_name*. Povratna vrednost metode označava uspešnost operacije, *false* u slučaju greške.
- Metoda *importKeypair(String keypair_name, String file, String password)* treba da iz fajla sa putanjom *file* učitava postojeći par ključeva koji je sačuvan u *PKCS#12* formatu i zaštićen lozinkom i *AES* enkripcijom i sačuva ga u lokalni *keystore* pod aliasom *keypair_name*. Povratna vrednost metode označava uspešnost operacije, *false* u slučaju greške.
- Metoda *exportKeypair(String keypair_name, String file, String password)* treba da postojeći par ključeva koji je u lokalnom *keystore*-u sačuvan pod aliasom *keypair_name* izveze u fajl sa putanjom *file* u *PKCS#12* formatu i zaštiti lozinkom i *AES* enkripcijom. Povratna vrednost metode označava uspešnost operacije, *false* u slučaju greške.
- Metoda *signCertificate(String issuer, String algorithm)* treba da potpiše algoritmom *algorithm* trenutno selektovani sertifikat na grafičkom korisničkom interfejsu privatnim ključem sertifikata koji je u lokalnom *keystore*-u sačuvan pod aliasom *issuer*. Povratna vrednost metode označava uspešnost operacije, *false* u slučaju greške.
- Metoda *importCertificate(File file, String keypair_name)* treba da iz fajla *file* (ekstenzije *.cer*) učitava postojeći sertifikat i sačuva ga u lokalni *keystore* pod aliasom *keypair_name*. Povratna vrednost metode označava uspešnost operacije, *false* u slučaju greške.
- Metoda *exportCertificate(File file, int encoding)* treba da u fajl *file* (ekstenzije *.cer*) izveze postojeći sertifikat trenutno selektovan na grafičkom korisničkom interfejsu i kodira ga na način naznačen vrednošću parametra *encoding* (0 za *DER*, 1 za *PEM*). Povratna vrednost metode označava uspešnost operacije, *false* u slučaju greške.
- Metoda *getIssuer (String keypair_name)* treba da vrati podatke o izdavaču sertifikata

koji je u lokalnom *keystore*-u sačuvan pod aliasom *keypair_name*.

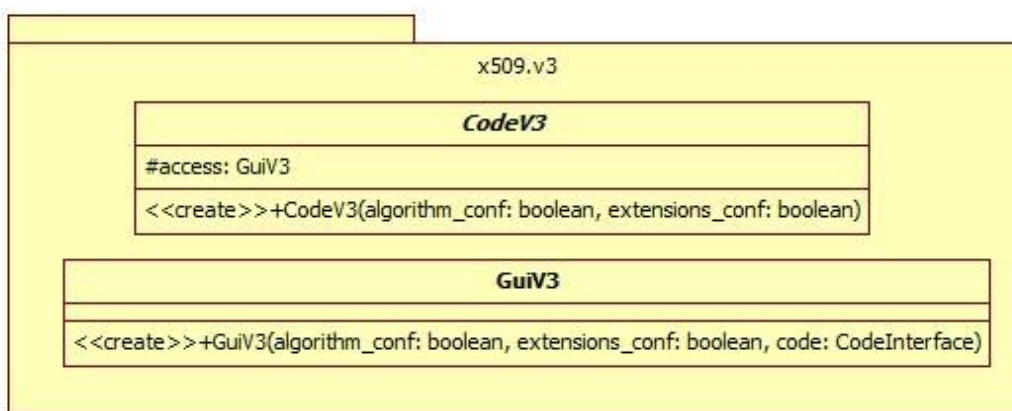
- Metoda *getIssuerPublicKeyAlgorithm (String keypair_name)* treba da vrati podatke o algoritmu koji je korišćen za generisanje para ključeva sertifikata koji je u lokalnom *keystore*-u sačuvan pod aliasom *keypair_name*.
- Metoda *getRSAKeyLength (String keypair_name)* treba da vrati dužinu ključa sertifikata koji je u lokalnom *keystore*-u sačuvan pod aliasom *keypair_name* u slučaju da je povratna vrednost prethodne metode bila *''RSA''*. Koristi se za proveravanje dozvoljenih kombinacija dužine ključeva *RSA* algoritma i *hash* algoritama.
- Metoda *getIssuers(String keypair_name)* treba da vrati listu *alias*-a svih sertifikata sačuvanih u lokalnom *keystore*-u koji mogu da potpišu sertifikat koji je u lokalnom *keystore*-u sačuvan pod aliasom *keypair_name*.
- Metoda *generateCSR(String keypair_name)* treba da generiše zahtev za potpisivanje sertifikata (*CSR*) koji je u lokalnom *keystore*-u sačuvan pod aliasom *issuer*. Povratna vrednost metode označava uspešnost operacije, *false* u slučaju greške.

Paketi *x509.v1* i *x509.v3* su osmišljeni tako da enkapsuliraju interfejske ka logici rada i interfejske ka GUI-u za određenu verziju X.509 digitalnih sertifikata. Tako paket *x509.vX* sadrži klase *GuiVX* i *CodeVX* (slike 5 i 6). Klasa *GuiVX* nasleđuje apstraktnu klasu *GuiInterfaceVX*. Objekat te klase iscrtava na ekranu korisnika GUI koji podržava generisanje X.509 digitalnih sertifikata do verzije X (ako je X=1 moguće je generisati sertifikate verzije 1, ako je X=3 moguće je generisati sertifikate verzija 1 i 3) i istovremeno poseduje *get/set* metode za sva polja za unos i prikaz podataka na grafičkom korisničkom interfejsu. Klasa *CodeVX* je apstraktna i implementira *CodeInterface*. Sadrži zaštićeno polje access tipa *GuiVX* koje predstavlja pristupnu tačku ka GUI-u. Na taj način klasa *CodeVX* obuhvata i interfejs ka GUI-u i interfejs ka samoj implementaciji funkcionalnosti aplikacije.



(Slika 5) UML prikaz klasa u paketu *x509.v1*.

Na slici 5 se može videti da se kao parametar konstruktora klase *CodeV1* prosleđuje niz *boolean* vrednosti *algorithm_conf*. U konstrukturu klase *CodeV1* se kreira novi *access* objekat tipa *GuiV1* i njemu se takođe prosleđuje referenca na taj niz *boolean* vrednosti, kao i referenca na objekat klase *CodeV1* koji se upravo kreira. Niz *algorithm_conf* predstavlja konfiguracioni parametar GUI-a. Vrednosti njegovih elemenata označavaju da li GUI treba da podrži odgovarajući algoritam za generisanje para ključeva. Analogno tome, konstrukturu klase *CodeV3* se prosleđuje niz *boolean* vrednosti *algorithm_conf*. Konstruktor klase *CodeV3* ima još jedan parametar i to je niz *boolean* vrednosti *extensions_conf*. Njegova svrha je slična, predstavlja konfiguracioni parametar GUI-a. Vrednosti njegovih elemenata označavaju da li GUI treba da podrži odgovarajuću ekstenziju X.509 v3 digitalnih sertifikata.

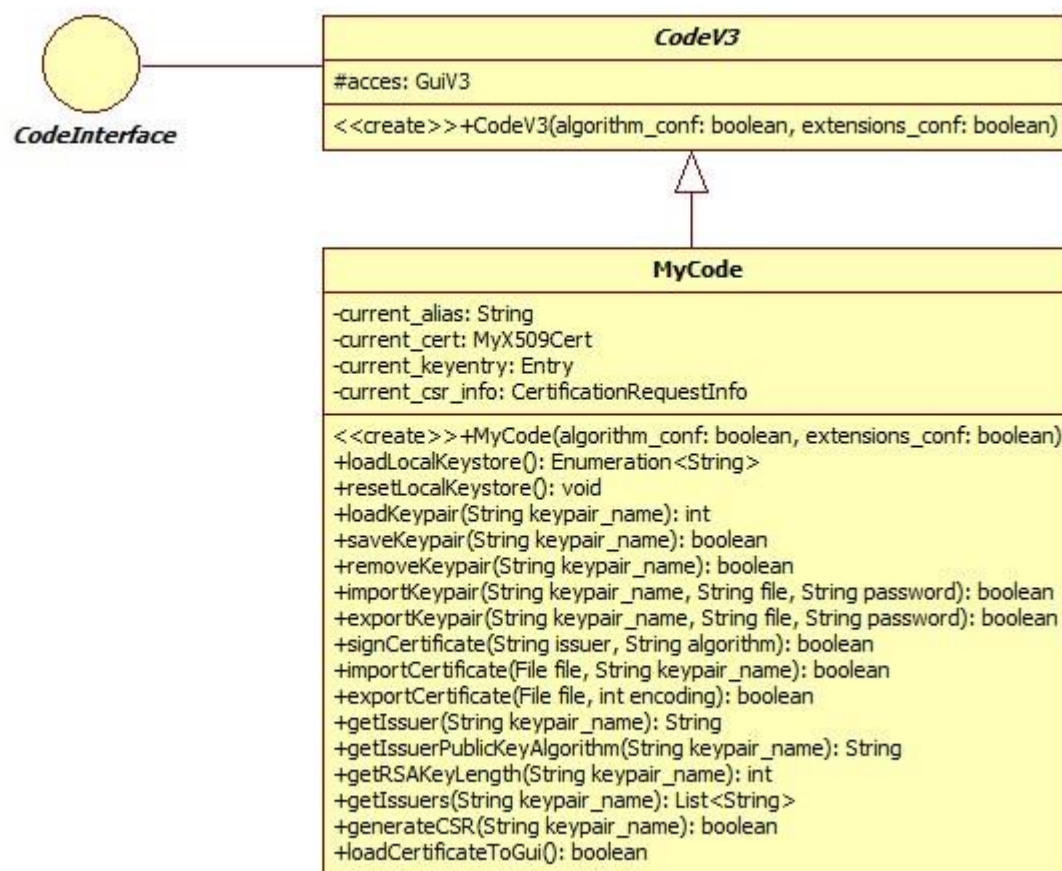


(Slika 6) UML prikaz klase u paketu *x509.v3*.

Celokupna implementacija funkcionalnosti aplikacije nalazi se u paketu **implementation**. Kompletan realizacija ovog paketa bila bi prepuštena studentima, s tim što taj paket mora da poseduje klasu *MyCode* koja nasleđuje klasu *CodeVX* u zavisnosti od toga za koje verzije sertifikata je GUI konfigurisan. Nasleđivanjem klase *CodeVX*, klasa *MyCode* nasleđuje i metode interfejsa *CodeInterface* koje je neophodno implementirati. Dakle, u slučaju da je GUI konfigurisan da podržava samo verziju 1, trebalo bi ispuniti sve zahteve da bi se dobila funkcionalna aplikacija za rad sa X.509 v1 digitalnim sertifikatima. U slučaju da je GUI konfigurisan da podržava i verziju 1 i verziju 3, trebalo bi ispuniti sve zahteve da bi se dobila funkcionalna aplikacija za rad i sa X.509 v1 i sa X.509 v3 digitalnim sertifikatima.

Što se tiče konkretnog sistema opisanog u ovom radu, implementirani su svi zahtevi za GUI koji je konfigurisan da radi sa sertifikatima verzija 1 i 3. Osovinu implementacije predstavlja, kao što je rečeno, klasa *MyCode* (slika 7). Pored toga što implementira sve metode interfejsa *CodeInterface* (njihov opis dat je ranije), sadrži i atribute koji pamte trenutno stanje na grafičkom korisničkom interfejsu. Polje *current_alias* pamti *alias* pod kojim je u lokalnom *keystore*-u sačuvan trenutno selektovani sertifikat na GUI-u. Analogno

tome *current_keyentry* čuva podatke iz ulaza lokalnog *keystore*-a o trenutno selektovanom sertifikatu na GUI-u. Polje *current_cert* je tipa *MyX509Cert*. Klasa *MyX509Cert* je korisnička omotač klasa koja olakšava rad sa Javinim X509 sertifikatima iz paketa *java.security.cert*. [20] Prilikom selektovanja određenog sertifikata na GUI-u učitava se njegov ulaz iz lokalnog *keystore*-a u *current_keyentry* i kreira se novi *current_cert* objekat tipa *MyX509Cert* koji, između ostalog, u sebi sadrži referencu na X.509 sertifikat povezan sa tim ulazom. Nakon toga se pomoću metode *loadCertificateToGui()* prikazuju podaci o tom sertifikatu na GUI-u. Polje *current_csr_info* je tipa *CertificationRequestInfo* iz *org.bouncycastle.asn1.pkcs* paketa. Njegova uloga je da čuva informacije o sertifikatu koji je odabran za potpisivanje. Novi *current_csr_info* objekat se kreira pri svakom pozivu *generateCSR(String keypair_name)*.



(Slika 7) UML prikaz hijerarhija klasa.

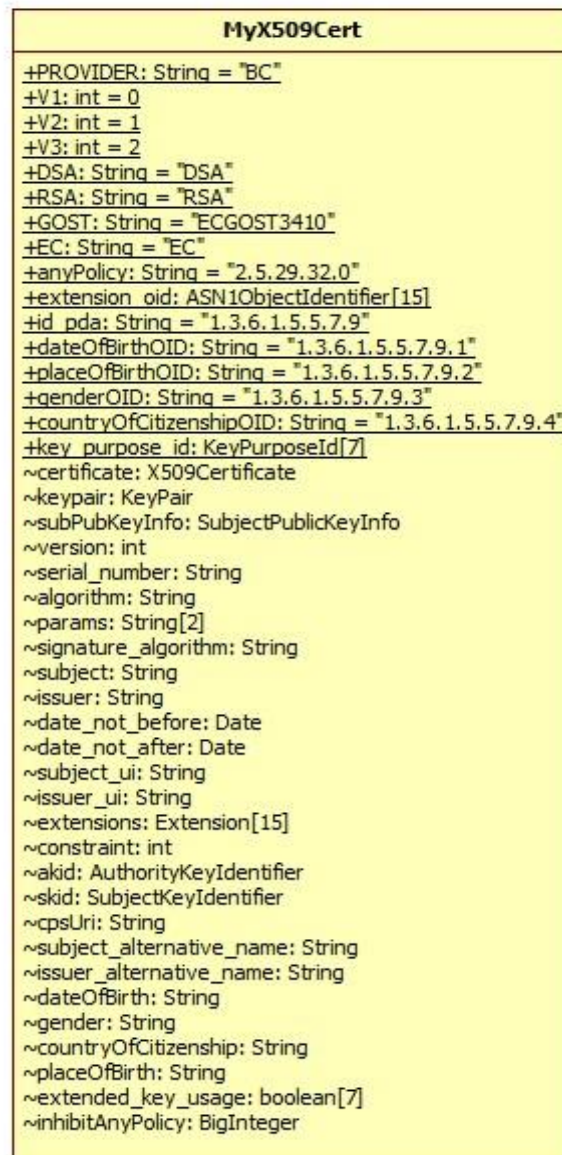
Struktura podataka odabrana za skladištenje sertifikata je Javin *KeyStore* iz paketa *java.security*. Klasa *KeyStore* obezbeđuje potpunu podršku za čuvanje privatnih ključeva i digitalnih sertifikata. Parovi ključeva se čuvaju u *PKCS#12* formatu, a sadržaj *KeyStore*-a se čuva u fajlu sa *.p12* ekstenzijom. Fajl se ažurira pri svakoj izmeni, i ponovo učitava pri svakom korišćenju. Implementirana je uslužna klasa *MyKeyStore* (slika 8) koja obavlja sve operacije vezane za čitanje i pisanje u *.p12* fajl.



(Slika 8) UML prikaz klase MyKeyStore.

Konstante *localKeyStore* i *localPassword* predstavljaju ime .p12 fajla u kome se čuva sadržaj lokalnog skladišta privatnih ključeva i sertifikata. Statičko polje *ks* tipa *KeyStore* predstavlja referencu na objekat tipa Javinog *KeyStore*-a.

- Metoda *delete(String path)* briše fajl sa putanjom *path* ako postoji.
- Metoda *removeEntry(String keypair_name)* briše iz *keystore*-a *ks* ulaz sačuvan pod alias-om *keypair_name*.
- Metoda *load(String path, String password)* učitava sadržaj .p12 fajla sa putanjom *path* koji je zaštićen lozinkom *password* u *KeyStore* strukturu podataka, ako on postoji. Referenca na *KeyStore* objekat se pamti u polju *ks*.
- Metoda *loadAES(String path, String password)* učitava sadržaj .p12 fajla sa putanjom *path* koji je zaštićen lozinkom *password* i AES enkripcijom u *KeyStore* strukturu podataka, ako on postoji. Referenca na *KeyStore* objekat se pamti u polju *ks*.
- Metoda *store(String path, String password)* ispisuje sadržaj *KeyStore* strukture podataka na koju ukazuje polje *ks* u .p12 fajl sa putanjom *path* i štiti ga lozinkom *password*.
- Metoda *storeAES(String path, String password)* ispisuje sadržaj *KeyStore* strukture podataka na koju ukazuje polje *ks* u .p12 fajl sa putanjom *path* i štiti ga lozinkom *password* i AES enkripcijom.
- Metode *putKey()*, *putCert()* i *putChain()* dodaju nove ulaze u *KeyStore* strukturu podataka na koju ukazuje polje *ks*.
- Metode *getKey()* i *getCert()* dohvataju ulaze iz *KeyStore* strukture podataka na koju ukazuje polje *ks*.
- Metoda *generateIssuerAlternativeName(Collection<List<?>> names)* kreira i vraća objekat tipa *Extension* iz paketa *org.bouncycastle.asn1.x509* na osnovu prosleđenih parametara.

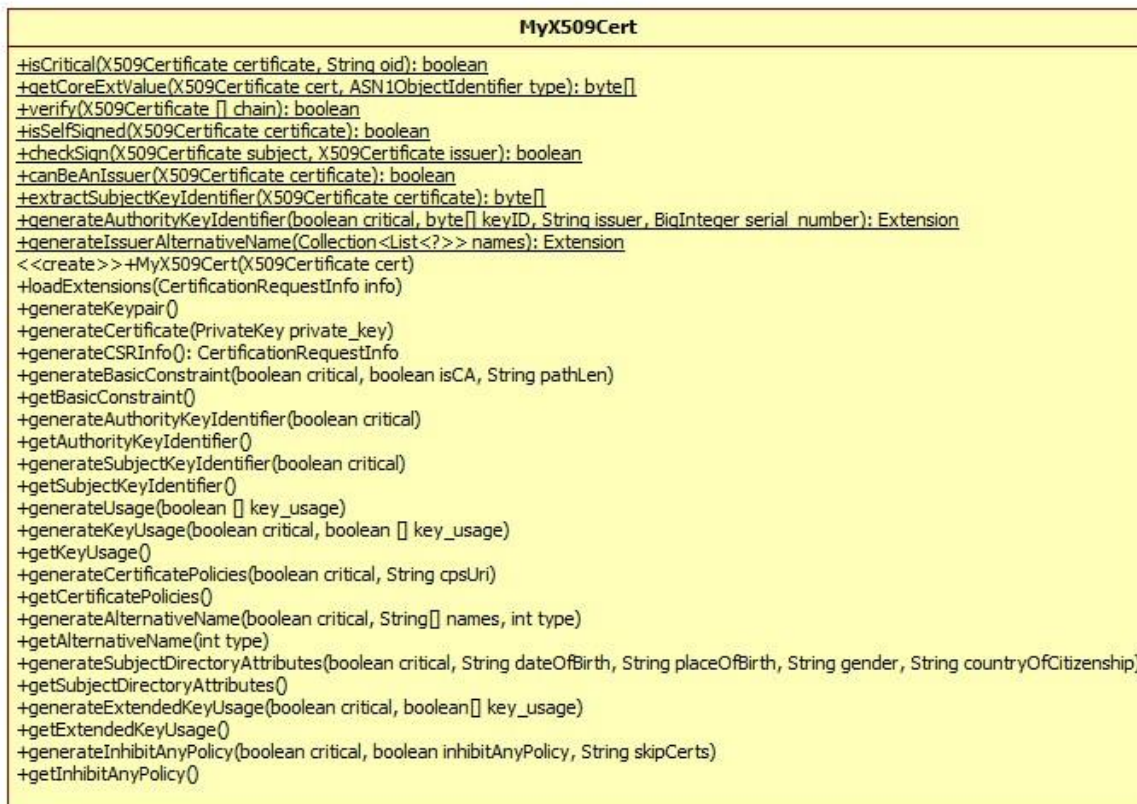


(Slika 9) UML prikaz polja klase *MyX509Cert*.

Klasa *MyX509Cert* (slike 9 i 10) zadužena je za generisanje parova ključeva specificiranim algoritmom, generisanje i potpisivanje sertifikata verzija 1 i 3 i izvlačenje vrednosti ekstenzija sertifikata verzije 3. Metode klase *MyX509Cert* koriste tipove podataka iz *Bouncy Castle* API-a i *java.security* paketa. Od najveće važnosti su polja *certificate* tipa *X509Certificate* iz *org.bouncycastle.cert* paketa i *keypair* tipa *KeyPair* iz *java.security* paketa. Polje *certificate* čuva referencu na objekat klase *X509Certificate* i predstavlja primerak X.509 digitalnog sertifikata. Polje *keypair* sadrži jedan par privatnog i javnog ključa generisan jednim od podržanih algoritama.

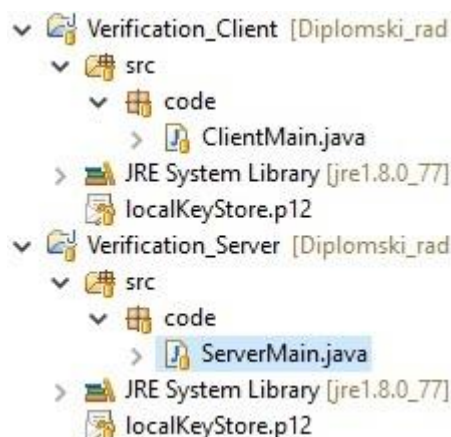
- Metoda *isCritical(X509Certificate certificate, String oid)* proverava da li je ekstenzija čiji je *Object Identifier* prosleđen parametrom *oid* označena kao *critical* u sertifikatu *certificate*. Povratna vrednost je *true* u slučaju da je ekstenzija *critical*.

- Metoda *getCoreExtValue(X509Certificate cert, ASN1ObjectIdentifier type)* vraća niz bajtova koji predstavlja vrednost ekstenzije označene *type OID*-om u sertifikatu *cert*. Vraća *null* u slučaju da *cert* ne poseduje tu ekstenziju.
- Metoda *verify(X509Certificate [] chain)* verifikuje lanac X.509 sertifikata *chain*. Vraća *true* u slučaju da je validan, *false* u slučaju da nije validan.
- Metoda *isSelfSigned(X509Certificate certificate)* vraća *true* u slučaju da je X.509 sertifikat *certificate* potpisao sam sebe, *false* u slučaju da nije.
- Metoda *checkSign(X509Certificate subject, X509Certificate issuer)* vraća *true* u slučaju da je X.509 sertifikat *issuer* potpisao X.509 sertifikat *subject*, *false* u slučaju da nije.
- Metoda *canBeAnIssuer(X509Certificate certificate)* vraća *true* u slučaju da X.509 sertifikat *certificate* može da potpisuje druge sertifikate, *false* u slučaju da ne može.
- Metoda *extractSubjectKeyIdentifier(X509Certificate certificate)* vraća identifikator ključa vlasnika X.509 sertifikata *certificate* u obliku niza bajtova, *null* u slučaju da *certificate* ne poseduje tu ekstenziju.
- Metoda *generateAuthorityKeyIdentifier(boolean critical, byte[] keyID, String issuer, BigInteger serial_number)* kreira i vraća objekat tipa *Extension* iz paketa *org.bouncycastle.asn1.x509* na osnovu prosleđenih parametara.
- Konstruktor *MyX509Cert(X509Certificate certificate)* učitava vrednosti polja kreirajućeg objekta iz *certificate* parametra.
- Metoda *loadExtensions(CertificationRequestInfo info)* učitava ekstenzije iz prosleđenog parametra *info* u polje *extensions*.
- Metoda *generateKeypair()* generiše novi par ključeva pomoću klase *KeyPairGenerator* iz paketa *java.security* algoritmom čija se oznaka nalazi u polju *algorithm*. Par ključeva se čuva u polju *keypair*.
- Metoda *generateCertificate(PrivateKey private_key)* kreira novi X.509 digitalni sertifikat verzije 1 ili 3, potpisuje ga privatnim ključem *private_key* algoritmom čija se oznaka nalazi u polju *signature_algorithm* i čuva ga u polju *certificate*.
- Metoda *generateCSRInfo()* kreira novi objekat tipa *CertificationRequestInfo* na osnovu sertifikata na koji ukazuje polje *certificate*.
- Metode *generateExtensionName(...)* na osnovu prosleđenih parametara generišu nove objekte tipa *Extension* iz paketa *org.bouncycastle.asn1.x509* i pamte ih u odgovarajućim elementima niza *extensions*.
- Metode *getExtensionName()* iz X.509 sertifikata u polju *certificate* izvlače vrednosti ekstenzija i generišu nove objekte tipa *Extension* iz paketa *org.bouncycastle.asn1.x509* i pamte ih u odgovarajućim elementima niza *extensions*.



(Slika 10) UML prikaz metoda klase MyX509Cert.

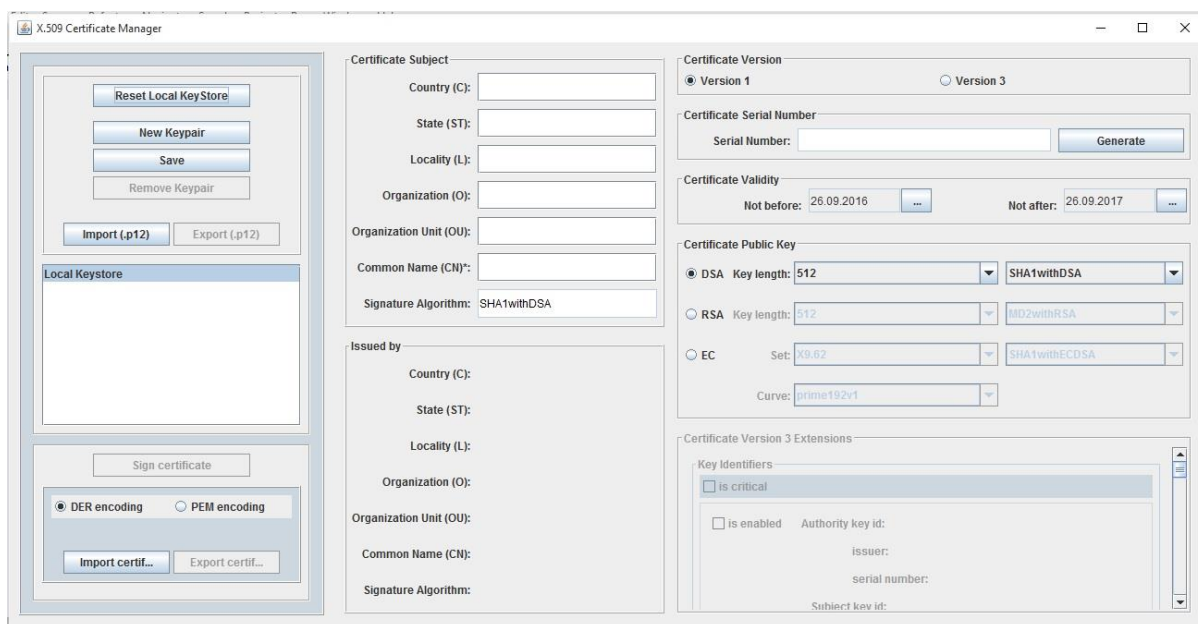
Drugi deo sistema, serverska i klijentska aplikacija u svom sastavu imaju po jednu klasu (*ServerMain* i *ClientMain*) koje predstavljaju ulazne tačke programa (slika 11). Upotrebom klasa iz paketa *javax.net.ssl* [21] omogućeno je uspostavljanje bezbedne konekcije između serverske i klijentske aplikacije, korišćenjem sertifikata generisanih aplikacijom za kreiranje i potpisivanje X.509 sertifikata. Interakcija korisnika sa ovim aplikacijama obavlja se preko konzole i komandne linije. Način na koji se prethodno opisani delovi sistema mogu zajedno upotrebiti biće predstavljen u poglavlju gde je dat primer korišćenja realizovanog sistema.



(Slika 11) Izgled strukture serverske i klijentske aplikacije.

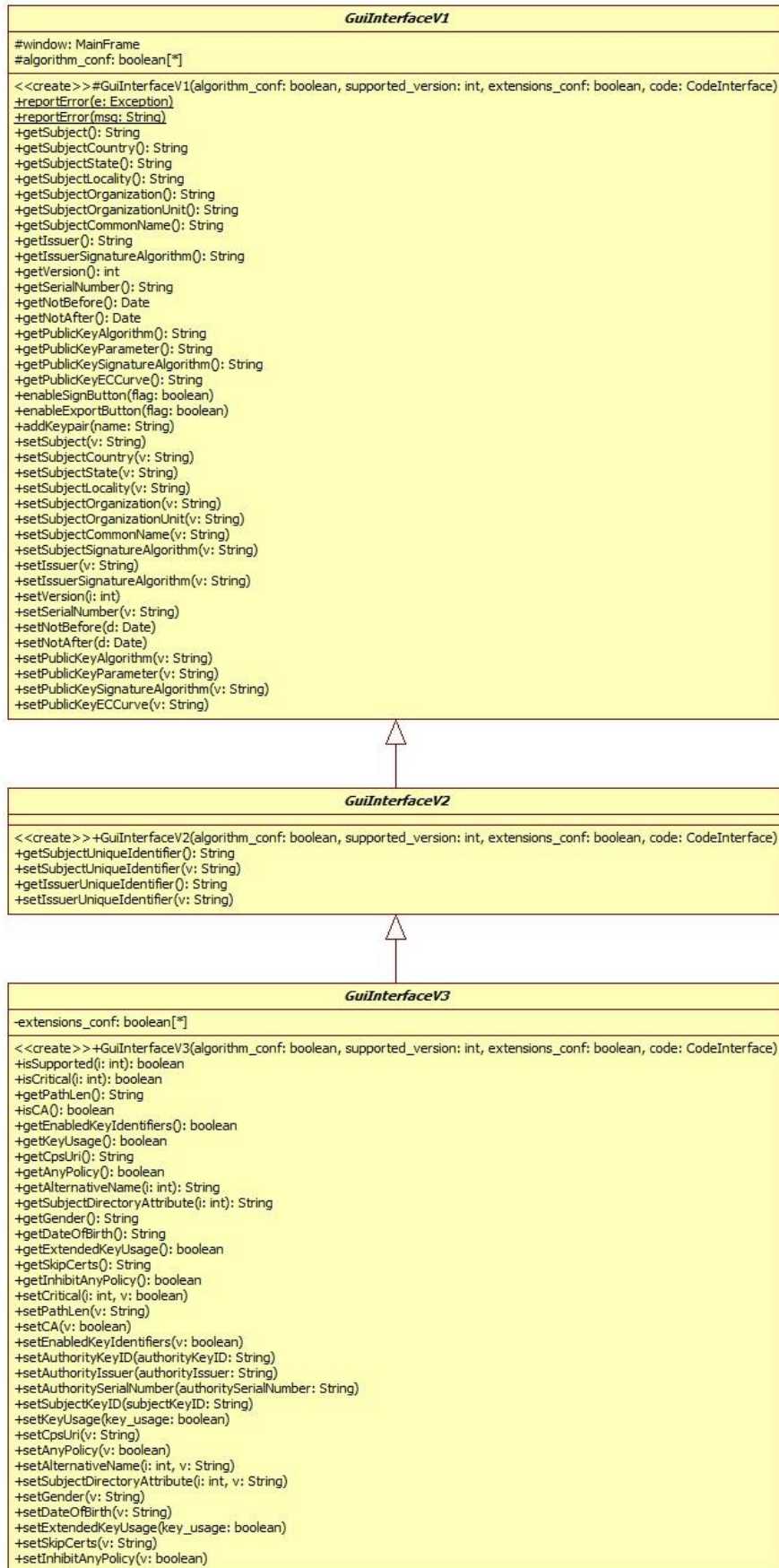
3.4. DIZAJN GRAFIČKOG KORISNIČKOG INTERFEJSA

Najveći izazov prilikom dizajniranja aplikacije za generisanje i potpisivanje X.509 digitalnih sertifikata bila je realizacija grafičkog korisničkog interfejsa (slika 12) koji je potpuno nezavisan od implementacije samih funkcionalnosti aplikacije. Takođe, obzirom da je plan da se implementacija logike aplikacije prepusti studentima, bilo je neophodno napraviti GUI otporan na greške prilikom rada sa nekorektnim podacima. Pored toga, omogućeno je postizanje različitih nivoa složenosti i broja zahteva koje bi studenti imali zadatak da implementiraju, tako što je realizovani GUI moguće konfigurisati po želji. Na primer, zadatak bi mogao biti da se implementira samo 1 ili 2 od GUI-om podržana 3 algoritma za generisanje parova privatnih i javnih ključeva. Isto tako, umesto implementacije svih 10 GUI-om podržanih ekstenzija X.509 v3 sertifikata, moguće je konfigurisati GUI tako da generisani sertifikati imaju samo neke od podržanih ekstenzija. Način na koji je moguće konfigurisati GUI je objašnjen u prethodnom potpoglavlju, učitavanjem konfiguracionih parametara iz fajla čiji je naziv prosleđen preko argumenata komandne linije. Struktura tog fajla biće opisana u poglavlju gde je ilustrovan primer korišćenja sistema.



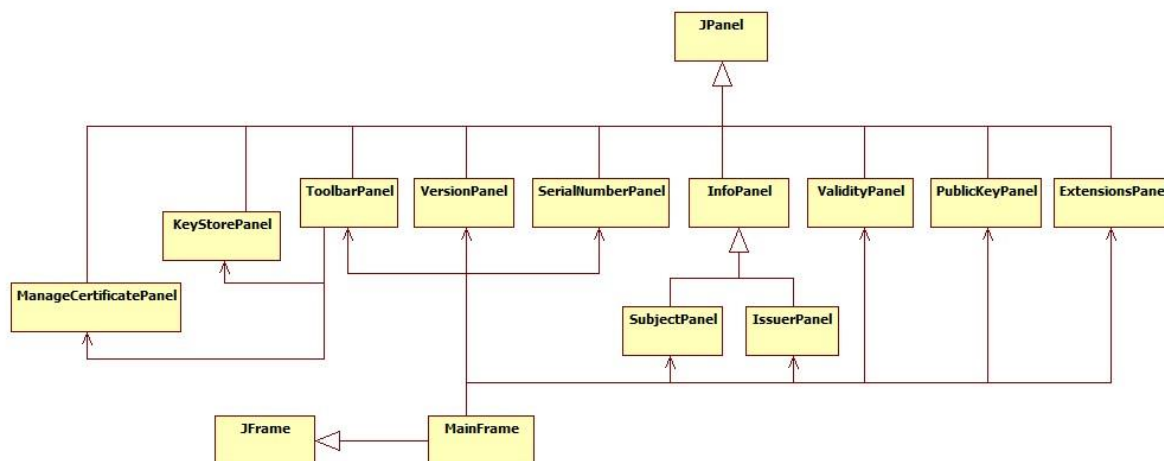
(Slika 12) Izgled aplikacije pri pokretanju.

Sav izvorni kod vezan za implementaciju GUI-a nalazi se u **gui** paketu. Paket *gui* je potpuno zatvoren za pristup osim javno dostupnih konstantnih vrednosti koje se nalaze u okviru klase *Constants* i preko javnih metoda apstraktnih klasa *GuiInterfaceV1*, *GuiInterfaceV2* i *GuiInterfaceV3*. Time se ograničava skup operacija koje studenti mogu da izvedu sa grafičkim korisničkim interfejsom prilikom implementiranja logike aplikacije. Dijagram odnosa te tri klase dat je na sledećoj stranici (slika 13).

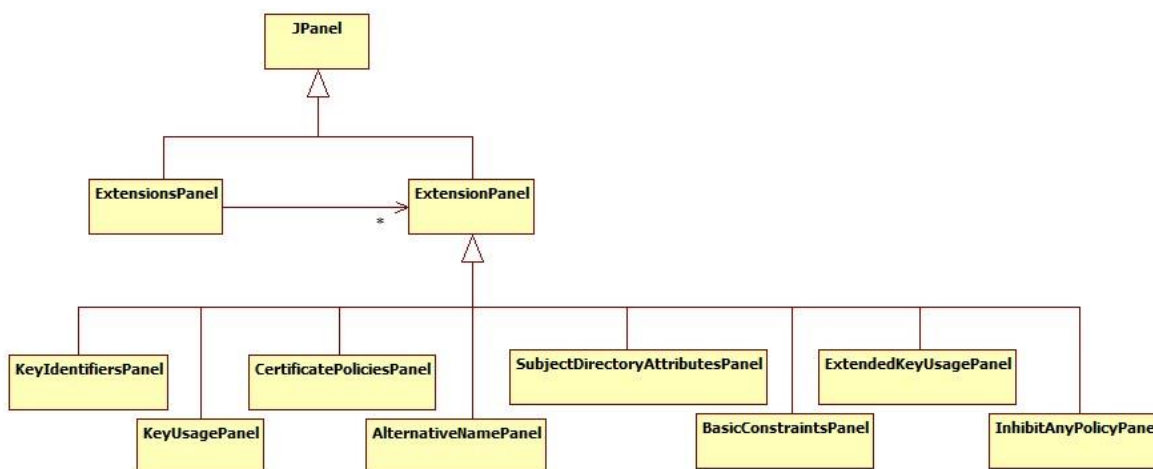


(Slika 13) UML prikaz odnosa klasa GuiInterfaceVX.

Na prethodnom dijagramu se može videti da se kao parametri konstruktora prosleđuju (pored konfiguracionih nizova za algoritme generisanja parova ključeva i ekstenzija) jedna celobrojna vrednost *supported_version* koja označava sa kojim sve verzijama sertifikata GUI može da radi (0 za verziju 1, 1 za verziju 2 i 2 za verziju 3) i referenca na objekat tipa *CodeInterface* čije metode se pozivaju prilikom klika na neko dugme ili odabira sertifikata iz lokalnog *keystore*-a na GUI-u. Klasa *GuiInterfaceVI* sadrži zaštićeno polje tipa *MainFrame*. *MainFrame* je korisnička klasa koja proširuje klasu *JFrame* iz *javax.swing* [22] paketa i predstavlja prozor koji se iscrtava na ekranu korisnika po pokretanju aplikacije. Prozor aplikacije je podeljen na 7 manjih panela, 8 u slučaju da je podržana verzija 3 X.509 sertifikata (slika 14). Svaki od tih panela predstavljen je posebnom klasom koja proširuje *JPanel* klasu iz *javax.swing* paketa. Novi objekti tih klasa kreiraju se u konstrukturu *MainFrame*-a, a tada se učitava sadržaj lokalnog *keystore*-a pozivom metode *loadLocalKeystore()* *CodeInterface*-a.

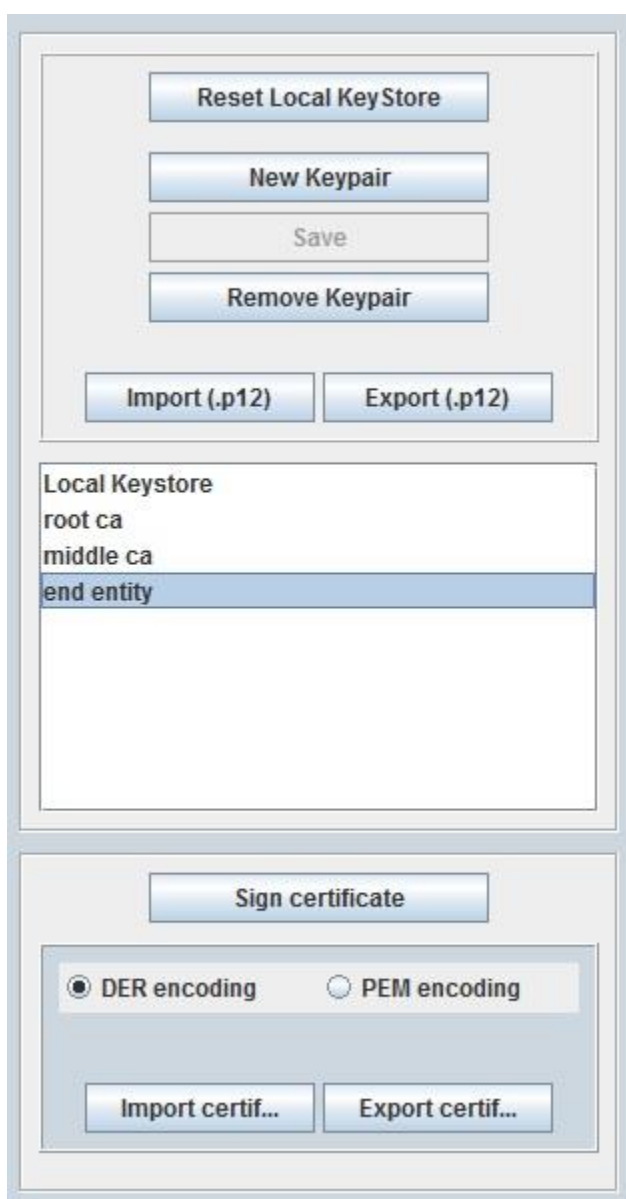


(Slika 14) Deo UML dijagrama klasa gui paketa.



(Slika 15) Deo UML dijagrama klasa gui paketa koji se odnosi na ekstenzije.

ExtensionsPanel se opciono iscertava na prozoru aplikacije, ako parametar *supported_version* prosleđen konstruktoru *MainFrame*-a ima vrednost veću od 1. *ExtensionsPanel* se sastoji iz više manjih panela od kojih je svaki predstavljen tipom *ExtensionPanel* (slika 15). Tačan broj tih manjih panela određen je konfiguracionim parametrom *extensions_conf* i predstavlja broj aktiviranih ekstenzija. Usvojena je ovakva realizacija dela GUI-a vezanog za ekstenzije da bi se kasnije lako mogle dodavati ekstenzije koje u ovoj verziji sistema nisu implementirane. Nova ekstenzija bi se dodala kreiranjem nove klase koja proširuje klasu *ExtensionPanel* i implementiranjem nasleđenih apstraktnih metoda.



(Slika 16) Izgled Toolbar panela.

Po pokretanju aplikacije na levoj strani prozora se nalazi *ToolbarPanel* (slika 16). Na njemu su postavljeni dugmići pomoću kojih se može upravljati sadržajem lokalnog *keystore*-a, kreirati i uvoziti/izvoziti parovi ključeva, potpisivati i uvoziti/izvoziti sertifikati i prikazivati detalji sertifikata na GUI-u, tako što se željeni sertifikat selektuje u listi koja oslikava sadržaj lokalnog *keystore*-a. Metode *CodeInterface*-a se zapravo pozivaju nakon što se proizvede određena akcija na *ToolbarPanel*-u. Klasa *ToolbarListener* (slika 17) koja implementira Javine *ActionListener* i *ListSelectionListener* interfejse osluškuje događaje na *ToolbarPanel*-u i poziva tačno definisanu metodu *CodeInterface*-a u zavisnosti od pritisnutog dugmeta ili selektovanja elementa liste. Pritom se obavljaju sve provere korektnosti stanja grafičkog korisničkog interfejsa. U slučaju neke greške korisnik se obaveštava prikazivanjem pop-up prozora sa informacijom o tipu greške.

```

20+ ToolbarListener(MainFrame mainFrame, CodeInterface code) {
24
25+ @Override
26 public void actionPerformed(ActionEvent event) {
27     switch (event.getActionCommand()) {
28         case "Reset Local KeyStore": resetLocalKeystorePerformed(); break;
29         case "New Keypair": newKeypairPerformed(); break;
30         case "Save": saveKeypairPerformed(); break;
31         case "Remove Keypair": removeKeypairPerformed(); break;
32         case "Import (.p12)": importKeypairPerformed(); break;
33         case "Export (.p12)": exportKeypairPerformed(); break;
34         case "Sign certificate": signCertificatePerformed(); break;
35         case "Import certificate": importCertificatePerformed(); break;
36         case "Export certificate": exportCertificatePerformed(); break;
37     }
38 }
39
41+ public void valueChanged(ListSelectionEvent ev) {

```

(Slika 17) Deo izvornog koda klase *ToolbarListener*.

Na prethodnoj slici je dat isečak koda klase *ToolbarListener* da bi se dobio bolji uvid u to šta se dešava pritiskom na neko dugme na *ToolbarPanel*-u ili odabirom nekog sertifikata iz liste. Klik na dugme generiše akciju koju osluškuje *ToolbarListener* i pokreće metodu *actionPerformed()*. Analogno tome, selektovanje nekog sertifikata u listi generiše događaj koji pokreće metodu *valueChanged()* *ToolbarListener*-a. Tabela poziva metoda *CodeInterface*-a iz metoda *ToolbarListener*-a data je ispod (tabela 3):

metoda <i>ToolbarListener</i> -a	metoda <i>CodeInterface</i> -a
<i>resetLocalKeystorePerformed()</i>	<i>resetLocalKeystore()</i>
<i>newKeypairPerformed()</i>	/
<i>saveKeypairPerformed()</i>	<i>saveKeypair(...)</i>
<i>removeKeypairPerformed()</i>	<i>removeKeypair(...)</i>
<i>importKeypairPerformed()</i>	<i>importKeypair(...)</i>
<i>exportKeypairPerformed()</i>	<i>exportKeypair(...)</i>
<i>signCertificatePerformed()</i>	<i>generateCSR(...)</i>
<i>importCertificatePerformed()</i>	<i>importCertificate(...)</i>
<i>exportCertificatePerformed()</i>	<i>exportCertificate(...)</i>
<i>valueChanged(ListSelectionEvent ev)</i>	<i>loadKeypair(...)</i>

(Tabela 3) Tabela poziva metoda *CodeInterface*-a iz metoda *ToolbarListener*-a.

Aktivnosti koje se odvijaju kada se pritisne dugme *Sign certificate* su nešto složenije. Nakon što se pozove metoda *generateCSR(...)* *CodeInterface*-a koja generiše novi objekat tipa *CertificationRequestInfo*, kreira se novi pop up prozor (slika 18) predstavljen klasom *SignRequestDialog*. Klasa *SignRequestDialog* implementira Javin interfejs *ActionListener* i obrađuje sve događaje generisane klikom na taj prozor. Za popunjavanje *dropdown* liste CA koristi se rezultat metode *getIssuers(...)* *CodeInterface*-a. Nakon što se odabere neki od CA iz *dropdown* liste, pozivaju se metode *getIssuer(...)* i *getIssuerPublicKeyAlgorithm(...)* za prikaz informacija o CA i popunjavanje *dropdown* liste algoritama za potpisivanje sertifikata, respektivno. Pritiskom na dugme *Sign* poziva se metoda *signCertificate(...)* *CodeInterface*-a.

(Slika 18) Izgled *SignRequestDialog* pop up prozora

3.5. UOČENI PROBLEMI

Prilikom implementacije sistema jedan od uočenih problema je taj da određene kombinacije algoritama u procesu potpisivanja sertifikata ne daju validne lance sertifikata. Međutim, to je posledica nedostataka biblioteka i paketa koji se koriste za rad sa kriptografskim algoritmima i sertifikatima.

X.509 standard nalaže da vrednost polja *serial_number* bude jedinstvena na nivou CA koji izdaje sertifikate. Kako je u ovom sistemu moguće simulirati više CA, bilo bi previše složeno

ispuniti takav zahtev. Zato je usvojeno da je vrednost polja *serial_number* jedinstvena na nivou svih CA, pod pretpostavkom da je malo verovatno da se generiše dovoljno veliki broj sertifikata da bi se počele ponavljati vrednosti koje proizvodi generator random brojeva.

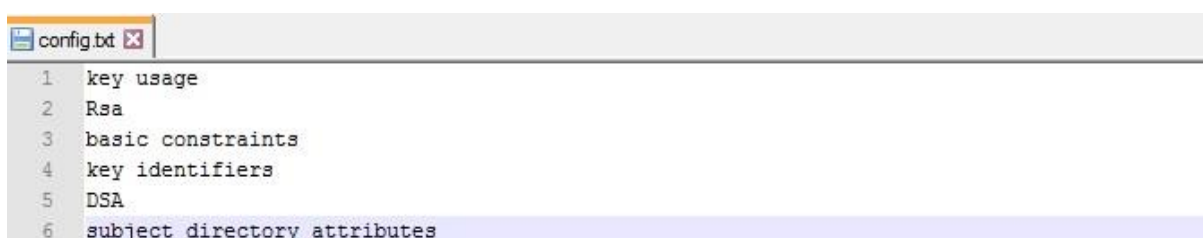
U implementaciji *policy constraints* ekstenzije X.509 sertifikata omogućeno je dodavanje samo *anyPolicy* politike izdavanja sertifikata. Razlog za to je što ne postoje neke predefinisane politike i odgovarajući *OIDs*, već je potrebno registrovati politike izdavanja kod međunarodno priznate organizacije za standarde.

4. PRIMER KORIŠĆENJA SISTEMA

Sistem je zamišljen tako da mogu da ga koriste 3 grupe korisnika. Prva grupa korisnika su ovlašćena lica, npr. profesor, sa pristupom izvornom kodu sistema i pravom izmene koda. Druga grupa korisnika su studenti - lica sa pravom uvida u deo izvornog koda programa i bez prava izmene koda. Treća grupa korisnika su lica koja pokreću izvršni fajl koji je nastao prevođenjem izvornog koda. U treću grupu spadaju i profesor i studenti. U potpoglavljima su opisani mogući slučajevi korišćenja sistema od strane svake grupe korisnika, a u poslednjem potpoglavlju je dat primer korišćenja aplikacije za verifikaciju digitalnih sertifikata.

4.1. PROFESOR

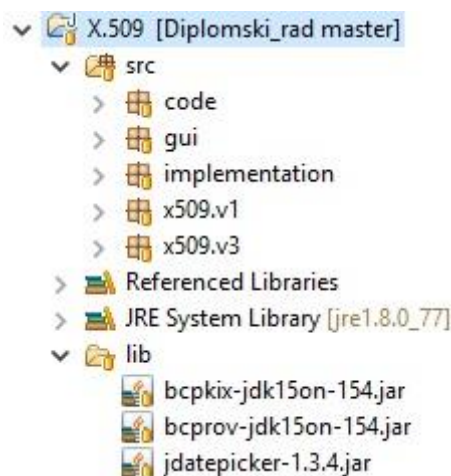
Cilj ovog korisnika je da studentima isporuči .jar biblioteku sa implementiranim grafičkim korisničkim interfejsom i javnim metodama za čitanje i prikaz podataka na njemu. U biblioteci se nalazi i interfejs sa potpisima metoda koje studenti treba da implementiraju. Ulazna tačka aplikacije se takođe nalazi u biblioteci, u fajlu *X509.java*. Prvi korak je kreiranje fajla koji služi za inicijalizaciju konfiguracionih nizova *algorithm_conf* i *extensions_conf* u klasi *X509*. Ranije je pomenuto da se vrednosti tih nizova koriste za podešavanje željenog izgleda grafičkog korisničkog interfejsa. Postavljanjem *true* ili *false* vrednosti u elemente niza *algorithm_conf* konfiguriše se GUI da obezbedi/ne obezbedi podršku za odgovarajući algoritam za generisanje parova ključeva. Postavljanjem *true* ili *false* vrednosti u elemente niza *extensions_conf* konfiguriše se GUI da obezbedi/ne obezbedi podršku za odgovarajuće ekstenzije, u slučaju da željeni GUI treba da omogući generisanje X.509 sertifikata verzije 3. Struktura konfiguracionog fajla je jednostavna (slika 19). Potrebno je nabrojati koje algoritme i ekstenzije željeni GUI treba da podrži, razdvajati ih novim redom. Vrednosti parametara su: *DSA*, *RSA* i *EC* za algoritme; *key identifiers*, *key usage*, *certificate policies*, *subject alternative name*, *issuer alternative name*, *subject directory attributes*, *basic constraints*, *extended key usage* i *inhibit any policy* za ekstenzije. Prilikom učitavanja fajla, aplikacija ne pravi razliku između malih i velikih slova. Takođe nije bitan ni redosled navođenja.



```
1 key usage
2 Rsa
3 basic constraints
4 key identifiers
5 DSA
6 subject directory attributes
```

(Slika 19) Primer konfiguracionog fajla.

Nakon konfiguracije GUI-a, drugi korak je export izvornog koda (slika 20) u .jar biblioteku. U biblioteku je potrebno exportovati izvorni kod iz *gui*, *code* i jednog od *x509.v1*/*x509.v3* paketa. Primera radi, biće izabran *x509.v3* paket. Pored .jar biblioteke studentima je potrebno isporučiti i *jdatepicker-1.3.4.jar* fajl.



(Slika 20) *Raspored izvornog koda po paketima.*

4.2. STUDENT

Student na raspolaganju ima sledeća dva fajla (slika 21), pomoću kojih treba da realizuje aplikaciju za generisanje i potpisivanje X.509 digitalnih sertifikata:

 jdatepicker-1.3.4.jar	Executable Jar File	42 KB
 library.jar	Executable Jar File	105 KB

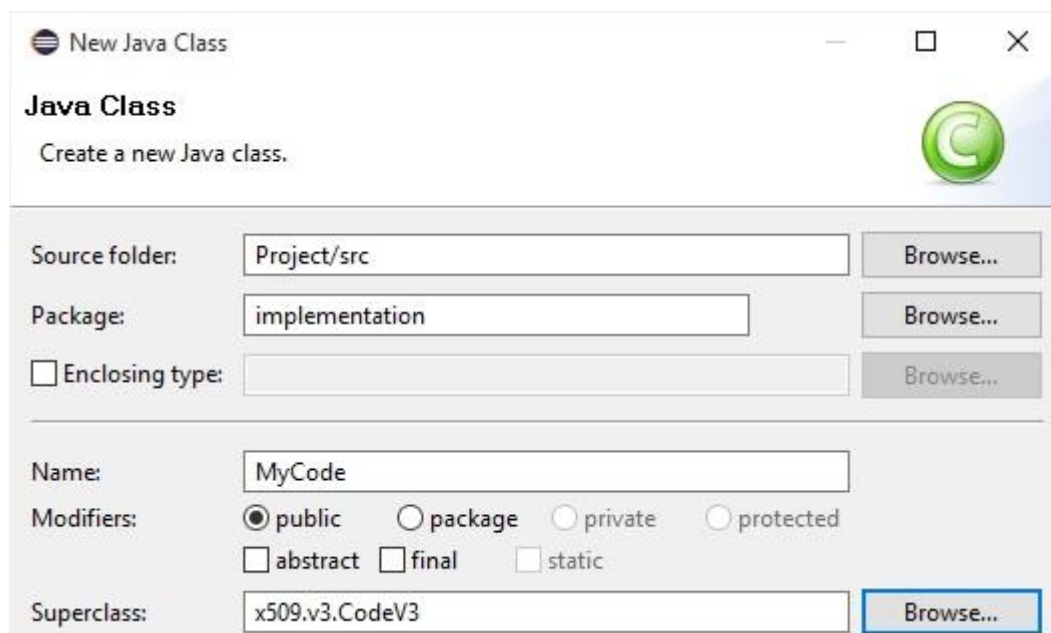
(Slika 21) *Biblioteke koje su na raspolaganju studentima.*

Nakon kreiranja novog Java projekta u Eclipse okruženju, potrebno je importovati date biblioteke. To se radi podešavanjem Java *build path*-a u konfiguraciji projekta (slika 22).



(Slika 22) *Podešavanje Java build path-a.*

Klasa *X509* sa *main* funkcijom koja predstavlja ulaznu tačku programa nalazi se u biblioteci *library.jar*. Da bi se dobila funkcionalna aplikacija potrebno je napraviti paket *implementation* i u njemu klasu *MyCode* (slika 23) koja proširuje klasu *CodeV3* iz paketa *x509.v3* iz biblioteke *library.jar*. Implementacijom nasleđenih metoda ispunjava se zadatak.



(Slika 23) Izvođenje klase *MyCode* iz klase *x509.v3.CodeV3*.

4.3. KORISNIK

Nakon što je implementirana aplikacija za generisanje i potpisivanje X.509 digitalnih sertifikata verzije 3 moguće je pokrenuti je. U ovom primeru simulirano je izdavanje sertifikata serveru od strane CA. Prvo se kreiraju sertifikati *root CA* (slika 24) i *intermediate CA*. To su obavezno sertifikati verzije 3 koji sadrže *critical key usage* ekstenziju sa setovanim *keyCertSign* bitom i *critical basic constraints* ekstenziju sa setovanim *CA* flagom. Ovo je neophodno da bi mogli da potpisuju druge sertifikate, a vrednosti i prisustvo ostalih ekstenzija su opcioni. Zatim *root CA* sertifikat potpisuje *intermediate CA* sertifikat (slika 25). Prilikom potpisivanja bira se jedna od mogućih kombinacija *hash* algoritama sa algoritmom kojim je generisan par ključeva *root CA* sertifikata. Potpisivanjem *intermediate CA* *root CA* sertifikatom uspostavlja se hijerarhija CA na čijem vrhu se nalazi *root CA*. Nakon toga se kreira *end entity* sertifikat (slika 26) koji će se naći na kraju tog lanca sertifikata i predstavlja sertifikat koji se izdaje serveru. Jako je bitno da, ako *end entity* sertifikat sadrži ekstenziju *key usage*, u njoj ima setovane bit *digitalSignature* i *keyAgreement* bit. Samo tako se može upotrebiti za autentikaciju u SSL konekciji. *Intermediate CA* sertifikat zatim potpisuje serverski *end entity* sertifikat i tako on postaje validan (slika 27).

Certificate Subject

Country (C): Serbia

State (ST): Serbia

Locality (L):

Organization (O): ETF

Organization Unit (OU): SI

Common Name (CN)*: ETF root CA

Signature Algorithm: SHA256withRSA

Certificate Version

☐ Version 1 ☒ Version 3

Certificate Serial Number

Serial Number: 16054159827231333458 Generate

Certificate Validity

Not before: 24.09.2012 ... Not after: 30.09.2016 ...

Certificate Public Key

☐ DSA Key length: 512 SHA1withDSA

☒ RSA Key length: 2048 SHA256withRSA

☐ X9.62 SHA1withECDSA

☐ prime192v1

3 Extensions

☒ is critical

☐ Digital Signature ☐ Content Commitment ☐ Key Encipherment

☐ Data Encipherment ☐ Key Agreement ☒ Certificate Signing

☐ CRL Signing ☐ Encipher Only ☐ Decipher Only

Certificate policies

Issued by

Country (C):

State (ST):

Locality (L):

Organization (O):

Organization Unit (OU):

Common Name (CN):

Signature Algorithm:

Input

? Name: ETF root CA

OK Cancel

(Slika 24) Kreiranje root CA sertifikata. Na isti naćin se kreira i intermediate CA sertifikat.

Certificate Signing Request

Info

Version 3

Serial number: 14927439671887620885

Not before: 24.09.2012

Not after: 30.09.2016

Choose issuer

Choose a CA: etf root ca

SHA256withRSA

Sign

Certificate Subject

Country (C): Serbia

State (ST): Serbia

Locality (L):

Organization (O): ETF

Organization Unit (OU): SI

Common Name (CN)*: ETF intermediate CA

Signature Algorithm: SHA512withRSA

Issued by

Country (C): Serbia

State (ST): Serbia

Locality (L):

Organization (O): ETF

Organization Unit (OU): SI

Common Name (CN): ETF root CA

Signature Algorithm: SHA256withRSA

(Slika 25) Potpisivanje intermediate CA sertifikata root CA sertifikatom.

Certificate Subject

Country (C): Serbia

State (ST): Serbia

Locality (L): Belgrade

Organization (O):

Organization Unit (OU):

Common Name (CN)*: 127.0.0.1

Signature Algorithm: SHA256withECDSA

Certificate Version

☐ Version 1 ☒ Version 3

Certificate Serial Number

Serial Number: 6016744353818500496 Generate

Certificate Validity

Not before: 24.09.2012 ... Not after: 30.09.2016 ...

Certificate Public Key

☐ DSA Key length: 512 SHA1withDSA

☒ RSA Key length: 512 MD2withRSA

Set: SEC SHA256withECDSA

Curve: secp256k1

Certificate Version 3 Extensions

Key usage

☒ is critical

☒ Digital Signature ☐ Content Commitment ☐ Key Encipherment

☐ Data Encipherment ☒ Key Agreement ☐ Certificate Signing

☐ CRL Signing ☐ Encipher Only ☐ Decipher Only

Issued by

Country (C): ?

State (ST):

Locality (L):

Organization (O):

Organization Unit (OU):

Common Name (CN):

Signature Algorithm:

Input

Name: server certificate

OK Cancel

(Slika 26) Kreiranje end entity serverskog sertifikata.

Certificate Signing Request

Info

Version 3

Serial number: 6016744353818500496

Not before: 24.09.2012

Not after: 30.09.2016

Choose issuer

Choose a CA: etf intermediate ca

SHA256withRSA

Sign

Certificate Subject

Country (C): Serbia

State (ST): Serbia

Locality (L): Belgrade

Organization (O):

Organization Unit (OU):

Common Name (CN)*: 127.0.0.1

Signature Algorithm: SHA256withECDSA

Issued by

Country (C): Serbia

State (ST): Serbia

Locality (L):

Organization (O): ETF

Organization Unit (OU): SI

Common Name (CN): ETF intermediate CA

Signature Algorithm: SHA256withRSA

(Slika 27) Potpisivanje end entity serverskog sertifikata intermediate CA sertifikatom.

4.4. VERIFIKACIJA GENERISANIH SERTIFIKATA


U prethodnom potpoglavlju simulirano je izdavanje X.509 sertifikata serveru. U ovom primeru taj sertifikat će se iskoristiti za uspostavljanje sigurne SSL konekcije servera sa klijentom i autentikaciju servera. To je moguće pomoću realizovane aplikacije za verifikaciju digitalnih sertifikata. Serverska aplikacija po pokretanju osluškuje na portu 9999 da li neki klijent pokušava da se poveže, a klijentska aplikacija zahteva povezivanje sa serverom na *localhost*-u preko istog porta. Preko parametara komandne linije potrebno je podesiti *keystore* serverske aplikacije u kom treba da se nalazi samo sertifikat čiji je vlasnik serverska aplikacija, *end entity* serverski sertifikat, i koji će biti poslat klijentu prilikom *handshake* protokola (slika 28). Taj *keystore* treba da se nalazi u *root* direktorijumu projekta.



```
-Djavax.net.ssl.keyStore=localKeyStore.p12 -  
Djavax.net.ssl.keyStorePassword=localKeyStorePassword -  
Djava.protocol.handler.pkgs=com.sun.net.ssl.internal.www.protocol -Djavax.net.debug=ssl
```

(Slika 28) Argumenti komandne linije za pokretanje *ServerMain*-a.

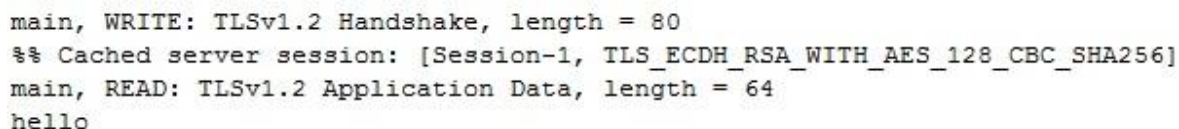
Preko parametara komandne linije potrebno je podesiti *trustStore* klijentske aplikacije u kome se nalaze sertifikati kojima klijent veruje. Potrebno je da se tu nalaze odgovarajući sertifikati koji služe za verifikaciju serverskog sertifikata da bi *handshake* protokol bio uspešan (slika 29). U konkretnom slučaju - *root CA* sertifikat. Taj *trustStore* treba da se nalazi u *root* direktorijumu projekta.



```
-Djavax.net.ssl.trustStore=localKeyStore.p12 -  
Djavax.net.ssl.trustStorePassword=localKeyStorePassword -  
Djava.protocol.handler.pkgs=com.sun.net.ssl.internal.www.protocol -Djavax.net.debug=ssl
```

(Slika 29) Argumenti komandne linije za pokretanje *ClientMain*-a.

Nakon pokretanja *ServerMain*-a i *ClientMain*-a moguće je poslati niz karaktera sa konzole klijentske aplikacije serveru, i taj niz se prikazuje na konzoli serverske aplikacije, uz *debug* informacije (slika 30).



```
main, WRITE: TLSv1.2 Handshake, length = 80  
%% Cached server session: [Session-1, TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256]  
main, READ: TLSv1.2 Application Data, length = 64  
hello
```

(Slika 30) Izgled konzole serverske aplikacije nakon primanja niza znakova "hello".

5. ZAKLJUČAK

Jasno je da su digitalni sertifikati od velike važnosti, obzirom da nalaze svoju primenu u mnoštvu protokola za komunikaciju, enkripciji podataka, čip karticama, itd. Zbog toga je veoma važno što bolje približiti ovaj pojam osobama koje su nove u oblasti zaštite informacija. Na predmetu Zaštita podataka na Elektrotehničkom fakultetu u Beogradu deo gradiva koje se obrađuje odnosi se na digitalne sertifikate, preciznije sertifikate po X.509 standardu. Pored teorijske osnove, postoji i projektni zadatak čija izrada podrazumeva implementaciju aplikacije sa grafičkim korisničkim interfejsom za generisanje i potpisivanje X.509 sertifikata. Cilj ovog dokumenta je da na jednom mestu pruži sve potrebne informacije o samoj strukturi X.509 sertifikata i načinu njihove upotrebe, a zatim pruži opis sistema koji je implementiran u sklopu praktičnog dela ovog rada. Bitan deo rada predstavlja poglavlje koje pruža teorijsku osnovu za razumevanje X.509 digitalnih sertifikata. Njegova važnost ogleda se u tome što može poslužiti i kao literatura, pošto je na ovu temu ima jako malo na srpskom jeziku.

Realizovani sistem opisan u ovom radu je zamišljen tako da studentima približi pojam digitalnih sertifikata prolaskom kroz ceo proces generisanja, potpisivanja i verifikacije X.509 sertifikata, a s druge strane, olakša posao izrade projektnog zadatka na predmetu Zaštita podataka uklanjanjem implementacije grafičkog korisničkog interfejsa iz liste zahteva. Na početku ovog rada su nabrojani glavni ciljevi kojima se težilo prilikom realizacije sistema. Dobijeni sistem ispunjava sve te ciljeve uz sledeća ograničenja:

- Aplikacija ne poseduje funkcionalnosti za rad sa sertifikatima verzije 2. Oni se danas retko upotrebljavaju i usled jako malo informacija i nedostatka literature o sertifikatima tog tipa, nije bilo moguće odrediti koje zahteve je potrebno ispuniti da bi se dobila aplikacija koja podržava X.509 v2 sertifikate. Ipak, grafički korisnički interfejs je dizajniran tako da poseduje mogućnost da prikaže informacije o poljima sertifikata v2 u slučaju eventualnog proširenja sistema za rad sa v2 sertifikatima.
- Sistem ne pruža mogućnost dodavanja *policy mappings*, *name constraints*, *policy constraints*, *CRL distribution points* i *freshhest CRL* ekstenzija, ali je dizajn aplikacije takav da je njihova kasnija implementacija što je moguće više olakšana.

Sistem je dizajniran s namerom da se kasnije lako može nadograditi. Želja autora je da se eventualno omogući dodavanje svih standardnih ekstenzija digitalnom sertifikatu i detaljnija implementacija trenutno podržanih ekstenzija. Još jedna tačka proširenja je implementacija novih algoritama za generisanje parova ključeva.

LITERATURA

Svi izvori koji su korišćeni u toku izrade ovog rada su navedeni ovde redom kojim su referencirani u radu. Za svaki navedeni izvor je priložena Internet adresa. Svim izvorima je poslednji put pristupano u septembru 2016. godine.

[1] Cryptography – History of cryptography, New World Encyclopedia

<http://www.newworldencyclopedia.org/entry/Cryptography>

[2] Threats addressed by Secure Shell, Van Dyke Software

https://www.vandyke.com/solutions/ssh_overview/ssh_overview_threats.html

[3] Digital Certificates What Are They, and What Are They Doing in My Browser? By Judith V. Boettcher and Amanda Powell

<http://www.cren.net/crenca/docs/syllabus.pdf>

[4] ITU Telecommunication Standardization Sector

<http://www.itu.int/en/ITU-T/Pages/default.aspx>

[5] SPKI Certificate Theory

<https://www.ietf.org/rfc/rfc2693.txt>

[6] X.500 data models, Network Security

<http://www.networxsecurity.org/members-area/glossary/x/x500.html>

[7] Introduction to ASN.1, ITU-T

<http://www.itu.int/en/ITU-T/asn1/Pages/introduction.aspx>

[8] Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile

<https://tools.ietf.org/html/rfc5280>

[9] Entrust Technologies White Paper Version 3 X.509 Certificates, Ian Curry

<ftp://ftp.dfn-cert.de/pub/pca/docs/misc/x509v3.pdf>

[10] <http://core0.staticworld.net/images/idge/imported/article/nww/2008/10/02fig04-100277996-orig.jpg>

[11] DER Encoding of ASN.1 Types, Microsoft

[https://msdn.microsoft.com/en-us/library/windows/desktop/bb648640\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/desktop/bb648640(v=vs.85).aspx)

[12] Certificate Policies extension – all you should know (part 1), Vadims Podāns

<https://www.sysadmins.lv/blog-en/certificate-policies-extension-all-you-should-know-part-1.aspx>

[13] PKI Design Considerations: Certificate Revocation and CRL Publishing Strategies

<https://blogs.technet.microsoft.com/xdot509/2012/11/26/pki-design-considerations-certificate-revocation-and-crl-publishing-strategies/>

[14] Public key infrastructure, Wikipedia

https://en.wikipedia.org/wiki/Public_key_infrastructure

[15] https://sites.google.com/site/ddmwsst/_/rsrc/1472874334886/digital-certificates/chain-of-trust.gif

[16] Website of The Legion of the Bouncy Castle

<https://www.bouncycastle.org/>

[17] <https://sourceforge.net/projects/jdatepicker/files/Releases/1.3.x/>

[18] <https://eclipse.org/luna/>

[19] <http://java.com/en/download/>

[20] Package java.security.cert, Oracle Help Center

<https://docs.oracle.com/javase/7/docs/api/java/security/cert/package-summary.html>

[21] Package javax.net.ssl, Oracle Help Center

<https://docs.oracle.com/javase/7/docs/api/javax/net/ssl/package-summary.html>

[22] Package javax.swing, Oracle Help Center

<https://docs.oracle.com/javase/7/docs/api/javax/swing/package-summary.html>