

# Sadržaj

<b>1</b>	<b>Uvod.....</b>	<b>1</b>
<b>2</b>	<b>Digitalni potpisi .....</b>	<b>2</b>
2.1	Asimetrična kriptografija .....	2
2.1.1	RSA potpisivanje .....	3
2.2	PKI .....	5
2.3	X.509.....	6
2.3.1	Opoziv certifikata.....	7
<b>3</b>	<b>eIDAS uredba .....</b>	<b>9</b>
3.1	Kvalificirani elektronski potpisi .....	11
<b>4</b>	<b>AdES digitalni potpisi .....</b>	<b>13</b>
4.1	AdES razine.....	14
4.2	XAdES.....	15
4.2.1	Pohrana potpisa .....	18
4.3	PADES .....	19
4.3.1	Sintaksa .....	21
4.4	CAdES.....	22
4.4.1	Sintaksa .....	23
4.5	Kriptografski algoritmi .....	25
4.5.1	Prihvaćeni algoritmi .....	25
<b>5</b>	<b>DSS.....</b>	<b>28</b>
5.1	Apstrakcija podataka .....	29
5.2	Potpisivanje .....	30
5.3	Provjera valjanosti potpisa .....	35
5.3.1	Politika valjanosti.....	40
5.4	Upravljanje povjerenjem .....	40
5.5	Izvori informacija o opozivu .....	41
<b>6</b>	<b>Arhitektura sustava.....</b>	<b>44</b>
6.1	Web aplikacija.....	44

6.2	Servis za potpisivanje .....	48
6.3	Servis za upravljanje digitalnim potpisima .....	48
6.4	CRL poslužitelj.....	49
6.5	OCSP poslužitelj .....	49
6.6	Baza podataka .....	49
7	Implementacija sustava .....	51
7.1	Web Aplikacija.....	53
7.1.1	Pregled PDF datoteke.....	55
7.2	Servis za potpisivanje .....	57
7.3	Servis za upravljanje digitalnim potpisima .....	59
7.4	Testni PKI.....	62
7.5	CRL poslužitelj.....	67
7.6	OCSP poslužitelj .....	68
8	Zaključak .....	69
	Literatura .....	70
	Popis oznaka i kratica .....	72
	Sažetak.....	73
	Summary .....	74
	Prilozi.....	75
	Kazalo slika.....	75

# 1 Uvod

Trend digitalizacije i modernizacije nezaobilazna je pojava 21. st. Veliki broj usluga razvija se u obliku web aplikacija koje se, u sve većem broju, izvršavaju u računalnim oblacima. Uz već poznate benefite ove tranzicije, javlja se niz problema koje je potrebno otkloniti kako bi postojeće usluge iz fizičkog svijeta bilo moguće zamijeniti digitalnima.

Jedna takva usluga, koja još nije u potpunosti digitalizirana, je razmjena dokumenata i ostalih datoteka uz očuvanje sigurnosnih značajki. Sustavi koji pružaju ovakve usluge moraju implementirati sigurnosne elemente primjenjive u fizičkom svijetu, poput potvrde vlasništva, a nerijetko ih i proširuju s konceptima koji nisu mogući izvan digitalnih sustava

U ovom diplomskom radu opisana je usluga elektroničkog potpisivanja datoteka u skladu s uredbama Europske Unije.

U prvom poglavlju opisani su elementarni koncepti asimetrične kriptografije na kojima se temelji čitava infrastruktura elektronskih potpisa.

Iduće poglavlje sažima bitne značajke europske uredbe koje je potrebno ispuniti prilikom implementacije elektronskih potpisa.

Detaljniji pregled AdES razine digitalnih potpisa definiranih prethodno navedenom uredbom izložen je u trećem poglavlju.

U četvrtom poglavlju istražuje se DSS biblioteka čiji je cilj podržati interoperabilno i sigurno stvaranje te provjeru valjanosti elektronskih potpisa.

Zadnja dva poglavlja opisuju arhitekturu realiziranog sustava i implementacijske detalje bitne za razumijevanje sustava. Povrh toga, dan je pregled tehnologija i odluka prilikom dizajniranja sustava.

## 2 Digitalni potpisi

Digitalni potpis je matematički postupak kojim se utvrđuje autentičnost i integritet digitalne poruke odnosno dokumenta. Često se naziva i digitalni otisak prsta (engl. *fingerprint*) jer može jedinstveno predstavljati osobu ili bilo koji drugi entitet. Ispravno izrađen digitalni potpis može služiti kao dokaz o podrijetlu, vremenu, identitetu i statusu digitalnog dokumenta [1].

Nazivi digitalni potpis i elektronski potpis se često koriste proizvoljno i naizmjenično iako se konceptualno razlikuju. Elektronski potpis je bilo koji elektronski podatak koji je povezan s nekim drugim elektronskim podatkom s namjerom potpisivanja toga podatka. Elektronski potpis može biti zvuk, simbol, ilustracija i sl. Elektronske potpise odlikuje lakoća korištenja uz relativno nisku razinu sigurnosti. Nerijetko se koriste za potpisivanje ugovora putem web-a [2].

Digitalni potpisi su podskup elektronskih potpisa, odnosno specifična implementacija elektronskih potpisa. Kategorije elektronskih potpisa su brojne, a razlikuju se prema tehničkoj implementaciji, pravnoj prihvaćenosti, geografskom području, itd.

Mnoge industrije i geografske regije su uspostavile standarde za digitalne potpise. Ispravno praćenje i implementacija ovih standarda omogućava brojne organizacijske, sigurnosne i pravne benefite.

### 2.1 Asimetrična kriptografija

Digitalni potpisi temelje se na asimetričnoj kriptografiji (kriptografija javnog ključa) iz čega proizlaze njihova sigurnosna svojstva. Neki od najčešće korištenih algoritama su RSA, DSA, ECDSA, EdDSA.

Privatni ključ definira transformaciju potpisa  $S_A$ :

$$S_A(m) = \sigma \quad (1)$$

Javni ključ definira transformaciju verifikacije  $V_A$ :

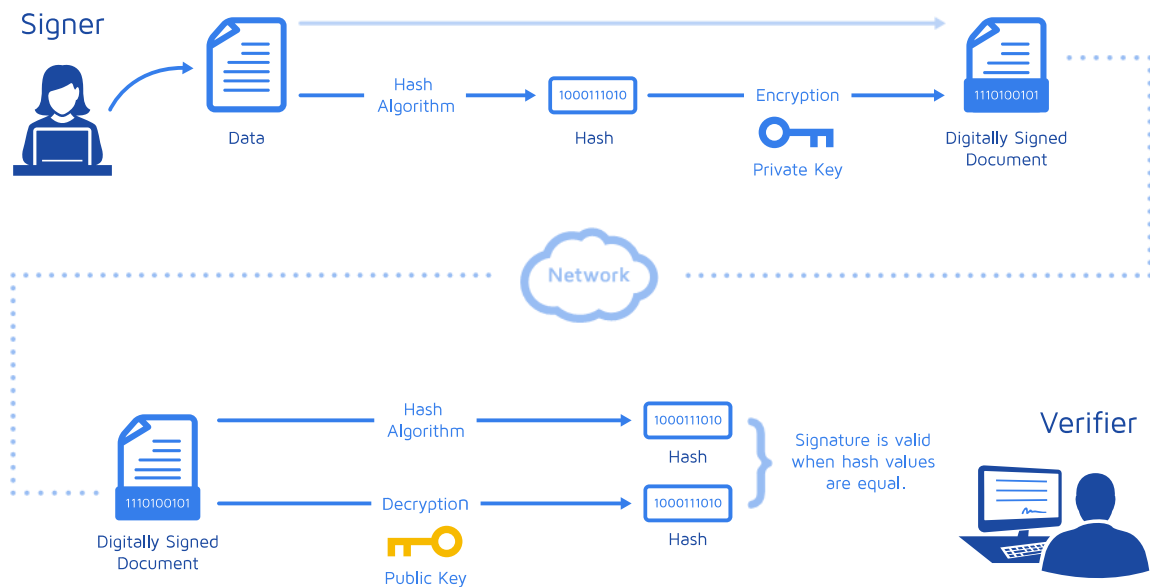
$$V_A(m, \sigma) = \omega \quad (2)$$

Verifikacija je uspješna ako i samo ako je ispunjeno [3]:

$$\omega = \sigma \quad (3)$$

Tehnologija digitalnih potpisa zasniva se na povjerenju da entitet koji je napravio potpis ima isključiv pristup pripadajućem privatnom ključu. U slučaju da je privatni ključ kompromitiran, moguće je stvaranje potpisa čiji autor nije vlasnik ključa.

Princip potpisivanja bazira se na tzv. „Hash and Sign“ paradigmi u kojoj se potpisuje hash vrijednost podatka umjesto samog podatka. Slika 2.1 ilustrira ovaj postupak.



Slika 2.1 Dijagram potpisa i verifikacije podataka (izvor: <https://www.docusign.com/how-it-works/electronic-signature/digital-signature/digital-signature-faq>, preuzeto 18.8.2021.)

### 2.1.1 RSA potpisivanje

Prethodno spomenutu „Hash and Sign“ paradigmu moguće je implementirati koristeći RSA algoritam. Matematički opis digitalnog potpisa koristeći RSA algoritam je izložen u nastavku.

Neka je zadan ulazni podatak  $M$ . U prvom koraku računa se hash vrijednost podatka  $M$  koristeći jednu od kriptografskih funkcija (npr. SHA256):

$$H(m) = h \quad (4)$$

Na dobivenu hash vrijednost primjeni se PKCS #1 formatiranje:

$$PKCS\#1(h) = h_f \quad (5)$$

Formatirana vrijednost se enkriptira koristeći prethodno generirani privatni ključ  $(e, n)$ :

$$\Sigma \equiv h_f^e \pmod{n} \quad (6)$$

Dobivena vrijednost  $\Sigma$  je potpis ulaznog podatka  $M$ .

Provjeru potpisa moguće je napraviti ako su poznati ulazni podatak  $M$  i potpis  $\Sigma$ . U prvom koraku potrebno je dohvatiti autentični javni ključ  $(d, e)$  te izračunati:

$$h'_f \equiv \Sigma^d \pmod{n} \quad (7)$$

$h'_f$  se obradi PKCS #1 formatiranjem. U ovom koraku moguće je odbaciti dobiveni rezultat u slučaju da  $h'_f$  nije dobro formatiran.

Izračuna se hash vrijednost ulaznog podatka:

$$H(m) = h \quad (8)$$

Digitalni potpis  $\Sigma$  može biti prihvaćen ako i samo ako vrijedi:

$$h'_f = h \quad (9)$$

U suprotnom, potpis  $\Sigma$  treba biti odbačen jer javni ključ ne odgovara privatnom ključu kojim je izračunat potpis ili ulazni podatak ne odgovara originalnom ulaznom podatku.

## 2.2 PKI

Sustav digitalnog potpisivanja mora omogućiti provjeru vlasništva javnog ključa. Drugim riječima, potrebno je odgovoriti na pitanje kako znati da korišteni javni ključ entiteta pripada upravo tom entitetu.

Infrastruktura javnog ključa (PKI) omogućava tu funkcionalnost pri čemu asocijaciju između javnog ključa i entiteta potvrđuje operater PKI-a tj. certifikacijsko tijelo (engl. *Certificate authority*).

PKI se sastoji od politika, standarda, ljudi i sustava koji podržavaju distribuciju javnih ključeva i provjeru identiteta pojedinaca ili drugih subjekata s digitalnim certifikatima i tijelom za izdavanje certifikata [4].

PKI se sastoji od [5]:

- CA – pohranjuje, izdaje i potpisuje digitalne certifikate
- RA – provjerava identitet subjekata koji zahtijevaju pohranu vlastitog digitalnog certifikata od CA
- Centralni direktorija – sigurna lokacija u kojoj su certifikati (zajedno s javnim ključevima) pohranjeni
- Sustav upravljanja certifikatima

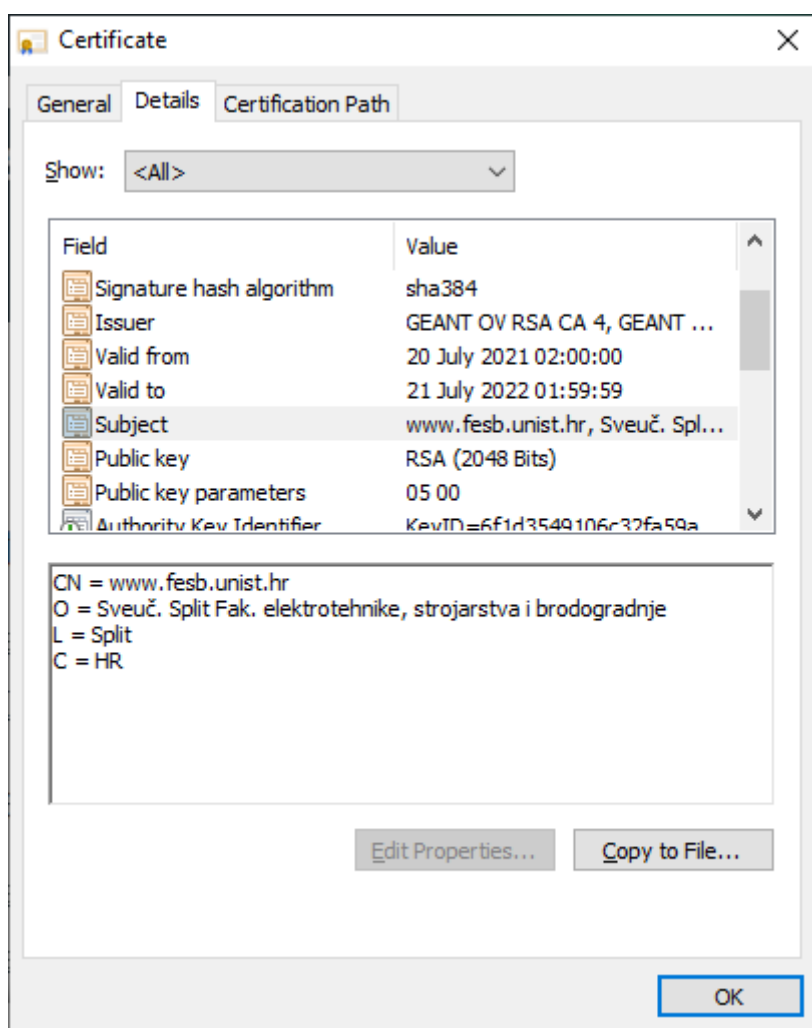
Postoje sigurnosna ograničenja vezana uz PKI. Glavna slabost javnog PKI je što bilo koji CA može potpisati certifikat za bilo koji entitet. CA postoje u brojnim državama, među kojima su i one s autoritarnim i potencijalno neprijateljskim režimima. Nepovjerljivi CA mogu izdati certifikate s namjerom špijunaže, krivotvorenja poruka ili dokaza te brojnih drugih prevara. Zbog toga, povjerenje u javne CA i certifikate koje su izdali mora biti ograničeno [6].

Neke organizacije stvaraju vlastite PKI. To znači da imaju kontrolu nad CA. Kada organizacija daje povjerenje isključivo internom CA za neku svrhu, šansa za prijevarno izdavanje certifikata može biti smanjena u odnosu na javni CA pod uvjetom da je interni PKI implementiran ispravno.

## 2.3 X.509

X.509 je standard koji definira format digitalnih certifikata. Bez obzira na primjenu, svaki X.509 certifikat sadrži javni ključ, digitalni potpis, informacije o subjektu vezanog uz taj certifikat i tijelu koje ga je izdalo. Pored tih informacija, može sadržavati i informacije o verziji i periodu u kojem je valjan, a verzija 3 definira skup proširenja koji definiraju dodatne informacije o certifikatu [7].

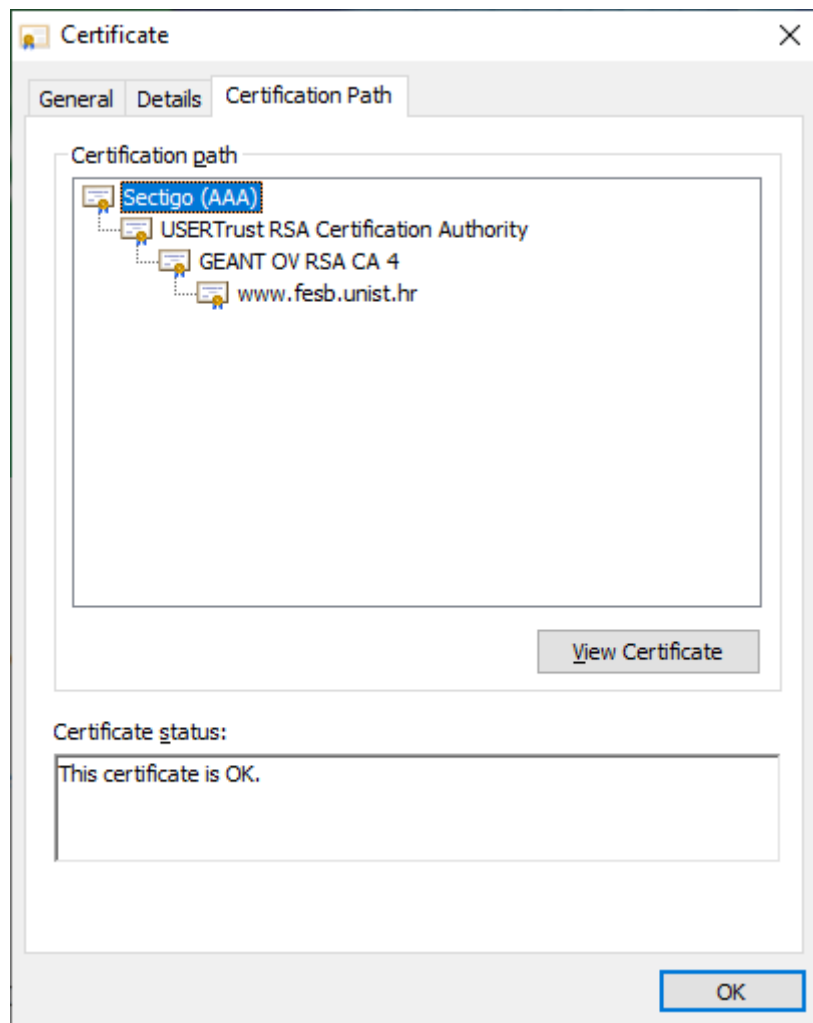
Slika 2.1 prikazuje digitalni certifikat web stranice FESB-a dohvaćen preglednikom Google Chrome.



Slika 2.2 Digitalni certifikat web stranice FESB-a

Zbog administrativnih i sigurnosnih razloga, X.509 certifikate se obično povezuju u lance. Slika 2.3 prikazuje takav lanac certifikata na primjeru FESB-ovog certifikata.





Slika 2.3 Lanac certifikata

Web poslužitelji obično šalju svoj krajnji certifikat i posredne certifikate koji su potrebni za provjeru lanca. Certifikat na vrhu lanca i njegov pripadajući javni ključ (Sectigo (AAA) u ovom slučaju) su obično pohranjeni u operacijski sustav ili preglednik korisnika čime se uspostavlja lanac povjerenja.

### 2.3.1 Opoziv certifikata

X.509 certifikate moguće je opozvati prije formalnog isteka njihovog trajanja. RFC 5280 definira liste opoziva certifikata (engl. *Certificate Revocation List*). CRL je potpisana lista opozvanih certifikata s vremenskim žigom koju CA obično obnavlja u regularnim intervalima. Klijentski softver može postavljati upite na ovu listu kako bi utvrdio valjanost digitalnog certifikata.

U praksi su se CRL pokazale nepraktične zbog nemogućnosti skaliranja. RFC 2560 definira OCSP koji daje mogućnost provjere certifikata bez pretraživanja liste opoziva. CA obično održava jednostavni HTTP poslužitelj koji odgovara na OCSP zahtjeve. Klijentski softver šalje HTTP zahtjev OCSP poslužitelju koji vraća potpisani odgovor o valjanosti certifikata.

RFC 6066 definira proširenje OCSP-a pod nazivom „OCSP Stapling“ koje rješava potencijalne probleme s performansama i sigurnosti OCSP-a.

Kod ove tehnike web poslužitelji u ime klijenata dohvaćaju OCSP odgovore za vlastiti certifikat te ga obično pohrane u privremenu memoriju i do 7 dana. Poslužitelj tada može poslati OCSP odgovor prilikom uspostave TLS veze (engl. *TLS handshake*) čime se smanjuje broj HTTP zahtjeva koje klijent mora poslati [8].

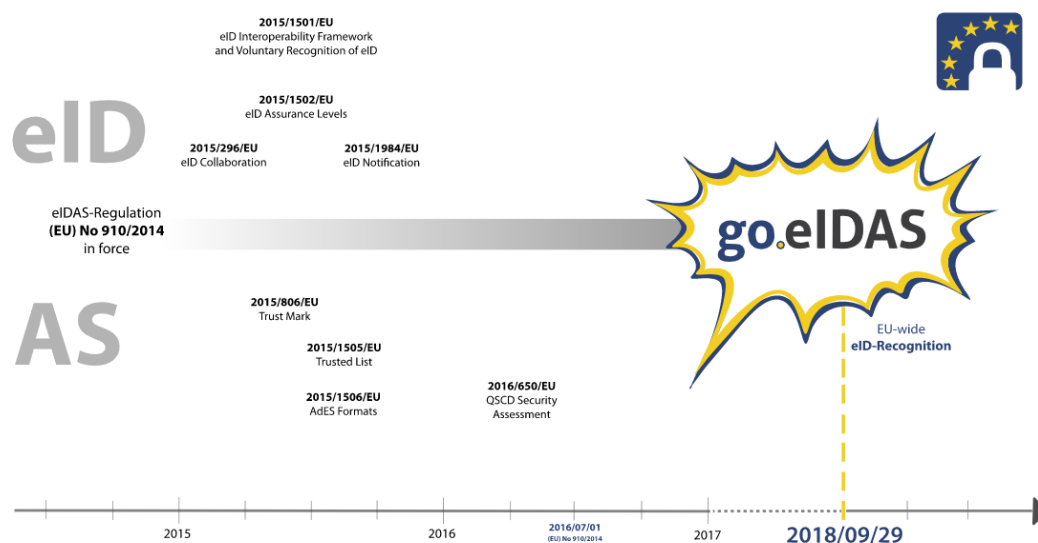
### 3 eIDAS uredba

Uredba (EU) br. 910/2014 Europskog parlamenta i vijeća o elektroničkoj identifikaciji i uslugama povjerenja za elektroničke transakcije na unutarnjem tržištu iznesena je 23. srpnja 2014. godine [9].

eIDAS igra ključnu ulogu u omogućavanju sigurnih prekograničnih transakcija. To je svojevrsna prekretnica u stvaranju predvidivog regulatornog okvira na području EU. Cilj je pomoći tvrtkama, privatnim osobama i javnim ustanovama u izvršavanju elektronskih radnji na jednostavan i siguran način.

S ovom uredbom, EU je postavila temelje i pravni okvir za sve sudionike kako bi na siguran način i uz manje prepreka pristupali elektroničkim uslugama. U tom smislu, eIDAS omogućuje usluge poput prijave poreza, upisa na sveučilišta diljem Europe, uspostavljanja tvrtke u zemljama članicama, itd.

Uredba je postala potpuno primjenjiva gotovo dvije godine nakon uvođenja, a od 29. rujna 2018. sve organizacije koje pružaju digitalne usluge unutar zemlje članice EU moraju priznati elektronsku identifikaciju definiranu ovom uredbom [10]. Slika 3.1 prikazuje značajne događaje u razvoju eIDAS uredbe.

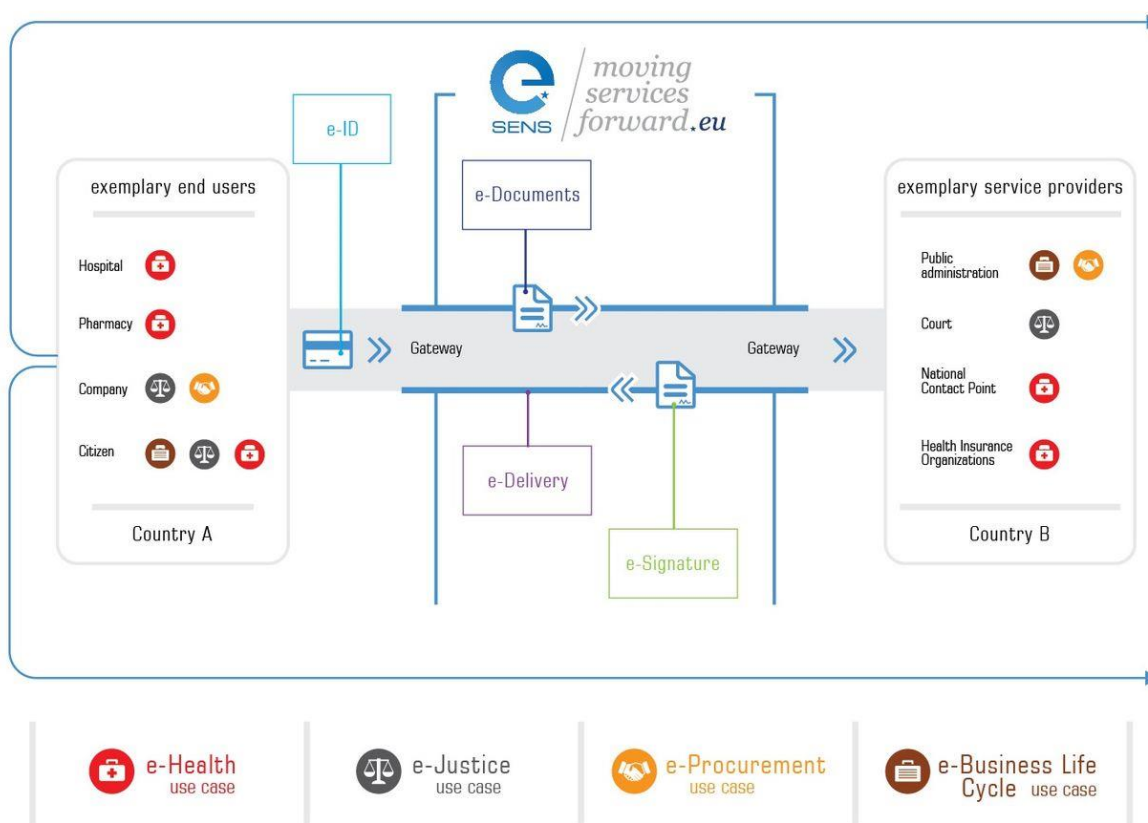


Slika 3.1 Vremenska traka razvoja eIDAS uredbe (izvor: <https://www.eid.as/>, preuzeto 17.8.2021.)

Iste godine kada je priznavanje elektronske identifikacije postalo obavezno na razini EU, vodeći Europski stručnjaci, organizacije i udruženja u području elektronske identifikacije su pokrenuli neprofitnu inicijativu go.eIDAS.

Cilj organizacije je podizanje svijesti kroz stvaranje lokaliziranog informativnog materijala, implementaciju i održavanje softvera te brojne druge aktivnosti kojima se promoviraju benefiti uredbe.

Slika 3.2 ilustrira ideju uredbe o jedinstvenom EU tržištu i olakšanom prekograničnom pristupu javnim uslugama.



Slika 3.2 Jedinstveno elektroničko EU tržište (izvor: [https://en.wikipedia.org/wiki/EIDAS#/media/File:E-SENS\\_architecture.jpg](https://en.wikipedia.org/wiki/EIDAS#/media/File:E-SENS_architecture.jpg), preuzeto 17.8.2021.)

eIDAS definira okvir za sljedeće aspekte elektronskih transakcije:

- Napredni elektronski potpisi (engl. *Advanced electronic signature*)
- Kvalificirani elektronski potpisi (engl. *Qualified electronic signature*)
- Kvalificirani digitalni certifikati (engl. *Qualified digital certificate*)

- Kvalificirani autentifikacijski certifikati za web stranice (engl. *Qualified website authentication certificate*)
- Usluge povjerenja (engl. *Trust service*) koje uključuju stvaranje i provjeru valjanosti elektronskih pečata i potpisa, vremenskih oznaka (engl. *Timestamps*) i certifikata)

U narednom poglavlju detaljnije će biti obrađeni napredni elektronski potpisi koji su temelj implementiranog sustava za digitalno potpisivanje dokumenata.

U sklopu izrade sustava za digitalno potpisivanje dokumenata bitno je uzeti u obzir ovu regulativu jer su njome definirani brojni pravni učinci koje je moguće ostvariti uz poštivanje implementacijskih smjernica.

Prema čl. 46 ove uredbe, elektronički dokumenti se mogu koristiti kao dokaz u sudskim postupcima te im se ne smije uskratiti pravni učinak isključivo zato jer su u elektroničkom obliku.

### **3.1 Kvalificirani elektronski potpisi**

Kvalificirani elektronski potpis je napredni elektronski potpis izrađen uz pomoć uređaja za izradu kvalificiranih elektronskih potpisa (engl. *Qualified signature creation device*) koristeći kvalificirani digitalni certifikat.

Kvalificirani digitalni certifikat mora biti izdan od strane kvalificiranog izdavatelja usluga povjerenja (engl. *Trust service provider*). Time se osigurava autentičnost elektronskog potpisa te on može služiti kao dokaz identiteta. U Republici Hrvatskoj postoje tri aktivna TSP-a: AKD d.o.o., FINA i Zagrebačka banka d.d. [11].

QES omogućava provjeru autorstva u elektronskoj razmjeni podataka u dugom roku te je prema čl. 25 uredbe ekvivalentan rukom napisanom potpisu.

QSCD je podvrsta uređaja za izradu sigurnih potpisa (engl. *Secure signature creation device*) koji ispunjavaju uvjete navedene u eIDAS uredbi. Potpunu listu uvjeta koje ovi uređaji moraju ispuniti moguće je pronaći u prilogu 2 uredbe.

U sklopu ovoga rada neće se dalje razmatrati ova vrsta potpisa zbog nemogućnosti nabave kvalificiranog certifikata i QSCD-a u svrhu testiranja i prezentacije.

## 4 AdES digitalni potpisi

Napredni elektronski potpis (engl. *Advanced electronic signature*) je elektronski potpis koji odgovara zahtjevima postavljenim EU uredbom eIDAS. Napredni elektronski potpisi moraju ispuniti stroge uvjete definirane uredbom. AdES elektronski potpis mora [10]:

- jedinstveno i nedvosmisleno biti povezan s entitetom koji ga je potpisao
- identificirati entitet koji ga je potpisao
- biti potpisan kroz sredstva koja su u potpunoj kontroli entiteta koji ga potpisuje, drugim riječima privatni ključ ne smije biti kompromitiran
- omogućiti provjeru integriteta popratnih podataka nakon potpisivanja te biti poništen u slučaju modifikacije

Ovakvi potpisi mogu se implementirati kroz AdES osnovne profile (engl. *AdES baseline profiles*) koje razvija ETSI:

- XAdES – XML AdES je skup proširenja na XML-DSig preporuku definiran standardom ETSI TS 103171 v.2.1.1
- CAdES – CMS AdES je skup proširenja sintaksi kriptografske poruke (engl. *Cryptographic message syntax*) definiran standardom ETSI TS 103173 v.2.2.1
- PAdES – PDF AdES je skup ograničenja i proširenja na PDF i ISO 32000-1 definiran standardom ETSI TS 103172 v.2.2.2
- ASiC – povezani spremnici potpisa (engl. *Associated Signature Containers*) definiraju korištenje struktura kojima se povezuje jedan ili više potpisanih objekata s naprednim elektronskim potpisom ili vremenskim žigom u jedan digitalni spremnik najčešće zip formata; definiran standardom ETSI TS 103174 v.2.2.1
- JAdES – JSON AdES je skup proširenja na JWS definiran standardom ETSI TS 119182-1

Brojne opcije, postavke i verzije gore navedenih standarda otežavaju interoperabilnost zbog čega su novi, osnovni (engl. *baseline*) profili objavljeni. Njihov je cilj ograničiti broj opcija i varijanti te time poboljšati interoperabilnost između sudionika.

Tablica 1 daje prikaz mogućnosti potpisivanja različitih vrsta podataka gore navedenim profilima.

Tablica 1 Vrste potpisa i formati podataka

Profil potpisa			XML	JSON	PDF	Binarni podatak	Digest	Više datoteka	Više potpisa	Protupotpis
XAdES	Enveloping	Base64 encoded	✓	✓	✓	✓	✗	✓	✗	✓
		Embed XML	✓	✗	✗	✗	✗	samo XML	✗	✓
		Manifest	✓	✓	✓	✓	✓	✓	✗	✓
		Canonicalization	✓	✗	✗	✗	✗	samo XML	✗	✓
	Enveloped	enveloped transformation	✓	✗	✗	✗	✗	✗	✗	✓
		based on XPath	✓	✗	✗	✗	✗	✗	✓	✓
		based on Filter2	✓	✗	✗	✗	✗	✗	✓	✓
		Canonicalization	✓	✗	✗	✗	✗	samo XML	✓	✓
	Detached		✓	✓	✓	✓	✓	✓	✗	✓
	Internally Detached		✓	✗	✗	✗	✗	samo XML	✓	✓
CAdES	Enveloping		✓	✓	✓	✓	✗	✗	✓	✓
	Detached		✓	✓	✓	✓	✓	✗	✓	✓
PAdES	Enveloped		✗	✗	✓	✗	✗	✗	✓	✗

#### 4.1 AdES razine

Dokumenti i drugi podatci mogu biti u uporabi dugi niz godina. U tom vremenskom periodu certifikati mogu isteći, CA može izgubiti akreditaciju, korišteni algoritmi postati ranjivi, itd.

Novi ETSI standard definira četiri razine sukladnosti kojima se rješava problem valjanosti potpisa kroz vrijeme [12]:

- B – (engl. *Basic electronic signature*) – najniža i najjednostavnija izvedba koja sadrži podatke o potpisu, ključu i dodatnim svojstvima potpisa
- T – (engl. *Signature with a timestamp*) – vremenski žig povjerljivog izvora dodan prethodnoj razini u svrhu zaštite protiv odricanja od odgovornosti (engl. *repudiation*)
- LT – (engl. *Signature with Long Term data*) – certifikati i podatci o opozivu su ugrađeni u potpis čime se omogućuje provjera čak i ako originalni izvori nisu dostupni



- LTA – (engl. *Signature with Long Term Data and Archive timestamp*) – periodičkim dodavanjem vremenskog žiga sprječava se kompromitacija potpisa kao posljedica slabljenja sigurnosti korištenih algoritama

B razina sadrži nepromjenjive (engl. *immutable*) potpisane podatke. Jednom kada je potpis ove razine izrađen njegovi se podaci ne mogu mijenjati.

Razine T, LT i LTA dodaju nepotpisane podatke potpisu. To znači da se podaci ovih razina mogu dodati naknadno na bilo koji AdES potpis. Proširenje ovim razinama može učiniti potpis sigurnijim u dugom roku. Proširenja se rade inkrementalno, odnosno, kada se napravi proširenje na LT razinu, niža razina T je također dodana.

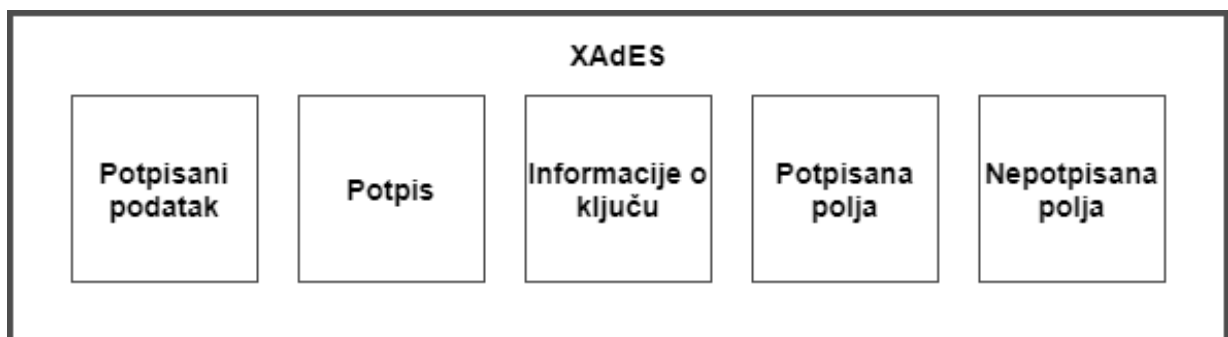
U praksi je moguće napraviti potpis direktno bilo kojom od ove četiri razine [12].

## 4.2 XAdES

XAdES je skup proširenja na XML-DSig preporuku koja implementiraju zahtjeve naprednih elektronskih potpisa. Organizacije W3C i ETSI rade na održavanju i ažuriranju [13].

XAdES je čitljiv čovjeku i računalima što ga čini pogodnim za brojne uporabe, a zbog svojih svojstava njime je moguće potpisati čitav niz različitih vrsta podataka.

Slika 4.1 prikazuje pojednostavljeni dijagram XAdES potpisa koji ilustrira podatke koji se dodaju originalnoj XML-DSig preporuci.



Slika 4.1 XAdES dijagram

XAdES podaci se dodaju u `ds:Object` element XML-DSig strukture. Potpisana polja se nalaze pod elementom `SignedProperties`, dok se nepotpisana nalaze pod elementom `UnsignedProperties`.

Slika 4.2 prikazuje XML strukturu XAdES-Baseline-B.



Obavezni podatci u polju `SignedProperties` su:

- jednoznačna referenca na certifikat autora potpisa - `SigningCertificate`
- jednoznačan način identifikacije politike potpisa pod kojom je potpis izrađen kako bi prilikom verifikacije bilo moguće ostvariti iste uvjete - `SignaturePolicyIdentifier`
- vrijeme potpisa (u ovom slučaju to nije nužno povjerljivi vremenski žig) - `SigningTime`

Potpis također može pokriti i ostala polja koja sadržavaju sljedeće informacije:

- format jednog ili više podataka koji su potpisani - `DataObjectFormat`
- vrsta opredjeljenja (engl. *commitment*) autora potpisa prilikom potpisivanja podataka u kontekstu odabrane politike potpisa; politika potpisa može imati višestruku pravnu interpretaciju - `CommitmentTypeIndication`
- tvrđena ili ovjerena uloga koju autor potpisa preuzima pri izradi potpisa - `SignerRole`
- navodno mjesto na kojem autor potpisa tvrdi da je izradio potpis - `SignatureProductionPlace`

Autor potpisa ili entitet koji obavlja provjeru može izraditi XAdES-Baseline-T dodavanjem djeteta elementu `UnsignedProperties`. Povjerljivi vremenski žig `SignatureTimeStamp` dokazuje da je elektronski potpis postojao prije naznačenog vremena.

Ostale razine na sličan način dodaju potrebne podatke kao djecu nepotpisanog XML elementa `UnsignedProperties`. To mogu biti lanci certifikata, liste opoziva, OCSP odgovori i sl.

#### 4.2.1 Pohrana potpisa

Potpis mora biti povezan s podatkom na koji se odnosi. To je moguće napraviti stvaranjem novog podatka koji obuhvaća potpis i originalni podatak ili pohranom potpisa u vanjski resurs te naknadnim povezivanjem podatka s potpisom.

Za XML datoteke moguće je izraditi omotnicu koja sadržava izvornu XML strukturu i potpis dok za binarne podatke to nije trivijalno napraviti.

XAdES definira načine ugrađivanja potpisa kroz sljedeće načine rada:

- Omotan (engl. *Enveloped*) – potpis se odnosi na podatak koji ga omeđuje
- Obavijajući (engl. *Enveloping*) – podatak koji se potpisuje je pod element samog potpisa
- Odvojen (engl. *Detached*) – potpis se odnosi na podatak od kojeg je odvojen
- Interno odvojen (engl. *Internally-detached*) – primjenjiv samo za XML podatke, potpis i podatak koji je potpisan su unutar elementa roditelja

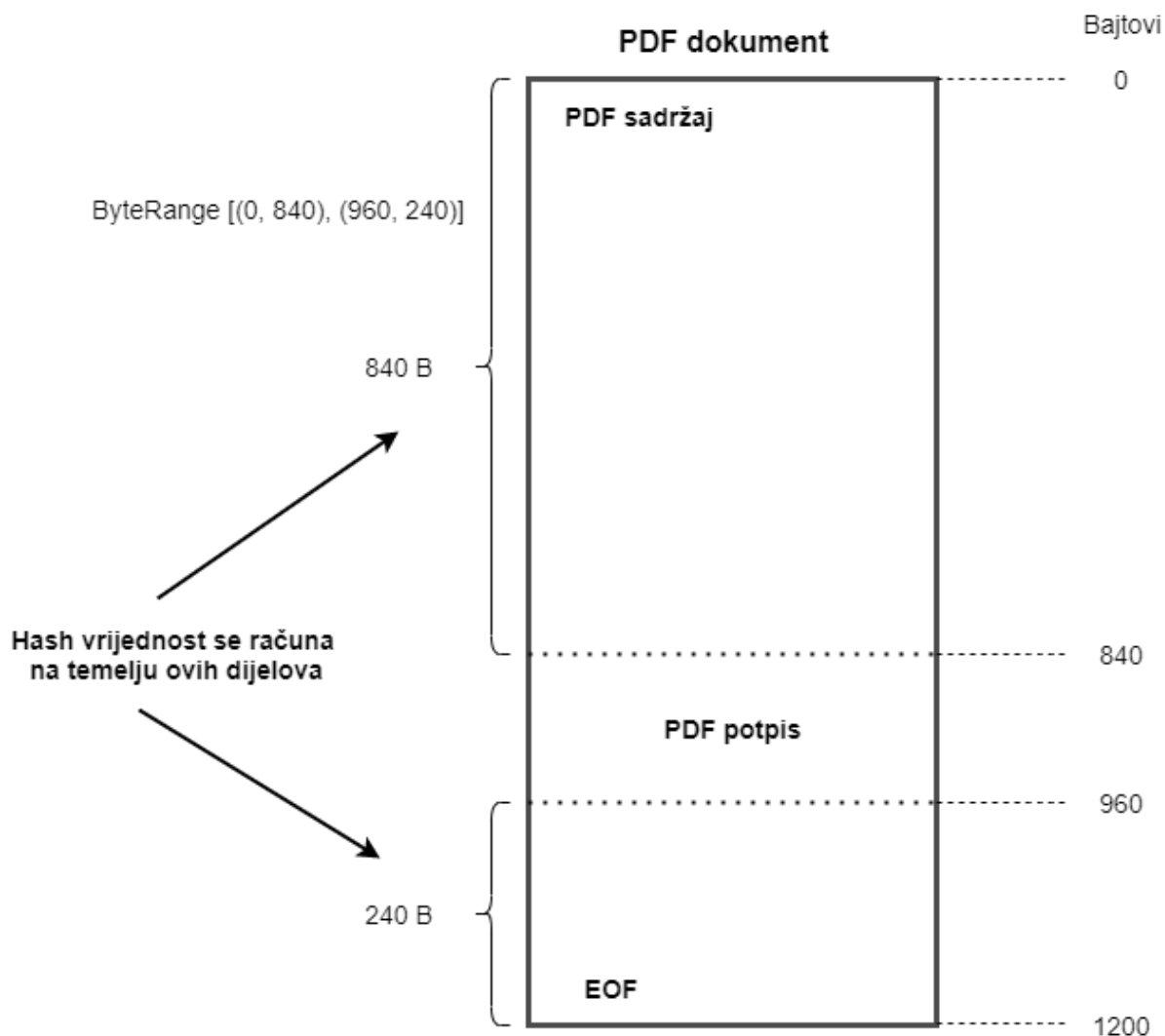
### 4.3 PAdES

PAdES je skup ograničenja i proširenja na PDF i ISO 32000-1 koja implementiraju zahtjeve naprednih elektronskih potpisa. PDF i ISO 32000-1 definiraju okvir za digitalno potpisivanje dokumenata, a PAdES definira profile kojima ih čini usklađenim s AdES standardom.

Digitalni potpisi u okviru ISO 32000-1 podržavaju tri značajke: dodavanje digitalnog potpisa dokumentu, rezerviranje mjesta za potpis u budućnosti i provjeru valjanosti potpisa. Potpis i ostale informacije se nalaze u strukturi PDF-a zvanj rječnik potpisa (engl. *signature dictionary*) [14].

Hash vrijednost se računa na temelju niza bajtova PDF datoteke uključujući i rječnik potpisa, ali isključujući potpis sam. Raspon u kojem se obavlja hash je definiran **ByteRange** poljem rječnika potpisa. Time se osigurava da su svi bajtovi datoteke pokriveni osim samog potpisa.

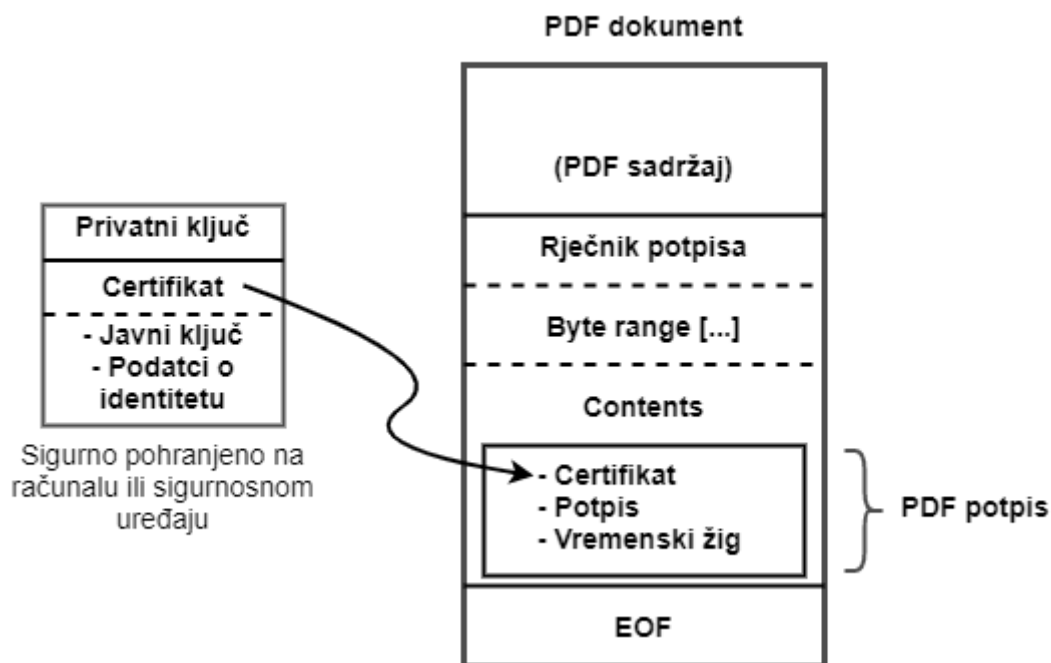
Slika 4.3 prikazuje primjer takvog potpisa za potpis između bajtova 840 i 960 pri čemu je raspon bajtova zadano kao niz uređenih parova (početak, pomak).



Slika 4.3 ISO32000-1 PDF potpis

Binarna vrijednost PDF potpisa postavljena je u **Contents** unos rječnika potpisa. Veličina **Contents** polja se računa na temelju procjene maksimalne veličine potrebne za pohranu potpisa i dodatnih podataka poput podataka o opozivu i vremenskih žigova. Vrijednost potpisa je inicijalno zapisana na disk kao niz 0x00 heksadecimalnih vrijednosti te je naknadno popunjena sa stvarnom vrijednošću [15].

Slika 4.4 prikazuje detaljniji pogled na rječnik potpisa i njegov sadržaj.



Slika 4.4 Rječnik potpisa

#### 4.3.1 Sintaksa

PAdES profili proširuju ISO 32000-1 standard s alternativnim kodiranjem potpisa dodavajući potpisane i nepotpisane attribute ekvivalentne CAdES formatu potpisa.

DER kodirani **SignedData** objekt mora biti dodan u **Contents** unos rječnika potpisa. Može postojati samo jedan autor potpisa, odnosno, može postojati samo jedan objekt tipa **SignerInfo** unutar **SignerInfos** elementa u bilo kojem PDF potpisu.

Zahtjevi za rukovanjem PDF potpisima navedeni u ISO 32000-1 standardu se primjenjuju osim ako nisu nadjačani u ETSI 319 142-1 standardu.

Neki atributi imaju isto ili slično značenje kao ključevi rječnika potpisa opisanim u ISO standardu. Takvi atributi i ključevi se interpretiraju prema tablici koju definira ETSI standard.

Za potrebe produljenja životnog vijeka PDF potpisa, ETSI definira dodatna proširenja na ISO standard. Za potpunu provjeru digitalnog potpisa potrebni su podatci poput lanca certifikata, liste opoziva, OCSP odgovori, itd.

Sigurno spremište dokumenata (engl. *Document security store*) je proširenje koje definira pohranu svih podataka potrebnih za provjeru potpisa. Opcionalno se dodaje i polje podataka povezanih s provjerom valjanosti (engl. *Validation related information*) koje povezuje podatke s konkretnim potpisom.

Životni vijek valjanosti potpisa produljuje se na način da se dodaju dodatne informacije u DSS prilikom dodavanja vremenskih žigova. Svako inkrementalno ažuriranje DSS-a zahtjeva očuvanje vrijednosti prethodnog DSS-a.

#### **4.4 CAdES**

CAdES je skup proširenja sintaksi kriptografske poruke (engl. *Cryptographic message syntax*) koja implementiraju zahtjeve naprednih elektronskih potpisa.

CMS je IETF standard definiran kroz RFC5652 koji opisuje sintaksu za zaštitu podataka. Podržava digitalne potpise i enkripciju. Sintaksa podržava višestruku enkapsulaciju što u omogućava višestruko potpisivanje podataka.

Također omogućeno je potpisivanje proizvoljnih atributa, poput vremena potpisa i protu potpisa, uz sam podatak koji se potpisuje.

CMS vrijednosti su definirane koristeći ASN.1 standard uz BER kodiranje pri čemu su vrijednosti obično niz okteta.

CMS standard je dovoljno generalan da može podržavati brojne vrste sadržaja. *ContentInfo* enkapsulira jednu identificiranu vrstu sadržaja, a identificirani tip može osigurati daljnju enkapsulaciju.

Svaka vrsta sadržaja omogućava obradu u jednom prolasku (engl. *single pass processing*) koristeći BER kodiranje proizvoljne duljine. Operacija jednog prolaska daje dobre rezultate kod velikih podataka i u slučajevima kada nasumičan pristup podacima nije moguć (npr. podatak dolazi kroz cjevovod iz drugog procesa).



Ova operacija nije trivijalna kada je potrebno napraviti DER kodiranje jer duljine različitih komponenti nisu unaprijed poznate. Potpisani atributi unutar **signed-data** sadržaja i ovjereni atributi unutar **authenticated-data** sadržaja moraju biti preneseni u DER formatu kako bi primatelj mogli provjeriti sadržaj koji sadrži jedan ili više nepoznatih atributa.

Potpisani i ovjereni atributi su jedine vrste podataka u CMS-u koje zahtijevaju DER kodiranje [16].

#### 4.4.1 Sintaksa

Slika 4.5 prikazuje identifikator objekta koji identificira vrstu sadržaja:

```
id-ct-contentInfo OBJECT IDENTIFIER ::= { iso(1) member-body(2)
      us(840) rsadsi(113549) pkcs(1) pkcs9(9) smime(16) ct(1) 6 }
```

*Slika 4.5 Identifikator vrste sadržaja*

CMS povezuje identifikator vrste sadržaja sa sadržajem. Sintaksa mora sadržavati ASN.1 tip **ContentInfo**. Slika 4.6 prikazuje definiciju tipa.

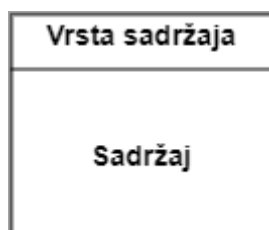
```
ContentInfo ::= SEQUENCE {
    contentType ContentType,
    content [0] EXPLICIT ANY DEFINED BY contentType }

ContentType ::= OBJECT IDENTIFIER
```

*Slika 4.6 Definicija ContentInfo tipa*

**ContentType** označava vrstu sadržaja. To je identifikator objekta u obliku jedinstvenog niza koji dodjeljuje tijelo koje definira taj tip.

Slika 4.7 ilustrira osnovni tip-vrijednost format koji se koristi u CMS-u.

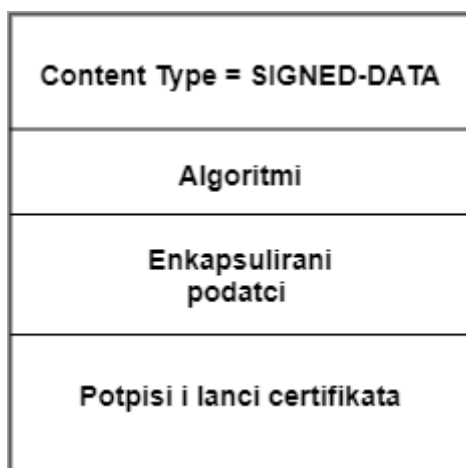


*Slika 4.7 Osnovni CMS tip-vrijednost format*

Standard definira 6 vrsta podataka:

- **Data** – koristi se za proizvoljni niz okteta (npr. tekst) i u pravilu je sadržan u ostalim tipovima
- **Signed-Data** – koristi se za digitalno potpisivanje sadržaja
- **Enveloped-Data** – sadrži enkriptirani sadržaj zajedno s jednim dekriptijskim ključem po primatelju
- **Digested-Data** – sadrži podatke i hash vrijednost sadržaja
- **Encrypted-Data** – sadrži samo enkriptirane podatke
- **Authenticated-Data** – sadrži podatak, MAC i enkriptirane autentikacijske ključeve za jednog ili više primatelja

Slika 4.8 ilustrira format za digitalno potpisivanje koji se koristi u CMS-u.



*Slika 4.8 CMS Signed-Data tip formata*

## 4.5 Kriptografski algoritmi

Odabir kriptografskih algoritama koji se primjenjuju kroz digitalne potpise bitan je parametar usluga koje implementiraju digitalne potpise. Jedan od bitnih faktora prilikom odabira algoritama je interoperabilnost. ETSI standard delegira procjenu sigurnosti kriptografskih algoritama SOG-IS Crypto WG grupi kako bi se izbjegli konflikti prilikom evaluacije sigurnosti pojedinih usluga i proizvoda [17].

Druga standardizacijska tijela i agencije zemalja članica objavljuju slične dokumente s djelomično preklapajućim opsegom preporuka. Takvi dokumenti mogu se iskoristiti kao dodatni materijali prilikom projektiranja implementacije usluga i proizvoda.

SOG-IS evaluacijski okvir razlikuje dvije vrste mehanizama:

- naslijeđene (engl. *legacy*) – algoritmi i parametri koji su široko korišteni, ali ne predstavljaju moderno, „state of the art“, kriptografsko rješenje
- predložene (engl. *recommended*) – „state of the art“ kriptografski algoritmi i parametri

U generalnom slučaju, digitalni potpisi trebali bi biti izrađeni predloženim algoritmima i parametrima osim kada postoji zahtjev za interoperabilnost s postojećim sustavima.

SOG-IS grupa provodi reviziju svake dvije godine ili neposredno nakon otkrivanja ranjivosti.

### 4.5.1 Prihvaćeni algoritmi

Tablica 2 prikazuje skup hash funkcija koje prihvaća SOG-IS grupa pri čemu se SHA-224 ne smatra preporučenom funkcijom [18].

Tablica 2 SOG-IS prihvaćene hash funkcije

Hash funkcija	Reference
SHA-224	FIPS publikacija 180-4
SHA-256	FIPS publikacija 180-4
SHA-384	FIPS publikacija 180-4
SHA-512	FIPS publikacija 180-4
SHA-512/256	FIPS publikacija 180-4
SHA3-256	FIPS publikacija 202

SHA3-384	FIPS publikacija 202
SHA3-512	FIPS publikacija 202

SHA-1 nije prihvaćena funkcija ali HMAC-SHA-1 je prihvaćen i ne smatra se preporučenom funkcijom.

Tablica 3 prikazuje skup algoritama za potpisivanje (engl. *signature algorithms*) koje prihvaća SOG-IS grupa. EC-GDSA je preporučen, ali nije naveden zbog loše interoperabilnosti [17].

Tablica 3 SOG-IS prihvaćeni algoritmi za potpisivanje

Algoritam	Reference
RSA-PKCS#1v1_5	IETF RFC 3447
RSA-PSS	IETF RFC 3447
DSA (FF-DLOG DSA)	FIPS Publication 186-4, ISO/IEC 14888-3
EC-DSA (EC-DLOG EC-DSA)	FIPS Publication 186-4
EC-SDSA-opt (EC-DLOG EC-Schnorr)	ISO/IEC 14888-3

Tablica 4 prikazuje skup kombinacija algoritama za potpisivanje i hash funkcija (engl. *signature suite*) koje prihvaća SOG-IS grupa. Kombinacije zasnovane na eliptičkim krivuljama su odabrane na način da duljina izlaza hash funkcije odgovara veličini ključa odgovarajuće eliptičke krivulje.

Tablica 4 SOG-IS prihvaćene kombinacije algoritama za potpisivanje i hash funkcija

Naziv	Hash funkcija	Algoritam za potpisivanje	Vrsta
sha224-with-rsa	SHA-224	RSA-PKCSv1_5	Nasljeđen
sha256-with-rsa	SHA-256	RSA-PKCSv1_5	Nasljeđen
sha512-with-rsa	SHA-512	RSA-PKCSv1_5	Nasljeđen
rsa-pss with mgf1SHA-256Identifier	SHA-256	RSA-PSS	Preporučen
rsa-pss with mgf1SHA-512Identifier	SHA-512	RSA-PSS	Preporučen

rsa-pss with mgf1SHA3- Identifier	SHA3-256, SHA3- 384 ili SHA3-512	RSA-PSS	Preporučen
sha224-with-ecdsa	SHA-224	EC-Dsa	Preporučen
sha2-with-ecdsa	SHA-256, SHA-384 ili SHA-512	EC-Dsa	Nasljeđen
sha2-with-ecdsa	SHA-256, SHA-384 ili SHA-512	EC-SDSA-opt	Preporučen
sha3-with-ecdsa	SHA3-256, SHA3- 384 ili SHA3-512	EC-Dsa	Preporučen
sha3-with-ecdsa	SHA3-256, SHA3- 384 ili SHA3-512	EC-SDSA-opt	Preporučen

SOG-IS također definira vremensku preporuku funkcija kroz vrijeme za olakšano planiranje i projektiranje. Grupa radi procjene na temelju trenutnih trendova računalne snage potrebne za probijanje algoritma.

Trenutna revizija ne predviđa mehanizme otporne na probijanje kvantnim računalima. Razvijanje sustava čija je svrha zaštita informacija u dugom roku treba uzeti u obzir napredak tehnologije kvantnih računala i implementirati nove mehanizme koji pružaju prikladnu zaštitu.

## 5 DSS

Servis digitalnih potpisa (engl. *Digital signature services*) je otvorena softverska biblioteka za stvaranje i provjeru valjanosti elektronskih potpisa. DSS podržava stvaranje i provjeru interoperabilnih i sigurnih elektronskih potpisa definiranih eIDAS uredbom [19].

DSS je Java biblioteka izdana pod LGPL 2.1 licencom koju razvija i aktivno održava Luksemburška tvrtka Nowina Solutions. Biblioteku mogu koristiti fizičke i pravne osobe u svrhu izrade rješenja za digitalno potpisivanje.

Korištenje DSS razvojnog okvira osigurava usklađenost izrađenih potpisa s EU uredbom i olakšava razvoj rješenja bez obzira na ciljanu platformu.

Glavne značajke razvojnog okvira su stvaranje, proširenje i provjera digitalnih potpisa uz brojne podržane koncepte i značajke:

- upravljanje podacima o opozivu
- izgradanja lanca certifikata
- upravljanje vremenskim žigovima
- REST i SOAP web usluge
- izvješća o provjeri potpisa
- upravljanje povjerenjem
- itd.

DSS također podržava provjeru valjanosti kvalificiranih i naprednih elektronskih potpisa. Biblioteka implementira algoritam koji određuje odgovor na sljedeća pitanja [20]:

- je li certifikat kvalificiran
- koji je tip certifikata
- je li odgovarajući privatni ključ zaštićen QSCD-om

Odgovor na ova tri pitanja određuje može li potpis biti kvalificiran u kontekstu eIDAS uredbe.

DSS projekt se sastoji od više modula koji se mogu referencirati u vlastitom rješenju koristeći Maven. Verzija 5.8 ima sljedeće minimalne zahtjeve za uspješno pokretanje u vlastitom okruženju:

- Java 9 i više
- Maven 3.6 i više

## 5.1 Apstrakcija podataka

Ulazni podatci, odnosno dokumenti, u DSS razvojnom okviru apstrahirani su tipom `DSSDocument` bez obzira na njihov tip. DSS predviđa tri implementacije:

- `InMemoryDocument` – u potpunosti učitava podatak u memoriju na temelju niza bajtova ili ulaznog toka (engl. *input stream*)
- `FileDocument` – odnosi se na postojeću datoteku u sustavu datoteka
- `DigestDocument` – sadrži unaprijed izračunatu hash vrijednost podatka; klijent ne mora poslati čitavi podatak, ali je moguće izraditi potpis jedino u `Detached` načinu rada

Slika 5.1 prikazuje stvaranje PDF dokumenta u memoriji na temelju bajtova ulaznog podatka i imena datoteke.

```
DSSDocument documentToSign = new InMemoryDocument(inputData.getBytes()),  
inputData.getName(), MimeType.PDF);
```

*Slika 5.1 Stvaranje DSS dokumenta u memoriji*

Stvoreni objekt predstavlja originalni podatak u DSS razvojnom okviru te se dalje koristi za operacije potpisivanja i validacije.

Razvojni okvir implementira modul `DSSUtil` koji sadrži pomoćne funkcije često korištene u implementaciji rješenja. Modul definira funkciju `toByteArray` kojom se objekt tipa `DSSDocument` na jednostavan način pretvara u niz bajtova koji je moguće koristiti van konteksta DSS-a.

## 5.2 Potpisivanje

DSS razvojni okvir podržava izradu potpisa za XAdES, CAdES, PAdES, JAdES i ASIC formate u osnovnim profilima. Od verzije 4 nije moguće izraditi potpis jednim od naslijeđenih (engl. *legacy*), standardnih profila, ali je moguće provjeriti valjanost takvih potpisa [12].

Tablica 5 definira podržane profile u DSS-u te ekvivalenciju osnovnih (engl. *baseline*) profila sa standardnim profilima.

Tablica 5 Podržani profili u DSS-u

XAdES		CAdES		PAdES		JAdES
STANDARD	BASELINE	STANDARD	BASELINE	STANDARD	BASELINE	BASELINE
XAdES-BES	XAdES-B	CAdES-BES	CAdES-B	PAdES-BES	PAdES-B	JAdES-B
XAdES-EPES		CAdES-EPES		PAdES-EPES		
XAdES-T	XAdES-T	CAdES-T	CAdES-T	PAdES-T	PAdES-T	JAdES-T
XAdES-XL	XAdES-LT	CAdES-XL	CAdES-LT	PAdES-XL	PAdES-LT	JAdES-LT
XAdES-A	XAdES-LTA	CAdES-A	CAdES-LTA	PAdES-LTV	PAdES-LTA	JAdES-LTA

Potpisivanje dokumenta se provodi kroz tri nedjeljiva koraka:

1. izračunaj sažetak (engl. *digest*) koji će biti potpisan
2. potpiši sažetak
3. potpiši dokument (dodaj potpisani sažetak dokumentu)

DSS okvir u potpunosti upravlja ovim koracima, a klijentski kod upravlja postavkama i ulaznim podacima operacije.

Klijentski kod mora navesti vrstu pohrane ključa (engl. *key store*) koji se koristi za potpisivanje dokumenata. DSS okvir predviđa četiri implementacije:

- `Pkcs11SignatureToken` – omogućava komunikaciju s pametnim karticama kroz PKCS#11 sučelje
- `Pkcs12SignatureToken` – omogućava potpisivanje s PKCS#12 datotekama



- `MSCAPISignatureToken` – omogućava potpisivanje s MS CAPI (Microsoftovo sučelje za komunikaciju s pametnim karticama)
- `JKSSignatureToken` – omogućava potpisivanje s Java Key Store datotekama (.jks)

DSS razvojni okvir koristi sučelje `SignatureTokenConnection` za upravljanje implementacijom procesa potpisivanja kako bi potpisivanje bilo neovisno o formatu kojim se potpisuje. Okvir implementira apstraktnu klasu `AsyncSignatureTokenConnection` koja omogućava računanje hash vrijednosti i potpisivanja u različitim nitima ili na različitom sklopovlju.

Kroz ovakav dizajn, biblioteka omogućava proširenje funkcionalnosti proizvođačima čitača kartica i razvojnim timovima koji implementiraju vlastita rješenja na temelju ove biblioteke.

Slika 5.2 prikazuje funkciju kojom se stvara PKCS12 objekt na temelju Base64 kodirane PKCS#12 datoteke i lozinke.

```
private Pkcs12SignatureToken BuildSignatureToken(String b64Certificate,
    String password) {

    byte[] certificateBytes = Base64.getDecoder().decode(b64Certificate);

    return new Pkcs12SignatureToken(certificateBytes,
        new KeyStore.PasswordProtection(password.toCharArray()));
}
```

*Slika 5.2 Stvaranje PKCS12 objekta za potpisivanje*

Ovako pripremljen objekt koji sadrži certifikat može se iskoristiti za potpisivanje podataka.

Slika 5.3 prikazuje funkciju za potpisivanje PDF dokumenta.

```

public DSSDocument SignPdf(CertificateModel certificate,
    DSSDocument document, SignatureLevel signatureLevel) {

    Pkcs12SignatureToken token = BuildSignatureToken(
        certificate.getB64Certificate(),
        certificate.getCertificatePassword());

    DSSPrivateKeyEntry privateKey = GetPrivateKey(token);

    PAdESSignatureParameters parameters = GetPadesParameters(
        privateKey.getCertificate().getCertificate(),
        privateKey.getCertificateChain(), signatureLevel);

    // Get the SignedInfo segment that need to be signed.
    ToBeSigned dataToSign = padesService.getDataToSign(document,
        parameters);

    SignatureValue signatureValue = token.sign(dataToSign,
        parameters.getDigestAlgorithm(),
        privateKey);

    // We invoke the padesService to sign the document with
    // the signature value obtained in the previous step.
    return padesService.signDocument(document, parameters, signatureValue);
}

```

*Slika 5.3 Potpisivanje PDF dokumenta*

U prvom koraku se kreira PKCS12 objekt za potpisivanje na temelju certifikata. Iz tog objekta se izvede privatni ključ koji će biti korišten za potpisivanje.

Dohvaćaju se parametri koji će biti korišteni za potpisivanje. Parametri uključuju razinu potpisa, hash algoritam, podatke o certifikatu kojim se potpisuje i čitavom lancu certifikata. Dodavanje lanca certifikata pojednostavljuje provjeru valjanosti potpisa.

Metoda `getDataToSign` dohvaća podatke iz dokumenta koje je potrebno potpisati na temelju proslijeđenih parametara. Ti podaci odgovaraju `SignedInfo` XMLDSig elementu u slučaju XAdES-a i rasponu bajtova zadanom u rječniku potpisa u slučaju PAdES-a.

Dohvaćeni podaci za potpisivanje koriste se kao ulaz metodi `sign` objekta spremišta ključeva `PKCS12SignatureToken`. Prvi parametar je niz bajtova koje treba potpisati, drugi parametar je hash algoritam, a treći je privatni ključ kojim se obavlja potpisivanje.

Potrebno je eksplicitno navesti privatni ključ jer jedno spremište ključeva (engl. *KeyStore*) može sadržavati više certifikata odnosno privatnih ključeva.

U posljednjem koraku se potpis integrira s dokumentom koji se potpisuje ovisno o načinu rada algoritma. PAdES podržava samo *Enveloped* način rada, dok je za XAdES i CAdES moguće postaviti više različitih načina rada prilikom postavljanja parametara.

Pozivom metodi `signDocument` obavlja se ova integracija. Uz originalni podatak i vrijednost potpisa, proslijeđuju joj se i parametri koji određuju način ugradnje potpisa u podatak. Metoda vraća objekt tipa `DSSDocument` koji je moguće dalje u implementaciji logike sustava.

Razdvajanje procesa potpisivanja u tri koraka omogućava razdvajanje odgovornosti prilikom implementacije sustava. Preporučena praksa je razdvojiti operacije potpisivanja i ostatak logike sustava u zasebne jedinice sustava.

Slika 5.4 prikazuje primjer jednostavnog XAdES potpisa B razine osnovnog profila.

```

<xades:SignedProperties Id="xades-id-ea3e16770317bb1a3e97244292931644">
  <xades:SignedSignatureProperties>
    <xades:SigningTime>2018-03-20T08:17:35Z</xades:SigningTime>
    <xades:SigningCertificateV2>
      <xades:Cert>
        <xades:CertDigest>
          <ds:DigestMethod Algorithm="http://www.w3.org/...#sha1" />
          <ds:DigestValue>2FeANjXzi09x2877SfclRlRVjlE=</ds:DigestValue>
        </xades:CertDigest>
        <xades:IssuerSerialV2>MD4...e7Vw==</xades:IssuerSerialV2>
      </xades:Cert>
    </xades:SigningCertificateV2>
    <xades:SignatureProductionPlaceV2>
      <xades:City>SimCity</xades:City>
      <xades:StateOrProvince>Luxembourg</xades:StateOrProvince>
      <xades:PostalCode>1234</xades:PostalCode>
      <xades:CountryName>BE</xades:CountryName>
    </xades:SignatureProductionPlaceV2>
    <xades:SignerRoleV2>
      ...
    </xades:SignerRoleV2>
  </xades:SignedSignatureProperties>
  <xades:SignedDataObjectProperties>
    <xades:DataObjectFormat ObjectReference="#r-id-1">
      <xades:MimeType>text/xml</xades:MimeType>
    </xades:DataObjectFormat>
    <xades:CommitmentTypeIndication>
      <xades:CommitmentTypeId>
        <xades:Identifier>http://uri.etsi.org/...</xades:Identifier>
      </xades:CommitmentTypeId>
      <xades:AllSignedDataObjects />
    </xades:CommitmentTypeIndication>
    <xades:CommitmentTypeIndication>
      <xades:CommitmentTypeId>
        <xades:Identifier>http://uri.etsi.org/...</xades:Identifier>
      </xades:CommitmentTypeId>
      <xades:AllSignedDataObjects />
    </xades:CommitmentTypeIndication>
    <xades:AllDataObjectsTimeStamp Id="TS-...279E">
      <ds:CanonicalizationMethod
        Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
      <xades:EncapsulatedTimeStamp>MI...AAA=</xades:EncapsulatedTimeStamp>
    </xades:AllDataObjectsTimeStamp>
  </xades:SignedDataObjectProperties>
</xades:SignedProperties>

```

Princip potpisivanja podataka svim podržanim standardima i profilima u DSS razvojnom okviru implementiran je kroz iste korake i isto klijentsko sučelje koji su opisani u ovom poglavlju.

### **5.3 Provjera valjanosti potpisa**

Proces provjere valjanosti potpisa temelji se na ETSI standardima pri čemu rezultat provjere mora biti TOTAL - FAILED, TOTAL - PASSED, ili INDETERMINATE.

TOTAL - PASSED rezultat označava da je potpis u potpunosti prošao provjeru valjanosti i da je sukladan definiranoj politici provjere potpisa.

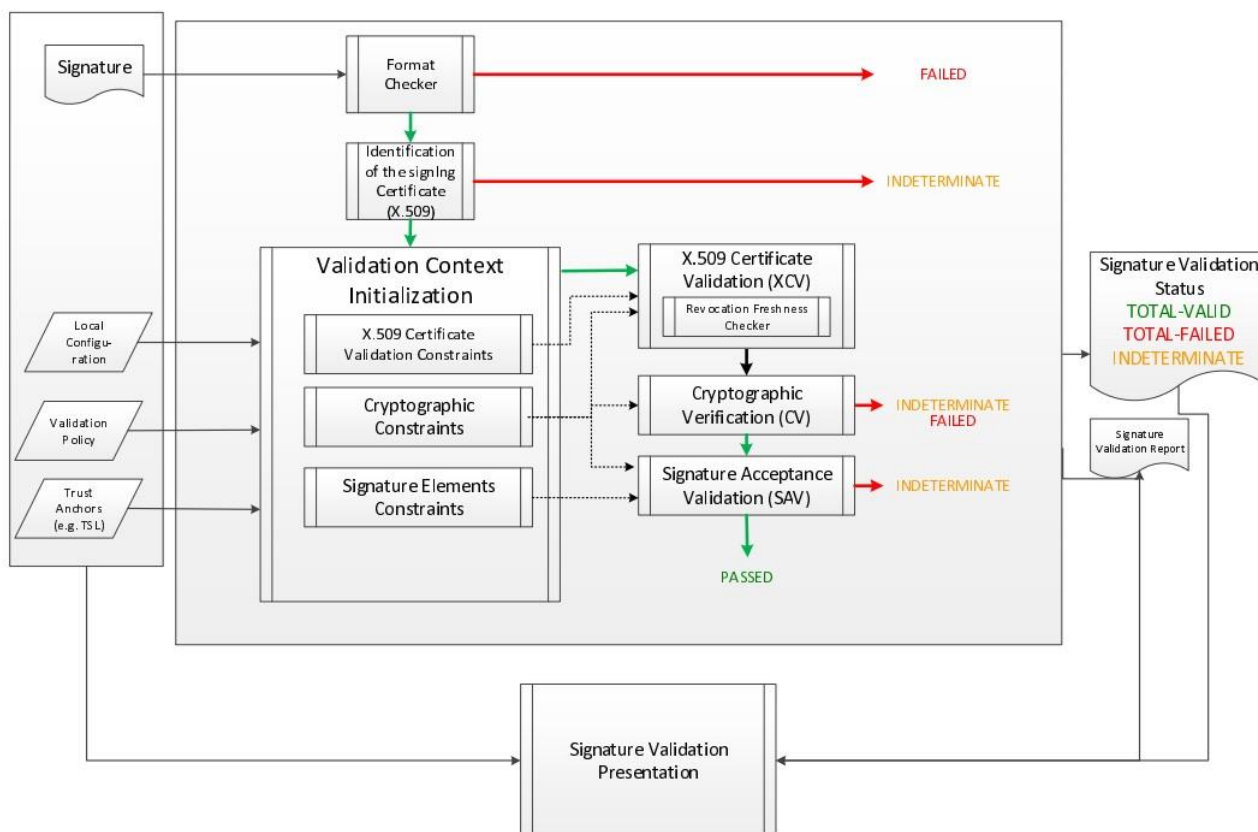
TOTAL - FAILED rezultat označava da je format potpisa neispravan ili da sama vrijednost potpisa nije valjana.

INDETERMINATE rezultat označava da format i vrijednost potpisa nisu neispravni, ali nema dovoljno informacija na temelju koji se može provjeriti ispravnost potpisa.

Za svaku stavku koja se provjerava, razvojni okvir daje informacije na temelju kojih je određeni status dodijeljen.

Provjera valjanosti potpisa u razvojnom okviru temeljena je na ETSI EN 319 102-1 standardu pri čemu je politika valjanosti glavni pokretač procesa. Proces validacije provjerava postojanje određenih podataka i njihovu valjanost te provjerava vremenske ovisnosti među tim podacima.

Slika 5.5 predstavlja pojednostavljeni dijagram procesa provjere valjanosti potpisa. Veze i elementi dijagrama definiraju logički skup provjera korištenih u procesu.



Slika 5.5 Pojednostavljeni process provjere valjanosti potpisa u DSS-u (izvor: [https://ec.europa.eu/cefdigital/DSS/webapp-demo/doc/dss-documentation.html#\\_validation\\_process](https://ec.europa.eu/cefdigital/DSS/webapp-demo/doc/dss-documentation.html#_validation_process), preuzeto 22.8.2021.)

Rezultat ovog procesa su četiri izvješća različitih razina detalja:

- jednostavni izvještaj
- detaljni izvještaj
- dijagnostički izvještaj
- ETSI validacijski izvještaj

Provjera valjanosti potpisa u razvojnom okviru izvršava se kroz iste korake i klijentska sučelja bez obzira na vrstu potpisa.

Slika 5.6 prikazuje funkciju za provjeru valjanosti potpisa proizvoljne binarne datoteke s odvojenim (engl. *detached*) potpisom.

```

public SimpleReport VerifySignedBinary(DSSDocument originalDocument,
    DSSDocument xadesSignature) {

    SignedDocumentValidator validator = SignedDocumentValidator
        .fromDocument(xadesSignature);

    validator.setValidationLevel(ValidationLevel.TIMESTAMPS);
    validator.setCertificateVerifier(this.certificateVerifier);
    validator.setTokenExtractionStrategy(
        TokenExtractionStrategy.EXTRACT_CERTIFICATES_ONLY);
    validator.setDetachedContents(Arrays.asList(originalDocument));

    Reports reports = validator.validateDocument(this.validationPolicy);

    return reports.getSimpleReport();
}

```

*Slika 5.6 Provjera valjanosti potpisa binarne datoteke*

U prvom koraku se stvara `SignedDocumentValidator` objekt koji upravlja provjerom pozivom statičkoj funkciji `fromDocument`.

Objekt `validator` koristi `certificateVerifier` objekt za provjeru statusa korištenih certifikata. Klasa `CertificateVerifier` definira izvore povjerenja koji se koriste tijekom validacije. Slika 5.7 prikazuje postupak stvaranja objekta tipa `CertificateVerifier`.

```

@Bean
public CertificateVerifier certificateVerifier() {
    CommonCertificateVerifier certificateVerifier =
        new CommonCertificateVerifier();

    certificateVerifier.setDataLoader(new CommonsDataLoader());
    certificateVerifier.setTrustedCertSources(rootCaCertificateSource());
    certificateVerifier.setAdjunctCertSources(adjunctCertificatesSource());

    certificateVerifier.setOcspSource(OCSPRevocationSource());
    certificateVerifier.setCrlSource(CRLRevocationSource());

    certificateVerifier.setCheckRevocationForUntrustedChains(true);

    return certificateVerifier;
}

```

*Slika 5.7 Stvaranje CertificateVerifier objekta*

Pozivom metode `validateDocument` obavlja se provjera valjanosti na temelju argumenta validacijske politike. Metoda vraća objekt tipa `Reports` koji sadrži različite poglede na rezultat provjere valjanosti.

Slika 5.8 prikazuje primjer jednostavnog izvještaja provjere valjanosti.



```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<SimpleReport ValidationTime="2020-12-
14T11:03:21" xmlns="http://dss.esig.europa.eu/validation/simple-report">
  <ValidationPolicy>
    <PolicyName>QES AdESQC TL based</PolicyName>
    <PolicyDescription>Validate electronic signatures and indicates whe
ther they are Advanced electronic Signatures (AdES), AdES supported by a Qu
alified Certificate (AdES/QC) or a
      Qualified electronic Signature (QES). All certificates and
their related chains supporting the signatures are validated against the EU
Member State Trusted ...
    </PolicyDescription>
  </ValidationPolicy>
  <DocumentName>EmptyPage-signed-pades-baseline-b.pdf</DocumentName>
  <ValidSignaturesCount>1</ValidSignaturesCount>
  <SignaturesCount>1</SignaturesCount>
  <Signature SignatureFormat="PAdES-BASELINE-B" ExtensionPeriodMin="2020-
12-14T11:04:21" ExtensionPeriodMax="2027-01-21T23:59:59" Id="S-
08113A9BAB65F6271F837FF4992635CC725B49D27B1AED0D714EAD428BE98C6E">
    <CertificateChain>
      <Certificate>
        <id>C-4...9F</id>
        <qualifiedName>Pierrick Vandenbroucke</qualifiedName>
      </Certificate>
      <Certificate>
        <id>C-2...DA</id>
        <qualifiedName>Citizen CA</qualifiedName>
      </Certificate>
      <Certificate>
        <id>C-7...8F</id>
        <qualifiedName>Belgium Root CA4</qualifiedName>
      </Certificate>
    </CertificateChain>
    <Indication>TOTAL_PASSED</Indication>
    <Warnings>The organization name is missing in the trusted certifica
te!</Warnings>
    <SigningTime>2019-08-27T14:06:11</SigningTime>
    <BestSignatureTime>2020-12-14T11:03:21</BestSignatureTime>
    <SignedBy>Pierrick Vandenbroucke (Signature)</SignedBy>
    <SignatureLevel description="Qualified Electronic Signature">QESig<
/SignatureLevel>
    <SignatureScope name="Full PDF" scope="FULL">Full document</Signatu
reScope>
  </Signature>
</SimpleReport>

```

Slika 5.8 Primjer jednostavnog izvještaja

### 5.3.1 Politika valjanosti

DSS razvojni okvir omogućava definiranje politike valjanosti potpisa kroz kod i kroz konfiguracijsku XML datoteku. Proces provjere valjanosti potpisa se tada temelji na postavkama navedenima kroz politiku.

Svako ograničenje ima dodijeljenu FAIL, WARN ili INFORM razinu koja definira ponašanje procesa provjere u slučaju da potpis krši zadano ograničenje.

Neka od ograničenja koja je moguće definirati politikom su ponašanje vezano uz certifikate, vremenske žigove, korištene algoritme i parametre algoritama, itd.

## 5.4 Upravljanje povjerenjem

DSS razvojni okvir omogućava upravljanje povjerenjem kroz tzv. sidra povjerenja (engl. *trust anchor*). Kroz njih se definira povjerenje prema entitetima odnosno potpisima koje ti entiteti generiraju.

Od verzije 5.6. razvojnog okvira omogućeno je postavljanje jednog ili više izvora pouzdanih certifikata. Izvori se mogu postaviti iz spremišta povjerenja (engl. *trust store*), pouzdane liste (engl. *trusted list*), i liste pouzdanih listi (engl. *list of trusted lists*).

U slučaju izrade vlastite PKI, moguće je dodati CA certifikat u popis pouzdanih certifikata. Na taj način moguće je stvoriti naprednih elektronski potpis koji je valjan u okviru ETSI standarda uz povjerenje prema privatnom CA.

Slika 5.9 prikazuje funkciju kojom se dodaje izvor pouzdanih certifikata za privatni CA certifikat i certifikat usluge pružanja vremenskog žiga.

```

@Bean
public CommonTrustedCertificateSource rootCaCertificateSource() {
    try {
        KeyStoreCertificateSource keystore = new KeyStoreCertificateSource(
            rootCA.getFile(),
            customRootCAksType,
            customRootCAksPassword);

        CommonTrustedCertificateSource trustedCertificateSource =
            new CommonTrustedCertificateSource();

        trustedCertificateSource.importAsTrusted(keystore);

        CertificateFactory certFactory = CertificateFactory.getInstance(
            "X.509");
        X509Certificate myPkiRootCert = (X509Certificate) certFactory
            .generateCertificate(rootCA2.getInputStream());
        X509Certificate timestampPkiRootCert = (X509Certificate)
            certFactory.generateCertificate(
                timestampRootCa.getInputStream());

        trustedCertificateSource.addCertificate(
            new CertificateToken(myPkiRootCert));
        trustedCertificateSource.addCertificate(
            new CertificateToken(timestampPkiRootCert));

        return trustedCertificateSource;
    } catch (IOException | CertificateException e) {
        throw new DSSException("Unable to load the root CA file ", e);
    }
}

```

*Slika 5.9 Dodavanje privatnog CA u pouzdane izvore*

## 5.5 Izvori informacija o opozivu

Svaki certifikat treba biti provjeren koristeći CRL ili OCSP prilikom provjere valjanosti potpisa. Informacija o opozivu može doći iz više različitih izvora. DSS razvojni okvir implementira CRLSource i OCSPSource sučelja koja pružaju generički način pristupanju CRL i OCSP izvorima.

U praksi je moguće implementirati predmemoriju (engl. *cache*) kako bi se ovaj postupak dodatno ubrzao.

Razvojni okvir prvo pokušava dohvatiti odgovor od OCSP poslužitelja, a ako to nije moguće šalje upit CRL poslužitelju.

Izvori ovih informacija mogu biti mrežni ili mogu biti lokalno dostupni. Razvojni okvir pruža implementaciju za svaki od ovih scenarija. Za mrežne izvore upit se šalje poslužiteljima koji su navedeni u certifikatu:

- CRL – naveden u polju CRL distribucijska mjesta (engl. *CRL distribution points*)
- OCSP - naveden u polju pristupa informacijama nadležnog tijela (engl. *Authority Information Access*) AIA

Slika 5.10 prikazuje dvije funkcije za stvaranje CRL i OCSP izvora informacija u sklopu DSS razvojnog okvira koje se mogu pridružiti `CertificateVerifier` objektu.

```

@Bean
public RevocationSource<CRL> CRLRevocationSource(){
    OnlineCRLSource onlineCRLSource = new OnlineCRLSource();

    onlineCRLSource.setDataLoader(new CommonsDataLoader());

    onlineCRLSource.setPreferredProtocol(Protocol.HTTP);

    return onlineCRLSource;
}

@Bean
public RevocationSource<OCSP> OCSPRevocationSource(){
    // Instantiates a new OnlineOCSPSource object
    OnlineOCSPSource onlineOCSPSource = new OnlineOCSPSource();
    // Allows setting an implementation of `DataLoader` interface,
    // processing a querying of a remote revocation server.
    // `CommonsDataLoader` instance is used by default.
    onlineOCSPSource.setDataLoader(new OCSPDataLoader());
    // Defines an arbitrary integer used in OCSP source
    // querying in order to prevent a replay attack.
    // Default : null (not used by default).
    onlineOCSPSource.setNonceSource(new SecureRandomNonceSource());
    // Defines a DigestAlgorithm being used to generate
    // a CertificateID in order to complete an OCSP request.
    // OCSP servers supporting multiple hash functions
    // may produce a revocation response with a digest
    // algorithm depending on the provided CertificateID's algorithm.
    // Default : SHA1 (as a mandatory requirement to be
    // implemented by OCSP servers. See RFC 5019).
    onlineOCSPSource.setCertIDDigestAlgorithm(DigestAlgorithm.SHA256);

    return onlineOCSPSource;
}

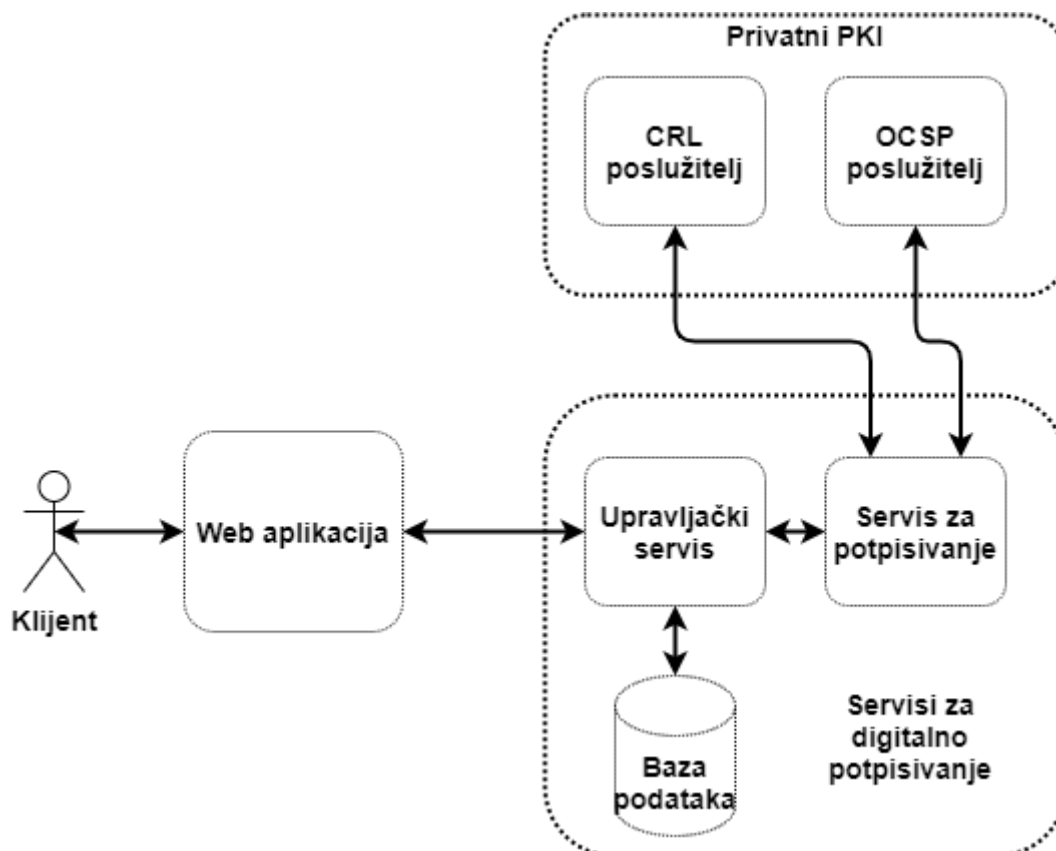
```

*Slika 5.10 Stvaranje objekata za dohvaćanje informacija o opozivu*

Razvojni okvir daje točke proširenja za `RevocationSource<T>` objekte, a zadane postavke omogućavaju dohvaćanje rezultata od svih standardnih CRL i OCSP poslužitelja.

## 6 Arhitektura sustava


Implementirani sustav za digitalno potpisivanje sastoji se od klijentske web aplikacije, skupa servisa zajedno s bazom podataka i potporne PKI. Slika 6.1 prikazuje arhitekturu implementiranog sustava.



Slika 6.1 Arhitektura sustava

### 6.1 Web aplikacija

Web aplikacija je sučelje sustava za digitalno potpisivanje. Sva interakcija korisnika odvija se putem web aplikacije. U nastavku je dan pregled sučelja web aplikacije.



Log in


Email Address \*

Password \*

LOG IN

[Don't have an account? Register now.](#)

*Slika 6.2 Prijava u sustav*



Sign up

First Name \* Last Name \*

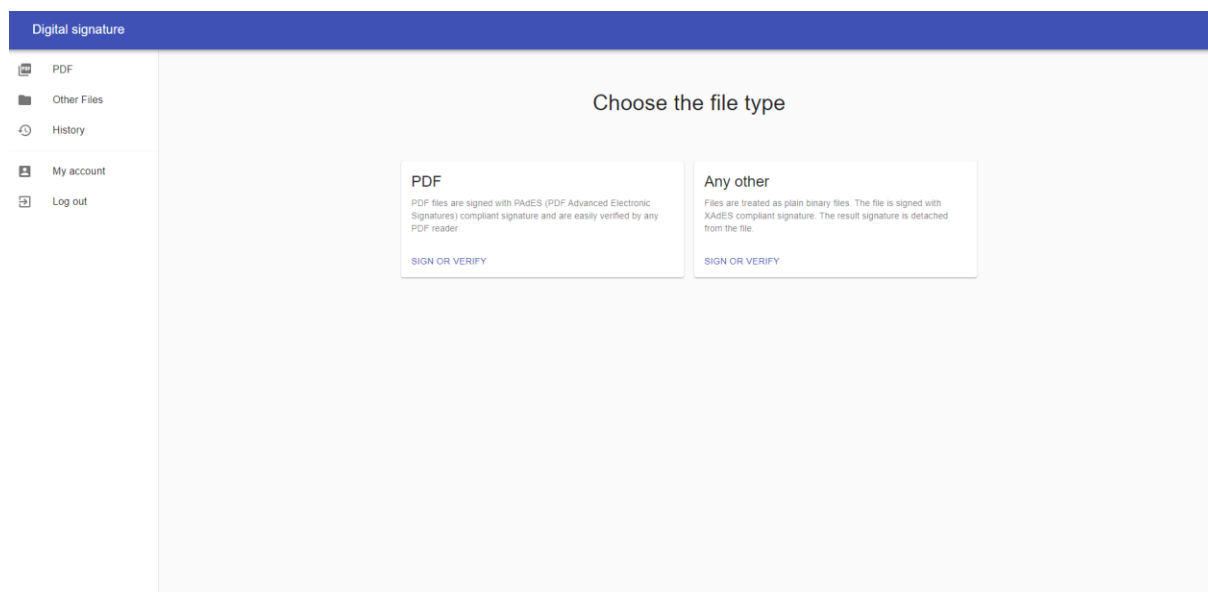
Email Address \*

Password \*

SIGN UP

[Already have an account? Sign in](#)

*Slika 6.3 Registracija*

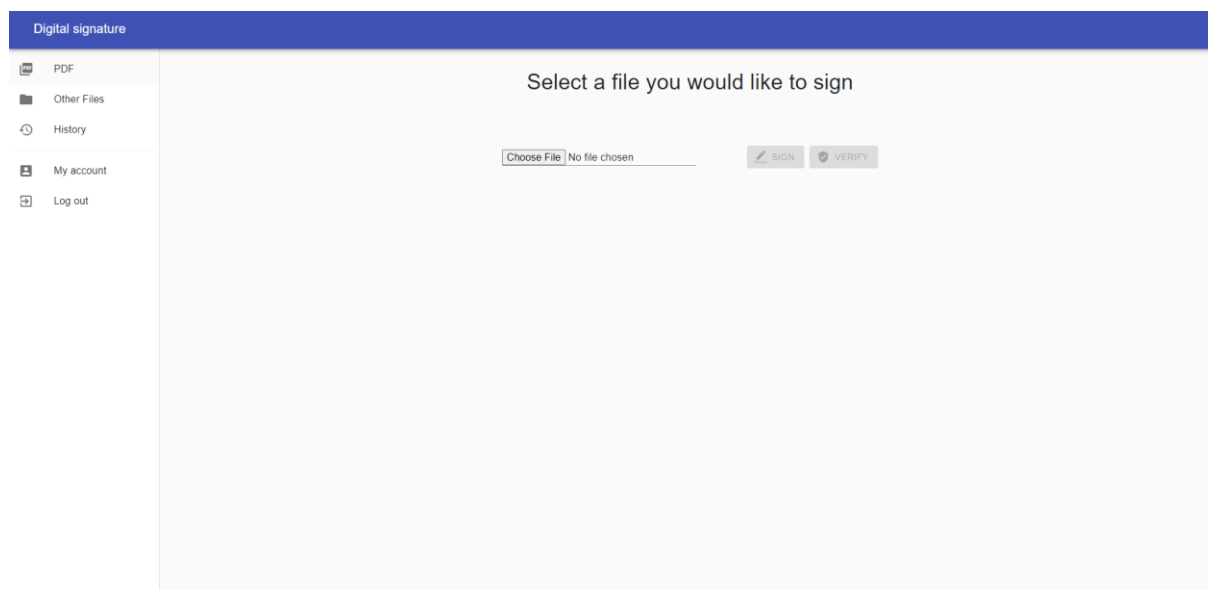


*Slika 6.4 Naslovna stranica*

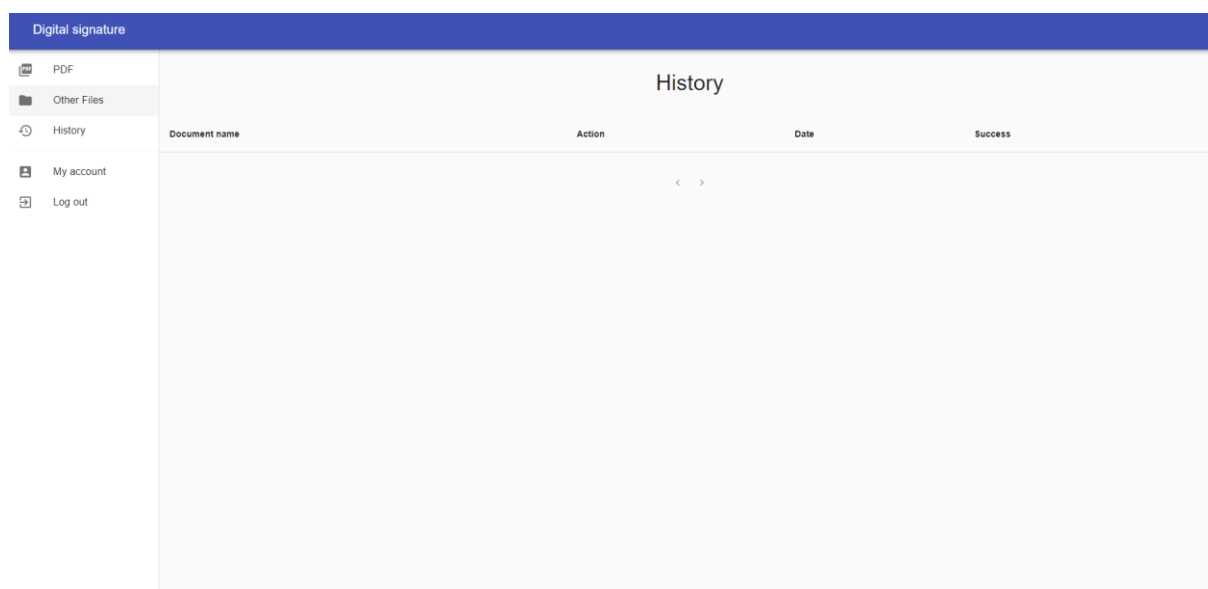


*Slika 6.5 Potpisivanje i verifikacija PDF dokumenata*





*Slika 6.6 Potpisivanje i verifikacija binarnih datoteka*



*Slika 6.7 Povijest*

Digital signature

PDF  
Other Files  
History  
My account  
Log out

## My account

Make changes to your account

### Personal information

Make changes to your personal information

Email  
antonio@fesb.hr

First name  
Antonio

Last name  
Kristicevic

Organization name

### Certificate

Import/Export certificate used for digital signature services

IMPORT EXPORT

SAVE CANCEL

*Slika 6.8 Upravljanje osobnim podacima*

## 6.2 Servis za potpisivanje

Servis za potpisivanje koristi DSS razvojni okvir za implementaciju izrade i provjere valjanosti digitalnih potpisa. Ovaj servis ne sadrži podatke o korisnicima te njihovim podacima i certifikatima. Ovaj servis implementira PAdES i XAdES vrste potpisivanja s bilo kojom od četiri razine potpisa.

Servis nije izložen na javnoj mreži te mu nije moguće direktno pristupiti. Komunikaciju s ovim servisom izvršava servis za upravljanje digitalnim potpisima u privatnoj mreži.

## 6.3 Servis za upravljanje digitalnim potpisima

Servis za upravljanje digitalnim potpisima implementira REST API koji omogućuje izvršavanje željenih operacija na sustav. Web aplikacija koristi ovaj servis za izvršavanje svih radnji. Ovaj servis je izložen na javnoj mreži pa je moguća direktna konzumacija njegovih funkcionalnosti kroz bilo koji REST klijent.

Ovaj servis upravlja popratnim informacijama i funkcionalnostima poput upravljanje korisnicima i njihovim ulogama, upravljanje povijesti operacija, itd. Pored toga, ovaj servis

pruža jednostavan pogled na uslugu digitalnog potpisivanja te omogućava korištenje ove usluge bez poznavanja detalja o digitalnim potpisima.

#### **6.4 CRL poslužitelj**

CRL poslužitelj je dio PKI koji je odgovoran za posluživanje liste opoziva. Implementirani sustav daje primjer privatnog PKI. U slučaju da se koristi certifikat koji nije izdao testni CA, ovaj CRL poslužitelj neće biti korišten, već će biti korišten onaj poslužitelj koji je naveden u samom certifikatu.

Teoretski bi ovaj poslužitelj mogao služiti kao posrednik (engl. *proxy*) preko kojeg se dohvaćaju sve ostale liste opoziva, ali to je van opsega implementacije ovog sustava.

#### **6.5 OCSP poslužitelj**

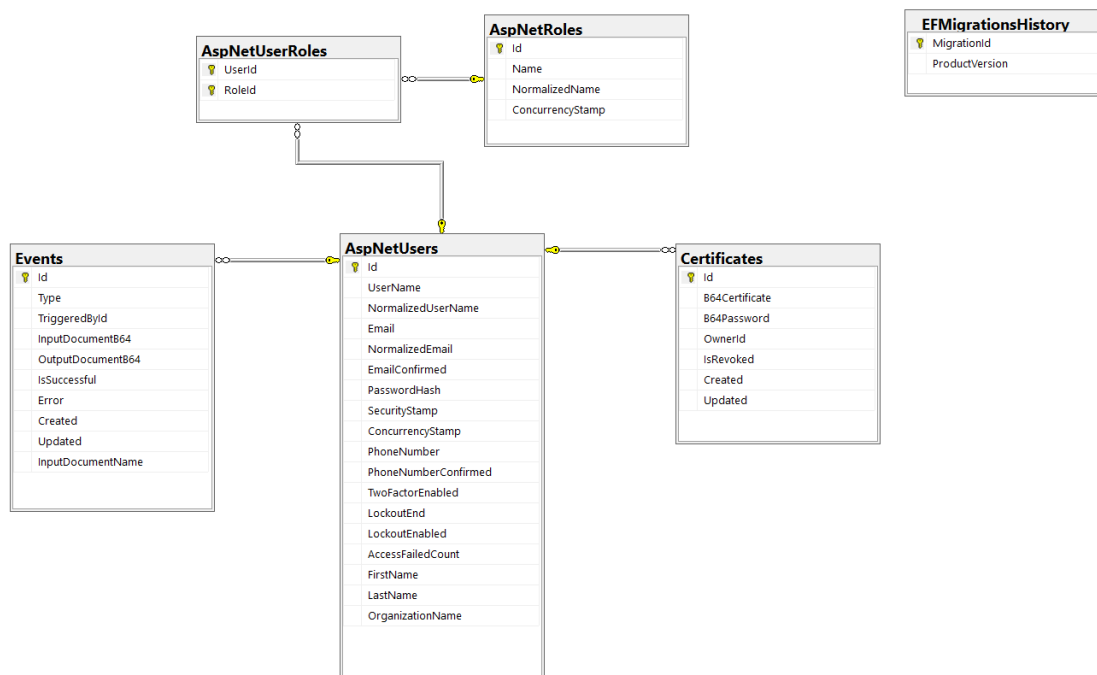
OCSP poslužitelj je dio PKI koji je odgovoran za davanje odgovora na upit o valjanosti certifikata. Slično kao CRL poslužitelj, biti će pozvan samo kada se radi provjera valjanosti certifikata kojeg je izdao testni CA.

Moguće je pohraniti OCSP odgovore u privremenu memoriju u kratkom roku, ali to je van opsega implementacije ovog sustava.

#### **6.6 Baza podataka**

Baza podataka služi za pohranu svih potrebnih podataka usluge za digitalno potpisivanje. Sustav koristi SQL Server relacijsku bazu podataka, ali uz minimalne modifikacije moguće je koristiti i druge relacijske baze podataka uz pomoć apstrakcijskog sloja na aplikacijskoj razini.

Slika 6.9 prikazuje relacijski model baze podataka generiran Microsoft SQL Server Management Studio alatom.



Slika 6.9 Relacijski model baze podataka

Baza podataka, između ostalog, pohranjuje i korisničke certifikate u enkriptiranom obliku. Poboljšanje u sigurnosnom smislu bi bilo korištenje HSM-a za pohranu certifikata.

Na tržištu postoji široka ponuda fizičkih i HSM-ova u oblaku. EU definira popis HSM-ova koji upadaju u kategoriju QSCD pa je uz pomoć njih moguće izraditi QES i time dodatno podići razinu sigurnosti sustava i pruženih usluga.

## 7 Implementacija sustava

U ovom poglavlju opisana je implementacija samog sustava, a naglasak je stavljen na specifičnosti implementacije ovog sustava. Ostali popratni mehanizmi i implementacijski detalji biti će navedeni uz minimalnu razinu informacija dovoljnu za shvaćanje funkcionalnosti sustava.

Svi servisi sustava pripremljeni su za rad u Docker kontejneru što omogućava niz korisnih svojstava poput prenosivosti, kompatibilnosti, lakoće pokretanja, itd. Čitav sustav i međuovisnosti individualnih servisa definirani su u Compose datoteci.

Compose je alat za definiciju i pokretanje više Docker aplikacija. Compose omogućava definiciju servisa sustava u obliku YAML datoteke i pokretanje čitavog sustava jednom naredbom [21].

Slika 7.1 prikazuje sadržaj Compose datoteke koja definira sustav. Detalji ove datoteke su izostavljeni zbog lakšeg prikaza.

```

version: '3.8'
services:
  manager-api:
    image: ${DOCKER_REGISTRY-}digital-signature-manager-api
    build:
      context: src/DigitalSignatureManager
      dockerfile: Api/Dockerfile
    ports:
      - "4200:4200"
      - "4201:4201"
    depends_on:
      - db
      - signer-api
  signer-api:
    image: ${DOCKER_REGISTRY-}digital-signature-signer-api
    build:
      context: src/DigitalSignatureSigner
      dockerfile: Dockerfile
    depends_on:
      - crl-provider
      - ocsp-provider
  db:
    image: "mcr.microsoft.com/azure-sql-edge:latest"
    volumes:
      - events_mssql:/var/opt/mssql
    ports:
      - 1433:1433
  crl-provider:
    image: ${DOCKER_REGISTRY-}digital-signature-crl-provider
    build:
      context: src/CRLProvider
      dockerfile: Dockerfile
    hostname: crl.example.com
    ports:
      - 80:80
  ocsp-provider:
    image: ${DOCKER_REGISTRY-}digital-signature-ocsp-provider
    build:
      context: src/OCSPProvider
      dockerfile: Dockerfile
    hostname: ocsp.example.com
    ports:
      - 2560:80

```

*Slika 7.1 Compose datoteka sustava*

## 7.1 Web Aplikacija

Web aplikacija implementirana je koristeći JavaScript jezik i React biblioteke. Web aplikacija je jednostavno sučelje oko sustava koje implementira nekoliko jednostavnih stranica.

Za dizajn web aplikacije korišten je Material-UI razvojni okvir za React. Material-UI definira brojne komponente konzistentno stilizirane sukladno Material-UI dizajnerskom sustavu.

Za održavanje stanja u aplikaciji korišten je React `Context`. `Context` omogućava dijeljenje vrijednosti između komponenata bez eksplicitnog prosljeđivanja vrijednosti kroz cijelo stablo komponenti. `Context` se obično koristi za dijeljenje globalnog stanja na razini čitave aplikacije [22].

U implementiranoj web aplikaciji `Context` se koristi za upravljanje podacima o autentifikaciji. Samo prijavljeni korisnici mogu koristiti usluge sustava. Uz prijavljenog korisnika vezuje se pripadajući certifikat pa autentifikacija osigurava da samo vlasnik certifikata može izvršiti potpis.

Postupak prijave u sustav može se dodatno ojačati u sigurnosnom smislu dodavanjem dvofaktorske autentifikacije ili drugih naprednih sigurnosnih mehanizama. U sklopu implementacije ovog sustava koristi se samo lozinka, ali je moguće proširenje drugim mehanizmima.

Slika 7.2 prikazuje implementaciju komponente višeg reda (engl. *higher order component*) koja zabranjuje pristup neprijavljenim klijentima na rute koje su djeca ove komponente. Ova komponenta koristi podatke iz prethodno spomenutog konteksta za autentifikaciju.

```

/* eslint-disable react/jsx-props-no-spreading */
import React, { useContext } from "react";
import { Route, Redirect } from "react-router-dom";
import AuthContext from "../context/auth/authContext";
import AppNavigation from "../AppNavigation";
import axios from "axios";

const axiosContext = {};

axios.interceptors.response.use(
  (response) => {
    return response;
  },
  (error) => {
    if (!!error.response && error.response.status === 401) {
      axiosContext.logout("Authentication failed");
    }

    return Promise.reject(error);
  }
);

const PrivateRoute = ({ component: Component, ...rest }) => {
  const authContext = useContext(AuthContext);
  const { isAuthenticated, logout } = authContext;

  axiosContext.logout = logout;

  return (
    <AppNavigation>
      <Route
        {...rest}
        render={
          (props) =>
            isAuthenticated
              ? <Component {...props} />
              : <Redirect to="/login" />
        }
      />
    </AppNavigation>
  );
};

export default PrivateRoute;

```

*Slika 7.2 Komponenta višeg reda za privatne rute*



Slika 7.3 prikazuje način korištenja prethodno spomenute komponente i daje pregled svih ruta u web aplikaciji. Rute za prijavu i registraciju su javne dok su sve ostale privatne.

```
function App() {
  return (
    <AuthState>
      <Router>
        <CssBaseline />
        <Switch>
          <Route exact path="/login" component={Login} />
          <Route exact path="/registration" component={Registration} />
          <PrivateRoute exact path="/" component={Home} />
          <PrivateRoute exact path="/pdf" component={Pdf} />
          <PrivateRoute exact path="/history" component={History} />
          <PrivateRoute exact path="/other-files" component={Binary} />
          <PrivateRoute exact path="/account" component={Account} />
        </Switch>
      </Router>
    </AuthState>
  );
}
```

*Slika 7.3 Primjer korištenja privatnih ruta*

#### 7.1.1 Pregled PDF datoteke

Web aplikacija omogućuje pregled PDF datoteke neposredno prije potpisivanja ili provjere valjanosti potpisa. U ovu svrhu korištena je `react-pdf` biblioteka.

Biblioteka omogućava pregled PDF datoteka koje se sastoje od više stranica te omogućava definiranje prilagođenog ponašanja prosljeđivanjem funkcija i drugih komponenti.

Slika 7.4 prikazuje implementaciju biblioteke u sustavu te korištenje prilagođene funkcionalnosti za stanje učitavanja i stanje bez prikazane PDF datoteke.

```

function NoData() {
  const { previewBox, previewBoxItem } = useStyles();

  return (
    <Box className={previewBox}>
      <Typography
        color="textSecondary"
        align="center"
        component="span"
        variant="h4"
        className={previewBoxItem}
      >
        PDF not selected
      </Typography>
    </Box>
  );
}

function UploadProgress() {
  const { previewBox, previewBoxItem } = useStyles();
  return (
    <Box className={previewBox}>
      <CircularProgress className={previewBoxItem} />
    </Box>
  );
}

export default function PdfViewer(props) {
  const [numPages, setNumPages] = useState(null);
  function onDocumentLoadSuccess({ numPages }) {
    setNumPages(numPages);
  }
  const { pdf } = props;
  return (
    <Document
      file={pdf}
      onLoadSuccess={onDocumentLoadSuccess}
      loading={<UploadProgress />}
      noData={<NoData />}
    >
      {Array.from(new Array(numPages), (_, index) => (
        <Page key={`page_${index + 1}`} pageNumber={index + 1} />
      ))}
    </Document>
  );
}

```

Slika 7.4 Korištenje react-pdf biblioteke

## 7.2 Servis za potpisivanje

Servis za potpisivanje implementiran je koristeći Java programski jezik uparen sa Spring Boot razvojnim okvirom. Servis za potpisivanje implementiran je kao REST API koji podržava funkcionalnosti potpisivanja i provjere valjanosti potpisa.

Servis koristi DSS razvojni okvir za implementaciju AdES potpisa. Konkretno, servis implementira PAdES potpis i XAdES potpis u četiri osnovna profila. XAdES se koristi u odvojenom (engl. *detached*) načinu rada te se njime mogu potpisivati bilo kakve datoteke koje sustav interpretira kao binarnu datoteku.

Ovaj servis na ulazu uzima podatke koje je potrebno potpisati i certifikat kojim se izrađuje potpis ili potpisani dokument te na izlazu daje potpisane podatke odnosno izvještaj o valjanosti potpisa.

Klijent prosljeđuje podatke koje želi potpisati (odnosno provjeriti valjanost potpisa), a servis za upravljanje digitalnim potpisima prosljeđuje osobni certifikat. To znači da je ovaj servis agnostičan prema izvoru podataka i certifikata kojim obavlja potpisivanje te se može koristiti kao zasebna jedinica izvan konteksta ovoga sustava.

Podatci se šalju u obliku Base64 niza znakova što omogućava slanje više binarnih podataka u jednom zahtjevu kao dio JSON objekta. Slika 7.5 prikazuje implementaciju modela zahtjeva za potpisivanje.

```

public class SignatureRequestModel extends BaseFileRequestModel {
    private CertificateModel certificate;
    private SignatureProfile profile;

    public CertificateModel getCertificate() {
        return certificate;
    }

    public void setCertificate(CertificateModel certificate) {
        this.certificate = certificate;
    }

    public SignatureProfile getProfile() {
        return profile;
    }

    public void setProfile(SignatureProfile profile) {
        this.profile = profile;
    }
}

public class BaseFileRequestModel {
    private String fileName;
    private String b64Bytes;

    public String getFileName() {
        return fileName;
    }

    public void setFileName(String fileName) {
        this.fileName = fileName;
    }

    public String getB64Bytes() {
        return b64Bytes;
    }

    public void setB64Bytes(String b64Bytes) {
        this.b64Bytes = b64Bytes;
    }
}

```

*Slika 7.5 Klasa zahtjeva za potpisom*

Certifikat je također kodirani kao Base64 niz znakova pa je potrebno obaviti dekodiranje i učitavanje bajtova neposredno prije izvršavanja operacije potpisivanja.

Razmjena certifikata i pripadajućeg privatnog ključa obavlja se u privatnoj mreži između dva servisa čime se postižu bolje sigurnosne značajke u usporedbi s prosljeđivanjem korisničkog certifikata iz klijentske aplikacije.

Korištenje certifikata na ovaj način znači da sustav ne može implementirati QES, već samo AdES potpise u četiri osnovna profila.

Uz preduvjet posjedovanja kvalificiranog certifikata, postoje dva pristupa kojima je moguće izraditi QES:

1. Korištenje kvalificiranog HSM-a
2. Korištenje kvalificiranog čitača kartica

Oba navedena načina zahtijevaju promjenu servisa za potpisivanje na način da se operacija potpisivanja odvija na sigurnom uređaju.

Korištenje HSM-a je bolje rješenje iz perspektive korisničkog iskustva jer trošak kupnje uređaja snosi davatelj usluge, a korisniku je za potpisivanje potreban pristup mreži s bilo kojeg suvremenog uređaja s mogućnosti pretraživanja Interneta.

S druge strane, davatelj usluge mora uživati povjerenje klijenata sustava da bi ovaj pristup bio valjan. Nepouzdana davatelj može imati pristup korisničkim podacima te ih iskoristiti na maliciozan način.

### **7.3 Servis za upravljanje digitalnim potpisima**

Servis za upravljanje digitalnim potpisima implementiran je koristeći C# programski jezik uparen s .NET 5 razvojnim okvirom. Servis je implementiran kao REST API koji služi kao sučelje klijentskoj aplikaciji.

Servis implementira značajke upravljanja korisničkim računima, korisničkim certifikatima i operacijama potpisivanja i provjera valjanosti potpisa.

Pristup funkcionalnostima servisa ograničen je na prijavljene korisnike korištenjem JWT-a. Slika 7.6 prikazuje metodu čijim se pozivom JWT autentifikacija može dodati u ASP NET cjevovod.

```
public static IServiceCollection AddJwtAuthentication(
    this IServiceCollection services, IConfiguration configuration)
{
    services.Configure<JwtSettings>(configuration.GetSection("Jwt"));
    services.AddScoped<IJwtTokenProvider, JwtTokenProvider>();

    services.AddAuthentication(options =>
    {
        options.DefaultAuthenticateScheme =
            JwtBearerDefaults.AuthenticationScheme;
        options.DefaultChallengeScheme =
            JwtBearerDefaults.AuthenticationScheme;
        options.DefaultScheme = JwtBearerDefaults.AuthenticationScheme;
    })
    .AddJwtBearer(options =>
    {
        options.SaveToken = true;
        options.RequireHttpsMetadata = false;
        options.TokenValidationParameters =
            new TokenValidationParameters
            {
                ValidateIssuer = true,
                ValidateAudience = true,
                ValidAudience = configuration["Jwt:Audience"],
                ValidIssuer = configuration["JWT:Issuer"],
                IssuerSigningKey =
                    new SymmetricSecurityKey(
                        Encoding.UTF8.GetBytes(
                            configuration["JWT:Secret"])),
                ClockSkew = TimeSpan.Zero
            };
    });

    return services;
}
```

*Slika 7.6 Metoda za dodavanje JWT autentifikacije u ASP NET cjevovod*

Ovaj mehanizam korišten je zbog jednostavnosti implementacije i široke prihvaćenosti u modernom razvoju REST API-ja.

Za produkcijsku uporabu ovaj mehanizam može biti ojačan implementacijom 2FA ili MFA mehanizama i korištenjem drugih davatelja usluga autentifikacije poput Google-a ili Facebook-a.

Funkcionalnost potpisivanja moguće je obilježiti politikom koja će onemogućiti korištenje te funkcionalnosti ako korisnik nije prijavljen jednim od povišenih (engl. *elevated*) mehanizama prijave.

Na ovaj način korisnik može koristiti usluge koje nisu kritične prijavom u sustav koristeći korisničko ime i lozinku, dok je za kritične usluge potrebna povišena razina prijave u sustav.

Servis koristi `Entity Framework` biblioteku za upravljanje operacijama vezanim uz bazu podataka. Biblioteka se sastoji od alata za upravljanje bazom podataka i ORM-a.

Ove biblioteke omogućava pisanje koda koji je neovisan o korištenoj bazi podataka. Baza podataka je apstrahirana kroz objekt tipa `DbContext`. Za promjenu baze podataka potrebno je pozvati drugu funkciju prilikom registracije `DbContext` tipa. Slika 7.7 prikazuje način dodavanja registracije za SQL Server.

```
services.AddDbContext<DigitalSignatureManagerDbContext>(options =>
    options.UseSqlServer(
        configuration.GetConnectionString("DefaultConnection"),
        b => b.MigrationsAssembly(
            typeof(DigitalSignatureManagerDbContext).Assembly.FullName)));
```

*Slika 7.7 Registracija servisa baze podataka*

Ovaj servis pohranjuje korisničke certifikate u enkriptirane AES-GCM-256 algoritmom u bazu podataka. Iako nije moguće dobiti status kvalificiranog potpisa na ovaj način, namjera je podizanje razine sigurnosti kroz dostupne kriptografske mehanizme.

Slika 7.8 prikazuje implementaciju metode za pridruživanje certifikata korisničkom računu.

```

private Certificate CreateCertificate(CertificateAssignmentModel model,
    int userId) =>
    new()
    {
        B64Certificate = CryptographyHelper.Encrypt(
            model.B64Certificate, _encryptionKey),
        B64Password = CryptographyHelper.Encrypt(
            model.CertificatePassword, _encryptionKey),
        IsRevoked = false,
        OwnerId = userId
    };

```

*Slika 7.8 Metoda za dodjeljivanje certifikata korisničkom računu*

Slika 7.9 prikazuje implementaciju pomoćne metode za enkripciju niza bajtova koja se koristi prilikom pridruživanja certifikata.

```

private static byte[] EncryptBytesAesGcm(byte[] toEncrypt,
    byte[] key, byte[] associatedData = null)
{
    using var rng = new RNGCryptoServiceProvider();

    var tag = new byte[TagBytes];
    var nonce = new byte[NonceBytes];
    var cipherText = new byte[toEncrypt.Length];

    rng.GetBytes(tag);
    rng.GetBytes(nonce);

    using var cipher = new AesGcm(key);
    cipher.Encrypt(nonce, toEncrypt, cipherText, tag, associatedData);

    return Concat(tag, Concat(nonce, cipherText));
}

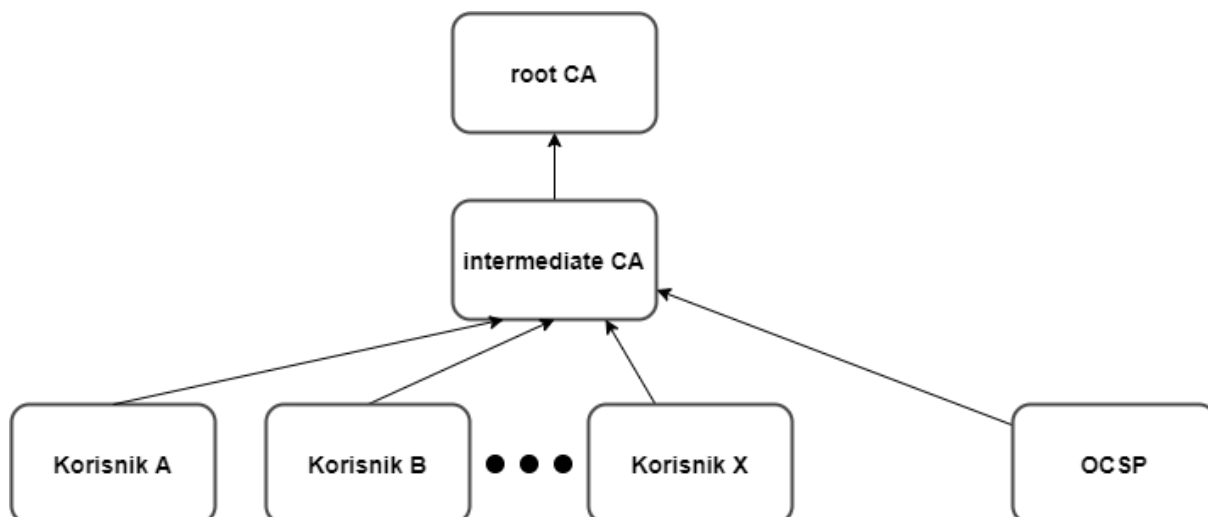
```

*Slika 7.9 Pomoćna metoda za enkripciju*

## 7.4 Testni PKI

PKI korišten u ovom sustavu generiran je alatom OpenSSL. Kroz ovaj alat implementiran je jednostavan PKI koji zadovoljava potrebe sustava. Slika 7.10 prikazuje dijagram implementiranog PKI.





*Slika 7.10 Testni PKI*

Matično certifikacijsko tijelo i posredno certifikacijsko tijelo održavaju liste opoziva, a posredno certifikacijsko tijelo podržava i OCSP upite. Posredno certifikacijsko tijelo izdaje korisničke certifikate.

Slika 7.11 prikazuje primjer certifikata izdanog korisniku Ivanu Iviću. Podatci u primjeru su reducirani zbog preglednosti.

Certificate:

Data:

```
Version: 3 (0x2)
Serial Number: 4099 (0x1003)
Signature Algorithm: sha256WithRSAEncryption
Issuer: C = HR, ST = Splitsko-
Dalmatinska, O = UNIST, OU = "UNIST Certificate Authority ", CN = www.unist
.hr Intermediate CA
Validity
    Not Before: Jul 24 11:03:18 2021 GMT
    Not After : Jul 24 11:03:18 2022 GMT
Subject: C = HR, ST = Splitsko-
Dalmatinska, L = Split, O = UNIST, OU = UNIST, CN = IvanIvic.unist
Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
    RSA Public-Key: (2048 bit)
    Modulus:
        00:...:ed
    Exponent: 65537 (0x10001)
X509v3 extensions:
    X509v3 Basic Constraints:
        CA:FALSE
    Netscape Cert Type:
        SSL Client, S/MIME
    Netscape Comment:
        OpenSSL Generated Client Certificate
    X509v3 Subject Key Identifier:
        78:...:FC
    X509v3 Authority Key Identifier:
        keyid:F2:...:BB
    X509v3 Key Usage: critical
        Digital Signature, Non Repudiation, Key Encipherment
    X509v3 Extended Key Usage:
        TLS Web Client Authentication, E-mail Protection
    Authority Information Access:
        OCSP - URI:http://ocsp.example.com
    X509v3 CRL Distribution Points:
        Full Name:
            URI:http://crl.example.com/intermediate.crl.pem
Signature Algorithm: sha256WithRSAEncryption
5b:...:bf:b9
-----BEGIN CERTIFICATE-----
MIIFtzCCA5+gAwIBAgICEAMwDQYJKoZIhvcNAQELBQAwYoxCzAJBgNVBAYTAkhS
...
RdBes8MMpF3dP36DMeoJgJmE57E27PPNcr+5
-----END CERTIFICATE-----
```

*Slika 7.11 Skraćeni primjer korisničkog certifikata*

Upravljanje PKI je automatizirano kroz bash skripte i openSSL konfiguracijske datoteke. Konfiguracijska datoteka definira parametre PKI i profile koji se mogu koristiti prilikom izdavanja certifikatu.

Slika 7.12 prikazuje implementaciju skripte za izdavanje korisničkog certifikata.

```
#!/bin/sh
#First generate the key for the client

openssl genrsa \
    -out /root/ca/intermediate/private/$1.key.pem 2048 &>/dev/null
chmod 400 /root/ca/intermediate/private/$1.key.pem

#Then create the certificate signing request
openssl req -config /root/ca/intermediate/openssl.cnf \
    -key /root/ca/intermediate/private/$1.key.pem \
    -new -sha256 -out /root/ca/intermediate/csr/$1.csr.pem \
    -subj "/C=HR/ST=Splitsko-
Dalmatinska/L=Split/O=UNIST/OU=UNIST/CN=$1.unist/EMAIL=$1@unist.hr" &>/dev/
null
#Now sign it with the intermediate CA
echo -e "y\ny\n" | openssl ca -config /root/ca/intermediate/openssl.cnf \
    -extensions usr_cert -days 365 -notext -md sha256 \
    -in /root/ca/intermediate/csr/$1.csr.pem \
    -out /root/ca/intermediate/certs/$1.cert.pem &>/dev/null

chmod 444 /root/ca/intermediate/certs/$1.cert.pem
echo "$(cat /root/ca/intermediate/certs/$1.cert.pem)"
```

*Slika 7.12 Bash skripta za izdavanje korisničkog certifikata*

U prvom koraku generira se privatni ključ korisničkog certifikata. Zatim se stvara zahtjev za potpisivanjem certifikata koji sadrži informacije o subjektu i privatni ključ. Certifikacijsko tijelo na temelju zahtjeva za potpisivanjem izdaje certifikat. Skripta se poziva s jednim argumentom koji definira ime subjekta i definira naziv izlaznih datoteka.

Prilikom izdavanja certifikata koristi se `usr_cert` proširenje definirano u konfiguracijskoj datoteci. Slika 7.13 prikazuje konfiguracijsku openSSL datoteku za posredno certifikacijsko tijelo.

```

[ ca ]
default_ca = CA_default
[ CA_default ]
# Directory and file locations.
dir                = root/ca/intermediate
certs              = $dir/certs
crl_dir            = $dir/crl
new_certs_dir      = $dir/newcerts
database           = $dir/index.txt
serial             = $dir/serial
RANDFILE           = $dir/private/.rand
# The root key and root certificate.
private_key        = $dir/private/intermediate.key.pem
certificate         = $dir/certs/intermediate.cert.pem
# For certificate revocation lists.
crlnumber          = $dir/crlnumber
crl                = $dir/crl/intermediate.crl.pem
crl_extensions     = crl_ext
default_crl_days   = 30
# SHA-1 is deprecated, so use SHA-2 instead.
default_md         = sha256
name_opt           = ca_default
cert_opt           = ca_default
default_days       = 375
preserve           = no
policy             = policy_loose
[ policy_strict ]
# The root CA should only sign intermediate certificates that match.
# See the POLICY FORMAT section of `man ca`.
countryName        = match
stateOrProvinceName = match
organizationName    = match
organizationalUnitName = optional
commonName          = supplied
emailAddress         = optional
[ usr_cert ]
# Extensions for client certificates (`man x509v3_config`).
basicConstraints = CA:FALSE
nsCertType = client, email
nsComment = "OpenSSL Generated Client Certificate"
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer
keyUsage = critical, nonRepudiation, digitalSignature, keyEncipherment
extendedKeyUsage = clientAuth, emailProtection
authorityInfoAccess = OCSP;URI:http://ocsp.example.com
crlDistributionPoints = URI:http://crl.example.com/intermediate.crl.pem

```

Slika 7.13 Skraćeni prikaz openSSL konfiguracijske datoteke

## 7.5 CRL poslužitelj

CRL poslužitelj implementiran je kao HTTP poslužitelj uz pomoć nginx web poslužitelja. On poslužuje prethodno generirane i potpisane liste opoziva glavnog i posrednog certifikacijskog tijela.

Koristeći Docker i nginx konfiguracijsku datoteku, moguće je ostvariti CRL poslužitelj u svega nekoliko linija koda. Slika 7.14 prikazuje nginx konfiguraciju.

```
server{
    listen 80;
    server_name crl.example.com;

    location / {
        root /srv/crl;
        autoindex on;
    }
    location /intermediate.crl.pem {
        alias /srv/crl/intermediate/intermediate.crl.pem;
    }
    location /ca.crl.pem {
        alias /srv/crl/ca/ca.crl.pem;
    }
}
```

*Slika 7.14 Konfiguracijska nginx datoteka*

Slika 7.15 prikazuje implementaciju bash skripte za pokretanje CRL poslužitelja koristeći Docker.

```
sh ./build.sh

pushd ../

echo "**** Running CRL provider on localhost:80"
docker run --rm -d \
--name crl \
-p 80:80 \
-v "$(pwd)"/ExamplePKI/root/ca/crl:/srv/crl/ca \
-v "$(pwd)"/ExamplePKI/root/ca/intermediate/crl:/srv/crl/intermediate \
crl-provider
```

*Slika 7.15 Skripta za pokretanje CRL poslužitelja koristeći Docker*

Liste opoziva se obično osvježavaju u regularnim intervalima pa je za produkcijsku upotrebu potrebno uzeti u obzir tu značajku.

## 7.6 OCSP poslužitelj

OCSP poslužitelj implementiran je kao HTTP poslužitelj uz pomoć openssl alata. OpenSSL implementira funkcionalnost OCSP poslužitelja. Na temelju prethodno generirane CA baze podataka i certifikata poslužitelja ovaj servis daje potpisani odgovor na OCSP upite.

Koristeći Docker i openssl alat, moguće je ostvariti OCSP poslužitelj u svega nekoliko linija koda. Slika 7.16 prikazuje bash skriptu za pokretanje OCSP poslužitelja.

```
#!/bin/sh
#This entrypoint is responsible for leaving the OSCP running
#to accept requests
openssl ocsp -port 0.0.0.0:80 -text -sha256 \
    -index /root/ca/intermediate/index.txt \
    -CA /root/ca/intermediate/certs/ca-chain.cert.pem \
    -rkey /root/ca/intermediate/private/ocsp.key.pem \
    -rsigner /root/ca/intermediate/certs/ocsp.cert.pem
```

*Slika 7.16 Bash skripta za pokretanje OCSP poslužitelja*

Slika 7.17 prikazuje implementaciju bash skripte za pokretanje OCSP poslužitelja koristeći Docker.

```
sh ./build.sh

pushd ../

echo "**** Running OCSP provider on localhost:2560"
docker run --rm -d \
    --name ocsp \
    -p 2560:2560 \
    -v "$(pwd)"/ExamplePKI/root/:/root \
    ocsp-provider
```

*Slika 7.17 Skripta za pokretanje OCSP poslužitelja koristeći Docker*

## 8 Zaključak

Trend modernizacije usluga i procesa postavlja niz sigurnosnih prepreka koje je potrebno otkloniti za uspješnu tranziciju u digitalno doba. Nadležna tijela, ustanove i organizacije definiraju pravne okvire unutar kojih se osjetljive usluge i procesi mogu odvijati.

Fizička razmjena dokumenata i ostalih podataka je fundamentalni proces na kojem se zasnivaju brojne usluge modernog društva. Prilikom digitalizacije ovog procesa, potrebno je riješiti problem potpisivanja s tehničkog i pravnog stajališta.

Digitalni potpisi tehnički su realizirani kroz algoritme asimetrične kriptografije, a jedan od pravnih okvira relevantnih za područje EU definiran je eIDAS uredbom. Uredba definira zahtjeve koje digitalni potpisi moraju ispunjavati kako bi imali pravnu osnovicu.

Najviša razina uredbom definiranih potpisa ekvivalentna je vlastoručnom potpisu u pravnom smislu. Iz sigurnosne perspektive, ovakav potpis nosi veći broj sigurnosnih značajki od usporedivog vlastoručnog potpisa.

Uredba je implementirana kroz javne standarde, a standardi su implementirani kroz brojna softverska rješenja koja mogu biti otvorenog i zatvorenog tipa.

Moguća je implementacija vlastitih rješenja koristeći otvorene razvojne okvire. Kroz njih je moguće izraditi sustav usklađen s definiranim pravnim okvirima koji je prilagođen za potrebe pojedine organizacije.

Implementacija ovakvih sustava složen je postupak s velikim brojem pokretnih dijelova koje je potrebno uskladiti za uspješan rad sustava. Implementacijske pogreške i propusti u ovakvim sustavima imaju vrlo visoku cijenu zbog čega je potrebno dobro razumjeti problematiku i osnovati specijaliziran tim koji će se baviti izradom i održavanjem ovakvog sustava.

## Literatura

- [1] Paul, E.: „What are digital signatures: how it works, benefits, objectives, concept“, s Interneta, <https://www.emptrust.com/blog/benefits-of-using-digital-signatures>, 18.8.2021.
- [2] Mark, T.: „Difference between digital signature and electronic signature“, s Interneta, <http://www.differencebetween.net/technology/difference-between-digital-signature-and-electronic-signature/>, 18.8.2021.
- [3] Čagalj, M.: „Asymmetric key cryptography“, ak.god. 2020./2021.
- [4] CISA: Security Tip (ST04-018), Understanding Digital Signatures, 24.8.2020
- [5] Vacca, Jhn R.: „Public Key Infrastructure: Building Trusted Applications and Web Services“, 2004
- [6] SSH.com: „What is PKI (Public Key Infrastructure)?“, s Interneta, <https://www.ssh.com/academy/pki>, 19.8.2021.
- [7] Cooper, et al.: RFC5280 Internet X.509 Public Key Infrastructure Certificate
- [8] Naziridis, N.: „Page Load Optimization: OCSP Stapling“, s Interneta, <https://www.ssl.com/article/page-load-optimization-ocsp-stapling/>, 8.2.2019.
- and Certificate Revocation List (CRL) Profile, svibanj 2008
- [9] go.eIDAS: „About“, s Interneta, <https://go.eid.as/#about>, 17.8.2021.
- [10] Službeni list Europske unije, L 257/73, 28.8.2014.
- [11] European commission: „Trusted List Croatia“, s Interneta, <https://webgate.ec.europa.eu/tl-browser/#/tl/HR>, 17.8.2021.
- [12] CEF digital: „Digital Signature Service Documentation v5.8, 7.2.2021.“, s Interneta, <https://ec.europa.eu/cefdigital/DSS/webapp-demo/doc/dss-documentation.html>, 20.8.2021.
- [13] W3C: „XML Advanced Electronic Signatures, 20.2.2003“, s Interneta, <https://www.w3.org/TR/XAdES/>, 20.8.2021.
- [14] ISO 32000-1:2008: „Document management – Portable document format – Part 1: PDF 1.7“, srpanj 2008.
- [15] ETSI TS 102 778-1: „Electronic Signatures and Infrastructures; PDF Advanced Electronic Signature Profiles; Part 1: PAdES Overview – a framework document for PAdES“, V.1.1.1, srpanj 2009.
- [16] Housley, R et al: „RFC5642 Cryptographic Message Syntax (CMS)“, rujan 2009.
- [17] ETSI TS 119 312: „Electronic Signatures and Infrastructures (ESI); Cryptographic Suites“, V1.2.1, svibanj 2017.



- [18] SOG-IS Crypto Working Group: „SOG-IS Crypto Evaluation Scheme Agreed Cryptographic Mechanisms“, Version 1.2, siječanj 2020.
- [19] CEF Digital: „Digital Signature Service – DSS“, s Interneta, <https://ec.europa.eu/cefdigital/wiki/display/CEFDIGITAL/Digital+Signature+Service++DSS>, 21.8.2021.
- [20] Barette, O.: „Qualified electronic signature (QES) validation algorithm“, V1.03, 9.9.2019.
- [21] Docker.com: „Overview of Docker Compose“, s Interneta, <https://docs.docker.com/compose/>, 24.8.2021.
- [22] ReactJS: „Context“, s Interneta, <https://reactjs.org/docs/context.html>, 24.8.2021.

## **Popis oznaka i kratica**

RSA - Rivest–Shamir–Adleman

DSA – Digital Signature Algorithm

ECDSA – Elliptic Curve Digital Signature Algorithm

EdDSA – Edwards-curve Digital Signature Algorithm

PKI – Public Key Infrastructure

CA – Certificate Authority

RA – Registration Authority

CRL – Certificate Revocation List

OCSP – Online Certificate Status Protocol

TSP – Trusted Source Provider

QES – Qualified Electronic Signature

QSCD – Qualified signature Creation Device

AdES – Advanced Electronic Signature

ETSI – European Telecommunications Standards Institute

XAdES – XML AdES

PAdES – PDF AdES

CAdES – CMS AdES

CMS – Cryptographic Message Syntax

DER – Distinguished Encoding Rules

BER – Basic Encoding Rules

ASN.1 – Abstract Syntax Notation One

DSS – Digital Signature Services

HSM – Hardware Security Module

REST – REpresentational State Transfer

2FA – Two-Factor Authentication

MFA – Multi-Factor Authentication

ORM – Object-Relational Mapping

## **Sažetak**

U okviru rada obrađen je postupak digitalnog potpisivanja podataka s pravnim učincima na razini EU. Opisani su osnovni kriptografski koncepti na kojima se temelje digitalni potpisi te uredba kojom je definirana njihova pravna valjanost. Opisan je DSS razvojni okvir kroz koji je implementiran sustav za digitalno potpisivanje. Sagledana je arhitektura implementiranog sustava te su opisani implementacijski detalji neophodni za njegovo razumijevanje. Kroz rad su istražene odluke prilikom dizajniranja i ideje za potencijalna proširenja implementiranog sustava.

**Naslov:** Sustav za elektroničko potpisivanje u skladu s EU eIDAS regulativom

**Ključne riječi:** Digitalni potpisi, AdES, QES, eIDAS, PKI, DSS

## Summary

Within this thesis, the procedure of digital data signatures with legal effects in the EU region is discussed. Elementary cryptography concepts which serve as the foundation of digital signatures are described in addition to exploring legal regulations that define their legal validity. The DSS development framework through which the digital signing system is implemented is described. The architecture of the implemented system is considered and the implementation details necessary for its understanding are outlined. Throughout this thesis design choices and alternatives are explored along with ideas for potential extensions to the implemented system.

**Title:** A system for electronic signature creation and validation in line with the European eIDAS regulation

**Keywords:** Digital signatures, AdES, QES, eIDAS, PKI, DSS

## Prilozi

### Kazalo slika

Slika 2.1 Dijagram potpisa i verifikacije podataka (izvor: <a href="https://www.docusign.com/how-it-works/electronic-signature/digital-signature/digital-signature-faq">https://www.docusign.com/how-it-works/electronic-signature/digital-signature/digital-signature-faq</a> , preuzeto 18.8.2021.) .....	3
Slika 2.2 Digitalni certifikat web stranice FESB-a .....	6
Slika 2.3 Lanac certifikata.....	7
Slika 3.1 Vremenska traka razvoja eIDAS uredbe (izvor: <a href="https://www.eid.as/">https://www.eid.as/</a> , preuzeto 17.8.2021.).....	9
Slika 3.2 Jedinstveno elektroničko EU tržište (izvor: <a href="https://en.wikipedia.org/wiki/EIDAS#/media/File:E-SENS_architecture.jpg">https://en.wikipedia.org/wiki/EIDAS#/media/File:E-SENS_architecture.jpg</a> , preuzeto 17.8.2021.).....	10
Slika 4.1 XAdES dijagram .....	15
Slika 4.2 XAdES-Baseline-B XML struktura .....	17
Slika 4.3 ISO32000-1 PDF potpis.....	20
Slika 4.4 Rječnik potpisa.....	21
Slika 4.5 Identifikator vrste sadržaja.....	23
Slika 4.6 Definicija ContentInfo tipa .....	23
Slika 4.7 Osnovni CMS tip-vrijednost format .....	24
Slika 4.8 CMS Signed-Data tip formata.....	24
Slika 5.1 Stvaranje DSS dokumenta u memoriji.....	29
Slika 5.2 Stvaranje PKCS12 objekta za potpisivanje.....	31
Slika 5.3 Potpisivanje PDF dokumenta.....	32
Slika 5.4 Primjer XAdES-Baseline-B potpisa.....	34
Slika 5.5 Pojednostavljeni process provjere valjanosti potpisa u DSS-u (izvor: <a href="https://ec.europa.eu/cefdigital/DSS/webapp-demo/doc/dss-documentation.html#_validation_process">https://ec.europa.eu/cefdigital/DSS/webapp-demo/doc/dss-documentation.html#_validation_process</a> , preuzeto 22.8.2021.).....	36
Slika 5.6 Provjera valjanosti potpisa binarne datoteke .....	37
Slika 5.7 Stvaranje CertificateVerifier objekta .....	37
Slika 5.8 Primjer jednostavnog izvještaja .....	39
Slika 5.9 Dodavanje privatnog CA u pouzdane izvore .....	41
Slika 5.10 Stvaranje objekata za dohvaćanje informacija o opozivu .....	43
Slika 6.1 Arhitektura sustava .....	44

Slika 6.2 Prijava u sustav .....	45
Slika 6.3 Registracija.....	45
Slika 6.4 Naslovna stranica .....	46
Slika 6.5 Potpisivanje i verifikacija PDF dokumenata.....	46
Slika 6.6 Potpisivanje i verifikacija binarnih datoteka.....	47
Slika 6.7 Povijest.....	47
Slika 6.8 Upravljanje osobnim podacima .....	48
Slika 6.9 Relacijski model baze podataka .....	50
Slika 7.1 Compose datoteka sustava .....	52
Slika 7.2 Komponenta višeg reda za privatne rute.....	54
Slika 7.3 Primjer korištenja privatnih ruta .....	55
Slika 7.4 Korištenje react-pdf biblioteke.....	56
Slika 7.5 Klasa zahtjeva za potpisom.....	58
Slika 7.6 Metoda za dodavanje JWT autentikacije u ASP NET cjevovod .....	60
Slika 7.7 Registracija servisa baze podataka.....	61
Slika 7.8 Metoda za dodjeljivanje certifikata korisničkom računu .....	62
Slika 7.9 Pomoćna metoda za enkripciju .....	62
Slika 7.10 Testni PKI .....	63
Slika 7.11 Skraćeni primjer korisničkog certifikata.....	64
Slika 7.12 Bash skripta za izdavanje korisničkog certifikata.....	65
Slika 7.13 Skraćeni prikaz openssl konfiguracijske datoteke.....	66
Slika 7.14 Konfiguracijska nginx datoteka .....	67
Slika 7.15 Skripta za pokretanje CRL poslužitelja koristeći Docker .....	67
Slika 7.16 Bash skripta za pokretanje OCSP poslužitelja.....	68
Slika 7.17 Skripta za pokretanje OCSP poslužitelja koristeći Docker.....	68