Johannes Stenberg

# LOHKOKETJUT JA HAJAUTETTU TIETOVARASTOINTI AVOIMEN DATAN TUKENA

JYVÄSKYLÄN YLIOPISTO
INFORMAATIOTEKNOLOGIAN TIEDEKUNTA
2020

# TIIVISTELMÄ

Stenberg, Johannes
Lohkoketjut ja hajautetut tietovarastot avoimen datan tukena
Jyväskylä: Jyväskylän yliopisto, 2019, 60 s.
Tietojärjestelmätiede, pro gradu -tutkielma
Ohjaaja(t): Hara, Veikko; Kyppö, Jorma

Avoimen datan periaatteena on se, että julkiseen käyttöön julkaistu data ja tieto sen alkuperästä ovat aina saatavilla. Nämä periaatteet eivät kuitenkaan välttämättä aina toteudu keskitettyyn tietovarastointiin tai keskeisen tahon hallintaan perustuvissa ratkaisuissa. Vaihtoehtona on tietovarastointi, jossa tallennettu data sijaitsee hajautetusti vertaisverkon noodeilla ja jossa lohkoketjua käytetään datan alkuperätietojen tallentamiseen. Tutkielma toteutettiin suunnittelutieteellisenä tutkimuksena, ja sen tuloksena luotiin suunnitteluartefakti. Artefakti koostuu ohjelmointiprototyypistä sekä prototyypin suunnitelmasta, johon sisältyy myös avoimen datan vertaisverkkopohjaisen tallentamisen vaatimusmäärittely.

Artefaktin suunnittelussa ja toteutuksessa hyödynnettiin Ethereum-lohkoketjua, älysopimuksia, sekä IPFS-tiedostojakelu- ja -varastointiprotokollaa. Keskeisimpinä tuloksina havaittiin, että hajautettu tallennus lisää datan saatavuutta ja että lohkoketjussa saadaan säilöttyä varmennettuna alkuperätietoja. IPFS-pohjainen tallentaminen vaatii erillisen tietokantatason, jotta datan hakeminen olisi mahdollista. Havaittiin myös, että hajautetussa varastoinnissa täytyy valita, painotetaanko datan saatavuutta vai eheyttä.

Asiasanat: lohkoketju, vertaisverkot, avoin data, tietovarasto, älysopimukset

# ABSTRACT

Stenberg, Johannes
Blockchain-based distributed storage in supporting Open Data
Jyväskylä: University of Jyväskylä, 2019, 60 pp.
Information Systems, Master's Thesis
Supervisor(s): Hara, Veikko; Kyppö, Jorma

One of the principles of open data is that public data and the information about its origin should be publicly available. However, this principle doesn't always apply when open data is stored in a centralized fashion or when controlled by a single entity. An alternative model is to store data in a de-centralized, peer-to-peer storage, where the data would be stored by multiple peers, and the provenance information of the data would be stored in a blockchain. This research was conducted according to Design science research (DSR) principles, and its main result is a design artefact. The artefact consists of a software prototype, and the design of the prototype, which also includes the requirements for the decentralized storing of open data.

The technologies used in the implementation of the artefact were Ethereum blockchain and smart contracts, and the IPFS protocol. The most relevant research finding was that the blockchain can be utilized in storing provenance information. IPFS-based storage needed a database layer to enable querying of data. Other finding was that in a distributed storage scheme, one must choose between data availability and consistency.

Keywords: blockchain, peer-to-peer, open data, data storage, smart contracts

## KUVIOT



## TAULUKOT

# SISÄLLYS

# KÄSITEHAKEMISTO

| | |
|---|---|
| ACID | Tietokantaparadigma |
| BASE | Tietokantaparadigma |
| CAP | *Consistency, availability, partition tolerance* |
| CRDT | *Conflict-free replicated data type* |
| DAG | *Directed acyclic graph*, tietorakenne |
| DHT | *Distributed hash table,* hajautustaulu |
| DSR | Suunnittelutieteellinen tutkimus |
| DApps | Hajautetut sovellukset |
| IPFS | *InterPlanetary File System* |
| LOD | Linkitetty avoin data |
| OGD | Avoin julkishallinnon data |
| OPM | *Open Provenance Model* |
| P2P | *Peer-to-peer* |
| PoS | *Proof-of-stake*, konsensusprotokolla |
| PoW | *Proof-of-work*, konsensusprotokolla |
| RDF | *Resource Description Format* |

# 1 JOHDANTO

Avoimen datan periaatteena on se, että yleisesti hyödyllinen data olisi julkisesti kaikkien saatavilla ja käytettävissä ilman veloituksia. Datan avaaminen ja julkiseen jakeluun saattaminen ei kuitenkaan ole yksiselitteinen prosessi: avoimella datalla ei useinkaan ole yhtenäistä julkaisuformaattia tai julkaisupaikkaa, datan tuottajilla ei ole kannustimia avata dataansa, ja datan varastoiminen tarvitsee ottaa huomioon. Perinteiset tietokantaratkaisut eivät ole tähän riittävä ratkaisu.

Datan tallennuksessa pilvitallennus onkin yleistynyt, ja markkinoilla on suuria palveluntarjoajia kuten Amazon, Google ja Dropbox niin yksityisille käyttäjille kuin yrityksille. Suurten palveluntarjoajien tarjoamat tietovarastot ovat monien yritysten käytössä, ja tarve varastoinnille on alati kasvava. Tämä on johtanut kilpailuun pilvitallennuksen palvelutarjonnassa, ja kilpailu siihen, että suurten datamäärien tallennus on keskittynyt muutaman isomman palveluntarjoajan vastuulle.

Dataa tallennetaan suuriin datasiiloihin keskitetysti. Tässä on puolensa esimerkiksi tietoturvan kannalta, sillä isot yritykset ovat investoineet miljoonia datan turvaamiseksi, ja käyttäjät voivat ollakin melko varmoja siitä, että data säilyy. Mutta datan keskitetty hallinta tuo myös siihen liittyvät riskit. Esimerkiksi vuonna 2018 Yhdysvaltain hallitus sulki avoimen datan sivustonsa, mikä johti ongelmiin sitä hyödyntäville organisaatioille. Entä, jos avoin data olisikin hajautetusti saatavilla vertaisverkkoon pohjautuvassa mallissa?

Vertaisverkkopohjaisessa tietovarastoinnissa data saadaan tallennettua ja hallinnoitua hajautetusti vertaisverkossa siten, että vertaisverkon noodit toimivat datan tallentajina. Tämä herättää kysymyksen siitä, voiko hajautetuista lähteistä tulevan datan paikkansapitävyyteen luottaa. Datan alkuperän ja muutoshistorian täytyy olla käyttäjille selviä, jotta datalla olisi luotettavuutta.

Mahdollista ratkaisua tarjoaa Bitcoinin myötä tunnetuksi tullut vertaisverkkopohjainen lohkoketjuteknologia. Lohkoketjun avulla voidaan mahdollistaa luotettavuus hajautetuissa ympäristöissä ja poistaa tarve hallinnoivalle osapuolelle niin, että toisilleen tuntemattomat noodit voivat luotettavasti ylläpitää alkuperädataa verkossa.

Avoimen datan säilöminen lohkoketjussa ei kuitenkaan ole kannattavaa. Koska lohkoketju kopioidaan jokaiselle noodille, monistuisi myös sama data ja tulisi säilöttävän datan koosta liian suuri. Lohkoketju ei siis sovi tietokantojen tai -varastojen korvaajaksi. Yksi mahdollisuus olisi säilöä data hajautetusti jossain muualla – lohkoketjuun tallennettaisiin vain viite dataan ja datan alkuperätiedot, sekä muuta metadataa (Swan, 2015).

Viime vuosina onkin kehitetty useita hajautettuja tietovarastoja ja tietovarastointiprotokollia, joiden toiminta perustuu vertaisverkkoon alustana. Tämän tutkielman tarkoituksena on selvittää, kuinka hajautettu tietovarastointia ja lohkoketjua voidaan hyödyntää yhdessä avoimen datan ja datan alkuperätietojen tallentamiseksi ja hakemiseksi.

## 1.1 Tutkimuskysymykset ja aiheen motivointi

Avoimen datan tärkeys yhteiskunnassamme korostuu erityisesti kaiken sen informaation myötä, joita erinäiset avoimen datan hyödyntäjät tuottavat loppukäyttäjille. Itse dataa arvokkaampaa on siitä jalostettu informaatio. Esimerkiksi säätiedot ovat ilmaista Ilmatieteen laitoksen julkaisemaa dataa, jota lähes jokainen kansalainen käyttää päivittäin. Organisaatiotasolla taas avoin data tuottaa informaatiota, jolla suuria strategisia ja taktisia päätöksiä voidaan tehdä. Esimerkiksi asuinaluesuunnittelussa Tilastokeskuksen tuottama asukasdata voi olla ratkaisevaa.

Datan kasvavan arvokkuuden vuoksi datan tuottajille on kannattavampaa veloittaa käyttäjiä datan käytöstä kuin pistää data avoimesti saataville. Datan ylläpito voi tulla myös kalliiksi. Nämä seikat johtavat usein siihen, että käyttäjät joutuvat maksamaan pääsystä dataan, tai että ylläpidon kustannusten vuoksi julkaistu data otetaan pois verkosta tai ettei sitä enää ylläpidetä. Datan avoimuuden pysyvyys ei siis useinkaan ole itsestään selvää, eikä datan saatavuus.

Olennainen kysymys avoimessa datassa on, miten data pitäisi säilöä ja saada käyttäjien ulottuville. Avoimen datan tuottajat ovat usein laaja, heterogeeninen ryhmä, koostuen muun muassa sekä kunnallissektorin että yksityisistä toimijoista, joilla ei ole välttämättä yhteistä standardia avoimen datan tuottamiseksi. Yleisin malli on, että jokainen avoimen datan tuottaja pitää yllä omia datalähteitänsä, jotka usein ovat vain kertaluontoisesti tehtyjä tiedostoja, kuten Excel-taulukoita. Tämä vaarantaa datan saatavuuden ja avoimuuden.

Avoimen datan kannalta myös tärkeä ominaisuus on sen luotettavuus ja tähän liittyen sen alkuperätietojen saatavuus. Kun avoimen datan tuottajia on useita ja datasta jalostettua informaatiota on saatavilla monesta eri lähteestä, on vaikea varmistaa datan validiteetti. Ratkaisuna olisi tapa, jolla datan voisi aikaleimata (engl. *timestamp*). Lohkoketjuteknologia mahdollistaisi datan aikaleimaamisen pitämällä kirjaa varsinaiseen dataan liittyvästä metadatasta, jonka kautta datan alkuperä ja muokkausajankohdat voisi luotettavasti varmentaa.

Avoimen datan jakelussa käytetään globaalisti useita julkaisupaikkoja, kuten Wikipedia ja Euroopan avoimen datan portaali (https://data.europa.eu),

sekä Suomen julkisella sektorilla Valtionvarainministeriön ylläpitämä Avoindata.fi-sivusto (https://www.avoindata.fi/fi), jonne avoimen datan tuottajat voivat julkaista datasettejä. Nämä mallit nojaavat keskitettyyn palvelimeen datan hallinnassa ja ovat täten alttiita kaikille keskitetyille palvelimille ominaisille ongelmille, kuten palvelunestohyökkäyksille.

Yksi mahdollinen tapa jakaa avointa dataa olisi vertaisverkkopohjainen malli, jossa datan tuottajat ja ylläpitäjät toimisivat noodeina (Hausenblas & Karnstedt, 2010). Dataa julkaistaisiin linkitetyssä muodossa, mikä mahdollistaisi datan hakemisen. Mallin etuja olisivat esimerkiksi datan jakamisen ja yhteistyön lisääntyminen datan jakajien kesken, pienemmät hallinnointikustannukset, data-aineistojen vertailu ja mahdollisesti suurempi saatavuus (Cowan, Alencar & McGarry, 2014).

Avoimelle datalle siis on olennaista muun muassa sen saatavuus ja eheys (Sicilia *et al*., 2016). Keskitettyyn palvelimeen nojaavilla tietovarastomalleilla näiden ehtojen toteutuminen ei ole aina varmaa. Siksi hajautetut tietovarastot voivat tarjota ratkaisua avoimen datan säilömiseen, ja lohkoketjuteknologia ratkaisua datan ja sen jakelun luotettavuuteen. Mallin toimivuudesta on kuitenkin hyvin vähän empiiristä aineistoa, joten tämän tutkielman tarkoituksena on analysoida aihetta suunnittelutieteellisen tutkimuksen (DSR, engl. sanoista *Design science research*) menetelmin. Tarkoituksena on soveltaen selvittää, miten lohkoketjupohjaiset hajautetut tietovarastot tukevat avoimen datan julkaisua ja ylläpitoa.

Tämän aihemotivaation pohjalta tutkimuskysymykset ovat seuraavat:

1. Miten avoin data saadaan tallennettua hajautetusti?
2. Kuinka tukea datan eheyttä (*consistency*)?
3. Kuinka tukea datan saatavuutta (*availability*)?

## 1.2 Tutkielman rakenne

Tutkielman rakenne on seuraavanlainen. Ensin lähdetään rakentamaan teoreettista viitekehystä niin vertaisverkkojen, lohkoketjuteknologioiden kuin avoimen datan periaatteiden ja hajautetun tietovarastoinnin pohjalta. Tutkielman toisessa luvussa keskitytään vertaisverkkoihin erityisesti peilaten niiden käyttöä hajautetussa tietovarastoinnissa. Kolmannessa luvussa analysoidaan lohkoketjuteknologioita ja erityisesti konsensusprotokollia. Neljännessä luvussa kuvataan avoimen datan ja pilvitallennuksen kenttää. Viidennessä eli viimeisessä teorialuvussa taas esitellään keskeisimmät lohkoketju- ja hajautettu tietovarastointisovellukset ja -protokollat, kuten Ethereum, Swarm ja IPFS.

Kun teoreettinen viitekehys on saatu muodostettua, kuvataan kuudennessa luvussa suunnittelutieteellisen tutkimusmenetelmän lähtökohdat ja toteutus. Kahdeksannessa luvussa tarkastellaan tutkimustuloksia teoreettiseen viitekehykseen nojaten ja esitetään vastaukset tutkimuskysymyksiin. Erityisesti käyte-

tään CAP-viitekehystä tarkastelun työkaluna. Viimeisessä luvussa esitetään yhteenveto ja johtopäätökset, sekä ehdotukset tulevalle tutkimukselle aiheesta.

# 2 VERTAISVERKOT

Tässä luvussa perehdytään tarkemmin vertaisverkkoihin (P2P, *peer-to-peer*). Vertaisverkot ovat teknologiana olleet jo verrattain pitkään olemassa. Näkyvin puoli vertaisverkkoteknologioista ovat olleet tiedostojen jakamiseen keskittyvät protokollat (mm. BitTorrent), mutta näiden lisäksi vertaisverkoilla on monia muita käyttöskenaarioita.

Tämän tutkielman kannalta olennaisinta vertaisverkkojen toiminnassa on, miten dataa säilötään vertaisverkoissa ja minkälaisia ratkaisuja on datan hakemiseksi. Näihin näkökulmiin nojaten seuraavaksi tarkastellaan vertaisverkkojen perusteita. Perusteiden selittämisen tavoitteena on, että saadaan muodostettua käsitteellinen perusta tutkielmassa myöhemmin esiteltäville asioille. Monet tutkielman myöhemmistä käsitteistä nimittäin perustuvat vahvasti vertaisverkkojen käsitteisiin.

## 2.1 Vertaisverkkojen perusteet

Termiä vertaisverkko (P2P) käytetään yleiskuvauksena sellaisista verkoista, jotka perustuvat hajautukseen. Perinteisen keskitetyn asiakas-palvelinmallin keskuspalvelimen sijaan vertaisverkoissa on vertaisessa asemassa olevia noodeja (engl. *node*, solmu), jotka muodostavat vertaisverkon yhtäläisinä toimijoina. Vertaisverkoille ominaista onkin korkea noodien itseohjautuvuus, orgaaninen kasvu (eli uusia noodeja voi tulla verkkoon ja vanhoja lähteä pois) ja hajautetun luonteen vuoksi korkea vastustuskyky tietoturvahyökkäyksille ja verkon katkostiloille (Rodrigues & Druschel, 2010).

Toisin kuin keskitetyissä malleissa, joissa haettu tieto löytyy aina palvelinpäästä, vertaisverkossa datan sijainti ei ole niin yksiselitteinen. Esimerkiksi tiedostonjakeluun perustuvissa vertaisverkossa ei pystytä suoraan sanomaan, mikä noodi pitää mitäkin dataa. Tarvitaan siis jonkinlainen mekanismi tiedon paikantamiseksi ja hakemiseksi. Hakemismekanismit voi jakaa kolmeen kate-

goriaan: keskitettyyn palvelimeen perustuviin, kuuluttamiseen (engl. *flooding*) ja hajautustauluihin perustuviin. Näihin perehdytään seuraavissa alaluvuissa.

Huomattavaa on myös, että vertaisverkossa noodi voi yleisesti katsoen toimia joko palvelimen tai asiakkaan roolissa. Tutkimuksissa onkin havaittu, että vertaisverkon käyttäjillä on selvästi taipumusta painottua jompaankumpaan rooliin (Saroiu, Gummadi & Gribble, 2002). Tämä voi kannustaa verkon väärinkäytöksille, ellei käyttäjille tarjota jotain kannustinta toimia halutunlaisesti.

Vertaisverkot voi jakaa kahteen kategoriaan niiden käyttämän tiedonhakumenetelmän mukaisesti: ei-strukturoituihin ja strukturoituihin. Seuraavaksi käsitellään näitä kahta kategoriaa tarkemmin.

## 2.2 Ei-strukturoidut vertaisverkot

Vertaisverkkojen ensimmäistä sukupolvea edustavat niin kutsutut ei-strukturoidut vertaisverkot. Näistä maineikkaimpina esimerkkeinä mainittakoon musiikinjakeluun 2000-luvun alkupuolella keskittynyt Napster, sekä Gnutella. Datan hakeminen näissä tapahtui kahdella pääsääntöisellä tavalla: joko keskitetyn palvelimen kautta tai kuuluttamalla.

Kuuluttamalla haku siirtyy verkossa noodilta toiselle ”sokeasti”, kunnes oikea data löytyy tai sitten haulle asetettu raja tulee vastaan. Tämä onkin hyvin kuormittava ja verrattain tehoton tapa. Ei-strukturoituihin verkkoihin on kuitenkin helppo muodostaa uusia noodeja, mutta yksinkertaisuudessaan se ei sovellu hyvin hajautettuun tietovarastointiin luottamuksellisesti. Skaalautuvuus on heikkoa; kun verkossa on useita tuhansia noodeja, niin tiedonhaku hidastuu huomattavasti ja kuormittaa verkkoa kyselyllään. Joissain protokollissa kuuluttaminen onkin rajoitettu tiettyyn määrään noodeja, jottei kyselystä tulisi liian kuormittavaa – tässä mallissa vain ongelmana on se, että ei-suosittu data saattavat jäädä löytymättä, jos mikään kuulutuksen saaneista noodeista ei sisällä haettua dataa.

Ei-strukturoitujen vertaisverkkojen toinen tapa paikantaa tietoa vertaisverkossa on keskitetyn palvelimen käyttö. Tämä palvelin pitää listaa eri tiedostojen sijainneista. Sitä käytti muun muassa Napster ja monet muut ensimmäisen sukupolven P2P-sovellutukset. Keskitetyn palvelimen käyttö kuitenkin tuo samat ongelmat, mitä muillakin keskitetyillä ratkaisuilla on, ja aiheuttaa pullonkaulan vertaisverkkoon, kun haettavan datan täytyy kulkea aina saman keskitetyn palvelimen kautta.

## 2.3 Strukturoidut vertaisverkot

Tämän tutkielman kannalta strukturoidut vertaisverkot ovat suuremmassa tarkastelussa, sillä ne ovat yleisesti katsoen tehokkaampia hajautetun tiedon käsit-

telyssä, ja teknologiat kuten IPFS nojaavat nimenomaan strukturoituihin vertaisverkkoteknologioihin.

Strukturoidut vertaisverkot perustuvat pitkälti jaetun hajautustaulun eli DHT:n (engl. sanoista *distributed hashtable*) käyttöön datan paikantamisessa. Kyseessä on rakenne, jolla pystytään tehostamaan hajautetun datan hakemista ja tallentamista vertaisverkossa. Tunnettuja DHT-protokollia ovat muun muassa Chord, Pastry ja Kademlia.

DHT perustuu siihen, että verkon noodeihin tallennetaan avain-arvopari jonkin strategian mukaisesti. Viittaus toiseen noodiin määritellään hajautusarvon perusteella. Näin saadaan muodostettua hajautettu viittaustaulu jokaiseen verkon noodiin, mikä nopeuttaa tietojen hakemista verkossa. Kun lähdetään noodista A hakemaan dataa, löydetään noodi B, joka sisältää kyseisen datan nopeasti hajautuksen avulla.

Kun verkkoon liittyy uusi noodi tai jos noodi eroaa verkosta, pitää viittaukset määritellä uudelleen. Usein tämä on toteutettu siten, että noodin viittausnoodien ohitettua tietyn alimmaismäärän määritetään uudet viittausnoodit. Verkon noodit ohjautuvat näissä tapauksissa itseohjautuvasti ja muodostavat uudet avain-arvosuhteet toisiin noodeihin. Näin DHT pystyy järjestäytymään ja pitämään datan integriteetin koossa siitä huolimatta, vaikka noodeja lähtisikin pois verkosta.

Datan säilömisen voi järjestää DHT:n avulla kahdella tapaa. Ensimmäinen vaihtoehto on, että noodit itsessään sisältävät datan ja toinen se, että noodi sisältää vain viittauksen dataan, mutta varsinainen data sijaitsee toisessa sijainnissa. Ensimmäisen vaihtoehdon hyöty on siinä, että data sijaitsee suoraan vertaisverkossa ja on saatavilla silloinkin, jos datan lisännyt noodi poistuu verkosta. Huono puoli taas on, että tämä vaatii vertaisverkolta paljon tallennus- ja tiedonsiirtokapasiteettia.

Datan sijaitessa taas vertaisverkon ulkopuolella lisäävä noodi tallentaa vain viitteen datan DHT:hen. Tällä menetelmällä säästytään edellä mainitun menetelmän ongelmilta, mutta datan saatavuuden kannalta ollaan riippuvaisia datan lisänneestä noodista – data on olemassa vain niin kauan, kuin kyseinen noodi on aktiivisena (Wehrle & Stefan, 2005).

## 2.4 Vertaisverkkojen haasteita ja ongelmia

Vertaisverkot nojaavat olennaisesti siihen, että sen käyttäjät toimivat yhtäläisesti ja yhteisten tavoitteiden mukaisesti. Usein vertaisverkkoteknologioiden ongelmana onkin, miten voidaan varmistaa tasapuolinen käyttö. Usein törmätään niin sanottuun vapaamatkustamisen ongelmaan (engl. *free riding* tai *leeching*), eli siihen, että joku hyödyntää verkkoa tarjoamalla itse siihen mitään (Saroiu *et al.*, 2003). Hajautettujen tietovarastojen tapauksessa riski on, että varastointitilan tarjoaja ei säilytäkään tiedostojaan – tätä varten pitää olla jokin varmennekeino, jolla voidaan tarkistaa tarvittaessa, että kyseiset tiedostot ovat edelleen säilöttyinä.

Tämän lisäksi tulee olla jokin kannustinmalli, jolla käyttäjät saadaan toimimaan halutulla tavalla. Ilman selvää kannustinta (tai jonkinlaista valvontamekanismia) vertaisverkkojen käyttäjät ovat taipuvaisempia pimittämään tietojaan tai olla jakamatta vertaisverkossa (Saroiu *et al.,* 2002). Esimerkiksi tiedostojen jakamisessa jakajat voivat tahallisesti ilmoittaa kaistanleveytensä pienemmäksi kuin onkaan vähentääkseen kohdistuvaa latauskuormaa. Pitää olla siis jokin kannustinmalli, jolla jakajat hyötyvät siitä, että luovuttavat kaistanleveyttään vertaisverkon käyttöön.

Vertaisverkot eivät myöskään ole täysin haavoittumattomia tietoturvahyökkäyksille. Yksi merkittävimmistä hyökkäysmuodoista on niin sanottu Sybil-hyökkäys (engl. *Sybil attack*), jossa yksi entiteetti muodostaa vertaisverkkoon lukuisia eri solmuja, joilla voi saada merkittävän osan vertaisverkosta hallintaansa ja toteuttaa palvelunestohyökkäyksiä. Tämä voi ilmetä esimerkiksi hajautustauluihin perustuvissa vertaisverkoissa siten, että hyökkääjä saa haltuunsa kaikki tietyn avain-arvoparin viitteet ja näin pimittää kyseisen tiedon koko vertaisverkolta (Rodrigues & Druschel, 2010).

Tietyn datan löytyvyys ei myöskään ole itsestään selvää vertaisverkoissa, vaan se on hyvin paljon kiinni datan sijoitusstrategiasta ja datan jakamisen kannustimista. Suosittu data löytyy helposti, kun taas harvemmin haettu data uhkaa jäädä pimentoon. Avoimen datan tasapuolisen jakelun ja saatavuuden kannalta olisi tärkeää, ettei datan saatavuus olisi sidottu sen suosioon.

Alla olevassa taulukossa on koostettuna vertaisverkkojen keskeisimmät haasteet (Taulukko 1).

Taulukko 1 Vertaisverkkojen haasteita

| **Haaste** |
|---|
| Vapaamatkustaminen |
| Ei-haluttu toiminta, tietojen pimitys |
| Tietoturvariskit (mm. Sybil-hyökkäys) |
| Ei-suositun datan löytämisen vaikeus |

Vertaisverkkojen haasteet johtuvat siis suurimmilta osin sen noodien toiminnasta ja itseohjautuvuudesta: vertaisverkon käyttäjillä on taipumusta toimia oman edun mukaisesti. Vertaisverkkojen toimivuus perustuu vahvasti sen noodien yhteistyöhön, joten jollakin tavalla yhteistyöhön pitäisi kannustaa.

## 2.5 Kannustimet

Jotta vertaisverkko voisi toimia luotettavasti ja mahdollisimman tehokkaasti, tulee vertaisverkon noodit sitouttaa toimimaan yhteistyökykyisesti (Druschel, 1999). Tätä varten vertaisverkoissa hyödynnetään erilaisia kannustinmenetelmiä. Yleisimmin nämä jakaantuvat kahteen luokkaan : rahallisiin kannustimiin tai parempaan palveluun. Parempaa palvelua tarjotaan suhteessa enemmän sille noodille, joka tuottaa vertaisverkolle eniten hyötyä, ja tällainen parempi palvelu voi esimerkiksi olla nopeammat latausajat.

Vertaisverkon noodit voidaan mieltää strategisina pelaajina, joilla on pyrkimyksenä maksimoida oma hyötynsä (Buragohain, Agrawal & Suri, 2003). Esimerkiksi Bitcoinin tapauksessa jokaiselle voittoa tavoitteleva louhijanoodi on suotuisaa pyrkiä maksimoimaan oman laskentakapasiteettinsa, jotta lohkon ratkaiseminen ja siten myös palkkiona saatavien bitoinien ansaitseminen olisi todennäköisempää. Hajautetun tietovarastoinnin tapauksessa taas esimerkiksi tekeillä olevan FileCoin-maksujärjestelmän kannustimina toimivat tiedon säilöjille maksetut palkkiot datan säilyttämisestä ja jakamisesta.

Kannustimien riskinä on kuitenkin, etteivät resurssit allokoidu tasapuolisesti vertaisverkossa. Esimerkiksi jos hyvin dataa säilöviä noodeja palkitaan, niin todennäköisemmin vain näiden kyseisten noodien ylläpitämä data on saatavilla kuin toisten. Datan jakamisessa kannustimien rinnalla onkin hyvä olla jonkinlainen datan replikoimisstrategia, joka toteutettaisiin vertaisverkossa automaattisesti. Näitä käsitellään lisää luvussa 4.2.

# 3 LOHKOKETJU

Tässä luvussa perehdytään tarkemmin lohkoketjuun. Lohkoketju (engl. *blockchain*) on vertaisverkkoteknologia, jonka perimmäisenä tarkoituksena on varmistaa luotettavuus ja tietojen oikeellisuus jossain kontekstissa olematta riippuvainen minkään keskitetyn tahon hallinnasta. Lohkoketju tarjoaa ratkaisua vertaisverkoissa yleiseen Bysantin kenraalien ongelmaan (engl. *Byzantine general problem*), eli kun vertaisverkon tilasta ei pystytä muodostamaan konsensusta esimerkiksi vilpillisten noodien takia.

Lohkoketju on pohjimmiltaan eräänlainen hajautettu tilikirja, jota jaetaan jokaiselle vertaisverkon noodille. Nimensä mukaisesti lohkoketju koostuu kryptografisesti toisiinsa liitetyistä ”lohkoista”, joissa on kuvattu lohkon lisäyksen aikaleima, edellisen lohkon tiivistearvo ja Bitcoinin tapauksessa transaktiodataa. Algoritmillisesti lohkoketjun tietorakenne on siis linkitetty lista.

Transaktioiden linkityksessä hyödynnetään Merkle-puuksi kutsuttua algoritmillista rakennetta. Merkle-puussa jokainen haara muodostuu edeltävien transaktioiden tiivisteestä, jolloin saadaan muodostettua varmennettu linkitys: lohkojen muokkaaminen muokkaisi tiivistearvoa, jolloin se ei enää olisi yhtenevä muun lohkoketjun kanssa. Lohkoketju on siis muuttamaton rakenne.

Lohkoketjut pohjautuvat siis vertaisverkkoteknologioihin ja kryptografisiin menetelmiin, jotka ovat olleet olemassa jo pitemmän aikaa, mutta lohkoketjun varsinainen innovaatio on näiden teknologioiden ja protokollien liittäminen taloudellisiin kannustimiin. Kannustimina toimii vertaisverkossa jaettava rahake (engl. *token*) eli esimerkiksi bitcoin, jolla palkitaan noodeja tiettyjen sääntöjen mukaan. Näitä tarkastellaan seuraavaksi, kun lähdetään kuvailemaan eri konsensusprotokollia.

## 3.1 Konsensusprotokollat

Lohkoketjujen perustana on niin sanottu konsensusmekanismi, jolla varmistetaan, että kaikki tai valtaosa vertaisverkon noodeista ovat yksimielisiä verkon

tilasta. Ongelma on tyypillinen vertaisverkoille, sillä keskitetyn hallinnan puuttuessa ei ole selvää, mikä on verkon todellinen tila, etenkin, jos verkossa on vilpillisiä noodeja, jotka yrittävät väärentää tietoja. Konsensusmekanismin toteuttamiseksi on kehitelty useita erilaisia protokollia, joita lähdetään kuvaamaan seuraavaksi.

### 3.1.1 Proof-of-work

Konsensusprotokollista yleisin ja Bitcoinissakin käytetty on *proof-of-work*-protokolla. Se perustuu siihen, että lohkon varmentamiseksi louhijoiksi kutsutut vertaisverkon noodit yrittävät ratkaista paljon laskentatehoa vaativan laskentaongelman, joka on vaikea ratkaista, mutta jonka ratkaisu on helppo todentaa oikeaksi muiden noodien toimesta, jotta verkossa saadaan muodostettua yhtenäinen konsensus verkon tilasta. Proof-of-work on ollut olemassa jo pitemmän aikaa, ja sitä on käytetty lukuisissa erilaisissa sovellutuksissa, kuten muun muassa roskapostien estossa.

Esimerkkinä Bitcoinin tapauksessa louhijoiden pitää laskea hajautusarvo, jonka alussa on tietty määrä nollabittejä. Kun joku louhijoista saa laskettua oikean arvon, kuulutetaan ratkaisu koko verkkoon, ja uusi lohko liitetään edeltävään lohkoon lohkoketjussa. Kannustimena louhijoille tarjotaan uusia bitcoineja ratkaisun löytämisestä: näin samalla kasvatetaan bitcoinien rahakantaa. Jokaisen ratkaistun lohkon myötä ongelma vaikenee; näin vaaditaan alati lisää laskentatehoa. Lohkoketjun kasvaessa sen murtamattomuus vahvistuu entisestään, sillä muokatakseen lohkoketjun historiaa murtautujalla pitäisi olla enemmän laskentatehoa kuin koko lohkoketjussa Merkle-puun rakenteen vuoksi, eikä tämä ole realistisesti saavutettavissa.

Proof-of-work siis perustuu laskentatehoon – mitä enemmän sitä tuottaa verkolle, niin sitä todennäköisemmin saa lohkon ratkaistua ja lunastettua ratkaisupalkkion. Bitcoinin tapaus on osoittanut sen, että tämä malli johtaa kilpavarusteluun laskentatehon kasvattamisessa ja suurten louhintayhtymien muodostumiseen. Proof-of-work siis käyttää suuren ja alati kasvavan määrän sähköä, eikä täten ole ekologisesti kovinkaan kannattava ratkaisu.

Lisäksi Bitcoin on osoittanut, ettei proof-of-work skaalaudu suurten datamäärien käsittelyyn erityisen hyvin. Nykyisellään Bitcoin-verkko on verrattain hidas transaktioiden käsittelyssä. Lohkon muodostamiseen kuluu aikaa keskimäärin 10 minuuttia (Casino, Dasaklis & Patsakis, 2019). Siksi vaihtoehtoisia ratkaisuja on jo kehitetty.

### 3.1.2 Proof-of-stake

Vaihtoehdoksi proof-of-work-protokollalle on kehitetty *proof-of-stake*-niminen protokolla. Verkkoon tuodun laskentatehon sijaan noodit laittavat panokseksi tietyn määrän verkon omia rahakkeita lohkon ratkaisussa. Algoritmi valitsee sitten satunnaisesti noodien joukosta lohkon ratkaisijan sen mukaan, että todennäköisyys tulla valituksi on suhteessa sijoitetun panoksen määrään.

Proof-of-staken hyötyjä ovat mahdollisesti muun muassa energiatehokkuus, parempi tietoturvan taso ja lyhyempi lohkongenerointiaika (Buterin, 2014). Energiatehokkuutensa ansiosta proof-of-stake onkin todennäköisemmin kannattavampi vaihtoehto, mikäli lohkoketjuteknologiat yleistyvät. Bitcoin on osoittanut, ettei proof-of-work skaalaudu kovin hyvin ja että se ei ole energiaystävällinen vaihtoehto alkuunkaan.

Toisaalta proof-of-stake voi törmätä ongelmiin niissä tapauksissa, joissa toimija yrittää kuluttaa saman rahakkeen useaan kertaan, tai jos vahingossa useampi noodi onnistuu muodostamaan uuden lohkoketjuun lisättävän lohkon samanaikaisesti (Buterin, 2014). Näin verkon tilasta on useampi kilpaileva käsitys, ja oikea lohkohistoria määrittyy sen mukaan, mille historialle muut louhijat laittavat rahakkeitaan. Proof-of-workin tapauksessa on kannattavaa allokoida laskentatehoa vain yhdelle historialle, sillä useamman historian kannattaminen vie enemmän laskentatehoa, mutta proof-of-staken tapauksessa useamman historian kannattaminen on halpaa ja täten kannattavampaa, sillä se lisää mahdollisuutta tulla valituksi lohkon ratkaisijaksi. Tämä voi pahimmassa tapauksessa jopa estää konsensuksen muodostumisen verkossa (Buterin, 2014). Proof-of-staken käyttö on tosin ollut niin vähäistä suhteessa Bitcoinin proof-of-work -protokollaan, ettei kyseisen skenaarion todellistumisesta ole ainakaan vielä todisteita.

Yksi variaatio proof-of-stakesta on niin sanottu delegoitu proof-of-stake -protokolla. Siinä uusien transaktioiden julkaiseminen verkkoon valtuutetaan tietyille luotetuille noodeille.

### 3.1.3 Muut konsensusprotokollat

Edellä mainittujen kahden yleisimmän protokollan lisäksi on kehitetty lukuisia muita konsensusprotokollia. Tähän mennessä niiden käyttö on kuitenkin ollut hyvin pienimuotoista, eikä siten empiirisesti todistettu, kuinka hyvin ne soveltuvat suurten tapahtumamäärien käsittelyyn.

Eräs mainittava konsensusprotokolla on niin sanottu *proof-of-burn* -protokolla. Kyseinen protokolla on muunnelma *proof-of-stake*sta ja perustuu siihen, että panokseksi laitettavat rahakkeet tuhotaan.

## 3.2 Lohkoketju datan alkuperätietojen varmentamisessa

Lohkoketjua voi hyödyntää datan alkuperän varmentamisessa ja tallentamisessa eri tavoin, ja lähdekirjallisuuden pohjalta löydettiin eri menetelmiä (Azaria, Ekblaw, Vieira & Lippman, 2016; Ramachandran & Kantarcioglu, 2017; García-Barriocanal, Sánchez-Alonso & Sicilia, 2017). Seuraavaksi kuvaillaan kehiteltyjä ratkaisuja ja pohditaan niiden soveltuvuutta tämän tutkielman suunnitteluartefaktin toteuttamisessa.

MedRec-projektissa lohkoketjua käytetään potilastietojen muutoshistorian varmentamiseen, kun potilastiedot liikkuvat eri tietojärjestelmät omaavien sairaanhoitopiirien välillä (Azaria *et al.*, 2016). Jokaisesta tapahtumasta tallennetaan kirjaus lohkoketjuun, jonka pohjalta voi rakentaa erehtymättömän potilashistorian. MedRec hyödyntää Ethereum-lohkoketjun (ks. luku 5.1) päälle rakennettua, älysopimuksiin perustuvaa hajautettua sovellusta (dApps) potilastietojen hallinnassa. Lohkojen varmentaminen perustuu omiin sisäisiin rahakkeisiin, joilla palkitaan esimerkiksi tutkijaa tämän julkaiseman metadatan perusteella. Tosin tutkimusartikkelista jää epäselväksi, kuinka tehokas kannustin sisäinen rahake loppujen lopuksi on MedRecin tapauksessa.

ProvChain-arkkitehtuuriratkaisu taas tarjoaa yleistä toteutusta datan alkuperätietojen ja metadatan säilömiselle pilvialustalla. ProvChain järjestää datan säilömiselle viisitasoarkkitehtuurin, joka hyödyntää lohkoketjusovellusten arkkitehtuuriratkaisuksi kehitettyä Blockstack-arkkitehtuuria (Ali, Nelson, Shea & Freedman, 2016). Blockstackin tarkoitus on toimia yleisenä luottamuksen takaavana sovellusalustana ilman kolmatta osapuolta. Blockstack nojaa ensisijaisesti siihen, että sen pohjalla käytetään Bitcoinin lohkoketjua; tällä voi olla eitoivottuja vaikutuksia esim. transaktioiden varmistusaikojen suhteen, sillä Bitcoin-verkko käsittelee transaktioita hitaasti.

Garía-Barriocanal ym. (2017) esittävät mallin avoimen datan metadatan säilömiselle. Malli perustuu Ethereum-lohkoketjun ja älysopimusten, IPFS-protokollan (ks. luku 5.2) ja lohkoketjuun perustuvan BigchainDB-tietokannan käyttöön. Ethereum tarjoaa alustan alkuperädatan varmentamiselle, IPFS toimii tiedostonjakelu- ja -tallennusalustana, sekä BigchainDB strukturoituina tietokantarakenteena, jotta datan indeksointi ja hakeminen olisi mahdollista.

Wienin kaupunki on hyödyntänyt lohkoketjua pilottihankkeessaan avoimen datan alkuperätietojen varmentamiseen. Hankkeessa lisätyn datan alkuperätiedot tallennetaan julkiseen lohkoketjuun – näin jokaisen datasetin alkuperä on varmennettavissa, ja lisäyshistoria on jäljitettävissä. Tätä ajatusta tullaan soveltamaan myös tämän tutkielman suunnitteluartefaktissa.

# 4 AVOIN DATA JA HAJAUTETTU TIETOVARASTOINTI

Tässä luvussa tarkastellaan syvemmin avointa dataa ja hajautettua tietovarastointia. Edellisissä luvuissa on esitelty perusteet siitä, miten vertaisverkkoon pohjautuva hajautus rakentuu.

Ensin lähdetään avaamaan avoimen datan käsitettä ja esitellään lähdekirjallisuuden pohjalta sen keskeisiä haasteita, joihin hajautettu tietovarastointi voisi tarjota ratkaisua. Tämän jälkeen käsitellään hajautettua tietovarastointia kuvaamalla datan hajautettuun tallentamiseen ja hakemiseen liittyviä seikkoja, ja niin kutsuttuja NoSQL-tietokantoja, joihin monet hajautetut tietovarastot perustuvat. Lopuksi esitellään avoimen datan haasteiden ja hajautetun tietovarastoinnin synteesinä vaatimusmäärittely avoimen datan varastoinnille hajautetusti. Tätä vaatimusmäärittelyä käytetään tutkielman artefaktin suunnitelman pohjana.

## 4.1 Avoin data

Avoin data on dataa, joka on julkisesti kaikkien saatavilla. Avoimen datan tunnusmerkkejä ovat, että millä tahansa osapuolella on oikeus käyttää kyseistä dataa jatkojalostaakseen siitä informaatiota erinäisiin tarkoituksiin; alkuperäistä dataa ei kuitenkaan saa väärentää tai esittää omana tuotoksenaan. Julkinen sektori erityisesti tuottaa paljon julkista dataa, kuten sää-, kartta- ja liikennetietoja. Tällainen data tukee niin yksittäisen ihmisen kuin organisaation päätöksentekoa ja on siksi tärkeää. (Cowan, Alencar & McGarry, 2014.)

Datan julkaisemista avoimeen käyttöön kutsutaan datavarantojen avaamiseksi (Poikola, Kola & Hintikka, 2010). Tämä on usein monivaiheinen projekti, jossa täytyy ottaa huomioon, mitä halutaan tasalleen julkaista ja missä formaatissa. Mutta avoimeen dataan liittyy olennaisesti myös avaamisen jälkeinen hallinnointi: esimerkiksi säätietoja tarvitsee päivittää jatkuvasti. Avoimen datan

käyttäjillä voi olla suurta tarvetta siis, että data on mahdollisimman viimeaikaista.

Datan avoimuutta voi mitata Tim Berners-Leen luomalla 5-portaisella asteikolla (ks. http://5stardata.info/en/). Portaikko kuvaa, miten hyvin data on käytettävissä ja koneluettavissa. Ihannetapauksessa julkaistu data on täysin koneluettavissa, toiseen dataan linkitettyä ja käyttäjien saatavilla sellaisessa formaatissa, jonka käsittelyyn ei tarvita maksullista ohjelmistoa (mm. Excel) ja hyödyntäen webresurssien kuvailuun luotua RDF-formaattia (engl. sanoista *Resource Description Framework)*. Tämä kuvataan tämän luvun seuraavassa alaluvussa Alla Kitchinin (2014) esittämä taulukko avoimen datan avoimuuden asteikosta (Taulukko 2).

Taulukko 2 Avoimen datan avoimuuden asteikko

| **Taso** | **Datan muoto** | **Hyödyt** | **Rajoitteet** |
|---|---|---|---|
| 1 | Ei-koneluettavassa muodossa (esim. kuva) | Data on saatavilla | Data on sidottu dokumenttiin ja vaikea siirtää |
| 2 | Koneluettavassa muodossa, mutta maksullisessa formaatissa (esim. Excel) | Dataa voi käsitellä vastaavalla ohjelmistolla | Riippuvainen maksullisesta ohjelmistosta |
| 3 | Koneluettavassa, eimaksullisessa formaatissa | Dataa voi käsitellä millä tahansa sopivalla ohjelmistolla. | Data ei ole linkitettävissä muuhun dataan |
| 4 | Koneluettavaa, eimaksullisessa formaatissa käyttäen RDF:ää ja julkista URI:a | Dataan pääsee käsiksi mistä tahansa Internetistä ja sen voi linkittää helposti toiseen dataan | Dataa työläämpi tuottaa ja ylläpitää |
| 5 | Sama kuin yllä, mutta linkitettynä muuhun dataan | Data on helpommin löydettävissä | Linkkien hallinta vaatii aktiivista ylläpitoa |

Avoin datan tulisi optimitapauksessa olla niin sanottua linkitettyä avointa dataa (LOD, engl. sanoista *Linked open data*). Tämä tarkoittaa sitä, että datan yhteydessä on linkityksiä muuhun dataan, mikä muodostaa semanttisen verkon datan hakemiselle. Nykyisellään Internetissä oleva linkitetty avoin data muodostaa pilvimäisen, hajautetun datarakenteen, mutta kyseiselle rakenteelle ei ole vielä olemassa yhtenäistä standardia (Hausenblas & Karnstedt, 2010).

LOD-pilvi on siis lukuisten eri palvelimien muodostama löyhä verkkorakenne, jossa palvelimet ylläpitävät omia datasettejään. Jonkin tietyn linkitetyn datasetin saatavuus on täysin riippuvainen siitä, että sitä ylläpitävä palvelin on toiminnassa: jos palvelin kaatuu tai joutuu vaikkapa palvelunestohyökkäysten kohteeksi, ei data enää ole saatavilla. Avoimen datan saatavuuden ja jaettavuuden kannalta olisi parempi, jos data sijaitsisi hajautetusti eri vertaisverkon palvelimilla. (Sicilia, Sanchez & García, 2016.)

Linkitetyn avoimen datan varastoiminen vaatii toimiakseen tehokkaasti erityisiä vaatimuksia sitä säilövältä tietokannalta. Avoimen datan pitää ensinnäkin olla saatavilla mahdollisesti koko Internetin laajuisesti – vertaisverkon siis tulee pystyä kattamaan maailmanlaajuisesti siihen suunnatut kyselyt. Tämä voi olla haasteellista, riippuen täysin vertaisverkon tiedonhakustrategiasta. Lisäksi datan linkittämisessä käytetyt sanastot tulee saada yhdistettyä. Ei ole nimittäin varmuutta siitä, että jokainen datan julkaisija käyttää samaa sanastoa kuvaamaan samaa tarkoittavia asioita. Avoimen datan pilven tulee myös voida tukea avainsanapohjaista hakua. Ja tärkeää on myös, että sekä datan alkuperä että mahdolliset lisenssit ovat helposti selvitettävissä. (Hausenblas & Karnstedt, 2010.)

Hausenblas ja Karnstedt (2010) esittävät kolme tapaa järjestää avoimen datan varastoinnin. Ensimmäinen keino olisi säilöä dataa keskitetyssä säilytyspaikassa. Tämän haasteina ovat datan ylläpidon vaikeudet ja heikko skaalautuvuus. Datan ylläpito olisi viimekädessä aina keskitetyn tahon vastuulla. Toinen vaihtoehto on, että dataa säilötään muualla ja että datalähteisiin on tavoitettavissa hauilla. Tämä on yleisin malli datakatalogeissa. Sen haasteena on luotto datalähteiden saatavuuteen – aivan hyvin datan ylläpitäjä saattaa lopettaa datan ylläpidon, jolloin linkitykset eivät enää johda mihinkään. Kolmas vaihtoehto on datan jakaminen vertaisverkossa, mikä on tämän tutkielman fokus. Tämän mallin haasteisiin ja hyötyihin syvennytään tämän tutkielman hajautettua tietovarastointia käsittelevässä luvussa.

### 4.1.1 Datan avaaminen ja tarjoaminen

Open Data Foundationin toteuttamassa laaja-alaisessa dataportaaleihin kohdistuneessa tutkimuksessa (2017) selvitettiin yleisiä käytänteitä avoimen datan avaamiselle. Datan avaaminen avoimeksi dataksi lähtee liikkeelle dataan sisältävien tiedostojen (esim. taulukkotiedostot) saattamisesta palvelinpäähän julkaisukelpoisessa kunnossa. Julkaisupaikkana voi yleisesti ottaen olla joko: datasetin tarjoaminen oman palvelimen ja nettisivun kautta, datakatalogilinkitys ja/tai API-rajapinta.

Näistä ensimmäinen vaihtoehto vaatii, että datan tuottaja toimii samalla datan julkaisijana tarjoamalla dataa oman nettisivun kautta. Tämä takaa sen, että datan tarjontaa voi räätälöidä juuri tuottajan tai datan käyttäjäkunnan omien vaatimusten mukaan. Läheskään kaikilla avoimen datan tuottajilla ei ole tähän mahdollisuutta resurssien puutteen takia (Open Data Foundation, 2017).

Datakatalogit ovat sivustoja, joihin on koottuna linkit useampaan datasettiin. Datakatalogit tarjoavat myös hakumahdollisuuksia datalle. Niiden tärkeänä tehtävänä on siis tarjota käyttäjälle helppoa rajapintaa, josta haluttu data on kätevästi löydettävissä. Katalogit vaativat siis kolmannen osapuolen tarjoamaa palvelinta julkaisualustanaan, mutta useimmiten varsinainen data on tallennettuna muualla. Datakatalogijulkaisu on huomattavasti yleisin vaihtoehto julkaisukanavana (Open Data Foundation, 2017).

Datan saattaminen API-rajapinnan päähän vaatii jo enemmän työtä ja perehtyneisyyttä rajapintojen tekemiseen ja ylläpitoon. Monesti datan tuottajilla ei ole resursseja tällaiseen, joten rajapintajulkaisu on harvinaisempaa.

### 4.1.2 Sanastot

Jotta kaksi toisilleen vierasta datasettiä voitaisiin linkittää toisiinsa ja jotta ylipäänsä avoimen datan hakeminen hajautetussa järjestelmässä olisi mahdollista, tarvitaan yhtenäinen sanasto, eli ontologia, samaa asiaa tarkoittaville käsitteille. Tämä tehostaa datan löydettävyyttä.

Avoimen datan linkittämisessä yleisin käytetty sanasto on Dublin Core. Toinen varteenotettava ontologia on FOAF. Sanastot kertovat, minkälaisia assosiaatioita datan eri entiteeteillä on ja mitä kukin entiteetti edustaa. Sanaston avulla voi kuvata esimerkiksi, että jokin tietty tietue datassa on vaikkapa sähköpostikenttä, jolloin dataa koneellisesti luettaessa osataan automaattisesti poimia sähköpostikenttä.

### 4.1.3 RDF datan linkittämisessä

RDF määrittää datalle tietyn esitysmuodon, mikä tekee eri datasettien toisiinsa linkittämisestä mahdollista. Tämä esitysmuoto mallinnetaan kolmikkona (engl. *triple*), jossa datalle määritellään subjekti, predikaatti ja objekti. Esimerkiksi lausumassa ”Pöytä on pyöreä” pöytä olisi datan subjekti, oleminen olisi predikaatti ja pyöreys olisi objekti. Tällainen esitysmuoto on tärkeää koneluettavuuden kannalta, sillä kone itsessään ei osaa muodostaa semanttisia assosiaatioita eri asioiden välillä.

### 4.1.4 Avoimen datan alkuperätiedot ja OPM

Tieto avoimen datan alkuperästä on useimmiten hyvin tärkeää. Alkuperätiedot kertovat muun muassa, milloin dataa on luotu tai muokattu, kenen toimesta ja mistä sijainnista varsinainen data löytyy. Alkuperätiedot tallennetaan usein metadatana tallennettavan datan ohella. Mitään takeita ei kuitenkaan ole, että alkuperätieto olisi aina ajankohtaista. Yhtä hyvin voidaan muutoksia dataan tehdä ilman, että siitä julkaistaan metadataa.

Yleisenä standardina datan alkuperätiedoille pidetään OPM-mallia (engl. sanoista *Open Provenance Model*). OPM-mallissa määritellään yleinen rakenne alkuperätietojen metadatalle. Entiteettejä tässä mallissa on neljä: datan käsittelijä, viite alkuperäiseen dataan, viite uuteen dataan, sekä muutostapahtuman tyyppi (esim. lisäys, poisto, muokkaus). (Moreau *et al.*, 2008.)

### 4.1.5 Avoimen datan lisenssit

Avoimelta datalta vaaditaan usein selvä tieto sen julkaisulisenssistä, jotta sitä voidaan hyödyntää. Epätietoisuus lisensseistä karkottaa käyttäjiä, sillä kukaan ei halua syyllistyä tekijänoikeusrikkomuksiin. Avoimen datan kannalta yleisesti käytettyjä lisenssejä ovat eri *Creative Common* -lisenssit. (Welle Donker & van Loenen, 2017.)

### 4.1.6 Avoimen datan keskeisiä haasteita

Avoimen datan ylläpidossa, käytettävyydessä, saatavuudessa ja alkuperän varmentamisessa on useita haasteita. Näistä saatavuuden ja alkuperän varmentamiseen liittyviin ongelmiin hajautettu tietovarastoinfrastruktuuri ja lohkoketjuteknologia voisivat tarjota ratkaisua.

Keskeinen haaste avoimessa datassa on, mistä saadaan rahavirta avoimen datan tuottamiseen ja ylläpitoon. Julkisen sektorin tapauksessa datan julkaisu ja ylläpito ovat usein verovaroin rahoitettavaa, mutta aina tilanne ei ole näin yksiselitteinen. Julkinen sektori voi siirtää vastuun datasta yksityiselle sektorille, jolloin menojen kattamiseksi yksityisen sektorin palvelu voi veloittaa datan käytöstä, mikä rikkoo avoimen datan periaatetta. (Kitchin, 2014.)

Ongelmallista avoimessa datassa on myös sen lähteiden heterogeenisyys ja pirstaloituneisuus (Janssen *et al*., 2012). Avoimen datan tuottajilla ei ole yhtenäistä formaattia tuottamalleen datalle, mikä vaikeuttaa esimerkiksi datan haettavuutta ja käyttöönottoa. Yhtenäisen formaatin puute myös vaikeuttaa koneluettavuutta, mikä on varsin tärkeä ominaisuus datan hakemisen kannalta.

Lähteiden pirstaloituneisuus linkittyy myös toiseen ongelmaan, nimittäin avoimen datan laadukkuuden vaihteluun. Useimmissa julkaistuissa dataseteissä päästään datan avoimuuden asteikolla vasta ensimmäiselle tasolle. Syynä on, ettei julkaisijoille useinkaan ole kannustinta tuottaa laadukkaampaa dataa, vaan tyydytään minimisuoritukseen datan avaamisessa. Tarvittaisiin jonkinlainen laadunvarmistusmekanismi ja keino palkita sen mukaan, mitä laadukkaampaa dataa julkaistaan.

Pirstaloituneisuuden ja tietovarastojen heterogeenisyyden vuoksi olisi tärkeää, että data olisi saatavilla yhtenäisessä formaatissa jonkinlaisen portaalin tai datakatalogin kautta (Poikola *et al.*, 2010). Datakatalogeja onkin Suomessa jo useita, tärkeää olisi vain taata näiden katalogien yhteentoimivuus ja linkittäminen toisiinsa. Tämä myöskin tehostaisi datan saatavuutta ja pysyvyyttä.

Datan saatavuuden ja haettavuuden kannalta myös metadatan merkitys korostuu (Welle Donker & van Loenen, 2017). Varsinaiseen dataan liittyvä metadata esimerkiksi datan julkaisijasta, julkaisuajankohdasta ja kategoriasta auttavat tiedon paikantamisessa. Laine, Lee ja Nieminen (2015) selvittivät terveydenhuollon metadatan kirjauksiin kohdistuneessa tutkimusartikkelissaan, että metadatan lähteet olivat usein epäselviä. Lisäksi datan käyttöoikeuksien kannalta on olennaista, että käyttöehdot ovat helposti saatavilla ja jäljitettävissä.

Datan yhteydessä olisi siis hyvä olla lisenssitiedot. Pelko käyttöoikeuksien rikkomisesta saattaa nimittäin vähentää datan hyödyntämistä.

Metadataan myös liittyy olennaisesti tieto datan alkuperästä. Avoimen datan tapauksessa tästä muodostuu hyvin tärkeä kysymys, sillä datan hyödyntäjille voi olla tärkeää olla selvillä siitä, mikä taho verkossa saatavilla olevan datan on alun perin tuottanut ja mistä data on peräisin. Linkitettyyn avoimen dataan kohdistuneessa tutkimuksessa selvisi, että vain noin 37 % linkitetystä avoimesta datasta sisältää tiedon alkuperästä metatiedoissaan (Schmachtenberg *et al.*, 2014). Welle Donker ja van Loenen (2017) päätyivät myös heikkoihin tuloksiin alkuperätietojen saatavuuden suhteen tutkimuksessaan. Alkuperän säilöminen metatiedoissa vähentää datan väärentämisen riskiä, jolloin voidaan olla varmempia datan validiteetista.

Ongelmallista on myös URI-pohjainen linkitys datasettien välillä (Rajabi, Sánchez-Alonso & Sicilia, 2014). Jos jostakin syystä URI-linkitys ei enää johdakaan haettavaan resurssiin, niin avoimen datan linkitys menetetään. Jo yhden URI:n rikkinäisyys saattaa johtaa useamman eri datasetin keskinäisten linkityksien hajoamiseen, jos LOD-pilvessä on paljon linkityksiä.

Alla olevassa taulukossa on tiivistetysti listattuna lähdekirjallisuuden pohjalta havaittuja avoimen datan ongelmia ja haasteita (Taulukko 3).

Taulukko 3 Avoimen datan haasteita

| **Ongelma tai haaste** | **Lähteet** |
|---|---|
| Rahoitus | Kitchin, 2014 |
| Datalähteiden heterogeenisyys | Janssen *et al.,* 2012 |
| Laadun vaihtelevuus | Welle Donker & van Loenen, 2017 |
| Metadata; erityisesti datan alkuperä | Laine *et al.,* 2015; Schmachtenberg *et al.*, 2014; Welle Donker & van Loenen, 2017 |
| URI-pohjainen linkitys | Rajabi *et al.*, 2014 |
| Ei panostettu koneluettavuuteen | Cowan *et al.*, 2014 |

## 4.2 Hajautettu tietovarastointi

Tässä luvussa puhutaan tarkemmin hajautetun varastoinnin infrastruktuurista. Viime vuosina on yleistynyt, että data tallennetaan pilveen lokaalin kiintolevytilan sijaan.

On syytä selventää ero hajautetun (engl. *distributed*) ja ei-keskitetyn (engl. *decentralized*) tietovarastoinnin välillä. Näistä ensimmäinen viittaa tallennetun datan sijaintiin. Hajautetussa varastoinnissa data voi sijaita fyysisesti eri paikoissa. Ei-keskitetty taas viittaa siihen, kuka hallitsee järjestelmään. Ei-keskitetyissä järjestelmissä ei ole yhtä keskeistä tahoa, vaan hallinta on jaettu useampien noodien kesken.

Vertaisverkkopohjaisessa tietovarastoinnissa täytyy erityisesti ottaa huomioon se, ettei säilöttävä tiedosto voi olla vain yhden noodin vastuulla. Datan on oltava sopivassa määrin siis replikoitu, jotta voidaan luottaa sen saatavuuteen. Yleisesti ottaen tämä ongelma ratkaistaan joko monistamalla sama tiedosto useammalle noodille kokonaisuudessaan (replikoimalla) tai jakamalla tiedosto useammaksi palaseksi (engl. *shard*), jotka jaetaan eri noodeille.

### 4.2.1 CAP-teoreema

Hajautettujen tietovarastojen tarkastelussa on hyödyllistä soveltaa niin sanottua CAP-teoreemaa (engl. sanoista *consistency, availabilty* ja *partition tolerance*), jonka on kehittänyt Eric Brewer. Teoreeman mukaan mikään hajautettu tietovarasto ei voi osituksen eli esimerkiksi verkon katkostilan sattuessa omata kuin toisen kahdesta ominaisuudesta, jotka ovat eheys ja saatavuus (Brewer, 2012). Osituksen sattuessa täytyy valita, kumpi ominaisuus on tärkeämpi käyttötapauksen ja datan pohjalta.

Katkostilanteissa vertaisverkon tulee valita joko eheyden tai saatavuuden väliltä. Eheys viittaa siihen, että kaikilla vertaisverkon noodeilla on samanaikainen tieto verkon tilasta. Jos esimerkiksi verkossa tapahtuu muutos, niin eheyden omaavassa tietovarastossa kaikki noodit saavat tiedon muutoksesta samanaikaisesti. Saatavuus taas viittaa siihen, että jokaisella noodilla on aina kirjoitus- ja/tai lukuoikeus verkkoon. Vertaisverkko on siis aina saatavilla.

Katkoksen tapahtuessa hajautettu tietovarasto, joka on eheyttä tukeva, palauttaa aina oikean, ajankohtaisimman datan, mutta vasta sitten, kun katkostila on päättynyt. Käyttäjä voi siis joutua odottelemaan, tai pyynnölle voi tulla aikakatkaisu, jolloin haettu data ei ole saatavilla. Saatavuuden takaava vertaisverkko taas palauttaa datan oikeaan aikaan, mutta data ei ole välttämättä ajankohtaista. Se, kumpi ominaisuus on tärkeämpi, riippuu pitkälti käsiteltävästä datasta ja sen käyttötarkoituksista.

Tämän tutkielman kannalta CAP-teoreemaa sovelletaan tutkielman ohjelmointiprototyypin suunnittelussa ja toteutuksessa. Tarkoituksena on kuvata, miten voidaan CAP-teoreeman mukaisesti tukea sekä datan saatavuutta että eheyttä.

### 4.2.2 ACID ja BASE

Relaatiotietokantojen kantava periaate on niin sanottu ACID-paradigma (engl. sanoista *atomicity, consistency, isolation, durability*), eli tietokannan tulee aina taata datan atomisuus, eheys, eristyneisyys ja pysyvyys. Datan käsittelyssä pyri-

tään takaamaan se, että data on käyttäjien kesken yhtä ajankohtaista. Tietokantatransaktioissa hyödynnetään muun muassa tietokantalukkoja, jotta käsiteltävä data saadaan käsiteltyä eristettynä sen eheyden takaamiseksi. Data ei siis aina välttämättä ole saatavilla.

Viime vuosina ACID-paradigman rinnalle on kuitenkin kohonnut niin sanottu BASE-paradigma (engl. sanoista *Basically Available, Soft state, Eventually consistent*). BASE tähtää datan saatavuuteen eheyden kustannuksella; eheyden kuitenkin taataan tulevan ajallaan. BASE toimii kantavana periaatteena NoSQL-tietokannoissa, jotka pyrkivät tällä olemaan skaalautuvampia kuin perinteisemmät relaatiotietokannat. BASE mahdollistaa useamman erillisen toimijan datan hyödyntämisen samanaikaisesti ja on täten soveltuvampi hajautettuihin järjestelmiin (Cattell, 2011).

Eheyden takaamiseksi BASE-pohjaisilla järjestelmillä pitää olla mekanismi, jolla esimerkiksi yhtäaikaisesti käsitelty ja siten useammassa eri tilassa oleva data saadaan taas eheäksi ja koko version laajuisesti taas yhdeksi versioksi. Git-versionhallinnassa tämä järjestyy merge-toiminnolla, jossa useammat muutokset sulautetaan yhteen. Jos yhdistäessä vastaan tulee kohtia, joita yhdistäjä ei pysty ratkomaan, esimerkiksi tiedoston samalle riville kohdistuneita muokkauksia, syntyy konflikti. Tässä tapauksessa vaaditaan, että käyttäjä ratkaisee konfliktin manuaalisesti, jotta yhdistäminen voidaan saattaa päätökseen.

Toinen tapa järjestää eri versioiden (replikoiden) yhdistäminen on niin kutsutuilla CRDT-tietorakenteilla (engl. sanoista *Conflict-free replicated data type*). Kyseinen tietorakenne mahdollistaa datan yhdentämisen ilman konflikteja. CRDT-tietorakenteita on kahdenlaisia: operaatio- tai tilapohjaisia. Operaatiopohjaisissa tietorakenteissa replikan muutokset (lisäykset, poistot) välitetään toiselle replikalle, kun taas tilapohjaisissa tietorakenteissa koko replikan tila välitetään toiselle versiolle yhdistäessä. (Shapiro, Preguiça, Baquero & Zawirski, 2011.)

### 4.2.3 Hyödyt ja haitat keskitettyyn varastointiin verraten

Hajautettu tietovarastointi tarjoaa mahdollisuuksia, joita perinteisempi keskitetty varastointi ei pysty tarjoamaan, mutta se aiheuttaa myös haasteita. Iacob ja Moise (2015) esittävät hajautetun ja keskitetyn tietovarastoinnin vertailuja. Vertailu on kuvattu alla olevassa taulukossa (Taulukko 4).

Taulukko 4 Hajautetun varastoinnin hyödyt ja haitat

| **Hyödyt** | **Selitys** |
|---|---|
| Parempi saatavuus | Useita eri lähteitä (noodit), joista dataa voi hakea keskitetyn palvelimen sijaan. |
| Modulaarinen kasvu | Uusien noodien lisääminen (esim. tallennustilan kasvattamiseksi) on hel- |

| | |
|---|---|
| | pompaa, järjestelmän toiminta ei katkea. |
| Alhaisemmat käsittelyajat | Dataa lisättäessä tai haettaessa kutsua ei tarvitse välttämättä lähettää kauas, vaan käsittelevä noodi voi löytyä läheltä (riippuu vertaisverkon koosta) |
| **Haitat** | |
| Enemmän verkkoliikennettä | Noodien tarvitsee vastaanottaa ja välittää eteenpäin lukuisia kutsuja esim. tiedoston haussa ja tallennuksessa. |
| Datan eheys | Datan eheyttä on vaikea taata nopeasti (päivittyy ajallaan). |
| Vaikea implementoida | Vaatii suunnittelua mm. datan replikoinnin ja/tai datan paloittelemisen suhteen. |

### 4.2.4 NoSQL

Perinteisten relationaalisten tietokantojen rinnalle on viime vuosien aikana noussut niin kutsuttuja NoSQL-tietokantoja. NoSQL-kannat painottuvat yleensä BASE-pohjaisuuteen ja ovat siten helpommin skaalautuvia.

## 4.3 Avoimen datan säilöminen hajautetuissa, ei-keskitetyissä tietovarastoissa

Nyt kun sekä avoimen datan että hajautetun tietovarastoinnin perusteet ja haasteet on esitelty, käsitellään tässä luvussa, mitä tulee ottaa huomioon, kun halutaan säilöä avointa dataa hajautetussa tietovarastossa. Tässä luvussa siis muodostetaan havaintojen pohjalta vaatimusmäärittely tutkielman ohjelmointityölle. Tarkastelussa otetaan huomioon luvussa 4.1.3 havaitut avoimen datan ongelmat, joihin hajautettu tietovarastointi ja alkuperätietojen tallentaminen voivat tarjota ratkaisumahdollisuuksia.

Datan löydettävyys tulee ottaa huomioon. Kun avointa linkitettyä dataa jaetaan vertaisverkkopohjaisesti, tulee olla jonkinlainen erillinen indeksointitaso, jolla dataa voidaan hakea (Sicilia *et al.*, 2016). Datasettien linkityksien tulee olla siis sellaisia, että ne mahdollistavat esimerkiksi hakusanapohjaiset assosiaatiot, ja datan julkaisijoiden tulisi käyttää termejä kuvaamaan datasettejä. RDF saattaa olla tähän hyvä standardi. Vaihtoehtoisesti myös datan strukturoiminen tietokantamuotoon helpottaa sen kategorisointia ja siten löydettävyyttä.

Kuten on jo esitetty, varmuus datan alkuperästä ja luotto siihen, ettei dataa ole muokattu jälkikäteen voi olla tietynlaisen avoimen datan kannalta erittäin tärkeää. Jos esimerkiksi jokin päätöksenteko perustuu tieliikenteestä kerättyyn avoimeen dataan, korostuu datan alkuperän tärkeys ja luotto siihen, että

haettu data on ajankohtaisinta. Lohkoketju saattaa tällaisissa tapauksissa ajaa alkuperän varmentamisen asiaa: kun tieto datan lisäysajankohdasta tallennetaan lohkoketjuun, voidaan olla varmoja sen oikeellisuudesta. Ratkaisumallina hyödynnetään avoimen datan alkuperätietojen OPM-mallia alkuperäntietojen säilömiseksi lohkoketjutransaktiossa.

Datan ajankohtaisuuteen liittyy myös vaatimus, että sitä säilövän tietovaraston pitää pystyä tarjoamaan tarvittaessa mahdollisimman tuoretta dataa. Tämä voi olla haasteellista hajautetuissa tietovarastoissa, sillä ne pystyvät useimmiten takaamaan eheyden vasta tietyn ajan kuluttua.

Kun data on lisätty vertaisverkkopohjaisesti, pitää ottaa myös kantaa sen saatavuuden pysymiseen. Noodien itseohjautuvuuden takia ei ole varmaa, että tietystä datasta vastaava noodi pysyy saavutettavissa, jolloin datan saatavuus heikentyy. Datan saatavuuden voi järjestää kahdella pääsääntöisellä tavalla: joko kannustamalla noodeja datan säilömiseen ja jakamiseen, tai käyttämällä replikoimisstrategioita, joilla data jaetaan noodien kesken säilöttäväksi.

Alla olevassa taulukossa on koostettuna vaatimukset ratkaisumalleineen (Taulukko 5).

Taulukko 5 Vaatimusmäärittely avoimen datan hajautetulle varastoinnille

| **Vaatimus** | **Ratkaisumalli** |
|---|---|
| Data löydettävyys | Indeksointitaso; datasettien linkitys |
| Alkuperän tieto | Tiivisteiden säilöminen lohkoketjussa; metadata |
| Datan eheyden tukeminen | Sisältöpohjainen linkitys (IPFS); eri versioiden välinen linkitys |
| Datan pysyminen saatavilla | Kannustetaan noodeja säilömiseen ja/tai replikoimisstrategiat |

Mainittavan arvoista on, että tässä tutkielmassa keskitytään nimenomaan tietovarastointiaspektiin datan varastoinnissa. Toisin sanoen esimerkiksi avoimen datan haasteisiin sen käytettävyydestä tai siihen, miten avoimen datan julkaisijoita saisi kannustettua julkaisemaan ei oteta kantaa, sillä näiden ongelmien katsotaan olevan tutkielman fokuksen ulkopuolella.

# 5 YLEISKATSAUS TEKNOLOGIOIHIN

Tässä luvussa kuvaillaan yhteenvedonomaisesti keskeisimpiä tutkielman aihepiiriin liittyviä teknologisia ratkaisuja. Näiden joukossa ovat muun muassa tutkielman ohjelmointiosuudessa hyödynnettävä Ethereum ja IPFS, sekä IPFS:n päälle rakennettu tietokantaratkaisu OrbitDB. Lyhyesti esitellään myös hajautetun tietovarastoinnin teknologioista Storj ja Swarm, vaikka niitä ei valittukaan tutkielman suunnitteluartefaktin osiksi. Niiden esittely katsotaan kuitenkin tärkeäksi, sillä ne edustavat vaihtoehtoista tapaa järjestää datan hajautettu varastointi.

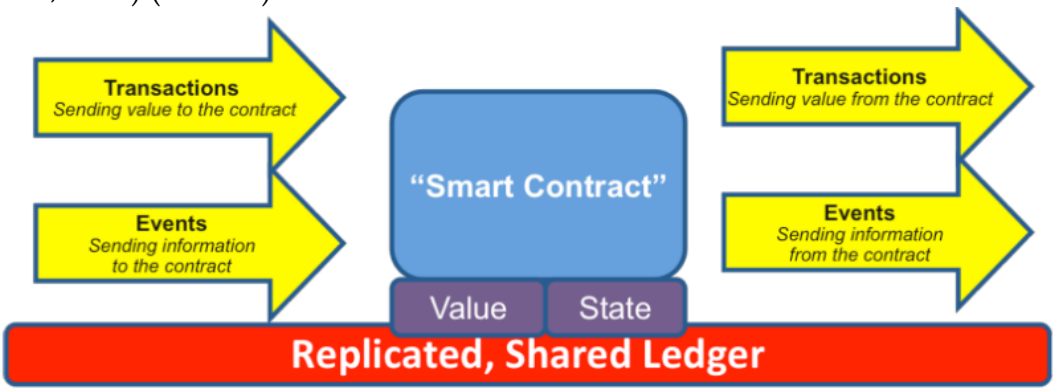
## 5.1 Ethereum

Ethereum on lohkoketjujärjestelmä, joka mahdollistaa lohkoketjuun perustuvien hajautettujen sovellusten kehittämisen (dApps, engl. sanoista *distributed apps*). Ethereum tarjoaa omaa lohkoketjuaan eräänlaisena alustana, jonka pohjalta voi luoda sovelluksia. Nämä sovellukset toimivat Ethereumin virtuaalikoneen päällä, joka kykenee suorittamaan erilaisia skriptejä – tämä virtuaalikone on Turing-täydellinen, eli sillä pystyy toteuttamaan minkä tahansa muun laskennallisen sovelluksen. (Buterin, 2009).

Ethereumin olennaisin piirre on sen tarjoamat älysopimukset. Älysopimukset ovat lohkoketjussa suoritettavia ohjelmia, jotka mahdollistavat sopimusehtojen varmentamisen ja täyttämisen automaattisesti vertaisverkon välityksellä. Älysopimuksilla pystyy siis täysin korvaamaan kolmannen osapuolen luottamusta vaativissa transaktioissa, ja ne mahdollistavat esimerkiksi Bitcoinin maksujärjestelmää monimutkaisempienkin sovellutusten toteuttamisen. Ethereum-älysopimusten ohjelmoinnissa käytetään Solidity-kieltä.

Älysopimusta voisi kuvailla eräänlaiseksi tilakoneeksi, joka ottaa vastaan syötteinä ja lähettää eteenpäin tulosteena transaktioita ja tapahtumia (engl. *events*) (Swanson, 2015). Ethereumin tapahtumat ovat transaktioiden ohella lähetettäviä ilmoituksia, joilla voidaan viestiä vaikkapa client-sovellukselle

transaktion onnistumisesta. Alla kuva älysopimuksen perusperiaatteesta (Swanson, 2015) (Kuva 1).




Kuva 1 Älysopimus

Älysopimusten suorittaminen Ethereum-verkossa vaatii maksuksi Ethereumin omaa kryptovaluuttaa, eli etheriä. Tästä käytetään nimitystä kaasumaksu (engl. *gas)*. Kaasumaksun tarkoituksena on paitsi toimia transaktiopalkkioina louhijoille, mutta myös estää tilanteet, joissa älysopimuksen ohjelmakoodi saattaisi juuttua loputtomaan toistoon ja siten kuluttaa ethereitä teoriassa loputtomasti. Kaasun loppuessa myös älysopimuskoodin suorittaminen lopetetaan.

Ethereumilla on pääverkko (main net) ja muutama eri testi- ja kehityskäyttöön tarkoitettu testiverkko, joissa kehittäjät voivat vapaasti testata älysopimuksiansa käyttäen kaasumaksuihin testiethereitä. Pääverkossa noudatetaan tällä hetkellä PoW-protokollaa, mutta tarkoituksena on jossain vaiheessa siirtyä kenties tehokkaampaan PoS-protokollaan. Nykyisellään Ethereumin haasteena onkin skaalautuvuushaasteet: vuonna 2018 suosiossa ollut CryptoKitties -peli hidasti koko Ethereumin pääverkon toimintaa. Selvää on, että nykyisellään Ethereumin pääverkko ei sovellu laaja-alaiseen toimintaan.

## 5.2 IPFS ja FileCoin

IPFS on tiedostonjakelu- ja varastointiprotokolla, joka perustuu sisältöpohjaiseen datan jakeluun vertaisverkossa. Sisältöpohjainen tarkoittaa sitä, että haluttu Internet-resurssi yksilöidään sen varsinaisen sisällön eikä sen sijainnin pohjalta. Toisin kuin http-protokollassa, jossa jokin resurssi löytyy tietystä URL-osoitteesta, IPFS:ssä resurssi löytyy, kun tietää sen yksilöllisen tunnuksen. Jokainen IPFS:ään lisätty tiedosto nimittäin saa yksilöllisen hajautusarvon, joka on muodostettu kyseisen tiedoston sisällöstä. Tällä vältetään URL-pohjaisten – tosin sanoen nykyisen Internetin tavan paikantaa dataa – kuolleiden linkkien ongelman, kun yritetään hakea dataa osoitteesta, joka ei enää ylläpidäkään kysyttyä dataa. IPFS:ssä tiedosto löytyy aina samalla tunnuksella, olettaen, että tiedosto on ylläpidettynä edes yhden noodin toimesta ja että noodi on yhteydessä muuhun verkkoon. (Benet, 2014.)

IPFS:ää voi kuvailla eräänlaisena Gitin ja Bittorrentin yhdistelmänä (Swan, 2015). Kuten Gitissä, myös IPFS:ssä tiedostohistoria muodostaa puumaisen DAG-rakenteen (engl. *Directed asyclic graph*, eli suunnattu asyklinen graafi). Ja kuten Gitissä, myös IPFS:ssä tiedostot muodostavat versiohistorian. Bittorrentista IPFS lainaa taas tiedonsiirtoprotokollansa ja noodien itseohjautuvuuden: jokainen noodi ottaa vastaan vain sen datan, minkä on valinnut säilöttäväksi.

IPFS itsessään ei anna mitään takeita siitä, että jokin tiedosto pysyy saatavilla. Jokainen noodi toimii datan säilöjänä siis eräänlaisena välimuistina, josta automaattisen roskienkeruun myötä kaikki ladatut tiedostot tuhotaan tietyllä aikavälillä. Jokainen noodi on siis itsenäinen toimija, joka voi ladata verkosta vain haluamansa tiedosto – mitään tiedostoja ei siis jaella automaattisesti vertaisverkon noodeille.

Tiedostot voi säilyttää noodissa pysyvästi komennolla *ipfs pin* - tästä edespäin tässä tutkielmassa käytetään termiä kiinnittäminen. Tämä merkitsee ladatun tiedoston, jolloin sitä ei poisteta automaattisessa roskienkeruussa; noodin itsensä lisäämä tiedosto pysyy roskienkeruulta suojassa (eli se tekee automaattisesti kiinnittämisen). Tiedosto on siis olemassa vertaisverkossa niin kauan kuin sen lisännyt noodi on aktiivisena verkossa, tai niin kauan kuin jokin muu noodi lataa kyseisen tiedoston ja kiinnittää sen. (Benet, 2014.)

Jotta siis tallennettu tiedosto olisi saatavilla, pitäisi taata, että sitä ylläpitäisi vähintään yksi noodi. Ratkaisuja tämän järjestämiseksi on kaksi: noodien kannustaminen datan säilömiseen jonkinlaisella kannustinmekanismilla, tai noodien yhteinen sopiminen säilömisen järjestettäväksi jonkinlaisessa klusterissa.

Noodien kannustamiseen oli tämän tutkielman tekemisaikaan vielä kehitteillä Protocol Labsin oma ratkaisu, FileCoin. FileCoin on IPFS:n kehittäjien hajautettu varastointi- ja samalla virtuaalivaluuttapalvelu, joka on rakennettu IPFS:n päälle. FileCoinin ajatuksena on, että halukkaat vertaisverkon noodit toimivat datan säilöjinä niin kutsutuilla varastointimarkkinoilla, sekä myös datan hakijoina niin sanotuilla tiedostonhakemismarkkinoilla (Protocol Labs, 2017). FileCoinin tarkoituksena on siis täydentää IPFS:n tarjoamaa datavarastoinfrastruktuuria kannustamalla noodeja datan säilömiseen ja hakemiseen. Tällä pyritään takaamaan se, että kaikki säilötyt tiedostot olisivat saatavilla.

FileCoin pyrkii takaamaan tiedostojen säilyvyyden niin kutsutulla *Proof-of-replication* -konsensusprotokollalla. Kyseinen protokolla perustuu siihen, että tiedostoa säilyvä noodi pystyy osoittamaan vertaisverkolle, että säilötty tiedosto on edelleen noodilla säilytettävänä.

## 5.3 OrbitDB

OrbitDB on vertaisverkkopohjainen tietokantaratkaisu, joka on rakennettu IPFS-protokollan päälle. Erityisesti hyödynnetään IPFS:n *pubsub*-aliprotokollaa: kyseessä on julkaisija-tilaaja -protokolla, jolla IPFS-noodit voivat julkaista ja tilata aiheita (engl. *topic*) toistensa välillä. OrbitDB hyödyntää tätä protokollaa

kuuluttamalla tietokannan tilaa vertaisverkon noodeille, joilla on sama tietokanta replikoituna. Näin taataan, että tietokanta on ehyt määrittämättömällä aikavälillä, mikä tekee OrbitDB:stä useiden muiden hajautettujen tietokantaratkaisujen tapaan BASE-tyyppisen tietokannan. Toisin sanoen tietokannan tila voi olla eriävä eri noodeilla niin kauan, kunnes *pubsub*-protokollan myötä tilat on saatettu taas yhteneviksi.

OrbitDB:n ytimenä on eräänlainen lokikirja, johon listataan kaikki tietokantatapahtumat (lisäykset, poistot, muokkaukset) alati kasvavana tietorakenteena. Jokainen noodi siis pitää yllä omaa versiotaan tästä lokikirjasta (eli hieman niin kuin lohkoketjuissa). Eriävät versiot lokikirjasta saatetaan yhteen operaatiopohjaista CRDT-tietorakennetta hyödyntämällä. Versioiden yhdistämistä kutsutaan OrbitDB:n termistössä synkkaamiseksi (engl. *syncing*).

OrbitDB tarjoaa useita eri tietokantatyyppejä, jotka ovat lokikirjan päälle rakennettuja abstraktioita. Näistä yksinkertaisin on lokitietokanta, johon voi vain lisätä kirjauksia. Monimutkaisin on dokumenttitietokanta, joka mahdollistaa strukturoidun datan (esimerkiksi avain-arvoparien ja taulukkorakenteiden) tallentamisen, muokkaamisen ja poistamisen.

## 5.4 Swarm

Swarm on Ethereumin tekijöiden tietovarastointiprotokolla. Se on tarkoitettu osaksi Ethereumin käyttöä, toimien tietovarastointitasona hajautetuissa sovelluksissa. Swarm on päälle kytkettävänä beta-versiona mukana Ethereum clientin uusimmissa versioissa. Swarm toimii kiinteästi Ethereum-clientin yhteydessä.

## 5.5 Storj

Storj on hajautettu pilvitallennussovellus, joka tarjoaa käyttäjilleen yksityistä pilvitallennustilaa. Tallennustilan tarjoajina toimivat ehdokkaina toimivat farmarit (engl. *farmers*), jotka saavat Storjin omaa kryptovaluuttaa palkkioksi tarjoamansa kiintolevytilan määrän perusteella.

Storjin tiedostonjakeluprotokollassa tallennettava tiedosto salataan salausmenetelmin ja jaetaan pirstaleiksi, jotka jaetaan eri noodeille tallennettavaksi. Tiedostopirstaleiden jäljittämiseen Storj käyttää Kademlia-nimistä DHT:ta, ja pirstaleiden eheyttää auditoidaan tietyin aikavälein toistettavilla kyselyillä, joihin säilövien noodien on vastattava ja pystyttävä todentamaan, että säilöttävä data on edelleen säilöttynä.

Storjin käyttötarkoitus on ensisijaisesti yksityisen datan säilöminen salattuna pilvessä, jossa keskitetyn palvelimen sijaan säilöjinä toimivat yksittäiset (anonyymit) toimijat, joille maksetaan kryptovaluuttaa säilömisestä. (Wilkinson *et al.*, 2016.)

# 6 TUTKIMUSMENETELMÄ

Tässä luvussa käsitellään tutkimuksellista osuutta, eli suunnittelutieteellistä tutkimusta. Tarkoituksena on tuottaa artefakti, jolla voidaan todentaa kirjallisuuskatsauksen pohjalta avoimen datan varastoinnin kannattavuus hajautetusti vertaisverkossa. Artefakti koostuu kahdesta osasta: itse artefaktin suunnitelmaa, joka esitellään aliluvussa 6.2, sekä artefaktin toteutusta, jota kuvaillaan tarkemmin aliluvussa 6.3 ja arvioidaan seuraavassa tuloksia käsittelevässä pääluvussa. Toteutustyön tuloksena syntyy ohjelmointiprototyyppi, jolla pyritään tarjoamaan ratkaisuja tutkimuskysymyksiin sekä vaatimusmäärittelyyn (ks. Taulukko 5 Vaatimusmäärittely avoimen datan hajautetulle varastoinnille).

Seuraavaksi esitellään tutkielman konstruktiivinen tutkimusmenetelmä. Sen jälkeen esitellään artefaktin suunnitelma. Kun suunnitelma on esitelty, kuvataan tutkimusprosessi ja ohjelmoinnillisia ratkaisuja tarkemmalla tasolla. Ja lopuksi tässä luvussa arvioidaan tutkimuksen reliabiliteettia ja validiteettia.

## 6.1 Tutkimusmetodologia

Tutkimusmetodologiana käytettiin suunnittelutieteellistä tutkimusmenetelmää. Tässä tutkielmassa kyseisen menetelmän määritelmänä käytetään Peffersin, Tuunasen, Rothenbergin ja Chatterjeen (2007) esittämää DSR-viitekehystä. Viitekehys jakaantuu kuuteen vaiheeseen, jotka ovat:

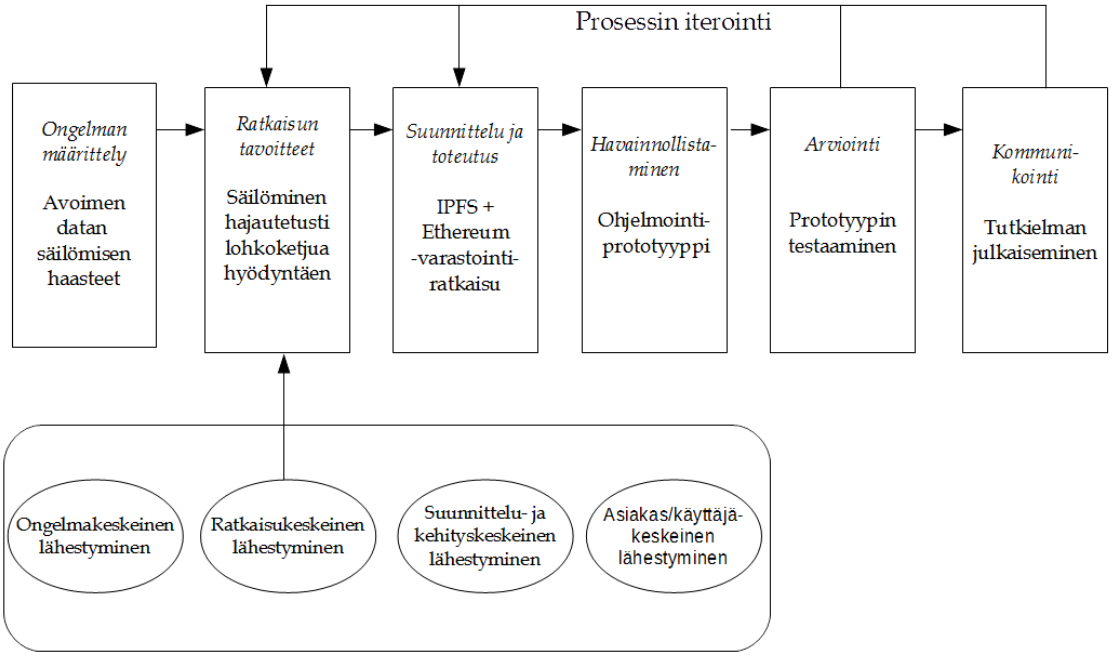
1. Ongelman määrittäminen ja motivointi
2. Ongelman ratkaisun tavoitteiden määrittely
3. Suunnittelu ja kehitys
4. Ratkaisun havainnollistaminen
5. Arviointi
6. Tutkimustulosten kommunikointi

Suunnittelutieteellisellä tutkimuksella voi olla neljä lähestymiskulmaa, joista tutkimus voi lähteä liikkeelle. Tämän tutkielman lähtöpiste oli ratkaisukeskei-

nen, eli halu selvittää, kuinka lohkoketjupohjaisia tietovarastoja voisi soveltaa datan tallentamisessa. Tutkielman kohteeksi määrittyi myöhemmin avoin data lähdekirjallisuuden pohjalta havaittujen ongelmakohtien vuoksi.

Tutkimuksella kerättävä data on luonteeltaan kvalitatiivista, sillä katsottiin, että aihepiirin uutuuden vuoksi kokeileva laadullinen tutkimusote on soveltuvampi. Paljoa aiempaa tutkimusta aihepiiristä ei ollut tämän tutkielman tekemisen aikoihin saatavilla, joten kokeilevampi konstruktiivinen lähestymistapa oli perusteltu.

Edellisissä luvuissa käsiteltiin tutkimusprosessista ongelman määrittely, eli avoimen datan säilömisen haasteet, ja mahdollisen ratkaisun tavoitteet (jotka löytyvät koostetusti taulukosta Taulukko 5). Tässä luvussa kuvaillaan tarkemmin seuraavaa vaihetta, eli artefaktin suunnittelua ja toteutusta. Alla olevassa kuviossa on esiteltynä tutkielman suunnittelutieteellinen prosessirakenne (Kuva 2).




Kuva 2 Tutkielman suunnittelutieteellinen prosessi

## 6.2 Artefaktin suunnittelu ja arkkitehtuuri

Tutkielman artefakti on avoimen datan jakeluun ja tallentamiseen tarkoitettu julkaisualusta. Datan tuottajat voivat lisätä dataa, ja lisäyksen yhteydessä mer-

kitään aikaleima ja muuta metadataa lohkoketjuun. Artefaktin idea perustuu Cowanin *et al*. (2014) esittämään malliin avoimen datan julkaisualustasta.
Artefaktin luomisessa käytetään hyväksi edellisessä luvussa esiteltyä Ethereum-lohkoketjua ja IPFS:ää. Tarkoituksena tässä tutkielmassa oli näitä soveltaen tehdä prototyyppitoteutus avoimen datan varastoinnista. Kyseiset teknologiat valittiin tutkimuksissa käytettäviksi, koska niiden käytöstä löytyy verrattain enemmän dokumentaatiota ja käyttöesimerkkejä kuin muista (esimerkiksi Swarm-protokollasta), ja kyseisten teknologioiden kehitys on aktiivista. IPFS oli vielä tämän tutkielman tekemisvaiheessa kehitysvaiheessa, mutta sen katsottiin kuitenkin olevan tarpeeksi toimiva protokolla artefaktin rakentamiseen.

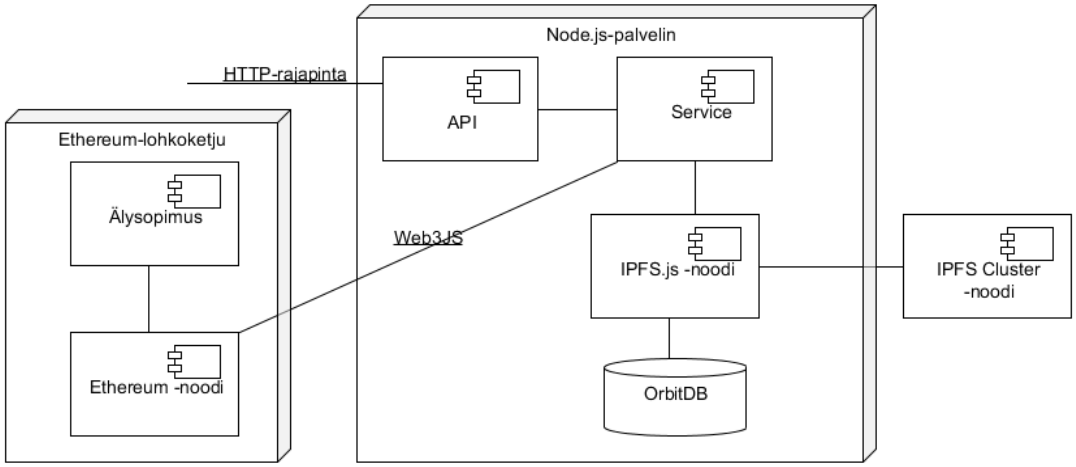
Ohjelmointiprototyyppinä toteutettiin palvelinkomponentti, joka yhdistää IPFS-klusterin ja Ethereumin, sekä tarjoaa API-rajapintaa tiedostojen lähettämistä, hakemista ja varmentamista varten. Tarkoituksena on, että jokainen palveluntarjoaja ylläpitää yhtä tai useampaa palvelinkomponenttia vertaisverkossa, ja jokainen noodi yhdistetään toisiinsa. Palveluntarjoajina voivat toimia esimerkiksi datakatalogien ylläpitäjät, eli julkishallinnolliset toimijat. UI-käyttöliittymää prototyypille ei toteutettu, vaan sen käyttö perustuu API-kutsuihin.

Artefakti muodostuu kolmesta päätason moduulista: Node.js-palvelinkomponentista, IPFS Cluster Service -vertaisverkkoklusterista ja Ethereum-client -noodista. Node.js -palvelimen tarkoituksena on yhdistää kommunikointi IPFS-, Ethereum- ja OrbitDB-komponenttien välillä, sekä kommunikaatio ulkoisiin rajapintoihin (esimerkiksi nettisivuihin) API-rajapinnan avulla. Perusidea oli, että API-rajapinnan kautta tulee kutsu (esimerkiksi tiedoston lisääminen klusteriin), jolloin palvelin välittää tarvittavat kutsut ja hoitaa sisäisen logistiikan.

Ethereum-lohkoketjun tehtäväksi määriteltiin tiedostojen transaktiotapahtumien (lisäykset, poistot, muokkaukset) aikaleimaus. Tätä ideaa ovat käyttäneet monet muutkin sovellutukset (Azaria *et al.*, 2016). Lyhyesti selitettynä siis lisätyn tiedoston IPFS-tiiviste tallennetaan lohkoketjuun muun tarvittavan metadatan ohella. Aikaleimauksella saadaan peruuttamaton todiste siitä, että dataa on lisätty tiettynä ajankohtana tietyn lisääjän toimesta. Tämä takaa datan alkuperän selvittämisen.

IPFS toimi datan tietovarastona ja jakeluverkostona, mutta siinä itsessään ei ole menetelmiä datan indeksoinniksi ja hakemiseksi. Tarvittiin siis tietokanta, jotta datan käsittely olisi strukturoidumpaa. Tätä varten päätettiin käyttää OrbitDB-tietokantaa IPFS:n yhteydessä, jotta lisättyä dataa pystyi kategorisoimaan ja hakemaan tietyin ehdoin. OrbitDB valittiin, koska se toimii natiivisti IPFS:n kanssa yhteensovitettuna ja koska se tarjosi useita vaihtoehtoisia tietokantatyyppejä. Monimutkaisempaan tietokantakäyttöön OrbitDB ei ole (ainakaan vielä) soveltuva, mutta sen katsottiin soveltuvan prototyypin käyttöön esimerkkinä.

Alla kuva artefaktin kokonaisarkkitehtuurista (Kuva 3).




Kuva 3 Artefaktin kokonaisarkkitehtuuri

### 6.2.1 Tiedoston tallentaminen ja hakeminen

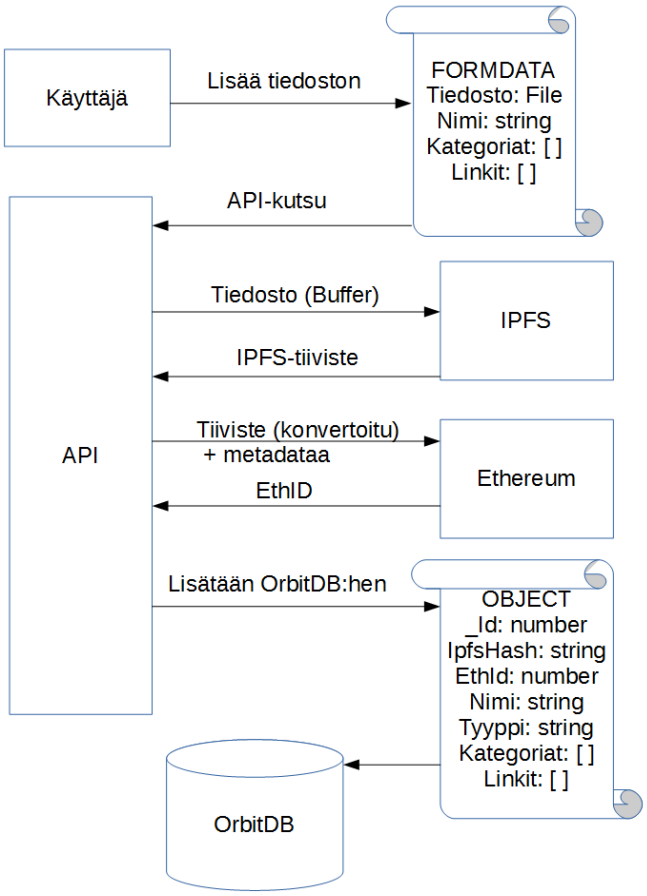
Tiedoston tallentamiseksi tehtiin palvelimeen API-rajapinta, joka ottaa vastaan FormData -formaatissa kutsuja. Kyseinen formaatti koostuu lähetettävistä avain-arvopareista, joissa avaimena on teksti ja arvona mikä tahansa muu ohjelmallinen formaatti, kuten tiedosto. FormData-tietueen muissa kentissä on varsinaiseen lisättävään tiedostoon liittyvää metadataa, kuten kategoriat, linkitykset muihin tiedostoihin IPFS-tiivisteiden pohjalta, sekä tiedoston nimi.

Tiedoston tallentamisessa tiedoston tallennetaan ensin IPFS-klusteriin kiinnitettäväksi, josta saadaan tallennetun tiedoston tiiviste. Tämän jälkeen tiiviste konvertoidaan 32-tavuiseen muotoon, sillä sitä suurempien kenttien tallentaminen älysopimukseen voi olla kallista Ethereumin kaasumaksujen takia. Konvertoitu tiiviste tallennetaan sitten älysopimukseen transaktiona, ja tallennuksen ohella myös muuta metadataa älysopimukseen.

Jokainen älysopimuksella tehty transaktio palauttaa yksilöllisen tunnusluvun. Tunnuslukuna käytetään tässä vain kokonaislukua (kenttä EthId), mutta se voi olla myös jokin yksilöivämpi, kuten vaikka lohkoketjutransaktion tunnus. Tämä tunnusluku tallennetaan yhdessä IPFS-tunnisteen (ja edellä mainitun metadatan) kanssa OrbitDB-tietokantaan.

Tallennuksen yhteydessä käsitellään siis kahdenlaista metadataa: ensin Ethereum-transaktiossa tapahtuman alkuperän varmentamiseen liittyvää metadataa ja sitten OrbitDB-tallennuksessa datan hakemiseen ja kategorisointiin liittyvää metadataa, kuten datasetin nimi, kategoriat ja linkit muihin datasetteihin (linkkinä käytetään IPFS-tiivistettä). Kategoriat ovat tässä suunnitelmassa vain vapaavalintaisia sanoja, mutta laajemmassa toteutuksessa ne voivat olla myös esimerkiksi RDF-linkityksiä.

Alla kuva tiedoston lisäämisestä pääpiirteittäin (Kuva 4).




Kuva 4 Tiedoston lisäämisen kokonaisprosessi

### 6.2.2 Kannustinmekanismi vai replikointistrategia?

Tämän tutkielman vertaisverkkoja käsittelevässä luvussa puhuttiin siitä, miten datan saatavuutta voidaan tukea vertaisverkossa. Havaittiin, että datan saatavuutta voidaan tukea joko kannustamalla noodeja jakamiseen tai käyttämällä datan replikoimisstrategioita. Artefaktin ohjelmoinnissa päädyttiin käyttämään replikoimisstrategiaa IPFS Cluster Servicen muodossa. Tähän päädyttiin, koska kannustinmekanismina toimiva FileCoin oli tämän tutkielman artefaktin toteutusvaiheessa vielä julkaisematta, ja koska toisaalta vertaisverkon kannustimien simuloiminen olisi ollut työlästä toteuttaa validisi lokaalissa testiympäristössä.

### 6.2.3 Simulointi prototyypin toteutuksessa

Artefaktin toteutus (eli ohjelmointiprototyyppi) päätettiin toteuttaa simuloimalla keskeisimpiä vertaisverkkopohjaisia komponentteja lokaalisti. Simulointi

kohdistui datan saatavuuden kannalta tärkeään IPFS Cluster Serviceen sekä alkuperätietojen säilyttämisen kannalta tärkeään Ethereum-lohkoketjuun.

IPFS-klusteri simuloitiin käyttämällä Docker-kontteja, joilla laitettiin käyntiin viisi toisiinsa yhdistettyä IPFS- ja IPFS Cluster -noodia. Tähän klusteriin otettiin http-yhteys palvelinkoodista. Näin saatiin testattua lisättyjen tiedostojen saatavuutta useammalla noodilla.

Ethereum-lohkoketjua simuloitiin Ganache-ohjelmalla. Kyseisellä ohjelmalla pystyy luomaan Ethereum-testilohkoketjun yksityiskäyttöön. Ohjelma generoi automaattisesti tietyn määrän Ethereum-osoitteita. Ohjelmointikomponenttina käytettiin Trufflea, joka tarjosi rajapinnan testilohkoketjun kanssa kommunikointiin. Node-js-palvelimelta yhteyden sai hoidettua web3-kirjastolla, joka on Ethereumin käsittelyyn suunniteltu javascript-kirjasto.

Simuloinnin eduksi katsottiin se, että artefaktin kokonaisjärjestelmää pystyi hallinnoimaan ja tarkkailemaan paremmin kuin oikeassa ympäristössä. Ympäristön muuttumattomuuden takia voitiin olla varmoja, että prototyypin testaaminen tapahtui aina samoilla asetuksilla.

### 6.2.4 Käyttäjät ja käyttötarkoitus

Suunnittelutieteellisessä tutkimuksessa on myös tärkeää selventää, mille käyttäjäkunnalle ratkaisua ollaan kehittämässä. Lähdekirjallisuuden pohjalta tämän tutkielman käyttäjäkunnaksi arvioitiin: avoimen datan tuottajat, ylläpitäjät (esim. julkishallinnolliset toimijat) ja avoimen datan käyttäjät (esim. yksityiset henkilöt tai organisaatiot).

Ohjelmointiprototyypin ajateltua käyttäjäkuntaa ovat erityisesti avoimen datan ylläpitäjät. Ylläpitäjät voivat ylläpitää palvelinpuolella kehitettyä prototyyppiä ja järjestää näin avoimen datan säilömisen ja varmentamisen. Tämän ratkaisun hyöty on, ettei käyttäjien tai tuottajien tarvitse ylläpitää IPFS- ja Ethereum-noodeja.

Tarkoituksena on, että ylläpitäjät tarjoavat rajapintaa, josta käyttäjät ja tuottajat pääsevät lisäämään ja hakemaan dataa. Tällaisena rajapintana voisi toimia esimerkiksi jo olemassa olevat datakatalogit.

### 6.2.5 Rajoitteet

Koska tutkielman fokuksena oli nimenomaan avoimen datan varastointi ja alkuperän varmentaminen, ei aivan kaikkea kokonaisvaltaiseen ratkaisuun kuuluvaa otettu mukaan suunnitteluprosessiin.

Käyttäjien tunnistautuminen on yksi kokonaisuus, jota ei otettu mukaan prototyypin suunnitteluun. Tunnistautuminen ja käytönhallinta lohkoketjupohjaisesti voi olla tärkeä asia avoimen datan jakelussa, mutta sen ei katsottu olevan olennainen tutkimuksen tekemisen pohjalta.

Myös datan salaaminen salausmenetelmin olisi ollut mahdollisesti tärkeä kokonaisuus. Käyttötapauksen mukaan voi olla tärkeää, että IPFS-pohjaisesti tallennettava data on salattua, jotta esimerkiksi pääsy dataan onnistuisi vain

palveluntarjoajien kautta. Tähän liittyen IPFS-klusterista olisi voinut tarvittaessa tehdä yksityisen, mutta toisaalta tämä olisi heikentänyt datan saatavuutta, sillä tällaisessa mallissa järjestelmä olisi riippuvainen pelkästään klusterinoodeista datan saatavuuden kannalta.

Datan linkityksen kannalta myös tallentaminen RDF-formaatissa koneluettavuuden tukemiseksi olisi ollut hyödyllinen asia soveltaa, mutta tämän katsottiin ylittävän tutkielman fokuksen rajat. Datan konvertoiminen RDF-formaattiin ei varsinaisesti kuulu tutkielman piiriin, ja datan linkitykset sai järjestettyä perustasoisesti OrbitDB:n avulla. Samaa mallia pystyy myös pääpiirteittäin käyttämään RDF-tiedostojen kanssa.

## 6.3 Tutkimusprosessi

Tutkimus eteni eri työvaiheiden kautta. Työvaiheet tapahtuivat pääsääntöisesti peräkkäisessä järjestyksessä, mutta myös jonkin verran limittäin. Suunnittelutieteelle ominaisesti myös tämän tutkielman prosessi oli siis iteratiivinen.

Suunnittelutieteelliselle lähestymistavalle on ominaista, että tutkimusprosessi on eräänlainen etsintäprosessi (Hevner, March & Park, 2004). Tämä tarkoittaa sitä, etteivät esimerkiksi toteutuksen tarkemmat tekniset ominaisuudet olleet vielä selviä aloitusvaiheessa, vaan tarkentuivat artefaktin tekemisen myötä. Artefaktin suunnitteluun palattiin, kun sen tekemisessä havaittiin puutteita, ja suunnittelusta taas siirryttiin tekemiseen.

Koko artefaktin suunnittelun ja tekemisen ohella pidettiin yllä tutkimuspäiväkirjaa. Tähän päiväkirjaan merkittiin päivän tekemiset, kohdatut ongelmat ja kehitysideat.

Artefaktin toteutuksen lähdekoodit tallennettiin GitHubiin [1]. Tämä tukee myös tutkimusprosessin viimeistä vaihetta, eli tutkimuksen kommunikointia, koska lähdekoodit ovat vapaasti saatavilla, ja kehitysympäristö on helposti rakennettavissa lyhyen asennusopastuksen avulla.

### 6.3.1 Kirjallisuuskatsaus

Tutkimuksen toteutus lähti liikkeelle kirjallisuuskatsauksesta, jonka tulokset löytyvät tämän tutkielmadokumentin aiemmista luvuista. Tavoitteena oli kartoittaa vertaisverkkojen, lohkoketjujen, hajautetun tietovarastoinnin ja avoimen datan keskeisimpiä piirteitä ja haasteita. Näin saatiin muodostettua tutkielman teoreettinen viitekehys sekä vaatimusmäärittely tutkielman ohjelmointiosuudelle.

Lähteiden haussa käytettiin enimmäkseen Google Scholaria sekä Jyväskylän yliopiston JYKDOKia. Lähdeaineisto muodostui konferenssi- ja eri tieteellisten aikakausilehtien julkaisuista. Toinen merkittävä lähdeaineisto oli merkittä-

[1] https://github.com/smackthat/open-data-repo

vimpien tutkielmassa esiteltyjen teknologioiden julkaisijoiden tekemät *white paper* -raportit, joissa kuvaillaan tarkemmin kyseessä olevaa teknologiaa.

### 6.3.2 Teknologioiden valitseminen ja asennukset

Tutkimusprosessi lähti liikkeelle käytettävien teknologioiden valitsemisesta. Teknologioiden valitsemisessa painotettiin ensisijaisesti niiden soveltuvuutta vaatimusmäärittelyn pohjalta ja toissijaisesti niihin liittyvän dokumentaation, käyttötapausten ja tutkimuksen määrää.

Hajautetun tietovaraston valinnassa oli kolme pääasiallista kandidaattia: Storj, Ethereumin Swarm, sekä IPFS. Nämä kaikki palvelevat toisaalta hyvinkin erilaisia käyttötarkoituksia: esimerkiksi Storj perustuu vahvasti yksityiseen dataan ja lähinnä henkilökohtaisen datan säilömiseen pilvessä, eikä siksi sovellu hyvin avoimen datan jakelualustaksi. Swarm taas oli vielä tämän tutkielman tekemisen aikaan suhteellisen uusi, eikä sen käytöstä ollut yhtä paljoa dokumentaatiota, saati käyttötapauksia kuin IPFS:n tapauksessa. Swarm lupasi parempaa taetta tiedostojen pysyvyydestä (eli tiedoston lisännyt noodi voi lisätä tiedoston ja poistua tiedoston pysyessä edelleen saatavilla), mutta tätä ominaisuutta ei ollut vielä toteutettu. IPFS:n katsottiin olevan monipuolisempi ja valmiimpi teknologia, joten se päätettiin valita keskeneräisyydestään ja rajoituksistaan huolimatta.

IPFS:n kannata tärkeää oli selvittää, miten IPFS:ään tallennettuja tiedostoja sai replikoitua vertaisverkossa automaattisesti. Jakamis- ja säilömiskannustimien puuttuessa pitää datan säilyminen vertaisverkossa taata silloinkin, kun datan lisännyt noodi poistuu verkosta ja dataa ei vielä ylläpidä mikään muu noodi.

Kannustinjärjestelmä olisi ollut yksi tapa toteuttaa datan saatavuuden vahvistaminen, mutta tähän tarkoitukseen kehitteillä ollut FileCoin ei ollut tämän tutkielman tekemisen aikaan vielä saatavilla. Tämän takia päätettiin keskittyä datan replikoimisstrategioihin kannustimien sijaan. Kannustimiin otetaan kuitenkin vielä kantaa tämän tutkielman seuraavassa luvussa.

Yhtenä vaihtoehtona replikointiin oli IPFS Cluster -ohjelma [2]. Ohjelma perustuu siihen, että erillisistä noodeista muodostetaan klustereita, joissa dataa jaetaan automaattisesti klusterin jäsennoodien välillä. Klusteriin kuuluvien noodien tulee tieten tahtoen kuitenkin liittyä klusteriin: IPFS-protokolla perustuu siihen, että noodit itse hakevat haluamansa datan. Avoimen datan jakamisen kannalta tämä tarkoittaa sitä, että datan säilömisestä vastaavat noodit muodostavat johonkin yhteiseen sopimukseen perustuen liittyen uusia IPFS-klustereita datan jakamista varten. Datan jakajina toimisivat siis tietyt luotettavat noodit. Tätä prosessia voisivat hallinnoida esimerkiksi julkishallinnon toimijat.

Tutkimuksen ohjelmointiosuus toteutettiin vapaan lähdekoodin ohjelmia hyödyntäen. Ohjelmat ovat saatavilla Githubista, ja tämänkin tutkimuksen tu-

[2] ks. https://cluster.ipfs.io/documentation/overview/

loksena syntyneet lähdekoodit tallennettiin Githubiin. Vapaan lähdekoodin ohjelmia katsottiin parhaaksi käyttää, koska esimerkiksi IPFS:n testaamiseen ja hajautettujen sovellusten toteuttamiseen tehtyjä työkaluja löytyy nimenomaan Githubista. Myös Githubista saatu tuki ja muiden ohjelmoijien tekemät bugiraportit osoittautuivat erittäin hyödyllisiksi tutkimuksen tekemisen kannalta.

### 6.3.3 IPFS-klusterin pystyttäminen

Tiedostojen jakamista varten perustettiin lokaali testivertaisverkko IPFS:ää protokollana hyödyntäen. Aluksi selvitettiin, kuinka IPFS-verkkoa saisi simuloitua järkevästi pienessä mittakaavassa. Parhaaksi ratkaisuksi katsottiin yksityisen testiverkon pystyttäminen. Testiverkko toteutettiin perustamalla Docker-säiliöitä sekä IPFS- että IPFS Cluster-noodien pyörittämiseen. Docker valittiin sen takia, koska sitä käyttäen oli helppo käynnistää useita noodeja samanaikaisesti ja tarkkailla niiden tiloja, sekä olla varma, että kehitysympäristö pysyi muuttumattomana ohjelmointiprototyyppiä testatessa.

Testiklusteri muodostettiin julkaisemalla jokainen IPFS-noodi ja siihen liittyvä IPFS Cluster Service -noodi toisiinsa linkitettyinä Docker-säiliöinä. Klusterin sai kätevästi pysytettyä Dockerin *docker compose* -komennolla. Klusteriin yhteyden muodostamisessa Node.js-palvelimelta käytettiin klusterin http-välitinporttia, jonka kautta onnistui komentojen suorittaminen klusterissa. Esimerkiksi lähettäessä tiedosto klusterille täytyy se vain lähettää klusterin välitinportille (oletuksena TCP-portti 9095), joka välittää komennot muille klusterinoodeille.

Jokaisella klusterinoodilla täytyy olla yhtenevä tieto verkon tilasta, jotta hajautettu tallennus toimisi. Tämän tutkielman tekemisen aikana IPFS Cluster Service käytti tähän Raft-nimistä konsensusprotokollaa [3]. Kyseinen protokolla perustuu siihen, että noodien joukosta valitaan yksi johtajanoodi, jonka kautta kaikki muutokset välitetään muille noodeille. Muut noodit ovat niin sanottuja seuraajanoodeja. Jos johtajanoodi putoaa verkosta, valitaan seuraava johtaja seuraajanoodien joukosta. Huomioitava asia on, että Raft vaatii toimiakseen vähintään 50 % noodeista olemaan verkossa. Tähän rajoitteeseen palataan tutkielman ohjelmointiprototyypin testaamisvaiheessa.

### 6.3.4 Ethereum-lohkoketjun hyödyntäminen ja älysopimuksen ohjelmointi

Lohkoketjun kanssa kommunikointia varten tehtiin yksikertainen älysopimus, jonka tehtävänä oli säilöä datan tallentamisesta tuotettu IPFS-tiiviste ja metadata lohkoketjuun transaktiona, sekä tietyn transaktion tietojen hakemisen tarvittaessa. Metadata koostui datan lähettäjän Ethereum-osoitteesta, transaktion aikaleimasta ja tapahtumatyypistä (joko lisäys, muokkaus tai poisto). Metadata-objektin suunnittelussa hyödynnettiin García-Barriocanal ym. (2017) kehittämää rakennetta metadatan säilömiseksi lohkoketjussa. Tallennettavien tietojen poh-

[3] ks. https://raft.github.io/

jana käytettiin OPM-mallia. Alla Solidity-kielinen koodipätkä transaktion tallennusobjektin rakenteesta, eli IPFS-tiiviste ja metadata (Kuva 5).

```
struct Hash {
    address sender;
    bytes32 contentHash;
    uint timestamp;
    EventType eventType;
}
```

Kuva 5 Älysopimukseen tallennettava objekti

Älysopimuksella pystyttiin lisäämään lohkoketjuun transaktio, sekä hakemaan OrbitDB-tietokantaan tallennettavan tunnusluvun perusteella tallennettua objektia. Tällä menetelmällä saadaan varmennettua, että tunnusluvulle löytyy sitä vastaava lohkoketjutransaktio ja siten datan lisäyksen aikaleima.

### 6.3.5 OrbitDB-tietokanta ja metadatan linkitykset

Seuraavaksi artefaktin toteuttamisessa lähdettiin selvittämään ja rakentamaan OrbitDB-tietokantatasoa IPFS-tietovarastotason päälle. OrbitDB:n tarjoamista tietokantatyypeistä valittiin dokumenttivarasto, sillä se tarjosi mahdollisuuden useiden eri dataan liittyvien tietueiden määrittämiseen, sekä datan hakemiseen kenttien perusteella. Tämä mahdollistaa sen, että dataa voidaan tallentaa strukturoidussa muodossa sisältäen viittauksia niin Ethereum-älysopimukseen kuin linkitettyihin IPFS-objekteihin.

OrbitDB-tietokannasta luodaan aina käyttäessä oma instanssinsa, joka sidotaan IPFS-noodiin. Kun API:n kautta haetaan linkityksiä ja strukturoitua dataa, kohdistetaan kutsu OrbitDB:lle. Kun taas halutaan suoraan tallennettua tiedostodataa, API-rajapinta kohdistaa kutsun IPFS:lle, josta data haetaan.

## 6.4 Tutkielman reliabiliteetti ja validiteetti

Tutkimuksen laatua on tärkeä mitata tutkimalla sen reliabiliteettia ja validiteettia (Golafshani, 2003). Reliabiliteetti tarkoittaa menetelmän ja tulosten luotettavuutta, validiteetti taas niiden oikeellisuutta. Seuraavaksi tarkastellaan näitä aspekteja tutkielmaan liittyen.

### 6.4.1 Reliabiliteetti

Reliabiliteettia eli luotettavuutta tukee se, että tutkimuksen tulokset ovat toistettavissa (Golafshani, 2003). Testaamisympäristö rakennettiin siten, että se on helposti uudelleenrakennettavissa samoin asetuksin ja muuttumattomassa ympäristössä. Tässä erityisesti IPFS-klusterin ja Ethereum-lohkoketjun simulointi olivat tärkeitä ympäristön muuttumattomuuden takaajia. Koska mikään ulkoinen asia ei vaikuttanut näiden toimivuuteen, oli varmaa, että prototyypin testaaminen tapahtui hallituissa olosuhteissa ja että saatavat tulokset ovat toistuvia. Toistuvuutta todennettiin lisäksi suorittamalla samoja testejä useasti prototyypin testaamisessa. Näin saatiin vähennettyä satunnaisten tulosten määrää.

### 6.4.2 Validiteetti

Tutkimuksen validiteetti määrittyy siitä, miten hyvin valitut menetelmät sopivat tutkimusaiheeseen ja kuinka oikeellisia tuloksia ne tuottavat (Golashani, 2003). Tutkielman suunnittelutieteellisen tutkimusmenetelmän ja sen tuloksena syntyvän prototyypin katsotaan olevan validi tapa tarjota ratkaisua tutkimuskysymyksiin, sillä prototyypin suunnittelun lähtökohtana käytettiin lähdekirjallisuuden pohjalta koostettua teoreettista viitekehystä. Esimerkiksi erityisesti CAP-teoreema osoittautui tutkimuskysymysten, prototyypin suunnittelun sekä prototyypin arvioinnin kannalta tärkeäksi viitekehykseksi. Laadullisen prototyyppitoteutuksen katsottiin myös olevan soveltuva tutkielman teknisen luonteen ja teknologioiden uutuuden suhteen.

# 7 TULOKSET

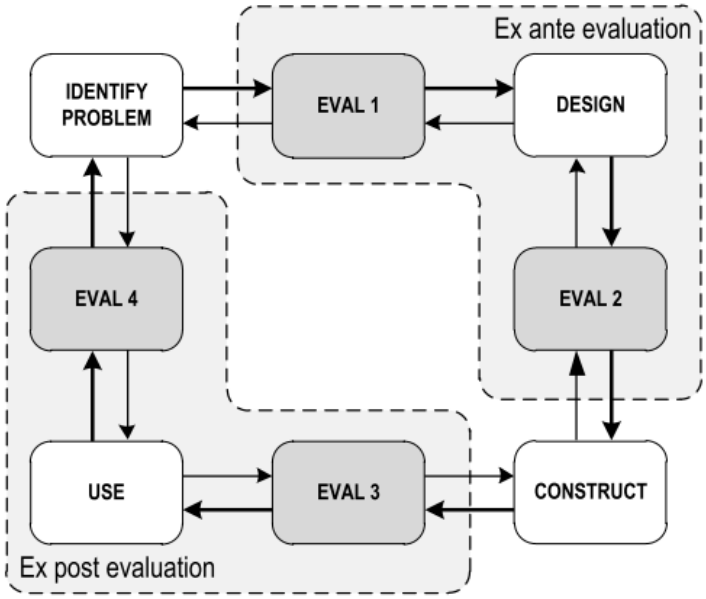
Tässä luvussa esitellään tutkimusmenetelmällä kerätyt tulokset ja analysoidaan niitä tutkielmassa muodostetun teoreettisen viitekehyksen valossa. Erityisesti käytetään luvussa 4 muodostettua avoimen datan hajautetun tietovarastoinnin vaatimustaulukkoa vastaamaan tutkimuskysymyksiin. Aluksi suoritetaan artefaktin arviointi kuvailemalla ensin, miten hyvin artefaktin suunnitelma onnistui vastaamaan vaatimusmäärittelyyn (ks. Taulukko 5 Vaatimusmäärittely avoimen datan hajautetulle varastoinnille). Sitten kuvaillaan ohjelmointiprototyypin toimivuutta sillä suoritettujen testien kautta. Lopuksi johdetaan tutkielman tutkimuskysymysten vastaukset tulosten pohjalta.

## 7.1 Artefaktin evaluaatio

Suunnittelutieteellisen tutkielman arviointia voi lähestyä kolmen eri vastakkainasettelun kautta (Pries-Heje, Baskerville & Venable, 2008). Ensimmäinen vastakkainasettelu on artefaktin suunnitelman ja toteutuksen arviointi. Nämä ovat kaksi erillistä kokonaisuutta, joita on hyvä arvioida omin kriteerein. Toinen asettelu on etu- tai jälkikäteen arviointi. Etukäteen arviointi tarkoittaa, että arvioidaan artefaktin suunnitelmaa ennen toteutusta, kun taas jälkikäteen arvioinnissa arvioidaan artefaktin toteutusta. Kolmannessa asettelussa taas otetaan kantaa, arvioidaanko artefaktia reaalimaailman vai keinotekoisessa arviointiympäristössä. Reaalimaailman arviointiympäristö voi olla esimerkiksi artefaktin testikäyttö käyttäjäkunnan välillä, kun taas keinotekoinen ympäristö voi olla esimerkiksi ohjelmointiprototyypin testaaminen kehitysympäristössä.

Sonnenberg ja Brocke (2012) esittävät, että suunnittelutieteellistä arviointia voidaan suorittaa neljän eri vaiheen yhteydessä, jotka ovat: teoreettisen viitekehyksen muodostaminen, artefaktin suunnitelman tekeminen, artefaktin toteutus ja toteutuksen käyttäminen. Kaksi ensimmäistä sijoittuvat edellä mainitun arviointiviitekehyksen vastakkainasetteluista etukäteen arviointiin, kaksi viimeistä taas suoritetaan jälkikäteen, kun artefakti on jo toteutettu ja kun sen

käyttöä on kokeiltu. Alla kuva artefaktin arvioinnin viitekehyksestä (Sonnenberg & Brocke, 2012) (Kuva 6).




Kuva 6 Artefaktin arvioinnin viitekehys

Tämän tutkielman kannalta evaluaatio kohdistetaan jälkikäteisesti artefaktin toteutukseen sekä sen käyttöön. Arviointiympäristö oli keinotekoinen, sillä arviointi toteutettiin lokaalissa ympäristössä tutkimuskoneella.

Artefaktin arvioinnissa käytetään hyväksi tutkielman kirjallisuuskatsauksen pohjalta koostettua vaatimusmäärittelytaulukkoa. Seuraavaksi kuvataan tiivisestetysti, miten vaatimuksiin päätettiin kehitellä ratkaisuja:

- **Datan löydettävyys** Ratkaisumenetelmänä tähän vaatimukseen käytettiin sekä OrbitDB-tietokantaa metadatan strukturoinnin varmistamiseksi että linkityksiä muihin datasetteihin datasetin metadatassa. Havaittiin, että data on näin järjestettynä löydettävissä, olettaen että linkityksiä tehdään ja että datalle lisätään sitä kuvaavia kategorioita tarpeeksi. Se, miten paljon linkityksiä liittää dataan, riippuu käyttäjästä.
- **Alkuperän tieto** Varastoitavan datan alkuperätietoja hallittiin tallentamalla niistä lokimerkintä Ethereum-testilohkoketjuun älysopimusta käyttäen. Tämä mahdollisti muuttumattoman versiointihistorian, joka on erillään IPFS-pohjaisen hajautetun tietovaraston noodeista – jos noodit jostain syystä kaatuvat tai dataa katoaa, niin kaikesta jää kuitenkin merkintä Ethereumiin. Alkuperätietojen tunnisteena käytettävä IPFS-tiiviste myös takaa sen, että datan alkuperä on jäljitettävissä niin kauan kuin tiedetään IPFS-tiiviste.
- **Datan eheyden tukeminen** Datan eheyden tukemiseksi hyödynnettiin ensisijaisesti IPFS:n avulla. IPFS:n rakenne tapa tallentaa dataa tiivistepohjaisesti takaa sen, että niin kauan kuin tiiviste on tie-

dossa, voi luottaa siihen, että sitä hakemalla palautuu aina sama data (olettaen tietysti, että data on saatavilla). Tämä kuitenkin tuo haasteeksi sen, ettei voida aina luottaa, että samalla IPFS-tiivisteellä saadaan ajankohtaisin versio jostakin tiedostosta. Tarvitaan siis jonkinlainen versiolinkitysmekanismi. Ratkaisuna käytettiin OrbitDB-tietokantaa, joka mahdollisti IPFS-versioiden linkityksen ja haettavuuden, sekä eriävien versioiden yhdistymisen CRDT-tietorakenteen avulla.

- **Datan pysyminen saatavilla** Saatavuuden takaamiseksi ratkaisuvaihtoehdot olivat noodien kannustaminen datan säilyttämiseen, datan jakaminen noodien välillä klusterissa, tai näiden kahden yhdistelmä. Artefaktissa käytettiin klusteripohjaista ratkaisua simuloimalla viiden IPFS- ja IPFS Cluster Service -noodin toimintaa Docker-kontteina. Datatallennusstrategiana käytettiin vain replikointia, sillä IPFS Cluster Service ei vielä tämän tutkielman tekemisaikana tukenut tiedostojen paloittelua.

Alla olevassa taulukossa on vielä yhteenvedonomaisesti kuvattu, miten vaatimukset saatiin täytettyä (Taulukko 6).

Taulukko 6 Artefaktin vaatimusten täyttyminen

| **Vaatimus** | **Vaatimuksen täyttäminen** |
|---|---|
| Datan löydettävyys | Teknologia mahdollistaa linkityksillä, mutta riippuu käyttäjistä; osittain täytetty |
| Alkuperän tieto | Jokaisesta tiedostolisäyksestä automaattisesti kirjaus lohkoketjuun; täytetty |
| Datan eheyden tukeminen | BASE-pohjainen tietovarasto; eheys taataan ajallaan, mikäli noodit pysyvät yhteydessä toisiinsa; osittain täytetty |
| Datan pysyminen saatavilla | Klusteripohjainen tallentaminen, data pysyy olemassa niin kauan, kuin yksikin noodi sitä suostuu säilömään ja niin kauan kuin klusterissa on tarpeeksi noodeja; pääosin täytetty |

## 7.2 Prototyypin testaaminen

Kun suunnittelutieteellisen tutkimuksen tuloksena syntyy jonkinlainen prototyyppi, on sen testaaminen tärkeää oikeellisuuden tukemiseksi (Sonnenberg & Brocke, 2012). Näin saadaan havainnollistettua kuinka hyvin artefakti vastaa

sille asetettuihin vaatimuksiin ja toisekseen minkälaisia muita havaintoja prototyypin käytöstä saattaa ilmentyä.

Prototyypin käyttöä testattiin lokaalissa kehitysympäristössä. Testaamisessa läpikäytiin kaikki toteutetut ominaisuudet, eli tiedoston lisääminen, yhden tai useamman tiedoston hakeminen, sekä alkuperätietojen hakeminen testilohkoketjusta. Testaamisessa käytettiin kannettavaa tietokonetta (2,6 GHz prosessori, 6 GB keskusmuistia).

Testaaminen suoritettiin lähettämällä API-rajapintakutsuja prototyypin Nodejs-palvelimelle. Tässä käytettiin Postman-sovellusta, jolla voi lähettää API-kutsuja ja tarkkailla niiden tuloksia.

Seuraavaksi kuvataan ohjelmointiprototyypin toimintoja sekä kuvataan niiden keskimääräisiä suoritusaikoja ja lohkoketjutransaktioiden tapauksessa transaktiomaksuja.

### 7.2.1 Tiedoston lisäämisen ja hakemisen testaaminen

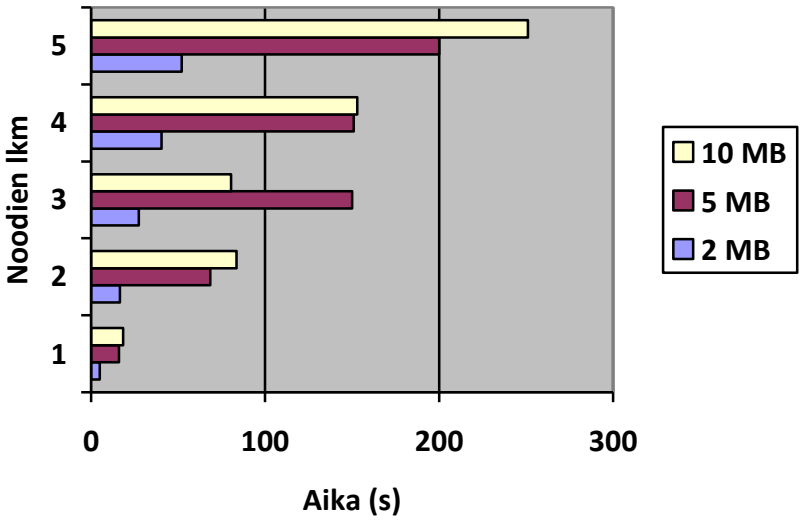
Tiedoston lisäämistä testattiin lisäämällä tiedostoja tallennettavaksi ja mittaamalla tallennukseen käytettyä aikaa. Aika oli siis testaamisen riippuva muuttuja. Saatavuuden kannalta oli tärkeää todentaa, että haettu tiedosto oli aina saatavilla ja kohtuullisella latausajalla. Liian pitkät latausajat nimittäin vaikuttavat negatiivisesti käyttäjäkokemukseen ja heikentävät datan saatavuutta.

Testaamista varten generoitiin eri kokoisia tiedostoja neljässä eri kokoluokassa avoimen datan yleisten tiedostokokojen pohjalta: 2, 5 sekä 10 MB (megatavua). Avoindata.fi-sivustolta löytyi useita datasettejä, joista sai vertailukohtaa yleisimpiin datan kokoluokkiin.

Lisättävän tiedoston kiinnittävien (*ipfs pin*) IPFS-noodien määrän säätely toimi riippumattona muuttujana. Tämä onnistui IPFS Cluster Servicen komennolla määrittämällä tallennettavalle tiedostolle minimi- ja maksimimäärän tallentavia noodeja. Minimissään tallentamiseen käytettiin yhtä noodia, maksimissaan viittä.

Testejä toistettiin useasti samoilla tiedostokokoluokilla ja noodien määrillä. Näin saatiin tarkennettua mittausten tulosta ja vähennettyä sattumatulosten määrää suhteessa testien kokonaismäärään. Tulosten pohjalta laskettiin jokaiselle säilövien noodien lukumäärän ja tiedostokokojen variaatiolle ajallinen keskiarvo sekunneissa.

Testaamisessa havaittiin, että mitä enemmän tiedostoa lisättäessä asetettiin tiedoston kiinnittäviä noodeja, sitä hitaampaa lisääminen oli. IPFS Cluster Servicen sisäinen konsensusmekanismi vei aikansa datan tallennusvastuiden määrittelyssä. Jos taas tiedoston määritti kiinnitettäväksi vain yhdelle noodille, oli lisääminen huomattavasti nopeampaa. Myös tiedostokoko vaikutti tallentamisen nopeuteen: pienemmät (noin 2 megatavun tai sitä pienemmät) tiedostot lisättiin klusteriin sekunneissa, kun taas suurempien tiedostojen (20 megatavua) lisääminen saattoi kestää jopa minuutteja. Alla olevassa kaaviossa on esitettynä keskimääräisiä tallennusaikoja eri tiedostokoilla ja kiinnittävien noodien lukumäärillä (Kuva 7).




Kuva 7 Tiedoston lisäämiseen kulunut keskimääräinen aika

Tiedostojen lisääminen oli siis verrattain hidasta suuremmilla tiedostokoilla ja kiinnittävien noodien määrillä. Tähän tosin voi vaikuttaa myös testikoneen suorituskyky. Testatessa huomattiin, että suurin aika meni tiedoston lisäämisessä IPFS-klusterille tallennettavaksi, sillä klusterilla meni aikansa, että se sai tarvittavat säilömisvastuut määriteltyä noodien kesken. IPFS Cluster Service tarjoaa uudempana ja ilmeisesti tehokkaampana konsensusprotokollana CRDT-tietorakenteisiin perustuvaa protokollaa, jolla tiedostojen lisääminen olisi mahdollisesti nopeampaa, mutta sitä ei ehditty testata tämän tutkielman toteutuksessa.

Toisaalta kiinnittävien klusterinoodien määrällä oli positiivinen vaikutus tiedostojen haettavuuteen. Mitä enemmän noodeja säilöi samaa tiedostoa, sitä nopeammin tiedosto oli haettavissa. Tämä toisaalta riippui paljolti siitä, mihin noodiin tallennettava data satuttiin IPFS Cluster Servicen sisäisen algoritmin mukaisesti tallentamaan – jos tallentavana noodina oli lokaali noodi, oli tiedoston haku nopeaa. Jos sen sijaan haettava noodi oli jokin muu kuin lokaali, kesti haku pitempään. Tässä maantieteelliselläkin etäisyydellä voi olla väliä (Caron *et al.*, 2014), mutta testausympäristön lokaaliuuden takia tätä ei päässyt todentamaan.

Yksi vaatimus datan säilömiselle oli, että data pysyy aina saatavilla, vaikka dataa ainoastaan säilövä noodi poistuisikin (esim. verkkokatkoksen takia) pois vertaisverkosta. IPFS Cluster Service järjestää automaattisesti tallennetun datan uudelleensijoittamisen. Tätä testattiin poistamalla klusterista yksittäisiä noodeja komennolla *ipfs-cluster-ctrl rm* [noodin tunnus]. Havaittiin, että noodin poistuessa uudelleensijoittamistoiminto toimi kuten pitikin, eli klusterin sisäinen algoritmi etsi säilötylle datalle yhden tai useamman uuden säilöjänoodin klusterista.

Jos sen sijaan poisti klusterinoodeista puolet tai yli (eli kolme testauksen viidestä klusterinoodista), niin klusteri joutui tilaan, jossa se ei kyennyt valitsemaan johtajanoodia kahden jäljellä olevan kandidaatin välillä. Tämä esti konsenuksen muodostumisen ja siten datan säilömisen toteutumisen. IPFS Cluster Serviceä käyttäessä on siis kiinnitettävä huomiota, että säilöviä noodeja on tarpeeksi (yli kaksi) ja että niitä on verkossa vähintään 50 %. Aiemmin mainittu CRDT-tietorakenteisiin perustuva konsensus ei vaadi tätä, joten se voi olla perustellumpi vaihtoehto, jos vertaisverkon noodien saapuminen ja verkosta poistuminen on yleistä.

### 7.2.2 Tiedoston alkuperätietojen tallentamisen ja hakemisen testaaminen

Aina, kun tiedosto lisättiin klusteriin, tallennettiin siitä transaktion myötä kirjaus Ganache-testilohkoketjuun. Lohkoketjuun tallennettava data koostui lisätyn tiedoston IPFS-tiivisteestä ja aiemmin tässä tutkielmassa kuvatusta metadatasta.

Keskimääräinen kaasumaksu oi 97 000 weitä (wei = tuhannesosa etheristä). Transaktion hinta saatiin laskettua kaavalla käytetty kaasun määrä * käytettävä kaasuraja (engl. *gas limit*), missä kaasuraja tarkoittaa maksimimäärää kaasua, jonka käyttäjä suostuu käyttämään transaktiossa. Suurempi kaasuraja tarkoittaa nopeampaa käsittelyaikaa ja lohkoon lisäämistä. Tämän tutkielman tekemisen aikaan keskimääräinen kaasuraja oli Ethereumin yleisen suositellun määrän mukaisesti 20 gweitä (gwei = miljoonakymmenesosa yhdestä etheristä). Tällä kaavalla laskettuna siis yhden transaktion keskimääräinen hinta euroissa oli noin 0,39 € (8/2019). Tämän kustannuksen voidaan ajatella olevan vain pieni osa siitä, mitä kuluisi tietokannan hallinnointikustannuksiin perinteisemmissä ratkaisuissa (García-Barriocanal *et al.*, 2017).

### 7.2.3 Testaamisen rajoitteet

Koska prototyypin testaaminen toteutettiin keinotekoisessa ympäristössä, ei sillä saanut täysin luotettavaa mallia siitä, miten toiminnot toimisivat oikeassa ympäristössä ja esimerkiksi monen (toisistaan etäällä olevan) koneen välillä. Reaaliympäristössä transaktioajat voivat vaihdella. Testaamisen uskottiin kuitenkin antavan riittävän hyvän kuvan ratkaisun toimivuudesta, sillä ainakin erilaiset tiedostokoot ja tallentavien IPFS-noodien määrän säätely antoivat eriäviä tuloksia.

OrbitDB-tietokantainstanssien yhteen toimivuutta useamman noodin välillä ei testattu. Tämä olisi vaatinut kattavamman testivertaisverkon luomista, jossa esimerkiksi jokaisen Docker-säiliössä suoritettavien IPFS- ja IPFS Cluster Service -noodien ohella olisi myös suoritettu OrbitDB-instanssia. Tätä ei toteutettu, sillä tutkielman fokus ei ollut niinkään tietokantaominaisuuksien tutkimisessa – suunnitellulla arkkitehtuurilla pyrittiin mahdollistamaan, että tietokantatason pystyisi tarvittaessa vaihtamaan. Siksi OrbitDB:n ominaisuuksien tarkempi testaaminen ei ollut tutkielman tekemisen kannalta relevanttia.

## 7.3 Tutkimuskysymysten vastaukset

Tutkielmalla oli kolme tutkimuskysymystä, joihin lähdettiin hakemaan vastausta tutkimusartefaktin suunnittelun ja toteuttamisen avulla. Nämä tutkimuskysymykset olivat:

1. Miten avoin data saadaan tallennettua hajautetusti?
2. Kuinka tukea datan eheyttä (*consistency*)?
3. Kuinka tukea datan saatavuutta (*availability*)?

Ensimmäiseen tutkimuskysymykseen saatiin vastaus kehittämällä tutkielman suunnitteluartefakti. Artefakti tarjoaa yhden tavan järjestää avoimen datan hajautettu tallennus IPFS:ää tietovarastona ja Ethereum-lohkoketjua datan alkuperätietojen säilömiseen hyödyntäen. Tämä ratkaisu on vain yksi mahdollinen tapa järjestää hajautettu tallennus, eikä tässä tutkielmassa oteta kantaa, olisiko se paras mahdollinen tapa.

Toiseen tutkimuskysymykseen vastaus saadaan CAP-teoreemaa avuksi käyttäen. Teoreeman mukaisesti verkon osituksen sattuessa hajautetun tietovaraston pitää valita joko saatavuuden tai eheyden perusteella. Koska suunnittelun tuloksena luotiin ennen kaikkea BASE-pohjainen tietovarastoratkaisu sen vertaisverkkopohjaisuuden takia, on ymmärrettävää, että eheys voi olla parhaimmillaan vain ajallaan tulevaa. Niinpä osittuneessa verkossa voi olla samanaikaisesti useampi versio verkon tilasta.

Eheyttä voi kuitenkin tukea erilaisilla strategioilla. Suunnitteluartefaktissa tämä otettiin huomioon ennen kaikkea IPFS:n käyttö tietovarastona, sillä sisältöpohjainen linkitys takaa, että käyttäjä saa aina hakiessa saman tiedoston samalla tiivistearvolla. Tämän lisäksi käytettiin OrbitDB-tietokantaa strukturoidun datan mahdollistamiseksi, sillä IPFS toimii vain raakadatan varastona. OrbitDB:n CRDT-tietorakenteet takaavat, että data saadaan eheäksi ajallaan, kun sitä käsitellään hajautetussa ympäristössä.

Kolmanteen tutkimuskysymykseen saatiin vastaukseksi, että datan hajautettu tallennus vertaisverkkopohjaisesti voi tukea sen saatavuutta. Jos data ei ole saatavilla yhdestä sijainnista (noodista), niin ideaalitapauksessa se on saatavilla jostain toisesta. Vertaisverkoissa ei kuitenkaan ole itsestään selvää, että data pysyy saatavilla noodien itseohjautuvuuden vuoksi. Datan saatavuutta vertaisverkossa voikin tukea kahdella pääasiallisella menetelmällä: datan replikoimisstrategioilla, sekä noodien kannustamisella datan säilöttämiseen. Tässä tutkielmassa käytettiin datan replikoimista saatavuuden takaamiseksi IPFS-noodien muodostamaa klusteria hyödyntäen.

Klusteri muodostettiin IPFS Cluster Service -ohjelmalla, jossa eheys taattiin Raft-protokollalla. Replikoimisstrategiat ovat hieman keskitetympiä ratkaisuja, sillä klusterinoodien täytyy olla keskenään luottamuksellisia toimijoita. Olemassa on myös kannustamiseen perustuvia ratkaisuja, kuten jo aiemmin tässä tutkielmassa esitelty Storj (Wilkinson *et al.*, 2016). Todennäköisintä on, että

vertaisverkkopohjaisessa säilömisessä tarvitaan sekä kannustin- että replikoimisstrategioita aina tapauksen mukaan.

# 8 YHTEENVETO JA POHDINTA

Tässä tutkielmassa tutkittiin avoimen datan varastointia vertaisverkkopohjaisesti lohkoketjua hyödyntäen datan alkuperätietojen tallentamisessa. Avoimen datan varastoinnissa havaittiin lähdekirjallisuuden pohjalta ongelmakohtia, joihin hajautetusta ratkaisusta etsittiin mahdollisia ratkaisuja tyypillisimpiin avoimen datan keskitettyyn varastointiin liittyviin ongelmiin. Ongelmia olivat erityisesti epävarmuus datan saatavuudesta sekä datan alkuperätietojen vähyys ja epäluotettavuus.

Tutkielman kirjallisuuskatsauksessa käytiin läpi vertaisverkkojen ja lohkoketjun perusteet, sekä kuvattiin avoimen datan ja hajautetun tietovarastoinnin kenttää. Katsauksessa perehdyttiin myös CAP-teoreemaan, jota käytettiin tutkielman tutkimuskysymysten muotoilussa lähtökohtana. Kirjallisuuskatsauksesta kerättyjen havaintojen ja yhtäläisyyksien pohjalta hahmoteltiin haasteita, joiden pohjalta laadittiin funktionaalinen vaatimusmäärittely tutkielman tutkimusartefaktin suunnitteluun ja toteutukseen.

Tutkimusartefaktin tuloksina syntyivät ohjelmointiprototyypin suunnitelma, sekä itse ohjelmointiprototyyppi. Prototyypin tarkoituksena on havainnollistaa vertaisverkkopohjaista datan tallentamista. Tietovarastoprotokollaksi valittiin IPFS, ja datan alkuperätietojen tallentamisessa hyödynnettiin Ethereum-lohkoketjua.

Tietovarastointi järjestettiin luomalla usean IPFS-noodin muodostama klusteri IPFS Cluster Service -sovellusta käyttäen. Datan strukturoimisessa ja haettavuuden tukemisessa hyödynnettiin OrbitDB-tietokantaratkaisua. Koko prototyyppi toteutettiin Node.js-palvelinsovelluksena. Prototyypin käyttö perustui palvelimelle lähetettäviin API-kutsuihin, joilla oli mahdollista tiedostojen lisääminen ja hakeminen. Hakemisessa pystyttiin hakemaan yksi tai useampi eri OrbitDB-tietokannan tietue tietyin hakuehdoin, sekä suoraan IPFS:stä tiettyä tiivistettä vastaava tiedostosisältö. Prototyypin lähdekoodit löytyvät Githubista.

Jokaisesta tiedostonlisäyksestä luotiin transaktio Ethereum-testilohkoketjuun älysopimusta hyödyntämällä. Lohkoketjun simuloinnissa käytettiin Ganache-sovellusta. Transaktiossa tallennettiin IPFS-tiiviste ja muuta metadataa OPM-mallin mukaisesti tiedoston ja lisäystapahtuman alkuperätieto-

jen takaamiseksi. Transaktiotietoja sai haettua API-rajapinnan kautta. Oikeassa Ethereum-ympäristössä testausta ei toteutettu.

Tuloksia arvioitiin prototyypin ominaisuuksien vertaamista vaatimusmäärittelyyn ja CAP-teoreemaan peilaten. Tulosten arvioitiin täyttävän vaatimusmäärittelyn vaatimukset pääosin. Datasettien linkittämisen vastuu jää viimekädessä käyttäjien vastuulle: jos linkittämistä ei toteuteta, niin data on heikommin löydettävissä. Alkuperätietojen kirjaamisen ja säilömisen kannalta artefakti suoriutui parhaiten. Lohkoketjupohjainen tallentaminen takaa alkuperätietojen muuttumattomuuden. Ethereum-verkon skaalatuvuudesta ei ole vain vielä tähän mennessä ollut merkittävää tutkimusta, joten on vaikea sanoa, miten hyvin laajamittainen avoimen datan alkuperätietojen tallentaminen onnistuisi nykyisessä Ethereum-verkossa. Prototyyppitasolla kuitenkin teknologia osoittautui toimivaksi.

Prototyypin tehokkuutta todennettiin suorittamalla tiedoston lisäämisen ja hakemisen suoritusaikatestejä. Testejä toteutettiin dataa säilövien noodien määrää ja säilöttävien tiedostojen kokoa säätämällä. Todettiin, että mitä enemmän noodeja määritettiin tiedostoa säilömään ja mitä suurempi oli tiedostokoko, sitä pitempään lisääminen kesti. Toisaalta tällä saatiin parempi varmuus siitä, että lisätty tiedosto on saatavilla, ja hakeminen olikin nopeampaa. Päinvastoin mitä vähemmän oli säilöviä noodeja, sitä lyhyemmän aikaa kesti tiedoston lisääminen, mutta hakeminen kesti pidempään. Käyttäjien on siis parasta pohtia aina datakohtaisesti säilömisstrategia esimerkiksi datan oletetun hakemistiheyden mukaisesti.

Prototyyppitoteutuksessa päädyttiin tallentamaan dataa kokonaisina tiedostoina. Sama tiedosto on siis säilöttynä useammalla noodilla erillisinä kopioina ja kokonaisina kappaleina. Datan pirstaloiminen voi olla tehokkaampi ratkaisu, sillä niin saadaan käytettä vähemmän noodien tallennustilaa tiedostojen säilömiseen.

Huomioitavaa vertaisverkkopohjaisessa datan tallentamisessa on, että data on olemassa niin kauan, kuin mikään vertaisverkon noodi suostuu sitä säilömään. Tämä tarkoittaa ensinnäkin sitä, että tallennettu tiedosto voi kadota, jos mikään noodi ei enää säilö sitä. Tähän voi vaikuttaa monet asiat, muun muassa noodien kaatumiset. Siksi on tärkeää, että vertaisverkossa on joko riittävästi datan replikointia jonkin strategian mukaisesti, tai että itsenäisiä toimijoina käyttäytyviä noodeja kannustetaan säilyttämään säilömiänsä tiedostoja.

Toisaalta toinen noodien itseohjautuvuuteen ja datan säilömiseen liittyvä ominaisuus on, että lisättyä dataa voi olla vaikeaa, ellei peräti mahdotonta enää poistaa lopullisesti vertaisverkosta, kun se on kerran lisätty. Tämä vaatii huolellista suunnittelua vertaisverkkoon laitettavan datan suhteen. Avoimen datan kannalta pitää olla varmaa, että lisenssi sallii vapaan käytön ja että muutoshistoria on aina selvillä.

Lohkoketjuteknologian käyttäminen datan alkuperän varmentamisessa perustui sen pohjimmaiseen tarkoitukseen, eli muuttumattomuuden ja luottamuksen takaamiseen hajautetuissa ympäristöissä. Tutkielman lohkoketjutoteutus suunniteltiin ja toteutettiin vastaavien projektien käyttötapojen pohjalta.

Ethereum tarjoaa mahdollisuutta toteuttaa avoimen datan tietovarastointiverkko, jossa datan alkuperään voi luottaa ilman kolmannen osapuolen hallintaa. Sen sijaan Ethereum-verkon (ja muiden lohkoketjunverkkojen) skaalautuvuus sekä energiankulutus havaittiin avoimiksi ongelmiksi, joihin lohkoketjujen tutkimus ja käyttö voivat ajan myötä tarjota ratkaisua. Yksi tärkeä tutkimusalue lohkoketjuteknologioiden saralla olisikin selvittää ratkaisuja sen skaalautuvuuden ja konsensuksen muodostamisen parantamiseksi.

Avoimen datan määrän ja lähteiden kasvaessa on tärkeää, että data on saatavilla ja että sen alkuperään voi luottaa. Tämä vahvistaa sen käyttöä ja jalostamista informaatioksi, joka voi tuottaa laajalti arvoa sitä tarvitseville. Tässä tutkielmassa on ehdotettu vaihtoehtoinen, vertaisverkkopohjainen varastointi- ja jakelualusta avoimelle datalle, jonka tarkoituksena on lisätä datan saatavuutta ja luottamusta sen alkuperään.

# LÄHTEET

Ali, M., Nelson, J., Shea, R., & Freedman, M. J. (2016). Bootstrapping trust in distributed systems with blockchains. *login: USENIX Mag.*, *41*(3).

Azaria, A., Ekblaw, A., Vieira, T., & Lippman, A. (2016). MedRec: Using Blockchain for Medical Data Access and Permission Management. http://doi.org/10.1109/OBD.2016.11

Benet, J. (2014). IPFS - content addressed, versioned, P2P file system. *arXiv preprint arXiv:1407.3561*.

Brewer, E. (2012). CAP twelve years later: How the" rules" have changed. *Computer*, 45(2), 23-29.

Buragohain, C., Agrawal, D., & Suri, S. (2003). A Game Theoretic Framework for Incentives in P2P Systems. *arXiv preprint cs/0310039*.

Buterin, V. (2009). A Next Generation Smart Contract & Decentralized Application Platform, (January), 1–36.

Buterin, V. (2014). On Stake. Haettu 12.9.2018 osoitteesta: https://blog.ethereum.org/2014/07/05/stake/

Casino, F., Dasaklis, T. K., & Patsakis, C. (2019). A systematic literature review of blockchain-based applications: Current status, classification and open issues. *Telematics and Informatics*, 36, 55-81.

Cattell, R. (2011). Scalable SQL and NoSQL data stores. *Acm Sigmod Record*, 39(4), 12-27.

Cowan, Donald & Alencar, Paulo & McGarry, Fred. (2014). Perspectives on Open Data: Issues and Opportunties. *Proceedings - 2014 IEEE International Conference on Software Science, Technology and Engineering*, SWSTE 2014. 24-33.

Druschel, P. (1999). Peer-to-Peer Systems. http://doi.org/10.1145/1831407.1831427

García-Barriocanal, E., Sánchez-Alonso, S., & Sicilia, M. A. (2017, November). Deploying metadata on blockchain technologies. In *Research Conference on Metadata and Semantics Research* (pp. 38-49). Springer, Cham.

Golafshani, N. (2003). Understanding reliability and validity in qualitative research. *The qualitative report, 8(4),* 597-606.

Haja Networks Oy. (n.d.). The OrbitDB Field Manual, 1-72.

Hausenblas, M., & Karnstedt, M. (2010, April). Understanding Linked Open Data as a Web-Scale Database. In *Proceedings of the 2010 Second International Conference on Advances in Databases, Knowledge, and Data Applications* (pp. 56-61). IEEE Computer Society.

Hevner, A., March, S. & Park J. (2004). Design Science in Information Systems Research. *MIS Quarterly Vol. 28 No. 1/March 2004*

Janssen, M., Charalabidis, Y., & Zuiderwijk, A. (2012). Benefits, adoption barriers and myths of open data and open government. *Information systems management,* 29(4), 258-268.

Kitchin, R. (2014). The data revolution: Big data, open data, data infrastructures and their consequences. Sage.

Laine, S., Lee, C., & Nieminen, M. (2015). Transparent Data Supply for Open Information Production Processes. In *ECIS*.

Moreau, L., Freire, J., Futrelle, J., McGrath, R. E., Myers, J., & Paulson, P. (2008). The open provenance model: An overview. In *International Provenance and Annotation Workshop* (pp. 323-326). Springer, Berlin, Heidelberg.

Open Data Institute. (2017). Recommendations for Open Data Portals: From Setup to Sustainability. Haettu osoitteesta: https://www.europeandataportal.eu/sites/default/files/edp_s3wp4_sustainability_recommendations.pdf

Peffers, K., Rothenberger, M., Tuunanen, T., & Vaezi, R. (2012). Design science research evaluation. In *International Conference on Design Science Research in Information Systems* (pp. 398-410). Springer, Berlin, Heidelberg.

Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2016). A Design Science Research Methodology for Information Systems Research, *1222* (December). http://doi.org/10.2753/MIS0742-1222240302

Poikola, A., Kola, P., & Hintikka, K. A. (2010). Julkinen data. Johdatus tietovarantojen avaamiseen. Helsinki: Edita Prima Oy, 1-96

Pries-Heje, J., Baskerville, R., & Venable, J. (2008). Strategies for Design Science Research Evaluation. In *European Conference on Information Systems (ECIS)*. National University of Ireland.

Protocol Labs. (2017). Filecoin: A Decentralized Storage Network, 1-36. Haettu osoitteesta: https://filecoin.io/filecoin.pdf

Rajabi, E., Sánchez-Alonso, S., & Sicilia, M. A. (2014). Analyzing broken links on the web of data: an experiment with DBpedia. *Journal of the Association for Information Science and Technology*, 65(8), 1721-1727

Rodrigues, R., & Druschel, P. (2010). Peer-to-peer systems. *Communications of the ACM*, *53*(10), 72-82.

Saroiu, S., Gummadi, P. K., & Gribble, S. D. (2001, December). Measurement study of peer-to-peer file sharing systems. *In Multimedia Computing and Networking 2002* (Vol. 4673, pp. 156-171). International Society for Optics and Photonics.

Schmachtenberg, M., Bizer, C., & Paulheim, H. (2014, October). Adoption of the linked data best practices in different topical domains. *International Semantic Web Conference* (pp. 245-260). Springer, Cham.

Shapiro, M., Preguiça, N., Baquero, C., & Zawirski, M. (2011, October). Conflict-free replicated data types. In *Symposium on Self-Stabilizing Systems* (pp. 386-400). Springer, Berlin, Heidelberg.

Sicilia, M. A., Sánchez-Alonso, S., & García-Barriocanal, E. (2016). Sharing linked open data over peer-to-peer distributed file systems: the case of IPFS. *Research Conference on Metadata and Semantics Research* (pp. 3-14). Springer, Cham.

Sonnenberg, Christian & Brocke, Jan vom. (2012). Evaluation Patterns for Design Science Research Artefacts. Practical Aspects of Design Science. 71-83. 10.1007/978-3-642-33681-2_7.

Swan, M. (2015). Blockchain: Blueprint for a new economy. " O'Reilly Media, Inc.".

Swanson, T. (2015). Consensus-as-a-service: a brief report on the emergence of permissioned, distributed ledger systems. Haettu osoitteesta: http://www.ofnumbers.com/wp-content/uploads/2015/04/Permissioned-distributed-ledgers.pdf

Welle Donker, F., & van Loenen, B. (2017). How to assess the success of the open data ecosystem?. *International Journal of Digital Earth*, 10(3), 284-306.

Wilkinson, S., Boshevski, T., Brandoff, J., Prestwich, J., Hall, G., Gerbes, P., … Pollard, C. (2016). Storj A Peer-to-Peer Cloud Storage Network, 1–37.