TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Raul Metsma 210074IADB

Web eID toe lisamine Safari veebilehitsejale

Bakalaureusetöö

Juhendaja: Meelis Antoi
Magistrikraad

Tallinn 2021

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Raul Metsma

17.05.2021

Annotatsioon

Käesoleva bakalaureusetöö eesmärgiks on luua töötav Web eID laiendus Safari veebislehitsejale, mis lahendab praeguse Open eID laienduse puudujääke ja parendab tunduvalt autentimise tuge. Laiendus on avatud lähtekoodiga ja avaldatud GitHub koodihoidlas ning iga arendaja saab tulevikus oma panuse anda.

Töö rõhk on Apple uue tehnoloogia *Safari Web Extensions* analüüsil ja selle ühilduvuses *Chrome Extensions* Web eID lahenduse integreerimisel. Antud bakalaureusetöös on välja toodud juhend kuidas hankida laienduse kood ja ehitada rakendus enda arvutis. Lisaks on lisatud juhised tarkvara silumiseks ja tõrgete otsimiseks.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 35 leheküljel, 5 peatükki, 15 joonist, 0 tabelit.

Abstract

Adding Web eID Support for Safari Web Browser

The aim of current thesis is to create working Web eID extension for Safari browser, which would resolve current Open eID extension shortcommings and improves considerably authentication support. Extension is open source and published in GitHub repository and every developer can participate to improve the solutuion.

The emphasis in the work is on the new Apple tehnology Safari Web Extensions analyze and its compatibility with Chrome Extensions Web eID solution integration. Current thesis also contains instructions how to get source code of the extension and build it on developer. Additionaly there are add instructions how to debug software and find solutions for errors.

The thesis is in Estonian and contains 35 pages of text, 5 chapters, 15 figures, 0 tables.

Lühendite ja mõistete sõnastik

APDU	*Application Protocol Data Unit*, rakendusprotokolli andmeühik[1]
C++	programmeerimiskeel
eID	elektrooniline identiteet
PC/SC	*Personal Computer/Smart Card*, kiipkaardiga suhtlemise liides
PKCS #11	avaliku võtme standard mis kirjeldab programmliidest kuidas manipuleerida krüptovõtmeid
RIA	Riigi Infosüsteemi Amet
TLS	*Transport Layer Security*, transpordikihi turbeprotokoll[1]
TLS CCA	*TLS Client Certificate Authentication*, transpordikihi turbeprotokolli kasutaja sertifikaadiga autentimine
X.509	avaliku võtme sertifikaadi standard

[1] http://www.vallaste.ee/

Sisukord

Jooniste loetelu

1 Sissejuhatus

Kiibipõhise eID toe tagamine pidevalt muutuvas keskkonnas on tõsine väljakutse. Kasutajakogemus on väga halb või olematu ning seetõttu on ID-kaardi kasutamine kohati lausa võimatu. Veebilehitseja tootjad võtavad pidevalt kasutusele uusi meetmeid, et maandada riske internetis leiduvate ohtude eest. See seab ka uusi väljakutseid kuidas hoida toimivana eID lahendusi. Kuna autentimine toimub Windowsi ja macOS keskkonnas enamlevinud veebilehitsejatega operatsioonisüsteemi alusteekidega esineb kohati veidraid ja ootamatuid käitumisi. Ka tekitab fustratsiooni kaarditootjate draiverite erinev kvaliteeti tase.

Projekti käigus analüüsiti võimalusi ja prooviti lahendada autentimine veebilehitseja laienduse liideses mis annab rohkem kontrolli protsessi üle ja on paindlikum. Lisaks implementeeriti lähiriikide (Eesti, Läti, Leedu, Soome) kaartide tugi APDU tasemel mis annab kontrolli kaartidega suhtlemisel, ühtlasema kvaliteedi taseme ja vähendab välistest komponentidest tulenevaid tõrkeid.

Ennekõige annab see lõppkasutajale lihtsama paigalduse ja vähendab erinevatest liidestest tulenevaid ebastabiilsuseid. Edukast autentimine viib ka eduka allkirjastamiseni, kui just allkirjastamise PIN lukus ei ole. Lisaks vähendab see ka RIA kasutajatoe halduskoormust.

2 Kasutusel olevate veebilehitseja laienduste võimalused ja puudused

Projekti raames analüüsiti ja kaardistati praeguse veebilehitseja laienduste võimalusi ja puudusi, sealhulgas ka veebilehitsejate TLS [1] liidese autentimise tuge. Analüüsi tulemusena koostati Web eID [2] veebilehitseja laiendusele järgmised funktsionaalsed nõuded:

- Võimaldab autentimist ja digiallkirjastamist lähiriikide kaartidega: Eesti, Läti, Leedu, Soome.
- Toetab levinumaid operatsioonisüsteemi versioone (Windows, macOS, Linux) ja veebilehitsejad (Chrome, Firefox, Safari, Edge).
- Toetab lihtsat paigaldust ja kohest käivitust ilma, et oleks vajalik operatsioonisüsteemi taaskäivitamine. Web eID töölauarakendus vajab eraldi paigaldust veebilehitseja laiendusest kuna seda ei saa levitada läbi veebilehitseja laienduste poe.
- Ei vaja toimimiseks kolmandaid tarkvara komponente peale operatsioonisüsteemi enda PC/SC liidese ja kaardilugeja draiveri.
- Laiendus on tõrkekindel ning toetab kaardi eemaldamist ja sisestamist kaardilugejast igas autentimis- ja allkirjastamisprotsessi etapis.
- Ei sega muu eID tarkvara kasutamist veebilehitsejas ega operatsioonisüsteemis ning Web eID veebilehitseja laiendus töötab paralleelselt Open eID [3] veebilehitseja laiendusega.
- Väljalogimisel puudub vajadus veebilehitsejat sulgeda ja kasutamine ei nõua veebiserveri erikonfiguratsiooni.
- Web eID veebilehitseja laiendus peab igal autentimisel uuesti küsima PIN1.

- Uue lahenduse allkirjastamise komponent on tagasiühilduv olemasoleva hwcrypto.js [4] teegiga.

- Edukas autentimine tähendab edukat allkirjastamist kui just PIN lukus pole.

3 Web eID veebilehitseja laienduse arhitektuur

Web eID projekt on oma olemuselt väga mahukas, mistõttu on käesolev lõputöö keskendunud *Safari Web Extensions* veebilehitseja laienduse lahendusele, mis on täies ulatuses töö autori looming. *Safari Web Extensions* veebilehitseja laiendusele kehtisid projekti skoobist tulenevalt mitmed erinevad nõuded. Nõuetest annab täpsema ülevaate käesolevast peatükist.

Arvestades kasutajaskonna erinevat arvutioskust on väga oluline, et tarkvara oleks lihtsalt paigaldatav. Sellest tulenevalt peab tarkvara omama võimalikult vähe väliseid komponente, mis võiksid tekitada ettenägematuid takistusi. Sellest lähtudes otsustati peamiseks programmeerimise keeleks valida C++. Viimane on piisavalt edasiarenenud C++17 standardi [5] versiooniga ning tunduvalt mugavam käsitleda. C++ keele valiku kasuks tuli ka nõue, et rakendus oleks kasutajale lihtsalt paigaldatav. C++ eelis Pythoni ja Java ees on tõik, et C++ ei vaja käivitamiseks lisa komponente ja on sarnaselt platvormist sõltumatu. C programmeerimis keele kasutuse vastu rääkis ka meeskonna otsus kasutada Qt raamistikku.

Kuna rakendus peab toimima Windows, macOS ja Linux platvormidel otsustati kasutada Qt [6] mitme platvormi raamistikku. Qt raamistik võimaldab ehitada keerulisemaid rakendusi mitmele platvormile kirjutades koodi ainult ühe korra. Qt raamistikul on ka kaasuvad moodulid, mis osutusid raamistiku valikul määravaks. Näiteks võrgu toe moodulis saadaval olev vajalik liides X.509 [7] sertifikaatide lugemiseks, ühiktestide kirjutamiseks olemasolev test moodul ning tööriistad mitmekeelse rakenduse loomiseks ja ka nägemispuudega inimeste juhendamiseks.

Kuna tegemist oli mitme platvormi projektiga siis meeskond välistas erinevate ehitamise süsteemide kasutamise, näiteks Visual Studio [8] ja Xcode [9]. Leiti, et sobivaim on CMake [10] ehitamise süsteem kuna see täitis kõik projekti ootused. Qt raamistikul on heal tasemel CMake tugi ja alates versioonist 6 kasutatakse CMake Qt [11] raamistiku ehitamiseks. CMake võimaldab ka genereerida Visual Studio ja Xcode projekte juhul kui arendajal on omad integreeritud programmeerimiskeskkonna eelistused.

Analüüsides praeguse veebilehitseja laienduste arhitektuuri puudusi leiti, et üks kitsaskoht kiipkaartidega suhtlemisel on draiverid. Windows operatsioonisüsteemil on kasutusel Minidriver [12], teistel platvormidel aga PKCS #11 [13] moodul. Moodulite kvaliteet on riigiti erinev ja sõltub suuresti kaardi tootjast. Lisaks on moodulitel ka teatud piiranguid. Näiteks Windowsi Minidriver, mis ei võimalda kasutajale teada anda enne toimingut kas PIN on lukus. Otsustati, et juhul kui oli olemas kaardi dokumentatsioon siis kasutatakse kiipkaardiga suhtlemiseks APDU käske üle PC/SC liidese. Sellega sai jätta kõrvale nii operatsioonisüsteemi krüptoteeke ja ka mitte kasutada draivereid. PC/SC liidese kasutamisel on eeliseks ka teave kaardi staatuse kohta. See võimaldab saada kohest teavet, kas ID-kaart on eemaldatud või on mõni kiipkaart lugejasse lisatud.

Kasutajatoe tagasisidest tulenevalt osutus problemaatiliseks ka autentimise toe lahendus. Enamus veebilehitsejaid kasutavad operatsioonisüsteemi krüptoteeke, erandiks on siin Firefox, mis kasutab PKCS #11 liidest. Kuna kõik veebilehitsejad rakendavad autentimise tuge erinevalt on ka kasutajakogemus erinev. Väga suur hulk probleeme ilmneb just turvalise TLS ühenduse loomisel. Veebilehitsejate tootjate poolne rutakus lõhub sageli ära igapäevakasutajale juba harjumuspäraseks saanud ID-kaardi kasutamise toiminguid (Näiteks [1] ja [2]). Kaardistati ka järgmise põlvkonna TLS 1.3 [14] standardi kitsaskohad[3], mis ajendasid ehitama autentimise lahendust veebilehitseja laienduse liideses.

Autentimise lahenduse ehitamine veebilehitseja laienduse liideses, et saavutada samasugune turvatase kui TLS CCA standardiga, oli meeskonnale aga suurem väljakutse. Aluseks võeti OpenID Connect [15] standard ja lisati täiendused. Kuna veebilehitsejalt läbi laienduste liideste saadav info kanali parameetrite kohta on sageli ebapiisav, siis ei ole võimalik saavutada samaväärset turvataset TLS CCA-ga. Ehitatud lahendusele telliti ka Cybernetica turvaanalüüs [16].

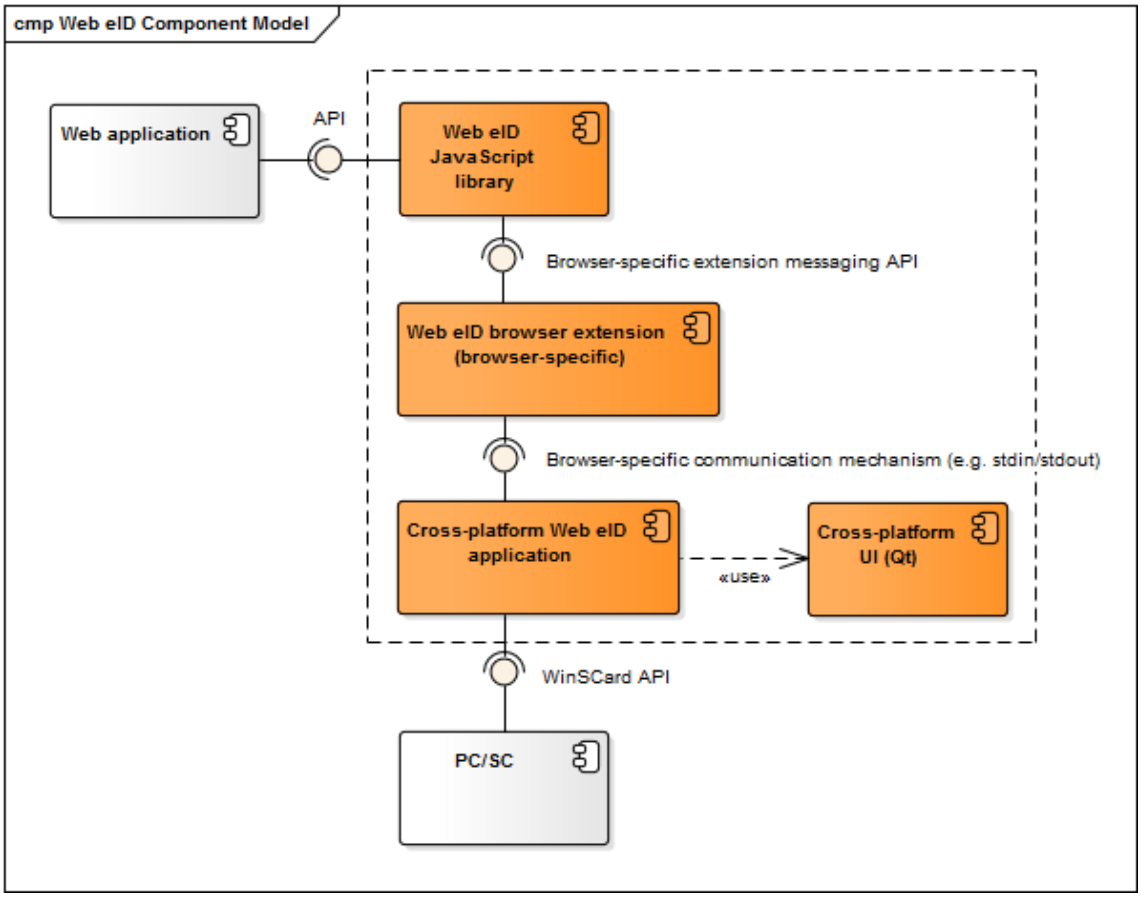
Web eID koosneb kolmest peamisest komponendist, mis on mudelina välja toodud joonisel 1. Kõige lähedasem komponent riistvarale on töölauarakendus (Joonis 1 -

[1] https://bugs.chromium.org/p/chromium/issues/detail?id=880835

[2] https://digi.geenius.ee/rubriik/uudis/firefoxi-brauseris-ei-saa-parst-viimast-uuendust-id-kaarti-kasutada-ria-tegeleb-probleemiga/

[3] https://www.id.ee/artikkel/tls-1-3-standardi-kasutuselevotmine-veebiserveris/

komponent “Cross-platform Web eID application”), mis on omakorda jagatud mitmeks alamkihiks, et neid saaks taaskasutada:



Joonis 1. Web eID komponentide mudel [17]

- libpcsc-cpp: teek mis lihtsustab PC/SC liidese kasutamist. Kuna PC/SC rakendusliides on kirjutatud C programmeerimise keeles, võimaldab antud teek selle korduvkasutamise eripärasid vältida. Näiteks saab libpcsc-cpp teek aru APDU vastusest, kui see viitab, et kaardilt on võimalik veel andmeid lugeda ning teostab vastavad käsud. Rakenduse kiht saab libpcsc-cpp teegi käest kogu andmehulga. Ka peidab see ära sõrmistikuga kiipkaardi lugeja kasutamise keerukuse ning ühtlustab selle kasutamist ülemistes liidestes.

- libpcsc-mock: teek mis emuleerib PC/SC liidest ja võimaldab taasesitada kaardi APDU käske ja vastuseid. See võimaldab jooksutada ühikteste pidevintegratsioonis ilma, et riistvara (kiipkaart ja kiipkaardi lugeja) peaks olema ühendatud arvuti külge. Sedasi saab juba varakult teada kas mõni koodi põhiharusse liidetav muudatus on vigane.

- libelectronic-id: teek, mis omab teavet kuidas suhelda erinevate kaartidega. Hetkel on selles teegis APDU tasemel implementeeritud Eesti Gemalto versioon 3.5.8 kaartide ja uute Eesti IDEMIA kaartide tugi; nii vana kui ka uue põlvkonna Läti IDEMIA kaartide tugi ja Soome kolmanda generatsiooni kaartide tugi. Teegil on ka olemas võimalus kasutada draiverite liidest juhuks kui kiipkaardi dokumentatsioon ei ole avalik.

- web-eid-app: töölauarakendus mis on omakorda jaotatud käivitusrakenduseks, kontrolleriks ja kasutajaliideseks. Rakendus on sündmuste põhine, näiteks kiipkaardi lisamisel kasutajaliidese sisu muutub ja kasutaja ei pea alustama protsessi algusest. Lisaks kasutamise lihtsusele võimaldab rakendus ka vaegnägijatel veebis autentimise ja allkirjastamise toiminguid teha. Rakendus on varustatud ka ühiktestidega, mis võimaldab pidevintegratsioonis enamlevinud kasutusjuhud kontrollerist läbi jooksutada.

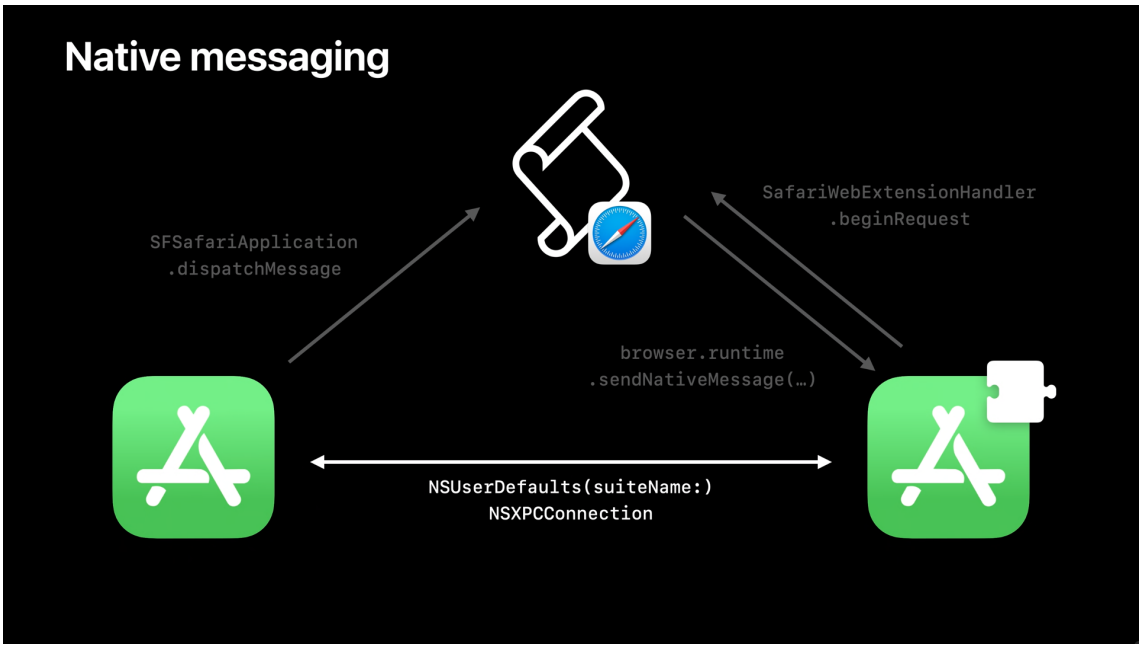
Web eID veebilehitseja JavaScript laiendus (Joonis 1 - komponent “Web eID browser extension”) on oluline komponent, sest see on sild veebilehe ja töölauarakenduse vahel. Laiendusel on õigus käivitada ja suhelda töölauarakendusega. See teeb esmased turvakontrollid sissetulevatelt sõnumitelt ning takistab ebaturvalistel lehtedel ligipääsu ID kaardile. Web eID veebilehitseja laiendused allkirjastatakse erinevate veebilehitsejate laienduste levitamiste teenustes ja nende levitamiseks on erinevad mehhanismid. Chrome laiendust levitatakse Google Chrome poes (Chrome Web Store), Edge laiendust Microsoft Edge Add-ons poes (Microsoft Edge Add-ons) ja Firefox laiendust saab levitada nii Mozilla Add-Ons poes (Firefox Browser Add-ons) kui ka koos töölauarakendusega. Töölauarakendus annab Chrome ja Edge veebilehitsejatele nende käivitamisel juhised kuidas paigaldada poest laiendused.

Web eID JavaScript teek (Joonis 1 - komponent “Web eID JavaScript library”) on mõeldud teenuste arendajatele. Teek sisestatakse veebilehele läbi mõne tuntud JavaScript teegi halduri ja see omab lihtsat programmiliidest Web eID käivitamiseks. Viimane peidab kogu keerukuse suhelda Web eID veebilehitseja laiendusega ja saab aru erinevate veebilehitseja laienduste eripärasustest. Ka saab see tulevikus hakkama erinevate laienduste versioonidega ning pakub tagasiühilduvust praeguse kasutusel olevate veebilehitseja laiendustega.

CMake projektifailid sisaldavad juhiseid paigalduspakkide valmistamiseks nii Windowsi, macOS kui Ubuntu Linuxile. Paigalduspakid sisaldavad kõiki vajalikke komponente nii laienduse käivitamiseks, Qt raamistiku faile ja ka vajalikke C ja C++ lisamooduleid. Paigalduspakid tekitavad peale nende paigaldamist juhised registrisse või seadete failidesse, mille järgi veebilehitseja paigaldab JavaScript laienduse. Pidevintregratsioonist saab paigalduspakid alla laadida, millega annab hõlpsalt testida arendusharude koodi.

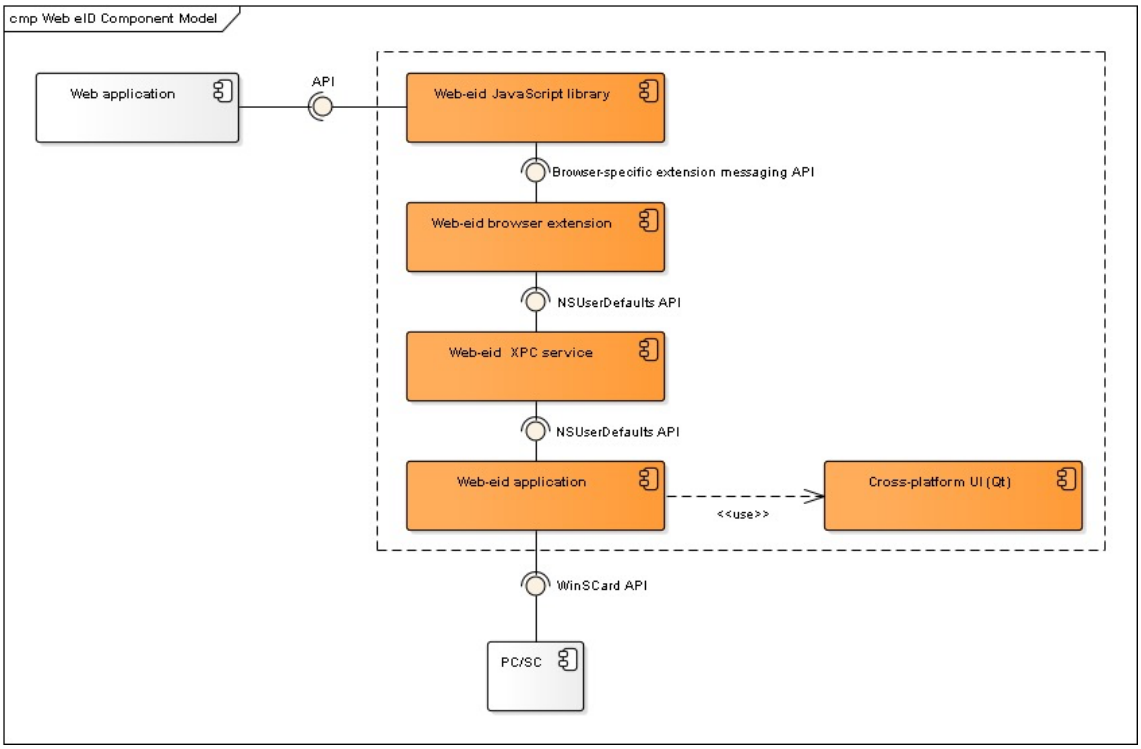
4 *Safari Web Extensions* veebilehitseja laiendus

Apple tutvustas 2020 juunis Worldwide Developer konverentsil uut *Safari Web Extensions* laienduste tuge. See oli Web eID projekti jaoks positiivne uudis kuna Chrome ja Firefox veebilehitseja *Web Extensions* prototüüpid olid projektis juba eelnevalt arendatud. *Safari Web Extensions* laienduste arhitektuur on identne olemasolevate laiendustega ja nõudis minimaalseid muudatusi JavaScript poolelt. Küll aga erines viis, kuidas Safari laiendus suhtles töölauarakendusega. Chrome laienduses käivitab töölauarakenduse JavaScript, suhtlus toimub kasutades protsessi sisend/väljund kanalit (*standard input/output stream*). Safari veebilehitseja *Web Extensions* laienduses on protsessiga suhtlemise tehnoloogia erinev. Seal käivitab JavaScript vahepealse turvamooduli ning vajadusel käivitab ja suhtleb turvamoodul töölauarakendusega. Joonisel 2 on näha kuidas liiguvad sõnumid JavaScript laienduse *background* lehe, turvamooduli ja töölauarakenduse vahel.



Joonis 2. Safari Web Extensions sõnumite liikumise skeem [18]

Safari komponent mudel erineb olemasoleva lahenduse komponent mudelist. Lisaks olemasolevale on Web eID Safari komponent mudelis XPC teenuse turvamoodul [19] (Joonis 3).



Joonis 3. Web eID Safari komponent mudel

Tööd alustades uuris autor esmajoones Apple Worldwide Developer 2020. aasta konverentsil jagatud näidis koodi, et tuvastada selle vastavus projekti ootustele ja nõuetele. Uurimuse tulemused kinnitasid, et *Safari Web Extensions* tehnoloogia on väga sarnane Chrome laienduses oleva tehnoloogiaga.

Seejärel ehitas autor lihtsustatud näite ja tuvastas esimese kitsaskoha. XPC teenuse mooduli ja töölauarakenduse vahel ei saanud saata sõnumeid. Selleks leidis autor alternatiivi- salvestada sõnum seadete faili ja saata operatsioonisüsteemile teade, mis levitaks seda kõikidele rakendustele. Teade omab viidet seadete failis olevale sõnumile. Seejärel tuvastab töölauarakendus selle teate, otsib seadete failist viidatud sõnumi, töötleb seda ning saadab vastuse. Töölauarakendusel on võimalus saata vastus JavaScript *background* scriptile kasutades otsekanalit või XPC teenuse moodulit. Kuna XPC teenus kasutab sõnumite vahetamiseks seadete faili ning see on sarnane sõnumi saatmise protsessile kasutab ka töölauarakendus vastuse saatmiseks sama mudelit.

macOS nõuab tungivalt kõikide rakenduste allkirjastamist. See tekitas järgmise probleemi. XPC teenuse moodul ja töölauarakendus allkirjastatakse eraldi ja neil puudub ligipääs üksteise turvaaedikule (*sandbox*), ehk nad ei saa omavahel faile jagada. Selleks

on Apple teinud App Groups allkirjastamise grupi, kuhu on kahel või enamal rakendusel ligipääs.

Web eID töölauarakendus on ehitatud nii, et erinevaid komponente saab taaskasutada. Seega saab taaskasutada kontrollerit ja sõltuvaid komponente. Rakenduse kirjutamisel piisas ainult Safari eripärade käsitlemisest ja Objective-C keelest sõnumite kohandamisest Web eID kontrollerile arusaadavaks.

Projekti skoobis oli hoida ühtset ehitamise mehhanismi ning suunis kasutada ka siin CMake tööriista. Näidiste kood kasutas XCode projekte, koodi ehitus möödus probleemideta. Koodi ehitamisel CMake-ga tuvastas autor programmimoodulite linkimisel, et XPC teenuse moodul ei ole päris käitusteegi ega ka rakenduse tüüpi. Lugedes Apple dokumentatsiooni sai selgeks, et tegemist on pistikprogramm süsteemiga, mis nõuab eri linkimise parameetreid "-e _NSExtensionMain -fapplication-extension".

Tulenevalt Safari eripäradest tuli teha väiksemad muudatused JavaScript laienduse koodi. Safari veebilehitseja laienduse levitamine erineb teistest veebilehitsejatest. Laiendus pakitakse töölaua rakenduse sisse ja levitatakse koos rakendusega. Laienduse olekusuvand veebilehitsejas on vaikimisi mitte lubatud. Käivitades kontrollib töölauarakendus kas laiendus on aktiveeritud ja vajadusel avab Safari seadete lehe, kus kasutaja saab teostada muudatuse.

Apple eelistab rakenduste kirjutamiseks Swift ja Objective-C programmeerimise keelt. Swift on kaasaegne ja moodne keel programmide kirjutamiseks aga see ei suuda täna veel otse kutsuda C++ koodi. Selleks on vaja kirjutada Objective-C koodis sild, mis vahendab sõnumeid Swift ja C++ vahel. Tulenevalt Qt raamistikust omab Web eID kontroller üksjagu keerukust, seetõttu ei ole selle silla ehitamine mõistlik. Optimaalne on kohe kasutada Objective-C keelt töölaua rakenduse kirjutamiseks.

Töölauarakendus käivitatakse kasutaja sisselogimisel ja jookseb taustal. See kiirendab tunduvalt rakenduse reageerimise kiirust kui kasutaja käivitab protsessi veebileheküljelt. Töö käigus tekkis probleem, et tavalist tüüpi töölauarakendusel on rakenduse ikoon tegumiribal. See aga võib kaasa tuua kasutajapoolse impulsiivse käitumise, mis suunab kasutajat rakendust sulgema. Selle olukorra lahendamiseks lisati *LSUIElement* rakenduse parameetrite hulka. See element viitab operatsioonisüsteemile, et töölauarakendus jookseb taustal ja ei tekita tegumiribale ikooni.

4.1 *Safari Web Extensions* veebilehitseja laienduse arendus

Safari Web Extensions laienduse arendus teostati kahes etapis. Esmajoones keskendus autor lihtsamate sõnumite vahetamisele laienduse ja rakenduse vahel, seejärel integreeriti väljatöötatud lahendus olemasoleva protsessiga.

Olemasoleva Web eID projekti struktuur[1]:

- debian - Debian pakenduse failide kaust
- install - Windows ja macOS pakenduse abifailid
- lib - vajalikud teegid, libelectronic-id
- script - git ja pidevintegratsiooni integreerimise scriptid
- src - projekti kood
- tests - projekti testid

Src kataloogi lisas autor uue kataloogi mac kuhu lisati Safari jaoks vajalikud failid[2].

- CMakeLists.txt fail sisaldab juhiseid kuidas ehitada *web-eid-safari* rakendus ja ka *web-eid-safari-extension* XPC teenuse moodul. Projekti ehitamise käigus paigaldatakse *web-eid-safari-extension* ning JavaScript laienduse failid *web-eid-safari* rakenduse sisse. Lisaks on selles failis toodud juhised macOS Safari laienduse paigalduspakkide ehitamiseks.
- main.mm on Objective-C++ fail, mis võimaldab kasutada nii Objective-C kui ka C++ programmeerimiskeelt. Fail sisaldab *main* funktsiooni, mida jooksutatakse rakenduse käivitamisel. *main* funktsioon initsialiseerib Qt raamistiku ja teeb rakenduse tööks vajalikud toimingud: laadib tõlkefailid, käivitab logimise, kontrollib Safaris laienduse olekut, käivitab teavituste süsteemi ning registreerib rakenduse kasutaja seadetesse, et kasutajakontoga sisselogimisel käivitataks töölauarakendus taustal. Lisaks on failis abifunktsioonid, mis aitavad Objective-

[1] https://github.com/web-eid/web-eid-app

[2] https://github.com/web-eid/web-eid-app/tree/main/src/mac

C andmetüübid konverteerida Qt andmetüüpideks. Kõige olulisem funktsioon on *notificationEvent*, mis käivitub globaalse teate saamisel ja loeb sõnumi failist. Seejärel käivitab see funktsioon Web eID kontrolleri, millele annab sõnumi kaasa. Peale kontrolleri tööd tagastatakse funktsioonile vastus, see salvestatakse faili ja saadetakse teade operatsioonisüsteemile, mis edastab selle kõikidele rakendustele.

- safari-extension.mm failis asub XPC teenuse mooduli kood. *SafariWebExtensionHandler* klassi laadimisel käivitub teavituste süsteem. Kui JavaScript saadab sõnumi, siis käivitatakse funktsioon *beginRequestWithExtensionContext*. See funktsioon kontrollib töölauarakenduse toimimist ja vajadusel käivitab töölauarakenduse. JavaScripti saadetud sõnum salvestatakse ja saadetakse globaalse sõnumina kõikidele rakendustele, sealhulgas salvestatakse ka kontekst, et hiljem saaks sama päringu tegijale saata vastuse. Ka siin failis on *notificationEvent* funktsioon, mis saab teate, kui *web-eid-safari* rakendus on lõpetanud sõnumi töötlemise ja saatnud päringule vastuse. Nüüd kasutab funktsioon salvestatud konteksti kuhu suunatakse ka vastus.

4.2 *Safari Web Extensions* veebilehitseja laienduse ehitamine

Ehitamiseks kasutas töö autor macOS operatsioonisüsteemiga varustatud Apple arvutit. Esmajoones paigaldas autor arvutisse Xcode integreeritud programmeerimiskeskkonna Apple AppStorest[1]. Viimasega tulevad kaasa vajalikud kompilaatorid, platvormi teegid ning raamistikud. Seejärel paigaldati arvutisse Homebrew pakihaldus. Töö alustamisel, avas autor käsurea rakenduse *Terminal* ning sisestas käsu, mis käivitas paigalduse ja kuvas vajalikke juhtnööre paigalduse edukaks lõpetamiseks:

```
/bin/bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

Seejärel tuli paigaldada pakid Qt raamistik, CMake ja Node tööriist, OpenSSL ja GTest teegid kasutades Homebrew käsurea tööriista:

```
brew install qt@5 cmake openssl@1.1 node web-eid/gtest/gtest
```

Järgmisena tuli hankida lähtekood GitHub-ist [20] https://github.com/web-eid/web-eid-app. Seda tegi autor käivitades käsurealt käsu:

```
git clone --recursive https://github.com/web-eid/web-eid-app.git
```

Järgmiseks tuli vahetada vaikekataloog *web-eid-app* vastu:

```
cd web-eid-app
```

Soovitatav oli teha eraldi *build* kataloog, kus rakendus ehitada ning seejärel vahetada vaikekataloog *build* kataloogi vastu:

```
mkdir build
cd build
```

[1] https://apps.apple.com/us/app/xcode/id497799835?mt=12

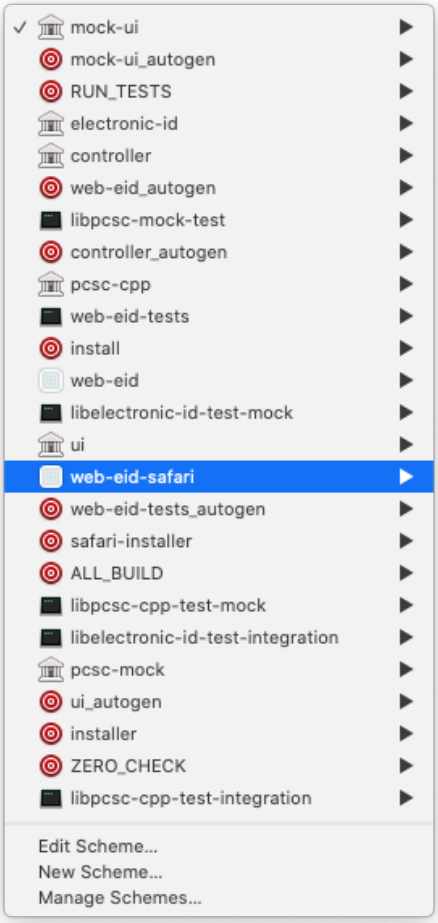
Järgmiseks oli vaja luua Xcode projekti failid kasutades CMake tööriista ning viidata OpenSSL teegi ja Qt raamistiku asukohale:

```
cmake -GXcode \
-DOPENSSL_ROOT_DIR=/usr/local/opt/openssl@1.1 \
-DQt5_DIR=/usr/local/opt/qt@5 ..
```

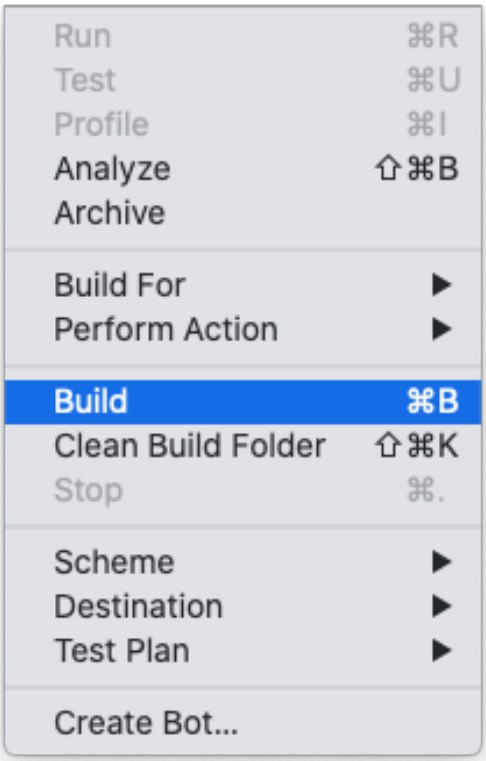
Edasi tuli avada projekt Xcode-s:

```
open web-eid.xcodeproj
```

Xcode projektis tuli valida vaikimisi rakenduseks web-eid-safari ning ülevalt menüüst (Joonis 4) anda ehitamise käsk (Joonis 5):

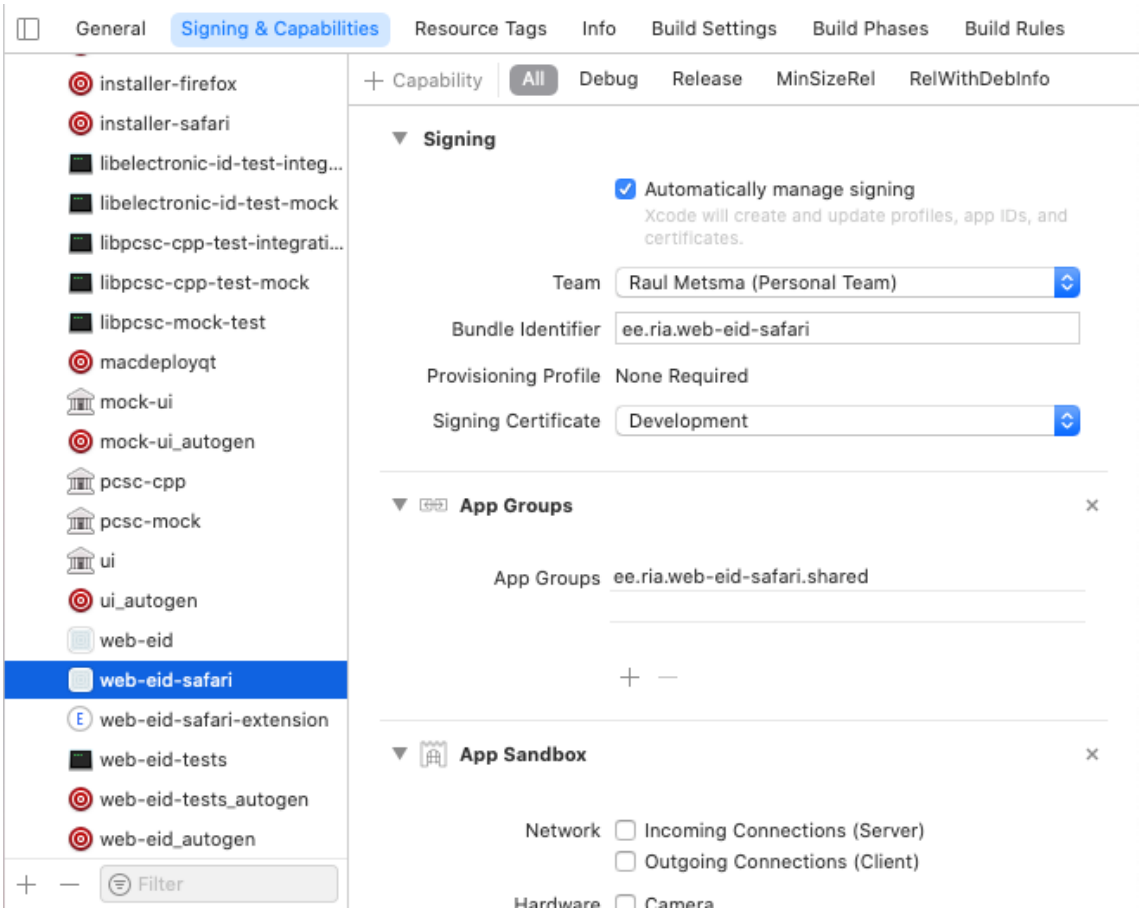


Joonis 4. Ehitamise sihtmärgi valik



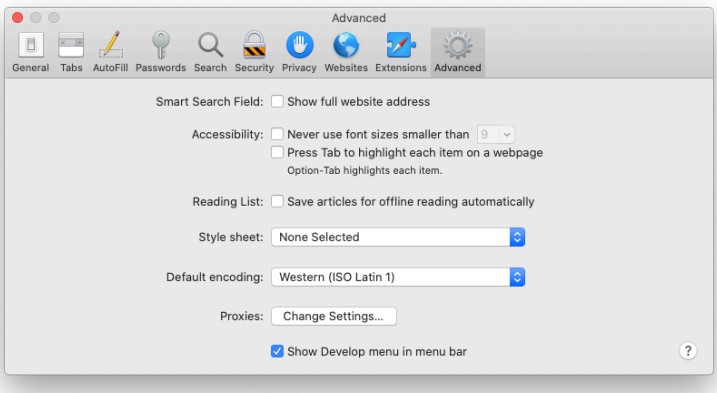
Joonis 5. Ehitamise käsk

Töö autoril oli Apple ID konto ja see oli sisestatud Xcode seadetesse, mis võimaldas ehitamise käigus kohe rakendus allkirjastada. See lihtsustas töö järgmist etappi, kus ei olnud vaja Safari veebilehitsejas eraldi sisse lülitada valikut, mis lubaks käivitada allkirjastamata veebilehitseja laiendusi. Selleks, et lülitada sisse allkirjastamine oli vaja valida *web-eid-safari* ja *web-eid-safari-extensions* komponendile Team: Eesnimi Perenimi (Personal Team) (Joonis 6)

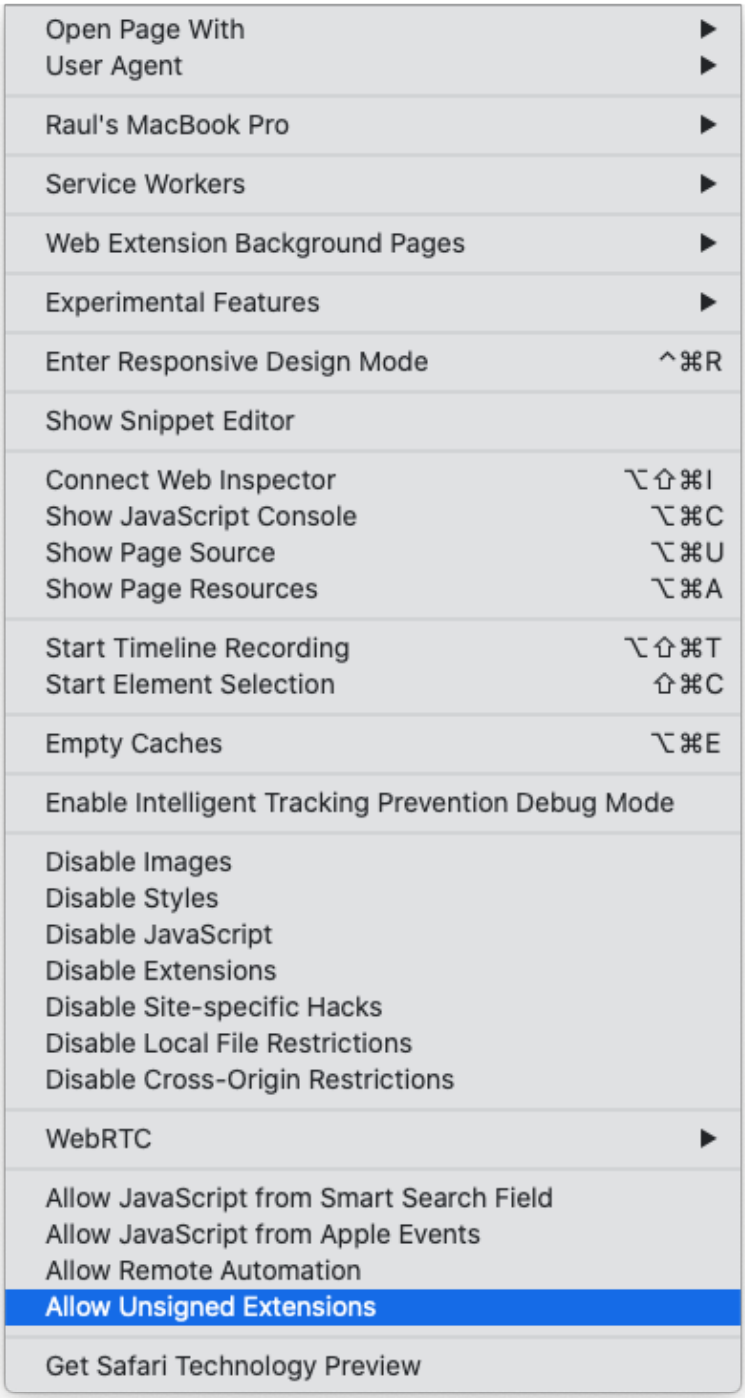


Joonis 6. Allkirjastamise meeskonna valik

Safari veebilehitsejaga töötamisel oli abiks veebilehitseja arendaja tööriistad. Arendajatele mõeldud tööriistade jaoks tuli autoril avada Safari seaded ning teha valik *Advanced* seadete kasuks (Joonis 7). Seejärel tekkis Safari menüüribale valik *Develop*. Juhul, kui arendaja ei soovi kasutada allkirjastatud rakendust, siis võib veebilehitseja seadetes valida *Allow Unsigned Extensions* (Joonis 8).

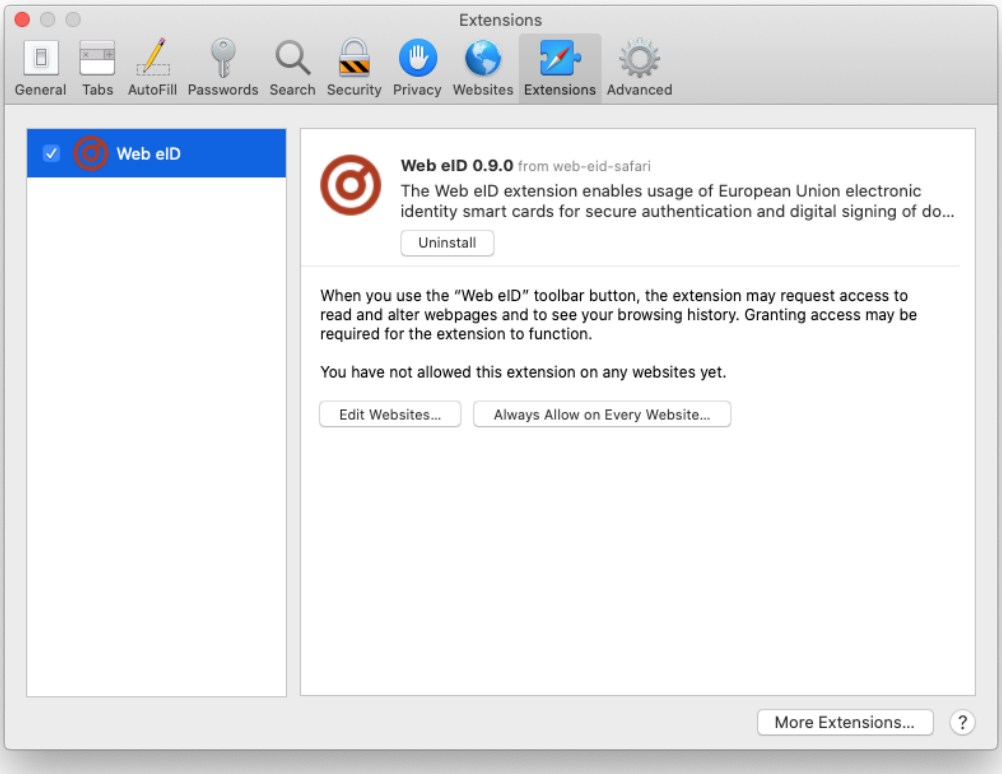


Joonis 7. Develop menüü aktiveerimine



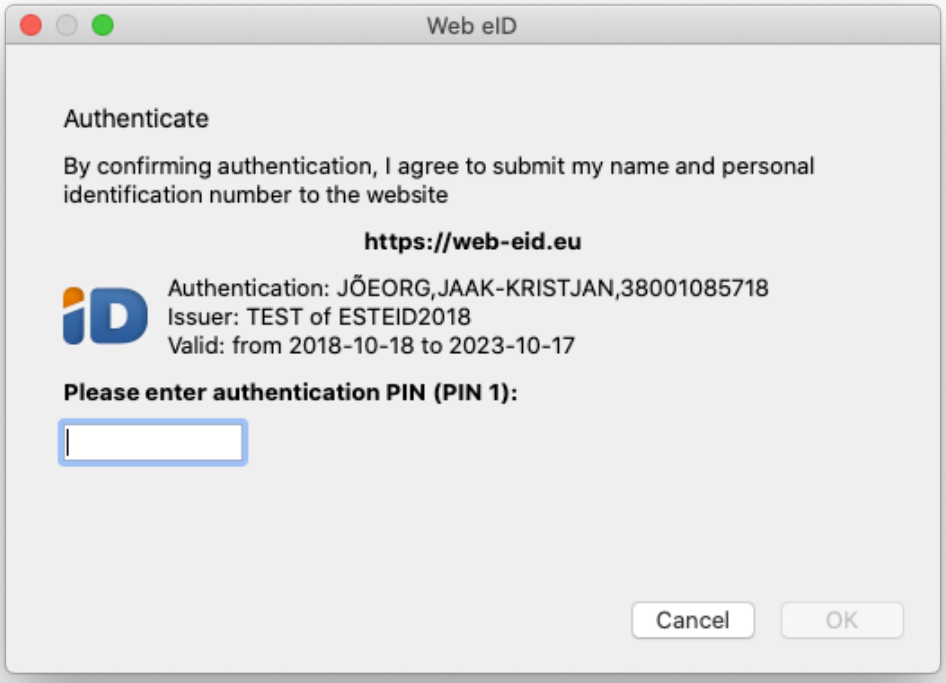
Joonis 8. Allkirjastamata laienduse lubamine

Järgmisena tuli autoril käivitada Xcodes *web-eid-safari* rakendus. Rakenduse käivitamisel kontrollis see esmajoones, kas laiendus oli veebilehitsejas lubatud, vajadusel avas rakendus seadete akna, kus sai laienduse lubamiseks loa anda. Arendamise ajaks oli soovitatav sisse lülitada ka "*Always Allow on Every Website…*" mis aitas vältida laienduse ligipääsu testitavale veebilehele. Sellekohane näide on toodud joonisel 9.



Joonis 9. Web eID laienduse kasutamise lubamine

Rakenduse toimimist testis autor avades Safaris lehekülje https://web-eid.eu [21] ning käivitades autentimise protsess (*Authenticate*). "Test loeti õnnestunuks kui testi järgselt ilmus menüüaken PIN koodi päringuga (Joonis 10). PIN koodi sisestamise järel suunati kasutaja lehele *Adding Signature*, mis võimaldas allkirja anda.



Joonis 10. Autentimise PIN koodi küsimise aken

4.3 *Safari Web Extensions* veebilehitseja laienduse tõrke otsimine

Allpool on toodud valik käsurea tööriistu, mis olid abiks veebilehitseja laienduse tõrke otsimisel:

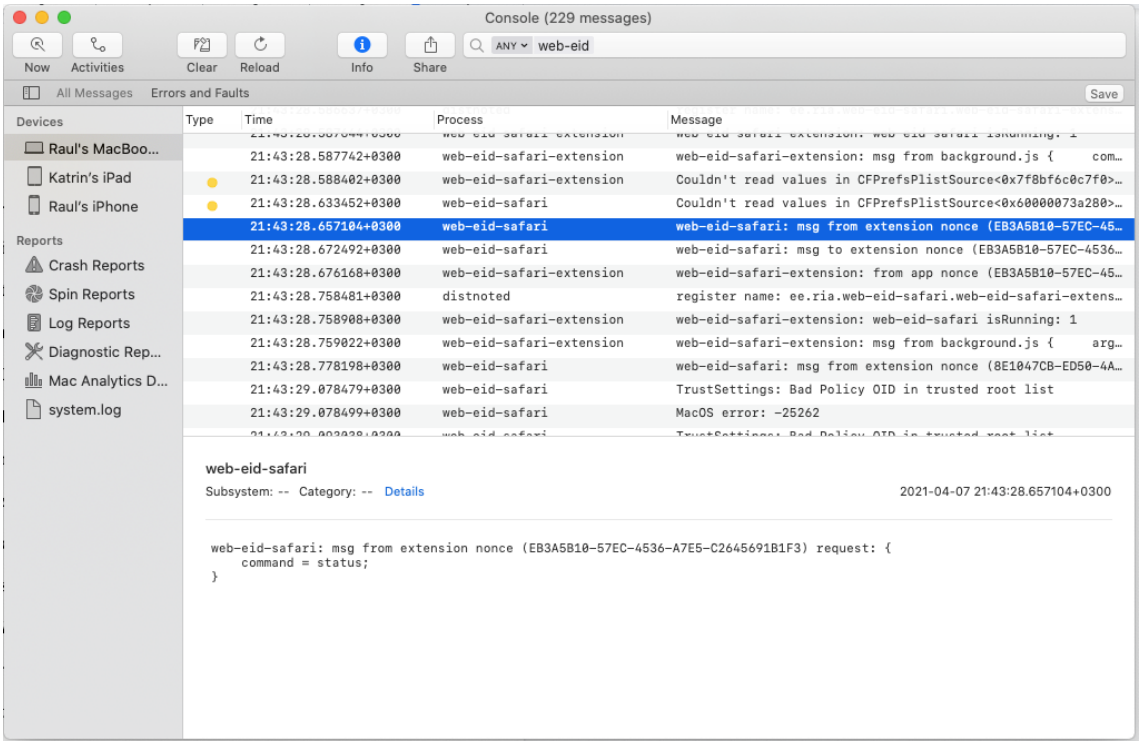
- *codesign* käsk- näitas rakenduse allkirja staatust ja detaile. Tööriista kohta sai autor rohkem infot käsuga "man codesign".

```
codesign -v -vvv build/src/mac/Debug/web-eid-safari.app
--prepared:build/src/mac/Debug/web-eid-safari.app/Contents/PlugIns/web-eid-
safari-extension.appex
--validated:build/src/mac/Debug/web-eid-safari.app/Contents/PlugIns/web-eid-
safari-extension.appex
build/src/mac/Debug/web-eid-safari.app: valid on disk
build/src/mac/Debug/web-eid-safari.app: satisfies its Designated Requirement

codesign -d -vvv build/src/mac/Debug/web-eid-safari.app
Executable=build/src/mac/Debug/web-eid-safari.app/Contents/MacOS/web-eid-
safari
Identifier=ee.ria.web-eid-safari
Format=app bundle with Mach-O thin (x86_64)
CodeDirectory v=20100 size=30982 flags=0x2(adhoc) hashes=961+5
location=embedded
Hash type=sha256 size=32
CandidateCDHash sha256=81c4c7223eb7f5e52b53aa0bcfbec0a8fb57930e
CandidateCDHashFull
sha256=81c4c7223eb7f5e52b53aa0bcfbec0a8fb57930e6d24bc37f3bc245f32eb380f
Hash choices=sha256
CMSDigest=81c4c7223eb7f5e52b53aa0bcfbec0a8fb57930e6d24bc37f3bc245f32eb380f
CMSDigestType=2
CDHash=81c4c7223eb7f5e52b53aa0bcfbec0a8fb57930e
Signature=adhoc
Info.plist entries=23
TeamIdentifier=not set
Sealed Resources version=2 rules=13 files=1
Internal requirements count=0 size=12
```

- *pluginkit* käsk- kuvas laienduse registreeringu süsteemis ning vajadusel aitas selle sinna lisada. Rohkema info saamiseks kasutas autor käsku "man pluginkit".

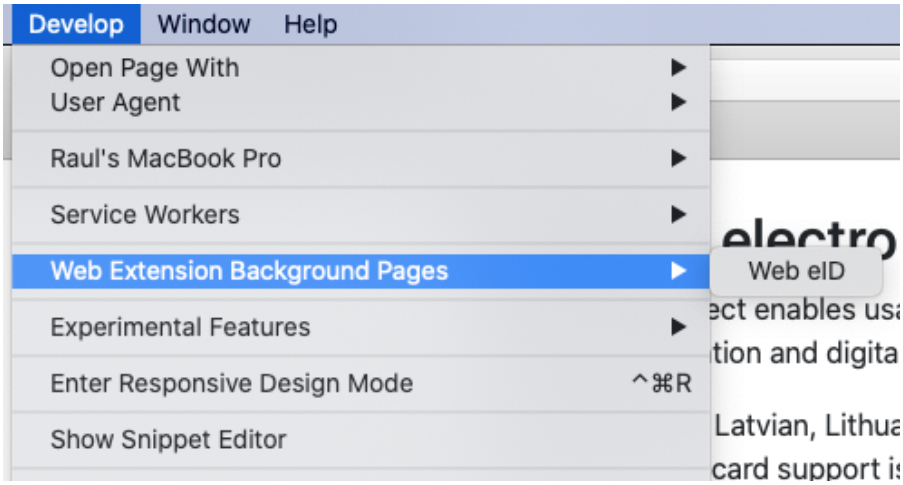
```
pluginkit -mvADi ee.ria.web-eid-safari.web-eid-safari-extension
ee.ria.web-eid-safari.web-eid-safari-extension(1.0.0)      E5FC0E14-A39E-
4510-A3B3-0F75CCE6DD4C    2021-03-21 17:47:16 +0000 build/src/mac/Debug/web-
eid-safari.app/Contents/PlugIns/web-eid-safari-extension.appex
(1 plug-in)

pluginkit -a build/src/mac/Debug/web-eid-safari.app/Contents/PlugIns/web-eid-
safari-extension.appex
```

- *Console.app* rakendust saab kasutada XPC teenuse mooduli silumise teadete lugemiseks.

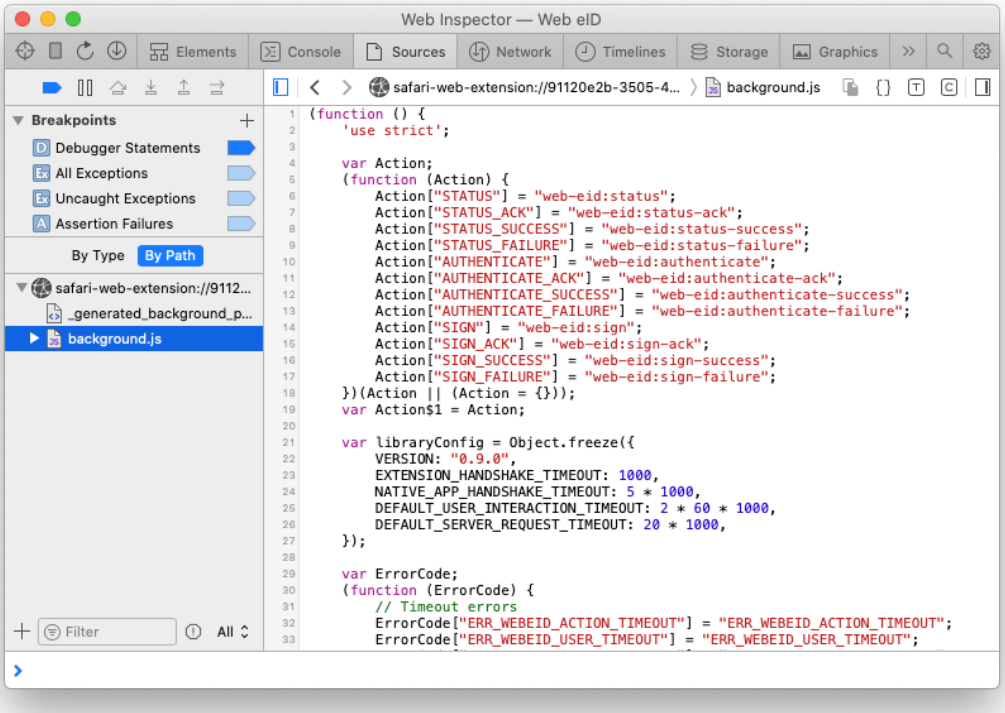


Joonis 11. Console.app silumis teadete lugemiseks

- Laienduse taustalehe JavaScript siluri käivitamiseks tuli autoril valida Safari *Develop* menüüst "*Web Extension Background Pages*" alammenüüst "Web eID" (Joonis 12, 13).

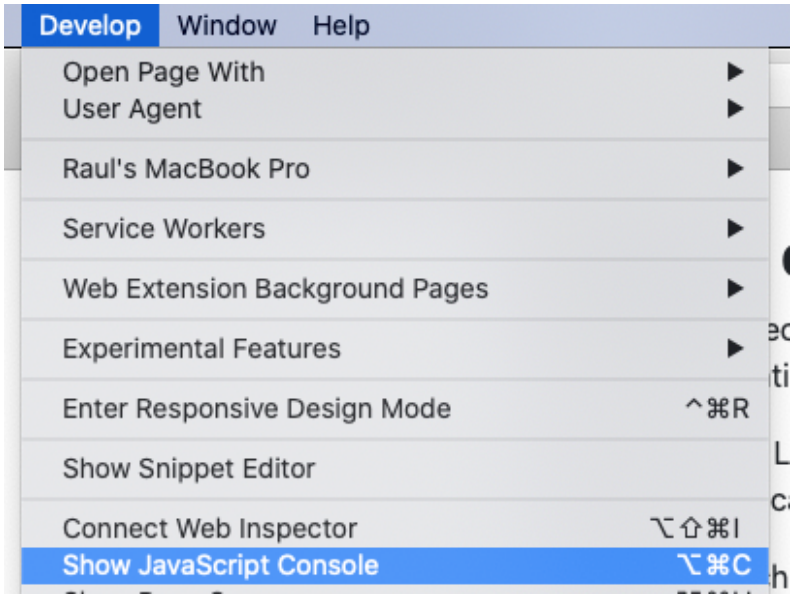


Joonis 12. Laienduse taustalehe siluri käivitamine

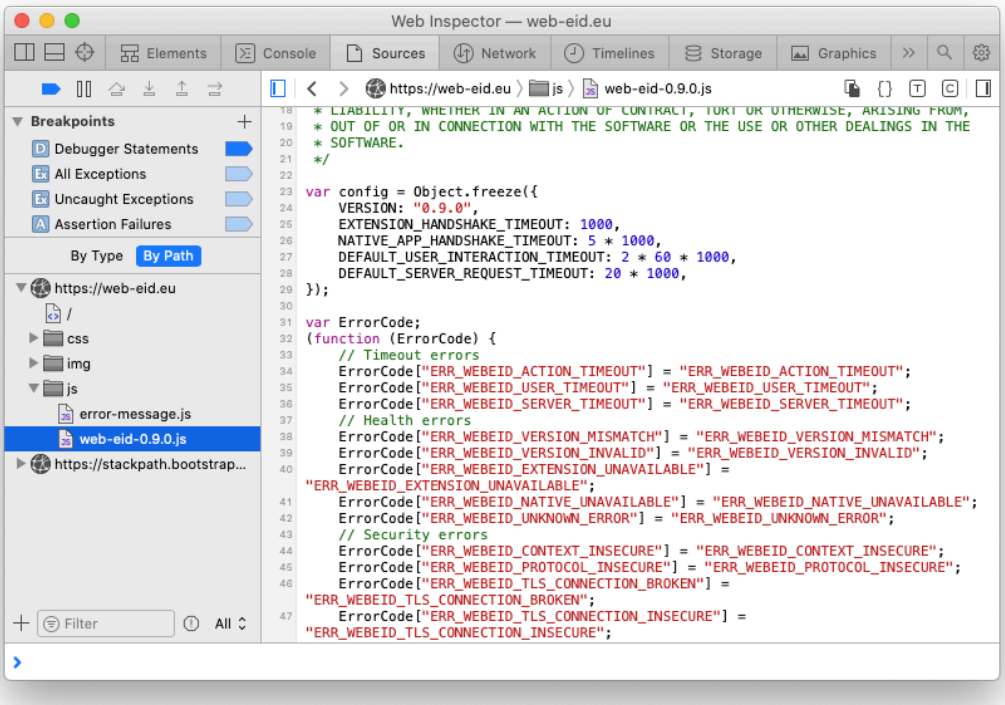


Joonis 13. Laienduse taustalehe silur

- Veebilehe JavaScript siluri käivitamiseks tuli valida Safari *Develop* menüüst "*Show JavaScript Console*" (Joonis 14, 15).



Joonis 14. Veebilehe siluri käivitamine



Joonis 15. Veebilehe silur

5 Kokkuvõte

Projekti käigus realiseeriti Web eID veebilehitseja laiendus, mis pakub lisaks allkirjastamisele ka autentimise tuge. Lisaks veebilehitseja laiendusele on saadaval ka Java keeles serveri poolne komponent autentimise tõendi valideerimiseks. Eraldi viidi läbi turvaanalüüs, mille tulemusest koostati dokument, mis analüüsib ja kirjeldab detailselt lahenduses realiseeritud tehnilised turvameetmed. Lisaks läbis lahendus ka välise turvaauditi, ning leitud vead parandati. Projekti skoop on katta enamlevinud veebilehitsejad. Kuna uue lahenduse juurutamisel integraatoritel ning kasutajatele tarkvara levitamisega läheb aega, jäeti välja vananenud veebilehitsejate tugi näiteks: Internet Explorer, Edge Legacy.

Analüüsi käigus selgus, et macOS operatsioonisüsteemil oli mõistlik võtta kasutusele *Safari Web Extensions* tehnoloogia mille võttis Apple kasutusele Safari versioonist 14-st. Teoreetiliselt oleks võimalik ka kasutada *Safari App Extensions* tehnoloogiat alates Safari 12-st mis oleks nõudnud tunduvalt keerukamat JavaScript integreerimist olemasoleva Chrome *Web Extensions Native Messaging* lahendusega. Ka oli teada mõningad kitsaskohad sellel tehnoloogial, mis avaldub praeguse *Safari Token Signing* [22] veebilehitseja laiendusega. Näiteks on piirang PIN koodi sisetamise ajale, mida ei ole võimalik seadistada ja mille tulemusena ei jõua allkiri veebiteenuseni, mis omakorda loeb allkirjastamise ebaõnnestunuks. Tulenevalt eelpool nimetatust ja levitamise ajakavast ei olnud majanduslikult mõistlik arendada lahendust juba vananevale tehnoloogiale.

Projekti käigus tutvuvus autor Apple Developer konverentsil avaldatud näidetega mis avaldati 2020 juulis. Võttes aluseks näidised ehitati esmalt lihtsamat sorti sõnumite edastamise prototüüp, et täpsemalt aru saada *Safari Web Extensions* tehnoloogia võimekusest ja puudustest. Tulemused kinnitasid, et *Safari App Extensions* teada olevatest puudustest saab mööda. See andis kindlust, et võib ehitada uue Web eID veebilehitseja laienduse kasutades *Safari Web Extensions* tehnoloogiat. Valmis lahenduse testimise käigus saadi ka kinnitust ja kindlust uue lahenduse paremas toimimises võrreldes *Safari Token Signing*-uga.

Majanduslikust aspektist lähtudes on arendaja/haldaja pool ja kasutaja pool. Seoses lahenduse kasutuselevõtuga väheneb arendajate koormus kuna lahendus on stabiilsem ja ei vaja pidevat arendamist. Samamoodi ka halduse seisukohalt on kasutajatoe koormus väiksem, sest lahendus ei ole pidevas muutumises. Kasutajamugavus on oluliselt parem tänu lahenduse stabiilsusele ning vastutasuks on kasutajal võimalik muretumalt kasutada kiibipõhist eID-d oma tööjaama küljes. Uus lahendus ühtlustab kasutajakogemuse eri veebilehitsejate, operatsioonisüsteemide ning autentimise ja allkirjastamise toimingu vahel.

Täiendav arendusvajadus tekib turvaanalüüsi leidude lahendamisega ja operatsioonisüsteemide arenguga. Turvaanalüüsist selgus, et see lahendus ei oma sama turvataset mis on praegusel Transport Layer Security (TLS) protokollil. Selle täiendamiseks on vaja teha täiendamise või liideste muudatuste ettepanekud koos põhjendustega veebilehitseja tootjatele.

Kasutatud kirjandus

[1] T. Dierks and E. Rescorla, "TLS," August 2008. [Online]. Available: https://tools.ietf.org/html/rfc5246.

[2] "Web eID," [Online]. Available: https://web-eid.eu.

[3] "Open eID," [Online]. Available: http://open-eid.github.io.

[4] "hwcrypto.js Open eID veebilehitseja laienduse JavaScript teek," [Online]. Available: https://github.com/hwcrypto/hwcrypto.js.

[5] "ISO/IEC 14882:2017 Programming languages — C++," Detsember 2017. [Online]. Available: https://www.iso.org/standard/68564.html.

[6] "Qt," [Online]. Available: https://www.qt.io/product/framework.

[7] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley and T. Polk, "X.509," Mai 2008. [Online]. Available: https://tools.ietf.org/html/rfc5280.

[8] "Visual Studio," [Online]. Available: https://visualstudio.microsoft.com.

[9] "Xcode," [Online]. Available: https://developer.apple.com/xcode/.

[10] "CMake," [Online]. Available: https://cmake.org.

[11] "Qt 6 Build System," 12 October 2020. [Online]. Available: https://www.qt.io/blog/qt-6-build-system.

[12] Microsoft, "Smart Card Minidriver standard," 25 Veebruar 2016. [Online]. Available: https://download.microsoft.com/download/3/3/2/332fd70b-f04d-470a-a135-040350b9563f/sc-minidriver_specs_v7.07.docx.

[13] OASIS, "PKCS #11," 13 Mai 2016. [Online]. Available: http://docs.oasis-open.org/pkcs11/pkcs11-base/v2.40/errata01/os/pkcs11-base-v2.40-errata01-os-complete.html.

[14] E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3," August 2018. [Online]. Available: https://tools.ietf.org/html/rfc8446.

[15] N. Sakimura, J. Bradley, M. B. Jones, B. d. Medeiros and C. Mortimore, "OpenID Connect Core 1.0," 8 November 2014. [Online]. Available: https://openid.net/specs/openid-connect-core-1_0.html.

[16] Cybernetica, "Web eID turvaanalüüs," 18 Detsember 2020. [Online]. Available: https://web-eid.gitlab.io/analysis/webextensions-main.pdf.

[17] M. Sõmermaa, "Web eID arhitektuur," 10 Jaanuar 2019. [Online]. Available: https://github.com/web-eid/web-eid-system-architecture-doc.

[18] "Meet Safari Web Extensions," [Online]. Available: https://developer.apple.com/videos/play/wwdc2020/10665/.

[19] Apple, "Creating XPC Service," [Online]. Available: https://developer.apple.com/library/archive/documentation/MacOSX/Conceptual/BPSystemStartup/Chapters/CreatingXPCServices.html.

[20] "GitHub Koodihoidla," [Online]. Available: https://github.com.

[21] "Web eID Demo keskkond," [Online]. Available: https://web-eid.eu.

[22] "Safari Token Signing," [Online]. Available: https://github.com/open-eid/browser-token-signing/tree/master/SafariAppExtension.

[23] L. S. Sterling, The Art of Agent-Oriented Modeling, London: The MIT Press, 2009.

Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

Mina, Raul Metsma

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "Web eID toe lisamine Safari veebilehitsejale", mille juhendaja on Meelis Antoi
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

17.05.2021

1 Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.