Analiza i primjena X.509 certifikata

Pintarić, Danijel

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
V A R A Ž D I N

Danijel Pintarić

ANALIZA I PRIMJENA X.509 CERTIFIKATA

DIPLOMSKI RAD

Varaždin, 2020.

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
V A R A Ž D I N

Danijel Pintarić
Matični broj: 35918/07–R
Studij: Informacijsko i programsko inženjerstvo

ANALIZA I PRIMJENA X.509 CERTIFIKATA

DIPLOMSKI RAD

Mentor:
Izv. prof. dr. sc. Sandro Gerić

Varaždin, rujan 2020

Danijel Pintarić

Izjava o izvornosti

Izjavljujem da je moj diplomski rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor potvrdio prihvaćanjem odredbi u sustavu FOI-radovi

_____

Sažetak

U teorijskom dijelu obrađeni su X509 certifikati o kojima naslov ovog diplomskog rada govori te glavni pojmovi vezani uz njih (SSL/TLS protokol s naglaskom na suradnju s HTTPS protokol, TLS rukovanje, digitalni potpis, PKI infrastruktura te povijest, struktura i namjena X.509 digitalnih certifikata. U uvodnom dijelu rada navedena je problematika te koja je motivacija korištenja digitalnih certifikata i općenito TLS protokola vezano za sigurnost te gdje se isti koriste. Sljedeći dio je ustvari teoretska priprema i podloga za dobro razumijevanje što se ustvari događa prilikom razmjene certifikata u TLS rukovanju te što se točno događa i te kakvi se mrežni paketi šalju između posrednika u komunikaciji. Nadalje, navedeno je što je PKI infrastruktura te motivacija korištenja, na koji način se koristi i koja je svrha te na koji način se certifikati koriste unutar nje. Nakon toga, ukratko je obrađena povijest verzija X509 certifikata te je se opisana njegova struktura, polja i proširenja sa fokusom na određena polja koja se naročito koriste u praktičnog dijelu. Sama struktura je opisana detaljnije u sklopu X.509 norme te je opisano kako se koristi te koji su formati i pravila.

U praktičnom dijelu napravljena je implementacija u Java programskom okruženju gdje je napravljen lanac certifikata, na način da su certifikati više razine potpisivali certifikate niže razine u lancu certifikata. Za kraj, naučeno i implementirano trebalo je isprobati pa se stoga sama aplikacija postavila na HTTPS port na poslužitelju, a kao serverski certifikat postavljen je izgenerirani certifikat iz praktičnog primjera. Generirani certifikat koji je za krajnjeg korisnika, odnosno ne koristi se za potpisivanje drugih certifikata u lancu, stavio se u *keystore* uz njegov pripadni privatni ključ te taj *keystore* postavio na poslužitelju u Tomcat postavkama kako bi se osigurao sigurnosni kanal na lokalnoj domeni na kojoj se nalazi praktični primjer diplomskog rada.

Ključne riječi: HTTPS; TLS; TLS rukovanje; Digitalni potpis; PKI infrastruktura, X509 norma, digitalni certifikati, lanac certifikata, Java BouncyCastle kriptografski API

Sadržaj

1. Uvod

Tema ovog diplomskog rada jest Analiza i primjena X.509 certifikata koji su važan dio sigurnosti web aplikacija koje su bazirane na HTTPS protokolu te koriste TLS protokol koji osigurava da su informacije između posrednika u komunikaciji kriptirane. Tema je od velikog značaja što se tiče sigurnosnog aspekta jer se certifikati između ostalog koriste za potvrđivanje vlastitog identiteta ili pristupa nekim informacijama te servisima kao što je slučaj kod elektroničkih plaćanja, Internet bankarstva, e-pošte itd. Zadnjom rečenicom rečeno je da se da se koristi TLS protokol i u servisima koji ne koriste i HTTPS protokol već i neke druge protokole aplikacijskog sloja kao što su protokoli za e-poštu, VoIP telefoniju, prijenos podataka udaljenim pristupom itd.

Danas, pogotovo u doba korona krize, sve više ljudi izvršava transakcije putem interneta te tvrtke sve više koriste e-poštu i razne kolaboracijske alate za komunikaciju između djelatnika kao što su Skype, Microsoft Teams itd. pa je stoga nužno ukloniti sve sigurnosne rizike koji se mogu javljati korištenjem navedenih servisa. Neki od najpoznatijih sigurnosnih rizika i problema koji se mogu javljati prilikom npr. kupovanja nekog proizvoda na web shopu su: lažno predstavljanje, prisluškivanje i mijenjanje podataka pri čemu može doći do krađe ili zloupotrebe osjetljivih podataka te na kraju nanijeti materijalnu štetu kupcu. Da bi se navedeni sigurnosni problemi uklonili, uveden je sustav javnih ključeva (*eng. Public Key Infrastructure*) koji koristi digitalne certifikate kao digitalno potpisane dokumente kako bi sudionici u komunikaciji mogli komunicirati putem kriptiranog kanala.

U prvom dijelu obraditi će se teorijski certifikati i pojmovi koji su vezani na njih (digitalni potpis, PKI itd.) te njihova primjena. Ukratko će se povezati SSL/TLS protokoli na način na koji koriste digitalne certifikate prilikom TLS rukovanja. Biti će opisani detaljno SSL/TLS rukovanje kao glavni pod protokol i temelj nad kojim se u našem primjeru u primjeni sa HTTPS-om jednim dijelom koriste certifikati kako bi se osigurao sigurni kanal u komunikaciji. Dalje će se još navesti razlika između jednosmjerne i dvosmjerne SSL autentifikacije kao i povijest verzija X509 certifikata te će se opisati struktura certifikata, njegova polja te proširenja.

U drugom dijelu, koji će biti praktičan, kreirati će se lanac certifikata od krajnjeg do korijenskog u Java programskom okruženju pri čemu će na taj način biti povezani pojmovi vezani za polja i proširenja koja su dio X509v3 certifikata.

U praktičnom dijelu rada napravljen je generator lanca certifikata u Java okruženju. Prvo se generira krovni certifikat koji je samo-potpisni, onda se dalje generira certifikat kojeg potpisuje krovni te je on na srednjoj razini te on služi za generiranje certifikata koji će koristiti krajnji klijent te kojeg potpisuje navedeni certifikat razine iznad. Kroz praktični primjer također

su navedeni još neki pojmovi koji u teorijskom dijelu nisu razrađeni te su još dodatno razjašnjena polja koja se nalaze u certifikatu te se na praktičnom primjeru mogu vidjeti naučena i primijenjena znanja iz teorijskog dijela koji je složio za uvod i pripremu na način da se prikazivalo što se ustvari točno događa i koja je točno primjera samih digitalnih certifikata.

2. Metode i tehnike rada

Ovaj diplomski rad sastoji se od teorijskog i praktičnog dijela. Prvo, u početku su definirani kriteriji i nad kojim pojmovima će biti fokus kako bi se mogli na kraju donijeti relevantni zaključci o temi. Kao podloga za izradu teorijskog dijela i kvalitetno shvaćanje same teme, korištena je stručna i znanstvena literatura. Korišteni su razni elektronički izvori te knjige i znanstveni članci. Korištena literatura dobro je analizirana te su uzeti najbitniji dijelovi i teorijska podloga pisana je u svrhu lakšeg shvaćanja teme te je za svaku teorijsku obradu prikazano kako to izgleda u praksi kako bi se navedeni teorijski pojmovi lakše shvatili. Praktični dio rada odnosi se na provedbu i implementaciju najvažnijih pojmova koji su obrađeni u radu. U zaključku rada su iznesene teze o naučenom i implementiranom, odnosno, kako digitalni certifikati funkcioniraju od samog prikaza kroz mrežne pakete u komunikaciji te dalje kako se putem njih ostvaruje sigurnosni kanal.

Za praktični dio izrađen je u programskom jeziku Java, verzija 1.8 Update 202 u najnovijoj verziji Eclipse IDE-a 2020-06. U samoj izradi i prikazu praktičnog primjera korišteno je više tehnologija. Za prikaz same web aplikacije, odnosno front-end dio korišten je JSF framework za prikaz komponenti korisničkog sučelja, dok je poslovna logika koja omogućava generiranje certifikata i keystorea rađena putem Java programskog jezika uz pomoć besplatnog programskog kriptografskog paketa BouncyCastle. Sama aplikacija, konfigurirana na HTTPS portu, bila je podignuta putem Tomcat servlet kontejnera koji održava aplikaciju živom.

3. Teoretska razrada teme

U sljedećim poglavljima napraviti će se teoretska obrada svih bitnih pojmova koji su potrebni za razumijevanje teme i izrade praktičnog dijela rada. Većina pojmova i naziva biti će prevedena na hrvatski ukoliko postoji prijevod. Prvo što će se obraditi jest SSL/TLS protokol koji pruža, mogli bismo reći, sigurnosnu osnovu zbog toga jer pruža zaštitu podataka (enkripciju) prilikom korištenja određenih web servisa. Poveznica sa digitalnim certifikatima je ta što se oni koriste za uspostavljanje sigurne veze u TLS rukovanju (eng. *Handshake*). Opisati će se i primjena TLS kod nekih određenih protokola (HTTPS, FTPS, SMTP itd.) te uloga digitalnih certifikata unutar njih. Biti će dan opis kako radi TLS rukovanje te razliku između jednosmjerne i dvosmjerne TLS autentifikacije zbog toga jer će se ti pojmovi koristiti u praktičnom dijelu.

Nadalje, sljedeći pojmovi koji će se teorijski obraditi su digitalni certifikati te pojmovi koji su vezani na njih (digitalni potpis, PKI infrastruktura itd.) te njihova primjena. Dalje će se obraditi povijest verzija X509 certifikata te će se opisati struktura certifikata, polja i proširenja.

3.1. SSL/TLS protokol

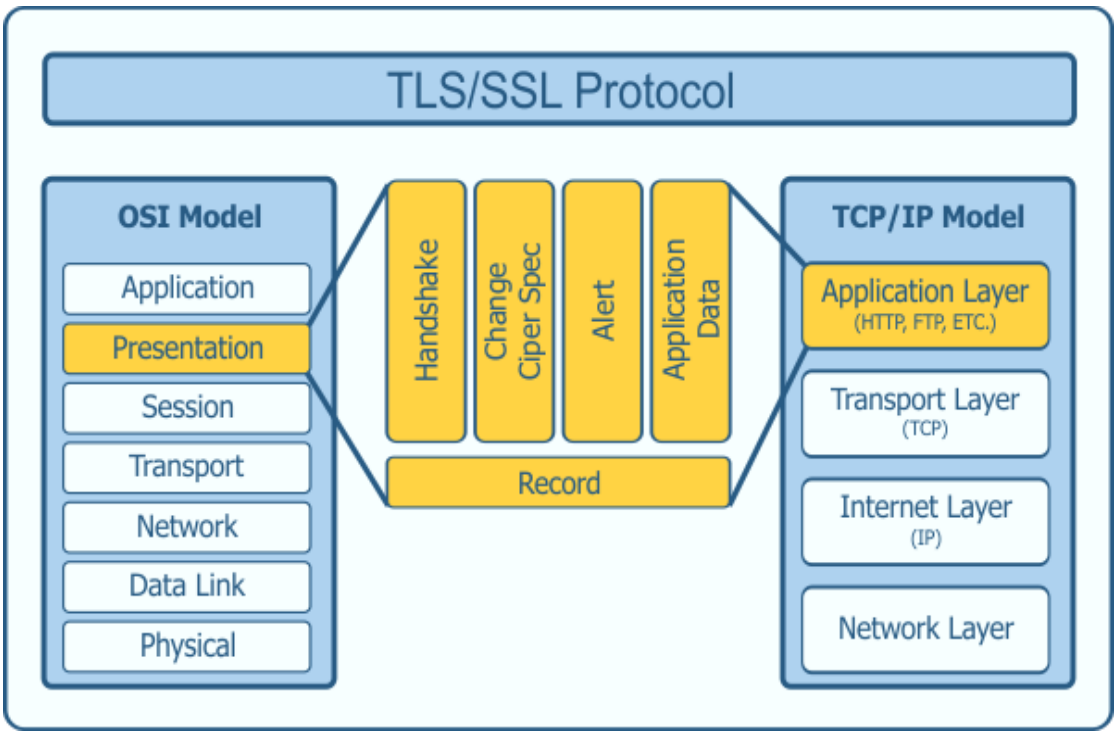
U ovom poglavlju prvo će se dati definicija i opis samog protokola, kako funkcionira. Navesti će se kratka povijest protokola, značajke koje je donijela najnovija verzija, suradnja s drugim protokolima te će se opisati najbitniji dijelovi protokola zasebno.

Kao što je već navedeno u uvodnom dijelu, današnjim povećanim korištenjem internetskih usluga, javlja se i veća potreba za brigom oko sigurnosnih standarda. Posebno je bitna sigurnost kod korištenja internetskih usluga kod kojih dolazi do izlaganja osobnih i osjetljivih informacija te je kod njih bitno uklanjati sve potencijalne sigurnosne rizike koji se mogu javiti. Umanjenje sigurnosnih rizika postiže se zaštitom navedenih osjetljivih podataka odnosno, njihovim kriptiranjem.

Jedan od načina zaštite navedenih osjetljivih podataka jest korištenje TLS (eng. *Transport Layer Security*) protokola između klijenta i servera pri čemu se postiže sigurna komunikacija kroz kriptirani kanal.

Definicija samog protokola glasi: TLS je kriptografski protokol koji omogućava razmjenu sigurnosno osjetljivih podataka na siguran način putem nezaštićenih računalnih mreža. Cilj TLS protokola jest omogućiti sigurnu komunikaciju između server/klijent aplikacija putem računalnih mreža na način da su iste zaštićene od prisluškivanja i promjene podataka.
[1]

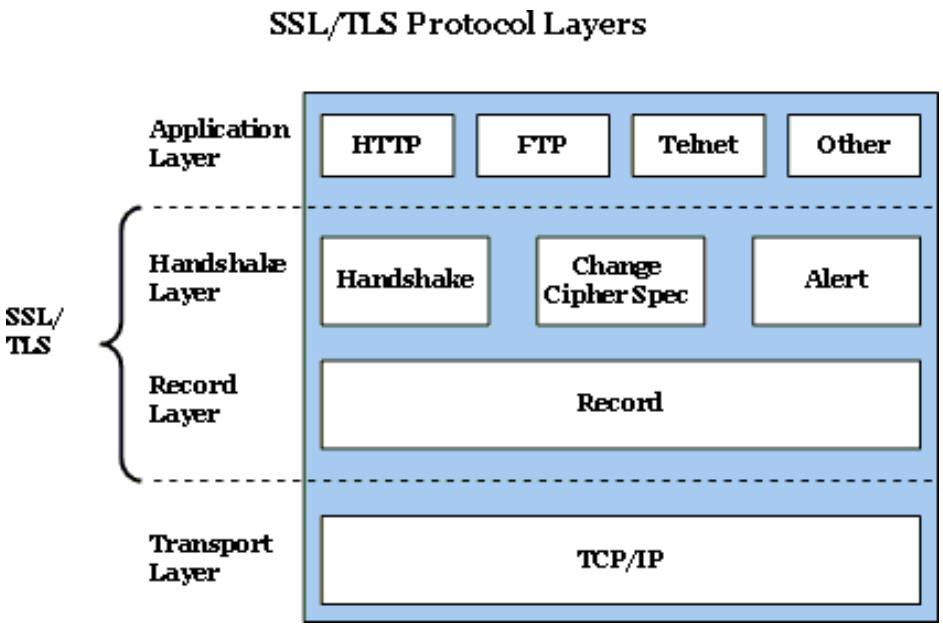
Na slici ispod prikazan je smještaj SSL/TLS protokola unutar oba mrežna modela (OSI i TCP/IP):



Slika 1: Prikaz SSL/TLS protokola unutar mrežnih modela OSI i TCP/IP [2]

Na *slici 1* je vidljivo da se SSL/TLS protokol nalazi unutar prezentacijskog sloja OSI modela koji je odgovoran za prezentaciju i enkripciju podataka prema aplikacijskom sloju. Prema TCP/IP modelu se SSL/TLS protokol nalazi unutar aplikacijskog sloja pri čemu aplikacije koriste SSL biblioteke koje omogućuju komunikaciju s TCP protokolom koji je unutar transportnog sloja. U nekim literaturama se spominje da SSL/TLS protokol radi unutar aplikacijskog sloja u OSI modelu zbog toga jer se koristi zajedno sa protokolima koji operiraju unutar tog sloja (HTTP, FTP, SMTP, POP3 itd.) Nadalje, negdje se spominje da se i koristi unutar sloja sesije zbog jer pruža sigurnost sesije od točke do točke. Krivog odgovora u ovom slučaju nema, a provedenim istraživanjem bih rekao da je prezentacijski sloj najbliže opisuje rad SSL/TLS protokola.

Navedeni TLS protokol uspostavlja dvije vrste veza za sigurni protok podataka, a to su *TLS sesija* i *TLS konekcija*. **TLS sesija** koristi se za usklađivanje parametara potrebnih za uspostavljanje TLS konekcije, dok se **TLS konekcija** koristi se za slanje podataka između dvije strane preko kriptiranog kanala. Na *slici 2* ispod, vidljivo je da se SSL/TLS protokol sastoji od 4 pod protokola: TLS Handshake (rukovanje), TLS Change Cipher Spec, TLS alert te TLS Application Data te 2 pod sloja: **pod sloj rukovanja** (eng. *Handshake Layer*) te **pod sloj zapisa** (eng. *Record Layer*).



Slika 2: Format SSL/TLS protokola [3]

Pod sloj rukovanja koristi se za određivanje koji će kriptografski algoritmi koristiti, služi za autentifikaciju servera i/ili klijenta te za razmjenu ključeva između klijenta i servera.

Pod sloj zapisa veže se na neki od protokola transportnog sloja, primjerice TCP/IP protokola te služi za preuzimanje i prijenos podataka iz aplikacijskog sloja, kriptiranje istih, fragmentiranje tih podataka ovisno o odabranom kriptografskom algoritmu. Opcionalno, podaci se mogu i komprimirati i dekomprimirati ovisno o primljenim/poslanim podacima. [3] [4]

TLS rukovanje pod protokol koristi se kako bi se klijent i server međusobno autentificirali i usuglasili parametre potrebne za komunikaciju (dogovor o kriptografskim algoritmima i uspostava ključeva), detaljnije će se u zasebnom pod poglavlju govoriti o TLS rukovanju. **TLS Change Cipher Spec** pod protokol koristi se za promjenu kriptografskih parametara koji su korišteni od strane servera i klijenta. Navedeni pod protokol podrazumijeva poruku koja govori primatelju u kriptiranom kanalu da pošiljatelj želi promjenu na novi par ključeva, koji će tada biti kreirani od razmijenjene informacije dobivene iz rukovanja. Obično se ovaj pod protokol koristi kod rukovanja za promjenu na simetrično kriptiranje ključeva. **TLS Alert** pod protokol omogućava izmjenu standardiziranih upozoravajućih poruka. Neke od poruka koje pod protokol može prijaviti su: npr. poruka ne može biti dekriptirana, primljena poruka je neispravna, veza je zatvorena itd. Također, unaprijeđena TLS verzija ima više standardiziranih poruka od njegovog prethodnika SSL protokola. [1] [5] [6]

Application Data pod protokol omogućava stranama razmjenu podataka putem kriptiranog tj. sigurnog kanala aplikativnog sloja te su to ustvari podaci koji dolaze iz aplikacijskog sloja.

Sva navedena 4 pod protokola koriste **pod protokol zapisa** (eng. *Record Protocol*) koji spada pod niži pod sloj od navedenih (pod sloj zapisa). Ovaj pod protokol zapisa osigurava podatke dobivene iz aplikacijskog sloja (*Application data*) koristeći ključeve kreirane tijekom rukovanja te također provjerava integritet i podrijetlo navedenih podataka akcijama koje su već navede da se vrše u pod sloju zapisa. [7] [8]

Ukratko, svime navedenim i opisanim može se reći jednostavno da se uspostavljanje sigurnog kanala sastoji od više različitih procesa: rukovanja, razmjene ključeva, transfera podataka uključujući i poruke greški te sve završava procesom zatvaranja **sjednice** (eng. *session*). U sljedećim pod poglavljima će se detaljnije opisivati kako ti pojedini procesi funkcioniraju.

Sigurnosnu komunikaciju uvijek pokreće klijent, dok server odgovara na zahtjev. Također, neki od protokola koji koriste SSL/TLS protokol za kriptiranje podataka su: HTTPS, SMTP, POP3, SFTP, SIP - Signaling over TLS itd.

Dalje, u razradi ove teme će se većinom isticati samo pojam TLS zbog toga jer je to novija i unaprijeđena verzija SSL-a pa da se ne piše stalno SSL/TLS.

3.1.1. Ciljevi protokola

Što se tiče aspekta mrežne sigurnosti TLS protokol osigurava sljedeće sigurnosne zahtjeve:

1. *Tajnost*: samo pošiljatelj i primatelj trebaju razumjeti sadržaj poruke
2. *Integritet*: pošiljatelj i primatelj žele osigurati da se poruka prilikom slanja/primanja ne mijenja
3. *Autentifikacija*: primatelj i pošiljatelj žele međusobno potvrditi identitete [9]

Kako bi se osigurali sljedeći sigurnosni zahtjevi, potrebno je još navesti ciljeve kojima protokol teži, a to su redom prema prioritetu:

1. **Kriptografska sigurnost**: cilj je uspostavljanje sigurnog kanala između primatelja i primatelja. Ovaj cilj predstavlja glavni problem.
2. **Interoperabilnost**: svim programerima trebaju se omogućiti sučelja za programiranje aplikacija (API) te biblioteke/klase koje mogu komunicirati s drugima koristeći kriptografske parametre. To znači da protokol mora biti dostupan i za ostale aplikacije koje ga koriste.
3. **Proširivost** (eng. *Extensibility*): ovaj važan cilj iskazuje važnost da se uklanja ovisnost o korištenim kriptografskim parametrima (korištene funkcije sažimanja i korišteni algoritmi kriptiranja) na način da se lako može napraviti

migracija i nadogradnja aplikacija promjenom parametara bez potrebe za kreiranjem novih protokola.

4. **Efikasnost**: ovaj zadnji cilj predstavlja efikasnost i brzinu izvođenja protokola, odnosno specifičnih skupih i dugotrajnih kriptografskih operacija gdje se koriste vrlo složene matematičke funkcije te je potrebno trajanje takvih operacija svesti na minimum te također provoditi keširanje nekih stvari u sesijama kako bi se izbjegavalo korištenje nepotrebnih mrežnih zahtjeva. [10]

   U sljedećim pod poglavljima spominjati će se neke nadogradnje kroz verzije koje su omogućile i poboljšale izvođenje navedenih ciljeva. Posebice je skrenuta pozornost na dio s efikasnošću gdje je dosta velik napredak napravljen sa najnovijom verzijom TLS 1.3.

3.1.2. Kratka povijest protokola

Prvi od navedenih protokola bio je SSL (eng. *Secure Sockets Layer*) te možemo reći da su on i njegova nadogradnja TLS protokola, usko povezani i imaju istu svrhu zaštite podataka. Razlika između njih je u tome što TLS koristi naprednije kriptografske algoritme za generiranje tajnih ključeva.

Prva verzija koja se stvarno koristila bila je SSL 2.0, dizajnirana od tvrtke Netscape te se koristila u njihovom proizvodu Netscape Navigator koji je bio najpopularniji web preglednik u to vrijeme (radilo se o verziji 1.1 izdanoj u ožujku 1995. godine). SSL 3.0 izašao je godinu kasnije kao nadogradnja te je bio dizajniran od strane troje ljudi i bio je široj uporabi sve do jeseni 2014. kada je otkrivena veća sigurnosna ranjivost od strane Google-ovog odjela za sigurnost. [11]

Sljedeća nadogradnja protokola bila je TLS 1.0 verzija koji je izašao u javnost 1999. godine. TLS je donio neke razlike i poboljšanja i činio se kao puno sigurnija verzija od SSL 3.0. Ime protokola je promijenjeno kako bi se zadovoljio Microsoft. Za razliku od SSL-a koji je bio razvijen od strane Netscape-a i temeljem toga postao *de facto* standard, TLS je razvijen od strane IETF radne grupe (eng. *Internet Engineering Task Force*). [11]

Daljnji razvoj protokola dao je verziju TLS 1.1, kreiranu 2006. godine koja je sadržavala određene sigurnosne zakrpe poput:

1) Implicitni inicijalizacijski vektor (IV) je zamijenjen sa eksplicitnim zbog potrebe zaštite od CBC napada. CBC napad podrazumijeva napad tekstom koji ovisi o vrijednosti inicijalizacijskog vektora

2) Dodane su dodatne informativne bilješke za razne nove napade na TLS protokol
3) Prerano zatvaranje više ne uzrokuje da trajanje sesije nije trajno

[12, p. 5]

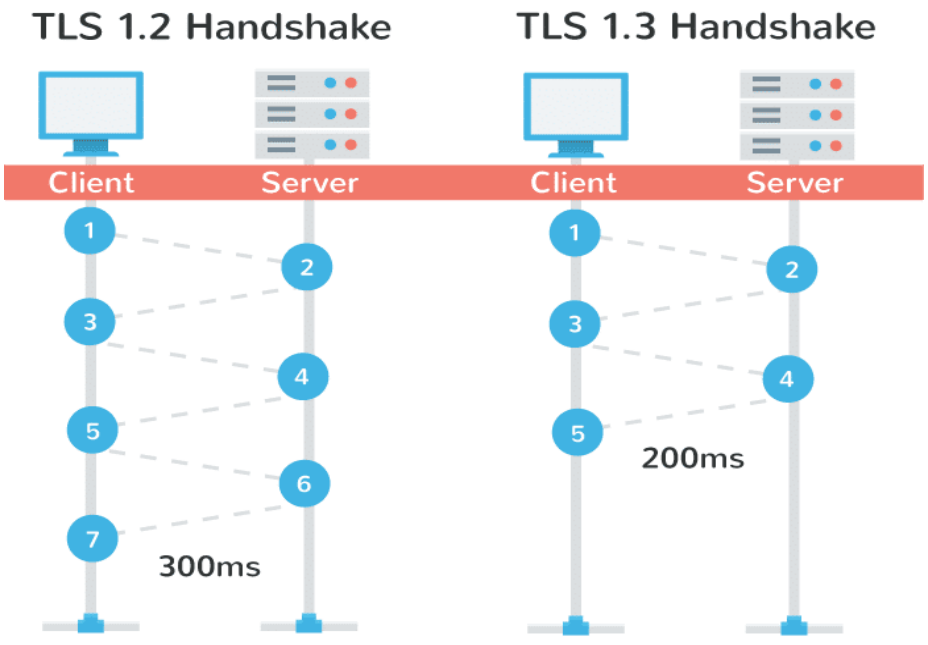
TLS verzija 1.2 je bila izdana u kolovozu 2008. godine u kojoj je bila dodana podrška za provjereno kriptiranje i bila je uklonjena ovisnost o MD5 i SHA-1 algoritmima za sažimanje. [13]

3.1.3. TLS 1.3 i njegove značajke

Najnovija verzija je TLS 1.3 koja je objavljena u kolovozu 2018. zajedno s unaprijeđenom verzijom HTTP/2 protokola čini kriptirane veze sigurnijima i bržima. Samo korištenje HTTP/2 protokola omogućava veće performanse korištenjem TLS-a, a dodatne performanse dobivene su novim mogućnostima kao što *TLS false start* i *Zero Round Trip Time Resumption* (0-RTT).

TLS lažno pokretanje (eng. *False Start)* jest optimizacijska ekstenzija TLS protokola koja omogućava klijentu i serveru pokretanje prijenosa kriptiranih podataka dok je rukovanje (eng. *Handshake*) tek djelomično gotovo što smanjuje korak rukovanja na jednu povratnu vezu. O rukovanjima će se detaljnije govoriti u sljedećim poglavljima. [14]

Na slici ispod prikazana je razlika u rukovanjima između TLS 1.2 i najnovije TLS 1.3 verzije gdje je vidljivo da se koristi jedna povratna veza manje pri čemu se dobiva na brzini odziva i koliko TLS rukovanje traje.



Slika 3: Usporedba performansi TLS rukovanja između verzija 1.2 i 1.3 [14]

Druga nova mogućnost koju je preostalo definirati, a koja je donijeta u TLS 1.3 verziju, jest **Zero Round Trip Time Resumption** (0-RTT) čime se omogućuje slanje podataka prilikom prve poruke prema serveru kod stranica koje su već prethodno posjećene čime se također smanjuje vrijeme odziva. Samo značenje **RTT**-a (eng. *Round Trip* Time) jest da je to vrijeme trajanja u milisekundama koje je potrebno podatkovnim paketima da budu poslani do odredišta zajedno sa vremenom koje je potrebno za prihvat paketa i slanja natrag na odredišnu adresu. [15]

Posebno se to očituje kod mobilnih mreža, a dosta je primjetno i kod latencija. Primjerice, zamislimo da je latencija prema web stranici kojoj želimo pristupiti 300 ms što znači da je server na kojem je ona hostana dosta udaljen od nas. U tom slučaju, nam samo TLS rukovanje može trajati i oko sekunde dok se uopće stranica ne počne učitavati što nije poželjno za korisničko iskustvo. Zbog toga je bitno da je broj povratnih veza u TLS rukovanju čim manji pošto je vrijeme trajanja samo 1 zahtjeva u slučaju velike latencije dosta veliko. [16]

3.1.4. TLS rukovanje

Kao što je već navedeno u uvodnom dijelu TLS rukovanje najbitniji dio TLS protokola tijekom kojeg se odvija autentifikacija poslužitelja i/ili klijenta te pregovor o kriptografskim parametrima koji će se koristiti tijekom zajedničke sjednice. Obično se kod ovog pod protokola izmjenjuje 6-10 poruka, ovisno o mogućnostima koji se koriste (npr. da li se radi o jednosmjernoj ili dvosmjernoj autentifikaciji itd.)

Poruka koja se šalje počinje sa zaglavljem u kojem se nalazi tip poruke (veličina 1 bajt), duljina poruke (3 bajta veličina), dok preostalo što se šalje su podaci poruke koji ovise o tipu poruke. Pobliže je u nastavku dan format poruke pod protokola te isječak iz programskog alata Wireshark na kojem je vidljivo da se u poslanoj poruci nalazi tip poruke, duljina te ostali podaci koji se šalju ovisno o tipu poruke:

```
struct {
        HandshakeType msg_type;
        uint24 length;
        HandshakeMessage message;
} Handshake;
```

```
▼ Handshake Protocol: Client Hello
    Handshake Type: Client Hello (1)
    Length: 294
    Version: TLS 1.2 (0x0303)
  › Random: b4d06a3075edc94f7d6a1ca8269bc6b931a47ac5c9d5342f…
    Session ID Length: 32
    Session ID: 013c0fcaea36cbb5010d29e8c7cf493124f3c973011549b2…
    Cipher Suites Length: 32
  › Cipher Suites (16 suites)
```

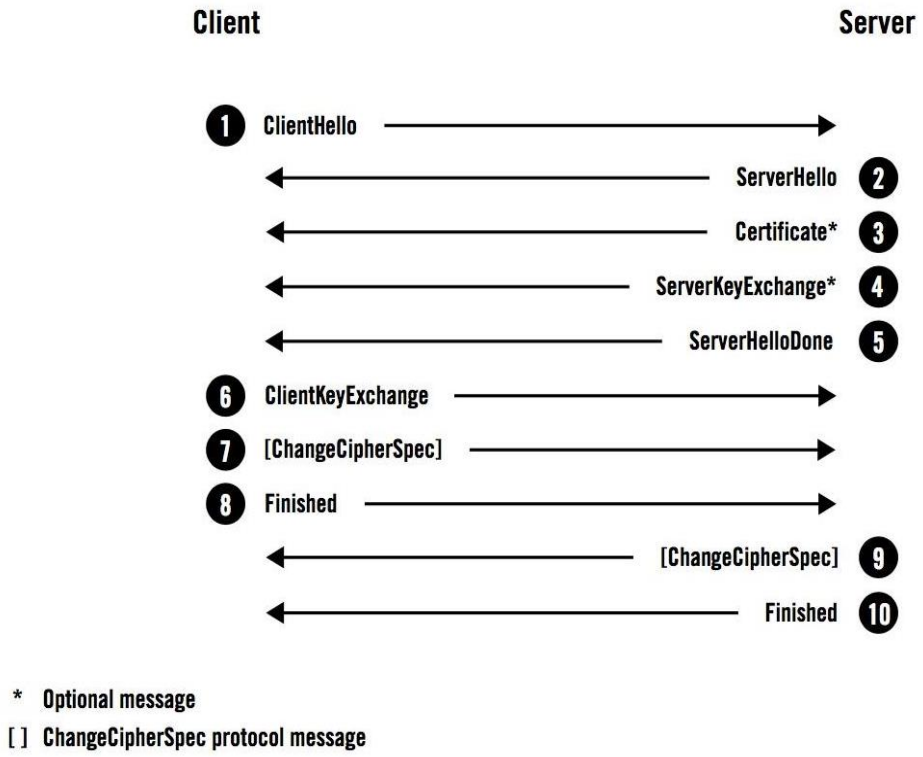
Slika 4: Format *Client Hello* poruke koja se šalje TLS rukovanjem

[17]

Dalje, pojednostavljenim prikazom, izvode se sljedeće akcije između klijenta i servera prilikom rukovanja te će one biti dalje detaljnije razrađene?:

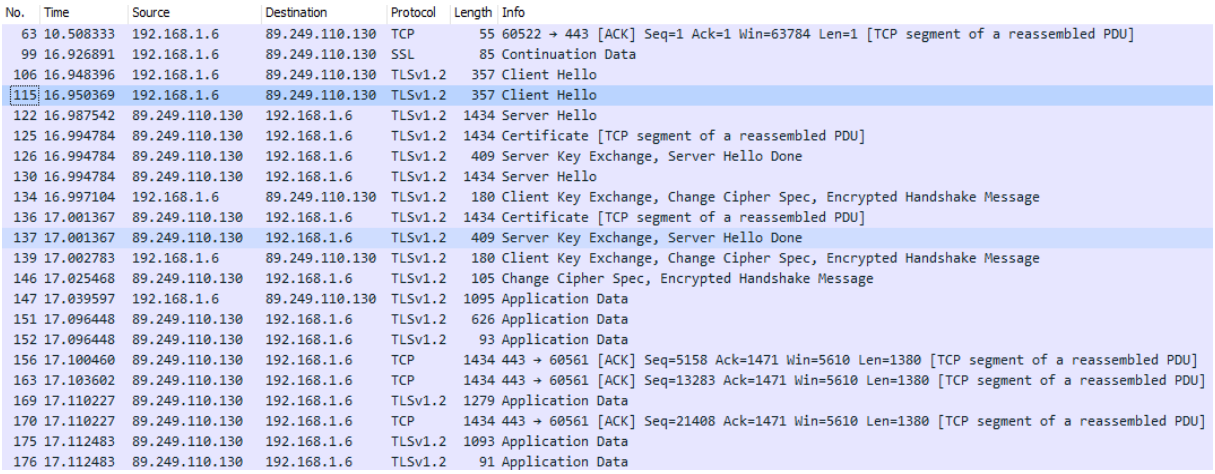
1.

Na sljedećoj slici prikazan je slijed izvođenja procesa TLS rukovanja korištenjem jednosmjerne (eng. *one-way*) autentifikacije:



Slika 5: Prikaz procesa rukovanja sa jednosmjernom autentifikacijom [18]

U sljedećem dijelu pod poglavlja biti će prikazani koraci procesa, odnosno poruke koje se šalju te će se dati opis pojedinih poruka. Razlika između jednosmjerne i dvosmjerne komunikacija je ta da se kod jednosmjerne autentificira samo server, dok kod dvosmjerne također klijent treba serveru potvrditi svoj identitet. U nastavku diplomskog rada se će detaljnije objasniti pojmovi jednosmjerne i dvosmjerne autentifikacije kada će se obraditi pojmovi digitalnih certifikata te PKI infrastrukture. Također, na sljedećem isječku iz Wireshark programskog alata vidljivo je također slijedno kako se odvija komunikacija tijekom rukovanja:



No.	Time	Source	Destination	Protocol	Length	Info
63	10.508333	192.168.1.6	89.249.110.130	TCP	55	60522 → 443 [ACK] Seq=1 Ack=1 Win=63784 Len=1 [TCP segment of a reassembled PDU]
99	16.926891	192.168.1.6	89.249.110.130	SSL	85	Continuation Data
106	16.948396	192.168.1.6	89.249.110.130	TLSv1.2	357	Client Hello
115	16.950369	192.168.1.6	89.249.110.130	TLSv1.2	357	Client Hello
122	16.987542	89.249.110.130	192.168.1.6	TLSv1.2	1434	Server Hello
125	16.994784	89.249.110.130	192.168.1.6	TLSv1.2	1434	Certificate [TCP segment of a reassembled PDU]
126	16.994784	89.249.110.130	192.168.1.6	TLSv1.2	409	Server Key Exchange, Server Hello Done
130	16.994784	89.249.110.130	192.168.1.6	TLSv1.2	1434	Server Hello
134	16.997104	192.168.1.6	89.249.110.130	TLSv1.2	180	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
136	17.001367	89.249.110.130	192.168.1.6	TLSv1.2	1434	Certificate [TCP segment of a reassembled PDU]
137	17.001367	89.249.110.130	192.168.1.6	TLSv1.2	409	Server Key Exchange, Server Hello Done
139	17.002783	192.168.1.6	89.249.110.130	TLSv1.2	180	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
146	17.025468	89.249.110.130	192.168.1.6	TLSv1.2	105	Change Cipher Spec, Encrypted Handshake Message
147	17.039597	192.168.1.6	89.249.110.130	TLSv1.2	1095	Application Data
151	17.096448	89.249.110.130	192.168.1.6	TLSv1.2	626	Application Data
152	17.096448	89.249.110.130	192.168.1.6	TLSv1.2	93	Application Data
156	17.100460	89.249.110.130	192.168.1.6	TCP	1434	443 → 60561 [ACK] Seq=5158 Ack=1471 Win=5610 Len=1380 [TCP segment of a reassembled PDU]
163	17.103602	89.249.110.130	192.168.1.6	TCP	1434	443 → 60561 [ACK] Seq=13283 Ack=1471 Win=5610 Len=1380 [TCP segment of a reassembled PDU]
169	17.110227	89.249.110.130	192.168.1.6	TLSv1.2	1279	Application Data
170	17.110227	89.249.110.130	192.168.1.6	TCP	1434	443 → 60561 [ACK] Seq=21408 Ack=1471 Win=5610 Len=1380 [TCP segment of a reassembled PDU]
175	17.112483	89.249.110.130	192.168.1.6	TLSv1.2	1093	Application Data
176	17.112483	89.249.110.130	192.168.1.6	TLSv1.2	91	Application Data

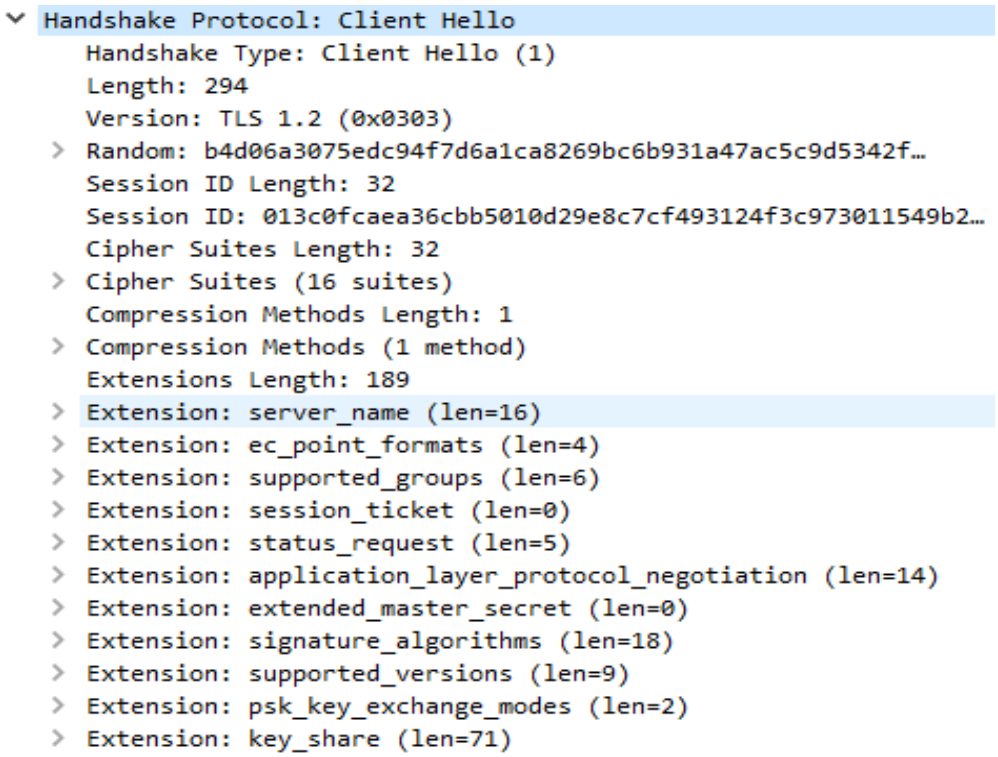
Slika 6: Prikaz procesa rukovanja iz Wireshark alata, također je korištena jednosmjerna autentifikacija u ovom scenariju

Paketi su snimani prilikom pristupa web preglednikom prema domeni *fina.hr* koja je na HTTPS portu. U stupcu info na slici 6 vidljiv je tip poruke koji se šalje. IP adresa *192.168.1.6* je adresa lokalnog računa s kojeg se pristupa prema *fina.hr* domeni (IP adresa: 89.249.110.130). U nastavku će biti opisane najbitnije poruke koje se šalju tijekom izvođenja rukovanja.

3.1.4.1. ClientHello

Na slikama 5 i 6 također je vidljivo da sam proces rukovanja započinje porukom **ClientHello** koju šalje klijent prema poslužitelju. Kod te poruke se šalju klijentske mogućnosti i njegove postavke, odnosno, klijent ponudi izbore poslužitelju koji mora odabrati jedan. Ova poruka se šalje kod slučajeva iniciranja veze, kod pregovaranja ili kod slučaja odgovora na zahtjev za pregovaranje koji šalje poslužitelj. [17]

Na sljedećem isječku iz alata Wireshark prikazan je primjer kako može ClientHello poruka izgledati:



Slika 7: Primjer ClientHello poruke

Strukturu poruke nije previše teško razumjeti, jer imena polja pobliže opisuju svrhu svakog polja pa tako će u nastavku biti ukratko opisana najvažnija polja koja se nalaze u ClientHello poruci:

1. *Version*: označava najveću podržanu verziju SSL/TLS protokola koju klijent može podržavati
2. *Random*: ovo polje sastoji se od 32 bajta, pri čemu su prvih 28 nasumično generirani. Za preostala 4 bajta web preglednici dodaju nasumično iskrivljeno vrijeme ili šalju također 4 nasumično generirana broja.

   Ovaj broj je ustvari *nonce*, što znači da je to slučajni broj koji se koristi samo jednom. [9] Također klijent i poslužitelj sudjeluju u razmjeni ovog polja, a nasumičnost omogućuje da je svaki proces rukovanja jedinstven te na taj način se štiti proces od potencijalnih napada te se ispravno može provjeravati integritet podataka koji se razmjenjuju.
3. *SessionID*: ovo polje predstavlja identifikator sjednice te ukoliko je prazan, to znači da klijent ne želi nastaviti koristiti postojeću sjednicu s danim poslužiteljem. Identifikator je također nasumično generirani broj duljine 32 bajta te zbog toga što je jedinstven, omogućuje poslužitelju da pronađe za njega odgovarajuće stanje sjednice u svojoj predmemoriji (eng. *cache-u*).
4. *Cipher suites*: ovo polje označava skup kriptografskih algoritama (eng. *Cipher suites*) koji mogu raditi zajedno kako bi se vršio proces rukovanja,

šifriranja/dešifriranja (eng. *encryption/decryption*) te procesa razmjene ključeva. Jednostavnijim rječnikom može se reći da je cipher suite skup algoritama koji definiraju točno na koji način će se osiguravati mrežni kanal koji koristi TLS/SSL protokol. Na sljedećoj slici prikazan je primjer podržanih kriptografskih algoritama koji se mogu koristiti od strane klijenta: [19]

```
Cipher Suites (16 suites)
    Cipher Suite: TLS_AES_128_GCM_SHA256 (0x1301)
    Cipher Suite: TLS_AES_256_GCM_SHA384 (0x1302)
    Cipher Suite: TLS_CHACHA20_POLY1305_SHA256 (0x1303)
    Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02b)
    Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)
    Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0xc02c)
    Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)
    Cipher Suite: TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca9)
    Cipher Suite: TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca8)
    Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013)
    Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)
    Cipher Suite: TLS_RSA_WITH_AES_128_GCM_SHA256 (0x009c)
    Cipher Suite: TLS_RSA_WITH_AES_256_GCM_SHA384 (0x009d)
    Cipher Suite: TLS_RSA_WITH_AES_128_CBC_SHA (0x002f)
    Cipher Suite: TLS_RSA_WITH_AES_256_CBC_SHA (0x0035)
    Cipher Suite: TLS_EMPTY_RENEGOTIATION_INFO_SCSV (0x00ff)
```

Slika 8: Primjer popisa kriptografskih algoritama koji se koriste tijekom rukovanja

U cipher suite-u mogu se nalaziti sljedeći atributi:

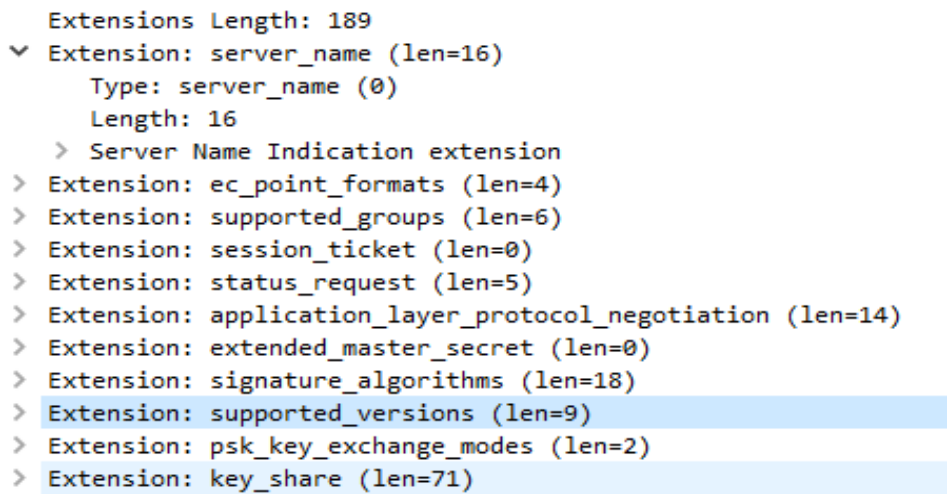
I. Metode autentifikacije
II. Metode razmjene ključeva
III. Kriptografski algoritmi i veličina ključeva koji će se koristiti prilikom kriptiranja podataka
IV. Način šifriranja (eng. *Cipher mode*)
V. MAC algoritmi (eng. *Message Authentication Code*) – algoritmi koji koriste simetričnu kriptografiju te služe za autentifikaciju poruka i garantiranje integriteta [20]
VI. PRF - pseudo slučajne funkcije (koriste se od TLS 1.2 verzije)
VII. Funkcije sažimanja koje se koriste kod poruke završetka rukovanja (koriste se od TLS 1.2 verzije)
VIII. Duljine *verify_data* strukture (koriste se od TLS 1.2 verzije) [21]

5. *Compression* – klijent može koristiti jednu ili više metoda komprimiranja (eng. *Compression*). Ukoliko je metoda kompresije *null,* tada kompresije u rukovanju nema.

6. *Extensions* - ovo polje može sadržavati veći broj ekstenzija (proširenja) od kojih svaka sadrži neke dodatne informacije. Cilj ovih ekstenzija je dodavati nove funkcionalnosti TLS protokolu bez potrebe internih promjena u njemu. Ove ekstenzije su se prvo pojavile još 2003. godine pod specifikacijom *RFC* 3456 te su dodane u TLS verziju 1.2 kada je izašla. Format ekstenzija sličan je kao i kod formata poruka koje se šalje tijekom rukovanja, a sastoji od 2 bajtnog jedinstvenog identifikatora koji označava vrstu ekstenzije te podataka koji se šalju ovisno o vrsti ekstenzije :

```
struct {
ExtensionType extension_type;
opaque extension_data;
} Extension;
```

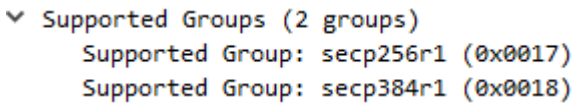
U isječku iz paketa prikazane su neke od ekstenzija koje se pojavljuju jedino u porukama *ClientHello* i *ServerHello*:



Slika 9: Prikaz ekstenzija analiziranog paketa poruke ClientHello

U nastavku će biti kratki opis najbitnijih ekstenzija:

a) *server_name* - sadrži domenu predviđenog domaćina (eng. *host*) za uspostavljanje sigurne veze
b) *ec_point_formats* (Supported Point Formats Extension) - označava skup formata točaka koje klijent može raščlaniti te je ova ekstenzija vezana uz algoritme eliptične krivulje (eng. *Eliptic curve*) [22]
c) *supported_groups* – sadrži imena grupa koje korisnik podržava vezano za razmjenu ključeva, sortirano od najprihvatljivije grupe nadalje: [23]

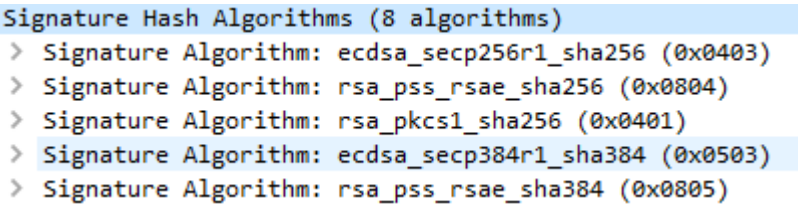


Slika 10: Primjer navedenih imena grupe koji su vezani uz korištenje algoritama krivulje

d) *session_ticket* - označava podršku za nastavak sjednice bez stanja

e) *application_layer_protocol_negotiation* – sadrži podržane protokole aplikacijskog sloja s kojima klijent želi pregovarati

f) *signature_algorithms* – sadrži popis podržanih funkcija sažimanja i algoritama za digitalno potpisivanje:

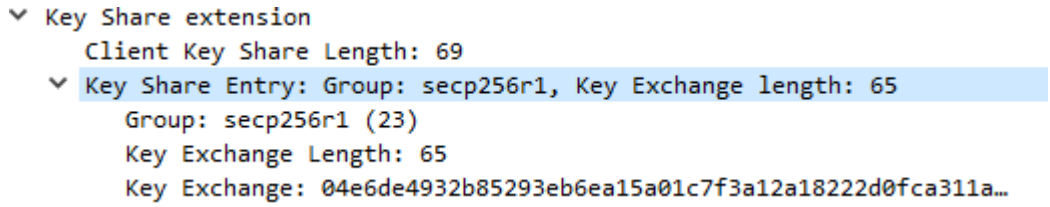


Slika 11: Primjer korištenih asimetričnih algoritama i algoritama Eliptične krivulje u analiziranoj ClientHello poruci

g) *supported_versions* – sadrži popis verzija SSL/TLS protokola

h) *psk_key_exchange_modes* – ukoliko se žele koristiti unaprijed dodijeljeni ključevi (engl. *pre-shared keys*), mora se koristiti ova ekstenzija koja označava da klijent podržava navedene ključeve koje podržavaju dane načine rada koji se nalaze u navedenoj ekstenziji [23]

i) *key_share* – ekstenzija sadrži kriptografske parametre krajnje točke [23] :

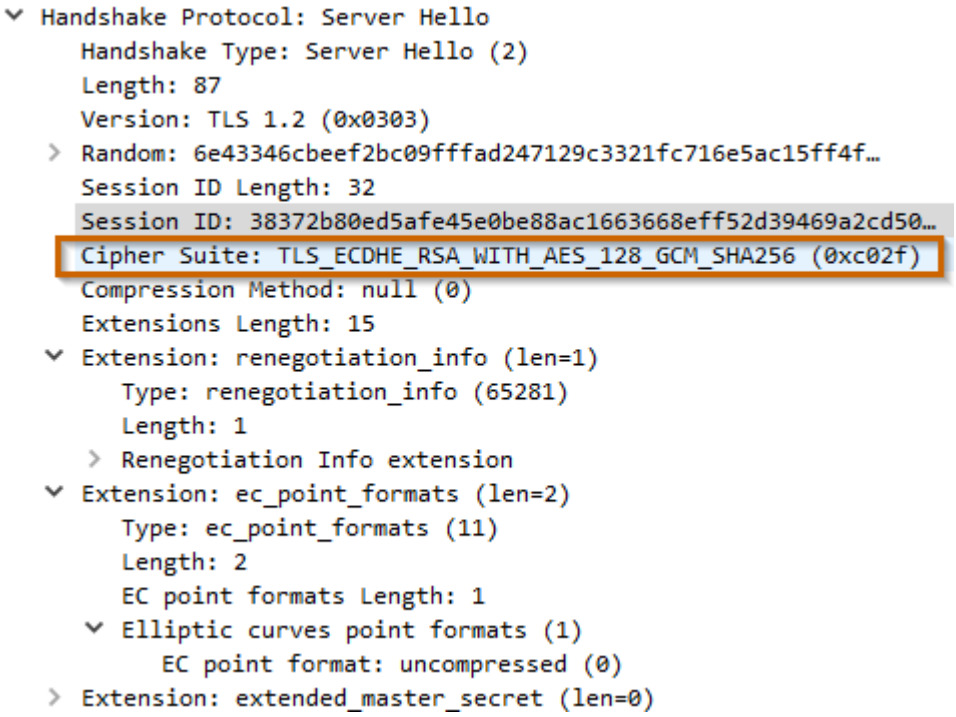


Slika 12: Primjer key_share proširenja analizirane ClientHello poruke

[24]

3.1.4.2. ServerHello

Sljedeće što se šalje jest *ServerHello* poruka koja je odgovor od strane poslužitelja prema klijentu gdje server vraća odabrane komunikacijske parametre te odabrani kriptografski skup koji će se koristiti dalje u komunikaciji što je vidljivo na isječku analiziranog prometa poruke ispod:



Slika 13: Analizirani promet ServerHello poruke

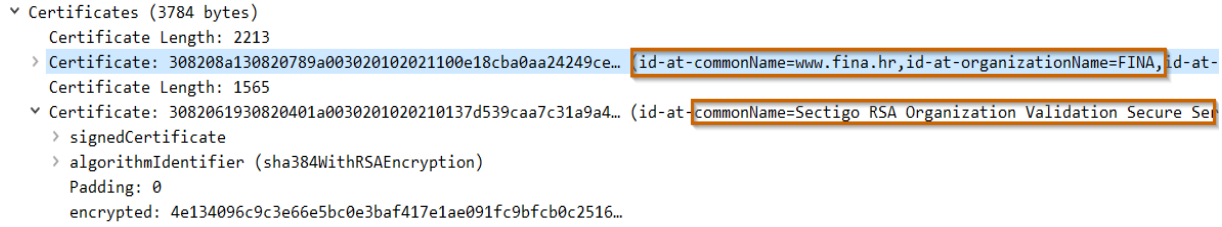
Na slici 13 vidljiv je označeni odabrani kriptografski skup od strane poslužitelja. Također, vidljivo je da metoda kompresije nije odabrana pošto je u prvoj poruci od strane klijenta proslijeđen *null* što znači da nema kompresije prilikom rukovanja u ovom danom analiziranom primjeru. Može se dogoditi ukoliko najnovija verzija koju zahtijeva klijent nije podržana od strane poslužitelja, tada poslužitelj nudi svoju verziju protokola u nadi da ju klijent podržati. Ovom porukom završen je prvi korak razmjene parametara veze, kriptografskih parametara i mogućnosti te preferencija između klijenta i poslužitelja. [17]

Usporedbom poruke poslane od strane klijenta i ove navedene može se reći da postoji razlika u poljima proširenja (ekstenzijama). Analizirana *ServerHello* poruka sadrži samo 3 proširenja, od kojih je jedno novo koje se ne nalazi u *ClientHello* poruci. Radi se proširenju renegotiation_info koji označava da će se ponovo izvoditi pregovaranje između istih dvaju učesnika koji su izvodili isto prethodno rukovanje. [24]

3.1.4.3. Certificate

Slijedeća poruka koja se šalje je *Certificate* poruka koju šalje poslužitelj te ona služi za predstavljanje poslužitelja koji šalje svoj lanac certifikata kako bi ga klijent mogao autentificirati. Certifikati se šalju u ASN.1 notaciji u DER kodiranom formatu. Također, poslužitelj mora osigurati da se šalje onaj certifikat na verifikaciju onaj koji zadovoljava dogovorene kriptografske algoritme. Neki mehanizmi za razmjenu ključeva ovise o određenim podacima koji su ugrađeni u certifikat te certifikati te moraju biti potpisani isključivo onim algoritmima koji su podržani od strane klijenta čiji se popis podržanih šalje u *ClientHello* poruci što je već

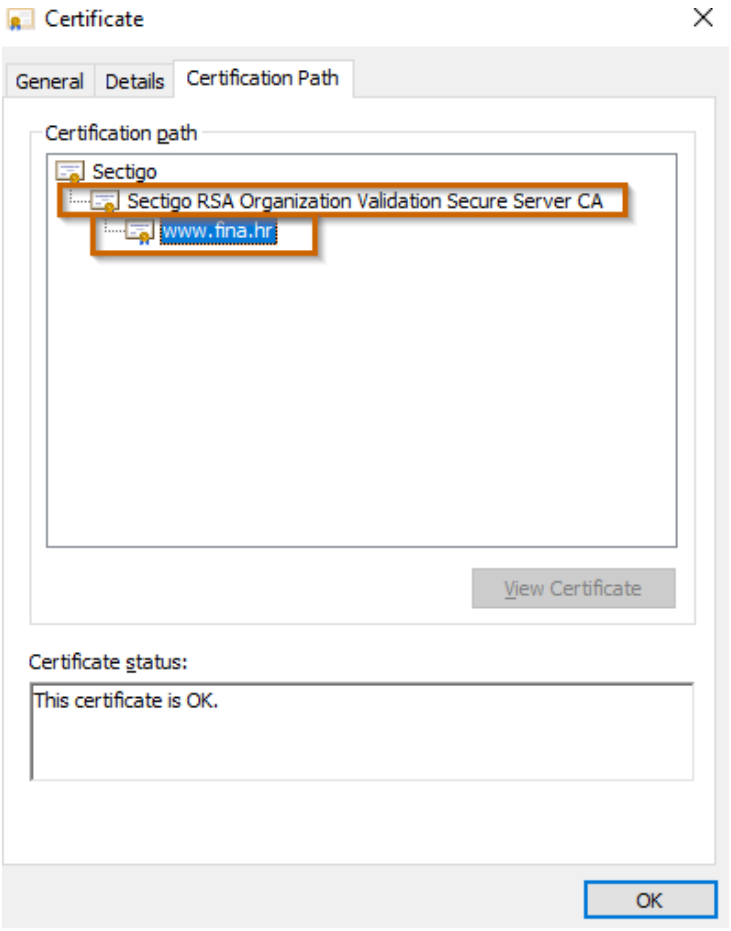
objašnjeno prethodno. Ova poruka je opcionalna jer određeni setovi algoritama ne zahtijevaju certifikate za autentifikaciju ili je konfiguracija složena da se ne koristi uopće autentifikacija. [17]



Slika 14: Primjer analiziranog paketa poruke Certificate koja se šalje za validaciju/autentifikaciju servera

Na prikazanom analiziranom primjeru paketa (slika 14) *Certificate* poruke vidljivo je da se šalju certifikati redom iz lanca certifikata (eng. *Certificate chain*): serverski certifikat koji se koristi za domenu za koju se radila analiza paketa te njegov potpisujući certifikat.

Usporedbom sa certifikatom koji se dobije od domene *fina.hr* dobije se isti lanac certifikata kao na slici 15 ispod gdje je vidljivo označeno da se commonName polja poklapaju s onim imenima dobivenim u analiziranom paketu:



Slika 15: Prikaz lanca certifikata dobivenog iz web preglednika analizirane domene

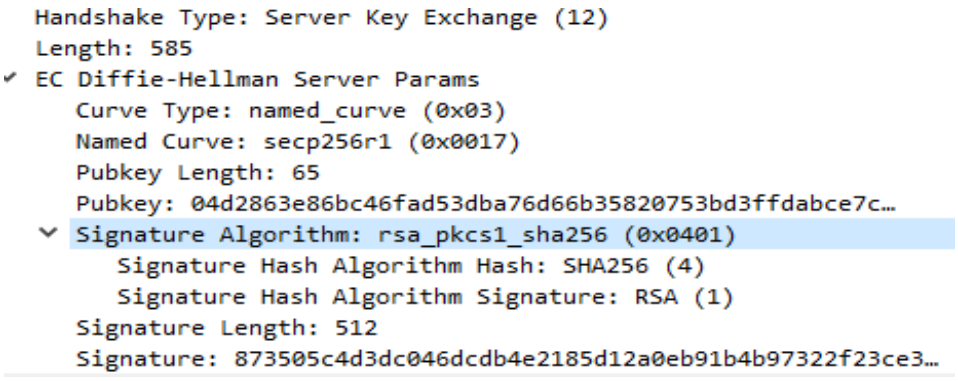
Također, navedeni pojmovi će se detaljnije obrađivati u poglavlju u kojem će se obrađivati struktura i format samog X509 certifikata.

Pod napomenu bih stavio da je web preglednik na različitim uređajima davao različite serverske certifikate za istu domenu pa nisam bio siguran u čemu je stvar, ali sam istraživanjem otkrio da je moguće imati više serverskih certifikata nad jednom domenom te tada poslužitelj odlučuje kojeg će servirati, ovisno o konfiguracijskim parametrima poslužitelja. Npr. mogu se staviti postavke da certifikat koji je najnovije stavljen na poslužitelj da se on poslužuje uz pretpostavku da je on tada najsigurniji. [25]

3.1.4.4. ServerKeyExchange i ServerHelloDone

Sljedeće dvije poruke koje se šalju od strane poslužitelja su *ServerKeyExchange* i *ServerHelloDone*.

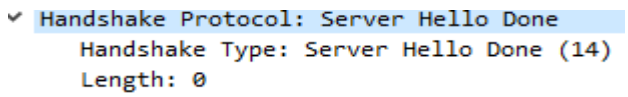
ServerKeyExchange poruka služi za slanje dodatnih podataka potrebnih za razmjenu ključeva te podaci koji se šalju također ovise o dogovorenom skupu algoritama koji se će koristiti za uspostavljanje sigurnog kanala kao na sljedećem primjeru:



Slika 16: Primjer parametara koji se šalju od strane poslužitelja kako bi se izvršila razmjena ključeva sa klijentom

Ova poruka također može biti opcionalna. Vidljivo je u primjeru sa slike 16 da se mogu slati podaci potrebni za razmjenu kao što su javni ključ, potpisni algoritam te sam digitalni potpis.

ServerHelloDone poruka je ustvari signal da je poslužitelj završio svoj dio sa slanjem poruka te nakon toga očekuje daljnje primanje poruka od strane klijenta te je na sljedećem primjeru vidljivo da se samo šalje signal klijentu da slanje od poslužitelja završilo:

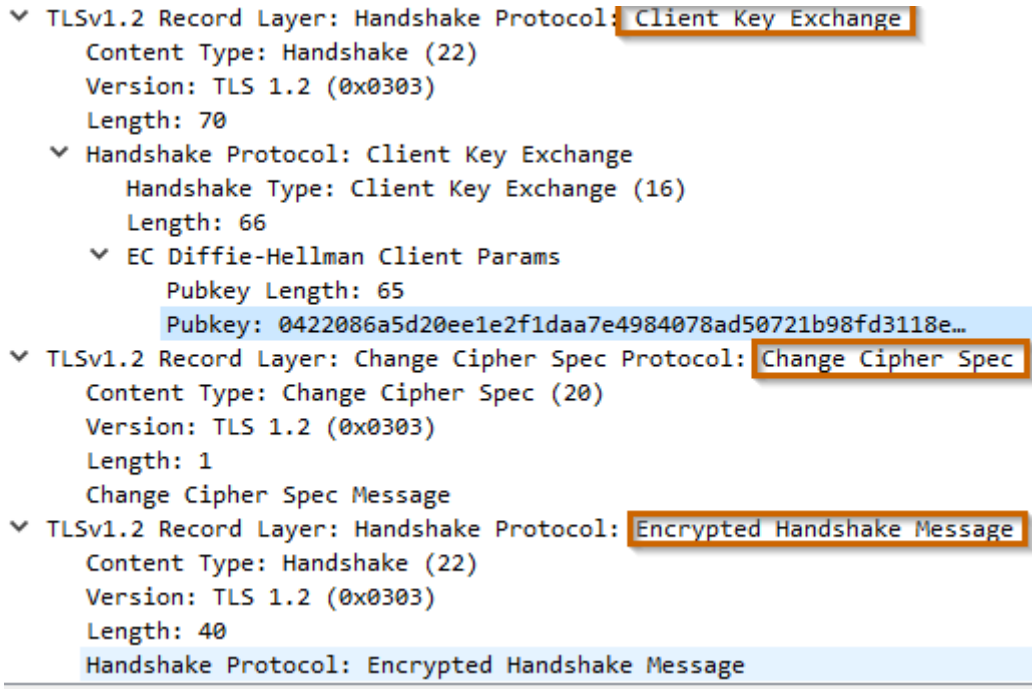


Slika 17: Primjer signale poruke ServerHelloDone

[17]

3.1.4.5. ClientKeyExchange, ChangeCipherSpec, Finished

Sljedeće navedene 3 poruke šalju se odjednom od strane klijenta:

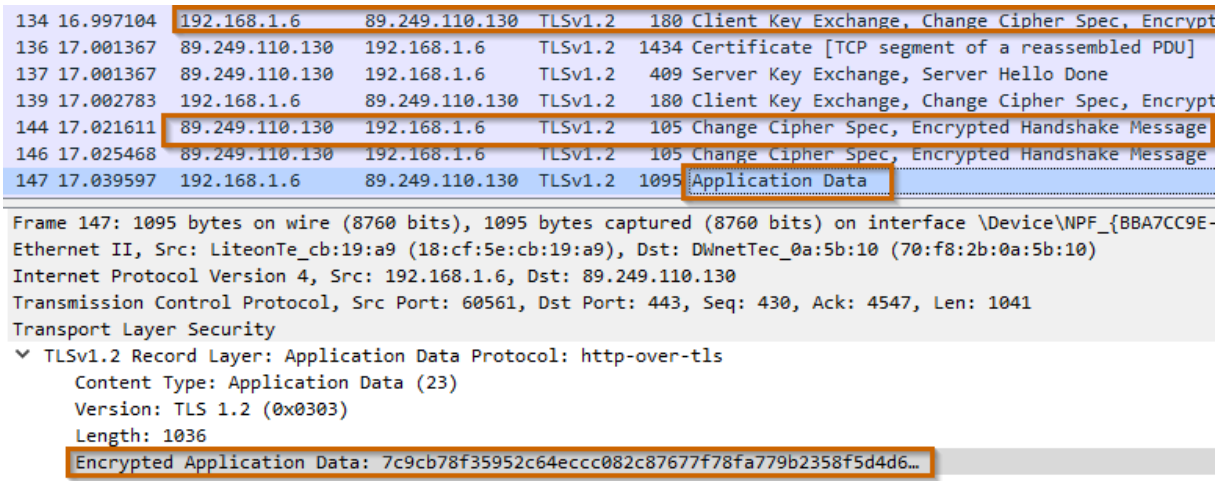


Slika 18: Prikaz sadržaja poruka ClientKeyExchange, ChangeCipherSpec i poruka završetka Finished

ClientKeyExchange je obavezna poruka koja se koristi kod razmjene ključeva (eng. *Key*-Exchange) te čiji sadržaj također ovisi o dogovorenom kriptografskom skupu.

ChangeCipherSpec, kao što je vidljivo na slici 18, ne spada pod protokol rukovanja, već zasebni pod protokol: **Change Cipher Spec Protocol**. Ova poruka označava signal da pošiljatelj sadrži dovoljno informacija za uspostavu mrežnih parametara, generiranje ključeva te sada prelazi na postupak kriptiranja samih podataka. Također, poruka označava drugoj strani da mora biti spremna prihvatiti podatke isključivo u kriptiranom formatu te da se podaci dalje neće moći pročitati ukoliko se ne dekriptiraju. Navedeni poruku šalju klijent i poslužitelj te poruka spada pod zasebni pod protokol zbog sigurnosnih razloga.

Poruka *Finished* jest prva poruka koja je kriptirana te ona predstavlja signal da je rukovanje završilo. Prvo poruku šalje klijent te ukoliko poslužitelj primi poruku i može ju dekriptirati, tada znači da može čitati kriptirane podatke na ispravan način. Kako bi klijent mogao dekriptirati podatke također je potrebno ponovno poslati *Change Cipher Spec* poruku, no ovaj put je pošiljatelj poslužitelj. Također vrijedi isto, ukoliko klijent primi poruku i može ju dekriptirati tada on može čitati kriptirane podatke poslane od poslužitelja na ispravan način. Ovaj postupak je vidljiv na slici ispod te je od tada nadalje sav promet kriptiran:



Slika 19: Proces slanja Finished poruka nakon kojih se dalje šalju kriptirani podaci

Sigurnost je dodatno poboljšanja poljem *verify_data* koji se šalje uz Finished poruku. To polje predstavlja sažetak svih poslanih poruka rukovanja zajedno za dogovorenom glavnom tajnom (eng. *master-secret*). [17] [26]

3.1.5. Izvođenje ključeva

Proces izvođenja/razmjene ključeva (eng. *Key Exchange*) označava najbitniji dio rukovanja gdje klijent i poslužitelj koriste zajedničku tajnu kako bi napravili skup ključeva. Ključevi se izvode korištenjem funkcije za izvođenje ključeva (eng. *Key derivation function*, KDF). Uzima se glavna tajna (48 bajta) i opcionalno neki dodatni pseudo - slučajni podaci te se kreiraju ključevi. [27]

Prvi razmjenu ključeva uspostavlja poslužitelj sa porukom *ServerKeyExchange* gdje je već navedeno da se šalju parametri potrebni za razmjenu ključeva: javni ključ, potpisni algoritam te digitalni potpis pomoću kojeg klijent može potvrditi vjerodostojnost poslužitelja. Vjerodostojnost je potvrđena ukoliko poslužitelj sadrži privatni ključ koji odgovara javnom ključu koji je dobiven iz *Certificate* poruke. [27]

Neovisno o korištenom asimetričnom algoritmu za razmjenu ključeva, klijent također porukom *ClientKeyExchange* šalje parametre za razmjenu ovisno o dogovorenom kriptografskom skupu pa tako isto može klijent slati svoj javni ključ. TLS podržava više asimetričnih algoritama za razmjenu ključeva kao što su RSA, algoritmi eliptičke krivulje te Diffie-Hellman algoritam za razmjenu ključeva. [27]

3.1.5.1. RSA

Kod ovog asimetričnog algoritma klijent prvo generira 48 bajtnu pred-glavnu tajnu (eng. *premaster secret*), od čega 2 bajta se koriste za verziju protokola, dok se preostali bajtovi generiraju od strane klijenta. Dobivenim javnim ključem iz poruke poslužitelja

ServerKeyExchange kriptira se generirana tajna de se šalje u *ClientKeyExchange* poruci te tada poslužitelj svojim privatnim ključem može dekriptirati tu pred-glavnu glavnu tajnu te tada oboje sudionika u kanalu dijele tu tajnu koju onda mogu koristiti za izračunavanje glavne tajne. Pred-tajna te generirani i razmijenjeni nonce brojevi od strane klijenta i poslužitelja se koriste u pseudo-slučajnoj funkciji, PRF (eng. *pseudorandom function*) pomoći kojih se proizvodi glavna tajna. [9] [27]

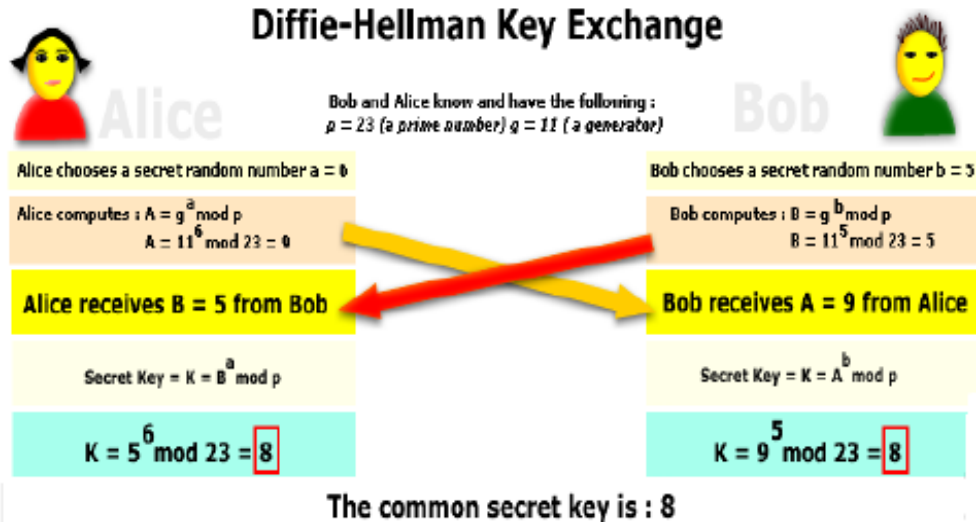
Definicija PRF-a jest da se koristi za generiranje proizvoljnog broja pseudo slučajnih podataka. [28]

Nakon generiranje glavne tajne, glavna tajna se ponovno obrađuje PRF funkcijama te se onda generiraju ključevi za klijenta i poslužitelja poput: ključa za MAC, ključa za kriptiranje te se još generira klijentski i poslužiteljski inicijalizacijski vektor. [9]

3.1.5.2. Diffie-Hellman

Razmjena ključeva na Diffie-Hellmanov način omogućava sudionicama komunikacije uspostavljanje dijeljenja tajne/ključa putem nesigurnog kanala. Ovaj način, tj. protokol za razmjenu ključeva predstavili su 1976. godine Whitfield Diffie i Martin Hellman s tezom kako je u pojedinim matematičkim strukturama bolje i jednostavnije koristiti potenciranje nego logaritmiranje. Zbog sigurnosnih razloga, koristi se zajedno s autentifikacijom jer ovaj način razmjene ključeva nema zaštitu od lažnog predstavljanja. [29]

Razmjena ključeva radi na sljedeći način kako je opisano na dijagramu ispod:



Slika 20: Prikaz korištenih operacija prilikom razmjene ključeva na Diffie-Hellmanov način [30]

Kako je i prikazano na slici 20, ovaj način razmjena ključeva funkcionira tako da *Alice* želi uspostaviti komunikaciju s Bobom te oboje odabiru nasumične prirodni broj *a, b* te šalju jedan drugome rezultate izračunavanja:

Alice šalje rezultat A:

$$A = g^a \ mod \ p$$

, dok Bob šalje rezultat B:

$$B = \ g^b mod \ p$$

Na kraju, tajni ključ koji Alice i Bob mogu koristiti za simetrično kriptiranje jest izračunata vrijednost K.

Alice izračunava K na način:

$$K = \ B^a mod \ p$$

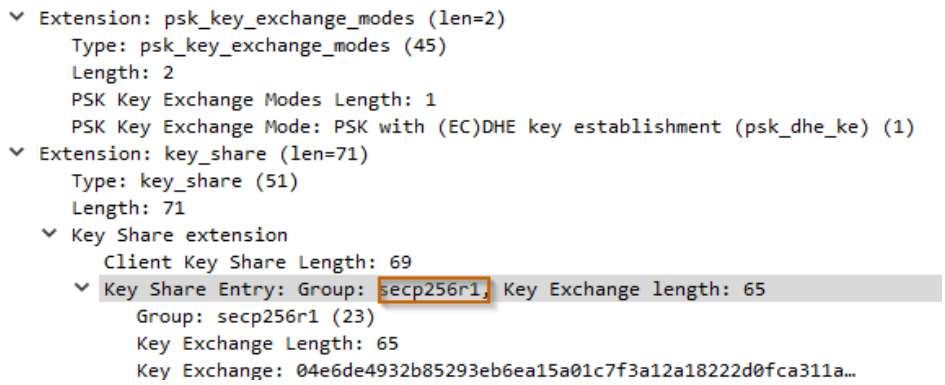
, dok Bob izračunava K:

$$K = \ A^b mod \ p$$

[29]

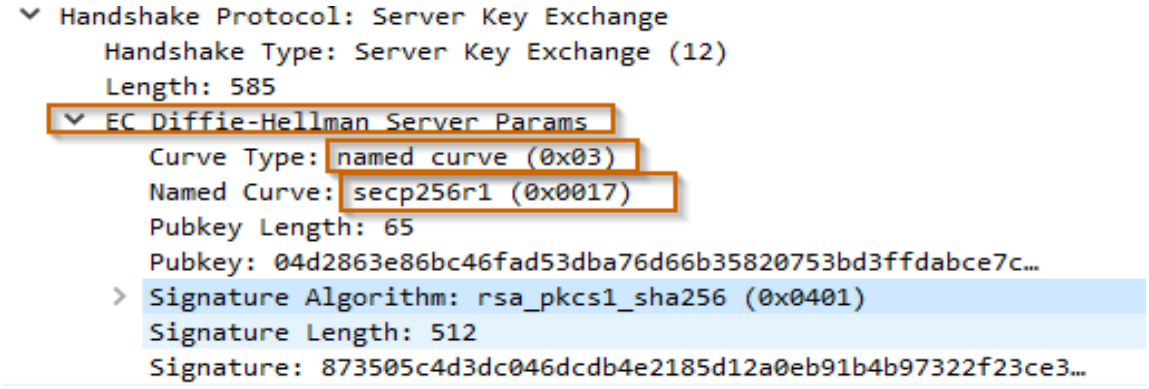
3.1.5.3. ECDH

ECDH (eng. *Elliptic Curve Diffie-Hellman protocol*) jest verzija Diffie-Hellmanovog protokola za razmjenu ključeva koji se temelji na eliptičnim krivuljama. Koncept razmjene je sličan kao kod samog Diffie-Hellman protokola, no koristi se u pozadini drugačija matematička podloga. U proširenjima poruke *ClientHello* (ekstenzije *PSK Key Exchange Modes* i *Key share*) nalaze se parametri kako će se razmjenjivati ključevi te u ovom slučaju koja će se grupa krivulja koristiti:



Slika 21: Proširenja ClientHello poruke u kojem je vidljiv koji algoritam će se koristi za razmjenu ključeva i točno koja grupa će se koristiti

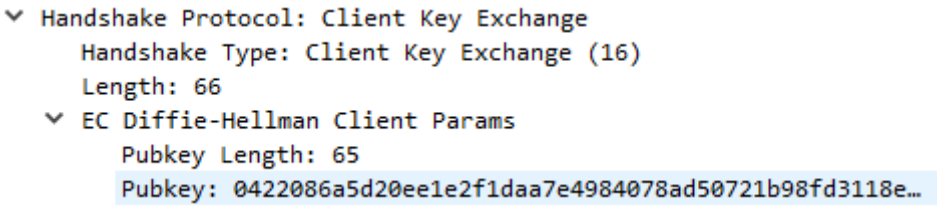
Poslužitelj počinje s razmjenom tako da odabire koja će se specifična eliptična krivulja koristiti za razmjenu te šalje u *ServerKeyExchange* poruci odabranu krivulju i javni parametar EC point ukoliko je ranije bio odabran kako na sljedećem primjeru:



Slika 22: Primjer slanja ECDH parametara prema klijentu zbog razmjene ključeva

Na prethodnom primjer (slika 22), vidljivo je da se šalje odabrana krivulja tipa *named_curve* te imena *secp256r1* pri čemu se ime krivulje može definirati imenom ili OID-om (eng. *unique object identifier*).

Nakon toga, klijent u *ClientKeyExchange* poruci šalje svoje javne parametre:



Slika 23: Javni parametri koje klijent šalje za razmjenu ključeva

Korištenjem predefiniranih parametara, zajedno za ekstenzijama koje se koriste u *ClientHello* poruci što je vidljivo u primjeru te poruke iznad, omogućuje poslužitelju odabrati onu krivulju koju podržavaju obje strane. [27]

Još je preostalo ukratko definirati ekstenziju koja se također vidjela u primjerima *ClientHello* i *ServerHello* poruka, a riječ je o *ec_point_formats* ekstenziji koja omogućava pregovaranje o točki kompresije, koja može poboljšati propusnost zahtjeva prilikom razmjene ključeva. Ipak, u praksi su te uštede ne znatne te se kompresija uglavnom ne koristi. [24]

3.2. PKI – infrastruktura javnih ključeva

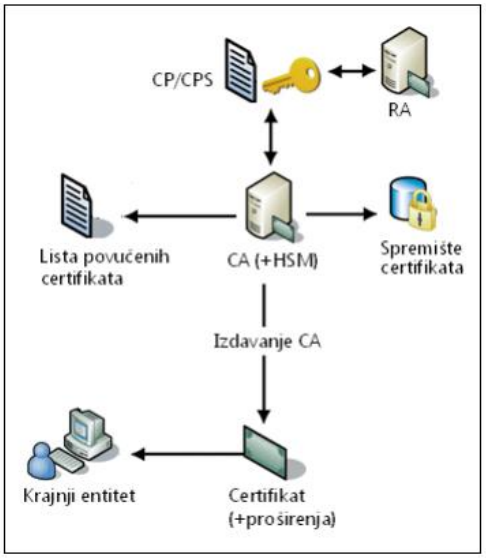
Infrastruktura javnih ključeva, odnosno PKI (eng. *public key infrastructure*) predstavlja skup tehnologija, normi, protokola i usluga koji omogućavaju sigurnu komunikaciju između sudionika temeljenu na sustavu javnim ključa putem nesigurnih kanala. Cilj infrastrukture jest

pružiti ostvarenje sigurnosnih zahtjeva autentičnosti, tajnosti i neporecivosti, dok je sama zadaća sustava povezivanje javnih ključeva sa korisnicima, odnosno koji ključ kome pripada ta provjera jesu li dani ključevi trenutno važeći. [31]

Postoji više ostvarenja sustava infrastrukture javnih ključeva kao PGP, SPKI (eng. *Simple Public Key Infrastructure*) te SDSI (eng. *Simple Distributed Security Infrastructure*). Ovaj rad, kao što mu i naslov govori, baziran je na modelu X509 pa će se u nastavku obrađivati PKI sustavi, njegovi sastavni dijelovi koji su zasnovani na X.509 modelu koji je uspostavljen kao međunarodni standard od strane Međunarodne telekomunikacijske unije (ITU-T). Radna grupa PKIX (eng. *Public-Key Infrastructure X.509*) je osnovana 1995. s ciljem razvoja standarda kako bi se nadogradili PKI sustave bazirane na X.509 modelu.

3.2.1. Arhitektura i dijelovi sustava

Sljedeće što je potrebno jest prikazati i opisati arhitekturu PKI sustava:



Slika 24: Model PKI sustava [32]

PKI sustav se sastoji od slijedećih dijelova kao što je navedeno na slici 24:

1. **Krajnji entitet** – korisnik/entitet (domena) koji se prijavljuje za dobivanje certifikata u svrhu pružanja neke usluge za koju je potrebna sigurnost
2. **Certifikacijsko tijelo**, **CA** (eng. *Certification Authority*) – povezuje javni ključ s pojedinim entitetom (domenom), tj. trećoj strani kojoj se vjeruje. CA je ustvari entitet kojem se vjeruje te on služi kao što je već navedeno za kreiranje i dodjelu javnih ključeva certifikata te sadrži par ključeva (privatni i javni). Neke od funkcija još koje CA provodi su generiranje i povlačenje certifikata na način da CA mora raditi unutar

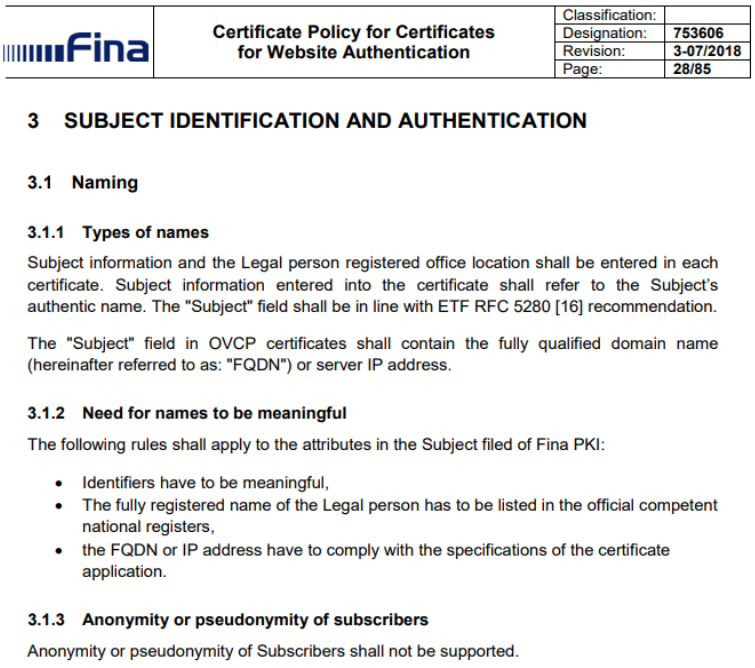
definirane poslovne politike (CP) kao i prema definiranim praksama (CPS).

3. **Registracijsko tijelo**, **RA** (eng. *Registration authority* – neobavezni dio sustava kojemu CA delegira određene funkcije vezane za izdavanje certifikata. Primjerice, neke od funkcija koje može izvoditi RA jest: izvođenje potrebnih validacija identiteta krajnjeg entiteta i određivanje ovlasti za dodjelu javnog ključa prije slanja zahtjeva prema CA za on može izdati certifikat. RA još također mora provesti sve politike i procedure koje su definirane CP-om i CPS-om kako bi se ispitao zahtjev za certifikatom. U praksi se može dogoditi i da CA obavlja zadaće koje su u nadležnosti RA.

4. **Poslovna politika certifikata**, **CP** (eng. *Certificate policy*) – dokument zajednice, tvrtke ili sl. koji označava skup pravila i smjernica koji uključuju primjenjivost javnih ključeva za određenu zajednicu s osnovnim sigurnosnim zahtjevima koji su sadržani unutar ovog dokumenta. U sljedećem dijelu prikazan je primjer naslova sadržaja koji se mogu nalaziti u dokumentu poslovne politike certifikata:

Slika 25: Naslovi sadržaja dokumenta politike pravila certifikata od strane Fine [33]

U sljedećem primjeru je prikazan sadržaj određenog poglavlja dokumenta politike pravila certifikata:



Fina | Certificate Policy for Certificates for Website Authentication | Classification: | | Designation: 753606 | Revision: 3-07/2018 | Page: 28/85

3 SUBJECT IDENTIFICATION AND AUTHENTICATION

3.1 Naming

3.1.1 Types of names

Subject information and the Legal person registered office location shall be entered in each certificate. Subject information entered into the certificate shall refer to the Subject's authentic name. The "Subject" field shall be in line with ETF RFC 5280 [16] recommendation.

The "Subject" field in OVCP certificates shall contain the fully qualified domain name (hereinafter referred to as: "FQDN") or server IP address.

3.1.2 Need for names to be meaningful

The following rules shall apply to the attributes in the Subject filed of Fina PKI:

- Identifiers have to be meaningful,
- The fully registered name of the Legal person has to be listed in the official competent national registers,
- the FQDN or IP address have to comply with the specifications of the certificate application.

3.1.3 Anonymity or pseudonymity of subscribers

Anonymity or pseudonymity of Subscribers shall not be supported.

Slika 26: Primjer sadržaja poglavlja dokumenta politike pravila certifikata od strane Fine [33]

5. **Izjava o praksama certifikata**, **CPS** (eng. *Certificate Practices Statement*) – dokument zajednice, tvrtke ili sl. koji označava prakse koji CA uključuje u izdavanje ključeva te sve procese koji spadaju pod njegovu domenu kao što su: generiranje, povlačenje, pohrana i upravljanje. Svaka implementacija PKI sustava treba sadržavati svrhu, doseg, posebne poslovne zahtjeve, sigurnosnu arhitekturu, odgovarajući model povjerenja te posebne sigurnosne usluge koji podržavaju navedeni PKI sustav.

Na sljedećoj slici naveden je primjer poglavlja koji govori o reduciranju i zabrani certifikata *OCSP* protokolom: *RDC 2015*:

4.9.9. Online revocation/status checking availability

Fina CAs support online status check of issued certificates revocation via Fina OCSP 2015 service operating based on OCSP protocol.

Information on the certificate revocation status via Fina's OCSP 2015 service is available in real time.

Fina OCSP 2015 service address is http://ocsp.fina.hr, and it is entered into the Authority Information Access extension of each certificate issued by Fina CAs.

CRL is available primarily through HTTP Internet address on the server of the corresponding repository, and secondarily through LDAP Directory, as described in Section 4.10.1. of this CPS_{OC} document. Data on access points for retrieving CRL are contained in each issued certificate.

Slika 27: Pod poglavlje dokumenta o praksama certifikata *FinaRDC 2015* koje govori načinu provjere reduciranosti certifikata OCSP protokolom [34]

6. **Baza izdanih certifikata** (eng. *Data storage area*) – spremište izdanih certifikata, za koje se najčešće koristi LDAP imenički servis koji služi za daljnju distribuciju certifikata te tim spremištem upravljaju RA i CA. Time je sam proces distribucije pojednostavljen, jer je prilikom izdavanja novog certifikata, ili prilikom reduciranja ili pak neke promjene potrebno samo ažurirati zapise o tom certifikatu koji se nalazi u spremištu.
7. **Lista reduciranih (opozvanih) certifikata** – **CRL** (eng. *Certificate Revocation List*) – sadrži popis svih opozvanih certifikata sustava te tim popisom upravljaju zajedno RA te CA. CRL je samo jedna od mogućih metoda provjera reduciranosti certifikata.
   [32]

3.2.2. Uporaba i prednosti

U nastavku će biti opisano jednostavnim rječnikom kako funkcionira PKI sustav, gdje se sve može koristiti te će se navesti njegove prednosti.

Kako je već spominjano kroz ovaj rad, ponovo zamislimo sudionike u komunikaciji, *Alice* i *Boba*. Prvo *Alice* generira svoj par javnog i privatnog ključa te predstavlja svoj javni ključ prema CA kako bi se predstavila PKI sustavu. CA tada potvrđuje autentičnost pomoću danog javnog ključa od Alice te potpisuje izjavu na način da izdaje certifikat koji potvrđuje da dani javni ključ stvarno pripada Alice. Ukoliko onda Alice želi uspostaviti komunikaciju sa Bobom, tada mu šalje također svoj javni ključ i izdani certifikat. S druge strane, Bob sadrži javni ključ od CA pa pomoću njega može potvrditi potpis koji se nalazi u certifikatu. Analogno navedenome, ukoliko Bob vjeruje CA, tada također vjeruje i u autentičnost od strane Alice, tj. da njezin javni ključ stvarno pripada njoj. Također se ista procedura provodi i ukoliko Bob komunicira sa Alice te šalje svoj javni ključ i certifikat prema njoj. Tada su ključevi međusobno razmijenjeni što je dio TLS rukovanja koji je već ranije bio detaljno opisan te je sada onda moguće dalje napraviti izvođenje ključeva kako bi se mogla uspostaviti sigurna veza. [35]

Zaključno i ukratko, u PKI infrastrukturi postoji centralni autoritet (CA) kojem svi sudionici komunikacije vjeruju te tada CA treba ovjeriti, odnosno certificirati javne ključeve strana koje žele međusobno komunicirati te im šalje svoj javni ključ. Kada su ovi koraci napravljeni i nakon izvođenja ključeva, tada je uspostavljenja sigurna veza između strana koje žele međusobno komunicirati.

PKI osigurava primjenu sigurnosnih mehanizama za niz usluga kao primjerice:

1. **usluga razmjene poruka putem elektroničke pošte**: slična procedura kako je navedena prilikom komunikacije između strana se dešava i kod slanja kriptirane email pošte drugoj strani. Kada se procedura izvrši, odnosno kad se potvrde identiteti i svaka strana dobije certifikat od CA, tada strane međusobno dijele javne ključeve slanjem potpisane email poruke. Na kraju, kada se razmijene javni ključevi između strana, tada one jedna drugoj vjeruju i mogu razmjenjivati kriptirane email poruke. Na umu još treba imati da posjedovanje certifikata ne dozvoljava nekome slanje kriptiranih email, već samo primanje istih. [36]
2. **pristup putem VPN**-a: administrator koji postavlja VPN pristup, daje svakom zaposleniku certifikat koji omogućava točki VPN pristupa da prepozna tog zaposlenika. [35]
3. **rad sa senzorima u industriji:** podaci koji daju senzori se mogu modificirati pa je potrebno ukloniti sve sigurnosne rizike koji se mogu pojaviti kao što je npr. neovlaštena promjena podataka. Stoga se također koristi PKI infrastruktura za metode autentifikacije i zaštite neovlaštenog pristupa senzorima te promjene podataka. Tvrtka se u tom slučaju ponaša kao centralni autoritet te je postavlja PKI sustave na svaki senzor, tako da svaki senzor može biti prepoznat i osiguran od strane kontrolne sobe. [35]
4. **elektroničko bankarstvo:** banka treba dozvoliti svojim korisnicima da vrše financijske transakcije te ih mora također identificirati. Stoga, banka se ponaša kao CA te izdaje javne ključeve svojim klijentima. [35]
5. **elektroničke transakcije kreditnim karticama:** Ukoliko klijent u trgovini želi platiti robu kreditnom karticom koja pripada jednoj banci, a trgovac ima račun u drugoj banci. Tada između banaka je također potrebno uspostaviti PKI sustav zbog sigurne komunikacije. U ovoj situaciji kreditna organizacija koja izdaje kreditne kartice ponaša se kao CA te izdaje certifikate pojedinim bankama. [35]
6. **razne vrste autentifikacije i provjere podataka itd.**

PKI sustav surađuje sa sljedećim protokolima:

1. SSL/TLS, HTTP te IPsec (grupa protokola koja se koristi za uspostavljanje kriptirane veze između uređaja, često se koristi za uspostavljanje VPN-a) protokol za komunikaciju i sigurnost [37]

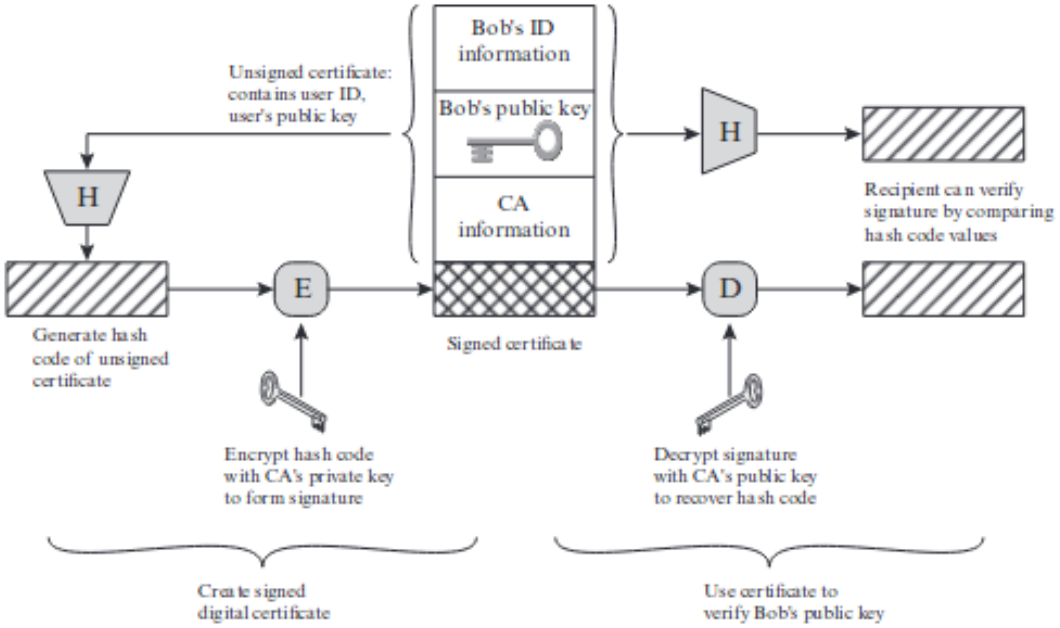
2. S/MIME (eng. *Secure/Multipurpose Internet Mail Extension*) i PGP (eng. *Pretty Good Privacy*) protokole za sigurnost razmjene poruka elektroničke pošte – S/MIME protokol predstavlja sigurnosno proširenje MIME protokola/stanarda, koji je baziran na tehnologiji *RSA Dana Security* te je ovaj protokol dosta srodan PGP protokolu. PGP je paket otvorenog koda koji također pruža sigurnost email poruka. PGP omogućava određene sigurnosne zahtjeve poput autentifikacije korištenjem digitalnom potpisa, povjerljivosti simetričnim kriptiranjem blokova podataka te mogućnosti kompresije, segmentacije i kompatibilnost email poruka. [38]
3. SET (eng. *Secure Electronic Transaction*) protokol za razmjenu vrijednosti (eng. *value exchange*)
4. podršku za B2B (eng. *Business-to-business*) poslovanje
   [32]

3.3. X509 Certifikati

U uvodnom dijelu ovog poglavlja dati će se kratki opis X.509 norme koja propisuje sadržaj certifikata i definira autentifikacijske usluge elektroničkih imenika korisnicima.. X.509 norma je dio serije preporuka X.500. X.500 serije preporuka definiraju usluge elektroničkih imenika, odnosno poslužitelj ili distribuirani skup poslužitelja koji sadrže bazu znanja o korisnicima. Norma X.509 je bazirana na kriptografiji javnog ključa te korištenjem digitalnih potpisa te određuje format zapisa, semantiku i sadržaj pojedinih polja, numeričke identifikatore pojedinih polja te normira proširenja . [39]

Digitalni certifikat predstavlja digitalni dokument koji sadrži javni ključ, informacije o entitetu/domeni za koju je izdan te informacije i digitalni potpis o izdavatelju tog certifikata. Certifikati omogućavaju korištenje, razmjenu i spremanje javnih ključeva te oni predstavljaju najbitniji dio PKIX sustava. [40]

Na sljedećoj slici je prikazan je način koji se certifikat koristi unutar PKI strukture što je već opisano u pod poglavlju *3.2.2 Uporaba i prednosti*:

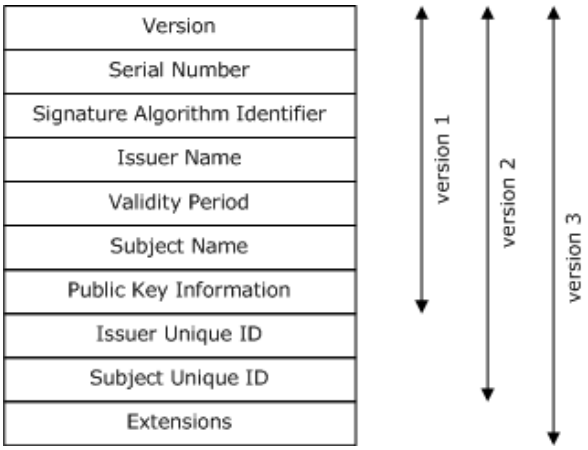


Slika 28: Uporaba potpisanog javnog ključa, odnosno certifikata unutar PKI sustava prilikom uspostavljanja sigurne veze [41]

3.3.1. Povijest verzija

Sustav za razmjenu ključeva je već naveden u obliku Diffie-Hellman pristupa. Ipak, kod toga tog načina je bio problem što je navedeni pristup bio ograničen na sudionike komunikacije iz istog okruženja. Stoga, za uspostavljanje sigurne komunikacije između sudionika iz različitih okolina L. Kohnfelder predložio je 1978. koncept digitalnog certifikata. [42]

Prvi format koji je postao standard bio je kreiran 1988. godine i bio je nazvan verzijom 1 (v1) sa osnovnim poljima. 1993. godine je X.500 norma bila revidirana te su dodana 2 novlja kao što je vidljivo na slici 29: *Issue Unique ID* te *Subject Unique* ID. Na istoj slici ukratko je prikazan razvoj strukture i dodavanje novih polja u X.509 certifikat kroz verzije:



Slika 29: Razvoj X.509 certifikata kroz verzije [43]

Kasnije, u lipnju 1996. godine, dovršena je verzija v3 koja je postala standard jer se zbog pojave **PEM** formata javila potreba za pohranjivanje više informacija i implementiranje novih funkcionalnosti. PEM format (eng. *Privacy-Enchanced Mail*) koji se koristi kao datotečni format za spremanje ključeva i certifikata kreiran je 1993 godine. Verzija v3 proširuje v2 format dodavanjem dodatnih ekstenzijskih polja koja mogu biti definirana prema nekom standardu ili mogu biti prilagođena nekoj organizaciji te namjeni. [44]

3.3.2. Struktura X.509 v3 certifikata

U uvodnom dijelu će se obraditi formati i notacija koji koriste certifikati. Formalni zapis certifikata dan koristi **ASN.1** notaciju te koristi izdvojenim pravilima (**DER** i **PEM** kodiranje), a u nastavku će ovi pojmovi još detaljnije objasniti.

3.3.2.1. ASN.1 notacija i DER, PEM pravila kodiranja

ASN.1 notacija (eng. *Abstract Syntax Notation One*) predstavlja set pravila koji podržavaju definiranje, transport i razmjenu kompleksnih tipova podataka i objekata te je dizajniran da podržava mrežnu komunikaciju između raznih sustava neovisno arhitekturi i implementaciji sustava, odnosno korištenom programskog jeziku. Navedena notacija definira podatke na apstraktan način te određuje kako će oni biti kriptirani. Primjer ASN.1 sintakse certifikata verzije v3 dan je na slici ispod:

```
CertificateToBeSigned ::= SEQUENCE
{
  version                 [0] CertificateVersion DEFAULT v1,
  serialNumber            CertificateSerialNumber,
  signature               AlgorithmIdentifier,
  issuer                  Name
  validity                Validity,
  subject                 Name
  subjectPublicKeyInfo    SubjectPublicKeyInfo,
  issuerUniqueIdentifier  [1] IMPLICIT UniqueIdentifier OPTIONAL,
  subjectUniqueIdentifier [2] IMPLICIT UniqueIdentifier OPTIONAL,
  extensions              [3] Extensions OPTIONAL
}
```

Slika 30: Sintaksa X.509 v3 certifikata kojeg je potrebno potpisati

Notacija sadrži neke predefinirane tipove podataka poput:

1. Cijelih brojeva (INTEGER)
2. Logičkih tipova podataka (BOOLEAN)
3. Znakovnih nizova (IA5String, UniversalString itd.)
4. Bitovni nizovi (BIT STRING)
5. Itd.

Također, moguće je i definirati kompleksne tipove poput:

1. Struktura (SEQUENCE)

2. Lista (SEQUENCE OF)
3. Izbor između tipova podataka (CHOICE)
4. Itd.

[45]

Notacija još koristi OID-ove (eng. *Object Identifier*) kao brojeve koji su jedinstveni, tj. jednoznačno određuju klasu objekata ili atribut. OID-ovi su hijerarhijski organizirani, tako npr. centralni autoriteti na nacionalnoj razini izdaju OID-ove pojedincima ili organizacijama, koje dalje tada mogu dalje svoje OID-ove izdavati dalje unutar lanca certifikata. OID izgleda u ovakvom formatu te je ovo primjer OID-a koji specificira algoritam sažimanja: `1.2.840.113549.1.1.5`. [46]

DER (*eng. Distinguished Encoding Rules*) je pravilo kodiranja temeljeno na BER (eng. *Basic Encoding Rules*) na kojem se temelji X.509 norma te se koristi prilikom digitalnog potpisivanja i određuje samo jedan način za kodiranje ASN.1 vrijednosti, kada je potrebno jednoznačno kriptiranje podataka. Kasnije će se još detaljno vidjeti kako se podaci kodiraju u binarnom obliku za certifikate i ključeve. *PEM* kodiranje temelji se na ASCII kodiranju koristeći Base64 kodiranje gdje se dobije niz znakova. [40]

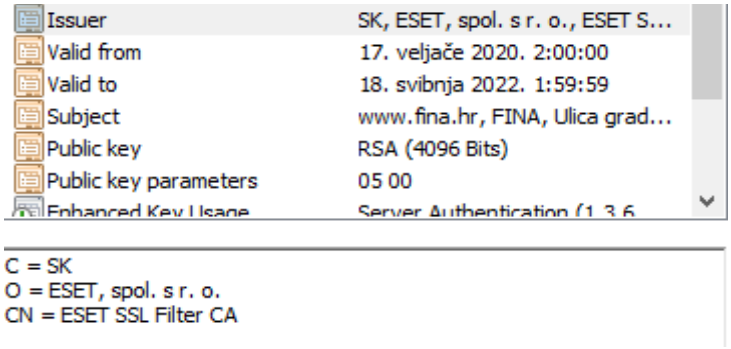
3.3.2.2. Osnovna polja

1. **Version** – ovo polje označava broj verzije kreiranog certifikata, a sintaksa je slijedeća: `CertificateVersion ::= INTEGER {v1(0), v2(1), v3(2)}`, što znači ukoliko je vrijednost 0 tada je verzija certifikata v1, ukoliko je vrijednost, tada je verzija v2 i za vrijednost 2, verzija certifikata je v3.
2. **Serial number** –polje sadržava jedinstvenu pozitivnu vrijednost koja jednoznačno identificira certifikat izdan od CA. Sintaksa:
   `CertificateSerialNumber ::= INTEGER`
3. **Signature Algorithm** –sadrži OID koji specificira algoritam koji CA koristi za potpisivanje certifikata. Stavlja se kao polje unutar certifikata kako bi moglo biti zaštićeno potpisom. Također, uz ovo polje još se u certifikatu dodaje i OID 1.2.840.113549.1.1.5 koji označava algoritam sažimanja prilikom potpisivanja certifikata.
4. **Issuer** – polje sadrži X.500 podatke o CA koji je kreirao i potpisao certifikat, te je ASN.1 sintaksa zadana ovako:

```
Name ::= SEQUENCE OF RelativeDistinguishedName

RelativeDistinguishedName ::= SET OF AttributeTypeValue

AttributeTypeValue ::= SEQUENCE
{
  type       OBJECT IDENTIFIER,
```

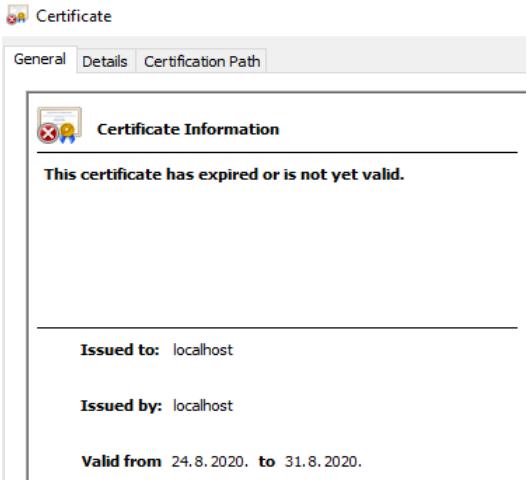
```
  value       ANY
}
```

Sintaksa omogućava dodavanje puno OID-ova, a neki od najviše korištenih su: CN (eng. *Common name*), O (eng. *Organization*), C (eng. *Country*), OU (eng. *Organizational Unit Name*) kao što je prikazano na primjer ispod iz već navedenog Fininog certifikata koji se nalazi na njihovoj fina.hr domeni:



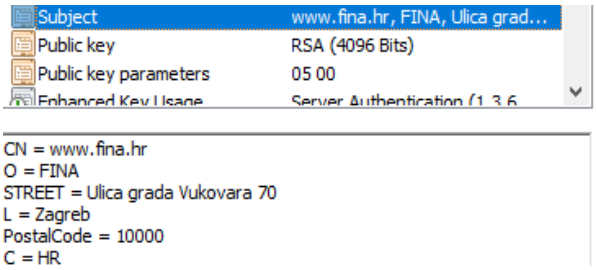
Slika 31: prikaz sadržaja *Issuer* polja [autorska izrada]

5. **Validity** - polje koje se sastoji od 2 datuma koji označuju interval u kojima je certifikat valjan. Na slici ispod prikazan je certifikat kojem je istekla valjanost:



Slika 32: Prikaz certifikata kojem je istekla valjanost [autorska izrada]

6. **Subject** - sadrži X.500 podatke o entitetu za kojeg je potpisan javni ključ u obliku certifikata. Sintaksa je slična kao i kod *Issuer* polja te sadrži također slične tipove podataka kao i *Issuer* s nekim dodatnim poljima kao ovdje na primjer gdje je još dodan poštanski broj i ulica:



Slika 33: Sadržaj polja *Subject* - [autorska izrada]

7. **Public key** – sadrži javni ključ i navedeni algoritam koji se koristi za potpisivanje kako je navedeno u sintaksi ASN.1 notacije:

```
SubjectPublicKeyInfo ::= SEQUENCE
{
  algorithm            AlgorithmIdentifier,
  subjectPublicKey     BITSTRING
}

AlgorithmIdentifier ::= SEQUENCE
{
  algorithm            OBJECT IDENTIFIER,
  parameters           ANY OPTIONAL
}
```

[40] [47]

3.3.2.3. Polja verzije v2

Ovdje će biti opisana nova polja koja su dodana u verziji v2.:

1. **Issue Unique Identifier** – sadrži bit string polje koje označuje jedinstvenu vrijednost za identificiranje CA koji je izdao certifikat u slučaju da se njegovo X.500 Issuer polje koristilo za entitete/domene. Primjer vrijednosti:

`7ee0a8c4b3fc9c98fccbd9319f554dccb1d95a90`

2. **Subject unique identifier** - sadrži bit string polje koje označuje jedinstvenu vrijednost za identificiranje entiteta za kojeg je izdan certifikat u slučaju da se njegovo X.500 Subject polje koristilo za različite entitete/domene
   [39]

3.3.2.4. Proširenja verzije v3

Navedena polja verzije v2 su u verziji v3 revidirana i stavljena kao proširenja, a ne kao polja te imaju nove nazive*: Authority Key Identifier* and *Subject Key Identifier*.

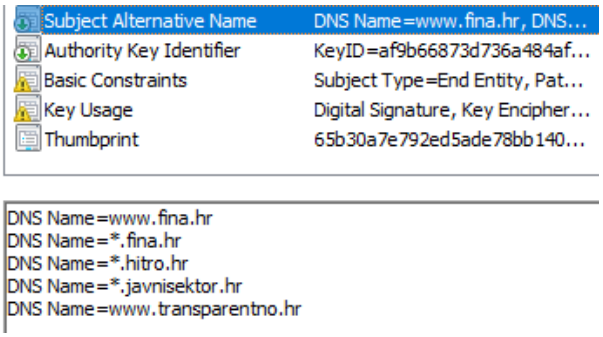
Proširenja su dodana za davanje novih mogućnosti i fleksibilnosti da se ne treba mijenjati sam format certifikata. Svako proširenje (ekstenzija) sadrži svoj OID i vrijednost unutar ASN.1 sintakse.

Proširenja su dana slijedećom sintaksom:

```
Extension ::= SEQUENCE
{
  Id                    OBJECT IDENTIFIER,
  critical              BOOLEAN DEFAULT FALSE,
  extnValue             OCTET STRING
}
```

Dodane ekstenzije su redom opisane prema kategorijama:

1. **Certificate Subject and Issuer Attributes** – ova grupa ekstenzija podržava alternativa imena, formate, te pruža dodatne informacije o izdavatelju certifikata i entitetu za kojeg se izdaje certifikat
    a. **Subject alternative name**: ovo proširenje je uvedeno zbog toga jer *Subject* polje nije dovoljno fleksibilno za povezivanje entiteta sa javnim ključem, zbog toga jer podržava samo imena računala (eng. *hostnames*) pa ne može se koristiti za povezivanje javnih ključeva na više entiteta. Zbog toga je uvedena ova ekstenzija koja podržava spajanje više entiteta definiranih DNS imenom, IP adresom ili URI-em. Na slici ispod prikazan je primjer gdje se koristi ekstenzija za više domena definiranih DNS imenom:



Slika 34: Primjer ekstenzije Subject Alternative Name

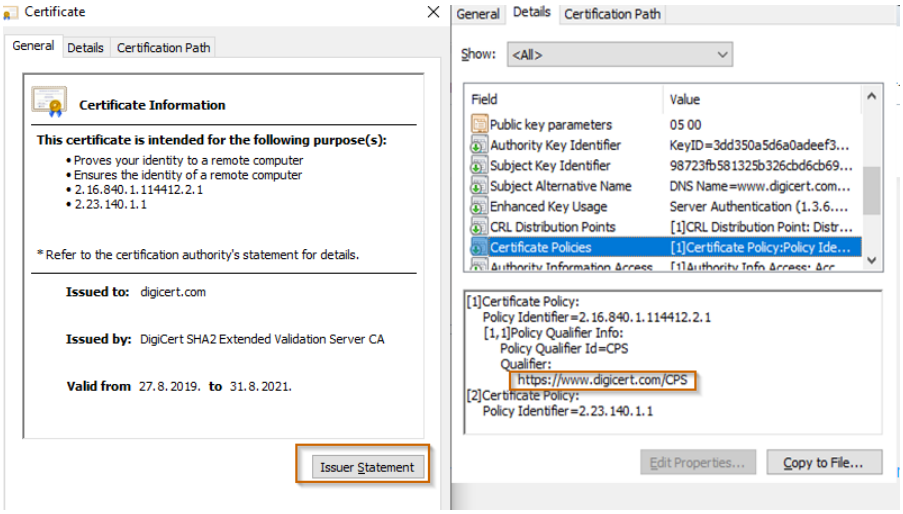
    b. **Issuer Alternative Name**: također slično kao i za prethodno navedeno proširenje, ovo proširenje definira jedno ili više imena izdavatelja iz certifikacijskog zahtjeva. Međutim, ovo proširenje se rijetko koristi.
    c. **Subject directory atributes:** prenosi identifikacijske atribute kao što je državljanstvo subjekta potvrde. Vrijednost proširenja je niz parova OID vrijednosti. Također se rijetko koristi ovo proširenje.
2. **Key and Policy Information**: ova grupa proširenja sadrže dodatne informacije o ključevima subjekta kojem je izdan certifikat te izdavača certifikata, kao i poveznice prema politikama certifikata o čemu je također bilo riječ u prethodnim poglavljima (3.2.1). Unutar ove grupe proširenja nalaze se slijedeća bitna proširenja kao što su:

a. **Authority key identifier**: proširenje sadrži jedinstvenu vrijednost privatnog ključa koji je koristio CA za potpisivanje izdanog certifikata te se može koristiti prilikom kreiranja lanca za identificiranje izdavateljskog certifikata. Svi certifikati koji su izdani moraju sadržavati ovo polje.
b. **Subject key identifier**: ovo proširenje također sadrži jedinstvenu vrijednost koja se koristi za identificiranje certifikata kojemu pripada javni ključ pomoću kojeg je potpisan. Preporučeno je da se ovaj identifikator konstruira sažimanjem. Certifikati koji su izdani od CA moraju sadržavati ovaj identifikator u *Authority Key Identifier* proširenju za sve svoje izdane certifikate.
c. **Key usage**: proširenje sadrži svrhu korištenja ključeva koja se razlikuje od certifikata do certifikata i da li se radi od strane entitetskog certifikata ili od certifikata koji izdaje ostale. Danom ASN.1 sintaksom vidljivo je koja je vrijednost postavljena unutar ovog proširenja, **podebljano** označeno:

```
KeyUsage ::= BIT STRING {
     digitalSignature        (0),
     nonRepudiation          (1), -- recent editions of -
- renamed this bit to X.509 have contentCommitment

     keyEncipherment         (2),
     dataEncipherment        (3),
     keyAgreement            (4),
     keyCertSign             (5),
     cRLSign                 (6),
     encipherOnly            (7),
     decipherOnly            (8) }
```

[48]

d. **Certificate policies**: proširenje sadrži listu politike certifikata i niza pravila koji se primjenjuju u radi i u tom PKI sustavu unutar kojeg je uključen navedeni certifikat koji vrijedi za danu organizaciju. Na sljedećem primjeri vidljivo je da se ovo proširenje sadrži od OID-ova te je opcionalno *qualifier* koji označava vrijednost, odnosno URI gdje se nalazi tekst politike certifikata:



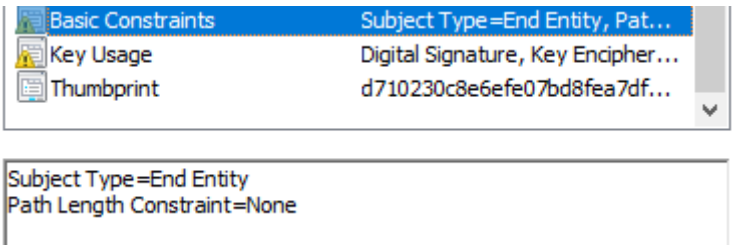
Slika 35: Primjer vrijednosti ekstenzija koji sadrži URI na popis politika koje su određene za danu organizaciju [vlastita izrada]

Klikom na *Issuer Statement* otvara se označeni URI (https://www.digicert.com/CPS) sa popisom politika sa slike 34.

3. **Certification path constraints**: ova grupa proširenja označava razna ograničenja koja se koriste u certifikatu, ovisno da li je certifikat potpisni ili ga se izdaje pa tako su ovdje najbitnija proširenja:
    a. **Basic contraints**: ograničenje koje označava da li je certifikat korišten kao CA ili je to krajnji entitet. Ukoliko nije, tada u lancu certifikata za može biti još certifikata hijerarhijski u razini ispod. Sintaksa je dana:

```
BasicConstraints ::= SEQUENCE
{
        cA                      BOOLEAN DEFAULT FALSE,
        pathLenConstraint       INTEGER (0..MAX) OPTIONAL
}
```

Što znači dano slijedećim primjerom ispod da ukoliko je End Entity vrijednost *Subject Type*-a i nema vrijednosti *Path Length Constraint*, tada certifikat nije CA, dok u suprotnom jest kao što je prikazano na primjeru ispod:



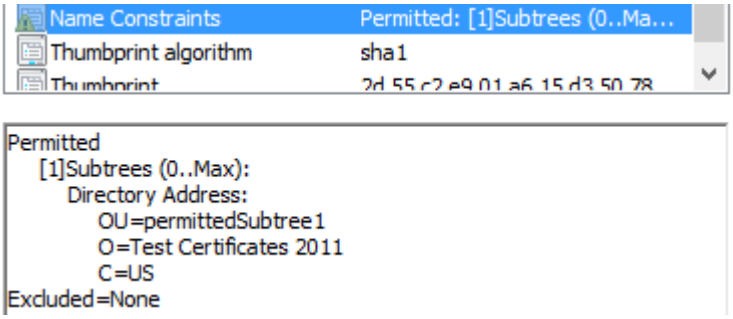
Slika 36: Primjer Basic Constraints proširenja [autorska izrada]

Vrijednost tipa *Path Lenght Constraint* označava broj koliko još CA dodatnih pod certifikata može biti u hijerarhiji ispod navedenog. To znači ukoliko je

vrijednost 0 tada više nije dozvoljen niti jedan CA certifikat ispod u hijerarhiji i uz to, ukoliko je vrijednost *SubjectType* jednaka *CA*. [49]

b. **Name constraints**: sadrži ograničenja identiteta za kojeg CA može izdavati certifikate. Određeni imenski prostor može se zabraniti, što je vrlo korisno jer se time može nekoj organizaciji odrediti da može izdavati dalje certifikati samo za domenu za koju joj je zadano. Ograničenja se izvode pomoću:
   - **i.** imena direktorija (OU, O, C) – ove skraćenice su već objašnjene u poglavlju 3.3.2.2.
   - **ii.** DNS imena
   - **iii.** Email adresa
   - **iv.** URI-a
   - **v.** IP adresa

   te vrijednosti ove ekstenzije dane su sljedećim primjerom:



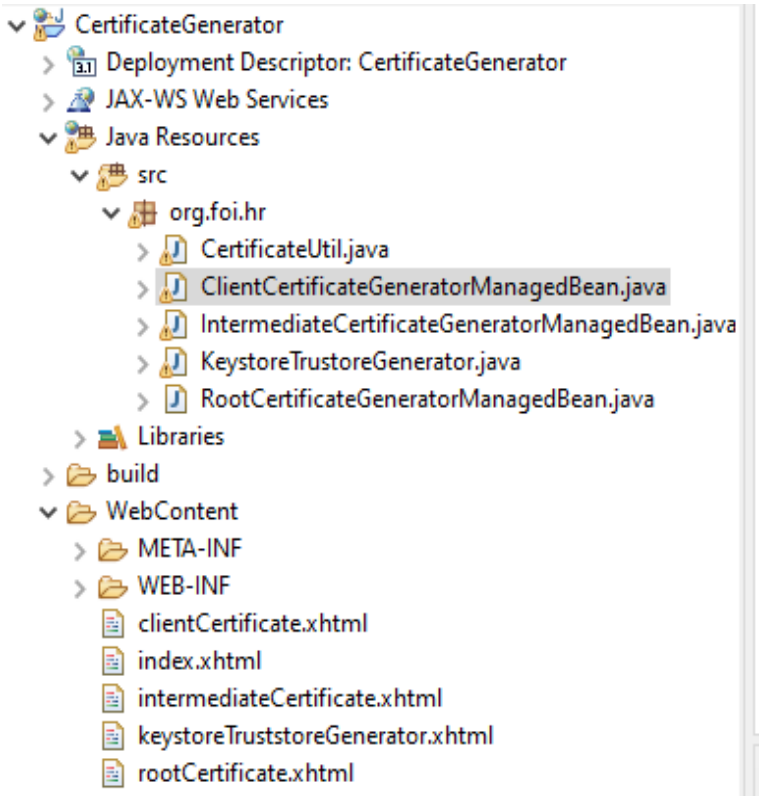
Slika 37: Primjer Name Constraints proširenja [autorska izrada]

4. Praktični dio

U praktičnom dijelu biti će opisano način na koji je napravljena aplikacija za generiranje lanaca certifikata, korak po korak te će se još dati dodatno opis pojmova koji nisu bili razrađeni u teoretskom dijelu.

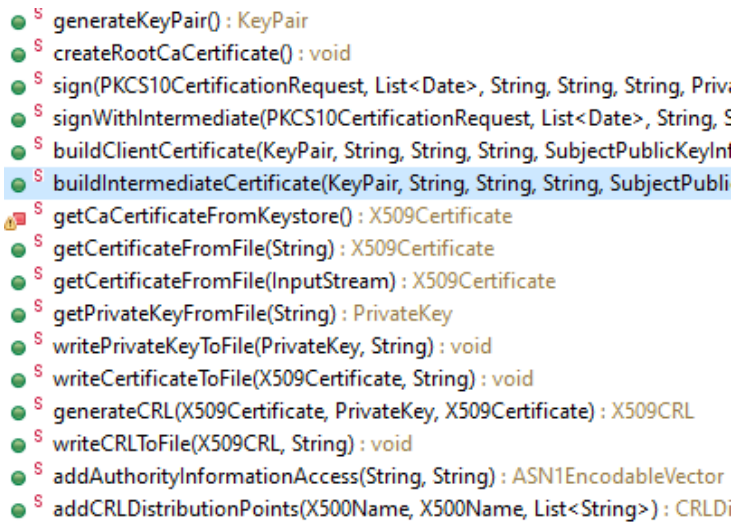
4.1. Generator lanca certifikata

Sama aplikacija rađena je kao *Dynamic Web Project*, odnosno kao Web aplikacija koja se sastoji od više web stranica koje su rađene pomoću JSF komponenti (eng. *Java Server Faces*). JSF je web aplikacijski okvir koji pojednostavljuje izradu web korisničkih sučelja, odnosno za prikaz komponenata na ekranu (front-end dio). [50] Na sljedećoj slici prikazana je struktura projekta aplikacije generatora lanaca certifikata:



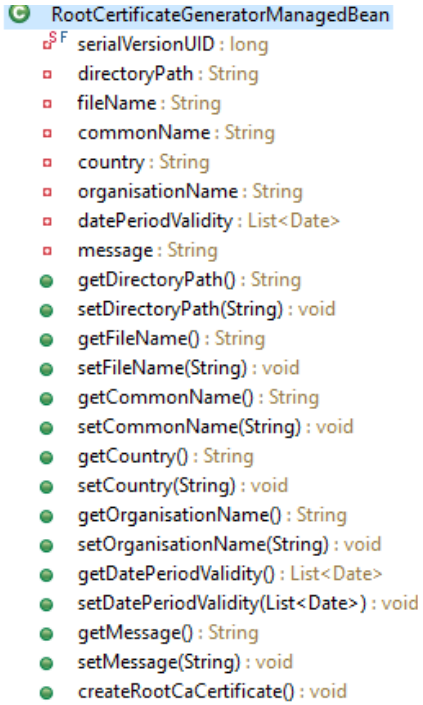
Slika 38: Struktura projekta aplikacije za generiranje certifikata u Eclipse IDE [autorska izrada]

CertificateUtil klasa jest pomoćna klasa u kojoj su sadržane mnoge metode koje sam koristio kroz preostale klasa (upravljačka zrna) kao poput metoda digitalnog potpisivanja, dobivanje certifikata i privatnog ključa iz datoteke, ispisa certifikata i ključeva u datoteku i sl. Ovakav način odabran je zbog optimizacije i iskoristivosti istog koda na više mjesta (reuseability). Pun popis dan metoda pomoćne klase dan je na slici ispod:



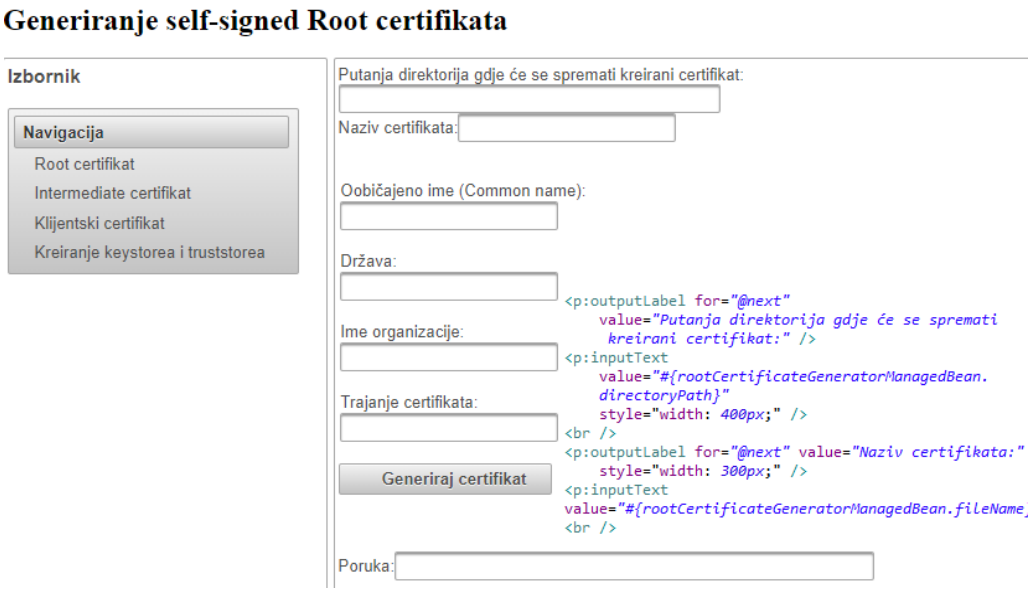
Slika 39: Popis svih metoda pomoćne (helper) klase CertificateUtil [autorska izrada]

Preostale klase unutar paketa su ustvari upravljačka zrna koja sadrže poslovnu logiku koja će se izvršavati te također predstavlja model podataka te sa UI komponenti (web stranica) prima podatke i šalje rezultate natrag na zaslon korisnika. Svako zrno zasebno zadrži svoju vlastitu logiku za kreiranje svojeg certifikata ili keystorea. Na sljedećoj slici je tako prikazan popis atributa koji služe za model te popis metoda koje izvršavaju neku poslovnu logiku na primjer zrna za generiranje krovnog certifikata (eng. *Root certificate*):



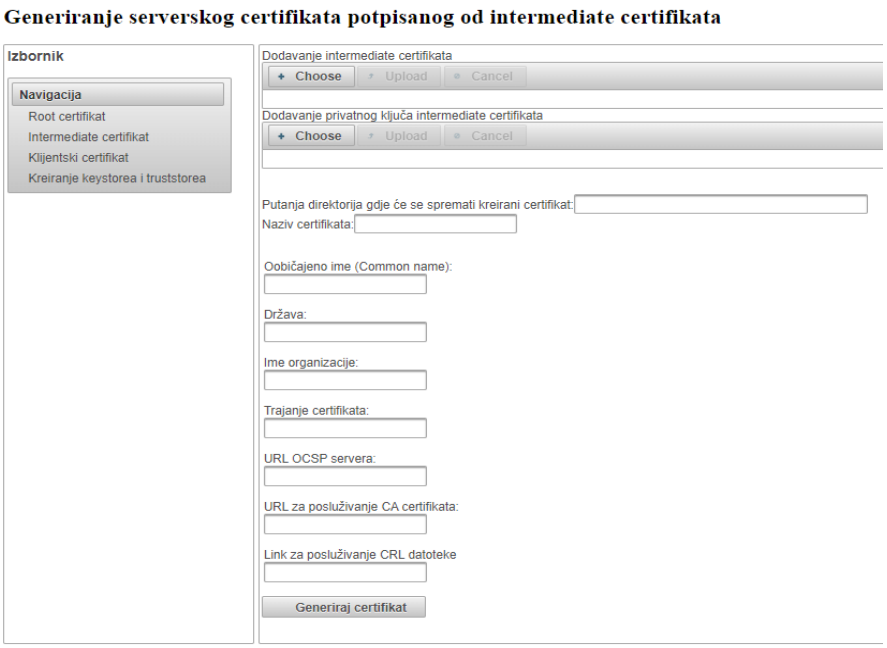
Slika 40: Popis atributa i metoda upravljačkog zrna RootCertificateGeneratorManagedBean [autorska izrada]

Preostale .xhtml datoteke služe za prikaz komponenti i rezultata podataka te za unos vrijednosti, a u nastavku su prikazani isječak koda i prikaz stranice *keystoreTrustoreGenerator.xhtml*:



Slika 41: Prikaz stranice i isječka koda funkcionalnosti za generiranje krovnog certifikata [autorska izrada]

Važno je još napomenuti da se za front-end, odnosno prikaz komponenti koristila *Primefaces* dodatna biblioteka za JSF koja je otvorenog koda te je besplatna za korištenje. [51] Sama aplikacija pokretana je na Tomcat 8.5.57 verziji. Tomcat je Java servlet kontejner koji služi pokretanje i održavanje životnog ciklusa aplikacija baziranih na Java Servlet i Java Server Pages (JSP) tehnologijama. [52]



Slika 42: Prikaz izgleda aplikacije za generiranje certifikata [autorska izrada]

4.1.1. Samo-potpisni krovni certifikat

Prvi korak prilikom generiranja lanca certifikata bio je kreirati samo-potpisni (*eng. Self-Signed*) krovni certifikat. Prije toga, objasniti će se uopće što znači pojam lanac certifikata dok je sam primjer prikaza lanca prikazan na *Slici 15*.

Lanac certifikata predstavlja listu certifikata: od krovnog, pa dalje opcionalno pod razinom ispod može biti potpisni certifikat (eng. *Intermediate CA Certificate*) kojeg izdaje krovni CA (eng. *root CA*) za izdavanje certifikata krajnjim korisnicima te su ti izdani certifikati zadnje razine klijentski (eng. *end-entity*). [53]

Samo-potpisni certifikat znači da je to certifikat koji nije potpisan od strane nekog autoriteta (CA) već su to certifikati koji su potpisani od vlasnika domene za kojeg se izdaju, odnosno potpisuju sami sebe (polja issuer i subject su jednaka). Stoga, autentifikacija za takve certifikate nije omogućena. Takvi certifikati se koristite za testne svrhe te web preglednici izdaju upozorenje da certifikat nije verificiran prilikom ulaska na domene koje posjeduju takav tip certifikata.

Ipak, praksa u stvarnosti jest da su krovni certifikati izdani od strane centralnih autoriteta, odnosno od organizacija koje su vjerodajne te koje izdaju certifikate i javne ključeve. Neki od svjetskih najpoznatijih izdavatelja certifikata koji nude svoje usluge izdavanja su: *Comodo SSL, DigiCert*, *GlobalSign, RapidSSL, SSL.com* te besplatni pružatelj usluga izdavanja certifikata *Let's Encrypt*. [54] U Republici Hrvatskoj usluge pružanja izdavanja certifikata nude: FINA (Financijska agencija) te AKD (Agencija za komercijalnu djelatnost proizvodno, uslužno i trgovačko d.o.o.).

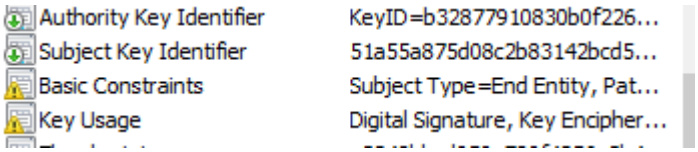
Sada kada je opisan samo-potpisni krovni certifikat, može se opisati po koracima način na koji je generiran takav certifikat. Na slici ispod nalazi se programski kod koji koristeći BouncyCastle kriptografski API te *java.security* paket omogućuje generiranje samo-potpisnog krovnog certifikata:

```
KeyPair kp = CertificateUtil.generateKeyPair();
byte[] pk = kp.getPublic().getEncoded();

SubjectPublicKeyInfo bcPk = SubjectPublicKeyInfo.getInstance(pk);
String subjectDN = "C="+country +", CN="+commonName +", O="+organisationName;
X500Name subjectDNName = new X500Name(subjectDN);

X509v3CertificateBuilder certGen = new X509v3CertificateBuilder(subjectDNName, BigInteger.ONE,
        datePeriodValidity.get(0),datePeriodValidity.get(1), subjectDNName, bcPk);

BasicConstraints basicConstraints = new BasicConstraints(true);
// Basic Constraints is usually marked as critical.
try {
    certGen.addExtension(new ASN1ObjectIdentifier("2.5.29.19"), true, basicConstraints);
} catch (CertIOException e2) {
}

// adding keyUsage extension
KeyUsage keyUsage = new KeyUsage(KeyUsage.keyCertSign | KeyUsage.cRLSign);
try {
    certGen.addExtension(Extension.subjectKeyIdentifier, false,
            new JcaX509ExtensionUtils().createSubjectKeyIdentifier(kp.getPublic()));
    certGen.addExtension(Extension.authorityKeyIdentifier, false,
            new JcaX509ExtensionUtils().createAuthorityKeyIdentifier(kp.getPublic()));
    certGen.addExtension(Extension.keyUsage, true, keyUsage);
} catch (NoSuchAlgorithmException | CertIOException e) {
}

X509CertificateHolder certHolder = null;
try {
    certHolder = certGen.build(new JcaContentSignerBuilder("SHA1withRSA").build(kp.getPrivate()));
} catch (OperatorCreationException e) {
}

Provider bcProvider = new BouncyCastleProvider();
Security.addProvider(bcProvider);

X509Certificate x509Cert = null;
try {
    x509Cert = new JcaX509CertificateConverter().setProvider(bcProvider).getCertificate(certHolder);
} catch (CertificateException e) {
}

CertificateUtil.writeCertificateToFile(x509Cert, directoryPath + fileName);
CertificateUtil.writePrivateKeyToFile(kp.getPrivate(), directoryPath + fileName+"_key");
```

Slika 43: Programski isječak čija je svrha za generiranje krovnog samo-potpisnog certifikata [autorska izrada]

Prvo se generira par ključeva (javni i privatni) koji će se koristiti u certifikatu, za autentifikaciju i za digitalno potpisivanje dalje certifikata. Za algoritam kriptiranja uzet je RSA asimetrični algoritam sa generiranjem ključeva veličine 2048 bitova. Ta veličina ključa je odabrana jer je preporučena kao minimalno sigurna verzija. Svakim dupliranjem veličine ključa, vrijeme deskripcije postaje 6-7 puta sporije. Za samo generiranje potrebno je vrijeme zbog težine matematičkih operacija koje se izvršava procesor računala. [55]

Dalje u kodu se gradi certifikat, kreiraju se polja *subject*, *validity, basic constraints* za kojeg je u kodu ovdje označen posebni OID te je ovo polje označeno kao kritično. Na sljedećoj slici označen je primjer kako su polja označena kao kritična tj. ukoliko sadrže žuti znak upozorenja:



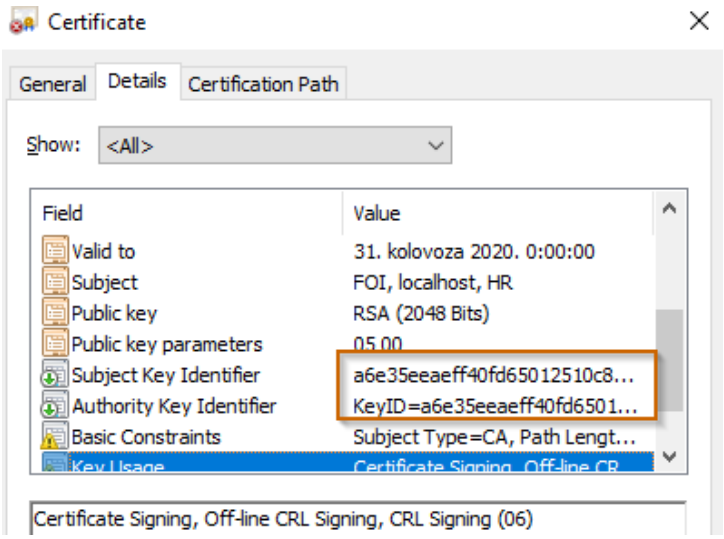
Slika 44: Prikaz izgleda kritičnih polja koja su označena žutim znakom za upozorenje

Ukoliko je polje kritično, primjerice kod *Extended key usage* polja, tada se certifikat mora koristiti isključivo u svrhe kako je navedeno u certifikatu, inače dolazi do kršenja pravila zadana od njegovog autoriteta (CA). Ukoliko polje nije kritično, tada je ono indikator za što se certifikat može koristiti te služi samo kao informacija. [56]

Nakon dodavanja polja, dodaju se određena proširenja, poput: *keyUsage*, *subjectKeyIdentifier* i *authorityKeyIdentifier*.

Tada je moguće pomoću klase CertificateHolder izgraditi certifikat, a navedena klasa sadrži samu strukturu: polja i proširenja certifikata. Dalje je potrebno potpisati certifikat privatnim ključem koji je generiran u paru ključeva u početku procesa generiranja certifikata. Potpisivanje se u ovom slučaju vrši *SHA256withRSA* algoritmom pri čemu se SHA256 koristi za kreiranje sažetka, a RSA se koristi za kriptiranje tog sažetka kako bi se izvršilo samo digitalno potpisivanje.

Nakon potpisivanja certifikata, putem *bouncyCastle* davatelja usluga (eng. *Providera*) se iz *holder* klase dobiva sam certifikat. Sigurnosni *provider* služi za povezivanje davatelja usluga za sigurnosnim uslugama koje nudi, poput: generiranja ključeva, kriptiranja, korištenja raznih algoritama sažimanja, kriptiranja i sl. Na kraju se na lokalni disk zapisuju generirani certifikat i njegov pripadni privatni ključ koji će se kasnije koristiti te je na slici ispod prikazan sadržaj certifikata, primjer zapisa PEM formata kodiranja te primjer zapisa privatnog ključa također PEM formata:



Slika 45: Prikaz pojedinih polja krovnog samo-potpisnog certifikata [autorska izrada]

Na slici 45 vidljiv je dokaz kako su vrijednosti označenih polja (subject i authority key identifier) jednake, što znači da je entitet za kojeg je izdan certifikat ujedno i njegov izdavač. Također, moguće je još vidjeti na slici za što se koristi certifikat, pod poljem *Key Usage*.

Dalje redom su prikazani dijelovi zapisa u PEM formatu certifikata i njegovog privatnog ključa:

```
-----BEGIN CERTIFICATE-----
MIIDPDCCAiSgAwIBAgIBATANBgkqhkiG9w0BAQUFADAvMQswCQYDVQQGEwJIUjES
MBAGA1UEAwwJbG9jYWxob3N0MQwwCgYDVQQKDANGT0kwHhcNMjAwODIzMjIwMDAw
WhcNMjAwODMwMjIwMDAwWjAvMQswCQYDVQQGEwJIUjESMBAGA1UEAwwJbG9jYWxo
b3N0MQwwCgYDVQQKDANGT0kwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIB
AQDXyB5FJveDAFvCH8Gk2K1pK3sTZ+IMlNAS+AnFY9obxQe9fH4ngHDtwq5V7XAF
Jk7TdrLxubOIFF+xHO+blwwlw2uxEqZlFYFG8GmAGL6JiUGIoXuQoBxU9EzZxSGM
7xNOtChnUDYmSYFCDN06pc922mDGDg/+pW+2KhzyRhBPkUC2ux8QUEdpy3XoMSKy
yRjqjAwOICgSwwrYvjjwuuAHbpZzs08wshiq3rYdA1+OW3oT4wyRDkQ370Qh5DM6
ju/wD8sqwCWjQl71ZAiCmkNs90c73uyttSpDMbMOwu/jI/9j1NtslLncH80GRE2i
Q+sBLT3Ba0SXX89TfNGGLrHfAgMBAAGjYzBhMA8GA1UdEwEB/wQFMAMBAf8wHQYD
VR0OBBYEFKbjXurv9A/WUBJRDIBcnkd/slm7MB8GA1UdIwQYMBaAFKbjXurv9A/W
UBJRDIBcnkd/slm7MA4GA1UdDwEB/wQEAwIBBjANBgkqhkiG9w0BAQUFAAOCAQEA
0KT4XsEOtMGu9ju7WKB7WmDaLjPQtuMNqAFe87aYZ2kfABCM+QxKZ4KBBe4tHUD9
i1ldiH45gA6+bXQMtvlic+1L2yq5ZCYk5e8SkF9lh1yadRRw0KnQQ85WTqQipmew
nhxBKtL1Qs/Yc9r7tABdd9v/MPXo1qRB0HFc5xVvfFKyDofZj28QBdYK3Wx5WF2d
Mm8eUPwGogPP0y0TRcKV+JGz39yFmACA7fBkr4tiIv18pLw5KT9tOxuPS7C9x5qh
qiSA9olGt7HBIalQxUTRd0erYm787+wJVRdm+S7AQ1TK9GwkpPECRZHYv6tQvOdY
sWIgOiv8SlS4M9JBjTWl7g==
-----END CERTIFICATE-----

-----BEGIN RSA PRIVATE KEY-----
MIIEwAIBADANBgkqhkiG9w0BAQEFAASCBKowggSmAgEAAoIBAQDXyB5FJveDAFvCH8Gk2K1pK3s
TZ+IMlNAS+AnFY9obxQe9fH4ngHDtwq5V7XAFJk7TdrLxubOIFF+xHO+blwwlw2uxEqZlFYFG8G
mAGL6JiUGIoXuQoBxU9EzZxSGM7xNOtChnUDYmSYFCDN06pc922mDGDg/+pW+2KhzyRhBPkUC2u
x8QUEdpy3XoMSKyyRjqjAwOICgSwwrYvjjwuuAHbpZzs08wshiq3rYdA1+OW3oT4wyRDkQ370Qh
5DM6ju/wD8sqwCWjQl71ZAiCmkNs90c73uyttSpDMbMOwu/jI/9j1NtslLncH80GRE2iQ+sBLT3
Ba0SXX89TfNGGLrHfAgMBAAECggEBAIwztKdN2Eg/9V6+b32CY7oPzFohh1iRYFjS3NZqOI8OiC
+UEPYzCJg6m1xJPTPmoEHlGAzjhvWXglHHp388sNsZjC3YvlooP7cUrFJeVp/lwYI=
-----END RSA PRIVATE KEY-----
```

4.1.2. Intermediate certifikat i CSR

Već je napomenuto ranije što je i kako funkcionira krovni (root) certifikat te je razlika između krovnog i intermediate u tome što su krovni certifikati vjerodajni od strane truststore-ova web preglednika, dok je pouzdanost intermediate certifikata garantirana vjerodajnim centralnim autoritetima koji su izdali krovni certifikat koji je potpisao taj intermediate certifikat. To znači da ti certifikati sami po sebi nisu vjerodajni od strane web preglednika.

Primjer prednost korištenja ovakvog načina izdavanja certifikata je u tome da ukoliko sve certifikate izdajemo i potpisujemo sa jednim krovnim, u slučaju da nam je privatni ključ krovnog certifikata ukraden, tada su nam kompromitirani svi certifikati koji su njime izdani te ih sve moramo reducirati kako bi se umanjila šteta. Da bi se ovakve stvari izbjegle, uvode se intermediate certifikati kao razine ispod kako bi se certifikati mogli grupirati i npr. ukoliko

izgubimo privatni ključ jednog od intermediate certifikata, tada trebamo reducirati samo tu grupu certifikata koja je potpisana tim kompromitiranim certifikatom. [57]

U nastavku je dan programski kod kojim se generira i potpisuje intermediate certifikat te će biti objašnjeni ukratko neki novi pojmovi koji se koriste u procesu generiranja i potpisivanja.

```
156 KeyPair kp = CertificateUtil.generateKeyPair();
157 final PKCS10 request = new PKCS10(kp.getPublic());
158 final String sigAlgName = "SHA256WithRSA";
159
160 Signature signature = null;
161 try {
162     signature = Signature.getInstance(sigAlgName);
163 } catch (NoSuchAlgorithmException e1) {
164 }
165
166 try {
167     signature.initSign(kp.getPrivate());
168
169 } catch (InvalidKeyException e) {
170 }
171 // Entering subject data for creating a CSR
172 String subjectDN = "C="+country +", CN="+commonName +", O="+organisationName;
173 // Sign the request and base-64 encode it
174 try {
175     request.encodeAndSign(new X500Name(subjectDN), signature);
176 } catch (CertificateException | SignatureException e) {
177 }
178
179 final ByteArrayOutputStream baos = new ByteArrayOutputStream();
180 final PrintStream writer = new PrintStream(baos);
181 try {
182     request.print(writer);
183 } catch (SignatureException e) {
184 }
185 // Remove -----BEGIN NEW CERTIFICATE REQUEST----- and -----END NEW
186 // CERTIFICATE REQUEST-----
187 final String requestBase64 = new String(baos.toByteArray());
188 String withoutTags = requestBase64.substring(41);
189 withoutTags = withoutTags.substring(0, withoutTags.length() - 39).trim();
190
191 final PKCS10CertificationRequest holder = new PKCS10CertificationRequest(
192         org.apache.commons.codec.binary.Base64.decodeBase64(withoutTags));
193
194 CertificateUtil.caCertificate = caCertificate;
195 try {
196     X509Certificate signedCertificate = null;
197     try {
198         signedCertificate = CertificateUtil.sign(holder, datePeriodValidity, urlOCSP, urlCaCert
199                 , urlCRLFile,caPrivateKey, kp);
```

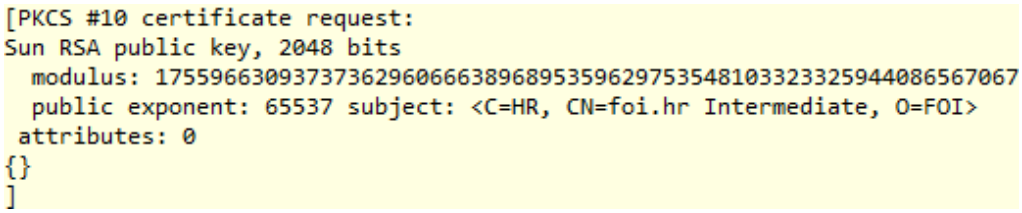
Slika 46: Programski kod kojim se generira i potpisuje navedeni intermediate certifikat

Prvo se opet generira par ključeva što je već objašnjeno u pod poglavlju prije, dalje se inicijalizira digitalni potpis sa algoritmima koji će se koristiti za sažimanje i kriptiranje te sa generiranim privatnim ključem iz para ključeva.

Nakon toga, potrebno je kreirati zahtjev za potpisivanje certifikata, skraćeno CSR (eng. *Certificate Signing Request*). CSR sadrži javni ključ koji će biti u sklopu izdanog certifikata te još u sebi sadrži informacije o entitetu za kojeg CA izdaje certifikat kao što su ista koja se

nalaze i u *subject* polju certifikata (ime organizacije, država itd.). Također, CSR sadrži i informaciju o tipu i veličina ključa te se također kreira u Base-64 baziranom PEM formatu. [58]

Slika 47: Primjer kako izgleda CSR u Base64 baziranom PEM formatu [autorska izrada]



Slika 48: Primjer sadržaja PKCS #10 zahtjeva za potpisivanje certifikata [autorska izrada]

PKCS označava kriptografske standarde i formate koji se koriste u kriptografiji javnih ključeva (eng. *Public Key Cryptography Standards*) te svaka verzija označava nešto drugo pa tako npr. verzija 10 opisuje standardan format kod zahtijevanja izdavanja X509 certifikata od centralnih autoriteta. [59]

Na kraju se pomoću danih podataka postepeno kreira struktura certifikata koji će se izdati (iz CSR-a se npr. uzimaju podaci za *subject* polje) te se taj certifikat potpisuje privatnim ključem krovnog certifikata koji ga izdaje.

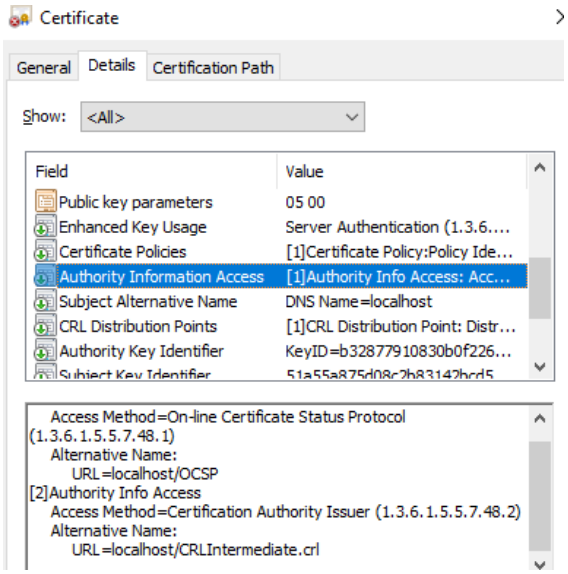
4.1.3. Klijentski certifikat i pripadajući Keystore

Za kreiranje certifikata kojeg će koristiti krajnji korisnik, postupak je jako sličan kao i kod postupka generiranja intermediate certifikata opisanog u pod poglavlju iznad. Ovdje će se samo staviti fokus na koji način se dodaju polja i proširenja:

```
// Using the current timestamp as the certificate serial number
BigInteger certSerialNumber = new BigInteger(Long.toString(now));

X509v3CertificateBuilder certBuilder = new X509v3CertificateBuilder(issuerDNName, certSerialNumber,
        validityDates.get(0), validityDates.get(1), pkcs10Holder.getSubject(), keyInfo);

// add Enhanced Key Usage
certBuilder.addExtension(Extension.extendedKeyUsage, false, new ExtendedKeyUsage(
        new KeyPurposeId[] { KeyPurposeId.id_kp_serverAuth, KeyPurposeId.id_kp_clientAuth }));

// add Certificate Policies extension
PolicyQualifierInfo pqInfo = new PolicyQualifierInfo("http://demo-pki.fina.hr/cps/cpsqcdemo2014v2-0-hr.pdf");
PolicyInformation policyInfo = new PolicyInformation(PolicyQualifierId.id_qt_cps, new DERSequence(pqInfo));
CertificatePolicies policies = new CertificatePolicies(policyInfo);
certBuilder.addExtension(Extension.certificatePolicies, false, policies);

// add AuthorityInformationAccess extension
ASN1EncodableVector aia_ASN = addAuthorityInformationAccess(urlOCSP, urlCaCert);
certBuilder.addExtension(Extension.authorityInfoAccess, false, new DERSequence(aia_ASN));

// add subject alternative name
List<GeneralName> altNames = new ArrayList<GeneralName>();
altNames.add(new GeneralName(GeneralName.dNSName, "localhost"));
GeneralNames subjectAltNames = GeneralNames
        .getInstance(new DERSequence((GeneralName[]) altNames.toArray(new GeneralName[] {})));
certBuilder.addExtension(Extension.subjectAlternativeName, false, subjectAltNames);

// add CRL Distribution Points extension
// DistributionPoint[] distPoints = addCRLDistributionPoints();
List<String> crlUris = new ArrayList<>();
crlUris.add(urlCRLFile);

CRLDistPoint distPoints = addCRLDistributionPoints(issuerDNName, pkcs10Holder.getSubject(), crlUris);
certBuilder.addExtension(Extension.cRLDistributionPoints, false, distPoints);

// add Subject and Authority Key Identifier
try {
    certBuilder.addExtension(Extension.authorityKeyIdentifier, false,
            new JcaX509ExtensionUtils().createAuthorityKeyIdentifier(caCertificate.getPublicKey()));
    certBuilder.addExtension(Extension.subjectKeyIdentifier, false,
            new JcaX509ExtensionUtils().createSubjectKeyIdentifier(pair.getPublic()));
} catch (NoSuchAlgorithmException e) {
}

// Basic Constraints, true is for CA, false for EndEntity
BasicConstraints basicConstraints = new BasicConstraints(false);
// Basic Constraints is usually marked as critical.
certBuilder.addExtension(new ASN1ObjectIdentifier("2.5.29.19"), true, basicConstraints);

// adding keyUsage extension
KeyUsage keyUsage = new KeyUsage(KeyUsage.digitalSignature | KeyUsage.keyEncipherment);
certBuilder.addExtension(Extension.keyUsage, true, keyUsage);
```

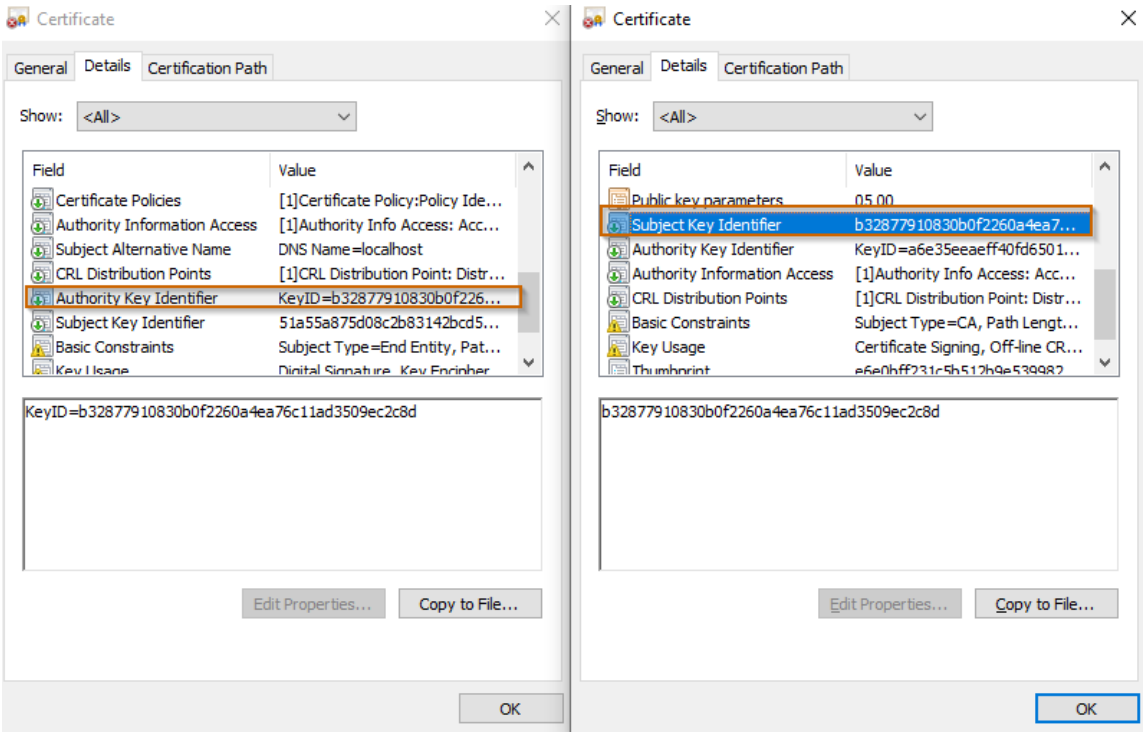
Slika 49: Postepeno izgrađivanje strukture klijentskog certifikata [autorska izrada]

Na slici 49 vidljiv je način na koji se dodaju polja i proširenja koja su u teorijskom dijelu već bila opisana te je rezultat generiranja slijedeći:



Slika 50: Dobiveni rezultat [autorska izrada]

Jedan od načina kako se može provjeriti da li je certifikat stvarno potpisan od nekog jest usporedbom vrijednosti proširenja certifikata: *Authority Key Identifier* (gleda se u klijentskom certifikatu) i *Subject Key Identifier* (gleda se u izdavateljskom certifikatu), što znači da se vrijednosti moraju poklapati:



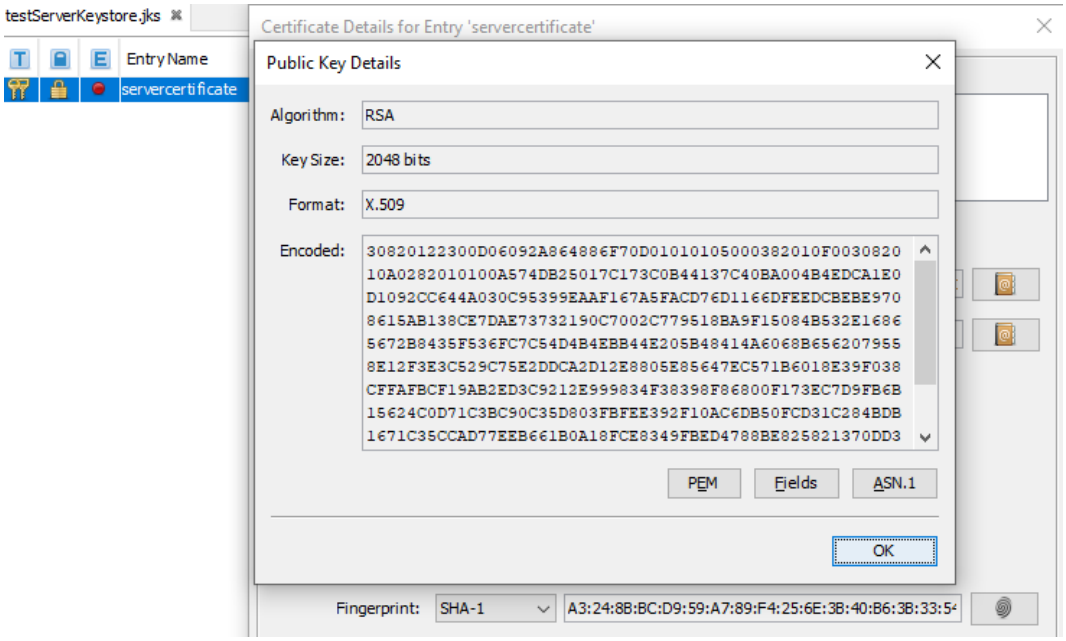
Slika 51: Jedan od načina provjere da li je certifikat stvarno potpisan od danog izdavateljskog certifikata [autorska izrada]

Inače, postoje drugačiji i bolji načini provjere kojim se verificira da li je certifikat izdan stvarno od zadanog centralnog autoriteta.

Za kraj je još za potrebe testiranja staviti certifikat i njegov pripadni ključ u *keystore*. U Java Keystore se mogu spremati privatni ključevi, certifikati sa njihovim javnim ključevima koji se dalje koriste za razne svrhe kao npr. omogućivanje HTTPS porta neke aplikacije na nekom poslužitelju. Svaki unos ključa ili certifikata se može još zaštititi lozinkom. Ispod je dan prikaz programskog koda kojim se kreirao *keystore* kao i prikaz unosa koji je pohranjen u njega:

```
KeyStore ks = null;
try {
    ks = KeyStore.getInstance(KeyStore.getDefaultType());
} catch (KeyStoreException e) {
    e.printStackTrace();
}

char[] password = "123456".toCharArray();
try {
    ks.load(null, password);
} catch (NoSuchAlgorithmException | CertificateException | IOException e) {
    e.printStackTrace();
}

try {
    ks.setCertificateEntry("serverCertificate", serverCertificate);
    Certificate[] certChain = new Certificate[1];
    certChain[0] = (Certificate) serverCertificate;
    ks.setKeyEntry("servercertificate", (Key)caPrivateKey, password ,certChain);
} catch (KeyStoreException e1) {
    e1.printStackTrace();
}

// Store away the keystore.
FileOutputStream fos = null;
try {
    fos = new FileOutputStream(directoryPath+keystoreFilename+".jks");
```

Slika 52: Programski isječak kojim se kreira keystore [autorska izrada]



Slika 53: Otvaranje *keystore*-a KeyStore Explorer alatom sa pripadnim podacima o certifikatu i priloženom ključu [autorska izrada]

4.1.4. Testiranje generiranih certifikata i podizanje aplikacije

Da bi se testirao izgenerirani klijentski certifikat, bilo ga je prvo potrebno podesiti HTTPS port na aplikaciji za generiranje lanaca certifikata i *keystore*-a. U Tomcatu je bilo potrebno u datoteci *server.xml* dodati HTTPS konektor na slijedeći način:

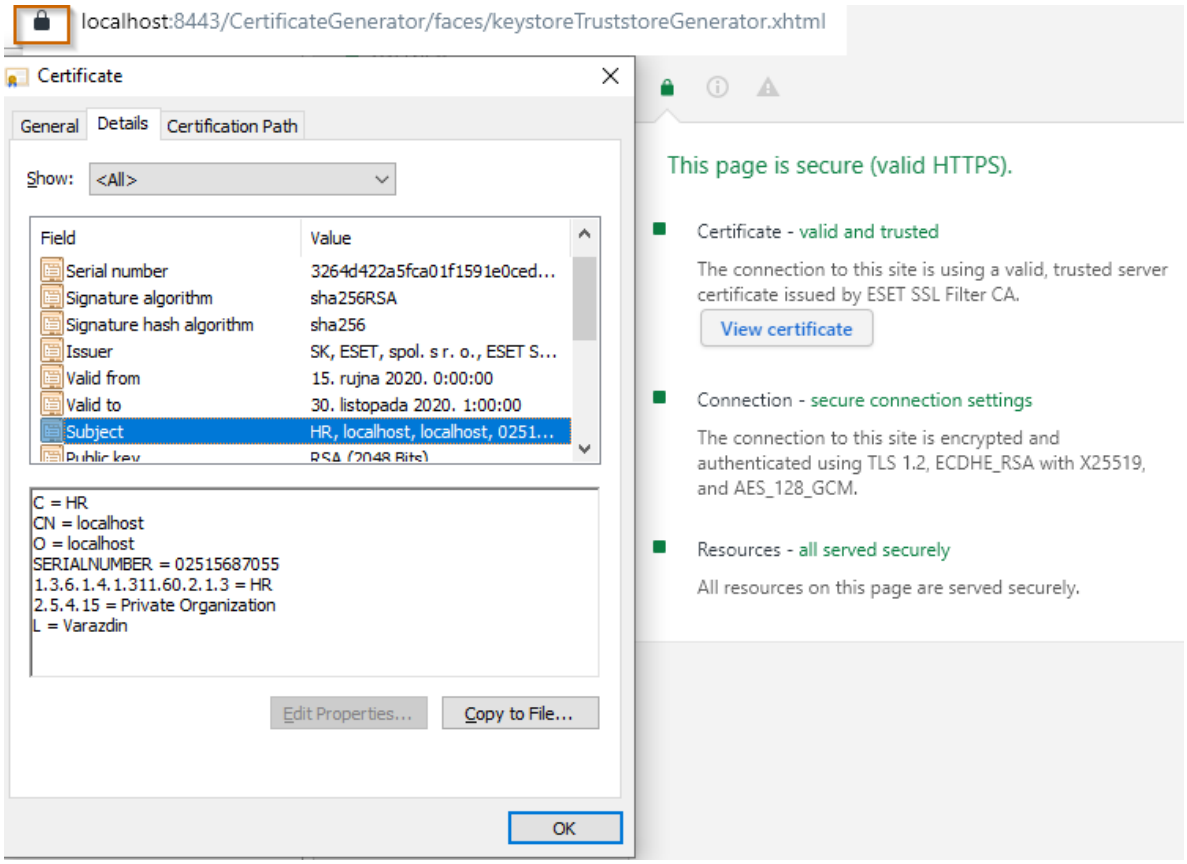
```
<Connector
               protocol="org.apache.coyote.http11.Http11NioProtocol"
port="8443"
               maxThreads="200" scheme="https" secure="true"
SSLEnabled="true"
               keystoreFile="C:\FOI\Diplomski\Kreirani certifikati i
keystore\testServerKeystore.jks" keystorePass="123456"
               keyAlias="servercertificate"
               clientAuth="false" sslProtocol="TLS" />
```

Navedeni atributi samim nazivom označavaju svoju svrhu te ću dodati napomenu da se pod vrijednosti atributa *sslProtocol* može staviti neka od važećih verzija protokola (npr. TLS 1.3). *ClientAuth* atribut označava način na koji će se izvršavati autentifikacija. *Truststore* također označava spremište certifikata te ključeva kao i *keystore*, ali je razlika u tome što se u *truststore-u* nalaze jedino certifikati koji mogu identificirati ostale. Pošto *truststore* nije dodan, nije moguće uspostaviti dvosmjernu autentifikaciju što znači da samo poslužitelj treba klijentu, odnosno njegovom web pregledniku potvrditi svoj identitet.

ClientAuth može imati slijedeće vrijednosti:

a) true – označava da je obavezno slati zahtjev sa certifikatom čiji se potpisni certifikat nalazi u *truststore-u* poslužitelja, u protivnom je zahtjev odbijen
b) want – omogućuje dvosmjernu autentifikaciju kao što je navedeno sa vrijednost *true,* ali nije nužno slati certifikat uz zahtjev. Zahtjev će biti jedino odbijen u slučaju ukoliko se šalje zahtjev sa certifikatom čiji se potpisni certifikat ne nalazi u *truststore*-u poslužitelja
c) false – isključuje dvosmjernu autentifikaciju te se šalju zahtjevi bez certifikata

Na kraju, kada je aplikacija dignuta, moguće je vidjeti da je naš generirani certifikat ispravan te je stranica zaštićena TLS protokolom uz jednosmjernu autentifikaciju:



Slika 54: Potvrda da je certifikat ispravan i da je stranica sada zaštićena TLS protokolom [autorska izrada]

5. Zaključak

Ovaj diplomski rad zamišljen je kako bi se razradila poveznica između TLS protokola, PKI infrastrukture te uloge X.509 certifikata u njima.

Bilo je potrebno opisati način na koji TLS protokol funkcionira u sigurnosnom dijelu te su pomoću Wireshark alata bili analizirani mrežni prometi prilikom pristupanja određenoj stranici koja je na HTTPS portu i koristi TLS protokol. Uz to, bile su navedene značajke, ciljevi i kratka povijest samog protokola kako bi se bolje razumjelo što se točno događa u mrežnim paketima prilikom slanja zahtjeva. Najvažnije stavke, odnosno procesi koji su se odvijali bili su rukovanje i izvođenje ključeva te je dan pregled poruka koje se šalju tijekom TLS rukovanja te i opisan način na koji se vrši izvođenje ključeva između klijenta i poslužitelja. Također, stavljen je naglasak na poruku gdje se šalje certifikat, odnosno gdje se poslužitelj svojim certifikatom predstavlja klijentu te tada ga web preglednik klijenta mora provjeriti, odnosno utvrditi da li je certifikat valjan i pouzdan raznim tehnikama npr. poput OCSP ili CRL provjere reduciranosti certifikata. Navedeni proces gdje se samo poslužitelj predstavlja klijentu svojim certifikatom, nazivamo jednosmjernom autentifikacijom.

Dalje u radu je dana arhitektura PKI infrastruktura te je također dan naglasak na korištenje certifikata za digitalno potpisivanje i autentifikaciju unutar PKI infrastrukture. Navedene su prednosti te način i gdje se certifikati unutar PKI infrastrukture mogu sve koristiti.

Na kraju teoretskog dijela, opisan je najvažniji dio vezan za certifikate, dana je struktura unutar X.509 norme te su opisana sva najvažnija polja i proširenja svih verzija X.509 certifikata. Kratko je opisana ANS.1 notacija kao i DER te PEM pravila kodiranja za lakše shvaćanje jer su navedenima X.509 certifikati opisani unutar norme.

Praktičnim dijelom rada napravljen je pokušaj implementacije i primjene sječenih znanja iz teorijskog dijela. Napravljena je aplikacija u Java programskom okruženju za generiranje lanaca certifikata te su je cijeli lanac bio uspješno napravljen. Uz to, još neki pojmovi su bili slijedno objašnjeni kako se javljala potreba za kod same implementacije, kao npr. pojmovi poput što je to uopće lanac certifikata, pa do toga što je zahtjev za potpisivanje certifikata itd. Cijeli lanac je na kraju uspješno generiran uz pomoć Javinih i besplatnih paketa kao što je BouncyCastle. Trebalo je još testirati uz provjeru da li su certifikati valjani. Pa je tako na kraju praktičnog dijela sama aplikacija bila podignuta na HTTPS port te je bila implementirana jednosmjerna autentifikacija sa certifikatom koji je bio izdan za danu lokalnu domenu. Web preglednik je pokazao da je stranica ispravno zaštićena te da je certifikat koji je stavljen na poslužitelj također ispravan.

O samoj temi sam, pogotovo u teoretskom dijelu, jako puno naučio. Neke stvari sam praktično znao i prije, ali nisam znao kako točno to funkcionira „ispod haube“ te sam htio to detaljno vidjeti da dobro shvatim tematiku problema kod ove osjetljive teme. Zadovoljan sam sječenim znanjem i literaturom koju sam proučavao te se ova tema može još dosta proširiti jer postoji još puno stvari koje nisu dotaknute u ovom diplomskom radu, a također su jako bitne za funkcioniranje TLS protokola i uloge certifikata među njima kao što su npr. provjera putem OCSP ili CRL metode reduciranosti.

6. Popis literature

[1] N. Cleglec i D. Žigman , »KRIPTOGRAFSKI PROTOKOL,« *Polytechnic and design,* svez. 5, br. 1, pp. 79-80, 2017.

[2] K. W. Alger, »IoT Security with SSL/TLS in MicroPython,« Blog of Ken W. Alger, 23 June 2017. [Mrežno]. Available: https://www.kenwalger.com/blog/tag/openssl/. [Pokušaj pristupa 15 September 2020].

[3] »SSL/TLS Protocol Layers,« Google, [Mrežno]. Available: https://sites.google.com/site/tlssloverview/ssl-tls-protocol-layers. [Pokušaj pristupa 15 September 2020].

[4] »Record Layer,« [Mrežno]. Available: https://sites.google.com/site/tlssloverview/ssl-tls-protocol-layers/record-layer. [Pokušaj pristupa 15 September 2020].

[5] »SSL/TLS Overview,« Google, [Mrežno]. Available: https://sites.google.com/site/tlssloverview/ssl-tls-protocol-layers/handshake-layer/change-cipher-spec-protocol. [Pokušaj pristupa 15 September 2020].

[6] »SSL/TLS Overview,« Google, [Mrežno]. Available: https://sites.google.com/site/tlssloverview/ssl-tls-protocol-layers/handshake-layer/alert-protocol. [Pokušaj pristupa 15 September 2020].

[7] M. Satran i D. Batchelor, »TLS Record Protocol,« Microsoft, 31 May 2018. [Mrežno]. Available: https://docs.microsoft.com/en-us/windows/win32/secauthn/tls-record-protocol. [Pokušaj pristupa 15 September 2020].

[8] T. Dierks i E. Rescorla, »RFC 5246 - The Transport Layer Security (TLS) Protocol,« Internet Engineering Task Force (IETF), August 2008. [Mrežno]. Available: https://tools.ietf.org/html/rfc5246#page-65. [Pokušaj pristupa 15 September 2020].

[9] N. Ivković, »Poglavlje 8: Sigurnost, nastavni materijali na predmetu Mreže računala 2 [Moodle],« Sveučilište u Zagrebu, Fakultet organizacije i, Varaždin, 2015.

[10] I. Ristić, »Transport Layer Security,« u *Bulletproof SSL and TLS*, London, Feisty Duck Limited, 2014, p. 2.

[11] E. Baier, »The Evolution of SSL and TLS,« 2015.

[12] T. Dierks i E. Rescorla, *Request for Comments: 4346,* 2006, p. 5.

[13] wolfSSL, »A Comparison of Differences in TLS 1.1 and TLS 1.2,« 2015.

[14] B. Jackson, »An Overview of TLS 1.3 – Faster and More Secure,« KINSTA BLOG, 2020.

[15] MDN suradnici, »MDN web docs - Round Trip Time (RTT),« Mozzila, 3 June 2019. [Mrežno]. Available: https://developer.mozilla.org/en-US/docs/Glossary/Round_Trip_Time_(RTT). [Pokušaj pristupa 15 September 2020].

[16] N. Sullivan, »Introducing Zero Round Trip Time Resumption (0-RTT),« Cloudflare, 15 March 2020. [Mrežno]. Available: https://blog.cloudflare.com/introducing-0-rtt/. [Pokušaj pristupa 15 September 2020].

[17] I. Ristić, »Handshake protocol,« u *Bulletproof SSL and TLS*, London, Feisty Duck Limited, 2014, pp. 25-34.

[18] W. Pan, »Analysis of TLS/SSL Handshake Failure Scenarios on Alibaba Cloud,« 6 February 2020. [Mrežno]. Available: https://www.alibabacloud.com/blog/analysis-of-tlsssl-handshake-failure-scenarios-on-alibaba-cloud_595800. [Pokušaj pristupa 15 September 2020].

[19] P. Nohe , »Cipher Suites: Ciphers, Algorithms and Negotiating Security Settings,« 7 May 2019. [Mrežno]. Available: https://www.thesslstore.com/blog/cipher-suites-algorithms-security-settings/. [Pokušaj pristupa 15 September 2020].

[20] »Message Authentication,« Tutorials point, [Mrežno]. Available: https://www.tutorialspoint.com/cryptography/message_authentication.htm. [Pokušaj pristupa 15 September 2020].

[21] I. Ristić, »Bulletproof SSL and TLS,« u *Cipher suites*, London, Feisty Duck Limited, 2014, pp. 49-50.

[22] Y. Nir, S. Josefsson i M. Pegourie-Gonnard, »Elliptic Curve Cryptography (ECC) Cipher Suites - RFC 8422,« August 2018. [Mrežno]. Available: https://tools.ietf.org/html/rfc8422. [Pokušaj pristupa 15 September 2020].

[23] E. Rescorla, »The Transport Layer Security (TLS) Protocol Version 1.3 - RFC 8446,« August 2018. [Mrežno]. Available: https://tools.ietf.org/html/rfc8446#section-4.2.7. [Pokušaj pristupa 15 September 2020].

[24] I. Ristić, »Extensions,« u *Bulletproof SSl and TLS*, London, Feisty Duck Limited, 2014, pp. 52-59.

[25] Comodo SSL Store, »Can you have multiple SSL certificates for one domain?,« [Mrežno]. Available: https://comodosslstore.com/resources/can-you-have-multiple-ssl-certificates-for-one-domain/. [Pokušaj pristupa 15 September 2020].

[26] A. Gonzalez, »Cybersecurity, Payment Security & Cryptography - HTTPS and the TLS handshake protocol,« 2 January 2013. [Mrežno]. Available: https://albertx.mx/https-handshake/. [Pokušaj pristupa 15 September 2020].

[27] I. Ristić, »Key Exchange,« u *Bulletproof SSL and TLS*, London, Feisty Duck Unlimited, 2014, pp. 35-41.

[28] I. Ristić, »Pseudorandom Function,« u *Bulletproof SSL and TLS*, London, Feisty Duck Unlimited, 2014, p. 48.

[29] L. Gudin, M. Golub, D. Jakobović i L. Jelenković, »11.7.1. Diffie-Hellmanov postupak za razmjenu tajnog ključa,« u *OPERACIJSKI SUSTAVI*, Zagreb, Element, 2010, p. 325.

[30] »Diffie Hellman Summarized Operation,« 2015. [Mrežno]. Available: https://www.researchgate.net/figure/Diffie-Hellman-Summarized-Operation_fig3_278702697. [Pokušaj pristupa 15 September 2020].

[31] M. Golub, »Sigurnost računalnih sustava, nastavni materijali na predmetu Operacijski sustavi 2,« Sveučilište u Zagrebu, Fakultet elektrotehnike i računalstva, Zagreb, 2006.

[32] L. CARNet CERT, »Nedostaci PKI infrastrukture - CCERT-PUBDOC-2009-02-255,« February 2009. [Mrežno]. Available: https://www.cis.hr/www.edicija/LinkedDocuments/CCERT-PUBDOC-2009-02-255.pdf. [Pokušaj pristupa 15 September 2020].

[33] F. Financijska agencija, »FINA PKI System,« 1 August 2018. [Mrežno]. Available: http://rdc.fina.hr/RDC2015/FinaRDC2015-CPWSA1-2-en.pdf. [Pokušaj pristupa 15 September 2020].

[34] F. Financijska agencija, »FINA PKI System,« 5 September 2016. [Mrežno]. Available: http://rdc.fina.hr/RDC2015/FinaRDC2015-CPSQC5-1-en.pdf. [Pokušaj pristupa 15 September 2020].

[35] N. Ferguson, B. Schneier i T. Kohno, »The Dream of PKI,« u *Cryptography Engineering*, Indianapolis, Indiana, Wiley Publishing, Inc., 2010¸, pp. 277-280.

[36] »PKI and E-mail Encryption - A Brief Explanation,« [Mrežno]. Available: http://phpki.sourceforge.net/phpki/help/PKI_basics.html. [Pokušaj pristupa 15 September 2020].

[37] Cloudflare, »What is IPsec? | How IPsec VPNs work,« [Mrežno]. Available: https://www.cloudflare.com/learning/network-layer/what-is-ipsec/. [Pokušaj pristupa 15 September 2020].

[38] W. Stallings, »Electronic Mail Security,« u *Cryptography and Network Security Principles and Practices Fourth Edition*, Upper Saddle River, NJ, Pearson Education, Inc., 2006, pp. 437-458.

[39] W. Stallings, »X.509 Authentication Service,« u *Cryptography and Network Security - Fourth Edition*, Upper Saddle River, NJ, Pearson Education, Inc., 2006, pp. 419-428.

[40] I. Ristić, »Certificates,« u *Bulletpoorf SSL and TLS*, London, Feisty Duck Unlimited, 2014, pp. 66-70.

[41] W. Stark, »Key management and Distribution,« 29 Jun 2019. [Mrežno]. Available: https://medium.com/@winstark_212/key-management-and-distribution-bf3d8da4f2e5. [Pokušaj pristupa 15 September 2020].

[42] L. Gudin, M. Golub, M. Jakobović i L. Jelenković, »11.10 Infrastruktura javnih klujčeva,« u *Operacijski sustavi*, Zagreb, Element, 2010, p. 350.

[43] D. Coulter, M. Jacobs i M. Satran, »X.509 Public Key Certificates,« 31 May 2018. [Mrežno]. Available: https://docs.microsoft.com/en-us/windows/win32/seccertenroll/about-x-509-public-key-certificates. [Pokušaj pristupa 15 September 2020].

[44] Network Working Group, »RFC 3280 - Internet X.509 Public Key Infrastructure,« April 2002. [Mrežno]. Available: https://www.ietf.org/rfc/rfc3280.txt. [Pokušaj pristupa 15 September 2020].

[45] »Introduction to ASN.1,« [Mrežno]. Available: https://www.itu.int/en/ITU-T/asn1/Pages/introduction.aspx. [Pokušaj pristupa 15 September 2020].

[46] M. Satran i D. Batchelor, »O (Security Glossary),« 31 May 2018. [Mrežno]. Available: https://docs.microsoft.com/en-us/windows/win32/secgloss/o-gly. [Pokušaj pristupa 15 September 2020].

[47] M. Satran , T. Sherer, M. Jacobs i D. Coulter, »Basic Fields,« 31 May 2018. [Mrežno]. Available: https://docs.microsoft.com/en-us/windows/win32/seccertenroll/about-basic-fields. [Pokušaj pristupa 15 September 2020].

[48] N. W. Group, D. Copper, S. Santesson, S. Farrell, S. Boeyen, w. Polk i R. Housley, »Internet X.509 Public Key Infrastructure Certificate,« May 2008. [Mrežno]. Available: https://tools.ietf.org/html/rfc5280#page-29. [Pokušaj pristupa 15 September 2020].

[49] V. Podāns, »Basic Constraints certificate extension,« 12 August 2019. [Mrežno]. Available: https://www.pkisolutions.com/basic-constraints-certificate-extension/. [Pokušaj pristupa 15 September 2020].

[50] L. Vogel, »JSF (JavaServer Faces) - Tutorial,« 26 October 2016. [Mrežno]. Available: https://www.vogella.com/tutorials/JavaServerFaces/article.html#jsf_application. [Pokušaj pristupa 15 September 2020].

[51] P. Informatics, »Licences,« [Mrežno]. Available: https://www.primefaces.org/licenses/. [Pokušaj pristupa 15 September 2020].

[52] M. Tyson, »What is Tomcat? The original Java servlet container,« 19 December 2019. [Mrežno]. Available: https://www.infoworld.com/article/3510460/what-is-apache-tomcat-the-original-java-servlet-container.html. [Pokušaj pristupa 15 September 2020].

[53] DNSSimple, »What is the SSL Certificate Chain?,« [Mrežno]. Available: https://support.dnsimple.com/articles/what-is-ssl-certificate-chain/. [Pokušaj pristupa 15 September 2020].

[54] M. Pickavance, »Best SSL certificate services to buy from in 2020: Get the cheapest price today,« 28 August 2020. [Mrežno]. Available: https://www.techradar.com/news/best-ssl-certificate-provider. [Pokušaj pristupa 7 September 2020].

[55] »RSA key lengths,« [Mrežno]. Available: https://www.javamex.com/tutorials/cryptography/rsa_key_length.shtml. [Pokušaj pristupa 15 September 2020].

[56] H. D. Corporation, »Key usage extensions and extended key usage,« [Mrežno]. Available: https://help.hcltechsw.com/domino/11.0.0/conf_keyusageextensionsandextendedkeyusage_r.html. [Pokušaj pristupa 15 September 2020].

[57] P. Nohe, »The Difference Between Root Certificates and Intermediate Certificates,« 26 June 2019. [Mrežno]. Available: https://www.thesslstore.com/blog/root-certificates-intermediate/. [Pokušaj pristupa 15 September 2020].

[58] R. Publico, »SSL Basics: What is a Certificate Signing Request (CSR)?,« 07 September 2017. [Mrežno]. Available: https://www.globalsign.com/en/blog/what-is-a-certificate-signing-request-csr. [Pokušaj pristupa 15 September 2020].

[59] M. Nystrom i B. Kaliski, »PKCS #10: Certification Request Syntax Specification,« November 2000. [Mrežno]. Available: https://tools.ietf.org/html/rfc2986. [Pokušaj pristupa 15 September 2020].

[60] Z. Wang, J. Zhao i G. Zhong, »Public-Key applications in E-commerce,« June 2019. [Mrežno]. Available: https://www.researchgate.net/figure/Public-Key-Infrastructure-structure-PKI-a-key-cryptographic-technology-is-widely-used_fig1_333876926. [Pokušaj pristupa 15 September 2020].

[61] [Mrežno].

7. Popis slika