Vizualizacija SSL protokola

Jansky, Tomislav

VELEUČILIŠTE U BJELOVARU
PREDDIPLOMSKI STRUČNI STUDIJ RAČUNARSTVO

VIZUALIZACIJA SSL PROTOKOLA

Završni rad br. 04/RAČ/2020

Tomislav Jansky

Bjelovar, Listopad 2020.

obrazac ZR - 001

Veleučilište u Bjelovaru

Trg E. Kvaternika 4, Bjelovar

1. DEFINIRANJE TEME ZAVRŠNOG RADA I POVJERENSTVA

Kandidat: **Jansky Tomislav** Datum: 31.08.2020. Matični broj: 001852

JMBAG: 0314017980

Kolegij: **SIGURNOST RAČUNALA I PODATAKA**

Naslov rada (tema): **Vizualizacija SSL protokola**

Područje: **Tehničke znanosti** Polje: **Računarstvo**

Grana: **Informacijski sustavi**

Mentor: **Dario Vidić, mag.ing.el.techn.inf.** zvanje: **predavač**

Članovi Povjerenstva za ocjenjivanje i obranu završnog rada:

1. **Tomislav Adamović, mag.ing.el., predsjednik**
2. **Dario Vidić, mag.ing.el.techn.inf., mentor**
3. **Ivan Sekovanić, mag.ing.inf.et comm.techn., član**

2. ZADATAK ZAVRŠNOG RADA BROJ: 04/RAČ/2020

U radu je potrebno detaljno prikazati način rada sigurnog protokola za komunikaciju - SSL. Omogućiti programskim rješenjem vizualizaciju napada na aplikaciju (npr. man in the middle) te prikazati razliku korištenja SSL protokola u komunikaciji u odnosu na nezaštićenu komunikaciju. Prikazati ranjivosti SSL protokola.

REPUBLIKA HRVATSKA 2 VELEUČILIŠTE U BJELOVARU BJELOVAR

Zadatak uručen: 31.08.2020. Mentor: **Dario Vidić, mag.ing.el.techn.inf.**

Zahvala

Zahvaljujem se mentoru Dariu Vidiću, magistru inžinjeru elektrotehnike i informacijske tehnologije. Zahvaljujem se na strpljenju i pruženoj pomoći tijekom pisanja završnog rada.

Zahvaljujem se svojim roditeljima na ukazanom povjerenju i pruženoj financijskoj pomoći.

Također se zahvaljujem svojoj djevojci na strpljenju i potpori tijekom pisanja završnog rada.

Sadržaj

1. UVOD

SSL protokol (*engl. secure socket layer*) je protokol kojem je glavna uloga kriptiranje podataka čija se izmjena vrši na relaciji između dva sustava, klijenta i poslužitelja. Radi se o protokolu široke namjene diljem cjelokupne internetske mreže bez koje velik postotak komunikacije, bila ona poslovna ili osobna, ne bi bilo moguće. SSL protokol omogućuje osiguranu komunikaciju. Sami SSL protokol razvila je računalna tvrtka *Netscape Communications* 1994. godine. Završni rad na temu Vizualizacija SSL protokola podijeljen je na teorijski i praktični dio.

Teorijski dio rada bavi se prikazom tematike vezane uz različite načine uspostave kriptirane komunikacije postignute izmjenjivanjem raznih SSL poruka između klijenta i poslužitelja. Prikazani su principi uspostave bez autentifikacije, uspostave uz autentifikaciju poslužitelja, uspostave uz autentifikaciju poslužitelja pri odvojenoj autentifikaciji i enkripciji, uspostave uz obostranu autentifikaciju, uspostave prilikom korištenja parametara iz prethodne sjednice i završavanja same kriptirane komunikacije. Nadalje, rad SSL protokola detaljnije je prikazan kroz njegovo funkcioniranje kroz SSL slojeve.

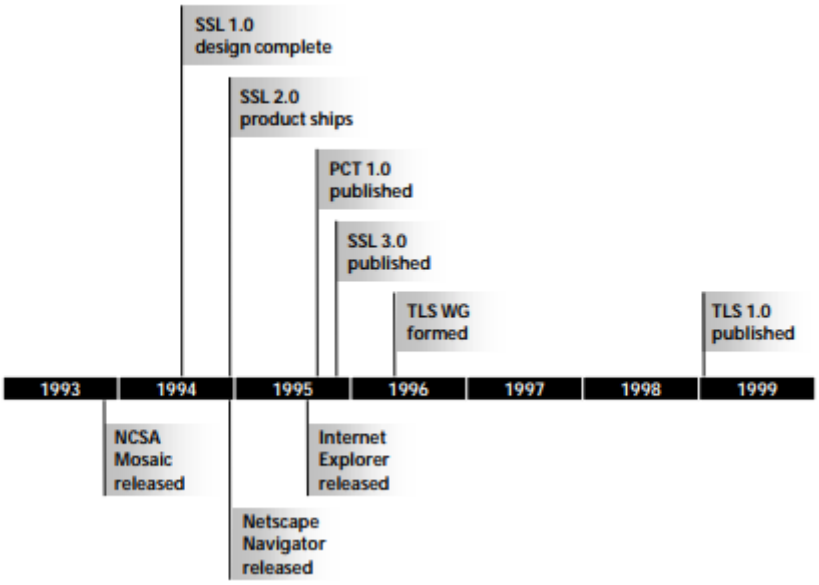
Praktični dio rada bavi se prikazom vizualizacije, a ujedno i implementacijom SSL protokola. Prilikom implementacije kreirani su samopotpisani certifikati; korijenski certifikat, poslužiteljski certifikat te klijentski certifikati. Prikazano je njihovo korištenje. Također prilikom implementacije podignut je lokalni IIS poslužitelj koji je konfiguriran tako da koristi poslužiteljski certifikat te da traži klijenta njegovo predstavljanje s vlastitim klijentskim certifikatom. Također, kreirana je web aplikacija postavljena na tom IIS poslužitelju. Prikazana je provjera klijentskog certifikata i na aplikativnom sloju. Sama web aplikacija, nakon uspješne klijentske autentifikacije, prikazuje vizualizaciju SSL protokola. Cilj ovoga rada je detaljno opisati načine rada SSL protokola te implementirati, prikazati i vizualizirati isto.

2. OPĆENITO O SSL PROTOKOLU

2.1 Povijest SSL-a

Dok je tvrtka za računalne usluge *Netscape Communications* razvijala svoj najpoznatiji produkt i ujedno prvi internetski preglednik (*engl. web browser*) *Navigator*, dogodio se početak bavljenja problematikom sigurnosti podataka dostupnih na internetskoj mreži.

Upravo je ova tvrtka razvila primarnu inačicu protokola za zaštitu podataka *Secure Sockets Layer*. Razvitak samoga protokola kroz vrijeme slikovno je prikazan slikom *2.1*.



Slika 2. 1: Povijest SSL protokola [1]

Netscape Communications dovršila je razvitak verzije 1.0 protokola 1994. godine. Ova verzija nikada nije upoznata s javnošću, iz razloga što je bila podložna velikom broju kritika zbog slabih kriptirajućih algoritama koje je koristila; „*This version circulated only internally (i.e., inside Netscape Communications), since it had several shortcomings and flaws*“. [2] Pet mjeseci kasnije došlo je do razvitka verzije 2.0 SSL protokola i posljedično prvog produkta koji je bio u mogućnosti podržavati SSL 2.0, *Netscape Navigator*-a.

SSL verzija 2.0 također je posjedovala pojedine slabosti. Ovi tipovi problema riješeni su pojavom SSL protokola verzije 3.0, koja se pojavila unutar razdoblja od tri mjeseca nakon zasnivanja internetskog preglednika *Internet Explorer* 1995. godine. Ubrzo su u ovoj verziji također pronađene sigurnosne mane. Stoga je razvijen TLS protokol koji se temelji na SSL 3.0

protokolu. Verzija koja se danas smatra ispravnom i najsigurnijom za korištenje kreirana je 2018. godine. Naziva se TLS 1.3. [3]

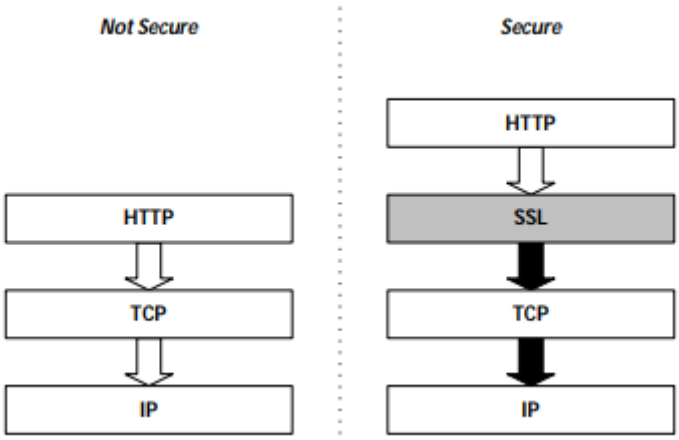
2.2 SSL u mrežnoj arhitekturi

Postoji više mogućih pristupa koji omogućuju sigurnu razmjenu podataka na internetu. Pristupi visoke razine zastupljenosti su: *Separate Security Protocol* koji se koristi u obliku zasebnog sloja integriranog u samu arhitekturu internetskog protokola i *Application Specific Security* koji se koristi na način da se sigurnosni servisi integriraju direktno u aplikacijski protokol. Tablični prikaz protokola za svaki pristup prikazan je u tablici *2.1*.

Tablica 2. 1: Pristupi mrežnoj sigurnosti i primjeri [4]

Arhitektura	Primjer
Separate Protocol Layer	SSL
Application Layer	S-HTTP
Integrated with Core	IPSEC
Parallel Protocol	Kerberos

Kao što je vidljivo u tablici, odlučeno je SSL razviti u obliku zasebnog protokolskog sloja (*engl. separate protocol layer*), što dovodi do zaključka da je SSL razvijen kao individualni sigurnosni sloj koji se integrira u arhitekturu internetskih protokola između slojeva zasnovanih na HTTP protokolu (*engl. HyperText Transfer Protocol*) i TCP protokolu (*engl. Transmission Control Protocol*). HTTP pripada aplikativnoj skupini protokola koji je esencijalan za početnu fazu prijenosa informacija na internetu, a TCP je protokol koji se bavi izgradnjom servisa internetskog protokola u svrhu osiguravanja stabilnosti komunikacije. [5] Slika *2.2* prikazuje navedenu tezu shematski. Osim HTTP protokola, dužnost početnog sloja u mogućnosti su obavljati i NNTP (*engl. Net News Transfer Protocol*) i FTP (*engl. File Transfer Protocol*).



Slika 2. 2: Slojevi internetske arhitekture [6]

2.3 SSL uloge

U SSL protokolu definirano je da postoje dvije uloge. Jedna je klijent, a druga je poslužitelj. Ispravna primjena SSL protokola zahtjeva dva sustava koja nastoje komunicirati. Bitno je da su im uloge različite, gdje jedan sustav mora biti klijent, a drugi poslužitelj. Klijentom se može smatrati sustav koji ima ulogu inicijacije same kriptirane komunikacije, dok poslužitelj ima ulogu odgovaranja na dobivene zahtjeve. Gledajući kroz praktičnu primjenu, SSL protokol se najčešće koristi prilikom sigurnog pregledavanja interneta pri čemu internetski preglednici obnašaju ulogu klijenata, a internetske stranice služe kao oblici poslužitelja.

2.4 SSL poruke

Pomoću razmjene većeg broja individualnih poruka dolazi do pojave komunikacije koja se događa između klijenata i poslužitelja, pri čemu je svaka pojedina poruka zasebno definirana. Tablični prikaz različitih oblika formata mogućih poruka prikazan je u tablici *2.2*.

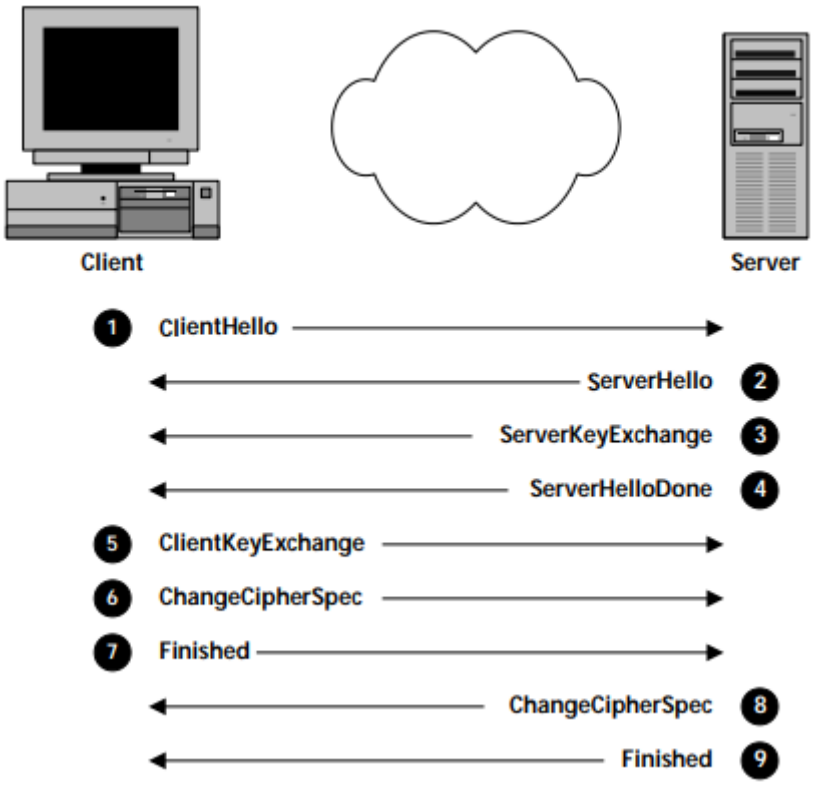
Tablica 2.2: Popis i kratki opis SSL poruka

Poruka	Opis
Alert	Poruka koja obavještava drugi sustav o mogućoj grešci ili sigurnosnom propustu za vrijeme komunikacije
AplicationData	Podaci koje izmjenjuju klijent i poslužitelj u kriptiranom, autentificiranom, te provjerenom stanju
Certificate	Poruka koja u sebi sadrži pošiljateljev certifikat s javnim ključem
CertificateRequest	Poruka kojom poslužitelj klijentu šalje zahtjev za certifikat
CertificateVerify	Poruka kojom klijent dokazuje poslužitelju da posjeduje privatni ključ koji odgovara javnom ključu certifikata
ChangeCipherSpec	Poruka koja obavještava o početku korištenja ugovorenih sigurnosnih servisa
ClientHello	Poruka kojom klijent započinje komunikaciju s poslužiteljem i javlja koje sigurnosne servise može koristiti
ClientKeyExchange	Poruka kojom klijent šalje kriptografske ključeve uz pomoć kojih će se odvijati kriptirana komunikacija
Finished	Poruka koja obavještava drugi sustav o uspješnom dogovoru i uspješno uspostavljenoj kriptiranoj komunikaciji
HelloRequest	Poruka kojom poslužitelj traži klijenta za dogovor oko odabira sigurnosnih servisa
ServerHello	Poruka kojom poslužitelj obavještava klijenta s kojim sigurnosnim servisima će biti postignuta komunikacija
ServerHelloDone	Poruka kojom poslužitelj obavještava klijenta da su svi zahtjevi, potrebni za kriptiranu komunikaciju, poslani
ServerKeyExchange	Poruka kojom poslužitelj šalje kriptografske ključeve uz pomoć kojih će se odvijati kriptirana komunikacija

3. RAZLIČITI NAČINI USPOSTAVE KOMUNIKACIJE

3.1 Uspostava kriptirane komunikacije

Osnovna mogućnost koju pruža SSL protokol je uspostava kriptirane komunikacije između klijenta i poslužitelja. Prilikom ove uspostave kriptirane komunikacije ni klijent ni poslužitelj nisu autentificirani. Slika *3.1* vizualno prikazuje slijed devet potrebnih SSL poruka za ostvarenje kriptirane komunikacije. Prilikom ostvarivanja kriptirane komunikacije pet poruka poslano je od strane poslužitelja ka klijentu dok su četiri SSL poruke poslane od strane klijenta ka poslužitelju.



Slika 3.1: Koraci pri uspostavi kriptirane komunikacije [7]

Tablica 3.1: Kratki opis koraka sa slike 3.1

1	Klijent poslužitelju predlaže SSL parametre
2	Poslužitelj izabire SSL parametre
3	Poslužitelj šalje svoj javni ključ
4	Poslužitelj završava pregovaranje s klijentom
5	Klijent kriptira ključ sjednice javnim ključem poslužitelja i šalje mu ga
6	Klijent aktivira sigurnosne opcije za nadolazeće poruke
7	Klijent javlja poslužitelju da je uspostava komunikacije uspješno provedena
8	Poslužitelj aktivira sigurnosne opcije za nadolazeće poruke
9	Poslužitelj javlja klijentu da je uspostava komunikacije uspješno provedena

3.1.1 **ClientHello**

ClientHello poruka označava pozdravnu poruku koju klijent šalje poslužitelju ukoliko ima namjeru uspostaviti komunikaciju s istim. Ukoliko jedan sustav ima namjeru uspostaviti kriptiranu komunikaciju, u tom slučaju njena uloga je klijent i on šalje *ClientHello* poruku prema drugom sustavu, koji je u tom slučaju poslužitelj. Klijent započinje dogovor s poslužiteljem oko sigurnosnih servisa koje će koristiti prilikom kriptirane komunikacije. Tablica *3.2* prikazuje parametre koji su proslijeđeni u *ClientHello* poruci.

Tablica 3.2: Popis i kratki opis parametara iz *ClientHello* poruke

Parametar	Opis
Version	Najveća SSL verzija koju klijent podržava
RandomNumber	Nasumični broj koji se koristi prilikom kriptiranja
SessionID	Identifikator SSL sjednice
CipherSuites	Popis kriptografskih servisa koje klijent podržava
CompressionMethods	Kompresijske metode podržane od strane klijenta

Parametar *Version* predstavlja najveći broj SSL verzije koju klijent ima mogućnost podržati. Odluka o mogućnosti podržavanja verzije ovisi o inačici iste. Komunikacija pokrenuta od strane klijenta podrazumijeva da se prenošenje poruka čini verzijom SSL-a koja je na istoj razini ili nižoj razini kao i verzija SSL-a koju podržava poslužitelj. Prvi dio ove poruke jest da klijent prikaže poslužitelju koje je inačice SSL protokola u mogućnosti koristiti. Nadalje, klijent šalje poslužitelju *RandomNumber,* nasumičan 32-bajtni broj koji je zadužen za enkripciju same poruke. Nasumičan broj također sadrži datum i vrijeme u 4-bajtnom formatu što osigurava jedinstvenost svakoga broja. Parametar *SessionID* je u ovom slučaju prazan. U parametru *CipherSuites* izlistan je popis kriptografskih servisa, algoritama i veličina ključeva koje klijent

podržava putem kojeg će poslužitelj odlučiti koji će se princip kasnije koristiti u procesu komunikacije. Kompresijske metode podrazumijevaju skupove procesa zaduženih za kompresiju podataka prije samoga čina slanja poruke iz razloga što se pri enkripciji nad podacima vrše razni matematički procesi.

3.1.2 **ServerHello**

Nakon što klijent započne komunikaciju s porukom *ClientHello*, poslužitelj odgovara s porukom *ServerHello*. Parametri poruke *ServerHello* prikazani su u tablici *3.3*. Slične su konstrukcije kao i parametri poruke *ClientHello*. Razlike će biti navedene nakon tablice u objašnjenjima parametara proslijeđenim u poruci. Glavna svrha ove poruke jest njeno sadržavanje sigurnosnih servisa za koje se poslužitelj odlučio iz liste koju je klijent ponudio u poruci *ClientHello*.

Tablica 3.3: Popis i kratki opis parametara iz *ServerHello* poruke

Parametar	Opis
Version	Verzija za koju se poslužitelj odlučio
RandomNumber	Broj koji se koristi prilikom kriptiranja
SessionID	Identifikator SSL sjednice
CipherSuite	Kriptografski servisi za koje se poslužitelj odlučio
CompressionMethod	Kompresijska metoda za koju se poslužitelj odlučio

U ovom slučaju parametar *Version* je broj verzije protokola za koju se odlučio poslužitelj. Iako je poslužitelj taj koji odabire kojom verzijom SSL-a će biti komunikacija uspostavljena, ne može izabrati bilo koju verziju. Dostupne verzije za ostvarenje kriptirane komunikacije su one koje je klijent poslao u poruci *ClientHello* pod istoimenim parametrom, *Version*. *RandomNumber*, nasumični broj kao i u *ClientHello* poruci potreban je za kriptografske izračune. Također, jednakog je formata kao i u *ClientHello* poruci, dakle sadrži 32 bajta od kojih su četiri zadužena za prikaz vremena i datuma, dok je ostatak broja generiran pomoću kriptografskog sigurnosnog generatora nasumičnih brojeva. Parametar *SessionID,* identifikator sjednice u ovom slučaju može sadržavati vrijednost, što nije slučaj u *ClientHello* poruci. Ova vrijednost je jedinstvena i služi prilikom identifikacije određene sjednice. U potpoglavlju *3.6* je detaljnije objašnjeno korištenje *SessionID* parametra pri korištenju kriptirane komunikacije iz prethodne sjednice. *CipherSuite* parametar predstavlja koji su sigurnosni servisi, točnije algoritmi i veličine ključeva, odabrani sa strane poslužitelja. Poslužitelj je mogao birati iste iz liste koju je klijent predložio parametrom *CipherSuites* u *ClientHello* poruci.

3.1.3 **ServerKeyExchange**

ServerKeyExchange poruka slijedi neposredno nakon *ServerHello* poruke te također dolazi s poslužiteljske strane. Ova poruka smatra se nadopunom parametru *CipherSuite* iz prethodne *ServerHello* poruke. U ovoj poruci nalazi se točan format ključeva potrebnih za enkripciju. Valja uzeti u obzir da ova poruka nije kriptirana te je stoga jedino siguran prijenos javnog ključa.

3.1.4 **ServerHelloDone**

ServerHelloDone poruka slijedi nakon *ServerKeyExchange* te također dolazi s poslužiteljske strane. Zadaća ove poruke jest obavijestiti klijenta da je poslužitelj završio s biranjem sigurnosnih servisa te da je spreman za iduću fazu.

3.1.5 **ClientKeyExchange**

Nakon poslužiteljeve *ServerHelloDone* poruke, klijent šalje poruku *ClientKeyExchange*. U poslužiteljevoj poruci *ServerKeyExchange*, poslužitelj je poslao svoj javni ključ. U *ClientKeyExchange* poruci klijent šalje ključ sjednice (*engl. session key*) koji će se koristiti u simetričnoj enkripciji i isti ključ će koristiti i klijent i poslužitelj. Razlika u odnosu na *ServerKeyExchange* poruku je i ta što je u ovom slučaju poruka kriptirana i to javnim ključem koji je poslužitelj poslao. Ovime se također verificira da poslužitelj ima privatni ključ koji odgovara javnom ključu koji je klijent primio, jer u protivnom poslužitelj neće biti u mogućnosti dekriptirati poruku koju mu je klijent poslao.

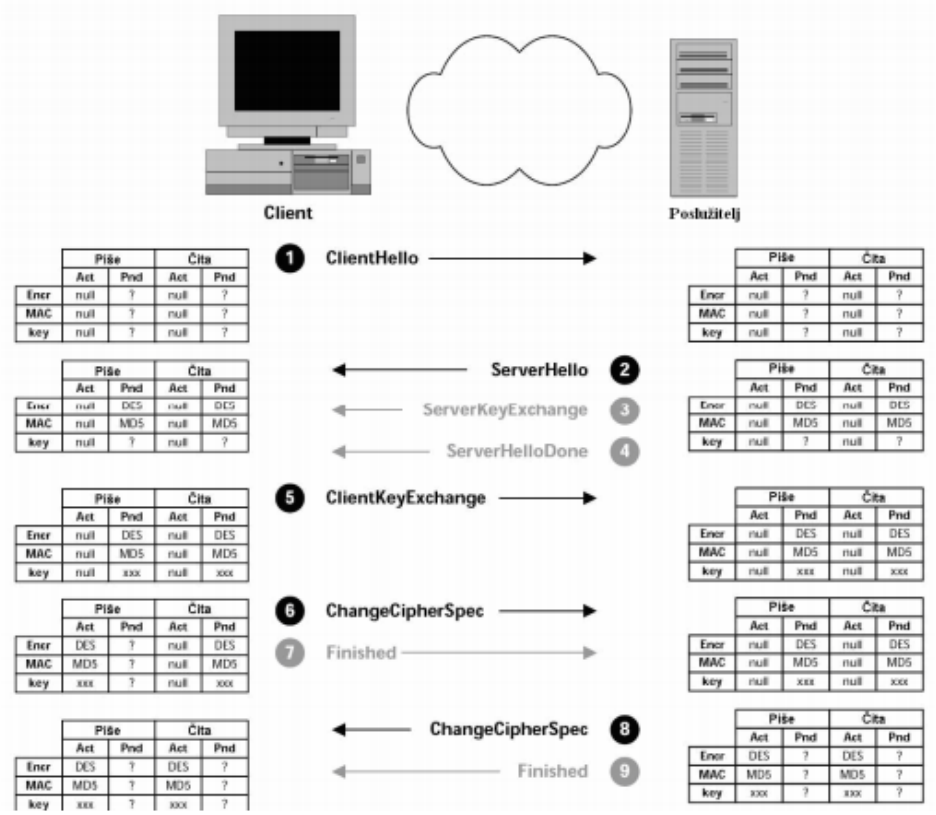
3.1.6 **ChangeCipherSpec**

Nakon klijentove *ClientKeyExchange* poruke, pregovori oko sigurnosnih servisa su završeni. Sada započinje njihovo korištenje. Poruka *ChangeCipherSpec* obavještava da će sve poruke od sada biti poslane koristeći sigurnosne servise. Prelazak na sigurnu komunikaciju je vrlo osjetljiva radnja. „*Definira se simetrični algoritam kriptiranja, algoritam za izračunavanje integriteta poruke i materijal za izračun ključeva za te algoritme*.“ [8] Ključevi klijenta i poslužitelja su različiti. „*Za svaki sustav, i klijentov i poslužiteljev, SSL definira stanje čitanja i stanje pisanja. Stanje čitanja definira sigurnosne informacije za podatke koje sustav prima, a stanje pisanja definira sigurnosne informacije za podatke koje sustav šalje.*“ [9]

„*ChangeCipherSpec služi kao znak da sustav počne koristi sigurnosne informacije. No prije nego što pošalje poruku mora znati sve sigurnosne informacije koje će aktivirati. Nakon što klijent pošalje ChangeCipherSpec poruku aktivira svoje stanje pisanja, a kad poslužitelj primi ChangeCipherSpec poruku aktivira stanje čitanja*.“ [10]

Slika *3.2* prikazuje stanja pisanja i čitanja poslužitelja i klijenta. „*SSL definira dva stanja čitanja i dva stanja pisanja. Jedno stanje je aktivno, a drugo je stanje u očekivanju. Iz toga slijedi da klijent i poslužitelj sadrže po četiri stanja. Aktivno pisanje, aktivno čitanje, u očekivanju pisanja i u očekivanju čitanja. Također su prikazani i najvažniji dijelovi stanja. To su algoritam kriptiranja, algoritam za provjeru integriteta poruke i podaci za ključ*.“ [11] U primjeru na slici *3.2* DES se koristi kao simetrični algoritam kriptiranja, MD5 kao algoritam za provjeru integriteta poruke. „*Svaki sustav počinje u aktivnom stanju bez definiranih algoritama. To se i podrazumijeva, jer dok se strane ne dogovore o sigurnosnim parametrima, ne može doći do sigurne komunikacije. Kako se razmjenjuju poruke sustavi prvo izgrađuju stanja u očekivanju. Prvo se dogovaraju oko algoritama, a zatim izmjenjuju podatke za ključ. Tek tada sustav ta stanja u očekivanju aktivira sa ChangeCipherSpec porukom i ona prelaze u aktivna stanja*.“ [12]

U nastavku slijede tablice *3.4* i *3.5* koje prikazuju promjene kod klijenta i promjene kod poslužitelja.



Slika 3.2 Koraci pri dogovoru sigurnosnih postavki [13]

Tablica 3.4: Kratki opis klijentovih koraka sa slike 3.2

1	Klijent nakon *ClientHello* poruke postavlja svoja aktivna stanja na *null* vrijednost što znači da sigurnosne postavke još ne postoje. Stanja čekanja su nepoznata.
2	Nakon *ServerHello* poruke klijent saznaje koji algoritmi će se koristiti za komunikaciju. Stanja mijenja u stanje čekanja i postavlja odabrane algoritme.
5	Nakon *ClientKeyExchange* poruke klijent zna koji će se ključevi koristiti prilikom komunikacije i stanja u stanju čekanja ažurira.
6	Nakon klijentove *ChangeCipherSpec* poruke, klijent prebacuje iz stanja u očekivanju pisanja u aktivno stanje pisanja, a stanje u očekivanju pisanja postavlja na nepoznato. Nikakve promjene nisu napravljene na stanju čitanja. Ovo znači da će od sada sve klijentove poruke biti poslane kodirane DES algoritmom i verificirane MD5 algoritmom
8	Nakon poslužiteljeve *ChangeCipherSpec* poruke klijent ažurira stanje čitanja i postavke iz stanja čekanja prebacuje u aktivno stanje. Ovime je poslužitelj obavijestio klijenta da će sve poruke koje ubuduće dobije biti kodirane DES algoritmom, a MD5 algoritmom verificirane.

Tablica 3.5: Kratki opis poslužiteljevih koraka sa slike 3.2

1	Poslužitelj nakon *ClientHello* poruke postavlja svoja aktivna stanja na *null* vrijednost. Stanja čekanja su nepoznata.
2	Nakon *ServerHello* poruke poslužitelj zna koji se algoritmi budu koristili prilikom kriptiranja i provjere integriteta poruke, ovisno o tome ažurira stanja u stanju čekanja.
5	Nakon klijentove *ClientKeyExchange* poruke poslužitelj zna točan format za ključeve i ovisno o tome ažurira stanja u stanju čekanja.
6	Nakon klijentove *ChangeCipherSpec* poruke, poslužitelj zna da će klijent od sada koristiti sigurnosne postavke i u skladu s tim postavlja u aktivno stanje čitanja postavke koje su bile u očekivanju stanja pisanja.
8	Nakon poslužiteljeve *ChangeCipherSpec* poruke poslužitelj postavlja svoje aktivno stanje pisanja na postavke koje su bile pod stanjem očekivanje pisanja. Poslije ove poruke sve koje će slati koristiti će sigurnosne postavke iz aktivnog stanja pisanja.

Važno je razumjeti da je na slici *3.2* označeno da u trenutku kad jedna strana ažurira stanje pisanja, druga ažurira stanje čitanja. U stvarnosti to nije tako. „*Jedna strana će ažurirati svoje stanje kad pošalje poruku, dok će druga to učiniti tek kad primi poruku*.“ [14]

3.1.7 **Finished**

Poruku *Finished* i klijent i poslužitelj šalju odmah nakon vlastite *ChangeCipherSpec* poruke. Zadaća ove poruke jest javiti drugoj strani da su dogovori oko uspostavljanja kriptirane komunikacije uspješno provedeni, te ova poruka još dodatno doprinosi sigurnosti. „*Dva aspekta poruke pridonose sigurnosti. Finished je prva poruka poslana nakon što su strane odlučile prijeći na sigurnosnu komunikaciju, pa su samim time i na njoj primijenjeni dogovoreni sigurnosni mehanizmi. Ako strana koja primi poruku nije u mogućnosti dekriptirati ju i verificirati, znači da je došlo do sigurnosnog propusta*.“ [15]

„*Sadržaj poruke također pridonosi sigurnosti. Finished poruka sadrži sažetak koji je izveden od svih poruka koje su sudjelovale u Handshake postupku*.“ [16] Ovaj postupak bit će detaljnije objašnjen u podnaslovu *4.1.1*. „*Ako bi napadač koji nije sudjelovao u razmjeni poruka pokušao izvesti napad, ne bi mogao jer nema sve prijašnje poruke i takav napad bi bio razotkriven*.“ [17]

3.2 Završavanje kriptirane komunikacije

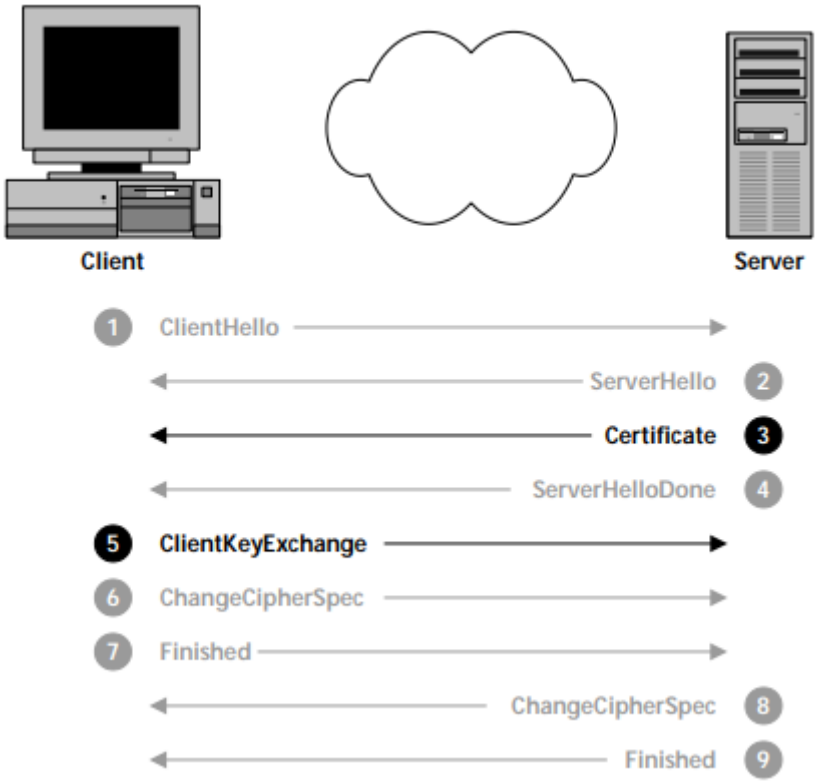
Završetak kriptirane komunikacije između klijenta i poslužitelja također je jasno definiran. Svaki sustav pojedinačno u ovome slučaju šalje *ClosureAlert* poruku koja služi za sprječavanje napada u obliku prekida poruke prije njenog svršetka i na taj način stvaranja neželjenih posljedica. Primjerice, ako poruka glasi „Obriši sve podatke ako se ne javim do sutra“, bez uporabe sigurnosne *ClosureAlert* poruke dolazi do mogućnosti da napadač presiječe poruku i iščita se samo dio koji glasi „Obriši sve podatke“. Uvođenjem *ClosureAlert* poruke zadužene za ovaj oblik sigurnosti, nestaje mogućnost takvog primjera napada.

3.3 Autentifikacija poslužitelja

U potpoglavlju *3.1* detaljno je objašnjena uspostava kriptirane komunikacije između dvije strane, klijenta i poslužitelja, no to samo po sebi nije dovoljno sigurna metoda komunikacije. Prilikom kriptirane komunikacije ni jedna strana ne može znati identitet druge. Razlog korištenja enkripcije jest da nitko osim klijenta i poslužitelja nema pristup podacima, no ako se netko prevarom uspješno predstavi kao sustav koji inače sudjeluje u komunikaciji, enkripcija više ne služi svojoj svrsi. Podaci bi u tom slučaju bili kriptirani, no napadač bi imao sve potrebne informacije za dekripciju istih. Upravo iz tog razloga SSL ima mehanizam pomoću kojeg jedan sustav može autentificirati drugi.

Ovo potpoglavlje će objasniti uspostavu kriptirane komunikacije prilikom autentifikacije poslužitelja. Slijedi primjer kada je u SSL komunikaciji važno izvršiti autentifikaciju poslužitelja; u situaciji kupovine proizvoda putem internetske trgovine potrebno je internetskoj stranici dozvoliti pristup osjetljivim podacima o kreditnoj kartici kako bi se kupnja uspješno izvršila. Do problema dolazi kada napadač pokuša imitirati stranicu internetske trgovine. U ovom slučaju svi osjetljivi podaci kreditne kartice bi bili pristupačni napadaču. Rješenje ovog problema leži u autentifikaciji poslužitelja, u ovom slučaju internetske stranice, dakle internetske aplikacije. Osim autentifikacije poslužitelja moguća je i autentifikacija klijenta, no u ovom slučaju nije potrebna iz razloga što internetskoj trgovini nije važno je li klijent autentificiran. Podaci nad kojima se vrši provjera su podaci vezani uz plaćanje. U potpoglavlju *3.5* slijedi detaljno objašnjenje autentifikacije klijenta.

Proces autentifikacije poslužitelja sličan je običnoj kriptiranoj komunikaciji. Razlika je u dvije poruke, *ClientKeyExchange* i *ServerKeyExchange* koja je zamijenjena porukom *Certificate*. Na slici *3.3* su istaknute poruke s razlikama dok su u tablici *3.6* kratko opisani svi koraci.



Slika 3.3: Koraci pri uspostavi kriptirane komunikacije uz autentifikaciju poslužitelja [18]

Tablica 3.6: Kratki opis koraka sa slike 3.3

1	Klijent poslužitelju predlaže SSL parametre
2	Poslužitelj izabire SSL parametre
3	Poslužitelj šalje certifikat s javnim ključem
4	Poslužitelj završava pregovaranje s klijentom
5	Klijent kriptira ključ sjednice javnim ključem poslužiteljevog certifikata i šalje mu ga
6	Klijent aktivira sigurnosne opcije za nadolazeće poruke
7	Klijent javlja poslužitelju da je uspostava komunikacije uspješno provedena
8	Poslužitelj aktivira sigurnosne opcije za nadolazeće poruke
9	Poslužitelj javlja klijentu da je uspostava komunikacije uspješno provedena

3.3.1 **Certificate**

Kako bi postigao autentifikaciju, poslužitelj klijentu šalje *Certificate* poruku, umjesto *ServerKeyExchange* poruke čiji je princip rada prikazan u prvom primjeru u odjeljku *3.1.3*. Poruka sadrži certifikat koji počinje javnim ključem, a završava certifikatom certifikacijskog centra (*engl. certificate authorities*). Klijentova odgovornost jest da se uvjeri u ispravnost certifikata i da stvori povjerenje prema poslužitelju. To znači da je potrebno izvršiti verifikaciju

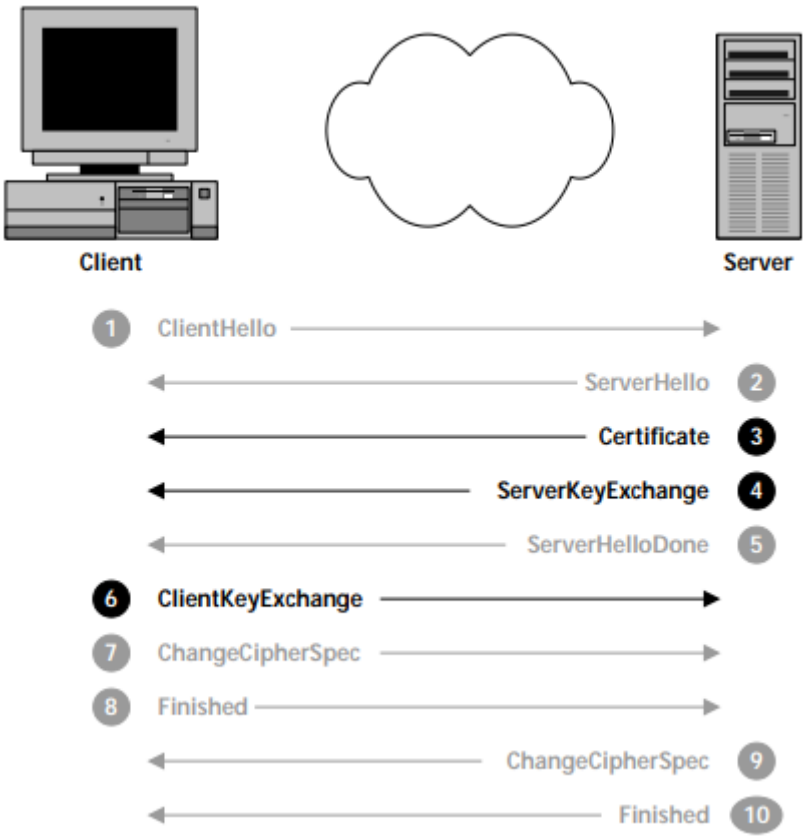
potpisa i provjeriti ispravnost opoziva i vremenskih rokova. Također, potrebno je uvjeriti se kako je certifikacijski centar uistinu vjerodostojan. Verifikacija certifikata ključan je segment pri ostvarenju sigurnosti. [19]

3.3.2 **ClientKeyExchange**

Iako nije velika, postoji razlika u procesu s autentifikacijom poslužitelja koristeći *ClientKeyExchange* poruku naspram procesa kod kojeg se ne koristi autentifikacija. Pri procesu bez autentifikacije klijent podatke koji se nalaze u ovoj poruci kriptira putem javnog ključa koji mu je poslan od strane poslužitelja u *ServerKeyExchange* poruci. Ovdje se vrši autentifikacija poslužitelja, te je stoga poslan certifikat. Nadalje, korišten je javni ključ sadržan u certifikatu i na taj način je osigurano da isključivo poslužitelj koji mu je poslao taj certifikat ima mogućnost dekriptiranja njegove poruke.

3.4 Odvajanje enkripcije i autentifikacije

U potpoglavlju *3.3* prikazan je način autentifikacije poslužitelja na način da umjesto poslužiteljeve poruke *ServerKeyExchange*, koja se koristi prilikom jednostavne uspostave kriptirane komunikacije, bude poslana *Certificate* poruka u kojoj se nalazi certifikat. Takav pristup ima svoju manu. Isti javni ključ se koristi za enkripciju i autentifikaciju. Problem nastaje kada npr. javni ključ algoritma može biti korišten samo za potpisivanje, a ne i za enkripciju. „*Some Public key algorithms (such as the Digital Signature Algorithm) can only be used for signing. By their very design, they cannot be used for encryption.*“. [20] SSL protokol nudi rješenje navedenom problemu. Rješenje se postiže odvajanjem enkripcije i autentifikacije. Pri usporedbi slike *3.3* i *3.4* moguće je zaključiti da je prilikom odvajanja enkripcije od autentifikacije potrebno proslijediti deset SSL poruka umjesto devet. Dodana poruka je *ServerKeyExchange* poruka. Ključna razlika za odvajanje enkripcije i autentifikacije je u *Certificate*, *ServerKeyExchange* i *ClientKeyExchange* porukama. U nastavku će biti objašnjenje navedenih SSL poruka. U tablici *3.7* su kratki opisi koraka sa slike *3.4*.



Slika 3.4: Koraci pri uspostavi kriptirane komunikacije uz autentifikaciju poslužitelja s odvojenom autentifikacijom od enkripcije [21]

Tablica 3.7: Kratki opis koraka sa slike 3.4

1	Klijent poslužitelju predlaže SSL parametre
2	Poslužitelj izabire SSL parametre
3	Poslužitelj šalje certifikat s javnim ključem
4	Poslužitelj šalje javni ključ koji je potpisan javnim ključem certifikata
5	Poslužitelj završava pregovaranje s klijentom
6	Klijent kriptira ključ sjednice javnim ključem poslužitelja koji je potpisan javnim ključem poslužiteljevog certifikata i šalje mu ga
7	Klijent aktivira sigurnosne opcije za nadolazeće poruke
8	Klijent javlja poslužitelju da je uspostava komunikacije uspješno provedena
9	Poslužitelj aktivira sigurnosne opcije za nadolazeće poruke
10	Poslužitelj javlja klijentu da je uspostava komunikacije uspješno provedena

3.4.1 **Certificate**

U ovom slučaju ova poruka je vrlo slična *Certificate* poruci u odjeljku *2.9.1*. Jedina razlika je u tome što se certifikatov javni ključ koristi samo prilikom autentifikacije, dok se prilikom same

autentifikacije bez odvojene enkripcije javni ključ koristi i za enkripciju. Klijent i dalje mora provjeriti ispravnost certifikata te autentičnost certifikacijskog centra koji ga je izdao.

3.4.2 **ServerKeyExchange**

Nakon poslužiteljeve *Certificate* poruke slijedi njegova *ServerKeyExchange* poruka. U njoj se nalazi drugi javni ključ koji klijent koristi prilikom kriptiranja ključa sjednice. *ServerKeyExchange* je poruka istog formata koja se koristi prilikom uspostave kriptirane komunikacije bez autentifikacije u odjeljku *3.1.3*. Razlika je u tome što se u ovome slučaju poslužiteljev javni ključ prije slanja klijentu kriptira javnim ključem iz poslužiteljevog certifikata. Zbog ovog koraka klijent se može uvjeriti u autentičnost poslužitelja jer bez odgovarajućeg privatnog ključa koji odgovara javnom ključu certifikata poslužitelj ne može dekriptirati daljnje poruke, te se nije uspješno autentificirao.

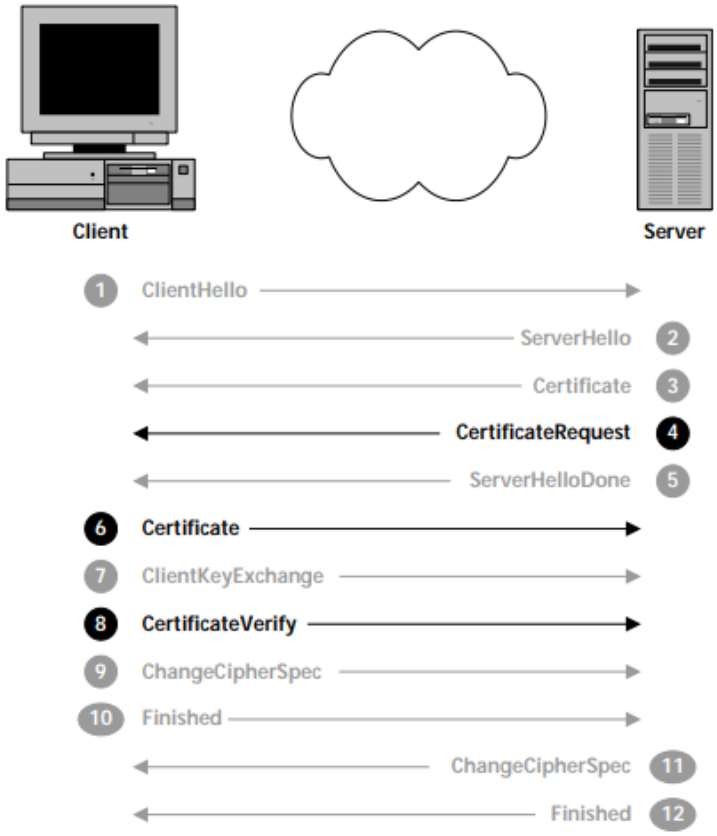
3.4.3 **ClientKeyExchange**

U *ClientKeyExchange* poruci kao i u ostalim primjerima klijent šalje informacije o ključu sjednice za odlučeni algoritam simetrične enkripcije (*engl. symmetric encryption*). Važno je razumjeti da je u ovome slučaju javni ključ koji je korišten prilikom enkripcije ključa sjednice zapravo javni ključ iz poslužiteljeve *ServerKeyExchange* poruke, a ne javni ključ iz poslužiteljeve *Certificate* poruke.

3.5 Autentifikacija klijenta

U potpoglavljima *3.3* i *3.4* prikazana je problematika autentifikacije poslužitelja, no SSL protokol nudi i opciju autentifikacije klijenta. Važno je znati da bez autentifikacije samoga poslužitelja nije moguće zatražiti autentifikaciju klijenta. Internetske stranice koriste autentifikaciju klijenta kako bi bile u mogućnosti odabrati određene klijente koji će se njome moći služiti. Organizacija ne želi da itko osim ovlaštenih osoba ima mogućnost pristupa informacijama koje se nalaze na internetskoj stranici. Štoviše, u svrhu još veće razine sigurnosti, sami zaposlenici ili ovlaštene osobe podacima sa internetske stranice mogu pristupiti samo putem službenih uređaja dodijeljenih od strane same organizacije; „*For example, an organization's intranet website typically exists for their employees to access information and communicate on official matters. The organization doesn't want anyone else to access such an internal website and would like to restrict the audience. Moreover, the employees should be*

accessing that website from their official devices only to further mitigate the risk of unsolicited access. In such cases, the organization can use a two-way SSL certificate to authenticate the clients before letting them access the website.“ [22] Na slici *3.5* prikazan je slijed SSL poruka potreban za ostvarenje autentifikacije klijenta dok tablica *3.8* prikazuje kraći opis svake poruke.



Slika 3.5: Koraci pri uspostavi kriptirane komunikacije uz obostranu autentifikaciju [23]

Tablica 3.8: Kratki opis koraka sa slike 3.5

1	Klijent poslužitelju predlaže SSL parametre
2	Poslužitelj izabire SSL parametre
3	Poslužitelj šalje certifikat s javnim ključem
4	Poslužitelj šalje zahtjev za klijentovim certifikatom
5	Poslužitelj završava pregovaranje s klijentom
6	Klijent šalje certifikat s javnim ključem
7	Klijent kriptira ključ sjednice javnim ključem poslužitelja koji je potpisan javnim ključem poslužiteljevog certifikata i šalje mu ga
8	Klijent potpisuje informacije o sjednici privatnim ključem certifikata i šalje ih poslužitelju
9	Klijent aktivira sigurnosne opcije za nadolazeće poruke
10	Klijent javlja poslužitelju da je uspostava komunikacije uspješno provedena
11	Poslužitelj aktivira sigurnosne opcije za nadolazeće poruke
12	Poslužitelj javlja klijentu da je uspostava komunikacije uspješno provedena

3.5.1 **CertificateRequest**

Poslužitelj je strana koja može zatražiti autentifikaciju klijenta, dok klijent nema kontrolu nad time hoće li biti zatražena njegova autentifikacija. Upravo s *CertificateRequest* porukom poslužitelj zahtjeva autentifikaciju od klijenta. Ona je poslana odmah nakon poslužiteljeve *Certificate* poruke. *CertificateRequest* poruka sadrži dva parametra:

- *CertificateTypes*
- *Distinguished-Names*

Parametar *CertificateTypes* predstavlja listu tipova certifikata koje poslužitelj prihvaća dok parametar *Distinguished-Names* predstavlja listu izdavatelja certifikata koje poslužitelj prihvaća.

3.5.2 **Certificate**

Nakon što poslužitelj zatraži autentifikaciju, klijent obično odgovara *Certificate* porukom istog formata poslužiteljevoj *Certificate* poruci opisanom u *3.3.1* odjeljku. Ukoliko klijent ne posjeduje certifikat koji poslužitelj prihvaća, ili ako uopće nema certifikat, odgovara

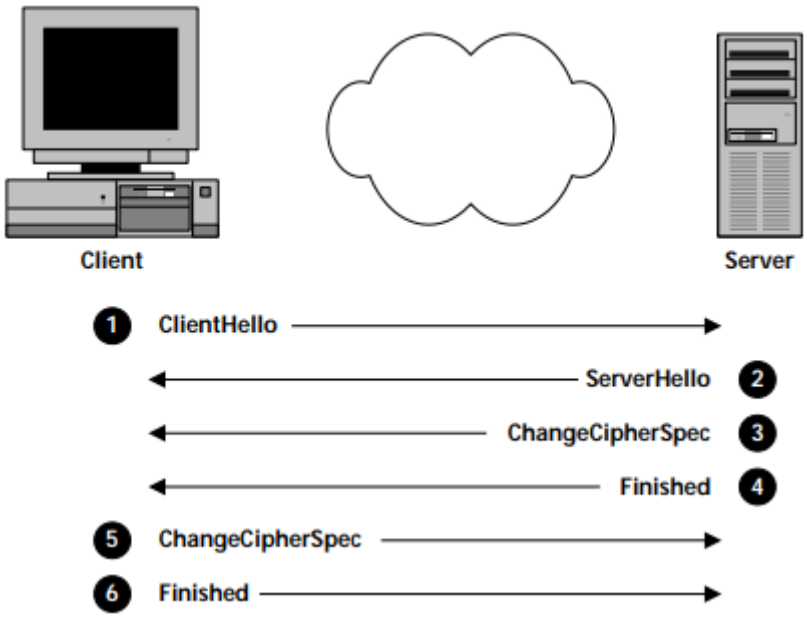
NoCertificateAlert porukom. Nakon toga poslužitelj ima izbor nastaviti uspostavljanje SSL komunikacije s neautentificiranim klijentom ili ju obustaviti. SSL protokol nema mogućnost odvajanja enkripcije i autentifikacije prilikom klijentove autentifikacije iz razloga što to nije potrebno. Klijentov javni ključ se koristi za digitalni potpis, ali ne i za enkripciju; „*U SSL specifikaciji klijentov javni ključ se koristi samo za potpisivanje, nikad za kriptiranje podataka, pa stoga nema potrebe, kao kod poslužitelja za razdvajanje autentifikacije i kriptiranja.“* [24]

3.5.3 **CertificateVerify**

Nakon klijentove *Certificate* poruke, klijent i dalje nije autentificiran. Potrebno je poslužitelju dokazati da klijent ima odgovarajući privatni ključ koji odgovara javnom ključu certifikata. Klijent to radi na način da digitalno potpiše informacije dostupne samo klijentu i poslužitelju te ih pošalje poslužitelju u *CertificateVerify* poruci. To se događa na način da se izvrši potpis informacije o ključu (*engl. key information*) i informacije o prethodno poslanim porukama prilikom uspostave komunikacije. Pošto poslužitelj ima sve potpisane informacije i ima klijentov javni ključ koji je dobio u klijentovoj poruci *Certificate*, u mogućnosti je verificirati digitalni potpis tj. uvjerit se da klijent uistinu posjeduje odgovarajući privatni ključ.

3.6 Ponovno korištenje sjednice

Iz ovog poglavlja moguće je zaključiti da je uspostavljanje kriptirane komunikacije putem SSL protokola kompleksan proces. Kako se ne bi prilikom svake interakcije klijenta i poslužitelja morala vršiti u potpunosti nova uspostava komunikacije i nepotrebno trošenje resursa, osmišljen je mehanizam prilikom kojeg klijent i poslužitelj mogu koristiti sve izglasane i korištene parametre iz prethodne sjednice. Na ovaj način se zaobilaze kriptografski izračuni i smanjuje se broj proslijeđenih SSL poruka. Slika *3.6* prikazuje slijed poruka koje su potrebne za uspostavu komunikacije pomoću parametara dogovorenih u prethodnoj sjednici. Tablica *3.9* prikazuje kratki opis tih poruka.



Slika 3.6: Koraci pri uspostavi kriptirane komunikacije iz prethodne sjednice [25]

Tablica 3.9: Kratki opis koraka sa slike 3.6

1	Klijent poslužitelju predlaže korištenje prethodne sjednice
2	Poslužitelj odlučuje koristiti sigurnosne servise iz prethodne sjednice
3	Poslužitelj aktivira sigurnosne opcije za nadolazeće poruke iz prethodne sjednice
4	Poslužitelj javlja klijentu da je uspostava komunikacije uspješno provedena
5	Klijent aktivira sigurnosne opcije za nadolazeće poruke iz prethodne sjednice
6	Klijent javlja poslužitelju da je uspostava komunikacije uspješno provedena

Klijent započinje komunikaciju s *ClientHello* porukom na način da zatraži korištenje prethodne sjednice tako da u parametru *SessionID* postavi vrijednost koja je identifikator prethodne sjednice. Zatim poslužitelj u slučaju prihvaćanja korištenja prethodne sjednice šalje u parametru *SessionID ServerHello* poruke jednaku vrijednost kakvu je klijent poslao u *ClientHello* poruci. Ako poslužitelj odluči ne prihvatiti korištenje prethodne sjednice, u *SessionID* parametar postavlja drugačiju vrijednost te nastupa jedan od načina ponovnog uspostavljanja komunikacije objašnjenog u prijašnjim potpoglavljima *3.1*, *3.3*, *3.4* i *3.5*.

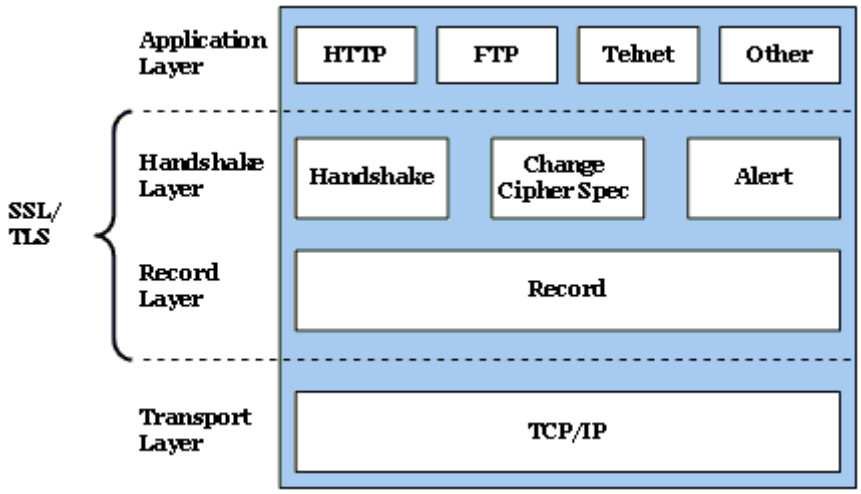
Nakon što klijent i poslužitelj dogovore da će koristiti prethodnu sjednicu, šalju vlastiti par *ChangeCipherSpec* i *Finished* poruka. Time odluče koristiti algoritme dogovorene u prethodnoj sjednici.

4. SLOJEVI SSL PROTOKOLA

U poglavlju *3* objašnjeno je korištenje SSL poruka prilikom različitih načina uspostave kriptirane komunikacije;

- jednostavna uspostava komunikacije
- uspostava komunikacije prilikom autentifikacije poslužitelja
- uspostava komunikacije prilikom autentifikacije poslužitelja gdje su enkripcija i autentifikacija odvojeni
- uspostava komunikacije prilikom obostrane autentifikacije
- uspostava komunikacije korištenjem prethodne sjednice

U ovom poglavlju će biti detaljnije objašnjen rad SSL protokola kroz njegove slojeve. SSL protokol je podijeljen na dva sloja: *Handshake* sloj i *Record* sloj. U sljedećim potpoglavljima će biti detaljnije opisani. Slika *4.1* vizualno prikazuje SSL slojeve.



Slika 4.1: SSL slojevi i protokoli [26]

4.1 Handshake sloj

Handshake sloj je sloj zadužen za početni dogovor u SSL protokolu. Sačinjen je od 3 protokola: *Handshake*, *Change Cipher Spec* i *Alert* protokola koji će detaljno biti izloženi i objašnjeni u sljedećim potpoglavljima ovoga poglavlja.

4.1.1 **Handshake protokol**

Protokol rukovanja (*engl. handshake protocol*) služi za autentifikaciju i razmjenu ključeva kako bi se uspostavila sigurna sjednica. Poruke koje se koriste u *Handshake* protokolu su

HelloRequest, ClientHello, ServerHello, Certificate, ServerKeyExchange, CertificateRequest, ServerHelloDone, CertificateVerify, ClientKeyExchange i *Finished*.

4.1.2 **Change Cipher Spec protokol**

Za *ChangeCipherSpec* protokol moguće je zaključiti da je najjednostavniji protokol u SSL protokol specifikaciji. Razlog tomu je što se sastoji od jedne poruke. Riječ je o *ChangeCipherSpec* poruci. Ova poruka je detaljnije objašnjena u odjeljku *3.1.6*. Razlog zašto je *ChangeCipherSpec* zaseban protokol jest to što SSL protokol primjenjuje sigurnosne servise na sve poruke u sloju zapisa (*engl. record layer*) odjednom. Način rada sloja zapisa opisan je u potpoglavlju *4.2*. „*SSL primjenjuje sigurnosne servise kao što je enkripcija na sve Record Layer poruke odjednom. ChangeCipherSpec poruka kazuje da se od nje nadalje prelazi na sigurnosne postavke koje su dogovorne, pa ako bi ona bila sa ostalim porukama unutar jedne Record Layer poruke na ostale poruke koje ju slijede bilo bi nemoguće primijeniti sigurnosne servise. Zato je najjednostavnije rješenje ChangeCipherSpec definirati kao zaseban protokol.“* [27]

4.1.3 **Alert protokol**

Alert protokol služi kako bi se signalizirala greška ili propust prilikom komunikacije između klijenta i poslužitelja. Ovaj protokol sadrži dva parametra:

- *Severity level*
- *Alert description*.

Severity level parametar je indikator ozbiljnosti faktora koji je uzrokovao upozorenje. Postoje dvije razine upozorenja; upozorenja kojima je razina ozbiljnosti (*engl. severity level*) označena s 1 i upozorenja fatalnih grešaka kojima je razina ozbiljnosti označena s 2. Fatalne greške su vrlo ozbiljan problem te stoga obje strane, dakle klijent i poslužitelj, prilikom obavijesti na jednu od njih, prekidaju trenutnu sjednicu. Sama upozorenja mogu a i ne moraju biti razlog prekida sjednice. Strana koja dobije upozorenja odlučuje o tome. U parametru *Alert Description* nalazi se detaljniji opis svakog pojedinog upozorenja. Tablica *4.1* prikazuje nazive i kratke opise mogućih upozorenja.

Tablica 4.1: Popis upozorenja i opisi

close_notify	Pošiljatelj više neće slati poruke u toj sjednici
unexpected_message	Primljena je neprimjerena poruka
decryption_failed	Greška prilikom dešifriranja
decompression_failure	Greška prilikom dekompresije
handshake_failure	Greška prilikom dogovora oko sigurnosnih parametara
bad_certificate	Certifikat nije ispravan
unsupported_certificate	Certifikat tog tipa nije podržan
certificate_expired	Certifikat je istekao i zato nije ispravan
certificate_revoked	Certifikat je opozvan sa strane njegovog izdavatelja
protocol_version	Verzija protokola je prepoznata, ali nije podržana

4.2 Record sloj

SSL poruke moguće je dobiti iz četiri izvora:

- *ChangeCipherSpec* protokol
- *Handshake* protokol
- *Alert* protokol
- Aplikacija

Sve poruke koje su dobivene iz ova četiri izvora prihvaćene su od strane sloja zapisa koji ih formatira i sprema u odgovarajuće okvire te ih dalje prosljeđuje protokolu koji se nalazi na transportnom sloju (*engl. transport layer*) kao što je TCP (*engl. Transmission Control Protocol*). Sloj zapisa se bavi stvarnim podacima tako da dohvaća podatke iz aplikacijskog sloja (*engl. application layer*) te ih zatim kriptira i fragmentira. Potom se tim podacima određuje odgovarajuća veličina putem zadanog algoritma te ih nakon navedenih postupaka sloj zapisa šalje u transportni sloj. S obzirom na to jesu li poslani ili primljeni, sloj zapisa ima mogućnost sažimanja (*engl. compress*) ili dekomprimiranja (*engl. decompress*).

5. VIZUALIZACIJA SSL PROTOKOLA

Ovo poglavlje će prikazati kako implementirati korištenje SSL protokola na ASP.NET MVC web aplikaciji koja je postavljena na lokalnom IIS poslužitelju. Osim SSL autentifikacije poslužitelja bit će prikazana i autentifikacija klijenta. Finalni produkt će prikazivati kako samo određeni korisnici koji posjeduju određeni certifikat i njegov odgovarajući privatni ključ mogu pristupiti internetskoj stranici na kojoj se nalazi vizualizacija SSL protokola. Aplikacija će prikazivati puni potencijal SSL protokola; ne samo da će komunikacija između dva sustava biti kriptirana nego će za vrijeme nje oba sustava biti uvjerena u međusobne identitete.

5.1 Kreiranje i postavljanje certifikata

Iz poglavlja *3* moguće je zaključiti da certifikati imaju važnu ulogu u sigurnosti prilikom primjene SSL protokola. Za uspješnu autentifikaciju sustava potrebni su ispravni certifikati. Kompanije kupuju certifikate od pouzdanog trećeg tijela za izdavanje certifikata kao što su *GoDaddy* ili *VeriSign*. U ovom slučaju nema potrebe za plaćanjem. Moguće je koristiti testne certifikate generirane pomoću *Makecert.exe* alata koji je besplatan. Za početak potrebno je kreirati korijenski, CA certifikat koji će se koristiti za potpisivanje ostalih certifikata. Nadalje, potrebno je kreirati certifikat koji će služiti za autentifikaciju poslužitelja, u ovom slučaju IIS poslužitelja. Također je potrebno kreirati tri klijentska certifikata nad kojima će biti prikazani različiti oblici ishoda prilikom autentifikacije. Prilikom kreiranja svakog certifikata nastat će tri datoteke različitih ekstenzija;

- .cer – datoteka koja sadrži certifikat s javnim ključem
- .pvk – datoteka koja sadrži privatni ključ certifikata
- .pfx – datoteka koja sadrži i certifikat .cer i privatni ključ .pvk

5.1.1 *Kreiranje CA certifikata*

Nadolazeći skup naredbi potrebno je spremiti u datoteku s ekstenzijom .cmd kako bi se kasnije mogla koristiti. Naredbom *makecert.exe* se kreira certifikat, a zatim se koristi *pvk2pfx.exe* za kopiranje javnog i privatnog ključa iz .pvk i .cer u .pfx datoteku. U tablici *5.1* prikazani su korišteni parametri s kratkim opisima.

```
makecert.exe ^
-n "CN=CARoot" ^
-r ^
-pe ^
```

-a sha512 ^
-len 4096 ^
-cy authority ^
-sv CARoot.pvk ^
CARoot.cer

pvk2pfx.exe ^
-pvk CARoot.pvk ^
-spc CARoot.cer ^
-pfx CARoot.pfx ^
-po lozinka123

Tablica 5.1: Parametri pri kreiranju CA certifikata

-n "CN=CARoot"	Naziv certifikatovog subjekta, „CARoot“ u ovom slučaju
-r	Oznaka da je certifikat samopotpisan
-pe	Izgenerirani privatni ključ je izvediv i može se uključiti u certifikat
-a sha512	Oznaka za koji će se algoritam potpisa koristiti
-len 4096	Oznaka za generiranu duljinu ključa u bitovima
-cy authority	Oznaka da je CA certifikat
-sv CARoot.pvk	Datoteka privatnog ključa certifikata .pvk
CARoot.cer	Datoteka certifikata s javnim ključem
-pvk CARoot.pvk	Naziv .pvk datoteke
-spc CARoot.cer	Naziv .cer datoteke
-pfx CARoot.pfx	Naziv .pfx datoteke
-po lozinka123	Lozinka za .pfx datoteku

Prethodno spremljenu .cmd datoteku potrebno je pokrenuti u *Visual Studio Developer Command Prompt*-u. Pokreće se na način da se u *Visual Studio Developer Command Prompt* napiše naziv te datoteke (npr. *CreateCARoot.cmd*) i zatim pritisne tipka *Enter*. Nakon navedenog sustav bi trebao zatražiti unos lozinki. Prilikom unošenja lozinki stvara se i koristi .pvk privatni ključ, te se stoga lozinke moraju podudarati. Pri uspješnom izvršavanju navedenih koraka, dolazi do kreiranja tri datoteke; *CARoot.cer*, *CARoot.pvk* i *CARoot.pfx*. Prilikom kreiranja samopotpisanih certifikata, certifikatima se prema zadanim postavkama ne vjeruje. Stoga je potrebno dodati korijenski certifikat u trgovinu *Trusted Root Certification Authority*.

Potrebno je otvoriti *CARoot.cer* datoteku te odabrati opciju *Install Certificate*. Pri odabiru lokacije pohrane potrebno je odabrati *Local Machine*. *CARoot.cer* datoteka se u ovome trenutku nalazi u listi *Trusted Root Certification Authorities* i ukoliko se otvori *CARoot.cer* datoteka više

nema upozorenja da računalo ne vjeruje certifikatu. Svim budućim certifikatima potpisanim ovim certifikatom računalo će automatski vjerovati.

5.1.2 *Kreiranje poslužiteljevog certifikata*

Naredbe za kreiranje poslužiteljevog certifikata mogu se vidjeti u nadolazećem skupu naredbi. Također ih je potrebno spremiti u .cmd datoteku. Parametri navedeni u ovom odjeljku, a koji se razlikuju od parametara navedenih u odjeljku *5.1.1*, objašnjeni su u tablici *5.2*.

```
makecert.exe ^
-n "CN=yourdomain.com" ^
-iv CARoot.pvk ^
-ic CARoot.cer ^
-pe ^
-a sha512 ^
-len 4096 ^
-b 01/01/2020 ^
-e 01/01/2021 ^
-sky exchange ^
-eku 1.3.6.1.5.5.7.3.1 ^
-sv %1.pvk ^
%1.cer

pvk2pfx.exe ^
-pvk %1.pvk ^
-spc %1.cer ^
 -pfx %1.pfx ^
 -po lozinka123
```

Tablica 5.2: Parametri pri kreiranju poslužiteljskog certifikata

-n "CN=yourdomain.com"	Naziv domene na kojoj se nalazi web stranica
-iv CARoot.pvk	Datoteka privatnog ključa izdavatelja
-ic CARoot.cer	Datoteka certifikata s javnim ključem izdavatelja
-b 01/01/2020	Početak razdoblja u kojem je certifikat valjan
-e 01/01/2021	Kraj važećeg razdoblja certifikata
-sky exchange	Oznaka da je ključ za šifriranje i razmjenu ključeva
-eku 1.3.6.1.5.5.7.3.1	OID – utvrđuje da je ovo certifikat SSL poslužitelja
%1	Parametar naredbenog retka i bit će ono što unesete nakon .cmd, ovo će biti naziv datoteke .cer, .pvk i .pfx datoteka

Prethodno spremljenu .cmd datoteku također je potrebno pokrenuti u *Visual Studio Developer Command Prompt*-u. Ovoga puta osim naziva te datoteke potrebno je napisati i proizvoljan naziv

za certifikat (npr. *CreateSslServerCert. cmd ServerSSL*). Nakon navedenog sustav će tražiti lozinku za privatni ključ, upotrebu ključa za potvrdu, lozinku za privatni ključ CA certifikata (lozinka koja se generirala prilikom kreiranja CA certifikata) i na kraju lozinku privatnog ključa koju je potrebno odabrati u prvom prozoru koji se pojavljuje pri pokretanju. Nakon ovih koraka, ukoliko nije došlo do neočekivane greške, dolazi do kreiranja nove tri datoteke; *ServerSSL.cer, ServerSSL.pvk* i *ServerSSL.pfx*. Zatim je potrebno *ServerSSL.pfx* datoteku uvesti u *Personal Certificates*.

Potrebno je otvoriti datoteku *ServerSSL.pfx* te pri odabiru lokacije pohrane odabrati *Local Machine*. Ukoliko se otvori *ServerSSL.cer* datoteka, nema upozorenja od strane računala da se certifikatu ne vjeruje. Razlog tome jest to što je potpisan CA certifikatom kojem se vjeruje, čije je kreiranje objašnjeno u odjeljku *5.1.1*.

5.1.3 ***Kreiranje klijentskih certifikata***

Naredbe za kreiranje klijentskog certifikata slične su naredbama za kreiranje poslužiteljskog certifikata. Također ih je potrebno spremiti u *.cmd* datoteku. Parametri navedeni u ovom odjeljku, a koji se razlikuju od parametara navedenih u odjeljcima *5.1.1* i *5.1.2*, objašnjeni su u tablici *5.3*.

```
makecert.exe ^
-n "CN=%1" ^
-iv CARoot.pvk ^
-ic CARoot.cer ^
-pe ^
-a sha512 ^
-len 4096 ^
-b 01/01/2020 ^
-e 01/01/2021 ^
-sky exchange ^
-eku 1.3.6.1.5.5.7.3.2 ^
-sv %1.pvk ^
%1.cer

pvk2pfx.exe ^
-pvk %1.pvk ^
-spc %1.cer ^
-pfx %1.pfx ^
-po Test123
```

Tablica 5.3: Parametri pri kreiranju klijentskih certifikata

-n „CN=%1“	Proizvoljan naziv, bit će ono što se upiše nakon .cmd
-eku 1.3.6.1.5.5.7.3.2	OID – utvrđuje da je ovo certifikat SSL klijenta

Prilikom stvaranja primjera kreiranja klijentskih certifikata potrebno je napraviti identične korake kao prilikom kreiranja poslužiteljevog certifikata. Nakon što datoteke (npr. *tjansky.cer, tjansky.pvk, tjansky.pfx*) budu kreirane, potrebno je uvesti *tjansky.pfx* datoteku u *Personal Certificate Store* u *Current User* i zatim postaviti dodatnu lozinku na privatni ključ certifikata.

Zatim slijedi kreiranje još jednog certifikata, dakle tri datoteke, no različitog naziva od prethodnog; u ovome slučaju *lbaltoric.* Slijedi i kreiranje certifikata koji više nije validan jer mu je istekao datum. Njemu je dodijeljeno ime *mgrgic*. Ovim certifikatima računalo također vjeruje jer su potpisani korijenskim certifikatom iz odjeljka *5.1.1,* no ne i certifikatu *mgrgic* iz razloga što nema valjani datum uporabe.

5.2 Konfiguriranje IIS poslužitelja

5.2.1 *Osnovna konfiguracija IIS-a*

Ukoliko IIS poslužitelj nije instaliran na stroj koji će djelovati kao hosting poslužitelj, potrebno je to učiniti. Jedan od načina je da se u *windows* tražilicu upiše „Turn Windows features on or off“ te odabere. Potrebno je potvrditi „Internet Information Services“ i neke od njegovih podređenih čvorova kao što su „WEB Management Tools“, „World Wide Web Services“.

Provjera instalacije je moguća na način da se u web preglednik upiše „localhost“ te ukoliko je sve u redu postavljeno trebala bi se prikazati zadana internetska stranica IIS poslužitelja. Upravo tu zadanu internetsku stranicu potrebno je zamijeniti vlastitom web aplikacijom čije će kreiranje i detalji biti prikazani potpoglavlju *5.3*. To se radi na način da se desnim klikom na *Default Web Site* odabere opcija *Manage Website,* a zatim *Advanced Settings*. Potom je potrebno promijeniti *Physical Path* s putanjom do vlastite web aplikacije. Budući da je u ovom primjeru korišten lokalni IIS poslužitelj, potrebno je dodati web lokaciju u datoteku lokalnih domaćina (engl. *host*) kako bi se vlastita IP adresa lokalnog domaćina (engl. *localhost*) povezala s imenom domaćina. To se radi pokretanjem programa *Notepad* kao administratora i otvaranjem datoteke *hosts* na putanji *%systemroot%\System32\drivers\etc* i na njenom kraju dodavanjem „127.0.0.1 yourdomain.com“.

Nakon ovih koraka navigiranjem na „yourdomain.com“ u web pregledniku slijedi pozivanje vlastite web aplikacije.

5.2.2 *SSL konfiguracija na IIS-u*

Slijedi proces kojim se osigurava web aplikacija dodavanjem SSL certifikata na vlastiti poslužitelj.

U *IIS Manageru* potrebno je odabrati vlastiti poslužitelj te dvostrukim klikom odabrati *Server Certificates*. Potom je potrebno uvesti svoj samopotpisan certifikat poslužitelja kako bi se omogućila *https* komunikacija sa SSL protokolom. To se radi tako da se izabere *Import* opcija i odabere se poslužiteljski certifikat kreiran u odjeljku *5.1.1*.

Tada je potrebno dodati *https* vezanje (engl. *binding*) na način da se dvostrukim klikom odabere *Default Web Site* i zatim opcija *Bindings*. Slijedi pojavljivanje prozora u kojemu slijedi odabir *https* protokola sa *portom* 443. Potom se vrši odabir domene (npr: *yourdomain.com*) kao ime domaćina te je potrebno odabrati vlastiti samopotpisani certifikat.

Potrebno je IIS poslužitelju zadati da klijenta traži certifikat. To se radi dvostrukim klikom na *Default Web Site* i odabirom opcije *SSL Settings*. Slijedi klik na *Require SSL* i odabir opcije *Require*.

Nakon ovih koraka stranici je moguće pristupiti pisanjem putanje https://yourdomain.com u web preglednik.

5.3 Razvoj aplikativnog sloja

Ovo potpoglavlje će prikazati mogućnost i način reagiranja na klijentski certifikat u aplikativnom sloju. U ovom primjeru prikazan je oblik u kojemu samo klijenti čiji se naziv nalazi u listi stringova mogu pristupiti internetskoj stranici. Nakon uspješne autentifikacije klijenta, ova web aplikacija vizualizira korake prilikom raznih načina uspostave kriptirane komunikacije između klijenta i poslužitelja. Potrebno je imati instaliran *Visual Studio 2019*. U njemu je potrebno kreirati *ASP.Net MVC* web aplikaciju. Korišteni programski jezici i tehnologije su *C#, HTML* (*engl. Hyper Text Markup Language*), *CSS* (*engl. Cascading Style Sheet*), *Bootstrap, Java Script* i *jQuery*. Jedan od načina kako pristupiti klijentskom certifikatu s kojim se izvršila autentifikacija na aplikativnom sloju je prikazan u programskom kodu *5.1*. U svrhu ovog rada kreirana je metoda *IsClientAllowed()* koja provjerava ima li klijent pristup internetskoj stranici koristeći se testnim podacima unutar podatkovnog skupa *allowedClients*. Ova metoda prikazana je u programskom kodu *5.2*. Programski kod *5.3* se izvršava prilikom dohvaćanja web stranice nakon uspješne klijentske autentifikacije na razini IIS poslužitelja. Prilikom njegovog izvršavanja dohvaćaju se podaci o klijentskom certifikatu, događa se provjera klijentskog naziva te ovisno o tome korisniku je vraćena stranica ili upozorenje o grešci 401

odnosno zabranjen pristup. Vraćena stranica prikazuje vizualizaciju različitih načina uspostava kriptirane komunikacije. Vizualizacija je izrađena pomoću *HTML-a, JavaScript*-a, *jQuery*-a i *CSS*-a. Kompletan programski kod web aplikacije moguće je preuzeti na https://gitlab.com/Djuro105/sslapp lokaciji.

Programski kod 5.1: Metoda GetClientCertificate() u programskom jeziku C#

```
public HttpClientCertificate GetClientCertificate()
{
HttpClientCertificate clientCert;
clientCert = this.Request.ClientCertificate;

return clientCert;
}
```

Programski kod 5.2: Metoda IsClientAllowed() u programskom jeziku C#

```
public bool IsClientAllowed(HttpClientCertificate cert)
    {
        string[] allowedClients = { "tjansky", "gmarkovic", "žjurić" };
        string clientName = cert.Subject.Substring(3);  //CN=

        if (allowedClients.Contains(clientName))
        {
            return true;
        } else
        {
            return false;
        }
    }
```

Programski kod 5.3: Kontroler Index() u programskom jeziku C#

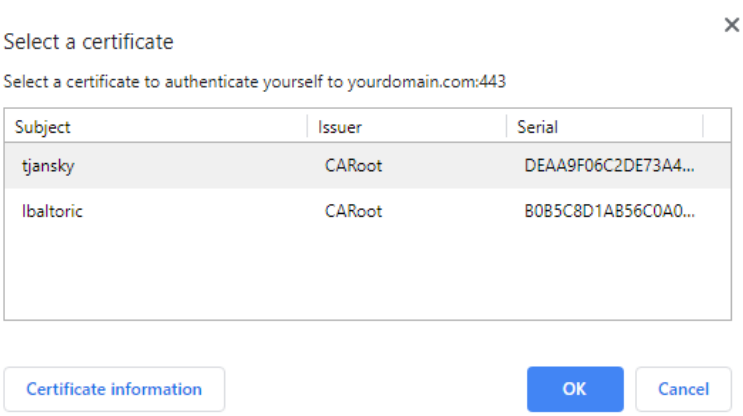
```
public ActionResult Index()
    {
        var cert = GetClientCertificate();

        if (IsClientAllowed(cert))
        {
            return View();
        } else
        {
            return new HttpStatusCodeResult(System.Net.HttpStatusCode.Unauthorized);
        }
    }
```

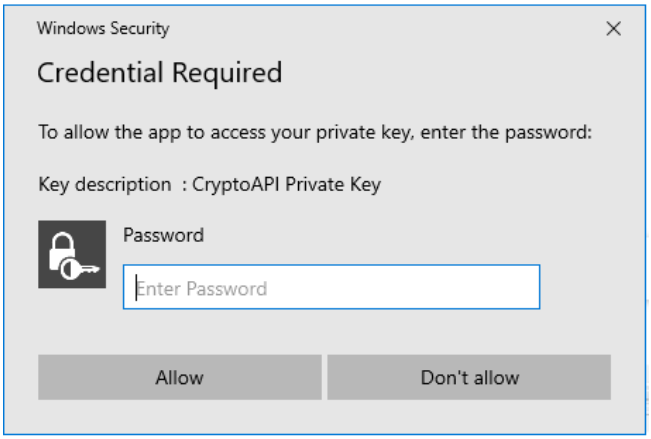
5.4 Prikaz rada aplikacije

Ovo potpoglavlje prikazuje rad finalnog produkta koji objedinjuje korištenje certifikata, IIS poslužitelja, web aplikacije te vizualizira korake prilikom uspostave kriptirane komunikacije. Za početak korisnik pristupa https://yourdomain.com adresi putem internetskog preglednika. Unosom navedene adrese u internetski preglednik korisnik traži stranicu na kojoj je prikazana vizualizacija. Zatim poslužitelj javlja klijentu, u ovom slučaju internetskom pregledniku, da je potrebna klijentska autentifikacija. Zatim se otvara prozor kojim internetski preglednik nudi korisniku popis dostupnih certifikata za klijentsku autentifikaciju. Korisnik bira kojim certifikatom će se provesti autentifikacija. Slika *5.1* prikazuje prozor koji sadržava popis certifikata s kojima je moguća klijentska autentifikacija te kojima IIS poslužitelj vjeruje. Sa slike *5.1* se može zaključiti da su certifikati s kojima je moguća klijentska autentifikacija *tjansky* i *lbaltoric*. Na računalu se nalazi treći certifikat pod nazivom *mgrgic*. Ovaj certifikat se ne nalazi na popisu jer IIS poslužitelj ga ne smatra ispravnim. Certifikat nije ispravan jer je datum valjanosti tog certifikata istekao. Kada korisnik odabere jedan od ispravnih certifikata s popisa pojavljuje mu se prozor za unos lozinke. Unosom ispravne lozinke korisnik dobiva pristup korištenja privatnog ključa klijentskog certifikata. Slika *5*.2 prikazuje prozor za unos lozinke privatnog ključa. Nakon unosa ispravne lozinke klijent je autentificiran od strane IIS poslužitelja te se izvršava provjera klijentskog certifikata na aplikativnom sloju. U ovome primjeru događa se provjera klijentskog naziva. Ako se pri autentifikaciji odluči za *lbaltoric* klijentski certifikat, korisnik neće biti propušten na stranicu. Ukoliko se odabere *tjansky* certifikat, on biva autentificiran od strane aplikativnog sloja te uspješno dohvaća stranicu. Na stranici se nalaze vizualno prikazane uspostave kriptiranih komunikacija. Korisnik ima mogućnost odabrati jedan

od četiri ponuđena načina uspostave kriptirane komunikacije. Ponuđeni načini uspostave kriptirane komunikacije su: Uspostava kriptirane komunikacije bez autentifikacije, uspostava kriptirane komunikacije s autentifikacijom poslužitelja, uspostava kriptirane komunikacije s autentifikacijom poslužitelja pri odvojenoj enkripciji od autentifikacije i uspostava kriptirane komunikacije s obostranom autentifikacijom. Nakon odabira pojavljuju se gumbi kojima je moguće upravljati animacijom vizualizacije protokola za odabrani način uspostave. Gumbi za upravljanje animacijom vizualizacije protokola prikazani su na slici *5.3*. Klikom na prvi gumb s lijeve strane slike *5.3*, animacija se pomiče s trenutnog na idući korak. Klikom na drugi gumb s lijeve strane, animacija se pomiče s trenutnog koraka na prethodni. Treći gumb s lijeve strane pokreće animaciju, dok posljednji gumb zaustavlja animaciju. Animacije vizualiziraju sve ranije objašnjene načine uspostave kriptirane komunikacije. Prvi korak, odnosno prva proslijeđena SSL poruka animacije uspostave kriptirane komunikacije prikazana je na slici *5.4*. Animacije ostalih načina uspostave kriptirane komunikacije prikazane su na jednak način.

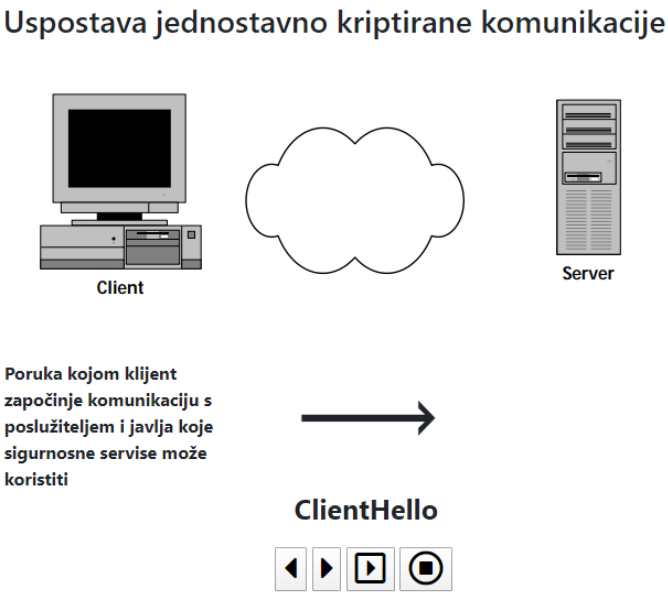


Slika 5.1: Popis klijentskih certifikata



Slika 5.2: Unos lozinke privatnog ključa

Slika 5.3: Gumbi za upravljanje animacijom



Slika 5.4: Animacija uspostave kriptirane komunikacije

6. ZAKLJUČAK

Završni rad na temu Vizualizacija SSL protokola prikazao je funkcioniranje SSL protokola kroz različite načine uspostava kriptirane komunikacije postignute izmjenjivanjem SSL poruka između klijenta i poslužitelja. Prikazana je važnost autentifikacije poslužitelja te i u nekim slučajevima i važnost klijentove autentifikacije. Internetska trgovina i razmjena bilo kakvih tajnih informacija putem interneta bila bi u puno manjoj mjeri moguća ako bi je uopće bilo kad se ona ne bi vršila putem ovakvog oblika kriptirane, dakle sigurne komunikacije. Dozu sigurnosti potrebnu za sigurnu razmjenu podataka između klijenta i poslužitelja donosi upravo SSL protokol.

U radu je prikazana važnost certifikata te jedan od načina kreiranja samopotpisanih certifikata u obliku korijenskog certifikata koji je korišten u svrhu potpisivanja svih ostalih certifikata, poslužiteljskog certifikata s kojim se vrši autentifikacija IIS poslužitelja i nekoliko klijentskih certifikata pomoću kojih je klijentska strana pokušala dokazati svoj identitet poslužiteljskoj strani. Dva od tri korištena samopotpisana certifikata su u radu uspješno prikazala svoj identitet poslužitelju, dok jedan certifikat nije zadovoljio uvjete autentifikacije iz razloga što mu je istekao rok valjanosti uporabe. Jedan od dva certifikata koji su uspješno dokazali svoj identitet IIS poslužitelju nije u potpunosti zadovoljio uvjete autentifikacije iz razloga što je pri provjeri na aplikativnom sloju dokučeno kako njegov identitet zapravo nije vjerodostojan. Ovim je putem dokazana kompleksnost protokola vezanih uz sigurnosnu komunikaciju između klijenta i poslužitelja što dokazuje važnost SSL protokola u kriptiranoj komunikaciji u obliku vizualnog prikaza navedene komunikacije.

U stvarnoj primjeni korištenja SSL protokola za ostvarenje sigurne komunikacije između klijentske strane i poslužiteljske strane ne bi se koristili primjerice samopotpisani certifikati naziva korištenih radi primjera u ovome radu, već bi se koristili certifikati izdani od ovlaštene izdavateljske kuće za certifikate kojoj se vjeruje na nacionalnoj ili međunarodnoj razini.

7. LITERATURA

[1] Stephen A. T. SSL & TLS Essentials, Securing the Web: 1. Kanada: Wiley Computer Publishing; 2000.

[2] Opplinger R. SSL and TLS, Theory and Practice: 3. Švicarska; 2009.

[3] Julie Olenski. SSL vs TLS – What's the Difference? [Online]. 2020. Dostupno na: https://www.globalsign.com/en/blog/ssl-vs-tls-difference (7.10.2020.)

[4, 5, 6] Stephen A. T. SSL & TLS Essentials, Securing the Web: 1. Kanada: Wiley Computer Publishing; 2000.

[7] Stephen A. T. SSL & TLS Essentials, Securing the Web: 3. Kanada: Wiley Computer Publishing; 2000.

[8, 9, 10, 11, 12, 13, 14, 15, 16, 17] Pačar B. Slabosti protokola SSL/TLS na napad čovjekom u sredini: 2. Diplomski rad. Zagreb: Fakultet elektrotehnike i računarstva; 2008.

[18] Stephen A. T. SSL & TLS Essentials, Securing the Web: 3. Kanada: Wiley Computer Publishing; 2000.

[19] Vidić D. Sigurnost računala i podataka: Digitalni certifikati, PKI, pdf. 2020.

[20, 21] Stephen A. T. SSL & TLS Essentials, Securing the Web: 3. Kanada: Wiley Computer Publishing; 2000.

[22] Sectigo. Personal Authentication Certificate: What Is a 2 Way SSL Certificate? [Online]. Dostupno na: https://sectigostore.com/page/2-way-ssl-certificate/ (7.10.2020.)

[23] Stephen A. T. SSL & TLS Essentials, Securing the Web: 3. Kanada: Wiley Computer Publishing; 2000.

[24] Pačar B. Slabosti protokola SSL/TLS na napad čovjekom u sredini: 2. Diplomski rad. Zagreb: Fakultet elektrotehnike i računarstva; 2008.

[25] Stephen A. T. SSL & TLS Essentials, Securing the Web: 3. Kanada: Wiley Computer Publishing; 2000.

[26] Vince T. SSL/TLS Overview [Online]. Dostupno na: https://sites.google.com/site/tlsssloverview/ssl-tls-protocol-layers (7.10.2020.)

[27] Pačar B. Slabosti protokola SSL/TLS na napad čovjekom u sredini: 3. Diplomski rad. Zagreb: Fakultet elektrotehnike i računarstva; 2008.

8. OZNAKE I KRATICE

SSL - Secure Socket Layer – Sigurnosni transportni protokol

CSS - Cascading Style Sheet – Stilski predlošci

TLS - Transport Layer Security – Transportni protokol

IIS - Internet Information Services – Internetski informacijski servisi

IP - Internet Protocol – Internetski protokol

HTML - Hyper Text Markup Language – Prezentacijski jezik za izradu web stranica

HTTP - Hyper Text Transfer Protocol – Protokol za prijenos hiperteksta

HTTPS - Hyper Text Transfer Protocol Secure – Protokol za osigurani prijenos hiperteksta

TCP - Transmission Control Protocol – Protokol za kontrolu prijenosa

NNTP - Network News Transfer Protocol – Protokol prijenosa internetskih novosti

FTP - File Transfer Protocol – Protokol za prijenos datoteka

9. POPIS TABLICA

10. POPIS SLIKA

11. SAŽETAK

Naslov: Vizualizacija SSL protokola

Završni rad na temu Vizualizacija SSL protokola prikazuje različite oblike prikaza uspostave kriptirane komunikacije što dovodi do sigurne komunikacije između klijenta i poslužitelja. Rad je podijeljen na teorijski i praktični dio pri čemu je u teorijskom dijelu ukratko prikazana povijest SSL protokola i varijacije pristupa samoj tematici važnosti SSL protokola u kriptiranoj komunikaciji. Praktični dio rada obuhvaća istraživanje koje podrazumijeva stvaranje tri certifikata kojima se pokušava prikazati sigurnosna razina autentifikacije potrebna za ostvarenje komunikacije između klijentske strane i poslužiteljske strane. Isto tako, prikazani su i razlozi zašto autentifikacija dva od tri certifikata nije uspješno izvršena. Rad prikazuje vizualizaciju navedene problematike.

Ključne riječi: SSL protokol, SSL poruke, certifikati, poslužitelj, kriptirana komunikacija, vizualizacija

12. ABSTRACT

Title: Visualization of SSL protocol

The final paper on the topic Visualization of SSL protocol shows different forms of display of the establishment of encrypted communication, which leads to secure communication between the client and server. The paper is divided into theoretical and practical part, where the theoretical part briefly presents the history of SSL protocols and variations of the approach to the topic of the importance of SSL protocols in encrypted communication. The practical part of the paper includes research that involves the creation of three certificates that try to show the security level of authentication required to achieve communication between the client side and the server side. Also, the reasons why the authentication of two of the three certificates was not performed successfully are presented. The paper presents a visualization of this issue.

Keywords: SSL protocol, SSL messages, certificates, server, encrypted communication, visualization.

IZJAVA O AUTORSTVU ZAVRŠNOG RADA

Pod punom odgovornošću izjavljujem da sam ovaj rad izradio/la samostalno, poštujući načela akademske čestitosti, pravila struke te pravila i norme standardnog hrvatskog jezika. Rad je moje autorsko djelo i svi su preuzeti citati i parafraze u njemu primjereno označeni.

Mjesto i datum	Ime i prezime studenta/ice	Potpis studenta/ice
U Bjelovaru, 27.10.2020	TOMISLAV JANSKY	Tomislav Jansky

Prema Odluci Veleučilišta u Bjelovaru, a u skladu sa Zakonom o znanstvenoj djelatnosti i visokom obrazovanju, elektroničke inačice završnih radova studenata Veleučilišta u Bjelovaru bit će pohranjene i javno dostupne u internetskoj bazi Nacionalne i sveučilišne knjižnice u Zagrebu. Ukoliko ste suglasni da tekst Vašeg završnog rada u cijelosti bude javno objavljen, molimo Vas da to potvrdite potpisom.

Suglasnost za objavljivanje elektroničke inačice završnog rada u javno dostupnom nacionalnom repozitoriju

TOMISLAV JANSKY

ime i prezime studenta/ice

Dajem suglasnost da se radi promicanja otvorenog i slobodnog pristupa znanju i informacijama cjeloviti tekst mojeg završnog rada pohrani u repozitorij Nacionalne i sveučilišne knjižnice u Zagrebu i time učini javno dostupnim.

Svojim potpisom potvrđujem istovjetnost tiskane i elektroničke inačice završnog rada.

U Bjelovaru, 27.10.2020

Tomislav Jansky

potpis studenta/ice