SAVONIA

OPINNÄYTETYÖ - **AMMATTIKORKEAKOULUTUTKINTO**
TEKNIIKAN JA LIIKENTEEN ALA

AUTENTIKOINTIMENETELMIEN SOVELTAMINEN

TEKIJÄ: Samuli Tanskanen

SAVONIA-AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Tiivistelmä

Koulutusala Tekniikan ja liikenteen ala	
Koulutusohjelma/Tutkinto-ohjelma Tietotekniikan koulutusohjelma	
Työn tekijä(t) Samuli Tanskanen	
Työn nimi Autentikointimenetelmien soveltaminen	
Päiväys 06.06.2018	Sivumäärä/Liitteet 26
Ohjaaja(t) Mikko Pääkkönen, TKI-Asiantuntija, Mikko Laasanen, TKI-Asiantuntija	
Toimeksiantaja/Yhteistyökumppani(t) NettiTieto Oy	

Tiivistelmä

Opinnäytetyön aiheena oli tutkia erilaisia autentikointi- ja autorisointimenetelmiä eli tapoja käyttäjän identiteetin ja oikeuksien varmistamiseen NettiTieto Oy:lle. Aihepiiriin kuului myös vahvan sähköisen tunnistamisen menetelmät Suomessa sekä niiden pohjalla oleva lainsäädäntö, sekä autentikointiportaalin toteutus näiden tietojen pohjalta.

Tämä työ on jaettavissa kolmeen pääosaan: autentikointi- ja autorisointimenetelmiin, vahvaan sähköiseen tunnistamiseen sekä autentikointiportaalin toteutukseen. Työtä tehdessä teoriatietoa kerättiin lähinnä sähköisistä lähteistä.

Työn tavoitteena oli kehittää helppokäyttöinen, helposti integroitava sovellus, jonka kautta vahvan sähköisen tunnistautumisen tapahtumia pystyisi välittämään asiakaspalvelun ja tunnistuspalvelun tarjoajan välillä. Autentikointiportaalin toteutuksessa hyödynnettiin Laravel-ohjelmistokehystä ja se toteutettiin MVC-mallia noudattaen.

Lopputuloksena syntyi sovellus, jota yritys voi nyt alkaa testaamaan omassa ympäristössään. Vaikka sovellusta ei saatu testattua tuotantoympäristössä, halutut ominaisuudet saatiin kuitenkin toteutettua.

Avainsanat
OAuth, OAuth2, Autentikointi, JWT, Rajapinta, PHP, Nets, Signicat, SAML

SAVONIA UNIVERSITY OF APPLIED SCIENCES

THESIS
Abstract

Field of Study Technology, Communication and Transport			
Degree Programme Degree Programme in Information Technology			
Author(s) Samuli Tanskanen			
Title of Thesis Applications of Authentication Methods			
Date	6 June 2018	Pages/Appendices	26
Supervisor(s) Mr Mikko Pääkkönen, RDI Specialist, Mr Mikko Laasanen, RDI Specialist			
Client Organisation /Partners NettiTieto			

Abstract

The purpose of this thesis was to study various authentication and authorizations methods, ie ways to confirm the identity and the rights of the user. The topic also included the methods of strong electronic authentication in Finland and the underlying regulations, as well as the implementation of an authentication portal based on this information.

This thesis can be split into three main parts: authentication and authorization methods, strong electronic authentication and lastly, the implementation of the authentication portal. Most of the information was gathered from electronical sources.

The goal of the thesis was to develop an easy-to-use, easy-to-integrate application that would act as the middle man between the customer service and the identification service provider. The Laravel software framework was used in the implementation of the authentication portal and it was implemented in accordance with the MVC model.

The result of this thesis was an application that can now be tested by the company in their own environment. Although the application was not tested in production environment, the desired properties were achieved.

Keywords
OAuth, OAuth2, Authentication, JWT, API, PHP, Nets, Signicat, SAML

SISÄLTÖ

1 JOHDANTO

Työn toimeksiantajana on NettiTieto Oy. NettiTieto on vuonna 1999 perustettu täyden palvelun ohjelmistotalo, jonka kotipaikkana toimii Kuopio. Ajatus työlle tuli EU:n yleisen tietosuoja-asetuksen GDPR:n (General Data Protection Regulation) sekä maksupalveludirektiivi PSD2:n (Payment Services Directive) tuomasta tarpeesta käyttäjien vahvaan tunnistamiseen.

Lisäksi yrityksellä oli tarve sovellukselle, jonka kautta sen omia ja sen asiakkaiden vahvan tunnistautumisen tapahtumia voitaisiin välittää tietoturvallisesti. Tätä sovellusta kutsutaan työssä myöhemmin nimellä autentikointiportaali. Autentikointiportaalin tarkoitus on toimia välikätenä asiakassovelluksen sekä vahvan tunnistautumisen tapahtumien tarjoajan välillä: autentikointiportaalia käyttämällä asiakassovelluksien ei esimerkiksi tarvitse tehdä hankalahkoja tarkistuksia tiedon oikeellisuudesta, vaan se voi luottaa autentikointiportaalin tekemiin tarkastuksiin. Se voi myös logittaa tapahtumat niiden myöhempää tarkastelua varten, jolloin pystytään ehdottoman varmasti todentamaan tunnistamisen tapahtuneen. Koska asiakassovellus keskustelee vain autentikointiportaalin kanssa, se ratkaisee myös ongelmat mahdollisissa päivityksissä tunnistamistapahtumien tarjoajien rajapinnoissa: asiakassovelluksien ei tarvitse itse muuttaa mitään, vaan riittää kun rajapintakutsut päivitetään autentikointiportaaliin vastaamaan uusia vaatimuksia.

Tärkeimpiä vaatimuksia autentikointiportaalille olivat sen helppo käytettävyys sekä korkea tietoturvan taso. Lisäksi huomioon oli otettava sen laajennettavuus jälkikäteen: sekä uusien asiakkaiden että tunnistautumistapahtumien tarjoajien lisäämisen oli oltava yksinkertaista.

2 AUTENTIKOINTI

Autentikointi, eli käyttäjän identiteetin varmistaminen, on yksi modernin internetin kulmakiviä. Autentikointia ei tule sekoittaa termiin autorisointi, joka tarkoittaa käyttäjän oikeuksien tarkistamista. Yhä useammalla internetpalvelulla on tarve varmistaa käyttäjänsä identiteetti. Autentikointi ei kuitenkaan terminä rajoitu pelkästään ihmisten henkilöllisyyden tunnistamiseen, vaan vähintään yhtä tärkeää monelle internetpalvelulle on myös autentikoida koneiden välisen kommunikoinnin toinen osapuoli. Siksi myöhempää ajatellen on tärkeää huomata myös ero termeissä käyttäjä ja loppukäyttäjä: käyttäjällä tarkoitetaan yleensä aina loppukäyttäjän puolesta toimivaa tietojärjestelmää, kun taas loppukäyttäjällä tarkoitetaan ihmistä.

Nykypäivänä lähes kaikki data yksittäisen henkilön henkilökohtaisista tiedoista aina yritysten liikesalaisuuksiin on internetissä saatavilla jossain muodossa. Ulkopuolisten pääsyn evääminen näihin tietoihin onkin todella suuri ongelma monelle yritykselle ja taholle, sillä vuotaessaan ne voivat aiheuttaa sekä yksilö- että yritystasolla erittäin suuria vahinkoja. Tästä syystä luotettavalle autentikoinnille on suuri kysyntä – yritysten ja muiden tahojen jotka käsittelevät salassapidettäviä tietoja on oltava täysin varmoja käyttäjän identiteetistä ennen kuin näitä tietoja luovutetaan eteenpäin tai niillä tehdään toimia, kuten tilisiirtoja tai muita pankkitapahtumia.

2.1 Lyhenteet ja määritelmät

JSON (JavaScript Object Notation) on tiedonvälitykseen tarkoitettu avoimen standardin tiedostomuoto.

Access Token on yleisesti rajapinnoissa käytettävä tieto, joka sisältää käyttäjän oikeudet tiettyyn resurssiin. Se voi olla periaatteessa missä muodossa tahansa: esimerkiksi satunnaisesti generoitu tekstimuuttuja. Tärkeintä on, että taho jolle access token toimitetaan ymmärtää sitä.

SOAP (aikaisemmin Simple Object Access Protocol) on tietoliikenneprotokolla, jonka pääasiallisena tehtävänä on mahdollistaa proseduurien etäkutsu (RPC eli Remote Procedure Call). Toimintaperiaatteeltaan SOAP on samankaltainen kuin muutkin RPC-protokollat, mutta sen vahvuuksiin lukeutuu XML-kieleen perustuminen sekä useiden eri protokollien yli toimiminen (W3C).

Rajapinta eli API on määritelmä jonka mukaan ohjelmisto voi tarjota palveluita tai tietoja muille sovelluksille.

REST (Representational State Transfer) on yksi arkkitehtuurimalleista rajapintojen toteutukseen.

Loppukäyttäjällä tarkoitetaan yleensä resurssien omistajaa eli ihmistä.

Käyttäjällä tarkoitetaan yleensä loppukäyttäjän puolesta tietoja pyytävää tietojärjestelmää.

IETF (the Internet Engineering Task Force) on internet-protokollien standardoinnista vastaava organisaatio.

W3C (World Wide Web Consortium) on kansainvälinen yritysten ja yhteisöjen yhteenliittymä, jonka tehtävänä on kehittää yhteisiä ja yhteensopivia webin pelisääntöjä ja teknologioita. W3C myös julkaisee ”web-suosituksia”, joka lyhyesti ilmaistuna tarkoittaa sitä, että kyseinen teknologia on pitkälle kehitetty ja testattu, ja sitä suositellaan käytettäväksi muiden vastaavanlaisten teknologioiden sijaan. Tällaisia suosituksia ovat muun muuassa HTML, CSS, SOAP ja XML.

3 OAUTH

OAuth on vuonna 2006 alkunsa saanut autentikointiprotokolla (OAuth.net). Sen ensimmäinen versio OAuth 1.0 julkaistiin vuonna 2007 ja se saavutti nopeasti standardin aseman alalla. Yleensä puhuttaessa pelkästä OAuthista tarkoitetaan kuitenkin sen uudempaa versiota, vuonna 2012 julkaistua OAuth2:ta, jota ei pidä sekoittaa edeltäjäänsä. Se julkaistiin ratkaisemaan aiemman version ongelmat ja rajoitteet, kuten sen skaalautuvuus laajempaan käyttöön sekä sen toteutuksen monimutkaisuus. Molemmat versiot toimivat pitkälti samalla periaatteella ja näkyvät loppukäyttäjälle hyvin samalla tavalla, mutta niiden implementoinnissa on eroja.

OAuthin tärkein ominaisuus on, että loppukäyttäjän ei tarvitse paljastaa tunnuksiaan kolmannen osapuolen sovellukselle, vaan tämän tarvitsee kirjautua vain varmasti luottamaansa palveluun (OAuth.net). Samassa kirjautumistapahtumassa loppukäyttäjä myös näkee palvelulle annettavat tiedot. Tällöin palveluntarjoaja saa vaihdossa access-tokenin, joka sisältää tiedon mitä valtuuksia sovellukselle on annettu. Palveluntarjoaja voi nyt käyttää tätä access-tokenia tehdessään toimia käyttäjän puolesta. Esimerkki eräänlaisesta access tokenista ja sen käytöstä on esitetty koodiesimerkissä 1.

Karkeasti molempien versioiden toimintatapa voidaan jakaa kahteen osaluokkaan: 2-legged flow tai 3-legged-flow, jotka tarkoittavat käytännössä koko prosessin osapuolten määrää: ”kolmijalkaisessa” osapuolina toimivat loppukäyttäjä, autentikointiserveri sekä palvelu johon loppukäyttäjä on autentikoitumassa. Jos loppukäyttäjä poistetaan yhtälöstä, puhutaan ”kaksijalkaisesta” eli koneiden välisestä autentikointiprosessista: tämä toteutaan versiossa 2.0 ”client-credentials”-tavalla, josta kerrotaan tarkemmin myöhemmässä luvussa.

```
GET /resource/1 HTTP/1.1
Host: example.com
Authorization: Bearer mF_9.B5f-4.1JqM
```

https://tools.ietf.org/html/rfc6749

KOODIESIMERKKI 1: Esimerkki OAuth2 Bearer-tokenista ja sen käytöstä

3.1 OAuth 1.0

Tässä työssä käsitellään tarkemmin vuonna 2009 julkaistua OAuth 1.0a:ta, joka julkaistiin paikkaamaan alkuperäisestä versiosta löytyneet turvallisuusongelmat. Vaikka useat yritykset ovatkin nykypäivänä siirtyneet käyttämään uudempaa OAuth2-protokollaa, OAuth1.0a ei missään tapauksessa ole vanhentunut ja on edelleen paljolti käytetty.

Versiossa 1.0a protokollan osapuolina toimivat kuluttaja, loppukäyttäjä ja palveluntarjoaja. Tässä tapauksessa palveluntarjoajalla tarkoitetaan sovellusta, joka tunnistaa käyttäjän ja prosessin loppuvaiheessa palauttaa kuluttajalle access tokenin. Kuluttajalla tarkoitetaan sovellusta joka pyytää resursseja loppukäyttäjän puolesta palveluntarjoajalta (OAuth Core Workgroup, 2009). Kuluttajan tulee rekisteröityä palveluntarjoajalle ennen kuin se voi alkaa toteuttamaan protokollan mukaista prosessia.

Prosessin kulku on seuraavanlainen:

1. Kuluttaja pyytää palveluntarjoajalta request tokenin
2. Kuluttaja ohjaa loppukäyttäjän palveluntarjoajan palveluun välittäen samalla em. request tokenin
3. Palveluntarjoaja tunnistaa loppukäyttäjän, pyytää tältä oikeutusta resursseihin kuluttajan puolesta ja ohjaa tämän takaisin kuluttajan palveluun välittäen samalla em. request tokenin.
4. Kuluttaja pyytää palveluntarjoajalta access tokenin
5. Saamallaan access tokenilla kuluttaja pääsee loppukäyttäjän resursseihin tai voi tehdä toimia tämän puolesta, access tokenin rajoitusten mukaisesti

Viestintä kuluttajan ja palveluntarjoajan välillä tapahtuu käyttämällä allekirjoitettuja pyyntöjä (OAuth Core Workgroup, 2009). Tämä tehdään, että palveluntarjoaja pystyy varmistumaan kuluttajan identiteetistä. OAuth ei dokumentaatiossaan ota kantaa siihen, miten allekirjoitus tulee toteuttaa, vaan tämä on aina palveluntarjoajan päätettävissä.

3.2 OAuth2

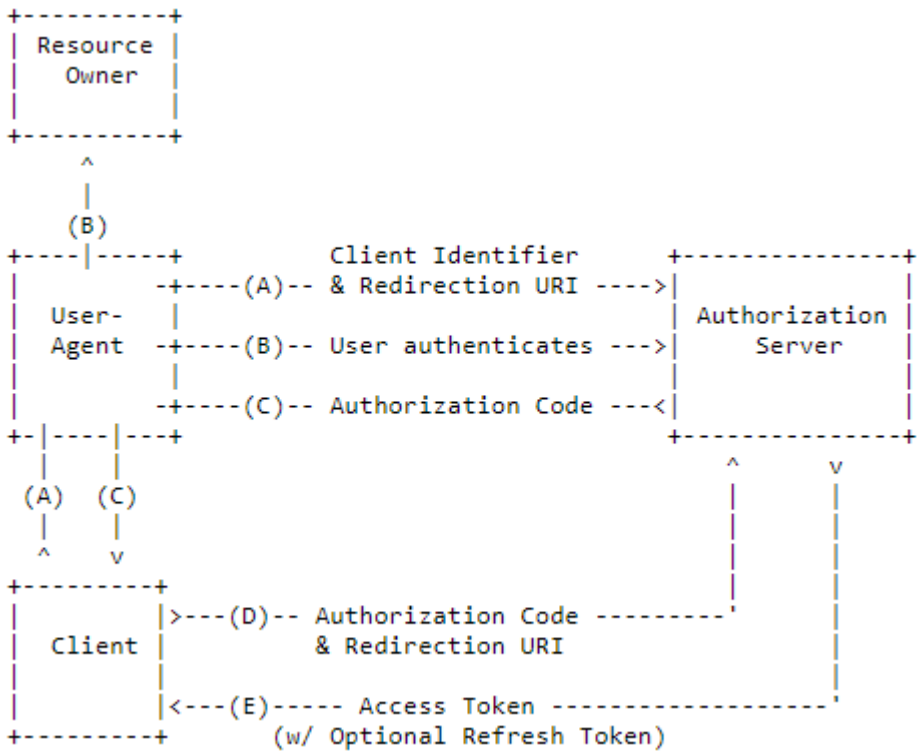
Vuonna 2012 julkaistu OAuth2 on todella laajasti käytössä oleva autorisointiprotokolla, jota käyttävät muun muassa webin suuryhtiöt Google, Microsoft ja Facebook. Sitä kehittää ja ylläpitää IETF:n OAuth-työryhmä. Toisin kuin edeltäjänsä, se luottaa TLS (Transport Layer Security) -salausprotokollaan viestin välityksen salaamiseksi. Vaikka virallisesti OAuth2 onkin autentikointi-ohjelmistokehys (Internet Engineering Task Force, 2012), se oikeastaan ottaa kantaa vain siihen, miten autorisointi tulee toteuttaa, ja jättää autentikoinnin toteutuksen autentikointipalvelun huoleksi.

Virallinen OAuth2-spesifikaatio (Internet Engineering Task Force, 2012) määrittelee seuraavanlaiset roolit:

1. Resurssien omistaja (yleensä loppukäyttäjä)
2. Resurssien haltija
3. Palvelu
4. Autentikointipalvelu

Ennen kuin palvelu voi toteuttaa autentikointipalvelun tarjoamaa OAuth-rajapintaa, sen täytyy rekisteröityä autentikointipalvelulle. Samalla palvelun tulee vähintään asettaa osoite, johon loppukäyttäjä palautetaan autentikointipalvelusta sekä kertoa onko se luotettavaa (confidential) vai julkista (public) tyyppiä, eli pystyykö se pitämään salassa autentikointipalvelun määrittämän salaisen avaimen.

Jos autentikointipalvelun tarjoaja tämän mahdollistaa, voi olla myös mahdollista kustomoida loppukäyttäjän näkymä autentikointivaiheessa tai määrittää muita asetuksia.
Spesifikaatiossa määritellään myös erilaiset grantit eli käyttötavat seuraavasti:
Authorization Code-grant on luonteva valinta sellaisille palveluille, jotka pystyvät pitämään salassa autentikointipalvelun määrittämän salaisen avaimen. Tämän grantin kulku on kuvattu kuvassa 1. Tämä grant on myös kaikista turvallisin, sillä loppukäyttäjälle (tai tämän käyttämälle selaimelle) ei missään vaiheessa paljasteta access tokenia, ja autentikointipalvelu voi tunnistaa myös pyynnön tehneen palvelun.



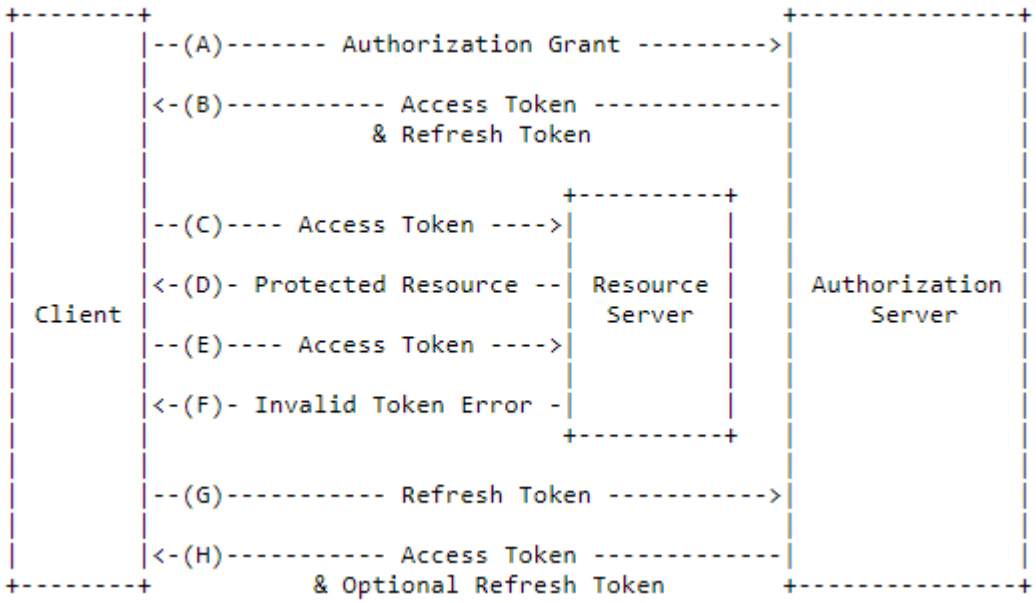
Kuva 1 - OAuth2 tunnistautumisen kulku – Authorization Code Flow (Hardt 2012)

Implicit-grant on samantyyppinen authorization code-grantin kanssa, kuitenkin sillä erolla, että se on tarkoitettu käytettäväksi sovelluksille, jotka eivät voi pitää salassa palvelun salaista avainta: esimerkiksi puhtaasti JavaScriptillä toteutetut web-sovellukset. Näin ollen em. authorization-code grantista eroten ei myöskään ole tarvetta vaiheelle, jossa authorization code vaihdetaan access tokeniksi, vaan autentikointipalvelu voi suoraan palauttaa access tokenin. Tätä grantia käytettäessä autentikointipalvelun ei myöskään tule palauttaa refresh tokenia, koska palvelu ei voi sitä pitää salassa.

Authorization Code- sekä Implicit-granteista eroten, Resource Owner Password Credentials-grantia käytettäessä loppukäyttäjä antaa tunnuksensa suoraan palveluntarjoajalle, joka välittää ne autentikointipalvelulle. Tämä tarkoittaa, että autentikointipalvelun on pystyttävä luottamaan palveluntarjoajaan täysin. Myös loppukäyttäjän tulee luottaa palveluntarjoajaan. Mahdollisia käyttökohteita ovat muun muuassa sovellukset, joissa palveluntarjoaja sekä autentikointipalvelun tarjoaja ovat sama taho. Tätä grantia tulee käyttää vain, jos muut grantit eivät ole sovellettavissa (Internet Engineering Task Force, 2012). Tätä grantia käytettäessä autentikointipalvelu voi suoraan palauttaa loppukäyttäjän access tokenin sekä mahdollisesti refresh tokenin.

Spesifikaation yksinkertaisin grant eli Client Credentials-grant on tehty tietojärjestelmien väliseen autentikointiin: palvelu lähettää tunnuksensa autentikointipalvelulle, joka palauttaa palvelulle tarkoitetun access tokenin. Autentikointipalvelun on mahdollista palauttaa myös refresh token. (Internet Engineering Task Force, 2012.)

Kuten edellä mainittiin, joitain granteja käytettäessä autentikointipalvelun on mahdollista palauttaa access tokenin lisäksi myös refresh token, jolla palvelu voi uusia access tokenin loppukäyttäjän puolesta: tämä tarkoittaa sitä, että välttämättä uutta tunnistautumista käyttäjältä ei tarvita. Tämä on hyödyllistä esimerkiksi niissä tilanteissa, kun alkuperäinen access token on jo vanhentunut tai sen antamia oikeuksia (scopeja) halutaan vähentää. Refresh tokenin tarkoitus on vähentää access tokenien keston väärinkäyttöä: aikaisemmissa OAuth-versioissa niiden kesto saattoi olla jopa ikuinen: ainakin siihen asti kunnes loppukäyttäjä itse peruutti ne.



Kuva 2 - OAuth2 Refresh tokenin käyttö (Hardt 2012)

3.3 OIDC

OIDC eli OpenID Connect on OAuth2 päälle rakennettu autentikointitaso, jota säätelee OpenID säätiö. Sen tärkeimpiä lisäyksiä OAuth2-protokollaan ovat loppukäyttäjän autentikointiin tarkoitettu ID Token-datastruktuuri, joka sisältää tietoja loppukäyttäjän autentikointitapahtumasta, sekä autentikoinnin mahdollistavat tavat eli flow:t. Lisäksi OIDC mahdollistaa autentikointipalvelun asetusten automaattisen haun käyttämällä discovery-palvelua. (OpenID Foundation, 2014.)

OIDC mahdollistaa autentikoinnin kolmella eri tavalla, joita ovat authentication code-flow, implicit flow sekä hybrid-flow. Näistä käytetyin on authentication code-flow (OpenID Foundation, 2014).

Kuten OAuth2 Authorization Code-grant, myös OIDC Authorization Code flow on vaihtoehdoista turvallisin: tokeneita ei tarvitse paljastaa loppukäyttäjälle (/selaimelle) sekä autentikointipalvelu voi tunnistaa myös pyynnön tehneen palvelun. Myös Implicit flow mukailee vahvasti OAuth2:n vastaavan nimistä granttia. Molempien em. flowien mukainen tunnistautumisen kulku onkin samanlainen kuin OAuth2-granttien kulku.

Hybrid-flow on vähiten käytetty, koska sen käyttötarkoitukset ovat hyvin rajattuja. Se on yhdistelmä kahdesta em. flowsta, jossa autentikointipalvelu voi tunnistaa pyynnön tehneen palvelun, mutta tämän toteuttaakseen sen on paljastettava saamansa tokenit loppukäyttäjälle tai tämän selaimelle. Hybrid flow mahdollistaa sen, että palvelun serveri-puolelle sekä sen asiakkaalle näkyvälle puolelle haetaan omat, erilliset access tokenit.

3.3.1 OIDC – Discovery

OIDC lisää OAuth2-protokollaan myös niin sanotun discovery-palvelun, jonka avulla autentikointipalvelun käyttäminen on palvelulle yksinkertaisempaa: sen sijaan että autentikointiprosessissa tarvittavat osoitteet, kuten authentication endpoint (osoite, johon loppukäyttäjän selain ohjataan prosessin alussa), olisivat tallessa jossain (esimerkiksi palvelun tietokannassa, config-tiedostossa tai vastaavassa), se voi hakea ne yksinkertaisella HTTP GET-pyynnöllä.


```
{"response_types_supported":["code"],"request_parameter_suppor-
ted":true,"request_uri_parameter_supported":false,"claims_parame-
ter_supported":false,"grant_types_supported":["authoriza-
tion_code"],"ui_locales_suppor-
ted":["nb_NO","en_GB","da_DK","fi_FI","sv_SE","sv_FI"],"scopes_sup-
ported":["openid","profile","ssn","cert"],"is-
suer":"https://www.ident-preprod1.nets.eu/oidc","authoriza-
tion_endpoint":"https://www.ident-preprod1.nets.eu/its/in-
dex.html","claims_supported":["amr","certpolicyoid","cn","dn","birth-
date","certificate","givenname","surname","notbefore","notaf-
ter","c","no_ssn","se_ssn","dk_ssn","no_bid_pid","dk_dan_pid","se_bid
_securitylevel","fi_tu-
pas_bank","aud","sub","iss","exp","iat","nonce","auth_time","dk_dan_r
id","birthdate","no_cel8","fi_ssn","fullname","fi_tu-
pas_pid","fi_trx_code","fi_verification_code"],"require_re-
quest_uri_registration":false,"jwks_uri":"https://www.ident-prep-
rod1.nets.eu/oidc/.well-known/jwk","subject_types_supported":["pub-
lic","pairwise"],"id_token_signing_alg_values_suppor-
ted":"RS256","claim_types_supported":["normal","aggregated","distri-
buted"],"token_endpoint_auth_methods_supported":["client_secret_ba-
sic","client_secret_post"],"token_endpoint":"https://www.ident-prep-
rod1.nets.eu/oidc/token"}
```


https://www.ident-preprod1.nets.eu/oidc/.well-known/openid-configuration

KOODIESIMERKKI 2 – OIDC-palvelun palauttamat tiedot

Tällöin useamman osoitteen sijaan voidaan tallentaa ja ylläpitää vain yksi osoite, josta muut tiedot haetaan aina tarvittaessa. Samalla voidaan varmistua niiden ajantasaisuudesta. Esimerkki erään OIDC-palvelun palauttamasta tiedosta koodiesimerkissä 2.

3.3.2 ID Token

Ehkä tärkein OIDC:n tuoma lisä OAuth2-protokollaan on ID Token, joka on datastruktuuri joka sisältää tietoja loppukäyttäjän autentikointitapahtumasta. Pakolliset tiedot, joita ID Tokenin tulee sisältää (OpenID Foundation, 2014), on esitetty taulukossa 1 ja vapaaehtoiset kentät taulukossa 2.

TAULUKKO 1 – OIDC ID Tokenin pakolliset kentät

iss	ID Tokenin palauttava taho	PAKOLLINEN
sub	Palauttavan tahon tunniste loppukäyttäjästä.	PAKOLLINEN
aud	Kenelle ID Token on tarkoitettu käytettäväksi.	PAKOLLINEN
exp	Vanhenemisaika. Tämän ajan jälkeen ID Tokenia ei tule käyttää.	PAKOLLINEN
iat	ID Tokenin luomisaika.	PAKOLLINEN
auth_time	Aika jolloin loppukäyttäjän tunnistaminen on tapahtunut. Pakollinen max_age-pyynnön yhteydessä tai jos auth_time on määritetty autentikointi-pyynnössä pakolliseksi tiedoksi.	JOSKUS

TAULUKKO 2 – OIDC ID Tokenin vapaaehtoiset kentät

nonce	Replay-hyökkäyksien estoon on mahdollista käyttää tätä tietoa: palvelu lähettää randomisoidun tiedon tässä kentässä, jonka autentikointipalvelu palauttaa muuttumattomana. Palvelun tulee verrata lähettämänsä tieto samaksi kuin vastaanottamasa.	EI PAKOLLINEN
acr	Autentikointitapahtuman vahvuus. Arvo 0 tarkoittaa sitä, että auten-	EI PAKOLLINEN

	tikointitapahtuma ei ole saavuttanut ISO/IEC 29115-standardin mukaisia vaatimuksia ((https://www.iso.org/standard/45138.html). Muut arvot tulee sopia autentikointipalveluntarjoajan sekä palvelun välillä.	
amr	Taulukko käytetyistä autentikointimenetelmistä: sovitaan erikseen palveluntarjoajan ja autentikointipalvelun tarjoajan välillä. Esimerkiksi [tupas].	EI PAKOLLINEN
azp	Valtuutettu osapuoli. Käytetään vain, jos tämä kenttä on eri kuin ID Tokenin aud-kenttä.	EI PAKOLLINEN

Näiden kenttien lisäksi OIDC sallii myös muiden kenttien lisäyksen, joiden nimet, sisällöt sekä pakollisuuden palveluntarjoaja ja autentikointipalvelun tarjoaja voivat sopia keskenään.
ID Tokenin tulee olla aina JWT (Json Web Token) -formaatissa, jolloin sen todellisuus voidaan varmistaa käyttämällä JWT:n sisältämää signaturea. JWT:n tulee siis olla allekirjoitettu käyttäen JSON Web Signaturea (JWS). Mahdollista on myös salata tokenin sisältämä data käyttäen JSON Web Encryptionia (JWE), kuitenkin sen jälkeen, kun JWS-allekirjoitus on toteutettu (Internet Engineering Task Force, 2015).

4 SAML

SAML (Security Assertion Markup Language) on XML-pohjainen avoin standardi autentikointi- ja autorisointitietojen välittämiseen osapuolten välillä (OASIS). Siitä on olemassa kolme versiota: 1.0, 1.1 sekä 2.0. Näistä versiot 1.0 ja 1.1 ovat hyvin samankaltaisia (Mishra, 2003), mutta versio 2.0 ei ole enää näiden kanssa taaksepäin yhteensopiva (OASIS, 2008).

Yksi ongelma, jonka SAML ratkaisee, on SSO:n (Single Sign-On) toteuttaminen. SSO:lla tarkoitetaan selaimella tehtävää kertakirjautumista yhteen palveluun, jonka jälkeen sama autentikointitapahtuma on mahdollisesti pätevä myös samassa verkostossa oleville muille palveluille. Tarkoituksena on sujuvoittaa käyttäjän kokemusta eri palveluita käyttettäessä.

SAML-standardi määrittelee seuraavat roolit: loppukäyttäjä (käyttäjä, jolla tarve autentikoida itsensä palvelulle), palveluntarjoaja (palvelu, jolle loppukäyttäjä autentikoitumassa) sekä henkilöllisyydentarjoaja (taho, jonka antamiin tietoihin palveluntarjoaja luottaa). Henkilöllisyydentarjoaja antaa palveluntarjoajalle tiedon loppukäyttäjän autentikointitapahtumasta. Standardi myös määrittelee seuraavat käsitteet:

SAML Binding määrittelee miten palveluntarjoaja ja henkilöllisyydentarjoaja keskustelevat keskenään, esimerkiksi voidaan määrittää käytettäväksi SOAP-protokolla. SAML 2.0 esitteli myös muita bindingejä, esimerkiksi HTTP:n yli tehtävät POST-kutsut.

SAML Assertion on yleensä henkilöllisyydentarjoajan kokoama tietopaketti, joka sisältää tietoa loppukäyttäjästä XML-muodossa. Esimerkki yhdestä SAML 2.0-assertionista on koodiesimerkissä 3.

SAML Protocol määrittelee, kuinka SAML-elementit sisällytetään SAML-viesteihin.

SAML Profile on standardin määrittelemä käyttötapaus: tärkeimmät näistä ovat Web Browser SSO Profile sekä Enhanced Client or Proxy (EPC) Profile (OASIS, 2005). SAML Profilet määrittävät sen miten em. SAML Bindingit, SAML Assertionit sekä SAML Protocol:t yhdistetään yhdeksi käyttötapaukseksi. SAML 2.0 esitteli myös monia uusia SAML Profileja, esimerkiksi Single Logout Profilen, joka ratkaisi SSO:n yhtenäisen uloskirjautumisen.

```
<saml:Assertion xmlns:saml=”urn:oasis:names:tc:SAML:2.0:assertion”
     Version="2.0"
     IssueInstant="2005-01-31T12:00:00Z">
     <saml:Issuer Format=urn:oasis:names:SAML:2.0:nameid-format:entity>
         http://idp.example.org
     </saml:Issuer>
     <saml:Subject>
         <saml:NameID
             Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
                 j.doe@example.com
         </saml:NameID>
     </saml:Subject>
    <saml:Conditions
        NotBefore="2005-01-31T12:00:00Z"
        NotOnOrAfter="2005-01-31T12:10:00Z">
    </saml:Conditions>
    <saml:AuthnStatement
        AuthnInstant="2005-01-31T12:00:00Z" SessionIndex="67775277772">
        <saml:AuthnContext>
            <saml:AuthnContextClassRef>
                urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtected-
Transport
            </saml:AuthnContextClassRef>
        </saml:AuthnContext>
    </saml:AuthnStatement>
</saml:Assertion>
```

https://www.oasis-open.org/committees/download.php/27819/sstc-saml-tech-overview-2.0-cd-02.pdf

KOODIESIMERKKI 3 – Esimerkki SAML 2.0 Assertionista

5 JWT

JWT eli JSON Web Token on JSON-pohjainen avoin standardi käyttöoikeustokeneiden hallinnoimiseen eri järjestelmien, sovellusten ja ohjelmistojen välille, jonka suurimpia etuja ovat sen helppokäyttöisyys ja keveys: sen muodostaminen onnistuu lähes millä tahansa ohjelmistokehyksellä ja kehityskielellä, ja se sisältää itsessään lähes kaiken tarvittavan käyttäjän autentikointiin, eli esimerkiksi tietokantakysely tai istunnon tietojen hakeminen ei välttämättä ole tarpeen käyttäjän esittämällä pyynnöllä.

JWT:t voivat olla joko JWS (JSON Web Signature) tai JWE (JSON Web Encryption) -tyyppisiä olioita, riippuen siitä onko JWT:n sisällä liikkuva data salattua vai ei – JWS-tyyppiä käytettäessä data liikkuu muunnettuna base64-muotoon, kun taas JWE-tyyppiä käytettäessä data salataan käyttäen jotain JWA (JSON Web Algorithms) -spesifikaatiossa määriteltyä algoritmia (Internet Engineering Task Force, 2015).

5.1 JWS

JWS-tyyppiset JWT:t koostuvat kolmesta osasta: JOSE (Javascript Object Signing And Encryption) -headerista, JWS payloadista sekä JWS signaturesta. Niiden sisältö noudattaa tiettyä kaavaa: JOSE-header-osuus kertoo signaturessa käytetyn algoritmin, sekä mahdollisesti JWS:lle ominaisia tietoja, kuten signaturen varmistamiseen tarvittavan avaimen tunnuksen. Payload sisältää itse datan eli claimin: esimerkiksi claimissa voi olla tieto, että tämän tokenin omistajalla on admin-tason oikeus. Payloadiin sisältyy monesti myös ”iat” (issued at) -kenttä, joka kertoo milloin tämä token on luotu. Spesifikaatiossa ei kuitenkaan määritetä mitään kenttää pakolliseksi tiedoksi, vaan claim voi sisältää käytännössä mitä tietoja tahansa (IETF 2015). Signature-kentän sisältö muodostuu seuraavasti: aluksi sekä header- että payload-osuudet muunnetaan base64-muotoon (erillään toisistaan). Tämän jälkeen kentät liitetään yhteen käyttäen pistemerkkiä erottimena. Tuloksena saatu string-tyyppinen muuttuja salataan sen jälkeen headerissa määritellyllä algoritmilla. Lopullinen JWT muodostuu muuttamalla kaikki kolme muodostettua osuutta base64-muotoon ja liittämällä ne toisiinsa käyttäen pistemerkkiä erottimena järjestyksessä header, payload, signature. Esimerkki JWS:n muodostamisesta on esitelty koodiesimerkissä 4.

Signaturen tarkoitus on varmistaa JWT:n lähde: jos signaturen varmistaminen onnistuu, voidaan varmistua siitä, että JWT on varmasti peräisin oikealta taholta eikä esimerkiksi hyökkääjältä joka yrittää kirjautua admin-käyttäjänä tekaistulla tokenilla (Auth0). Siksi onkin tärkeää varmistaa tokenin signature aina ennen kuin payloadin tietoja aletaan käsitellä tarkemmin. Mahdollista on myös muodostaa JWE-tyyppinen olio, joka allekirjoitetaan käyttäen JWS:a. Tämä takaa sen, että data liikkuu turvallisesti lähettäjän ja vastaanottajan välillä, ja että se on peräisin oikealta taholta.

```
header = '{"alg":"HS256","typ":"JWT"}'
payload = '{"loggedInAs":"admin","iat":1422779638}'
key           = 'secretkey'
unsignedToken = encodeBase64Url(header) + '.' + encodeBase64Url(payload)
signature     = HMAC-SHA256(key, unsignedToken)

token = encodeBase64Url(header) + '.' + encodeBase64Url(payload) + '.' +
encodeBase64Url(signature)
# token is now: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJsb2dnZWRJbkFzI-
joiYWRtaW4iLCJpYXQiOjE0MjI3Nzk2Mzh9.gzSra-
SYS8EXBxLN_oWnFSRgCzcmJmMjLiuyu5CSpyHI
```

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

KOODIESIMERKKI 4 – JWS:n muodostaminen

5.2 JWK

JWK eli JSON Web Key on JSON-muotoinen tietopaketti, joka sisältää tietoa kryptologisesta avaimesta. Sitä käytetään muun muassa JWS:n oikeellisuuden varmistamiseen (Internet Engineering Task Force, 2015). JWK:n sisältämät tiedot ovat esitettynä taulukossa 3. Näiden tietojen lisäksi on mahdollista sisällyttää muita, algoritmikohtaisia tietoja (Internet Engineering Task Force, 2015).

Koodiesimerkissä 5 on esitetty yksi RSA-tyyppinen JWK, jota on käytetty OIDC-protokollan mukaisen JWT:n signaturen varmistamiseen. Se sisältää taulukossa 3 määritettyjen kenttien lisäksi RSA-avaimelle tyypillisiä kenttiä (”e” ja ”n”-kentät).

```
{
"kty":"RSA",
"e":"AQAB",
"x5c":
[
"MIIDUTCCAjmgAwIBAgIEFePjxTANBgkqhkiG9w0BAQsFADBZMQswCQYDVQQGEwJOT-
zEPMA0GA1UECBMGTm9yd2F5MQ0wCwYDVQQHEwRvc2xvMQ0wCwYDVQQKEwRORVRTMQ0wCwY-
DVQQLEwRORVRTMQwwCgYDVQQDEwNTSVMwHhcNMTcwNjE2MDcyNDMzWhcNMjEwNjE1MDcyNDMzWjBZM-
QswCQYDVQQGEwJOTzEPMA0GA1UECBMGTm9yd2F5MQ0wCwYDVQQHEwRvc2xvMQ0wCwYDVQQKE-
wRORVRTMQ0wCwYDVQQLEwRORVRTMQwwCgYDVQQDEwNTSVMwggEiMA0GCSqGSIb3DQEBAQUAA4IB-
DwAwggEKAoIBAQCaacFPmzkhwS1/Qz4+aMYN-
fED4yI6dRty+dUA5GL+wCpAk7XrsvD8xB5lB0kNZs0RL0PmyuEIoZZyLYzRse8Ax2JTLE1BauK-
Zhbg+k/2NLDbKCDdBm6CZfo-
jLWPTFkRniY1B0XM9TbBRNTSu5PFrGl3YitQ27tKXHoPdxMhIQt9w32jS/JgFJRpCPHzzy8QMBMb-
```

```
BYpo6dwk5IQ3Uv6CbYSsbD0KL15zjtnZWn/Q6Cao+I4P5jIUY97amHZSFc64jApJYtOzCdy-
vQR1JkKvOkhdxNVywbEi0zdqozPNFFH6Wm/s2oEsrTGLoqdneVatnzgP2dpKSFpNy7UKRJppX/BhAg-
MBAAGjITAfMB0GA1UdDgQWBBTELlgXo6I3oJdNIuzM8glCKIhJCTANBgkqhkiG9w0BAQs-
FAAOCAQEALnMNONaA06XSPS2iQoC54AvHED+CglIgmbdD0GWlop-
TtFb07qc86NPiE9AS9iIBjPUNAe51CQH+4soOI/8meoSZmgpU12S98fmVDBGf+hz4+x3UKup-
lUoZV08CH1GTekfRQdYv1LJ0lCsgtcVvgigRALOvyNf0RepBOXQS2uxmw5KtYTyTjSwXY-
rqZ0QBiHuhNORRLpLb0G2EboGVeiv0imOnN/uDtRfM0Dea3Id+NwsOK48445qEx-
mXW7M1g8QDysWUjE48MtySpIo5iFDL1HxKzPvmUxjg6QuMXIzvhw+71aoTzumt+/WBdjU-
dOfDMl7lGn0DBCSOFfQCatdMTQA=="
],
"kid":"367256517",
"n":"mmnBT5s5IcEtf0M-PmjGDXxA-MiOnUbcvnVAORi_sAqQJO167Lw_MQeZ-
QdJDWbNES9D5srhCKGWci2M0bHvAMdiUyxNQWri-
mYW4PpP9jSw2ygg3QZugmX6Iy1j0xZEZ4mNQdFzPU2wUTU0ruTxaxpd2IrUNu7Slx6D3cTI-
SELfcN9o0vyYBSUaQjx888vEDATGwWKaOncJOSEN1L-gm2ErGw9Ci9ec47Z2Vp_0OgmqPiOD-
YyFGPe2ph2UhXOuIwKSWLTswncr0EdSZCrzpIXcTVcsGxItM3aqMzzRRR-
lpv7NqBLK0xi6KnZ3lWrZ84D9naSkhaTcu1CkSaaV_wYQ"}
```


KOODIESIMERKKI 5 – Esimerkki JWK-avaimesta.

TAULUKKO 3 – JWK:n sisältämät tiedot

Parametri	Selite	Pakollisuus
kty	Avaimen tyyppi (key type)	Kyllä
use	Mihin avainta käytetään: joko ”sig” (signature) tai ”enc” (encryption)	Ei
key_ops	Mitä avaimella tehdään (key operations)	Ei
alg	Käytetty algoritmi	Ei
kid	Avaimen id (key id)	Ei
x5u	Osoite, josta avaimelle kuuluva X.509-sertifikaatti on saatavilla (X.509 URL)	Ei
x5c	X.509-sertifikaattiketju (X.509 Certificate Chain)	Ei
x5t	X.509-sertifikaatti SHA-1-lyhenne (X.509 Certificate SHA-1 Thumbprint)	Ei
x5t#S256	X.509-sertifikaatti SHA-256-lyhenne (X.509 Certificate SHA-256 Thumbprint)	Ei

6 VAHVA TUNNISTAUTUMINEN

Vahvalla tunnistautumisella tarkoitetaan ihmisen henkilöllisyyden todentamista sähköisesti. Vahvan tunnistautumisen avulla yritykset ja muut tahot voivat varmistua kuluttajan henkillölisyydestä. Myös kuluttaja voi turvallisesti vahvistaa henkilöllisyytensä sähköisille palveluille. Suomessa yleisimmät tunnistusvälineet ovat pankkien käyttämät verkkopankkitunnukset (TUPAS) sekä teleoperaattoreiden (Telia, Elisa & DNA) tarjoama mobiilivarmenne. Lisäksi on myös Väestörekisterikeskuksen tarjoama kansalaisvarmenne, jota ei tässä työssä käsitellä sen vähäisen käytön takia (Väestörekisterikeskus).

Vahva tunnistautuminen perustuu aina vähintään kahteen todentamistapaan kolmesta eri todentamistekijästä (Laki vahvasta sähköisestä tunnistamisesta ja sähköisistä luottamuspalveluista, 2009), jotka ovat:

1. Tiedossa oloon perustuva todentamistekijä
   -Esimerkiksi käyttäjätunnus & salasana, PIN-koodi
2. Hallussapitoon perustuva todentamistekijä
   -Esimerkiksi puhelin, tunnuslukukortti, fyysinen autentikointilaite
3. Luontainen todentamistekijä, joka perustuu johonkin luonnollisen henkilön fyysiseen ominaisuuteen
   -Esimerkiksi sormenjälki, iiris-skanneri

6.1 PSD2

PSD2 on EU-tason maksupalveludirektiivi, joka tuli voimaan tammikuussa 2018. Yksi sen myötä tulleista muutoksista on velvollisuus tunnistaa asiakas vahvasti, jos tämän maksutiliä käytetään sähköisesti tietoverkon välityksellä, tehtäessä sähköisiä maksutapahtumia tai toimenpiteitä, joihin liittyy petoksen tai väärinkäytöksen riski. (Finanssiala Ry, 2018)

PSD2 myös pakottaa pankit antamaan kolmansien osapuolien palveluntarjoajille mahdollisuuden päästä käsiksi pankin asiakkaiden maksutapahtumiin sekä tilitietoihin – tietysti vain asiakkaan näin halutessa. Käytännössä tämä tarkoittaa, että pankkien on avattava kolmansille osapuolille avoin rajapinta.

6.2 TUPAS

Suomessa käytetyistä vahvan tunnistautumisen välineistä TUPAS on ehdottomasti käytetyin kattaen jopa yli 90% kaikista tunnistautumiskerroista (Lehto, 2014). TUPAS-tunnukset ovat käytössä yli neljällä miljoonalla suomalaisella (Lehto, 2014). Alun perin suomalaisten pankkien kehittämän tunnistautumisjärjestelmän tavaramerkin omistaa tätä nykyä Finanssiala Ry, joka myös hallinnoi, kehittää ja ylläpitää siihen liittyvää dokumentaatiota ja määritystä.

Tämän työn kirjoitushetkellä TUPAS-tunnistautuminen ei täytä vahvan sähköisen tunnistamisen vaatimuksia uudistetun EU-sääntelyn myötä, mutta sillä on siirtymäaikaa syyskuuhun 2019 asti tehdä tarvittavat muutokset (Viestintävirasto). Finanssiala Ry on kuitenkin ilmoittanut, ettei TUPAS-protokollaa tulla enää kehittämään vahvan tunnistuksen vaatimuksia täyttäväksi, vaan todennäköisimmin sen sijaan siirrytään käyttämään

SAML- tai OIDC-protokollia (Viestintävirasto, 2018). TUPAS-palvelun rajapinnan toteutus ei myöskään ole enää mahdollista yksittäiselle kolmannen osapuolen asiointipalvelulle suoraan, vaan tunnistamispalvelut on ostettava luottamusverkostosta (Finanssiala Ry).

Vaikka TUPAS onkin alun perin suomalaisten pankkien kehittämä tunnistautumisjärjestelmä, sitä käytetään laajasti myös muihin palveluihin tunnistautumiseen: muun muassa Verohallinto sekä Kansaneläkelaitos tarjoavat mahdollisuuden tunnistautua palveluihinsa käyttäen TUPAS-tunnistautumista.

6.3 Mobiilivarmenne

Mobiilivarmenne on kolmen matkapuhelinoperaattorin (DNA, Elisa ja Telia) tarjoama sähköinen tunnistuspalvelu, joka liitetään matkapuhelimen SIM-korttiin. Se toimii sähköisenä henkilöllisyystodistuksena samaan tapaan kuin em. TUPAS-tunnistautuminen. Mobiilivarmenne täyttää vahvan tunnistautumisen kriteerit vaatimalla tietyn sim-kortin (hallussapito) sekä puhelimeen syötettävän PIN-koodin, jonka käyttäjä itse valitsee rekisteröityessään mobiilivarmenteen käyttäjäksi (tieto). Erillistä tunnuslukulistaa ei vaadita palvelun käyttämiseen. Mobiilivarmenne hyödyntää SAML-protokollaa viestien välittämiseen (Ficom Ry, 2014).

Tupas-tunnistautumisen tavoin myös mobiilivarmennetta voidaan käyttää muun muassa Kansaneläkelaitokselle ja Verohallinnolle tunnistautumiseen, vaikka sen ylläpidosta vastaavatkin matkapuhelinoperaattorit. Myöskään mobiilivarmenteen rajapinnan toteutus yksittäiselle kolmannen osapuolen asiointipalvelulle ei ole enää mahdollista, vaan tunnistamispalvelut on ostettava luottamusverkostosta (Finanssiala Ry).

7 AUTENTIKOINTIPORTAALI

Työn toimeksiantajan toiveena oli portaali, jonka kautta voisi reitittää kaikki sen omat sekä sen asiakkaiden tekemät vahvan tunnistautumisen tapahtumat. Tämä helpottaisi muun muassa tapahtumien tallentamista myöhempää tarkastelua varten, jolloin kaikki tunnistautumistapahtumat voitaisiin tallentaa yhteen, keskitettyyn paikkaan.

Myöhemmässä vaiheessa tarkoitus on myös liittää portaaliin niin sanottu allekirjoitustoiminnallisuus, jonka avulla vahvaa tunnistautumista voitaisiin käyttää hyväksi laillisesti sitovien dokumenttien allekirjoituksessa, kuten sopimuksissa. Portaalin tuli olla sekä helppo integroida asiakkaan puolelta että myös helposti laajennettavissa mahdollisesti myöhemmin tulevia asiakkaita tai tunnistuspalvelun tarjoajia varten.

7.1 Toimintaperiaate

Autentikointiportaali on toteutettu PHP-ohjelmointikielellä käyttäen apuna Laravel-ohjelmistokehystä. Sen toimintaperiaate on seuraavanlainen:

1. Palvelu kirjautuu autentikointiportaalin käyttäjäksi ja antaa sille tarvittavat tiedot.
2. Kun palvelulla on tarvetta käyttäjän vahvalle tunnistamiselle, se ohjaa käyttäjän autentikointiportaaliin.
3. Autentikointiportaali ohjaa käyttäjän tunnistuspalvelun tarjoajan palveluun, jossa tämä voi tunnistautua käyttäen haluamaansa tapaa (esimerkiksi Tupas).
4. Tieto käyttäjän tunnistautumistapahtumasta palautuu autentikointiportaalille, ja käyttäjä ohjataan takaisin alkuperäiseen palveluun. Palvelulle välitetään tieto käyttäjän tunnistautumistapahtumasta.

7.2 Nets

Nets Oy on luottamuspalveluiden tarjoaja, jonka palveluihin kuuluvat tunnistautumis- ja allekirjoituspalvelut. Nets on osa Luottamusverkostoa eli se on Viestintäviraston rekisteröimä tunnistuspalvelun tarjoaja. Nets ei tarjoa omaa tunnistusvälinettä, vaan se välittää muiden osapuolien tunnistusvälineitä oman palvelunsa kautta. Suomalaisista palveluista Nets tarjoaa asiakkailleen Tupas-tunnistautumista sekä Mobiilivarmennetta oman palvelunsa kautta. Netsin oma rajapinta tarjoaa vaihtoehdoksi SAML1.1- tai OpenID Connect-protokollan käytön, joista toteutettiin OpenID Connect-protokolla.

7.3 Signicat

Signicat on norjalainen vuonna 2007 perustettu luottamuspalveluiden tarjoaja, joka tarjoaa sähköistä tunnistautumis- ja välityspalvelua sekä allekirjoituspalvelua. Signicat on Viestintäviraston rekisteröimä tunnistuspalvelun tarjoaja, eli se on osa Luottamusverkostoa. Signicat välittää muiden osapuolien tunnistusvälineitä oman palvelunsa kautta asiakkailleen, eli sillä ei ole omaa välinettä loppukäyttäjän tunnistamiseen. Pohjoismaissa Signicatin tarjoamia tunnistusvälineitä ovat kaikki vahvat tunnisteet, kuten Suomen TUPAS-tunnistus. Tällä hetkellä Signicatin tarjoamia protokollia ovat SAML1.1, SAML2.0 sekä OpenID Connect. Myös Signicatin tapauksessa päädyttiin toteuttamaan OIDC-protokolla, jota ei kuitenkaan päästy testaamaan erinäisistä syistä johtuen.

7.4 Valmiit paketit

Autentikointiportaalin toteutusta varten selvitettiin monen erilaisen valmiin paketin käyttöä lähinnä rajapintojen toteutukseen. Selvitetyistä paketeista useimmissa oli erinäisiä puutteita tai muita ongelmia, joiden takia pakettia ei voitu ottaa käyttöön. OIDC:n toteuttamiseen liittyvissä paketeissa suurimmat ongelmat liityivät niiden kankeuteen tai dokumentaation puuttumiseen, näin ollen sekä Nets- että Signicat-rajapintateutukset tehtiin käyttämättä mitään valmista pakettia tai kirjastoa.

Kommunikaatioon autentikointiportaalin sekä asiakassovelluksen välillä päädyttiin käyttämään Laravelin Passport-pakettia, joka on OAuth2-paketti. Se tarjoaa tarvittavat työkalut koko OAuth2-prosessin toteuttamiseen serverin puolelle, luoden valmiiksi muun muassa tarvittavat tietokantataulut. Passport-paketti on myös erittäin helppokäyttöinen ja hyvin dokumentoitu.

8 YHTEENVETO JA POHDINTA

Työn tavoitteena oli ottaa selvää erilaisista autentikointimenetelmistä ja niiden toteutuksista. Lisäksi tavoitteena oli toteuttaa autentikointiportaali NettiTieto Oy:n ja sen asiakkaiden käyttöön, jonka kautta nämä voisivat välittää kaikki vahvan sähköisen tunnistautumisen tapahtumat. Tähän lopputulokseen päästiin, tosin kaikkia haluttuja ominaisuuksia ei saatu toteutettua erinäisistä syistä johtuen.

Työn aikataulu oli melko tiukka johtuen uusista vaatimuksista työlle: toive autentikointiportaalin toteuttamisesta lisättiin työhön vasta melko myöhäisessä vaiheessa. Suurin osa työhön käytetystä ajasta kului erilaisten autentikointimenetelmien tutkimiseen ja niiden ymmärtämiseen, itse toteutus oli melko nopea tehtävä. Lisäksi paljon aikaa kului kryptologisten toimenpiteiden suorittamiseen, esimerkiksi JWS:n oikeellisuuden varmistaminen oli haastavaa PHP:llä.

Jatkokehityksen aiheita työlle jäi vielä: autentikointiportaaliin voisi liittää lisää ominaisuuksia, kuten allekirjoitustoiminnallisuuden. Lisäksi Signicat:n rajapinnan toteutusta autentikointiportaaliin ei ehditty testaamaan käytännössä.

9 LAINATUT LÄHTEET

Auth0. Introduction to JSON Web Tokens. *JWT.* [Online] [Viitattu: 16. Toukokuu 2018.] https://jwt.io/introduction/.

Ficom Ry. 2014. FICOM'S (THE FINNISH FEDERATION FOR TELECOMMUNICATIONS AND TELEINFORMATICS) APPLICATION GUIDELINE FOR ETSI'S MSS STANDARDS. *Mobiilivarmenne.fi.* [Online] 24. Maaliskuu 2014. [Viitattu: 24. Toukokuu 2018.] http://mobiilivarmenne.fi/wp-content/uploads/2017/05/MSS_FiCom_Implementation_guideline.pdf.

Finanssiala Ry. 2018. Kysymyksiä ja vastauksia toisesta maksupalveludirektiivistä (PSD2). [Online] 11. Tammikuu 2018. [Viitattu: 13. Toukokuu 2018.] http://www.finanssiala.fi/uutismajakka/Sivut/QA-Toinen-maksupalveludirektiivi.aspx.

—. Luottamusverkosto, TUPAS ja tunnistamisen muutokset. *Finanssiala.* [Online] [Viitattu: 23. Huhtikuu 2018.] http://www.finanssiala.fi/uutismajakka/Sivut/Luottamusverkosto,-TUPAS-ja-tunnistamisen-muutokset.aspx.

Finlex. 2009. Laki vahvasta sähköisestä tunnistamisesta ja sähköisistä luottamuspalveluista. *Finlex.* [Online] 7. Elokuu 2009. [Viitattu: 13. Toukokuu 2018.] https://www.finlex.fi/fi/laki/ajantasa/2009/20090617.

Internet Engineering Task Force. 2012. RFC 6749: The OAuth 2.0 Authorization Framework. [Online] Lokakuu 2012. [Viitattu: 13. Toukokuu 2018.] https://tools.ietf.org/html/rfc6749.

—. 2015. RFC 7515: JSON Web Signature (JWS). [Online] Toukokuu 2015. [Viitattu: 23. Toukokuu 2018.] https://tools.ietf.org/html/rfc7515.

—. 2015. RFC 7516: JSON Web Encryption (JWE). [Online] Toukokuu 2015. [Viitattu: 23. Toukokuu 2018.] https://tools.ietf.org/html/rfc7516.

—. 2015. RFC 7517: JSON Web Key (JWK). [Online] Toukokuu 2015. [Viitattu: 13. Toukokuu 2018.] https://tools.ietf.org/html/rfc7517.

—. 2015. RFC 7518: JSON Web Algorithms (JWA). [Online] Toukokuu 2015. [Viitattu: 23. Toukokuu 2018.] https://tools.ietf.org/html/rfc7518.

—. 2015. RFC 7519: JSON Web Token (JWT). [Online] Toukokuu 2015. [Viitattu: 23. Toukokuu 2018.] https://tools.ietf.org/html/rfc7519.

Lehto, Tero. 2014. Lakimuutos pakottaa operaattorit ja pankit yhteistyöhön nettitunnistautumisessa. *Tekniikka & Talous.* [Online] Tekniikka & Talous, 28. 10 2014. https://www.tekniikkatalous.fi/tekniikka/ict/2014-10-28/Lakimuutos-pakottaa-operaattorit-ja-pankit-yhteistyöhön-nettitunnistautumisessa-3256698.html.

Mishra, Prateek. 2003. OASIS. *Differences between OASIS Security Assertion MArkup Language (SAML) V1.1 and V1.0.* [Online] 21. Toukokuu 2003. [Viitattu: 24. Toukokuu 2018.] https://www.oasis-open.org/committees/download.php/3412/sstc-saml-diff-1.1-draft-01.pdf.

OASIS. 2008. Differences between SAML 2.0 and 1.1. [Online] 23. Tammikuu 2008. [Viitattu: 24. Toukokuu 2018.] http://saml.xml.org/differences-between-saml-2-0-and-1-1.

—. OASIS Security Services (SAML) TC. [Online] [Viitattu: 13. Toukokuu 2018.] https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security.

—. 2005. Security Assertion Markup Language (SAML) 2.0 Technical Overview. [Online] 20. Helmikuu 2005. [Viitattu: 13. Toukokuu 2018.] https://www.oasis-open.org/committees/download.php/11511/sstc-saml-tech-overview-2.0-draft-03.pdf.

OAuth Core Workgroup. 2009. OAuth Core 1.0 Revision A. *OAuth Core 1.0 Revision A.* [Online] OAuth Core Workgroup, 24. Kesäkuu 2009. [Viitattu: 13. Toukokuu 2018.] oauth.net/core/1.0a/.

OAuth.net. Introduction. [Online] [Viitattu: 16. Toukokuu 2018.] https://oauth.net/about/introduction/.

OpenID Foundation. 2014. OpenID Connect Core 1.0 incorporating errata set 1. [Online] 8. Marraskuu 2014. [Viitattu: 13. Toukokuu 2018.] http://openid.net/specs/openid-connect-core-1_0.html.

Siriwardena, Prabath. 2016. JWT, JWS and JWE for Not So Dummies! . [Online] 26. Huhtikuu 2016. [Viitattu: 23. Toukokuu 2018.] https://medium.facilelogin.com/jwt-jws-and-jwe-for-not-so-dummies-b63310d201a3.

W3C. 2007. SOAP Version 1.2 Part 1: Messaging Framework (Second Edition). [Online] 27. Huhtikuu 2007. [Viitattu: 13. Toukokuu 2018.] https://www.w3.org/TR/soap12/.

Viestintävirasto. 2018. Rekisteri tunnistamispalvelun tarjoajista. *Viestintävirasto.* [Online] Viestintävirasto, 16. Maaliskuu 2018. [Viitattu: 13. Toukokuu 2018.] https://www.viestintavirasto.fi/kyberturvallisuus/sahkoinentunnistaminenjaallekirjoitus/rekisteritunnistamispalveluntarjoajista.html.

—. 2018. TUPAS-tunnistamista käyttäviltä asiointipalveluilta edellytetään muutoksia. *Viestintävirasto.* [Online] Viestintävirasto, 10. Huhtikuu 2018. [Viitattu: 22. Huhtikuu 2018.] https://www.viestintavirasto.fi/viestintavirasto/ajankohtaista/2018/tupas-tunnistamistakayttaviltaasiointipalveluiltaedellytetaanmuutoksia.html.

—. 2017. Vahva sähköinen tunnistaminen, sähköinen allekirjoitus ja varmenne. *Viestintävirasto.* [Online] Viestintävirasto, 20. Marraskuu 2017. [Viitattu: 8. Huhtikuu 2018.] https://www.viestintavirasto.fi/kyberturvallisuus/sahkoinentunnistaminenjaallekirjoitus.html.

—. 2018. Verkkopalveluissa varauduttava pankkitunnistuksen muuttumiseen. [Online] 21. Toukokuu 2018. [Viitattu: 23. Toukokuu 2018.] https://www.viestintavirasto.fi/viestintavirasto/ajankohtaista/2018/verkkopalveluissavarauduttavapankkitunnistuksenmuuttumiseen.html.

Väestörekisterikeskus. Kansalaisvarmenteen käyttöönotto. *Väestörekisterikeskus.* [Online] [Viitattu: 16. Toukokuu 2018.] http://vrk.fi/kansalaisvarmenne.